SAS Publishing

Base SAS® 9 Procedures Guide

Volume 1

The Power to Know™

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2002. *Base SAS® 9 Procedures Guide*. Cary, NC: SAS Institute Inc.

Base SAS® 9 Procedures Guide

ISBN 1-58025-942-1

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, June 2002

Contents

What's New

Overview

Enhancements to Base SAS 9 procedures improve ODS formatting, enable import and export of Microsoft Excel 2002 spreadsheets and Microsoft Access 2002 tables, support long format and informat names, list and compare SAS registries, support parallel sorting operations, enhance statistical processing, and enhance printer definitions.

The new DOCUMENT procedure enables you to customize or modify your output hierarchy and replay your output to different destinations without rerunning the PROC or DATA step. Enhancements to the TEMPLATE procedure enable you to customize or create your own markup language for your output. For complete information about what's new in ODS, see *SAS Output Delivery System User's Guide*.

Details

The CONTENTS Procedure

Output from the CONTENTS procedure and the CONTENTS statement in PROC DATASETS provides a new look and additional information. The new look for the output provides a better format for the Output Delivery System (ODS). PROC CONTENTS output now displays the data representation of a file by reporting the native platform for each file, rather than just telling you whether the data representation is native or foreign. Also, PROC CONTENTS output also now provides the encoding value and whether the data set is part of a generation group.

The COPY Procedure

The follwoing options are new or enhanced in the COPY procedure and the COPY statement in PROC DATASETS:

- The FORCE option enables you to use the MOVE option for a SAS data set that has an audit trail.

- The CLONE option now copies the data representation data set attribute.

The CORR Procedure

- A list of ODS table names is now provided. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets.

The DATASETS Procedure

Directory listings from the DATASETS procedure provide a new look for its output, which improves the format for the Output Delivery System (ODS).

The EXPORT Procedure

The EXPORT procedure now enables you to

- export to Microsoft Excel 2002 spreadsheets and Microsoft Access 2002 tables. The new data sources are available for the Windows operating environment on 32-bit platforms if your site has a license for SAS/ACCESS Interface to PC File Formats.
- specify SAS data set options in the DATA= argument when you are exporting to all data sources except for delimited, comma-separated, and tab-delimited external files. For example, if the data set that you are exporting has an assigned password, use the ALTER=, PW=, READ=, or WRITE= data set option. To export only data that meets a specified condition, use the WHERE= data set option.
- specify the SHEET= option to identify a specific spreadsheet in a workbook. Exporting to multiple sheets is available for Microsoft Excel 97, 2000, and 2002 spreadsheets for the Windows operating environment on 32-bit platforms if your site has a license for SAS/ACCESS Interface to PC File Formats.

The FORMAT Procedure

- The maximum length for character format names is now 31. The maximum length for numeric format names is now 32.
- The maximum length for character informat names is now 30. The maximum length for numeric informat names is now 31.

The FREQ Procedure

- A list of ODS table names is now provided. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets.
- The TABLES statement now has the CONTENTS= option that allows you to specify the text for the HTML contents file links to crosstabulation tables.
- The TABLES statement now has the BDT option to request Tarone's adjustment in the Breslow-Day test for homogeneity of odds ratios when you use the CMH option to compute the Breslow-Day test for stratified 2×2 tables.
- The TABLES statement now has the NOWARN option that suppresses the log warning message that the asymptotic chi-square test may not be valid when more than 20 percent of the table cells have expected frequencies less than five.

- The WEIGHT statement now has the ZEROS option to includes observations with zero weight values. The frequency and crosstabulation tables will display any levels that correspond to observations with zero weights. PROC FREQ includes levels with zero weights in the chi-square goodness-of-fit test for one-way tables, in the binomial computations for one-way tables, and in the computation of kappa statistics for two-way tables.

The IMPORT Procedure

The IMPORT procedure now enables you to

- import Microsoft Excel 2002 spreadsheets and Microsoft Access 2002 tables. The new data sources are available for the Windows operating environment on 32-bit platforms if your site has a license for SAS/ACCESS Interface to PC File Formats.
- specify SAS data set options in the OUT= argument when you are importing from all data sources except for delimited, comma-separated, and tab-delimited external files. For example, in order to assign a password for a resulting SAS data set, use the ALTER=, PW=, READ=, or WRITE= data set option. To import only data that meets a specified condition, use the WHERE= data set option.

The MEANS Procedure

The new THREADS|NOTHREADS option (SAS 9 Early Adopter Feature) enables or prevents the activation of multi-threaded processing.

The PRTDEF Procedure

There are 15 new variables now supported by the PRTDEF procedure to control the default printer settings.

The PRTEXP Procedure

The new PRTEXP procedure enables you to write attributes used by PROC PRTDEF to define a printer to a SAS data set or the SAS log, which enables you to replicate and modify those attributes easily.

The REGISTRY Procedure

Ther REGISTRY procedure has three new options:

- The LISTREG option lists the contents of the registry in the log.
- The COMPAREREG1 and COMPAREREG2 options are used together to compare two registries. The results appear in the log.

The REPORT Procedure

In the REPORT procedure, numeric class variables that do not have a format assigned to them are automatically formatted with the BEST12. format.

The SORT Procedure

The SORT procedure has two new options:

- The new DATECOPY option copies to the output data set the SAS internal date and time when the input data set was created and the date and time when it was last modified prior to the sort.
- The new THREADS|NOTHREADS option enables or prevents the activation of multi-threaded sorting.

The SQL Procedure

The SQL procedure has the following new features:

- The PROC SQL statement now has a THREADS NOTHREADS option. THREADS enables PROC SQL to take advantage of the new parallel processing capabilities in SAS when performing sorting operations.
- There are new DICTIONARY tables, new columns in existing DICTIONARY tables, and SASHELP views of the new tables.
- You can now reference a permanent SAS data set by its physical filename.
- When using the INTO clause to assign values to a range of macro variables, you can now specify leading zeroes in the macro variable names. For example,

  ```
  select * into :x01 -- :x10
  ```

  will create the macro variables x01, x02, x03, and so on.

The SYLK Procedure (Experimental)

The new SYLK procedure enables you to read an external SYLK-formatted spreadsheet into SAS, including data, formulas, and formats. You can also use PROC SYLK as a batch spreadsheet, using programming statements to manipulate data, perform calculations, generate summaries, and format the output.

For more information on PROC SYLK, go to **http://www.sas.com/service/library/onlinedoc**. Select Base SAS from the Product-Specific Documentation list.

The TABULATE Procedure

The TABULATE procedure has the following new features:

- Available statistics include upper and lower confidence limits, skewness, and kurtosis. PROC TABULATE now supports the ALPHA= option, which enables you to specify a confidence level.
- Numeric class variables that do not have a format assigned to them are automatically formatted with the BEST12. format.

The TIMEPLOT Procedre

The TIMEPLOT procedure now supports the SPLIT= option, which enables you to specify a character at which labels will be split into multiple lines.

The UNIVARIATE Procedure

The following are new to the UNIVARIATE procedure:

- A list of ODS table names is now provided. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets.

- The LOWER= and NOUPPER= suboptions in the KERNEL option in the HISTOGRAM statement specify the lower and upper bounds for fitted kernel density curves.
- The FRONTREF option in the HISTOGRAM statement draws reference lines in front of the histogram bars instead of behind them.

PART 1

Concepts

CHAPTER
1

Choosing the Right Procedure

Functional Categories of Base SAS Procedures

Base SAS software provides a variety of procedures that produce reports, compute statistics, and perform utility operations.

Report Writing

These procedures display useful information, such as data listings (detail reports), summary reports, calendars, letters, labels, forms, multipanel reports, and graphical reports:

CALENDAR	MEANS*	SQL*
CHART*	PLOT	SUMMARY*
FORMS	PRINT	TABULATE*
FREQ*	REPORT*	TIMEPLOT

* These procedures produce reports and compute statistics.

Statistics

These procedures compute elementary statistical measures that include descriptive statistics based on moments, quantiles, confidence intervals, frequency counts, cross-tabulations, correlations, and distribution tests. They also rank and standardize data:

CHART	RANK	SUMMARY
CORR	REPORT	TABULATE
FREQ	SQL	UNIVARIATE
MEANS	STANDARD	

Utilities

These procedures perform basic utility operations. They create, edit, sort, and transpose data sets, create and restore transport data sets, create user-defined formats, and provide basic file maintenance such as to copy, append, and compare data sets:

APPEND	EXPLODE	PRTDEF
BMDP**	EXPORT	PRTEXP
CATALOG	FORMAT	REGISTRY
CIMPORT	FSLIST	RELEASE**
COMPARE	IMPORT	SORT
CONTENTS	OPTIONS	SOURCE**
CONVERT**	OPTLOAD	SQL
COPY	OPTSAVE	TAPECOPY**
CPORT	PDS**	TAPELABEL**
CV2VIEW***	PDSCOPY**	TEMPLATE*
DATASETS	PMENU	TRANSPOSE
DOCUMENT*	PRINTTO	TRANTAB

* See *SAS Output Delivery System User's Guide* for a description of these procedures.
** See the SAS documentation for your operating environment for a description of these procedures.
***See *SAS/ACCESS for Relational Databases: Reference* for a description of this procedure.

Report-Writing Procedures

Table 1.1 on page 5 lists report-writing procedures according to the type of report.

Table 1.1 Report-Writing Procedures by Task

To produce...	Use this procedure...	Which...
Detail reports	PRINT	produces data listings quickly; can supply titles, footnotes, and column sums.
	REPORT	offers more control and customization than PROC PRINT; can produce both column and row sums; has DATA step computation abilities.
	SQL	combines Structured Query Language and SAS features such as formats; can manipulate data and create a SAS data set in the same step that creates the report; can produce column and row statistics; does not offer as much control over output as PROC PRINT and PROC REPORT.
Summary reports	MEANS or SUMMARY	computes descriptive statistics for numeric variables; can produce a printed report and create an output data set.
	PRINT	produces only one summary report: can sum the BY variables.
	REPORT	combines features of the PRINT, MEANS, and TABULATE procedures with features of the DATA step in a single report writing tool that can produce a variety of reports; can also create an output data set.
	SQL	computes descriptive statistics for one or more SAS data sets or DBMS tables; can produce a printed report or create a SAS data set.
	TABULATE	produces descriptive statistics in a tabular format; can produce stub-and-banner reports (multidimensional tables with descriptive statistics); can also create an output data set.
Miscellaneous highly formatted reports		
Calendars	CALENDAR	produces schedule and summary calendars; can schedule tasks around nonwork periods and holidays, weekly work schedules, and daily work shifts.
Labels, Forms	FORMS	produces labels, such as mailing and inventory, or other forms that have a repetitive format.
Name/address listings	FORMS	produces multicolumn name and address listings.
Multipanel reports (telephone book listings)	REPORT	produces multipanel reports.
Low-resolution graphical reports*		
	CHART	produces bar charts, histograms, block charts, pie charts, and star charts that display frequencies and other statistics.

To produce...	Use this procedure...	Which...
	PLOT	produces scatter diagrams that plot one variable against another.
	TIMEPLOT	produces plots of one or more variables over time intervals.

* These reports quickly produce a simple graphical picture of the data. To produce high-resolution graphical reports, use SAS/GRAPH software.

Statistical Procedures

Table 1.2 on page 6 lists statistical procedures according to task. Table A1.1 on page 1579 lists the most common statistics and the procedures that compute them.

Table 1.2 Elementary Statistical Procedures by Task

To produce...	Use this procedure...	Which...
Descriptive statistics	CORR	computes simple descriptive statistics.
	MEANS or SUMMARY	computes descriptive statistics; can produce printed output and output data sets. By default, PROC MEANS produces printed output and PROC SUMMARY creates an output data set.
	REPORT	computes most of the same statistics as PROC TABULATE; allows customization of format.
	SQL	computes descriptive statistics for data in one or more DBMS tables; can produce a printed report or create a SAS data set.
	TABULATE	produces tabular reports for descriptive statistics; can create an output data set.
	UNIVARIATE	computes the broadest set of descriptive statistics; can create an output data set.
Frequency and cross-tabulation tables	FREQ	produces one-way to *n*-way tables; reports frequency counts; computes chi-square tests; computes tests and measures of association and agreement for two-way to *n*-way cross-tabulation tables; can compute exact tests and asymptotic tests; can create output data sets.
	TABULATE	produces one-way and two-way cross-tabulation tables; can create an output data set.
	UNIVARIATE	produces one-way frequency tables.
Correlation analysis	CORR	computes Pearson's, Spearman's, and Kendall's correlations and partial correlations; also computes Hoeffding's D and Cronbach's coefficient alpha.
Distribution analysis	UNIVARIATE	computes tests for location and tests for normality.
	FREQ	computes a test for the binomial proportion for one-way tables; computes a goodness-of-fit test for one-way tables; computes a chi-square test of equal distribution for two-way tables.

To produce...	Use this procedure...	Which...
Robust estimation	UNIVARIATE	computes robust estimates of scale, trimmed means, and Winsorized means.
Data transformation		
Computing ranks	RANK	computes ranks for one or more numeric variables across the observations of a SAS data set and creates an output data set; can produce normal scores or other rank scores.
Standardizing data	STANDARD	creates an output data set that contains variables that are standardized to a given mean and standard deviation.
Low-resolution graphics*		
	CHART	produces a graphical report that can show one of the following statistics for the chart variable: frequency counts, percentages, cumulative frequencies, cumulative percentages, totals, or averages.
	UNIVARIATE	produces descriptive plots such as stem and leaf, box plot, and normal probability plot.

* To produce high-resolution graphical reports, use SAS/GRAPH software.

Efficiency Issues

Quantiles

For a large sample size n, the calculation of quantiles, including the median, requires computing time proportional to $n\log(n)$. Therefore, a procedure, such as UNIVARIATE, that automatically calculates quantiles may require more time than other data summarization procedures. Furthermore, because data is held in memory, the procedure also requires more storage space to perform the computations. By default, the report procedures PROC MEANS, PROC SUMMARY, and PROC TABULATE require less memory because they do not automatically compute quantiles. These procedures also provide an option to use a new fixed-memory quantiles estimation method that is usually less memory intense. See “Quantiles” on page 680 for more information.

Computing Statistics for Groups of Observations

To compute statistics for several groups of observations, you can use any of the previous procedures with a BY statement to specify BY-group variables. However, BY-group processing requires that you previously sort or index the data set, which for very large data sets may require substantial computer resources. A more efficient way to compute statistics within groups without sorting is to use a CLASS statement with one of the following procedures: MEANS, SUMMARY, or TABULATE.

Additional Information about the Statistical Procedures

Appendix 1, “SAS Elementary Statistics Procedures,” on page 1577 lists standard keywords, statistical notation, and formulas for the statistics that base SAS procedures compute frequently. The individual statistical procedures discuss the statistical concepts that are useful to interpret the output of a procedure.

Utility Procedures

Table 1.3 on page 8 groups utility procedures according to task.

Table 1.3 Utility Procedures by Task

To perform these utility tasks...	Use this procedure...	Which...
Supply information	COMPARE	compares the contents of two SAS data sets.
	CONTENTS	describes the contents of a SAS data library or specific library members.
	OPTIONS	lists the current values of all SAS system options.
	SQL	supplies information through dictionary tables on an individual SAS data set as well as all SAS files active in the current SAS session. Dictionary tables can also provide information about macros, titles, indexes, external files, or SAS system options.
Manage SAS system options	OPTIONS	lists the current values of all SAS system options.
	OPTLOAD	reads SAS system option settings that are stored in the SAS registry or a SAS data set.
	OPTSAVE	saves SAS system option settings to the SAS registry or a SAS data set.
Affect printing and Output Delivery System output	DOCUMENT**	manipulates procedure output that is stored in ODS documents.
	EXPLODE	produces oversized text on printed output; can produce displays such as posters, flip charts, and header pages.
	FORMAT	creates user-defined formats to display and print data.
	PRINTTO	routes procedure output to a file, a SAS catalog entry, or a printer; can also redirect the SAS log to a file.
	PRTDEF	creates printer definitions.
	PRTEXP	exports printer definition attributes to a SAS data set.
	TEMPLATE**	customizes ODS output.
Create, browse, and edit data	FSLIST	browses external files such as files that contain SAS source lines or SAS procedure output.
	SQL	creates SAS data sets using Structured Query Language and SAS features.
Transform data	FORMAT	creates user-defined informats to read data and user-defined formats to display data.
	SORT	sorts SAS data sets by one or more variables.
	SQL	sorts SAS data sets by one or more variables.
	TRANSPOSE	transforms SAS data sets so that observations become variables and variables become observations.
	TRANTAB	creates, edits, and displays customized translation tables.
Manage SAS files	APPEND	appends one SAS data set to the end of another.

To perform these utility tasks…	**Use this procedure…**	**Which…**
	BMDP*	invokes a BMDP program to analyze data in a SAS data set.
	CATALOG	manages SAS catalog entries.
	CIMPORT	restores a transport sequential file that PROC CPORT creates (usually under another operating environment) to its original form as a SAS catalog, a SAS data set, or a SAS library.
	CONVERT*	converts BMDP system files, OSIRIS system files, and SPSS portable files to SAS data sets.
	COPY	copies a SAS data library or specific members of the library.
	CPORT	converts a SAS catalog, a SAS data set, or a SAS library to a transport sequential file that PROC CIMPORT can restore (usually under another operating environment) to its original form.
	CV2VIEW***	converts SAS/ACCESS view descriptors to PROC SQL views.
	DATASETS	manages SAS files.
	EXPORT	reads data from a SAS data set and writes them to an external data source.
	IMPORT	reads data from an external data source and writes them to a SAS data set.
	PDS*	lists, deletes, and renames the members of a partitioned data set.
	PDSCOPY*	copies partitioned data sets from disk to tape, disk to disk, tape to tape, or tape to disk.
	REGISTRY	imports registry information to the USER portion of the SAS registry.
	RELEASE*	releases unused space at the end of a disk data set under the OS/390 environment.
	SOURCE*	provides an easy way to back up and process source library data sets.
	SQL	concatenates SAS data sets.
	TAPECOPY*	copies an entire tape volume or files from one or more tape volumes to one output tape volume.
	TAPELABEL*	lists the label information of an IBM standard-labeled tape volume under the OS/390 environment.
Control windows	PMENU	creates customized pull-down menus for SAS applications.

* See the SAS documentation for your operating environment for a description of these procedures.
** See *SAS Output Delivery System User's Guide* for a description of these procedures.
***See *SAS/ACCESS for Relational Databases: Reference* for a description of this procedure.

Brief Descriptions of Base SAS Procedures

APPEND procedure
: adds observations from one SAS data set to the end of another SAS data set.

BMDP procedure
: invokes a BMDP program to analyze data in a SAS data set. See the SAS documentation for your operating environment for more information.

CALENDAR procedure
: displays data from a SAS data set in a monthly calendar format. PROC CALENDAR can display holidays in the month, schedule tasks, and process data for multiple calendars with work schedules that vary.

CATALOG procedure
: manages entries in SAS catalogs. PROC CATALOG is an interactive, nonwindowing procedure that enables you to display the contents of a catalog, copy an entire catalog or specific entries in a catalog, and rename, exchange, or delete entries in a catalog.

CHART procedure
: produces vertical and horizontal bar charts, block charts, pie charts, and star charts. These charts provide a quick visual representation of the values of a single variable or several variables. PROC CHART can also display a statistic associated with the values.

CIMPORT procedure
: restores a transport file created by the CPORT procedure to its original form (a SAS data library, catalog, or data set) in the format appropriate to the operating environment. Coupled with the CPORT procedure, PROC CIMPORT enables you to move SAS data libraries, catalogs, and data sets from one operating environment to another.

COMPARE procedure
: compares the contents of two SAS data sets. You can also use PROC COMPARE to compare the values of different variables within a single data set. PROC COMPARE produces a variety of reports on the comparisons that it performs.

CONTENTS procedure
: prints descriptions of the contents of one or more files in a SAS data library.

CONVERT procedure
: converts BMDP system files, OSIRIS system files, and SPSS portable files to SAS data sets. See the SAS documentation for your operating environment for more information.

COPY procedure
: copies an entire SAS data library or specific members of the library. You can limit processing to specific types of library members.

CORR procedure
: computes Pearson product-moment and weighted product-moment correlation coefficients between variables and descriptive statistics for these variables. In addition, PROC CORR can compute three nonparametric measures of association (Spearman's rank-order correlation, Kendall's tau-b, and Hoeffding's measure of dependence, D), partial correlations (Pearson's partial correlation, Spearman's partial rank-order correlation, and Kendall's partial tau-b), and Cronbach's coefficient alpha.

CPORT procedure
: writes SAS data libraries, data sets, and catalogs in a special format called a transport file. Coupled with the CIMPORT procedure, PROC CPORT enables you to move SAS libraries, data sets, and catalogs from one operating environment to another.

CV2VIEW procedure
: converts SAS/ACCESS view descriptors to PROC SQL views. Starting in Version 9, conversion of SAS/ACCESS view descriptors to PROC SQL views is recommended because PROC SQL views are platform independent and enable you to use the LIBNAME statement. See *SAS/ACCESS for Relational Databases: Reference* for details.

DATASETS procedure
: lists, copies, renames, and deletes SAS files and SAS generation groups, manages indexes, and appends SAS data sets in a SAS data library. The procedure provides all the capabilities of the APPEND, CONTENTS, and COPY procedures. You can also modify variables within data sets, manage data set attributes, such as labels and passwords, or create and delete integrity constraints.

DOCUMENT procedure
: manipulates procedure output that is stored in ODS documents. PROC DOCUMENT enables a user to browse and edit output objects and hierarchies, and to replay them to any supported ODS output format. See *SAS Output Delivery System User's Guide* for details.

EXPLODE procedure
: produces oversized printing of text to generate displays such as posters, flip charts, and header pages.

EXPORT procedure
: reads data from a SAS data set and writes it to an external data source.

FORMAT procedure
: creates user-defined informats and formats for character or numeric variables. PROC FORMAT also prints the contents of a format library, creates a control data set to write other informats or formats, and reads a control data set to create informats or formats.

FORMS procedure
: produces labels for envelopes, mailing labels, external tape labels, file cards, and other printer forms that have a regular pattern.

FREQ procedure
: produces one-way to n-way frequency tables and reports frequency counts. PROC FREQ can compute chi-square tests for one-way to n-way tables, tests and measures of association and of agreement for two-way to n-way cross-tabulation tables, risks and risk difference for 2×2 tables, trends tests, and Cochran-Mantel-Haenszel statistics. You can also create output data sets.

FSLIST procedure
: displays the contents of an external file or copies text from an external file to the SAS Text Editor.

IMPORT procedure
: reads data from an external data source and writes them to a SAS data set.

MEANS procedure
: computes descriptive statistics for numeric variables across all observations and within groups of observations. You can also create an output data set that contains

specific statistics and identifies minimum and maximum values for groups of observations.

OPTIONS procedure
lists the current values of all SAS system options.

OPTLOAD procedure
reads SAS system option settings from the SAS registry or a SAS data set, and puts them into effect.

OPTSAVE procedure
saves SAS system option settings to the SAS registry or a SAS data set.

PDS procedure
lists, deletes, and renames the members of a partitioned data set. See the SAS documentation for your operating environment for more information.

PDSCOPY procedure
copies partitioned data sets from disk to tape, disk to disk, tape to tape, or tape to disk. See the SAS documentation for your operating environment for more information.

PLOT procedure
produces scatter plots that graph one variable against another. The coordinates of each point on the plot correspond to the two variables' values in one or more observations of the input data set.

PMENU procedure
defines menus that you can use in DATA step windows, macro windows, and SAS/AF windows, or in any SAS application that enables you to specify customized menus.

PRINT procedure
prints the observations in a SAS data set, using all or some of the variables. PROC PRINT can also print totals and subtotals for numeric variables.

PRINTTO procedure
defines destinations for SAS procedure output and the SAS log.

PRTDEF procedure
creates printer definitions for individual SAS users or all SAS users.

PRTEXP procedure
exports printer definition attributes to a SAS data set so that they can be easily replicated and modified.

RANK procedure
computes ranks for one or more numeric variables across the observations of a SAS data set. The ranks are written to a new SAS data set. Alternatively, PROC RANK produces normal scores or other rank scores.

REGISTRY procedure
imports registry information into the USER portion of the SAS registry.

RELEASE procedure
releases unused space at the end of a disk data set in the OS/390 environment. See the SAS documentation for this operating environment for more information.

REPORT procedure
combines features of the PRINT, MEANS, and TABULATE procedures with features of the DATA step in a single report-writing tool that can produce both detail and summary reports.

SORT procedure
: sorts observations in a SAS data set by one or more variables. PROC SORT stores the resulting sorted observations in a new SAS data set or replaces the original data set.

SOURCE procedure
: provides an easy way to back up and process source library data sets. See the SAS documentation for your operating environment for more information.

SQL procedure
: implements a subset of the Structured Query Language (SQL) for use in SAS. SQL is a standardized, widely used language that retrieves and updates data in SAS data sets, SQL views, and DBMS tables, as well as views based on those tables. PROC SQL can also create tables and views, summaries, statistics, and reports and perform utility functions such as sorting and concatenating.

STANDARD procedure
: standardizes some or all of the variables in a SAS data set to a given mean and standard deviation and produces a new SAS data set that contains the standardized values.

SUMMARY procedure
: computes descriptive statistics for the variables in a SAS data across all observations and within groups of observations and outputs the results to a new SAS data set.

TABULATE procedure
: displays descriptive statistics in tabular form. The value in each table cell is calculated from the variables and statistics that define the pages, rows, and columns of the table. The statistic associated with each cell is calculated on values from all observations in that category. You can write the results to a SAS data set.

TAPECOPY procedure
: copies an entire tape volume or files from one or more tape volumes to one output tape volume. See the SAS documentation for your operating environment for more information.

TAPELABEL procedure
: lists the label information of an IBM standard-labeled tape volume under the OS/390 environment. See the SAS documentation for this operating environment for more information.

TEMPLATE procedure
: customizes ODS output for an entire SAS job or a single ODS output object. See *SAS Output Delivery System User's Guide* for details.

TIMEPLOT procedure
: produces plots of one or more variables over time intervals.

TRANSPOSE procedure
: transposes a data set that changes observations into variables and vice versa.

TRANTAB procedure
: creates, edits, and displays customized translation tables.

UNIVARIATE procedure
: computes descriptive statistics (including quantiles), confidence intervals, and robust estimates for numeric variables. Provides detail on the distribution of numeric variables, which include tests for normality, plots to illustrate the distribution, frequency tables, and tests of location.

CHAPTER

2

Fundamental Concepts for Using Base SAS Procedures

Language Concepts

Temporary and Permanent SAS Data Sets

SAS data sets can have a one-level name or a two-level name. Typically, names of temporary SAS data sets have only one level and are stored in the WORK data library. The WORK data library is defined automatically at the beginning of the SAS session and is automatically deleted at the end of the SAS session. Procedures assume that SAS data sets that are specified with a one-level name are to be read from or written to the WORK data library, unless you specify a USER data library (see "USER Data Library" on page 17). For example, the following PROC PRINT steps are equivalent. The second PROC PRINT step assumes that the DEBATE data set is in the WORK data library:

```
proc print data=work.debate;
run;

proc print data=debate;
run;
```

The SAS system options WORK=, WORKINIT, and WORKTERM affect how you work with temporary and permanent libraries. See *SAS Language Reference: Dictionary* for complete documentation.

Typically, two-level names represent permanent SAS data sets. A two-level name takes the form *libref.SAS-data-set*. The *libref* is a name that is temporarily associated with a *SAS data library*. A SAS data library is an external storage location that stores SAS data sets in your operating environment. A LIBNAME statement associates the libref with the SAS data library. In the following PROC PRINT step, PROCLIB is the libref and EMP is the SAS data set within the library:

```
libname proclib 'SAS-data-library';
proc print data=proclib.emp;
run;
```

USER Data Library

You can use one-level names for permanent SAS data sets by specifying a USER data library. You can assign a USER data library with a LIBNAME statement or with the SAS system option USER=. After you specify a USER data library, the procedure assumes that data sets with one-level names are in the USER data library instead of the WORK data library. For example, the following PROC PRINT step assumes that DEBATE is in the USER data library:

```
options user='SAS-data-library';
proc print data=debate;
run;
```

Note: If you have a USER data library defined, then you can still use the WORK data library by specifying WORK.*SAS-data-set*.

SAS System Options

Some SAS system option settings affect procedure output. The following are the SAS system options that you are most likely to use with SAS procedures:

BYLINE | NOBYLINE
DATE | NODATE
DETAILS | NODETAILS
FMTERR | NOFMTERR
FORMCHAR=
FORMDLIM=
LABEL | NOLABEL
LINESIZE=
NUMBER | NONUMBER
PAGENO=
PAGESIZE=
REPLACE | NOREPLACE
SOURCE | NOSOURCE

For a complete description of SAS system options, see *SAS Language Reference: Dictionary*.

Data Set Options

Most of the procedures that read data sets or create output data sets accept data set options. SAS data set options appear in parentheses after the data set specification. Here is an example:

```
proc print data=stocks(obs=25 pw=green);
```

The individual procedure chapters contain reminders that you can use data set options where it is appropriate.

SAS data set options are

ALTER=	OBS=
BUFNO=	OPTSET=
BUFSIZE=	OUTREP=
CNTLLEV=	POINTOBS=

COMPRESS=	PW=
DLDMGACTION=	PWREQ=
DROP=	READ=
ENCODING=	RENAME=
ENCRYPT=	REPEMPTY=
FILECLOSE=	REPLACE=
FIRSTOBS=	REUSE=
GENMAX=	ROLE=
GENNUM=	SORTEDBY=
IDXNAME=	SORTSEQ=
IDXWHERE=	TOBSNO=
IN=	TYPE=
INDEX=	WHERE=
KEEP=	WHEREUP=
LABEL=	WRITE=

For a complete description of SAS data set options, see *SAS Language Reference: Dictionary*.

Global Statements

You can use these global statements anywhere in SAS programs except after a DATALINES, CARDS, or PARMCARDS statement:

comment	ODS
DM	OPTIONS
ENDSAS	PAGE
FILENAME	RUN
FOOTNOTE	%RUN
%INCLUDE	SASFILE
LIBNAME	SKIP
%LIST	TITLE
LOCK	X

For information about all but the ODS statement, refer to *SAS Language Reference: Dictionary*. For information about the ODS statement, refer to “Output Delivery System” on page 32 and to *SAS Output Delivery System User’s Guide*.

Procedure Concepts

Input Data Sets

Many base procedures require an input SAS data set. You specify the input SAS data set by using the DATA= option in the procedure statement, as in this example:

```
proc print data=emp;
```

If you omit the DATA= option, the procedure uses the value of the SAS system option _LAST_=. The default of _LAST_= is the most recently created SAS data set in the current SAS job or session. _LAST_= is described in detail in *SAS Language Reference: Dictionary*.

RUN-Group Processing

RUN-group processing enables you to submit a PROC step with a RUN statement without ending the procedure. You can continue to use the procedure without issuing another PROC statement. To end the procedure, use a RUN CANCEL or a QUIT statement. Several base SAS procedures support RUN-group processing:

CATALOG

DATASETS

PLOT

PMENU

TRANTAB

See the section on the individual procedure for more information.

Note: PROC SQL executes each query automatically. Neither the RUN nor RUN CANCEL statement has any effect. △

Creating Titles That Contain BY-Group Information

BY-group processing uses a BY statement to process observations that are ordered, grouped, or indexed according to the values of one or more variables. By default, when you use BY-group processing in a procedure step, a BY line identifies each group. This section explains how to create titles that serve as customized BY lines.

Suppressing the Default BY Line

When you insert BY-group processing information into a title, you usually want to eliminate the default BY line. To suppress it, use the SAS system option NOBYLINE.

Note: You must use the NOBYLINE option if you insert BY-group information into titles for the following base SAS procedures:

MEANS

PRINT

STANDARD

SUMMARY

If you use the BY statement with the NOBYLINE option, then these procedures always start a new page for each BY group. This behavior prevents multiple BY groups from appearing on a single page and ensures that the information in the titles matches the report on the pages. △

Inserting BY-Group Information into a Title

The general form for inserting BY-group information into a title is

#BY-specification<.suffix>

BY-specification
is one of the following:

BYVAL*n* | BYVAL(*BY-variable*)
places the value of the specified BY variable in the title. You specify the BY variable with one of the following:

n
is the *n*th BY variable in the BY statement.

BY-variable
is the name of the BY variable whose value you want to insert in the title.

BYVAR*n* | BYVAR(*BY-variable*)
places the label or the name (if no label exists) of the specified BY variable in the title. You designate the BY variable with one of the following:

n
is the *n*th BY variable in the BY statement.

BY-variable
is the name of the BY variable whose name you want to insert in the title.

BYLINE
inserts the complete default BY line into the title.

suffix
supplies text to place immediately after the BY-group information that you insert in the title. No space appears between the BY-group information and the suffix.

Example: Inserting a Value from Each BY Variable into the Title

This example

1 creates a data set, GROC, that contains data for stores from four regions. Each store has four departments. See “GROC” on page 1626 for the DATA step that creates the data set.
2 sorts the data by Region and Department.
3 uses the SAS system option NOBYLINE to suppress the BY line that normally appears in output that is produced with BY-group processing.
4 uses PROC CHART to chart sales by Region and Department. In the first TITLE statement, #BYVAL2 inserts the value of the second BY variable, Department, into the title. In the second TITLE statement, #BYVAL(Region) inserts the value of Region into the title. The first period after Region indicates that a suffix follows. The second period is the suffix.

5 uses the SAS system option BYLINE to return to the creation of the default BY line with BY-group processing.

```
data groc; ❶
   input Region $9. Manager $ Department $ Sales;
   datalines;
Southeast    Hayes       Paper       250
Southeast    Hayes       Produce     100
Southeast    Hayes       Canned      120
Southeast    Hayes       Meat         80
...more lines of data...
Northeast    Fuller      Paper       200
Northeast    Fuller      Produce     300
Northeast    Fuller      Canned      420
Northeast    Fuller      Meat        125
;

proc sort data=groc;  ❷
   by region department;
run;
options nobyline nodate pageno=1
        linesize=64 pagesize=20;  ❸
proc chart data=groc;  ❹
   by region department;
   vbar manager / type=sum sumvar=sales;
   title1 'This chart shows #byval2 sales';
   title2 'in the #byval(region)..';
run;
options byline;   ❺
```

This partial output shows two BY groups with customized BY lines:

```
                This chart shows Canned sales                  1
                     in the Northwest.

      Sales Sum

      400 +       *****                   *****
          |       *****                   *****
      300 +       *****                   *****
          |       *****       *****       *****
      200 +       *****       *****       *****
          |       *****       *****       *****
      100 +       *****       *****       *****
          |       *****       *****       *****
          --------------------------------------------
                 Aikmann     Duncan     Jeffreys

                             Manager
```

```
                This chart shows Meat sales                  2
                     in the Northwest.

         Sales Sum

         75 +       *****       *****
            |       *****       *****
         60 +       *****       *****
            |       *****       *****
         45 +       *****       *****
            |       *****       *****
         30 +       *****       *****       *****
            |       *****       *****       *****
         15 +       *****       *****       *****
            |       *****       *****       *****
            --------------------------------------------
                   Aikmann     Duncan     Jeffreys

                               Manager
```

Example: Inserting the Name of a BY Variable into a Title

This example inserts the name of a BY variable and the value of a BY variable into the title. The program

1 uses the SAS system option NOBYLINE to suppress the BY line that normally appears in output that is produced with BY-group processing.

2 uses PROC CHART to chart sales by Region. In the first TITLE statement, #BYVAR(Region) inserts the name of the variable Region into the title. (If Region had a label, #BYVAR would use the label instead of the name.) The suffix **al** is appended to the label. In the second TITLE statement, #BYVAL1 inserts the value of the first BY variable, Region, into the title.

3 uses the SAS system option BYLINE to return to the creation of the default BY line with BY-group processing.

```
options nobyline nodate pageno=1
        linesize=64 pagesize=20;  ❶
proc chart data=groc;  ❷
   by region;
   vbar manager / type=mean sumvar=sales;
   title1 '#byvar(region).al Analysis';
   title2 'for the #byval1';
run;
options byline;  ❸
```

This partial output shows one BY group with a customized BY line:

```
                    Regional Analysis                         1
                    for the Northwest

       Sales Mean

       300 +                                *****
           |                                *****
       200 +          *****                 *****
           |          *****       *****     *****
       100 +          *****       *****     *****
           |          *****       *****     *****
           --------------------------------------------
                     Aikmann     Duncan    Jeffreys

                                 Manager
```

Example: Inserting the Complete BY Line into a Title

This example inserts the complete BY line into the title. The program

1 uses the SAS system option NOBYLINE to suppress the BY line that normally appears in output that is produced with BY-group processing.

2 uses PROC CHART to chart sales by Region and Department. In the TITLE statement, #BYLINE inserts the complete BY line into the title.

3 uses the SAS system option BYLINE to return to the creation of the default BY line with BY-group processing.

```
options nobyline nodate pageno=1
        linesize=64 pagesize=20;  ❶
proc chart data=groc;  ❷
   by region department;
   vbar manager / type=sum sumvar=sales;
   title 'Information for #byline';
run;
options byline;  ❸
```

This partial output shows two BY groups with customized BY lines:

```
      Information for Region=Northwest Department=Canned      1

       Sales Sum

       400 +          *****                 *****
           |          *****                 *****
       300 +          *****                 *****
           |          *****       *****     *****
       200 +          *****       *****     *****
           |          *****       *****     *****
       100 +          *****       *****     *****
           |          *****       *****     *****
           --------------------------------------------
                     Aikmann     Duncan    Jeffreys

                                 Manager
```

```
        Information for Region=Northwest Department=Meat       2

        Sales Sum

        75 +       *****       *****
           |       *****       *****
        60 +       *****       *****
           |       *****       *****
        45 +       *****       *****
           |       *****       *****
        30 +       *****       *****       *****
           |       *****       *****       *****
        15 +       *****       *****       *****
           |       *****       *****       *****
           --------------------------------------------
                 Aikmann     Duncan     Jeffreys

                             Manager
```

Error Processing of BY-Group Specifications

SAS does not issue error or warning messages for incorrect #BYVAL, #BYVAR, or #BYLINE specifications. Instead, the text of the item simply becomes part of the title.

Shortcuts for Specifying Lists of Variable Names

Several statements in procedures allow multiple variable names. You can use these shortcut notations instead of specifying each variable name:

Notation	Meaning
x1-x*n*	specifies variables X1 through X*n*. The numbers must be consecutive.
x:	specifies all variables that begin with the letter X.
x--a	specifies all variables between X and A, inclusive. This notation uses the position of the variables in the data set.
x-numeric-a	specifies all numeric variables between X and A, inclusive. This notation uses the position of the variables in the data set.
x-character-a	specifies all character variables between X and A, inclusive. This notation uses the position of the variables in the data set.
numeric	specifies all numeric variables.
character	specifies all character variables.
all	specifies all variables.

Note: You cannot use shortcuts to list variable names in the INDEX CREATE statement in PROC DATASETS. △

See *SAS Language Reference: Concepts* for complete documentation.

Formatted Values

Typically, when you print or group variable values, base SAS procedures use the formatted values. This section contains examples of how base procedures use formatted values.

Example: Printing the Formatted Values for a Data Set

The following example prints the formatted values of the data set PROCLIB.PAYROLL. (See "PROCLIB.PAYROLL" on page 1648 for the DATA step that creates this data set.) In PROCLIB.PAYROLL, the variable Jobcode indicates the job and level of the employee. For example, **TA1** indicates that the employee is at the beginning level for a ticket agent.

```
libname proclib 'SAS-data-library';


options nodate pageno=1
        linesize=64 pagesize=40;
proc print data=proclib.payroll(obs=10)
           noobs;
   title  'PROCLIB.PAYROLL';
   title2 'First 10 Observations Only';
run;
```

This is a partial printing of PROCLIB.PAYROLL:

```
                    PROCLIB.PAYROLL                          1
               First 10 Observations Only

    Id
  Number   Gender  Jobcode    Salary      Birth      Hired

   1919      M       TA2       34376    12SEP60    04JUN87
   1653      F       ME2       35108    15OCT64    09AUG90
   1400      M       ME1       29769    05NOV67    16OCT90
   1350      F       FA3       32886    31AUG65    29JUL90
   1401      M       TA3       38822    13DEC50    17NOV85
   1499      M       ME3       43025    26APR54    07JUN80
   1101      M       SCP       18723    06JUN62    01OCT90
   1333      M       PT2       88606    30MAR61    10FEB81
   1402      M       TA2       32615    17JAN63    02DEC90
   1479      F       TA3       38785    22DEC68    05OCT89
```

The following PROC FORMAT step creates the format $JOBFMT., which assigns descriptive names for each job:

```
proc format;
    value $jobfmt
          'FA1'='Flight Attendant Trainee'
          'FA2'='Junior Flight Attendant'
          'FA3'='Senior Flight Attendant'
          'ME1'='Mechanic Trainee'
          'ME2'='Junior Mechanic'
          'ME3'='Senior Mechanic'
```

```
          'PT1'='Pilot Trainee'
          'PT2'='Junior Pilot'
          'PT3'='Senior Pilot'
          'TA1'='Ticket Agent Trainee'
          'TA2'='Junior Ticket Agent'
          'TA3'='Senior Ticket Agent'
          'NA1'='Junior Navigator'
          'NA2'='Senior Navigator'
          'BCK'='Baggage Checker'
          'SCP'='Skycap';
run;
```

The FORMAT statement in this PROC MEANS step temporarily associates the $JOBFMT. format with the variable Jobcode:

```
options nodate pageno=1
        linesize=64 pagesize=60;
proc means data=proclib.payroll mean max;
   class jobcode;
   var salary;
   format jobcode $jobfmt.;
   title 'Summary Statistics for';
   title2 'Each Job Code';
run;
```

PROC MEANS produces this output, which uses the $JOBFMT. format:

```
                  Summary Statistics for                    1
                      Each Job Code

                   The MEANS Procedure

               Analysis Variable : Salary

                             N
Jobcode                    Obs            Mean         Maximum
---------------------------------------------------------------
Baggage Checker              9        25794.22        26896.00

Flight Attendant Trainee    11        23039.36        23979.00

Junior Flight Attendant     16        27986.88        28978.00

Senior Flight Attendant      7        32933.86        33419.00

Mechanic Trainee             8        28500.25        29769.00

Junior Mechanic             14        35576.86        36925.00

Senior Mechanic              7        42410.71        43900.00

Junior Navigator             5        42032.20        43433.00

Senior Navigator             3        52383.00        53798.00

Pilot Trainee                8        67908.00        71349.00

Junior Pilot                10        87925.20        91908.00

Senior Pilot                 2        10504.50        11379.00

Skycap                       7        18308.86        18833.00

Ticket Agent Trainee         9        27721.33        28880.00

Junior Ticket Agent         20        33574.95        34803.00

Senior Ticket Agent         12        39679.58        40899.00
---------------------------------------------------------------
```

Note: Because formats are character strings, formats for numeric variables are ignored when the values of the numeric variables are needed for mathematical calculations. △

Example: Grouping or Classifying Formatted Data

If you use a formatted variable to group or classify data, then the procedure uses the formatted values. The following example creates and assigns a format, $CODEFMT., that groups the levels of each job code into one category. PROC MEANS calculates statistics based on the groupings of the $CODEFMT. format.

```
proc format;
    value $codefmt
          'FA1','FA2','FA3'='Flight Attendant'
          'ME1','ME2','ME3'='Mechanic'
          'PT1','PT2','PT3'='Pilot'
          'TA1','TA2','TA3'='Ticket Agent'
                'NA1','NA2'='Navigator'
                      'BCK'='Baggage Checker'
```

```
                      'SCP'='Skycap';
run;

options nodate pageno=1
        linesize=64 pagesize=40;
proc means data=proclib.payroll mean max;
   class jobcode;
   var salary;
   format jobcode $codefmt.;
   title 'Summary Statistics for Job Codes';
   title2 '(Using a Format that Groups the Job Codes)';
run;
```

PROC MEANS produces this output:

```
                Summary Statistics for Job Codes              1
           (Using a Format that Groups the Job Codes)

                      The MEANS Procedure

                  Analysis Variable : Salary

                          N
   Jobcode              Obs             Mean         Maximum
   ---------------------------------------------------------
   Baggage Checker        9         25794.22        26896.00

   Flight Attendant      34         27404.71        33419.00

   Mechanic              29         35274.24        43900.00

   Navigator              8         45913.75        53798.00

   Pilot                 20         72176.25        91908.00

   Skycap                 7         18308.86        18833.00

   Ticket Agent          41         34076.73        40899.00
   ---------------------------------------------------------
```

Example: Temporarily Associating a Format with a Variable

If you want to associate a format with a variable temporarily, then you can use the FORMAT statement. For example, the following PROC PRINT step associates the DOLLAR8. format with the variable Salary for the duration of this PROC PRINT step only:

```
options nodate pageno=1
        linesize=64 pagesize=40;
proc print data=proclib.payroll(obs=10)
           noobs;
   format salary dollar8.;
   title 'Temporarily Associating a Format';
   title2 'with the Variable Salary';
run;
```

PROC PRINT produces this output:

```
                Temporarily Associating a Format                1
                    with the Variable Salary

     Id
   Number   Gender  Jobcode       Salary      Birth      Hired

    1919      M       TA2       $34,376    12SEP60    04JUN87
    1653      F       ME2       $35,108    15OCT64    09AUG90
    1400      M       ME1       $29,769    05NOV67    16OCT90
    1350      F       FA3       $32,886    31AUG65    29JUL90
    1401      M       TA3       $38,822    13DEC50    17NOV85
    1499      M       ME3       $43,025    26APR54    07JUN80
    1101      M       SCP       $18,723    06JUN62    01OCT90
    1333      M       PT2       $88,606    30MAR61    10FEB81
    1402      M       TA2       $32,615    17JAN63    02DEC90
    1479      F       TA3       $38,785    22DEC68    05OCT89
```

Example: Temporarily Dissociating a Format from a Variable

If a variable has a permanent format that you do not want a procedure to use, then temporarily dissociate the format from the variable by using a FORMAT statement.

In this example, the FORMAT statement in the DATA step permanently associates the $YRFMT. variable with the variable Year. Thus, when you use the variable in a PROC step, the procedure uses the formatted values. The PROC MEANS step, however, contains a FORMAT statement that dissociates the $YRFMT. format from Year for this PROC MEANS step only. PROC MEANS uses the stored value for Year in the output.

```
proc format;
   value $yrfmt  '1'='Freshman'
                 '2'='Sophomore'
                 '3'='Junior'
                 '4'='Senior';
run;
data debate;
    input Name $ Gender $  Year $  GPA  @@;
    format year $yrfmt.;
    datalines;
Capiccio m 1 3.598 Tucker   m 1 3.901
Bagwell  f 2 3.722 Berry    m 2 3.198
Metcalf  m 2 3.342 Gold     f 3 3.609
Gray     f 3 3.177 Syme     f 3 3.883
Baglione f 4 4.000 Carr     m 4 3.750
Hall     m 4 3.574 Lewis    m 4 3.421
;

options nodate pageno=1
        linesize=64 pagesize=40;
proc means data=debate mean maxdec=2;
   class year;
   format year;
   title 'Average GPA';
run;
```

PROC MEANS produces this output, which does not use the YRFMT. format:

```
                         Average GPA                         1

                     The MEANS Procedure

                  Analysis Variable : GPA

                             N
                Year        Obs             Mean
                ------------------------------
                1             2             3.75

                2             3             3.42

                3             3             3.56

                4             4             3.69
                ------------------------------
```

Formats and BY-Group Processing

When a procedure processes a data set, it checks to see if a format is assigned to the BY variable. If it is, then the procedure adds observations to the current BY groups until the formatted value changes. If *nonconsecutive* internal values of the BY variable(s) have the same formatted value, then the values are grouped into different BY groups. This results in two BY groups with the same formatted value. Further, if different and *consecutive* internal values of the BY variable(s) have the same formatted value, then they are included in the same BY group.

Formats and Error Checking

If SAS cannot find a format, then it stops processing and prints an error message in the SAS log. You can suppress this behavior with the SAS system option NOFMTERR. If you use NOFMTERR, and SAS cannot find the format, then SAS uses a default format and continues processing. Typically, for the default, SAS uses the BEST*w*. format for numeric variables and the $*w*. format for character variables.

Note: To ensure that SAS can find user-written formats, use the SAS system option FMTSEARCH=. How to store formats is described in "Storing Informats and Formats" on page 466. △

Processing All the Data Sets in a Library

You can use the SAS Macro Facility to run the same procedure on every data set in a library. The macro facility is part of base SAS software.

Example 9 on page 875 shows how to print all the data sets in a library. You can use the same macro definition to perform any procedure on all the data sets in a library. Simply replace the PROC PRINT piece of the program with the appropriate procedure code.

Operating Environment-Specific Procedures

Several base SAS procedures are specific to one operating environment or one release. Appendix 2, "Operating Environment-Specific Procedures," on page 1613 contains a table with additional information. These procedures are described in more detail in the SAS documentation for operating environments.

Statistic Descriptions

Table 2.1 on page 31 identifies common descriptive statistics that are available in several base procedures. See "Keywords and Formulas" on page 1578 for more detailed information about available statistics and theoretical information.

Table 2.1 Common Descriptive Statistics That Base Procedures Calculate

Statistic	**Description**	**Procedures**
confidence intervals		FREQ, MEANS/SUMMARY, TABULATE, UNIVARIATE
CSS	corrected sum of squares	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
CV	coefficient of variation	MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
goodness-of-fit tests		FREQ, UNIVARIATE
KURTOSIS	kurtosis	MEANS/SUMMARY, TABULATE, UNIVARIATE
MAX	largest (maximum) value	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
MEAN	mean	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
MEDIAN	median (50^{th} percentile)	CORR (for nonparametric correlation measures), MEANS/SUMMARY, TABULATE, UNIVARIATE
MIN	smallest (minimum) value	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
MODE	most frequent value (if not unique, the smallest mode is used)	UNIVARIATE
N	number of observations on which calculations are based	CORR, FREQ, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
NMISS	number of missing values	FREQ, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
NOBS	number of observations	MEANS/SUMMARY, UNIVARIATE
PCTN	the percentage of a cell or row frequency to a total frequency	REPORT, TABULATE
PCTSUM	the percentage of a cell or row sum to a total sum	REPORT, TABULATE
Pearson correlation		CORR
percentiles		FREQ, MEANS/SUMMARY, REPORT, TABULATE, UNIVARIATE
RANGE	range	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE

Statistic	Description	Procedures
robust statistics	trimmed means, Winsorized means	UNIVARIATE
SKEWNESS	skewness	MEANS/SUMMARY, TABULATE, UNIVARIATE
Spearman correlation		CORR
STD	standard deviation	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
STDERR	the standard error of the mean	MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
SUM	sum	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
SUMWGT	sum of weights	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
tests of location		UNIVARIATE
USS	uncorrected sum of squares	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE
VAR	variance	CORR, MEANS/SUMMARY, REPORT, SQL, TABULATE, UNIVARIATE

Computational Requirements for Statistics

The following requirements are computational requirements for the statistics that are listed in Table 2.1 on page 31. They do not describe recommended sample sizes.

- □ N and NMISS do not require any nonmissing observations.
- □ SUM, MEAN, MAX, MIN, RANGE, USS, and CSS require at least one nonmissing observation.
- □ VAR, STD, STDERR, and CV require at least two observations.
- □ CV requires that MEAN is not equal to zero.

Statistics are reported as missing if they cannot be computed.

Output Delivery System

What Is the Output Delivery System?

Prior to Version 7, most SAS procedures generated output that was designed for a traditional line-printer. This type of output has limitations that prevents you from getting the most value from your results:

- □ Traditional SAS output is limited to monospace fonts. In a time of desktop document editors and publishing systems, you want more versatility in printed output.
- □ Some commonly used procedures do not produce output data sets. Prior to ODS, if you wanted to use output from one of these procedures as input to another

procedure, then you relied on PROC PRINTTO and the DATA step to retrieve results that otherwise could not be stored in an output data set.

ODS is designed to overcome these limitations and make it easier for you to format your output. The SAS Output Delivery System (ODS) gives you greater flexibility in generating, storing, and reproducing SAS procedure and DATA step output along with a wide range of formatting options. ODS provides formatting functionality that is not available from individual procedures or the DATA step alone.

Gallery of ODS Samples

Here is a sample of the different kinds of formatted output that you can produce with ODS. The input file contains sales records for a company, TruBlend Coffee Makers, that distributes coffee machines.

Traditional SAS Output

Traditional SAS output is Listing output. You do not need to change your SAS programs to create listing output. By default, you continue to create this kind of output even if you also want to create a type of output that contains more formatting.

Output 2.1 Listing Output

```
        Average Quarterly Sales Amount by Each Sales Representative        1

--------------------------------- Quarter=1 ----------------------------------

                              The MEANS Procedure

                        Analysis Variable : AmountSold

                   N
SalesRep         Obs      N          Mean       Std Dev       Minimum       Maximum
___________________________________________________________________________________
Garcia             8      8       14752.5       22806.1         495.0       63333.7

Hollingsworth      5      5       11926.9       12165.2         774.3       31899.1

Jensen             5      5       10015.7        8009.5        3406.7       20904.8
___________________________________________________________________________________

        Average Quarterly Sales Amount by Each Sales Representative        2

--------------------------------- Quarter=2 ----------------------------------

                              The MEANS Procedure

                        Analysis Variable : AmountSold

                   N
SalesRep         Obs      N          Mean       Std Dev       Minimum       Maximum
___________________________________________________________________________________
Garcia             6      6       18143.3       20439.6        1238.8       53113.6

Hollingsworth      6      6       16026.8       14355.0        1237.5       34686.4

Jensen             6      6       12455.1       12713.7        1393.7       34376.7
___________________________________________________________________________________

        Average Quarterly Sales Amount by Each Sales Representative        3

--------------------------------- Quarter=3 ----------------------------------

                              The MEANS Procedure

                        Analysis Variable : AmountSold

                   N
SalesRep         Obs      N          Mean       Std Dev       Minimum       Maximum
___________________________________________________________________________________
Garcia            21     21       10729.8       11457.0        2787.3       38712.5

Hollingsworth     15     15        7313.6        7280.4        1485.0       30970.0

Jensen            21     21       10585.3        7361.7        2227.5       27129.7
___________________________________________________________________________________

        Average Quarterly Sales Amount by Each Sales Representative        4

--------------------------------- Quarter=4 ----------------------------------

                              The MEANS Procedure

                        Analysis Variable : AmountSold

                   N
SalesRep         Obs      N          Mean       Std Dev       Minimum       Maximum
___________________________________________________________________________________
Garcia             5      5       11973.0       10971.8        3716.4       30970.0

Hollingsworth      6      6       13624.4       12624.6        5419.8       38093.1

Jensen             6      6       19010.4       15441.0        1703.4       38836.4
___________________________________________________________________________________
```

Postscript Output

With ODS, you can produce output in PostScript format.

Average Quarterly Sales Amount by Each Sales Representative

The MEANS Procedure

Quarter=1

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	8	8	14752.49	22806.09	495.0000000	63333.65
Hollingsworth	5	5	11926.94	12165.16	774.2500000	31899.10
Jensen	5	5	10015.70	8009.46	3405.70	20904.75

Quarter=2

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	5	5	18143.25	20439.55	1238.80	53113.55
Hollingsworth	5	5	16026.75	14355.04	1237.50	34556.40
Jensen	5	5	12455.10	12713.73	1393.65	34575.70

Quarter=3

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	21	21	10729.82	11457.05	2787.30	38712.50
Hollingsworth	15	15	7313.62	7280.44	1485.00	30970.00
Jensen	21	21	10585.29	7361.66	2227.50	27129.72

Quarter=4

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	5	5	11973.00	10971.77	3716.40	30970.00
Hollingsworth	5	5	13624.42	12624.61	5419.75	33096.10
Jensen	5	5	19010.42	15440.96	1703.65	38836.88

HTML Output

With ODS, you can produce output in Hypertext Markup Language (HTML.) You can browse these files with Internet Explorer, Netscape, or any other browser that fully supports the HTML 3.2 tagset.

Note: To create HTML 4.0 tagsets, use the ODS HTML4 statement. In SAS 9, the ODS HTML statement generates HTML3.2 tagsets. In future realeases of SAS, the ODS HTML statement will support the most current HTML tagsets available. △

Average Quarterly Sales Amount by Each Sales Representative

The MEANS Procedure

Quarter=1

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	8	8	14752.49	22806.09	495.0000000	63333.65
Hollingsworth	5	5	11926.94	12165.18	774.2500000	31899.10
Jensen	5	5	10015.70	8009.46	3406.70	20904.75

Quarter=2

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	6	6	18143.26	20439.58	1238.80	53113.55
Hollingsworth	6	6	16026.76	14355.04	1237.50	34686.40
Jensen	6	6	12455.10	12713.73	1393.65	34376.70

Quarter=3

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	21	21	10729.82	11457.05	2787.30	38712.50
Hollingsworth	15	15	7313.62	7280.44	1485.00	30970.00
Jensen	21	21	10585.29	7361.58	2227.50	27129.72

Quarter=4

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	5	5	11973.00	10971.77	3716.40	30970.00
Hollingsworth	6	6	13624.42	12624.61	5419.75	38093.10
Jensen	6	6	19010.42	15440.98	1703.35	38836.39

RTF Output

With ODS, you can produce output in rich text format (RTF) that can be used with Microsoft Word.

Average Quarterly Sales Amount by Each Sales Representative

The MEANS Procedure

Quarter=1

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	8	8	14752.49	22806.09	495.0000000	63333.65
Hollingsworth	5	5	11926.94	12165.18	774.2500000	31899.10
Jensen	5	5	10015.70	8009.46	3406.70	20904.75

Quarter=2

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	6	6	18143.26	20439.58	1238.80	53113.55
Hollingsworth	6	6	16026.76	14355.04	1237.50	34686.40
Jensen	6	6	12455.10	12713.73	1393.65	34376.70

Quarter=3

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	21	21	10729.82	11457.05	2787.30	38712.50
Hollingsworth	15	15	7313.62	7280.44	1485.00	30970.00
Jensen	21	21	10585.29	7361.68	2227.50	27129.72

Quarter=4

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	5	5	11973.00	10971.77	3716.40	30970.00
Hollingsworth	6	6	13624.42	12624.61	5419.75	38093.10
Jensen	6	6	19010.42	15440.98	1703.35	38836.38

PDF Output

With ODS, you can produce output in Portable Document Format (PDF), which can be viewed with the Adobe Acrobat Reader.

Average Quarterly Sales Amount by Each Sales Representative 1

The MEANS Procedure

Quarter=1

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	8	8	14752.49	22806.09	495.0000000	63333.65
Hollingsworth	5	5	11926.94	12165.18	774.2500000	31899.10
Jensen	5	5	10015.70	8009.46	3406.70	20904.75

Quarter=2

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	6	6	18143.26	20439.58	1238.80	53113.55
Hollingsworth	6	6	16026.76	14355.04	1237.50	34686.40
Jensen	6	6	12455.10	12713.73	1393.65	34376.70

Quarter=3

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	21	21	10729.82	11457.05	2787.30	38712.50
Hollingsworth	15	15	7313.62	7280.44	1485.00	30970.00
Jensen	21	21	10585.29	7361.68	2227.50	27129.72

Quarter=4

Analysis Variable : AmountSold						
SalesRep	N Obs	N	Mean	Std Dev	Minimum	Maximum
Garcia	5	5	11973.00	10971.77	3716.40	30970.00
Hollingsworth	6	6	13624.42	12624.61	5419.75	38093.10
Jensen	6	6	19010.42	15440.98	1703.35	38836.38

XML Output

With ODS, you can produce output that is tagged with Extensible Markup Language (XML) tags.

```
  <?xml version="1.0" encoding="windows-1252" ?>
- <odsxml>
  - <head>
      <meta operator="cabeam" />
    </head>
  - <body>
    - <proc name="Means">
        <label name="IDX" />
        <title class="SystemTitle" toc-level="1">Average Quarterly Sales Amount by Each Sales
          Representative</title>
        <proc-title class="ProcTitle" toc-level="1">The MEANS Procedure</proc-title>
      - <branch class="ContentProcName" toc-level="1" label="Means">
        - <bygroup>
          - <branch class="ByContentFolder" toc-level="2" label="Quarter=1">
            - <leaf class="ContentItem" toc-level="3" label="Summary statistics">
                <byline class="Byline" toc-level="4">Quarter=1</byline>
              - <output name="Summary" label="Summary statistics" clabel="Summary
                  statistics">
                - <output-object type="table" class="Table">
                  - <style>
                      <border spacing="1" padding="7" rules="GROUPS" frame="BOX" />
                    </style>
                  - <colspecs columns="7">
                    - <colgroup>
                        <colspec name="1" width="14" type="string" />
                        <colspec name="2" width="3" align="decimal" type="double" />

                              ...more xml tagged output...

                      - <header name="nobs" type="double" class="Data" row="5"
                          column="2">
                          <value>6</value>
                        </header>
                      - <data raw-value="QBgAAAAAAAA=" name="n" type="double"
                          class="Data" row="5" column="3">
                          <value>6</value>
                        </data>
                      - <data raw-value="QNKQmsX5LF8=" name="mean"
                          type="double" class="Data" row="5" column="4">
                          <value>19010.42</value>
                        </data>
                      - <data raw-value="QM4ofSEjjrI=" name="stddev" type="double"
                          class="Data" row="5" column="5">
                          <value>15440.98</value>
                        </data>
                      - <data raw-value="QJqdZmZmZmY=" name="min"
                          type="double" class="Data" row="5" column="6">
                          <value>1703.35</value>
                        </data>
                      - <data raw-value="QOL2jCj1wo8=" name="max" type="double"
                          class="Data" row="5" column="7">
                          <value>38836.38</value>
                        </data>
                      </row>
                    </output-body>
                  </output-object>
                </output>
              </leaf>
            </branch>
          </bygroup>
        </branch>
      </proc>
    </body>
  </odsxml>
```

Commonly-Used ODS Terminology

data component
: is a form similar to a SAS data set that contains the results (numbers and characters) of a DATA step or PROC step that supports ODS.

table definition
: is a set of instructions that describes how to format the data. This description includes but is not limited to

- the order of the columns
- text and order of column headings
- formats for data
- font sizes and font faces.

output object

is an object that contains both the results of DATA step or PROC step and information about how to format the results. An output object has a name, label, and path. For example, the Basic Statistical Measurement table generated from the UNIVARIATE procedure is an output object. It contains the data component and formatted presentation of the mean, median, mode, standard deviation, variance, range, and interquartile range.

Note: Although many output objects include formatting instructions, not all of them do. In some cases the output object consists of only the data component. △

ODS destinations
produce specific types of output. ODS supports a number of destinations, including the following:

LISTING
produces traditional SAS output (monospace format).

Markup Languages
produce SAS output that is formatted using one of many different markup languages such as Hypertext Markup Language (HTML), Extensible Markup Language (XML), and Latex that you can access with a web browser. SAS supplies many markup languages for you to use ranging from DOCBOOK to TROFF. You can specify a markup language that SAS supplies or create one of your own and store it as a user-defined markup language.

DOCUMENT
produces a hierarchy of output objects that enables you to render multiple ODS output formats without rerunning a PROC or DATA step and gives you more control over the structure of the output.

OUTPUT
produces a SAS data set.

Printer Family
produces output that is formatted for a high-resolution printer such as a PostScript (PS), PDF, or PCL file.

RTF
produces output that is formatted for use with Microsoft Word.

ODS output
ODS output consists of formatted output from any of the ODS destinations. For example, the OUTPUT destination produces SAS data sets; the LISTING destination produces lisiting output; the HTML destination produces output that is formatted in hyper-text markup language.

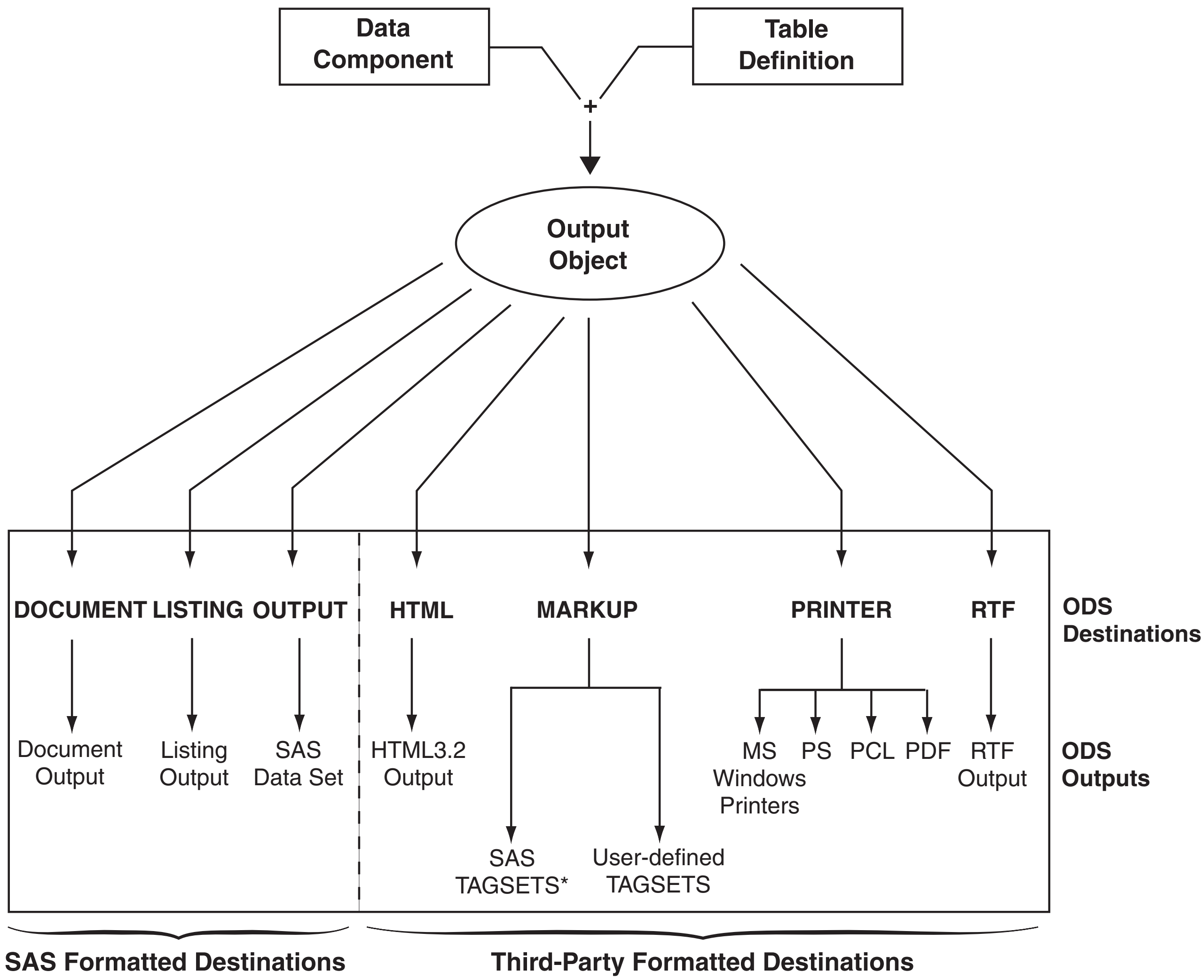
How Does ODS Work?

Components of SAS Output

The Output Delivery System removes responsibility for formatting output from individual procedures and from the DATA step. The procedure or DATA step supplies raw data and the name of the table definition that contains the formatting instructions, and ODS formats the output.

The following figure illustrates how SAS produces ODS output.

ODS Processing: What Goes In and What Comes Out



* List of Tagsets that SAS Supplies and Supports

CHTML	HTML4	SASREPORT	HTMLCSS
SASXMOG	CSVALL	IMODE	SASXMOH
SASXMOIM	WML	DEFAULT	SASXMOR
DOCBOOK	SASXML	EVENT_MAP	SASIOXML
PHTML			
COLORLATEX	GRAPH	PYX	TEXT_MAP
CSV	TPL_STYLE_LIST	TPL_STYLE_MAP	TROFF
CSVBYLINE	LATEX	LATEX2	WMLOLIST

NAMEDHTML	SHORT_MAP	ODSSTYLE	STYLE_DISPLAY
GTABLEAPPLET	STYLE_POPUP		

Features of ODS

ODS is designed to overcome the limitations of traditional SAS output and to make it easy to access and create new formatting options that are available to users. ODS is a method of delivering output in a variey of formats and making the formatted output easy to access.

Important features of ODS include the following:

- ODS combines raw data with one or more table definitions to produce one or more *output objects*. These objects can be sent to any or all ODS destinations. You control the specific type of output from the Output Delivery System by selecting an ODS destination. The currently available ODS destinations can produce
  - traditional monospace output
  - an output data set
  - a SAS document that contains a hierarchy file of the output objects
  - output that is formatted for a high-resolution printer
  - output that is formatted in various markup languages such as Hyper-Text Markup Language (HTML)
  - output that is formatted in rich text format for use with Microsoft Word.
- ODS provides table definitions that define the structure of the output from SAS procedures and from the DATA step. You can customize the output by modifying these definitions or by creating your own.
- ODS provides a way for you to choose individual output objects to send to ODS destinations. For instance, PROC UNIVARIATE produces five output objects. You can easily create HTML output, an output data set, traditional listing output, or printer output from any or all of these output objects. You can send different output objects to different destinations.
- In the SAS windowing environment, ODS stores a link to each output object in the Results folder in the Results window.
- Because formatting is now centralized in ODS, the addition of a new ODS destination does not affect any procedures or the DATA step. As future destinations are added to ODS, they will automatically become available to the DATA step and all procedures that support ODS.
- ODS provides a way for you to produce output for numerous destinations from a single source without having to maintain separate sources for each destination. This feature saves you time and system resources by enabling you to produce multiple kinds of output with a single run of your procedure or data query.

What are ODS Destinations?

Definition of Destination-Independent Input

A fundamental idea of the destination-independent input is that one destination can support a feature even though another destination does not support it. In this case, the request is quietly ignored by the destination that does not support it. Otherwise, ODS would support features that are the least common denominator. You would be forced to insert formats into your input making it difficult to move reports from one output format to another output format. For example, it is easier to use a default style sheet that SAS provides if you are producing only HTML than to use a stylesheet that is not specifically designed for HTML output. However, when you try to print that output or produce a Microsoft Word document from it, you will have to re-do all your work because the stylesheet is spevific to HTML. ODS provides many output format options making it possible to use the appropriate format for the output you want.

Each ODS destination is designed with a different purpose in mind. Although it is possible to use a destination for some other purpose, it is best to use the appropriate destination suited for your purpose. One of the major goals of ODS is to enable the user to produce output for numerous destinations from a single source without having to maintain separate sources for each destination. ODS encourages portable solutions.

The SAS Formatted Destinations

The SAS formatted destinations are designed to create SAS specific entities such as a SAS data set, SAS output listing, or a SAS document. The statements in the ODS SAS Formatted category create the SAS entities.

The three SAS formatted destinations are:

DOCUMENT Destination

The DOCUMENT destination enables you to re-structure, navigate, and replay your data as much as you like to as many destinations as you like without having to rerun your analysis or repeat your database query. The DOCUMENT destination makes your entire output stream available in "raw" form and accessible to you to customize. The output is kept in the original internal representation as a data component plus a table definition. Once the output is in a DOCUMENT form, it is possible to rearrange, restructure, retry, and rerender formatting without rerunning your analysis. Unlike other ODS destinations, the DOCUMENT destination has a GUI interface. However, everything that you can do through the GUI, you can do with batch commands using the ODS DOCUMENT statement and the DOCUMENT procedure.

In the past, each procedure or DATA step produced output that was sent to each destination that you specified. While you could always send your output to as many destinations as you wanted, you had to re-run your procedure or data query if you decided to use another destination that you had not originally designated. The DOCUMENT destination eliminates the need to re-run procedures or repeat data queries by enabling you to store your output objects and simply replay them to different destinations.

LISTING Destination

The LISTING destination produces output that looks the same as the legacy SAS output. Thus ODS is always being used, even when you do not explicitily invoke ODS. The LISTING destination is the default destination that opens when you start your SAS session.

The purpose of the LISTING destination is to enable you to produce output as you always have. You can feel secure knowing that your listing output maintains the same look and presentation as it always has.

Because most procedures share some of the same table defitinions, the output is more consistent. For example, if you have two different procedures producing an ANOVA table, they will produce it in the same way because each procedure uses the same template to describe the table. However, there are four procedures that do not use a defult table definition to porduce their output: PRINT procedure, REPORT procedure, TABULATE procedure, and FREQ procedure's n-way tables.

These procedures use the structure that you specified in your program code to define their tables.

OUTPUT Destination

The OUTPUT destination produces SAS output data sets. Because ODS already knows the logical structure of the data and its native form, ODS can output a SAS data set that represents exactly what the procedure worked with internally. The data sets can be used for further analysis or particularly sophisticated reports where you want to combine similar statistics across different data sets into a single table. You can easily access and process your data using all the SAS data set features. For instance, you can access your data using variable names and perform where–processing just as you would all data from any other SAS data set.

The Third-Party Formatted Destinations

The third-party formatted destinations are where you can apply styles to the output objects that are used by applications outside of SAS. For example, these destinations support attributes such as "font" and "color."

Note: For a list of style elements and valid values, see the style elements appendix in the *SAS Output Delivery System User's Guide*. △

The four categories of third-party formatted destinations are:

□ *Hypertext Markup Language (HTML)*

  The HTML destination produces HTML3.2-compatible output. You can, however, produce (HTML4 stylesheet) output using the HTML4 tagsets.

  The HTML destination can create some or all of the following:

  □ an HTML file (called the *body file*) that contains the results from the procedure

  □ a table of contents that links to the body file

  □ a table of pages that links to the body file

  □ a frame that displays the table of contents, the table of pages, and the body file.

  The body file is required with all ODS HTML output. If you do not want to link to your output, then you do not have to create a table of contents, a table of pages, or a frame file. However, if your output is very large, you may want to create a table of contents and a table of pages for easier reading and transversing through your file.

  The HTML destination is intended only for on-line use, not for printing. To print hardcopies of the output objects, use the PRINTER destination.

□ *Markup Languages (MARKUP) Family*

  The MARKUP destination uses the idea of "tagsets." Just as table definitions describe how to lay out a table and style attributes describe the style of the output, tagsets describe how to produce a markup language output. You can use a tagset that SAS supplies or you can create your own using the TEMPLATE procedure. Like a table definition and style attributes, tagsets enable you to modify your markup language output. For example, each variety of XML can be specified as a new tagset. SAS supplies you with a collection of XML tagsets and enables you to produce a customized variety of XML. The important point is that you can implement a tagset that SAS supplies or a cutomized tagset that you created without having to wait for the next release of SAS. With the additon of modifying and creating your own tagsets by using PROC TEMPLATE, now you have greater flexibility in customizing your output.

Because the MARKUP destination is so flexible, you can use either the SAS tagsets or a tagset that you created. For a complete listing of the markup language tagsets that SAS supplies, see the section on listing tagset names in the *SAS Output Delivery System User's Guide*. To learn how to define your own tagsets, see the section on methods to create your own tagsets in the *SAS Output Delivery System User's Guide*.

The MARKUP destination cannot replace ODS PRINTER or ODS RTF because it has one major limitation: it cannot do text measurement. Therefore, it cannot produce output for a page description language or a hybrid language like RTF which requires all of the text to be measured and placed at a specific position on the page.

- *PRINTER Family*

  The PRINTER destination produces output for

  - printing to physical printers such as Windows printers under Windows, PCL, and PostScript printers on other operating systems
  - producing portable PostScript, PCL, and PDF files.

  The PRINTER destinations produce ODS output that contain page description languages: they describe precise positions where each line of text, each rule, and each graphical element are to be placed on the page. In general, you cannot edit or alter these formats. Therfore, the output from ODS PRINTER is intended to be the final form of the report.

- *Rich Text Format (RTF)*

  RTF produces output for Microsoft Word. While there are other applications that can read RTF files, the RTF output may not work successfully with them.

  The RTF destination enables you to edit the RTF output by viewing a file. For this reason, ODS does not define the "vertical measurement," meaning that SAS does not determine the optimal place to position each item on the page. For instance, page breaks are not always fixed, so when you edit your text, you do not want your RTF output tables to split at inapporpriate places. Your tables can remain whole and in tact on one page or have logical breaks where you specified.

  However, because Microsoft Word needs to know the widths of table columns and it doesn't know how to "panel" tables if they are too wide for the page, ODS does measure the width of the text and tables (horizontal measurement). Therefore, all the column widths can be set properly by SAS and the table can be divided into panels if it is too wide to fit on a single page.

  In short, when producing RTF output for input to Microsoft Word, SAS determines the horizontal measurement and lets Microsoft Word handle the vertical measurement. Because Microsoft Word knows how much room there is on the page even when you edit the file, your tables will display consistently as you specified.

What Controls the Formatting Features of Third-Party Formats?

All the formatting features that control the appearance of the third-party formatted destinations beyond what the LISTING destination can do are controlled by two mechanisms:

- ODS statement options
- ODS style attributes

The ODS statement options control three things:

1. Features that are extremely specific to a given destination, such as stylesheets for HTML.

2 Features that are global to the document, such as AUTHOR and table of contents generation.

3 Features that we expect users to change on virtually every document, such as the output file name.

The ODS style features control the way that individual elements are rendered. Attributes are aspects of a given style, such as type face, weight, font size, and color. The values of the attributes collectively determine the appearance of each part of the document to which the style is applied. The style attributes prevent the necessity to insert destination-specific code (such as raw HTML) into the document by providing a mechanism to describe what the document is intended to do. Each output destination will interpret the attributes to render the best presentation of the document. Because not all destinations are the same, not all attributes can be interpreted by all destinations. The style is defined so that any aspects of the style that cannot be handled by a given destination are ignored by it. For example, PostScript does not support active links, so the URL= attribute is ignored when producing PostScript output.

ODS Destinations and System Resources

ODS destinations can be open or closed. You open and close a destination with the appropriate ODS statement. When a destination is open, ODS sends the output objects to it. An open destination uses system resources even if you use the selection and exclusion features of ODS to select or exclude all objects from the destination. Therefore, to conserve resources, close unnecessary destinations. For more information about using each destination, see the chapter on ODS statements in the *SAS Output Delivery System User's Guide*.

By default, the LISTING destination is open and all other destinations are closed. Consequently, if you do nothing, your SAS programs run and produce listing output looking just as they did in previous releases of SAS before ODS was available.

What Are Table Definitions, Table Elements, and Table Attributes?

A *table definition* describes how to render the output for a tabular output object. (Almost all ODS output is tabular.) A table definition determines the order of column headers and the order of variables, as well the overall look of the output object that uses it. For information about customizing the table definition, see the chapter on the TEMPLATE procedure in the *SAS Output Delivery System User's Guide*.

In addition to the parts of the table definition that order the headers and columns, each table definition contains or references *table elements*. A table element is a collection of table attributes that apply to a particular header, footer, or column. Typically, a *table attribute* specifies something about the data rather than about its presentation. For example, FORMAT specifies the SAS format to use in a column such as the number of decimals to use. However, some table attributes describe presentation aspects of the data such as how many blank characters to place between columns.

Note: The parts of table definitions that control the presentation of the data have no effect on output objects that go to the LISTING or OUTPUT destination. However, the parts that control the structure of the table and the data values do affect listing output. △

For information on table attributes, see the section on table attributes in the *SAS Output Delivery System User's Guide*.

What Are Style Definitions, Style Elements, and Style Attributes?

To customize the output at the level of your entire output stream in a SAS session, you specify a *style definition*. A style definition describes how to render the presentation aspects (color, font face, font size, and so forth) of the entire SAS output. A style definition determines the overall look of the documents that use it.

Each style definition is composed of *style elements*. A style element is a collection of style attributes that apply to a particular part of the output. For example, a style element may contain instructions for the presentation of column headers or for the presentation of the data inside cells. Style elements may also specify default colors and fonts for output that uses the style definition.

Each *style attribute* specifies a value for one aspect of the presentation. For example, the BACKGROUND= attribute specifies the color for the background of an HTML table or for a colored table in printed output. The FONT_STYLE= attribute specifies whether to use a Roman, a slant, or an italic font. For information on style attributes, see the section on style attributes in the *SAS Output Delivery System User's Guide*.

Note: Because style definitions control the presentation of the data, they have no effect on output objects that go to the LISTING or OUTPUT destination. △

What Style Definitions Are Shipped with the Software?

Base SAS software is shipped with many style definitions. To see a list of these styles, you can view them in the SAS Explorer Window, use the TEMPLATE procedure, or use the SQL procedure.

- *SAS Explorer Window:*

  To display a list of the available styles using the SAS Explorer Window, follow these steps:

  1. From any window in an interactive SAS session, select

     View ► Results

  2. In the Results window, select

     View ► Templates

  3. In the Templates window, select and open **`Sashelp.tmplmst`**.
  4. Select and open the **`Styles`** folder, which contains a list of available style definitions. If you want to view the underlying SAS code for a style definition, then select the style and open it.

     Operating Environment Information: For information on navigating in the Explorer window without a mouse, see the section on "Window Controls and General Navigation" in the SAS documentation for your operating environment. △

- *TEMPLATE Procedure:*

  You can also display a list of the available styles by submitting the following PROC TEMPLATE statements:

```
proc template;
   list styles;
run;
```

- *SQL Procedure:*

```
proc sql;
select * from styles.style--name;
```

The *style–name* is the name of any style from the template store (for example, **`styles.default`** or **`styles.beige`**).

For more information on how ODS destinations use styles and how you can customize styles, see the section on the DEFINE STYLE statement in the *SAS Output Delivery System User's Guide*.

How do I Use Style Definitions with Base Procedures?

- Most Base Procedures

  Most Base SAS procedures that support ODS use one or more table definitions to produce output objects. These table definitions include definitions for table elements: columns, headers, and footers. Each table element can specify the use of one or more style elements for various parts of the output. These style elements cannot be specified within the syntax of the procedure, but you can use customized styles for the ODS destinations that you use. For more information abotu customizing tale and styles, see the TEMPLATE procedure in the *SAS Output Delivery System User's Guide*.

- The PRINT, REPORT and TABULATE Procedures

  The PRINT, REPORT and TABULATE procedures provide a way for you to access table elements from the procedure step itself. Accessing the table elements enables you to do such things as specify background colors for specific cells, change the font face for column headers, and more. The PRINT, REPORT, and TABULATE procedures provide a way for you to customize the markup language and printed output directly from the procedure statements that create the report. For more information about customizing the styles for these procedures, see the *Base SAS Procedures Guide*

Customized ODS Output

SAS Output

By default, ODS output is formatted according to instructions that a PROC step or DATA step defines. However, ODS provides ways for you to customize the output. You can customize the output for an entire SAS job, or you can customize the output for a single output object.

Selection and Exclusion Lists

You can specify which output objects that you want to produce by selecting or excluding them in a list. For each ODS destination, ODS maintains either a selection list or an exclusion list. A selection list is a list of output objects that are sent to the destination. An exclusion list is a list of output objects that are excluded from the destination. ODS also maintains an overall selection list or an overall exclusion list. You can use these lists to control which output objects go to the specified ODS destinations.

To see the contents of the lists use the ODS SHOW statement. The lists are written to the SAS log. The following table shows the default lists:

Table 2.2 Default List for Each ODS Destination

ODS Destination	Default List
OUTPUT	EXCLUDE ALL
All others	SELECT ALL

How Does ODS Determine the Destinations for an Output Object?

To specify an output object, you need to know what output objects your SAS program produces. The ODS TRACE statement writes a trace record to the SAS log which includes the path, the label, and other information about each output object that is produced. For more information, see the ODS TRACE statement in the *SAS Output Delivery System User's Guide*. You can specify an output object as

- a full path. For example,

  ```
  Univariate.City_Pop_90.TestsForLocation
  ```

  is the full path of the output object.

- a partial path. A partial path consists of any part of the full path that begins immediately after a period (.) and continues to the end of the full path. For example, if the full path is

  ```
  Univariate.City_Pop_90.TestsForLocation
  ```

  then the partial paths are:

  ```
  City_Pop_90.TestsForLocation
  ```

  ```
  TestsForLocation
  ```

- a label that is surrounded by quotation marks.

  For example,

  ```
  "The UNIVARIATE Procedure"
  ```

- a label path. For example, the label path for the output object is

  ```
  "The UNIVARIATE Procedure"."CityPop_90"
  ."Tests For Location"
  ```

  Note: The trace record shows the label path only if you specify the LABEL option in the ODS TRACE statement. △

- a partial label path. A partial label path consists of any part of the label that begins immediately after a period (.) and continues to the end of the label. For example, if the label path is

  ```
  "The UNIVARIATE Procedure"."CityPop_90"
  ."Tests For Location"
  ```

  then the partial label paths are:

  ```
  "CityPop_90"."Tests For Location"
  ```

  ```
  "Tests For Location"
  ```

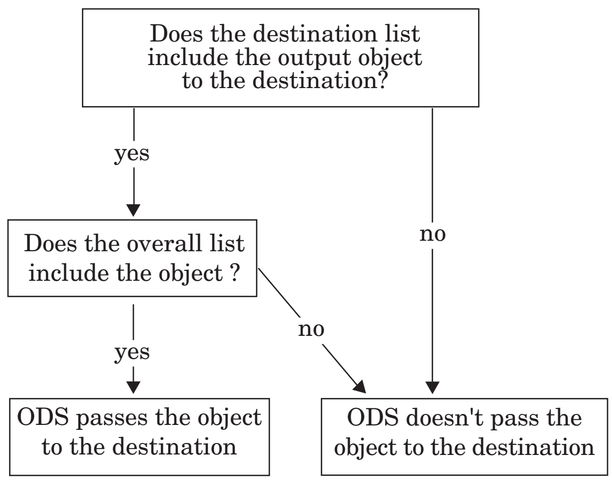
- a mixture of labels and paths.

- any of these specifications followed by a pound sign (#) and a number. For example, TestsForLocation#3 refers to the third output object named TestsForLocation.

As each output object is produced, ODS uses the selection and exclusion lists to determine which destination or destinations to send the output object. The following figure illustrates this process:

Figure 2.1 Directing an Output Object to a Destination

For each destination, ODS first asks if the list for that destination includes the object. If it does not, ODS does not send the output object to that destination. If the list for that destination does include the object, ODS reads the overall list. If the overall list includes the object, ODS sends it to the destination. If the overall list does not include the object, ODS does not send it to the destination.



Note: Although you can maintain a selection list for one destination and an exclusion list for another, it is easier to understand the results if you maintain the same types of lists for all the destinations where you route output. △

Customized Output for an Output Object

For a procedure, the name of the table definition that is used for an output object comes from the procedure code. The DATA step uses a default table definition unless you specify an alternative with the TEMPLATE= suboption in the ODS option in the FILE statement. For more information, see the section on the suboption TEMPLATE= in the *SAS Output Delivery System User's Guide*.

To find out which table definitions a procedure or the DATA step uses for the output objects, you must look at a trace record. To produce a trace record in your SAS log, submit the following SAS statements:

```
ods trace on;
your-proc-or-DATA-step
ods trace off;
```

Remember that not all procedures use table definitions. If you produce a trace record for one of these procedures, no definition appears in the trace record. Conversely, some procedures use multiple table definitions to produce their output, such as the more complex statistical procedures. If you produce a trace record for one of these procedures, more than one definition appears in the trace record.

The trace record refers to the table definition as a template. For a detailed explanation of the trace record, see the section on the ODS TRACE statement in the *SAS Output Delivery System User's Guide*.

You can use PROC TEMPLATE to modify an entire table definition. When a procedure or DATA step uses a table definition, it uses the elements that are defined or referenced in its table definition. In general, you cannot directly specify a table element for your procedure or DATA step to use without modifying the definition itself.

Note: Three base procedures, PROC PRINT, PROC REPORT and PROC TABULATE, do provide a way for you to access table elements from the procedure step itself. Accessing the table elements enables you to customize your report . For more information about these procedures, see the *Base SAS Procedures Guide* △

Conclusion

In the past, the term "output " has generally referred to the outcome of a SAS procedure and DATA step. With the advent of the Output Delivery System, "output" takes on a much broader meaning. ODS is designed to optimize output from SAS procedures and the DATA step. It provides a wide range of formatting options and greater flexibility in generating, storing, and reproducing SAS output.

Important features of ODS include the following:

- ODS combines raw data with one or more table definitions to produce one or more *output objects*. An output object tells ODS how to format the results of a procedure or DATA step.
- ODS provides table definitions that define the structure of the output from SAS procedures and from the DATA step. You can customize the output by modifying these definitions or by creating your own.
- ODS provides a way for you to choose individual output objects to send to ODS destinations.
- ODS stores a link to each output object in the Results folder in the Results window for easy retrieval and access.
- As future destinations are added to ODS, they will automatically become available to the DATA step and all procedures that support ODS.

One of the major goals of ODS is to enable you to produce output for numerous destinations from a single source without having to maintain separate sources for each destination. ODS supports many destinations:

DOCUMENT
: enables you to capture output objects from single run of the analysis and produce multiple reports in various formats whenever you want without re-running your SAS programs.

LISTING
: produces output that looks the same as the legacy SAS v6 output.

HTML
: produces output meant for on-line viewing.

MARKUP
: produces output meant for markup language tagsets.

OUTPUT

produces SAS output data sets thereby eliminating the need to parse PROC PRINTTO output.

PRINTER
: produces presentation-ready printed reports.

RTF
: produces output suitable for Microsoft Word reports.

By default, ODS output is formatted according to instructions that the procedure or DATA step defines. However, ODS provides ways for you to customize the output. You can customize the output for an entire SAS job, or you can customize the output for a single output object.

CHAPTER 3

Statements with the Same Function in Multiple Procedures

Overview

Several statements are available and have the same function in a number of base SAS procedures. Some of the statements are fully documented in *SAS Language Reference: Dictionary*, and others are documented in this section. The following list shows you where to find more information about each statement:

ATTRIB
: affects the procedure output and the output data set. The ATTRIB statement does not permanently alter the variables in the input data set. The LENGTH= option has no effect. See *SAS Language Reference: Dictionary* for complete documentation.

BY
: orders the output according to the BY groups. See “BY” on page 54.

FORMAT
: affects the procedure output and the output data set. The FORMAT statement does not permanently alter the variables in the input data set. The DEFAULT= option is not valid. See *SAS Language Reference: Dictionary* for complete documentation.

FREQ
: treats observations as if they appear multiple times in the input data set. See “FREQ” on page 56.

LABEL
: affects the procedure output and the output data set. The LABEL statement does not permanently alter the variables in the input data set except when it is used with the MODIFY statement in PROC DATASETS. See *SAS Language Reference: Dictionary* for complete documentation.

QUIT
: executes any statements that have not executed and ends the procedure. See “QUIT” on page 58.

WEIGHT

specifies weights for analysis variables in the statistical calculations. See "WEIGHT" on page 59.

WHERE
subsets the input data set by specifying certain conditions that each observation must meet before it is available for processing. See "WHERE" on page 63.

Statements

BY

Orders the output according to the BY groups.

See also: "Creating Titles That Contain BY-Group Information" on page 19

BY <DESCENDING> *variable-1*
<... <DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, then the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are grouped in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, then the procedure treats each contiguous set as a separate BY group.

Note: You cannot use the NOTSORTED option in a PROC SORT step. △

Note: You cannot use the GROUPFORMAT option, which is available in the BY statement in a DATA step, in a BY statement in any PROC step. △

BY-Group Processing

Procedures create output for each BY group. For example, the elementary statistics procedures and the scoring procedures perform separate analyses for each BY group. The reporting procedures produce a report for each BY group.

Note: All base procedures except PROC PRINT process BY groups completely independently. PROC PRINT can report the number of observations in each BY group as well as the number of observations in all BY groups. Similarly, PROC PRINT can sum numeric variables in each BY group and across all BY groups. △

You can use only one BY statement in each PROC step. When you use a BY statement, the procedure expects an input data set that is sorted by the order of the BY variables or one that has an appropriate index. If your input data set does not meet these criteria, then an error occurs. Either sort it with the SORT procedure or create an appropriate index on the BY variables.

Depending on the order of your data, you may need to use the NOTSORTED or DESCENDING option in the BY statement in the PROC step.

For more information on

- the BY statement, see *SAS Language Reference: Dictionary*.
- PROC SORT, see Chapter 39, “The SORT Procedure,” on page 1091.
- creating indexes, see “INDEX CREATE Statement” on page 363.

Procedures That Support the BY Statement

CALENDAR	RANK
CHART	REPORT (nonwindowing environment only)
COMPARE	SORT (required)
CORR	STANDARD
FORMS	SUMMARY
FREQ	TABULATE
MEANS	TIMEPLOT
PLOT	TRANSPOSE
PRINT	UNIVARIATE

Note: In the SORT procedure, the BY statement specifies how to sort the data. With the other procedures, the BY statement specifies how the data are currently sorted. △

Example

This example uses a BY statement in a PROC PRINT step. There is output for each value of the BY variable, Year. The DEBATE data set is created in “Example: Temporarily Dissociating a Format from a Variable” on page 29.

```
options nodate pageno=1 linesize=64
        pagesize=40;
proc print data=debate noobs;
   by year;
   title 'Printing of Team Members';
   title2 'by Year';
run;
```

```
                  Printing of Team Members                  1
                          by Year

------------------------ Year=Freshman -------------------------

                  Name      Gender     GPA

                Capiccio      m       3.598
                Tucker        m       3.901


------------------------ Year=Sophomore ------------------------

                  Name      Gender     GPA

                 Bagwell      f       3.722
                 Berry        m       3.198
                 Metcalf      m       3.342


------------------------- Year=Junior --------------------------

                  Name    Gender     GPA

                  Gold      f       3.609
                  Gray      f       3.177
                  Syme      f       3.883


------------------------- Year=Senior --------------------------

                Name        Gender     GPA

                Baglione      f       4.000
                Carr          m       3.750
                Hall          m       3.574
                Lewis         m       3.421
```

FREQ

Treats observations as if they appear multiple times in the input data set.

Tip: You can use a WEIGHT statement and a FREQ statement in the same step of any procedure that supports both statements.

FREQ *variable*;

Required Arguments

variable

specifies a numeric variable whose value represents the frequency of the observation. If you use the FREQ statement, then the procedure assumes that each observation represents *n* observations, where *n* is the value of *variable*. If *variable* is not an integer, then SAS truncates it. If *variable* is less than 1 or is missing, then the procedure does not use that observation to calculate statistics. If a FREQ statement does not appear, then each observation has a default frequency of 1.

The sum of the frequency variable represents the total number of observations.

Procedures That Support the FREQ Statement

- □ CORR
- □ FORMS
- □ MEANS/SUMMARY
- □ REPORT
- □ STANDARD
- □ TABULATE
- □ UNIVARIATE

Note: PROC FORMS does not calculate statistics. In PROC FORMS, the value of the frequency variable affects the number of form units that are printed for each observation. △

Example

The data in this example represent a ship's course and speed (in nautical miles per hour), recorded every hour. The frequency variable, Hours, represents the number of hours that the ship maintained the same course and speed. Each of the following PROC MEANS steps calculates average course and speed. The different results demonstrate the effect of using Hours as a frequency variable.

The following PROC MEANS step does not use a frequency variable:

```
options nodate pageno=1 linesize=64 pagesize=40;

data track;
   input Course Speed Hours @@;
   datalines;
30  4  8 50 7 20
75 10 30 30 8 10
80  9 22 20 8 25
83 11  6 20 6 20
;


proc means data=track maxdec=2 n mean;
   var course speed;
   title 'Average Course and Speed';
run;
```

Without a frequency variable, each observation has a frequency of 1, and the total number of observations is 8.

```
                    Average Course and Speed                    1

                      The MEANS Procedure

                  Variable    N            Mean
                  -----------------------------
                  Course      8           48.50
                  Speed       8            7.88
                  -----------------------------
```

The second PROC MEANS step uses Hours as a frequency variable:

```
proc means data=track maxdec=2 n mean;
   var course speed;
   freq hours;
   title 'Average Course and Speed';
run;
```

When you use Hours as a frequency variable, the frequency of each observation is the value of Hours, and the total number of observations is 141 (the sum of the values of the frequency variable).

```
                    Average Course and Speed                    1

                      The MEANS Procedure

             Variable              N            Mean
             ----------------------------------------
             Course              141           49.28
             Speed               141            8.06
             ----------------------------------------
```

QUIT

Executes any statements that have not executed and ends the procedure.

QUIT;

Procedures That Support the QUIT Statement

- CATALOG
- DATASETS
- PLOT
- PMENU
- SQL

WEIGHT

Specifies weights for analysis variables in the statistical calculations.

Tip: You can use a WEIGHT statement and a FREQ statement in the same step of any procedure that supports both statements.

WEIGHT *variable*;

Required Arguments

variable

specifies a numeric variable whose values weight the values of the analysis variables. The values of the variable do not have to be integers. The behavior of the procedure when it encounters a nonpositive weight variable value is as follows:

Weight value ...	The procedure ...
0	counts the observation in the total number of observations
less than 0	converts the weight value to zero and counts the observation in the total number of observations
missing	excludes the observation from the analysis

Different behavior for nonpositive values is discussed in the WEIGHT statement syntax under the individual procedure.

Prior to Version 7 of SAS, no base procedure excluded the observations with missing weights from the analysis. Most SAS/STAT procedures, such as PROC GLM, have always excluded not only missing weights but also negative and zero weights from the analysis. You can achieve this same behavior in a base procedure that supports the WEIGHT statement by using the EXCLNPWGT option in the PROC statement.

The procedure substitutes the value of the WEIGHT variable for w_i, which appears in "Keywords and Formulas" on page 1578.

Procedures That Support the WEIGHT Statement

- CORR
- FREQ
- MEANS/SUMMARY
- REPORT
- STANDARD
- TABULATE
- UNIVARIATE

Note: In PROC FREQ, the value of the variable in the WEIGHT statement represents the frequency of occurrence for each observation. See "WEIGHT Statement" on page 540 for more information. △

Calculating Weighted Statistics

The procedures that support the WEIGHT statement also support the VARDEF= option, which lets you specify a divisor to use in the calculation of the variance and standard deviation.

By using a WEIGHT statement to compute moments, you assume that the *i*th observation has a variance that is equal to σ^2/w_i. When you specify VARDEF=DF (the default), the computed variance is a weighted least squares estimate of σ^2. Similarly, the computed standard deviation is an estimate of σ. Note that the computed variance is not an estimate of the variance of the *i*th observation, because this variance involves the observation's weight which varies from observation to observation.

If the values of your variable are counts that represent the number of occurrences of each observation, then use this variable in the FREQ statement rather than in the WEIGHT statement. In this case, because the values are counts, they should be integers. (The FREQ statement truncates any noninteger values.) The variance that is computed with a FREQ variable is an estimate of the common variance, σ^2, of the observations.

Note: If your data come from a stratified sample where the weights w_i represent the strata weights, then neither the WEIGHT statement nor the FREQ statement provides appropriate stratified estimates of the mean, variance, or variance of the mean. To perform the appropriate analysis, consider using PROC SURVEYMEANS, which is a SAS/STAT procedure that is documented in the *SAS/STAT User's Guide*. △

Weighted Statistics Example

As an example of the WEIGHT statement, suppose 20 people are asked to estimate the size of an object 30 cm wide. Each person is placed at a different distance from the object. As the distance from the object increases, the estimates should become less precise.

The SAS data set SIZE contains the estimate (ObjectSize) in centimeters at each distance (Distance) in meters and the precision (Precision) for each estimate. Notice that the largest deviation (an overestimate by 20 cm) came at the greatest distance (7.5 meters from the object). As a measure of precision, 1/Distance, gives more weight to estimates that were made closer to the object and less weight to estimates that were made at greater distances.

The following statements create the data set SIZE:

```
options nodate pageno=1 linesize=64 pagesize=60;

data size;
   input Distance ObjectSize @@;
   Precision=1/distance;
   datalines;
1.5 30 1.5 20 1.5 30 1.5 25
3   43 3   33 3   25 3   30
4.5 25 4.5 36 4.5 48 4.5 33
6   43 6   36 6   23 6   48
7.5 30 7.5 25 7.5 50 7.5 38
;
```

The following PROC MEANS step computes the average estimate of the object size while ignoring the weights. Without a WEIGHT variable, PROC MEANS uses the default weight of 1 for every observation. Thus, the estimates of object size at all distances are given equal weight. The average estimate of the object size exceeds the actual size by 3.55 cm.

```
proc means data=size maxdec=3 n mean var stddev;
   var objectsize;
   title1 'Unweighted Analysis of the SIZE Data Set';
run;
```

```
           Unweighted Analysis of the SIZE Data Set          1

                    The MEANS Procedure

               Analysis Variable : ObjectSize

  N            Mean        Variance         Std Dev
 --------------------------------------------------
 20          33.550          80.892           8.994
 --------------------------------------------------
```

The next two PROC MEANS steps use the precision measure (Precision) in the WEIGHT statement and show the effect of using different values of the VARDEF= option. The first PROC step creates an output data set that contains the variance and standard deviation. If you reduce the weighting of the estimates that are made at greater distances, the weighted average estimate of the object size is closer to the actual size.

```
proc means data=size maxdec=3 n mean var stddev;
   weight precision;
   var objectsize;
   output out=wtstats var=Est_SigmaSq std=Est_Sigma;
   title1 'Weighted Analysis Using Default VARDEF=DF';
run;

proc means data=size maxdec=3 n mean var std
                    vardef=weight;
   weight precision;
   var objectsize;
   title1 'Weighted Analysis Using VARDEF=WEIGHT';
run;
```

In the first PROC MEANS step, the variance is an estimate of σ^2, where the variance of the *i*th observation is assumed to be $var\,(x_i) = \sigma^2/w_i$ and w_i is the weight for the *i*th observation. In the second PROC MEANS step, the computed variance is an estimate of $(n - 1/n)\ \sigma^2/\overline{w}$, where $\overline{w}$ is the average weight. For large n, this is an approximate estimate of the variance of an observation with average weight.

```
          Weighted Analysis Using Default VARDEF=DF          1

                    The MEANS Procedure

               Analysis Variable : ObjectSize

  N            Mean        Variance         Std Dev
 --------------------------------------------------
 20          31.088          20.678           4.547
 --------------------------------------------------
```

```
             Weighted Analysis Using VARDEF=WEIGHT           2

                     The MEANS Procedure

                Analysis Variable : ObjectSize

         N            Mean        Variance         Std Dev
        --------------------------------------------------
        20          31.088          64.525           8.033
        --------------------------------------------------
```

The following statements create and print a data set with the weighted variance and weighted standard deviation of each observation. The DATA step combines the output data set that contains the variance and the standard deviation from the weighted analysis with the original data set. The variance of each observation is computed by dividing Est_SigmaSq, the estimate of σ^2 from the weighted analysis when VARDEF=DF, by each observation's weight (Precision). The standard deviation of each observation is computed by dividing Est_Sigma, the estimate of σ from the weighted analysis when VARDEF=DF, by the square root of each observation's weight (Precision).

```
data wtsize(drop=_freq_ _type_);
   set size;
   if _n_=1 then set wtstats;
   Est_VarObs=est_sigmasq/precision;
   Est_StdObs=est_sigma/sqrt(precision);

proc print data=wtsize noobs;
   title 'Weighted Statistics';
   by distance;
   format est_varobs est_stdobs
          est_sigmasq est_sigma precision 6.3;
```

```
run;
```

```
                     Weighted Statistics                            4

------------------------- Distance=1.5 -------------------------

 Object                      Est_         Est_      Est_       Est_
  Size      Precision       SigmaSq      Sigma     VarObs     StdObs

   30         0.667        20.678       4.547     31.017      5.569
   20         0.667        20.678       4.547     31.017      5.569
   30         0.667        20.678       4.547     31.017      5.569
   25         0.667        20.678       4.547     31.017      5.569


-------------------------- Distance=3 --------------------------

 Object                      Est_         Est_      Est_       Est_
  Size      Precision       SigmaSq      Sigma     VarObs     StdObs

   43         0.333        20.678       4.547     62.035      7.876
   33         0.333        20.678       4.547     62.035      7.876
   25         0.333        20.678       4.547     62.035      7.876
   30         0.333        20.678       4.547     62.035      7.876


------------------------- Distance=4.5 -------------------------

 Object                      Est_         Est_      Est_       Est_
  Size      Precision       SigmaSq      Sigma     VarObs     StdObs

   25         0.222        20.678       4.547     93.052      9.646
   36         0.222        20.678       4.547     93.052      9.646
   48         0.222        20.678       4.547     93.052      9.646
   33         0.222        20.678       4.547     93.052      9.646


-------------------------- Distance=6 --------------------------

 Object                      Est_         Est_      Est_       Est_
  Size      Precision       SigmaSq      Sigma     VarObs     StdObs

   43         0.167        20.678       4.547     124.07     11.139
   36         0.167        20.678       4.547     124.07     11.139
   23         0.167        20.678       4.547     124.07     11.139
   48         0.167        20.678       4.547     124.07     11.139


------------------------- Distance=7.5 -------------------------

 Object                      Est_         Est_      Est_       Est_
  Size      Precision       SigmaSq      Sigma     VarObs     StdObs

   30         0.133        20.678       4.547     155.09     12.453
   25         0.133        20.678       4.547     155.09     12.453
   50         0.133        20.678       4.547     155.09     12.453
   38         0.133        20.678       4.547     155.09     12.453
```

WHERE

Subsets the input data set by specifying certain conditions that each observation must meet before it is available for processing.

WHERE *where-expression*;

Required Arguments

where-expression
is a valid arithmetic or logical expression that generally consists of a sequence of operands and operators. See *SAS Language Reference: Dictionary* for more information on where processing.

Procedures That Support the WHERE Statement

You can use the WHERE statement with any of the following base SAS procedures that read a SAS data set:

CALENDAR	RANK
CHART	REPORT
COMPARE	SORT
CORR	SQL
DATASETS (APPEND statement)	STANDARD
FORMS	TABULATE
FREQ	TIMEPLOT
MEANS/SUMMARY	TRANSPOSE
PLOT	UNIVARIATE
PRINT	

Details

- The CALENDAR and COMPARE procedures and the APPEND statement in PROC DATASETS accept more than one input data set. See the documentation for the specific procedure for more information.
- To subset the output data set, use the WHERE= data set option:

```
proc report data=debate nowd
            out=onlyfr(where=(year='1'));
run;
```

For more information on WHERE=, see *SAS Language Reference: Dictionary*.

Example

In this example, PROC PRINT prints only those observations that meet the condition of the WHERE expression. The DEBATE data set is created in "Example: Temporarily Dissociating a Format from a Variable" on page 29.

```
options nodate pageno=1 linesize=64
        pagesize=40;


proc print data=debate noobs;
   where gpa>3.5;
   title 'Team Members with a GPA';
```

```
   title2 'Greater than 3.5';
run;
```

```
                Team Members with a GPA                       1
                   Greater than 3.5

         Name        Gender    Year          GPA

         Capiccio      m       Freshman     3.598
         Tucker        m       Freshman     3.901
         Bagwell       f       Sophomore    3.722
         Gold          f       Junior       3.609
         Syme          f       Junior       3.883
         Baglione      f       Senior       4.000
         Carr          m       Senior       3.750
         Hall          m       Senior       3.574
```

PART 2

Procedures

PART 2

Procedures

CHAPTER

4

The APPEND Procedure

Overview: APPEND Procedure

The APPEND procedure adds the observations from one SAS data set to the end of another SAS data set.

Generally, the APPEND procedure functions the same as the APPEND statement in the DATASETS procedure. The only difference between the APPEND procedure and the APPEND statement in PROC DATASETS is the default for *libref* in the BASE= and DATA= arguments. For PROC APPEND, the default is either WORK or USER. For the APPEND statement, the default is the libref of the procedure input library.

Syntax: PROC APPEND

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

Reminder: You can use data set options with the BASE= and DATA= options. See "Data Set Options" on page 17 for a list.

Reminder: Complete documentation for the APPEND statement and the APPEND procedure is in "APPEND Statement" on page 335 .

PROC APPEND BASE=<*libref.*>*SAS-data-set* <DATA=<*libref.*>*SAS-data-set*>
<FORCE> <APPENDVER=V6>;

CHAPTER 5

The CALENDAR Procedure

Overview: CALENDAR Procedure

The CALENDAR procedure displays data from a SAS data set in a monthly calendar format. You can produce a *schedule calendar*, which schedules events around holidays and nonwork periods. Or you can produce a *summary calendar*, which summarizes data and displays only one-day events and holidays. When you use PROC CALENDAR you can

- schedule work around holidays and other nonwork periods
- display holidays
- process data about *multiple calendars* in a single step and print them in a separate, mixed, or combined format
- apply different holidays, weekly work schedules, and daily work shifts to multiple calendars in a single PROC step
- produce a mean and a sum for variables based on either the number of days in a month or the number of observations.

PROC CALENDAR also contains features specifically designed to work with PROC CPM in SAS/OR software, a project management scheduling tool.

Simple Schedule Calendar — 7-Day Default Calendar

Output 5.1 on page 75 illustrates the simplest kind of schedule calendar that you can produce. This calendar output displays activities planned by a banking executive. The following statements produce Output 5.1 on page 75.

```
options nodate pageno=1 linesize=132 pagesize=60;

proc calendar data=allacty;
   start date;
   dur long;
run;
```

For the activities data set shown in this calendar, see Example 1 on page 108.

Output 5.1 Simple Schedule Calendar

This calendar uses one of the two default calendars, the 24-hour-day, 7-day-week calendar.

```
                                                       The SAS System                                                    1

 -------------------------------------------------------------------------------------------------------------------------------
 |                                                                                                                             |
 |                                                          July  1996                                                         |
 |                                                                                                                             |
 |-----------------------------------------------------------------------------------------------------------------------------|
 |     Sunday      |     Monday      |     Tuesday     |    Wednesday    |    Thursday     |     Friday      |    Saturday     |
 |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
 |                 |        1        |        2        |        3        |        4        |        5        |        6        |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |+=Interview/JW==+|                 |                 |                 |
 |                 |+Dist. Mtg./All=+|+====Mgrs. Meeting/District 6=====+|                 |+VIP Banquet/JW=+|                 |
 |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
 |        7        |        8        |        9        |       10        |       11        |       12        |       13        |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |+Planning Counci+|+=Seminar/White=+|                 |
 |                 |+=================Trade Show/Knox==================+|+====Mgrs. Meeting/District 7=====+|                 |
 |                 |+===============================Sales Drive/District 6================================+|                 |
 |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
 |       14        |       15        |       16        |       17        |       18        |       19        |       20        |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |+NewsLetter Dead+|+Co. Picnic/All=+|                 |
 |                 |                 |+==Dentist/JW===+|+Bank Meeting/1s+|+Planning Counci+|+=Seminar/White=+|                 |
 |                 |+===============================Sales Drive/District 7================================+|                 |
 |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
 |       21        |       22        |       23        |       24        |       25        |       26        |       27        |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |+=Birthday/Mary=+|+======Close Sale/WYGIX Co.=======+|                 |
 |                 |+==============Inventors Show/Melvin===============+|+Planning Counci+|                 |                 |
 |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
 |       28        |       29        |       30        |       31        |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 |                 |                 |                 |                 |                 |                 |                 |
 -------------------------------------------------------------------------------------------------------------------------------
```

Advanced Schedule Calendar

Output 5.2 on page 77 is an advanced schedule calendar produced by PROC CALENDAR. The statements that create this calendar

- schedule activities around holidays
- identify separate calendars
- print multiple calendars in the same report
- apply different holidays to different calendars

- apply different work patterns to different calendars.

For an explanation of the program that produces this calendar, see Example 4 on page 120.

Output 5.2 Advanced Schedule Calendar

```
                                  Well Drilling Work Schedule: Combined Calendars                                  1

          ------------------------------------------------------------------------------------------------------------------
          |                                                                                                                |
          |                                                July  1996                                                      |
          |                                                                                                                |
          |----------------------------------------------------------------------------------------------------------------|
          |     Sunday     |     Monday     |    Tuesday     |   Wednesday    |    Thursday    |     Friday     |    Saturday    |
----------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |                |       1        |       2        |       3        |       4        |       5        |       6        |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |                |                |                |**Independence**|+Assemble Tank/>|                |
|         |                |                |                |                |                |+Lay Power Line>|                |
|         |                |+==============Drill Well/$1,000.00==============>|                |<Drill Well/$1,+|                |
|.........|................|................|................|................|................|................|................|
| CAL2    |                |                |                |+=======================Excavate/$3,500.00========================>|
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       7        |       8        |       9        |       10       |       11       |       12       |       13       |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |+==================Build Pump House/$2,000.00====================+|                |                |
|         |                |<====================Assemble Tank/$1,000.00=====================+|                |                |
|         |                |<===Lay Power Line/$2,000.00====+|                |+===Pour Foundation/$1,500.00===>|                |
|.........|................|................|................|................|................|................|................|
| CAL2    |                |<Excavate/$3,50>|****Vacation****|<Excavate/$3,50+|                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       14       |       15       |       16       |       17       |       18       |       19       |       20       |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |+==============================Install Pump/$500.00==============================+|                |
|         |                |<===========Pour Foundation/$1,500.00============+|                |+Install Pipe/$>|                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       21       |       22       |       23       |       24       |       25       |       26       |       27       |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |+============================Erect Tower/$2,500.00==============================>|                |
|         |                |<====Install Pipe/$1,000.00=====+|                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       28       |       29       |       30       |       31       |                |                |                |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |<Erect Tower/$2+|                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
----------------------------------------------------------------------------------------------------------------------------
```

More Advanced Scheduling and Project Management Tasks

For more complex scheduling tasks, consider using the CPM procedure in SAS/OR software. PROC CALENDAR requires that you specify the starting date of each activity. When the beginning of one task depends on the completion of others and a date slips in a schedule, recalculating the schedule can be time-consuming. Instead of manually recalculating dates, you can use PROC CPM to calculate dates for project activities based on an initial starting date, activity durations, and which tasks are identified as *successors* to others. For an example, see Example 6 on page 128.

Simple Summary Calendar

Output 5.3 on page 78 shows a simple summary calendar that displays the number of meals served daily in a hospital cafeteria:

```
options nodate pageno=1 linesize=132 pagesize=60;

proc calendar data=meals;
   start date;
   sum brkfst lunch dinner;
   mean brkfst lunch dinner;
run;
```

In a summary calendar, each piece of information for a given day is the value of a variable for that day. The variables can be either numeric or character, and you can format them as necessary. You can use the SUM and MEAN options to calculate sums and means for any numeric variables. These statistics appear in a box below the calendar, as shown in Output 5.3 on page 78. The data set shown in this calendar is created in Example 7 on page 134.

Output 5.3 Simple Summary Calendar

```
                                        The SAS System                                        1

      ---------------------------------------------------------------------------------------------------
      |                                                                                                 |
      |                                        December  1996                                          |
      |                                                                                                 |
      |-------------------------------------------------------------------------------------------------|
      |   Sunday    |   Monday    |   Tuesday   |  Wednesday  |  Thursday   |   Friday    |  Saturday   |
      |-------------+-------------+-------------+-------------+-------------+-------------+-------------|
      |      1      |      2      |      3      |      4      |      5      |      6      |      7      |
      |             |             |             |             |             |             |             |
      |             |         123 |         188 |         123 |         200 |         176 |             |
      |             |         234 |         188 |         183 |         267 |         165 |             |
      |             |         238 |         198 |         176 |         243 |         177 |             |
      |-------------+-------------+-------------+-------------+-------------+-------------+-------------|
      |      8      |      9      |     10      |     11      |     12      |     13      |     14      |
      |             |             |             |             |             |             |             |
      |             |         178 |         165 |         187 |         176 |         187 |             |
      |             |         198 |         176 |         176 |         187 |         187 |             |
      |             |         187 |         187 |         231 |         222 |         123 |             |
      |-------------+-------------+-------------+-------------+-------------+-------------+-------------|
      |     15      |     16      |     17      |     18      |     19      |     20      |     21      |
      |             |             |             |             |             |             |             |
      |             |         176 |         156 |         198 |         178 |         165 |             |
      |             |         165 |           . |         143 |         198 |         176 |             |
      |             |         177 |         167 |         167 |         187 |         187 |             |
      |-------------+-------------+-------------+-------------+-------------+-------------+-------------|
      |     22      |     23      |     24      |     25      |     26      |     27      |     28      |
      |             |             |             |             |             |             |             |
      |             |         187 |             |             |             |             |             |
      |             |         187 |             |             |             |             |             |
      |             |         123 |             |             |             |             |             |
      |-------------+-------------+-------------+-------------+-------------+-------------+-------------|
      |     29      |     30      |     31      |             |             |             |             |
      |             |             |             |             |             |             |             |
      |             |             |             |             |             |             |             |
      |             |             |             |             |             |             |             |
      |             |             |             |             |             |             |             |
      ---------------------------------------------------------------------------------------------------

                            ----------------------------------------
                            |          |      Sum     |    Mean     |
                            |          |              |             |
                            | Brkfst   |         2763 |     172.688 |
                            | Lunch    |         2830 |     188.667 |
                            | Dinner   |         2990 |     186.875 |
                            ----------------------------------------
```

Syntax: CALENDAR Procedure

Required: You must use a START statement.

Required: For schedule calendars, you must also use a DUR or a FIN statement.

Tip: If you use a DUR or FIN statement, PROC CALENDAR produces a schedule calendar.

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can use the FORMAT, LABEL, and WHERE statements as well as any global statements.

PROC CALENDAR <*option(s)*>;

START *variable*;

BY <DESCENDING> *variable-1*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

CALID *variable*
</ OUTPUT=COMBINE | MIX | SEPARATE>;

DUR *variable*;

FIN *variable*;

HOLISTART *variable*;
HOLIDUR *variable*;
HOLIFIN *variable*;
HOLIVAR *variable*;

MEAN *variable(s)* </ FORMAT=*format-name*>;

OUTSTART *day-of-week*;
OUTDUR *number-of-days*;
OUTFIN *day-of-week*;

SUM *variable(s)* </ FORMAT=*format-name*>;

VAR *variable(s)*;

The following table lists the statements and options available in the CALENDAR procedure according to function.

To do this	Use this statement
Create summary calendar	MEAN SUM
Create schedule calendar	DUR or FIN
Create multiple calendars	CALID
Specify holidays	HOLISTART HOLIDUR HOLIFIN HOLIVAR
Control display	OUTSTART OUTDUR OUTFIN
Specify grouping	BY CALID

PROC CALENDAR Statement

PROC CALENDAR <*option(s)*>;

To do this	Use this option
Specify data sets containing	
weekly work schedules	CALEDATA=
activities	DATA=
holidays	HOLIDATA=
unique shift patterns	WORKDATA=
Control printing	
display all months, even if no activities exist	FILL
define characters used for outlines, dividers, and so on	FORMCHAR=
specify the type of heading for months	HEADER=
display month and weekday names in local language (experimental)	LOCALE
specify how to show missing values	MISSING
suppress the display of Saturdays and Sundays	WEEKDAYS
Specify time or duration	
specify that START and FIN variables are in DATETIME format	DATETIME
specify the number of hours in a standard work day	DAYLENGTH=
specify the units of the DUR and HOLIDUR variables	INTERVAL=
Control summary information	
identify variables in the calendar	LEGEND
specify the type of mean to calculate	MEANTYPE=

Options

CALEDATA=*SAS-data-set*
specifies the *calendar data set*, a SAS data set that contains weekly work schedules for multiple calendars.

Default: If you omit the CALEDATA= option, PROC CALENDAR uses a default work schedule, as described in “The Default Calendars” on page 98.

Tip: A calendar data set is useful if you are using multiple calendars or a nonstandard work schedule.

See also: “Calendar Data Set” on page 104

Featured in: Example 3 on page 115

DATA=*SAS-data-set*
specifies the *activities data set*, a SAS data set that contains starting dates for all activities and variables to display for each activity. Activities must be sorted or indexed by starting date.

Default: If you omit the DATA= option, the most recently created SAS data set is used.

See also: "Activities Data Set" on page 102

Featured in: All examples. See "Examples: CALENDAR Procedure" on page 108

DATETIME

specifies that START and FIN variables contain values in DATETIME. format.

Default: If you omit the DATETIME option, PROC CALENDAR assumes that the START and FIN values are in the DATE. format.

Featured in: Example 3 on page 115

DAYLENGTH=*hours*

gives the number of hours in a standard working day. The hour value must be a SAS TIME value.

Default: 24 if INTERVAL=DAY (the default), 8 if INTERVAL=WORKDAY.

Restriction: DAYLENGTH= applies only to schedule calendars.

Interaction: If you specify the DAYLENGTH= option and the calendar data set contains a D_LENGTH variable, PROC CALENDAR uses the DAYLENGTH= value only when the D_LENGTH value is missing.

Interaction: When INTERVAL=DAY and you have no CALEDATA= data set, specifying a DAYLENGTH= value has no effect.

Tip: The DAYLENGTH= option is useful when you use the DUR statement and your work schedule contains days of varying lengths, for example, a 5 half-day work week. In a work week with varying day lengths, you need to set a standard day length to use in calculating duration times. For example, an activity with a duration of 3.0 workdays lasts 24 hours if DAYLENGTH=8:00 or 30 hours if DAYLENGTH=10:00.

Tip: Instead of specifying the DAYLENGTH= option, you can specify the length of the working day by using a D_LENGTH variable in the CALEDATA= data set. If you use this method, you can specify different standard day lengths for different calendars.

See also: "Calendar Data Set" on page 104 for more information on setting the length of the standard workday

FILL

displays all months between the first and last activity, start and finish dates inclusive, including months that contain no activities.

Default: If you do not specify FILL, PROC CALENDAR prints only months that contain *activities*. (Months that contain only *holidays* are not printed.)

Featured in: Example 5 on page 125

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'

defines the characters to use for constructing the outlines and dividers for the cells in the calendar as well as all identifying markers (such as asterisks and arrows) used to indicate holidays or continuation of activities in PROC CALENDAR output.

position(s)

identifies the position of one or more characters in the SAS formatting-character string. A space or a comma separates the positions.

Default: Omitting (*position(s)*) is the same as specifying all 20 possible system formatting characters, in order.

Range: PROC CALENDAR uses 17 of the 20 formatting characters that SAS provides. Table 5.1 on page 83 shows the formatting characters that PROC CALENDAR uses. Figure 5.1 on page 84 illustrates their use in PROC CALENDAR output.

formatting-character(s)

lists the characters to use for the specified positions. PROC CALENDAR assigns characters in *formatting-character(s)* to *position(s)*, in the order that they are listed. For instance, the following option assigns an asterisk (*) to the twelfth position, assigns a single dash (-) to the thirteenth, and does not alter remaining characters:

```
formchar(12 13)='*-'
```

These new settings change the activity line from this:

```
+=================ACTIVITY===============+
```

to this:

```
*------------------ACTIVITY---------------*
```

Interaction: The SAS system option FORMCHAR= specifies the default formatting characters. The SAS system option defines the entire string of formatting characters. The FORMCHAR= option in a procedure can redefine selected characters.

Tip: You can use any character in *formatting-characters*, including hexadecimal characters. If you use hexadecimal characters, you must put an **x** after the closing quote. For instance, the following option assigns the hexadecimal character 2D to the third formatting character, the hexadecimal character 7C to the seventh character, and does not alter the remaining characters:

```
formchar(3,7)='2D7C'x
```

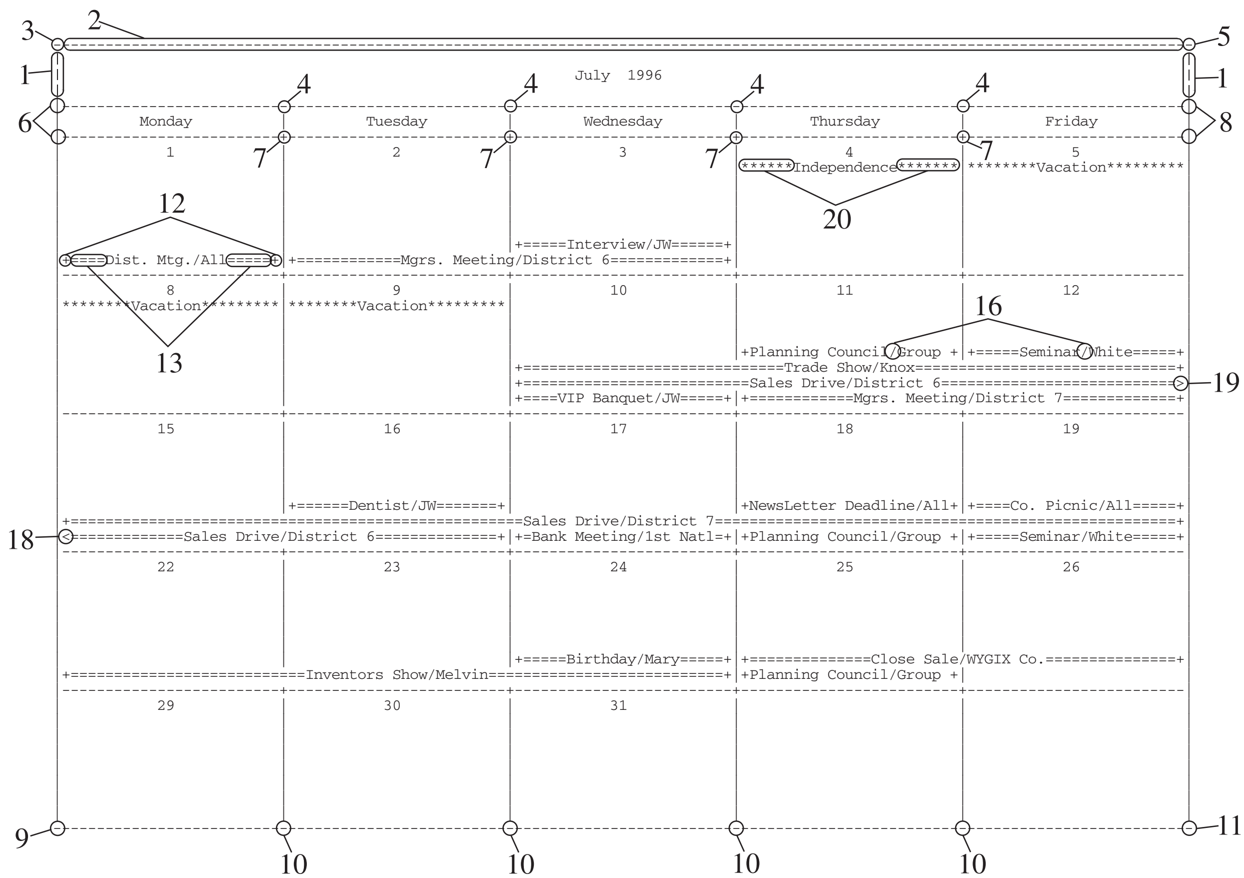
See also: For information on which hexadecimal codes to use for which characters, consult the documentation for your hardware.

Table 5.1 Formatting Characters Used by PROC CALENDAR

Position	Default	Used to draw
1	|	vertical bar
2	-	horizontal bar
3	-	cell: upper left corner
4	-	cell: upper middle intersection
5	-	cell: upper right corner
6	|	cell: middle left cell side
7	+	cell: middle middle intersection
8	|	cell: middle right cell side
9	-	cell: lower left corner
10	-	cell: lower middle intersection
11	-	cell: lower right corner
12	+	activity start and finish
13	=	activity line
16	/	activity separator
18	<	activity continuation from

Position	Default	Used to draw
19	>	activity continuation to
20	*	holiday marker

Figure 5.1 Formatting Characters in PROC CALENDAR Output



HEADER=SMALL | MEDIUM | LARGE
specifies the type of heading to use in printing the name of the month.

SMALL
prints the month and year on one line.

MEDIUM
prints the month and year in a box four lines high.

LARGE
prints the month seven lines high using asterisks (*). The year is included if space is available.

Default: MEDIUM

HOLIDATA=*SAS-data-set*
specifies the *holidays data set*, a SAS data set containing the holidays you want to display in the output. One variable must contain the holiday names and another must contain the starting dates for each holiday. PROC CALENDAR marks holidays in the calendar output with asterisks (*) when space permits.

Interaction: Displaying holidays on a calendar requires a holidays data set and a HOLISTART statement. A HOLIVAR statement is recommended for naming holidays. HOLIDUR is required if any holiday lasts longer than one day.

Tip: The holidays data set does not require sorting.

See also: "Holidays Data Set" on page 103

Featured in: All examples. See "Examples: CALENDAR Procedure" on page 108

INTERVAL=DAY | WORKDAY

specifies the units of the DUR and HOLIDUR variables to one of two default daylengths:

DAY

specifies the values of the DUR and HOLIDUR variables in units of 24-hour days and specifies the default 7-day calendar. For instance, a DUR value of 3.0 is treated as 72 hours. The default calendar work schedule consists of seven working days, all starting at 00:00 with a length of 24:00.

WORKDAY

specifies the values of the DUR and HOLIDUR variables in units of 8-hour days and specifies that the default calendar contains five days a week, Monday through Friday, all starting at 09:00 with a length of 08:00. When WORKDAY is specified, PROC CALENDAR treats the values of the DUR and HOLIDUR variables in units of working days, as defined in the DAYLENGTH= option, the CALEDATA= data set, or the default calendar. For example, if the working day is 8 hours long, a DUR value of 3.0 is treated as 24 hours.

Default: DAY

Interaction: In the absence of a CALEDATA= data set, PROC CALENDAR uses the work schedule defined in a default calendar.

Interaction: The WEEKDAYS option automatically sets the INTERVAL= value to WORKDAY.

See also: "Calendars and Multiple Calendars" on page 99 and "Calendar Data Set" on page 104 for more information on the INTERVAL= option and the specification of working days; "The Default Calendars" on page 98

Featured in: Example 5 on page 125

LEGEND

prints the names of the variables whose values appear in the calendar. This identifying text, or legend box, appears at the bottom of the page for each month if space permits; otherwise, it is printed on the following page. PROC CALENDAR identifies each variable by name or by label if one exists. The order of variables in the legend matches their order in the calendar.

Restriction: LEGEND applies only to summary calendars.

Interaction: If you use the SUM and MEAN statements, the legend box also contains SUM and MEAN values.

Featured in: Example 8 on page 138

LOCALE (Experimental)

prints the names of months and weekdays in the language that is indicated by the value of the LOCALE= SAS system option. The LOCALE option in PROC CALENDAR does not change the starting day of the week.

Default: If LOCALE is not specified, then names of months and weekdays are printed in English.

CAUTION:

LOCALE is an experimental option that is available in Version 9. Do not use this option in production jobs. △

MEANTYPE=NOBS | NDAYS
specifies the type of mean to calculate for each month.

NOBS
calculates the mean over the number of *observations* displayed in the month.

NDAYS
calculates the mean over the number of *days* displayed in the month.

Default: NOBS

Restriction: MEANTYPE= applies only to summary calendars.

Interaction: Normally, PROC CALENDAR displays all days for each month. However, it may omit some days if you use the OUTSTART statement with the OUTDUR or OUTFIN statement.

Featured in: Example 7 on page 134

MISSING
determines how missing values are treated, based on the type of calendar.

Summary Calendar
If there is a day without an activity scheduled, PROC CALENDAR prints the values of variables for that day using the SAS or user-defined format specified for missing values.

Default: If you omit MISSING, days without activities contain no values.

Schedule Calendar
variables with missing values appear in the label of an activity, using the format specified for missing values.

Default: If you do not specify MISSING, PROC CALENDAR ignores missing values in labeling activities.

See also: "Missing Values in Input Data Sets" on page 106 for more information on missing values

WEEKDAYS
suppresses the display of Saturdays and Sundays in the output. It also specifies that the value of the INTERVAL= option is WORKDAY.

Default: If you omit WEEKDAYS, the calendar displays all seven days.

Tip: The WEEKDAYS option is an alternative to using the combination of INTERVAL=WORKDAY and the OUTSTART and OUTFIN statements, as shown here:

Example Code 5.1 Illustration of Formatting Characters in PROC CALENDAR Output

```
proc calendar weekdays;
   start date;
run;

proc calendar interval=workday;
   start date;
   outstart monday;
   outfin friday;
run;
```

Featured in: Example 1 on page 108

WORKDATA=*SAS-data-set*

specifies the *workdays data set*, a SAS data set that defines the work pattern during a standard working day. Each numeric variable in the workdays data set denotes a unique workshift pattern during one working day.

Tip: The workdays data set is useful in conjunction with the calendar data set.

See also: "Workdays Data Set" on page 106 and "Calendar Data Set" on page 104

Featured in: Example 3 on page 115

BY Statement

Processes activities separately for each BY group, producing a separate calendar for each value of the BY variable.

Calendar type: Summary and schedule

Main discussion: "BY" on page 54

See also: "CALID Statement" on page 88

BY <DESCENDING> *variable-1*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable, but the observations in the data set must be sorted by all the variables that you specify or have an appropriate index. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are grouped in another way, for example, chronological order.

Showing Multiple Calendars in Related Groups

When you use the CALID statement, you can process activities that apply to different calendars, indicated by the value of the CALID variable. Because you can specify only one CALID variable, however, you can create only one level of grouping. For example, if you want a calendar report to show the activities of several departments within a company, you can identify each department with the value of the CALID variable and produce calendar output that shows the calendars for all departments.

When you use a BY statement, however, you can further divide activities into related groups. For example, you can print calendar output that groups departmental calendars by division. The observations for activities must contain a variable that identifies which department an activity belongs to and a variable that identifies the division that a department resides in. Specify the variable that identifies the department with the CALID statement. Specify the variable that identifies the division with the BY statement.

CALID Statement

Processes activities in groups defined by the values of a calendar identifier variable.

Calendar type: Summary and schedule

Tip: Useful for producing multiple schedule calendars and for use with SAS/OR software.

See also: "Calendar Data Set" on page 104

Featured in: Example 2 on page 112, Example 3 on page 115, and Example 6 on page 128

CALID *variable*
 </ OUTPUT=COMBINE | MIX | SEPARATE>;

Required Arguments

variable

a character or numeric variable that identifies which calendar an observation contains data for.

Requirement: If you specify the CALID variable, both the activities and holidays datasets must contain this variable. If either of them does not contain it, a default calendar is used.

Interaction: SAS/OR software uses this variable to identify which calendar an observation contains data for.

Tip: You do not need to use a CALID statement to create this variable. You can include the default variable _CALID_ in the input data sets.

See also: "Calendar Data Set" on page 104

Options

OUTPUT=COMBINE | MIX | SEPARATE

controls the amount of space required to display output for multiple calendars.

COMBINE

produces one page for each month that contains activities and subdivides each day by the CALID value.

Restriction: The input data must be sorted by or indexed on the START variable.

Featured in: Example 2 on page 112 and Example 4 on page 120

MIX

produces one page for each month that contains activities and does not identify activities by the CALID value.

Restriction: The input data must be sorted by or indexed on the START variable.

Tip: MIX requires the least space for output.

Featured in: Example 4 on page 120

SEPARATE

produces a separate page for each value of the CALID variable.

Restriction: The input data must be sorted by the CALID variable and then by the START variable or must contain an appropriate composite index.

Featured in: Example 3 on page 115 and Example 8 on page 138

Default: COMBINE

DUR Statement

Specifies the variable that contains the duration of each activity.

Alias: DURATION

Calendar type: Schedule

Interaction: If you use both a DUR and a FIN statement, DUR is ignored.

Tip: To produce a schedule calendar, you must use either a DUR or FIN statement.

Featured in: All schedule calendars (see "Examples: CALENDAR Procedure" on page 108)

DUR *variable*;

Required Arguments

variable

contains the duration of each activity in a schedule calendar.

Range: The duration may be a real or integral value.

Restriction: This variable must be in the activities data set.

See also: For more information on activity durations, see "Activities Data Set" on page 102 and "Calendar Data Set" on page 104

Duration

- Duration is measured inclusively from the start of the activity (as given in the START variable). In the output, any activity lasting part of a day is displayed as lasting a full day.
- The INTERVAL= option in a PROC CALENDAR statement automatically sets the unit of the duration variable, depending on its own value as follows:

If INTERVAL= . . .	Then the default length of the duration unit is . . .
DAY (the default)	24 hours
WORKDAY	8 hours

- You can override the default length of a duration unit by using
  - the DAYLENGTH= option
  - a D_LENGTH variable in the CALEDATA= data set.

FIN Statement

Specifies the variable in the activities data set that contains the finishing date of each activity.

Alias: FINISH

Calendar type: Schedule

Interaction: If you use both a FIN and a DUR statement, FIN is used.

Tip: To produce a schedule calendar, you must use either a FIN or DUR statement.

Featured in: Example 6 on page 128

FIN *variable*;

Required Arguments

variable
contains the finishing date of each activity.

Restriction: The values of *variable* must be either SAS date or datetime values.

Restriction: If the FIN variable contains datetime values, you must specify the DATETIME option in the PROC CALENDAR statement.

Restriction: Both the START and FIN variables must have matching formats. For example, if one contains datetime values, so must the other.

HOLIDUR Statement

Specifies the variable in the holidays data set that contains the duration of each holiday for a schedule calendar.

Alias: HOLIDURATION

Calendar type: Schedule

Default: If you do not use a HOLIDUR or HOLIFIN statement, all holidays last one day.

Restriction: Cannot use with a HOLIFIN statement.

Featured in: Example 1 on page 108 through Example 5 on page 125

HOLIDUR *variable*;

Required Arguments

variable

contains the duration of each holiday.

Range: The duration may be a real or integral value.

Restriction: This variable must be in the holidays data set.

Featured in: Example 3 on page 115 and Example 8 on page 138

Holiday Duration

- If you use both the HOLIFIN and HOLIDUR statement, PROC CALENDAR uses the HOLIFIN variable value to define each holiday's duration.
- Set the *unit* of the holiday duration variable in the same way that you set the unit of the duration variable; use either the INTERVAL= and DAYLENGTH= options or the CALEDATA= data set.
- Duration is measured inclusively from the start of the holiday (as given in the HOLISTART variable). In the output, any holiday lasting at least half a day appears as lasting a full day.

HOLIFIN Statement

Specifies the variable in the holidays data set containing the finishing date of each holiday.

Alias: HOLIFINISH

Calendar type: Schedule

Default: If you do not use a HOLIFIN or HOLIDUR statement, all holidays last one day.

HOLIFIN *variable*;

Required Arguments

variable
contains the finishing date of each holiday.

Restriction: This variable must be in the holidays data set.

Restriction: Values of *variable* must be in either SAS date or datetime values.

Restriction: If the HOLIFIN variable contains datetime values, you must specify the DATETIME option in the PROC CALENDAR statement.

Holiday Duration

If you use both the HOLIFIN and the HOLIDUR statement, PROC CALENDAR uses only the HOLIFIN variable.

HOLISTART Statement

Specifies a variable in the holidays data set that contains the starting date of each holiday.

Alias: HOLISTA, HOLIDAY

Calendar type: Summary and schedule

Requirement: When you use a holidays data set, HOLISTART is required.

Featured in: Example 1 on page 108 through Example 5 on page 125

HOLISTART *variable*;

Required Arguments

variable
contains the starting date of each holiday.

Restriction: Values of *variable* must be in either SAS date or datetime values.

Restriction: If the HOLISTART variable contains datetime values, specify the DATETIME option in the PROC CALENDAR statement.

Details

- The holidays data set need not be sorted.
- All holidays last only one day, unless you use a HOLIFIN or HOLIDUR statement.
- If two or more holidays occur on the same day, PROC CALENDAR uses only the first observation.

HOLIVAR Statement

Specifies a variable in the holidays data set whose values are used to label the holidays.

Alias: HOLIVARIABLE, HOLINAME

Calendar type: Summary and schedule

Default: If you do not use a HOLIVAR statement, PROC CALENDAR uses the word `DATE` to identify holidays.

Featured in: Example 1 on page 108 through Example 5 on page 125

HOLIVAR *variable*;

Required Arguments

variable
a variable whose values are used to label the holidays. Typically, this variable contains the names of the holidays.

Range: character or numeric.

Restriction: This variable must be in the holidays data set.

Tip: You can format the HOLIVAR variable as you like.

MEAN Statement

Specifies numeric variables in the activities data set for which mean values are to be calculated for each month.

Calendar type: Summary

Tip: You can use multiple MEAN statements.

Featured in: Example 7 on page 134

MEAN *variable(s)* </ FORMAT=*format-name*>;

Required Arguments

variable(s)
numeric variable for which mean values are calculated for each month.

Restriction: This variable must be in the activities data set.

Options

FORMAT=*format-name*
names a SAS or user-defined format to be used in displaying the means requested.

Alias: F=

Default: BEST. format

Featured in: Example 7 on page 134

What Is Displayed and How

- The means appear at the bottom of the summary calendar page, if there is room; otherwise they appear on the following page.
- The means appear in the LEGEND box if you specify the LEGEND option.
- PROC CALENDAR automatically displays variables named in a MEAN statement in the calendar output, even if the variables are not named in the VAR statement.

OUTDUR Statement

Specifies in days the length of the week to be displayed.

Alias: OUTDURATION

Requirement: The OUTSTART statement is required.

OUTDUR *number-of-days*;

Required Arguments

number-of-days
an integer expressing the length in days of the week to be displayed.

Length of Week

Use either the OUTDUR or OUTFIN statement to supply the procedure with information about the length of the week to display. If you use both, PROC CALENDAR ignores the OUTDUR statement.

OUTFIN Statement

Specifies the last day of the week to display in the calendar.

Alias: OUTFINISH

Requirement: The OUTSTART statement is required.

Featured in: Example 3 on page 115 and Example 8 on page 138

OUTFIN *day-of-week*;

Required Arguments

day-of-week
the name of the last day of the week to display. For example,

```
outfin friday;
```

Length of Week

Use either the OUTFIN or OUTDUR statement to supply the procedure with information about the length of the week to display. If you use both, PROC CALENDAR uses only the OUTFIN statement.

OUTSTART Statement

Specifies the starting day of the week to display in the calendar.

Alias: OUTSTA

Default: If you do not use OUTSTART, each calendar week begins with Sunday.

Featured in: Example 3 on page 115 and Example 8 on page 138

OUTSTART *day-of-week*;

Required Arguments

day-of-week
the name of the starting day of the week for each week in the calendar. For example,

```
outstart monday;
```

Interaction with OUTDUR and OUTFIN

By default, a calendar displays all seven days in a week. Use OUTDUR or OUTFIN, in conjunction with OUTSTART, to control how many days are displayed and which day starts the week.

START Statement

Specifies the variable in the activities data set that contains the starting date of each activity.

Alias: STA, DATE, ID

Required: START is required for both summary and schedule calendars.

Featured in: All examples

START *variable*;

Required Arguments

variable
contains the starting date of each activity.

Restriction: This variable must be in the activities data set.

Restriction: Values of *variable* must be in either SAS date or datetime values.

Restriction: If you use datetime values, specify the DATETIME option in the PROC CALENDAR statement.

Restriction: Both the START and FIN variables must have matching formats. For example, if one contains datetime values, so must the other.

SUM Statement

Specifies numeric variables in the activities data set to total for each month.

Calendar type: Summary

Tip: To apply different formats to variables being summed, use multiple SUM statements.

Featured in: Example 7 on page 134 and Example 8 on page 138

SUM *variable(s)* </ FORMAT=*format-name*>;

Required Arguments

variable(s)
specifies one or more numeric variables to total for each month.

Restriction: This variable must be in the activities data set.

Options

FORMAT=*format-name*
names a SAS or user-defined format to use in displaying the sums requested.

Alias: F=

Default: BEST. format

Featured in: Example 7 on page 134 and Example 8 on page 138

What Is Displayed and How

- The sum appears at the bottom of the calendar page, if there is room; otherwise, it appears on the following page.
- The sum appears in the LEGEND box if you specify the LEGEND option.
- PROC CALENDAR automatically displays variables named in a SUM statement in the calendar output, even if the variables are not named in the VAR statement.

VAR Statement

Specifies the variables that you want to display for each activity.

Alias: VARIABLE

VAR *variable(s)*;

Required Arguments

variable(s)
specifies one or more variables that you want to display in the calendar.

Range: The values of *variable* can be either character or numeric.

Restriction: These variables must be in the activities data set.

Tip: You can apply a format to this variable.

Details

When VAR Is Not Used

If you do not use a VAR statement, the procedure displays all variables in the activities data set in the order that they occur in the data set, except for the BY, CALID, START, DUR, and FIN variables. All variables are not displayed, however, if the LINESIZE= and PAGESIZE= settings do not allow enough space in the calendar.

Display of Variables

- PROC CALENDAR displays variables in the order that they appear in the VAR statement. All variables are not displayed, however, if the LINESIZE= and PAGESIZE= settings do not allow enough space in the calendar.
- PROC CALENDAR also displays any variable named in a SUM or MEAN statement for each activity in the calendar output, even if you do not name that variable in a VAR statement.

Concepts: CALENDAR Procedure

Type of Calendars

PROC CALENDAR can produce two kinds of calendars: schedule and summary.

Use a ...	if you want to ...	and can accept this restriction
schedule calendar	schedule activities around holidays and nonwork periods	cannot calculate sums and means
	schedule activities that last more than one day	
summary calendar	calculate sums and means	activities can last only one day

Note: PROC CALENDAR produces a summary calendar if you do not use a DUR or FIN statement in the PROC step. △

Schedule Calendar

Definition

A report in calendar format that shows when activities and holidays start and end.

Required Statements

You must supply a START statement and either a DUR or FIN statement.

Use this statement . . .	to specify a variable whose value indicates the . . .
START	starting date of an activity
DUR*	duration of an activity
FIN*	ending date of an activity

* Choose one of these. If you do not use a DUR or FIN statement CALENDAR assumes you want to create a summary calendar report.

Examples

See "Simple Schedule Calendar — 7-Day Default Calendar" on page 75, "Advanced Schedule Calendar" on page 76, as well as Example 1 on page 108, Example 2 on page 112, Example 3 on page 115, Example 4 on page 120, Example 5 on page 125, and Example 6 on page 128

Summary Calendar

Definition

A report in calendar format that displays activities and holidays that last only one day and that can provide summary information in the form of sums and means.

Required Statements

You must supply a START statement. This statement identifies the variable in the activities data set that contains an activity's starting date.

Multiple Events on a Single Day

A summary calendar report can display only one activity on a given date. If more than one activity has the same START value, therefore, only the last observation that was read is used. In such situations, you may find PROC SUMMARY useful in collapsing your data set to contain one activity per starting date.

Examples

See "Simple Summary Calendar" on page 78, Example 7 on page 134, and Example 8 on page 138

The Default Calendars

Description

PROC CALENDAR provides two default calendars for simple applications. You can produce calendars without having to specify detailed workshifts and weekly work patterns if your application can use one of two simple work patterns. Consider using a default calendar if

- □ your application uses a 5-day work week with 8-hour days or a 7-day work week with 24-hour days. See Table 5.2 on page 99.

- you want to print all activities on the same calendar.
- you do not need to identify separate calendars.

Table 5.2 Default Calendar Settings and Examples

If scheduled work days are	Then set INTERVAL=	By default DAYLENGTH=	So work periods are	Shown in Example
7 (M-Sun)	DAY	24	24-hour days	2
5 (M-F)	WORKDAY	8	8-hour days	1

When You Unexpectedly Produce a Default Calendar

If you want to produce a specialized calendar, but do not provide all the necessary information, PROC CALENDAR attempts to produce a default calendar. These errors cause PROC CALENDAR to produce a calendar with default features:

- If the activities data set does not contain a CALID variable, then PROC CALENDAR produces a default calendar.
- If *both* the holidays and calendar data sets do not contain a CALID variable, then PROC CALENDAR produces a default calendar *even if the activities data set contains a CALID variable*.
- If the activities and calendar data sets contain the CALID variable, but the holidays data set does not, then the default holidays are used.

Examples

See the 7-day default calendar in Output 5.1 on page 75 and the 5-day default calendar in Example 1 on page 108

Calendars and Multiple Calendars

Definitions

calendar
: a logical entity that represents a weekly work pattern, which consists of weekly work schedules and daily shifts. PROC CALENDAR contains two default work patterns: 5-day week with an 8-hour day or a 7-day week with a 24-hour day. You can also define your own work patterns using CALENDAR and WORKDAYS data sets.

calendar report
: a report in calendar format that displays activities, holidays, and nonwork periods. A calendar report can contain multiple calendars in one of three formats

  separate
  : Each identified calendar prints on separate output pages.

  combined
  : All identified calendars print on the same output pages and each is identified.

  mixed

All identified calendars print on the same output pages but are not identified as belonging to separate calendars.

multiple calendar
: a logical entity that represents multiple weekly work patterns.

Why Create Multiple Calendars

Create a multiple calendar if you want to print a calendar report that shows activities that follow different work schedules or different weekly work patterns. For example, a construction project report might need to use different work schedules and weekly work patterns for work crews on different parts of the project.

Another use for multiple calendars is to identify activities so that you can choose to print them in the same calendar report. For example, if you identify activities as belonging to separate departments within a division, you can choose to print a calendar report that shows all departmental activities on the same calendar.

And finally, using multiple calendars, you can produce separate calendar reports for each calendar in a single step. For example, if activities are identified by department, you can produce a calendar report that prints the activities of each department on separate pages.

How to Identify Multiple Calendars

Because PROC CALENDAR can process only one data set of each type (activities, holidays, calendar, workdays) in a single PROC step, you must be able to identify for PROC CALENDAR which calendar an activity, holiday, or weekly work pattern belongs to. Use the CALID statement to specify the variable whose values identify the appropriate calendar. This variable can be numeric or character.

You can use the special variable name _CAL_ or you can use another variable name. PROC CALENDAR automatically looks for a variable named _CAL_ in the holiday and calendar data sets, even when the activities data set uses a variable with another name as the CALID variable. Therefore, if you use the name _CAL_, at least in your holiday and calendar data sets, you can more easily reuse these data sets for different calendar applications.

Using Holidays or Calendar Data Sets with Multiple Calendars

When using a holidays or calendar data set with multiple calendars, PROC CALENDAR treats the variable values in the following way:

- Every value of the CALID variable that appears in either the holidays or calendar data sets defines a calendar.
- If a CALID value appears in the HOLIDATA= data set but not in the CALEDATA= data set, the work schedule of the default calendar is used.
- If a CALID value appears in the CALEDATA= data set but not in the HOLIDATA= data set, the holidays of the default calendar are used.
- If a CALID value does not appear in either the HOLIDATA= or CALEDATA= data set, the work schedule and holidays of the default calendar are used.
- If the CALID variable is not found in the holiday or calendar data sets, PROC CALENDAR looks for the default variable _CAL_ instead. If neither the CALID variable nor a _CAL_ variable appears in a data set, the observations in that data set are applied to a default calendar.

Types of Reports That Contain Multiple Calendars

Because you can associate different observations with different calendars, you can print a calendar report that shows activities that follow different work schedules or different work shifts or that contain different holidays. You can

- print separate calendars on the same page and identify each one.
- print separate calendars on the same page without identifying them.
- print separate pages for each identified calendar.

As an example, consider a calendar that shows the activities of all departments within a division. Each department can have its own calendar identification value and, if necessary, can have individual weekly work patterns, daily work shifts, and holidays.

If you place activities associated with different calendars in the same activities data sets, you use PROC CALENDAR to produce calendar reports that print

- the schedule and events for each department on a separate pages (separate output)
- the schedule and events for the entire division, each identified by department (combined output)
- the schedule and events for the entire division, but *not* identified by department (mixed output).

The multiple-calendar feature was added specifically to enable PROC CALENDAR to process the output of PROC CPM in SAS/OR software, a project management tool. See Example 6 on page 128.

How to Identify Calendars with the CALID Statement and the Special Variable _CAL_

To identify multiple calendars, you must use the CALID statement to specify the variable whose values identify which calendar an event belongs with. This variable can be numeric or character.

You can use the special variable name _CAL_ or you can use another variable name. PROC CALENDAR automatically looks for a variable named _CAL_ in the holiday and calendar data sets, even when the activities data set uses a variable with another name as the CALID variable. Therefore, if you use the name _CAL_, at least in your holiday and calendar data sets, you can more easily reuse these data sets for different calendar applications.

When You Use Holidays or Calendar Data Sets

When you use a holidays or calendar data set with multiple calendars, PROC CALENDAR treats the variable values in the following way:

- Every value of the CALID variable that appears in either the holidays or calendar data sets defines a calendar.
- If a CALID value appears in the HOLIDATA= data set but not in the CALEDATA= data set, the work schedule of the default calendar is used.
- If a CALID value appears in the CALEDATA= data set but not in the HOLIDATA= data set, the holidays of the default calendar are used.
- If a CALID value does not appear in either the HOLIDATA= or CALEDATA= data set, the work schedule and holidays of the default calendar are used.
- If the CALID variable is not found in the holiday or calendar data sets, PROC CALENDAR looks for the default variable _CAL_ instead. If neither the CALID variable nor a _CAL_ variable appear in a data set, the observations in that data set are applied to a default calendar.

Examples

Example 2 on page 112, Example 3 on page 115, Example 4 on page 120, and Example 8 on page 138

Input Data Sets

You may need several data sets to produce a calendar, depending on the complexity of your application. PROC CALENDAR can process one of each of four data sets. See Table 5.3 on page 102.

Table 5.3 Four Possible Input Data Sets for PROC CALENDAR

Data Set	Description	Specify with the . . .
activities	Each *observation* contains information about a single activity.	DATA= option
holidays	Each *observation* contains information about a holiday	HOLIDATA= option
calendar	Each *observation* defines one weekly work schedule.	CALEDATA= option
workdays	Each *variable* represents one daily schedule of alternating work and nonwork periods.	WORKDATA= option

Activities Data Set

Purpose

The activities data set, specified with the DATA= option, contains information about the activities to be scheduled by PROC CALENDAR. Each observation describes a single activity.

Requirements and Restrictions

- An activities data set is required. (If you do not specify one with the DATA= option, PROC CALENDAR uses the _LAST_ data set.)
- Only one activities data set is allowed.
- The activities data set must always be sorted or indexed by the START variable.
- If you use a CALID (calendar identifier) variable and want to produce output that shows multiple calendars on separate pages, the activities data set must be sorted by or indexed on the CALID variable and then by the START variable.
- If you use a BY statement, the activities data set must be sorted by or indexed on the BY variables.

Structure

Each observation in the activities data set contains information about one activity. One variable must contain the starting date. If you are producing a schedule calendar, another variable must contain either the activity duration or finishing date. Other variables can contain additional information about an activity.

If a variable contains an activity's . . .	Specify it with the . . .	For this type of calendar. . .
starting date	START statement	Schedule Summary
duration	DUR statement	Schedule
finishing date	FIN statement	Schedule

Multiple Activities per Day in Summary Calendars

A summary calendar can display only one activity on a given date. If more than one activity has the same START value, therefore, only the last observation read is used. In such situations, you may find PROC SUMMARY useful to collapse your data set to contain one activity per starting date.

Examples

Every example in the Examples section uses an activities data set.

Holidays Data Set

Purpose

You can use a holidays data set, specified with the HOLIDATA= option, to

- □ identify holidays on your calendar output
- □ identify days that are not available for scheduling work. (In a schedule calendar, PROC CALENDAR does not schedule activities on these days.)

Structure

Each observation in the holidays data set must contain at least the holiday starting date. A holiday lasts only one day unless a duration or finishing date is specified. Supplying a holiday name is recommended, though not required. If you do not specify which variable contains the holiday name, PROC CALENDAR uses the word **`DATE`** to identify each holiday.

If a variable contains a holiday's . . .	Then specify it with this statement . . .
starting date	HOLISTART
name	HOLIVAR
duration	HOLIDUR
finishing date	HOLIFIN

No Sorting Needed

You do not need to sort or index the holidays data set.

Using SAS Date Versus SAS Datetime Values

PROC CALENDAR calculates time using SAS datetime values. Even when your data are in DATE. format, the procedure automatically calculates time in minutes and seconds. If you specify only date values, therefore, PROC CALENDAR prints messages similar to the following ones to the SAS log:

```
NOTE: All holidays are assumed to start at the
      time/date specified for the holiday variable
      and last one DTWRKDAY.
WARNING: The units of calculation are SAS datetime
         values while all the holiday variables are
         not. All holidays are converted to SAS
         datetime values.
```

Create a Generic Holidays Data Set

If you have many applications that require PROC CALENDAR output, consider creating a generic holidays data set that contains standard holidays. You can begin with the generic holidays and add observations that contain holidays or nonwork events specific to an application.

CAUTION:

Do not schedule holidays during nonwork periods. Holidays defined in the HOLIDATA= data set cannot occur during nonwork periods defined in the work schedule. For example, you cannot schedule Sunday as a vacation day if the work week is defined as Monday through Friday. When such a conflict occurs, the holiday is *rescheduled to the next available working period* following the nonwork day. △

Examples

Every example in the Examples section uses a holidays data set.

Calendar Data Set

Purpose

You can use a calendar data set, specified with the CALEDATA= option, to specify work schedules for different calendars.

Structure

Each observation in the calendar data set defines one weekly work schedule. The data set created in the DATA step shown below defines weekly work schedules for two calendars, CALONE and CALTWO.

```
data cale;
   input _sun_ $ _mon_ $ _tue_ $ _wed_ $ _thu_ $ /
         _fri_ $ _sat_ $ _cal_ $ d_length time6.;
   datalines;
holiday workday workday workday workday
```

```
workday holiday calone 8:00
holiday shift1 shift1 shift1 shift1
shift2 holiday caltwo 9:00
;
```

The variables in this calendar data set consist of

SUN through _SAT_
: the name of each day of the week that appears in the calendar. The values of these variables contain the name of workshifts. Valid values for workshifts are

- **WORKDAY** (the default workshift)
- **HOLIDAY** (a nonwork period)
- names of variables in the WORKDATA= data set (in this example, **SHIFT1** and **SHIFT2**).

CAL
: the CALID (calendar identifier) variable. The values of this variable identify different calendars. If this variable is not present, the first observation in this data set defines the work schedule that is applied to all calendars in the activities data set.

If the CALID variable contains a missing value, the character or numeric value for the default calendar (**DEFAULT** or 0) is used. See “The Default Calendars” on page 98 for further details.

D_LENGTH
: the daylength identifier variable. Values of D_LENGTH indicate the length of the standard workday to be used in calendar calculations. You can set the workday length either by placing this variable in your calendar data set or by using the DAYLENGTH= option.

Missing values for this variable default to the number of hours specified in the DAYLENGTH= option; if the DAYLENGTH= option is not used, the day length defaults to 24 hours if INTERVAL=DAY, or 8 hours if INTERVAL=WORKDAY.

Using Default Workshifts Instead of a Workdays Data Set

You can use a calendar data set with or without a workdays data set. Without a workdays data set, WORKDAY in the calendar data set is equal to one of two standard workdays, depending on the setting of the INTERVAL= option:

If INTERVAL=	Then the work-shift begins at . . .	And the day length is . . .
DAY	00:00	24 hours
WORKDAY	9:00	8 hours

You can reset the length of the standard workday with the DAYLENGTH= option or a D_LENGTH variable in the calendar data set. You can define other work shifts in a workdays data set.

Examples

Example 3 on page 115, Example 4 on page 120, and Example 7 on page 134 feature a calendar data set.

Workdays Data Set

Purpose

You can use a workdays data set, specified with the WORKDATA= option, to define the daily workshifts named in a CALEDATA= data set.

Use Default Work Shifts or Create Your Own?

You do not need a workdays data set if your application can use one of two default work shifts:

If INTERVAL=	Then the work-shift begins at . . .	And the day length is. . .
DAY	00:00	24 hours
WORKDAY	9:00	8 hours

See the INTERVAL= option on page 85.

Structure

Each *variable* in the workdays data set contains one daily schedule of alternating work and nonwork periods. For example, this DATA step creates a data set that contains specifications for two work shifts:

```
data work;
   input shift1 time6. shift2 time6.;
   datalines;
7:00  7:00
12:00 11:00
13:00   .
17:00   .
;
```

The variable SHIFT1 specifies a 10-hour workday, with one nonwork period (a lunch hour); the variable SHIFT2 specifies a 4-hour workday with no nonwork periods.

How Missing Values Are Treated

The missing values default to 00:00 in the first observation and to 24:00 in all other observations. Two consecutive values of 24:00 define a zero-length time period, which is ignored.

Examples

See Example 3 on page 115

Missing Values in Input Data Sets

Table 5.4 on page 107 summarizes the treatment of missing values for variables in the data sets used by PROC CALENDAR.

Table 5.4 Treatment of Missing Values in PROC CALENDAR

Data set	Variable	Treatment of missing values
Activities (DATA=)	CALID	default calendar value is used
	START	observation is not used
	DUR	1.0 is used
	FIN	START value + daylength is used
	VAR	if a summary calendar or the MISSING option is specified, the missing value is used; otherwise, no value is used
	SUM, MEAN	0
Calendar (CALEDATA=)	CALID	default calendar value is used
	SUN *through* _SAT_	corresponding shift for default calendar is used
	D_LENGTH	if available, DAYLENGTH= value is used; otherwise, if INTERVAL=DAY, 24:00 is used; otherwise 8:00 is used
	SUM, MEAN	0
Holiday (HOLIDATA=)	CALID	all holidays apply to all calendars
	HOLISTART	observation is not used
	HOLIDUR	if available, HOLIFIN value is used instead of HOLIDUR value; otherwise 1.0 is used
	HOLIFIN	if available, HOLIDUR value is used instead of HOLIFIN value; otherwise, HOLISTART value + day length is used
	HOLIVAR	no value is used
Workdays (WORKDATA=)	any	for the first observation, 00:00 is used; otherwise, 24:00 is used

Results: CALENDAR Procedure

What Affects the Quantity of PROC CALENDAR Output

The quantity of printed calendar output depends on

- the range of dates in the activities data set
- whether the FILL option is specified
- the BY statement
- the CALID statement.

PROC CALENDAR always prints one calendar for every month that contains any activities. If you specify the FILL option, the procedure prints every month between the first and last activities, including months that contain no activities. Using the BY statement prints one set of output for each BY value. Using the CALID statement with OUTPUT=SEPARATE prints one set of output for each value of the CALID variable.

How Size Affects the Format of PROC CALENDAR Output

PROC CALENDAR always attempts to fit the calendar within a single page, as defined by the SAS system options PAGESIZE= and LINESIZE=. If the PAGESIZE= and LINESIZE= values do not allow sufficient room, PROC CALENDAR may print the legend box on a separate page. If necessary, PROC CALENDAR truncates or omits values to make the output fit the page and prints messages to that effect in the SAS log.

What Affects the Lines that Show Activity Duration

In a schedule calendar, the duration of an activity is shown by a continuous line through each day of the activity. Values of variables for each activity are printed on the same line, separated by slashes (/). Each activity begins and ends with a plus sign (+). If an activity continues from one week to another, PROC CALENDAR displays arrows (< >) at the points of continuation.

The length of the activity lines depends on the amount of horizontal space available. You can increase this by specifying

- a larger linesize with the LINESIZE= option in the OPTIONS statement
- the WEEKDAYS option to suppress the printing of Saturday and Sunday, which provides more space for Monday through Friday.

Customizing the Calendar Appearance

PROC CALENDAR uses 17 of the 20 SAS formatting characters to construct the outline of the calendar and to print activity lines and to indicate holidays. You can use the FORMCHAR= option to customize the appearance of your PROC CALENDAR output by substituting your own characters for the default. See Table 5.1 on page 83 and Figure 5.1 on page 84.

If your printer supports an *extended character set* (one that includes graphics characters in addition to the regular alphanumeric characters), you can greatly improve the appearance of your output by using the FORMCHAR= option to redefine formatting characters with hexadecimal characters. For information on which hexadecimal codes to use for which characters, consult the documentation for your hardware. For an example of assigning hex values, see FORMCHAR= on page 83.

Examples: CALENDAR Procedure

Example 1: Schedule Calendar with Holidays – 5-Day Default

Procedure features:

PROC CALENDAR statement options:

DATA=
HOLIDATA=
WEEKDAYS

DUR statement

HOLISTART statement

HOLIVAR statement

HOLIDUR statement

START statement

Other features:

PROC SORT statement

BY statement

5-day default calendar

This example

- □ creates a schedule calendar
- □ uses one of the two default work patterns: 8-hour day, 5-day week
- □ schedules activities around holidays
- □ displays a 5-day week

Program

Create the activities data set. ALLACTY contains both personal and business activities information for a bank president.

```
data allacty;
   input date : date7. event $ 9-36 who $ 37-48 long;
   datalines;
01JUL96 Dist. Mtg.                  All          1
17JUL96 Bank Meeting                1st Natl     1
02JUL96 Mgrs. Meeting               District 6   2
11JUL96 Mgrs. Meeting               District 7   2
03JUL96 Interview                   JW           1
08JUL96 Sales Drive                 District 6   5
15JUL96 Sales Drive                 District 7   5
08JUL96 Trade Show                  Knox         3
22JUL96 Inventors Show              Melvin       3
11JUL96 Planning Council            Group II     1
18JUL96 Planning Council            Group III    1
25JUL96 Planning Council            Group IV     1
12JUL96 Seminar                     White        1
19JUL96 Seminar                     White        1
18JUL96 NewsLetter Deadline         All          1
05JUL96 VIP Banquet                 JW           1
19JUL96 Co. Picnic                  All          1
16JUL96 Dentist                     JW           1
24JUL96 Birthday                    Mary         1
```

```
25JUL96 Close Sale                  WYGIX Co.    2
;
```

Create the holidays data set.

```
data hol;
   input date : date7. holiday $ 11-25 holilong @27;
   datalines;
05jul96   Vacation        3
04jul96   Independence    1
;
```

Sort the activities data set by the variable containing the starting date. You are not required to sort the holidays data set.

```
proc sort data=allacty;
   by date;
run;
```

Set LINESIZE= appropriately. If the linesize is not long enough to print the variable values, PROC CALENDAR either truncates the values or produces no calendar output.

```
options nodate pageno=1 linesize=132 pagesize=60;
```

Create the schedule calendar. DATA= identifies the activities data set; HOLIDATA= identifies the holidays data set. WEEKDAYS specifies that a week consists of five eight-hour work days.

```
proc calendar data=allacty holidata=hol weekdays;
```

The START statement specifies the variable in the activities data set that contains the starting date of the activities; DUR specifies the variable that contains the duration of each activity. Creating a schedule calendar requires START and DUR.

```
   start date;
   dur long;
```

The HOLISTART, HOLIVAR, and HOLIDUR statements specify the variables in the holidays data set that contain the start date, name, and duration of each holiday, respectively. When you use a holidays data set, HOLISTART is required. Because at least one holiday lasts more than one day, HOLIDUR is required.

```
   holistart date;
   holivar holiday;
   holidur holilong;
   title1 'Summer Planning Calendar:  Julia Cho';
   title2 'President, Community Bank';
run;
```

Output

Output 5.4 Schedule Calendar: 5-Day Week with Holidays

```
                                 Summer Planning Calendar:  Julia Cho                                             1
                                      President, Community Bank

-----------------------------------------------------------------------------------------------------------------------------------
|                                                                                                                                 |
|                                                           July  1996                                                            |
|                                                                                                                                 |
|---------------------------------------------------------------------------------------------------------------------------------|
|         Monday          |         Tuesday         |        Wednesday        |        Thursday         |         Friday          |
|-------------------------+-------------------------+-------------------------+-------------------------+-------------------------|
|            1            |            2            |            3            |            4            |            5            |
|                         |                         |                         |******Independence*******|********Vacation*********|
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |+=====Interview/JW======+|                         |                         |
|+====Dist. Mtg./All=====+|+============Mgrs. Meeting/District 6=============+|                         |                         |
|-------------------------+-------------------------+-------------------------+-------------------------+-------------------------|
|            8            |            9            |           10            |           11            |           12            |
|********Vacation*********|********Vacation*********|                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |+Planning Council/Group +|+=====Seminar/White=====+|
|                         |                         |+=============================Trade Show/Knox==============================+|
|                         |                         |+=========================Sales Drive/District 6============================>|
|                         |                         |+====VIP Banquet/JW=====+|+============Mgrs. Meeting/District 7=============+|
|-------------------------+-------------------------+-------------------------+-------------------------+-------------------------|
|           15            |           16            |           17            |           18            |           19            |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |+======Dentist/JW=======+|                         |+NewsLetter Deadline/All+|+====Co. Picnic/All=====+|
|+====================================================Sales Drive/District 7=====================================================+|
|<=============Sales Drive/District 6==============+|+=Bank Meeting/1st Natl=+|+Planning Council/Group +|+=====Seminar/White=====+|
|-------------------------+-------------------------+-------------------------+-------------------------+-------------------------|
|           22            |           23            |           24            |           25            |           26            |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |+=====Birthday/Mary=====+|+==============Close Sale/WYGIX Co.===============+|
|+==========================Inventors Show/Melvin==========================+|+Planning Council/Group +|                         |
|-------------------------+-------------------------+-------------------------+-------------------------+-------------------------|
|           29            |           30            |           31            |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
|                         |                         |                         |                         |                         |
-----------------------------------------------------------------------------------------------------------------------------------
```

Example 2: Schedule Calendar Containing Multiple Calendars

Procedure features:

CALID statement:

CAL variable
OUTPUT=COMBINE option

DUR statement

24-hour day, 7-day week

This example builds on Example 1 by identifying activities as belonging to one of two calendars, business or personal. This example

- produces a schedule calendar report
- prints two calendars on the same output page
- schedules activities around holidays
- uses one of the two default work patterns: 24-hour day, 7-day week
- identifies activities and holidays by calendar name.

Program

Create the activities data set and identify separate calendars. ALLACTY2 contains both personal and business activities for a bank president. The _CAL_ variable identifies which calendar an event belongs to.

```
data allacty2;
  input date:date7. happen $ 10-34 who $ 35-47 _CAL_ $ long;
  datalines;
01JUL96  Dist. Mtg.               All           CAL1   1
02JUL96  Mgrs. Meeting            District 6    CAL1   2
03JUL96  Interview                JW            CAL1   1
05JUL96  VIP Banquet              JW            CAL1   1
06JUL96  Beach trip               family        CAL2   2
08JUL96  Sales Drive              District 6    CAL1   5
08JUL96  Trade Show               Knox          CAL1   3
09JUL96  Orthodontist             Meagan        CAL2   1
11JUL96  Mgrs. Meeting            District 7    CAL1   2
11JUL96  Planning Council         Group II      CAL1   1
12JUL96  Seminar                  White         CAL1   1
14JUL96  Co. Picnic               All           CAL1   1
14JUL96  Business trip            Fred          CAL2   2
15JUL96  Sales Drive              District 7    CAL1   5
16JUL96  Dentist                  JW            CAL1   1
17JUL96  Bank Meeting             1st Natl      CAL1   1
17JUL96  Real estate agent        Family        CAL2   1
18JUL96  NewsLetter Deadline      All           CAL1   1
18JUL96  Planning Council         Group III     CAL1   1
19JUL96  Seminar                  White         CAL1   1
22JUL96  Inventors Show           Melvin        CAL1   3
24JUL96  Birthday                 Mary          CAL1   1
```

```
25JUL96  Planning Council         Group IV     CAL1   1
25JUL96  Close Sale               WYGIX Co.    CAL1   2
27JUL96  Ballgame                 Family       CAL2   1
;
```

Create the holidays data set and identify which calendar a holiday affects. The _CAL_ variable identifies which calendar a holiday belongs to.

```
data vac;
   input hdate:date7.  holiday $ 11-25 _CAL_ $ ;
   datalines;
29JUL96   vacation                CAL2
04JUL96   Independence            CAL1
;
```

Sort the activities data set by the variable containing the starting date. When creating a calendar with combined output, you sort only by the activity starting date, not by the CALID variable. You are not required to sort the holidays data set.

```
proc sort data=allacty2;
   by date;
run;
```

Set LINESIZE= appropriately. If the linesize is not long enough to print the variable values, PROC CALENDAR either truncates the values or produces no calendar output.

```
options nodate pageno=1 pagesize=60 linesize=132;
```

Create the schedule calendar. DATA= identifies the activities data set; HOLIDATA= identifies the holidays data set. By default, the output calendar displays a 7-day week.

```
proc calendar data=allacty2 holidata=vac;
```

The CALID statement specifies the variable that identifies which calendar an event belongs to. OUTPUT=COMBINE places all events and holidays on the same calendar.

```
   calid _CAL_ / output=combine;
```

Schedule an activity. The START statement specifies the variable in the activities data set that contains the starting date of the activities; DUR specifies the variable that contains the duration of each activity. Creating a schedule calendar requires START and DUR.

```
   start date ;
   dur long;
```

The HOLISTART and HOLIVAR statements specify the variables in the holidays data set that contain the start date and name of each holiday, respectively. HOLISTART is required when you use a holidays data set.

```
   holistart hdate;
   holivar holiday;
   title1 'Summer Planning Calendar:  Julia Cho';
   title2 'President, Community Bank';
   title3 'Work and Home Schedule';
run;
```

Output

Output 5.5 Schedule Calendar Containing Multiple Calendars

```
                                   Summer Planning Calendar:  Julia Cho                                    1
                                       President, Community Bank
                                         Work and Home Schedule

          --------------------------------------------------------------------------------------------------------------------
          |                                                                                                                  |
          |                                                    July  1996                                                    |
          |                                                                                                                  |
          |------------------------------------------------------------------------------------------------------------------|
          |     Sunday     |     Monday     |    Tuesday     |   Wednesday    |    Thursday    |     Friday     |    Saturday    |
----------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |                |       1        |       2        |       3        |       4        |       5        |       6        |
|.........|................|................|................|................|................|................|................|
| CAL2    |                |                |                |                |                |                |+Beach trip/fam>|
|.........|................|................|................|................|................|................|................|
| CAL1    |                |                |                |+=Interview/JW=+|**Independence**|                |                |
|         |                |+Dist. Mtg./All+|+===Mgrs. Meeting/District 6====+|                |+VIP Banquet/JW+|                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       7        |       8        |       9        |       10       |       11       |       12       |       13       |
|.........|................|................|................|................|................|................|................|
| CAL2    |<Beach trip/fam+|                |+Orthodontist/M+|                |                |                |                |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |                |                |                |+Planning Counc+|+Seminar/White=+|                |
|         |                |+================Trade Show/Knox=================+|+===Mgrs. Meeting/District 7====+|                |
|         |                |+==============================Sales Drive/District 6==============================+|                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       14       |       15       |       16       |       17       |       18       |       19       |       20       |
|.........|................|................|................|................|................|................|................|
| CAL2    |+======Business trip/Fred=======+|                |+Real estate ag+|                |                |                |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |                |                |                |+Planning Counc+|                |                |
|         |                |                |+==Dentist/JW==+|+Bank Meeting/1+|+NewsLetter Dea+|+Seminar/White=+|                |
|         |+Co. Picnic/All+|+==============================Sales Drive/District 7==============================+|                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       21       |       22       |       23       |       24       |       25       |       26       |       27       |
|.........|................|................|................|................|................|................|................|
| CAL2    |                |                |                |                |                |                |+Ballgame/Famil+|
|.........|................|................|................|................|................|................|................|
| CAL1    |                |                |                |+Birthday/Mary=+|+=====Close Sale/WYGIX Co.======+|                |
|         |                |+=============Inventors Show/Melvin==============+|+Planning Counc+|                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       28       |       29       |       30       |       31       |                |                |                |
|.........|................|................|................|................|................|................|................|
| CAL2    |                |****vacation****|                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
----------------------------------------------------------------------------------------------------------------------------------
```

Example 3: Multiple Schedule Calendars with Atypical Workshifts (Separated Output)

Procedure features:

PROC CALENDAR statement options:

CALEDATA=
DATETIME

WORKDATA=
CALID statement:
CAL variable
OUTPUT=SEPARATE option
DUR statement
OUTSTART statement
OUTFIN statement

This example

- produces separate output pages for each calendar in a single PROC step
- schedules activities around holidays
- displays an 8-hour day, 5 1/2-day week
- uses separate work patterns and holidays for each calendar.

Producing Different Output for Multiple Calendars

This example and Example 4 on page 120 use the same input data for multiple calendars to produce different output. The only differences in these programs are how the activities data set is sorted and how the OUTPUT= option is set.

To print . . .	Sort the activities data set by . . .	And set OUTPUT= to	See Example
Separate pages for each calendar	calendar id and starting date	SEPARATE	3, 8
All activities on the same page and identify each calendar	starting date	COMBINE	4, 2
All activities on the same page and NOT identify each calendar	starting date	MIX	4

Program

Specify a library so that you can permanently store the activities data set.

```
libname well 'SAS-data-library';
```

Create the activities data set and identify separate calendars. WELL.ACT is a permanent SAS data set that contains activities for a well construction project. The _CAL_ variable identifies the calendar that an activity belongs to.

```
data well.act;
   input task & $16. dur : 5. date : datetime16.  _cal_ $ cost;
   datalines;
```

```
Drill Well          3.50  01JUL96:12:00:00  CAL1   1000
Lay Power Line      3.00  04JUL96:12:00:00  CAL1   2000
Assemble Tank       4.00  05JUL96:08:00:00  CAL1   1000
Build Pump House    3.00  08JUL96:12:00:00  CAL1   2000
Pour Foundation     4.00  11JUL96:08:00:00  CAL1   1500
Install Pump        4.00  15JUL96:14:00:00  CAL1    500
Install Pipe        2.00  19JUL96:08:00:00  CAL1   1000
Erect Tower         6.00  20JUL96:08:00:00  CAL1   2500
Deliver Material    2.00  01JUL96:12:00:00  CAL2    500
Excavate            4.75  03JUL96:08:00:00  CAL2   3500
;
```

Create the holidays data set. The _CAL_ variable identifies the calendar that a holiday belongs to.

```
data well.hol;
   input date date. holiday $ 11-25 _cal_ $;
   datalines;
09JUL96   Vacation            CAL2
04JUL96   Independence        CAL1
;
```

Create the calendar data set. Each observation defines the workshifts for an entire week. The _CAL_ variable identifies to which calendar the workshifts apply. CAL1 uses the default 8-hour workshifts for Monday through Friday. CAL2 uses a half day on Saturday and the default 8-hour workshift for Monday through Friday.

```
data well.cal;
   input _sun_ $ _sat_ $ _mon_ $ _tue_ $ _wed_ $ _thu_ $
         _fri_ $ _cal_ $;
   datalines;
Holiday Holiday  Workday Workday Workday Workday Workday CAL1
Holiday Halfday  Workday Workday Workday Workday Workday CAL2
;
```

Create the workdays data set. This data set defines the daily workshifts that are named in the calendar data set. Each variable – not observation – contains one daily schedule of alternating work and nonwork periods. The HALFDAY workshift lasts 4 hours.

```
data well.wor;
   input halfday time5.;
   datalines;
08:00
12:00
;
```

Sort the activities data set by the variables containing the calendar identification and the starting date, respectively. You are not required to sort the holidays data set.

```
proc sort data=well.act;
   by _cal_ date;
run;
```

Set LINESIZE= appropriately. If the linesize is not long enough to print the variable values, PROC CALENDAR either truncates the values or produces no calendar output.

```
options nodate pageno=1 linesize=132 pagesize=60;
```

Create the schedule calendar. DATA= identifies the activities data set; HOLIDATA= identifies the holidays data set; CALEDATA= identifies the calendar data set; WORKDATA= identifies the workdays data set. DATETIME specifies that the variable specified with the START statement contains values in SAS datetime format.

```
proc calendar data=well.act
              holidata=well.hol
              caledata=well.cal
              workdata=well.wor
              datetime;
```

The CALID statement specifies that the _CAL_ variable identifies calendars. OUTPUT=SEPARATE prints information for each calendar on separate pages.

```
   calid _cal_ / output=separate;
```

The START statement specifies the variable in the activities data set that contains the activity starting date; DUR specifies the variable that contains the activity duration. START and DUR are required for a schedule calendar.

```
   start date;
   dur dur;
```

HOLISTART and HOLIVAR specify the variables in the holidays data set that contain the start date and name of each holiday, respectively. HOLISTART is required when you use a holidays data set.

```
   holistart date;
   holivar holiday;
```

OUTSTART and OUTFIN specify that the calendar display a 6-day week, Monday through Saturday.

```
   outstart Monday;
   outfin Saturday;
   title1 'Well Drilling Work Schedule: Separate Calendars';
   format cost dollar9.2;
run;
```

Output

Output 5.6 Separate Output for Multiple Schedule Calendars

```
                                    Well Drilling Work Schedule: Separate Calendars                                     1

............................................................ _cal_=CAL1 ...........................................................

 -------------------------------------------------------------------------------------------------------------------------------
 |                                                                                                                             |
 |                                                          July  1996                                                         |
 |                                                                                                                             |
 |-----------------------------------------------------------------------------------------------------------------------------|
 |       Monday       |      Tuesday       |     Wednesday      |      Thursday      |       Friday       |      Saturday      |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |         1          |         2          |         3          |         4          |         5          |         6          |
 |                    |                    |                    |****Independence****|                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |+Assemble Tank/$1,0>|                    |
 |                    |                    |                    |                    |+Lay Power Line/$2,>|                    |
 |+===================Drill Well/$1,000.00====================>|                    |<Drill Well/$1,000.+|                    |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |         8          |         9          |         10         |         11         |         12         |         13         |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |+==========================Build Pump House/$2,000.00===========================+|                    |                    |
 |<============================Assemble Tank/$1,000.00============================+|                    |                    |
 |<=======Lay Power Line/$2,000.00========+|                    |+=======Pour Foundation/$1,500.00=======>|                    |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |         15         |         16         |         17         |         18         |         19         |         20         |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |+========================================Install Pump/$500.00========================================+|                    |
 |<=================Pour Foundation/$1,500.00==================+|                    |+Install Pipe/$1,00>|                    |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |         22         |         23         |         24         |         25         |         26         |         27         |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |+=======================================Erect Tower/$2,500.00=======================================>|                    |
 |<========Install Pipe/$1,000.00=========+|                    |                    |                    |                    |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |         29         |         30         |         31         |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |<Erect Tower/$2,500+|                    |                    |                    |                    |                    |
 -------------------------------------------------------------------------------------------------------------------------------
```

```
                              Well Drilling Work Schedule: Separate Calendars                              2

............................................ _cal_=CAL2 ............................................

 ------------------------------------------------------------------------------------------------------------------------------------
 |                                                                                                                                  |
 |                                                            July  1996                                                            |
 |                                                                                                                                  |
 |----------------------------------------------------------------------------------------------------------------------------------|
 |      Monday        |      Tuesday       |     Wednesday      |      Thursday      |       Friday       |      Saturday      |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |         1          |         2          |         3          |         4          |         5          |         6          |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |+==============================Excavate/$3,500.00================================>|
 |+=================Deliver Material/$500.00==================+|                    |                    |                    |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |         8          |         9          |        10          |        11          |        12          |        13          |
 |                    |******Vacation******|                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |<Excavate/$3,500.00>|                    |<Excavate/$3,500.00+|                    |                    |                    |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |        15          |        16          |        17          |        18          |        19          |        20          |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |        22          |        23          |        24          |        25          |        26          |        27          |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
 |        29          |        30          |        31          |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 |                    |                    |                    |                    |                    |                    |
 ------------------------------------------------------------------------------------------------------------------------------------
```

Example 4: Multiple Schedule Calendars with Atypical Workshifts (Combined and Mixed Output)

Procedure features:

- PROC CALENDAR statementoptions:
  - CALEDATA=
  - DATETIME
  - WORKDATA=
- CALID statement:
  - _CAL_ variable

  OUTPUT=COMBINE option
  OUTPUT=MIXED option
DUR statement
OUTSTART statement
OUTFIN statement

Data sets:
There are input data sets on page 116.

This example

- produces a schedule calendar
- schedules activities around holidays
- uses separate work patterns and holidays for each calendar
- uses an 8-hour day, 5 1/2-day work week
- displays and identifies multiple calendars on each calendar page (combined output)
- displays *but does not identify* multiple calendars on each calendar page (mixed output).

Two Programs and Two Pieces of Output

This example creates both combined and mixed output. Producing combined or mixed calendar output requires only one change to a PROC CALENDAR step: the setting of the OUTPUT= option in the CALID statement. Combined output is produced first, then mixed output.

Producing Different Output for Multiple Calendars

This example and Example 3 on page 115 use the same input data for multiple calendars to produce different output. The only differences in these programs are how the activities data set is sorted and how the OUTPUT= option is set.

To print . . .	Sort the activities data set by . . .	And set OUTPUT= to	See Example
Separate pages for each calendar	calendar id and starting date	SEPARATE	3, 8
All activities on the same page and identify each calendar	starting date	COMBINE	4, 2
All activities on the same page and NOT identify each calendar	starting date	MIX	4

Program for Combined Calendars

Specify the SAS data library where the activities data set is stored.

```
libname well 'SAS-data-library';
```

Sort the activities data set by the variable containing the starting date. Do not sort by the CALID variable when producing combined calendar output.

```
proc sort data=well.act;
   by date;
run;
```

Set PAGESIZE= and LINESIZE= appropriately. When you combine calendars, check the value of PAGESIZE= to ensure that there is enough room to print the activities from multiple calendars. If LINESIZE= is too small for the variable values to print, PROC CALENDAR either truncates the values or produces no calendar output.

```
options nodate pageno=1 linesize=132 pagesize=60;
```

Create the schedule calendar. DATA= identifies the activities data set; HOLIDATA= identifies the holidays data set; CALEDATA= identifies the calendar data set; WORKDATA= identifies the workdays data set. DATETIME specifies that the variable specified with the START statement contains values in SAS datetime format.

```
proc calendar data=well.act
              holidata=well.hol
              caledata=well.cal
              workdata=well.wor
              datetime;
   title1 'Well Drilling Work Schedule: Combined Calendars';
   format cost dollar9.2;
```

The CALID statement specifies that the _CAL_ variable identifies the calendars. OUTPUT=COMBINE prints multiple calendars on the same page and identifies each calendar.

```
   calid _cal_ / output=combine;
```

The START statement specifies the variable in the activities data set that contains the starting date of the activities; DUR specifies the variable that contains the duration of each activity. START and DUR are required for a schedule calendar.

```
   start date;
   dur dur;
```

HOLISTART and HOLIVAR specify the variables in the holidays data set that contain the start date and name of each holiday, respectively. HOLISTART is required when you use a holidays data set.

```
   holistart date;
   holivar holiday;
```

```
run;
```

Output for Combined Calendars

Output 5.7 Multiple Schedule Calendars with Atypical Workshifts (Combined Output)

```
                                       Well Drilling Work Schedule: Combined Calendars                                        1

          -------------------------------------------------------------------------------------------------------------------
          |                                                                                                                 |
          |                                                    July  1996                                                   |
          |                                                                                                                 |
          |-----------------------------------------------------------------------------------------------------------------|
          |     Sunday     |     Monday     |    Tuesday     |   Wednesday    |    Thursday    |     Friday     |    Saturday    |
----------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |                |       1        |       2        |       3        |       4        |       5        |       6        |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |                |                |                |**Independence**|+Assemble Tank/>|                |
|         |                |                |                |                |                |+Lay Power Line>|                |
|         |                |+==============Drill Well/$1,000.00==============>|                |<Drill Well/$1,+|                |
|.........|................|................|................|................|................|................|................|
| CAL2    |                |                |                |+=======================Excavate/$3,500.00========================>|
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       7        |       8        |       9        |       10       |       11       |       12       |       13       |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |+===================Build Pump House/$2,000.00====================+|                |                |
|         |                |<=====================Assemble Tank/$1,000.00=====================+|                |                |
|         |                |<===Lay Power Line/$2,000.00====+|                |+===Pour Foundation/$1,500.00===>|                |
|.........|................|................|................|................|................|................|................|
| CAL2    |                |<Excavate/$3,50>|****Vacation****|<Excavate/$3,50+|                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       14       |       15       |       16       |       17       |       18       |       19       |       20       |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |+===============================Install Pump/$500.00===============================+|                |
|         |                |<===========Pour Foundation/$1,500.00============+|                |+Install Pipe/$>|                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       21       |       22       |       23       |       24       |       25       |       26       |       27       |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |+==============================Erect Tower/$2,500.00===============================>|                |
|         |                |<====Install Pipe/$1,000.00=====+|                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       28       |       29       |       30       |       31       |                |                |                |
|.........|................|................|................|................|................|................|................|
| CAL1    |                |<Erect Tower/$2+|                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
----------------------------------------------------------------------------------------------------------------------------------
```

Program for Mixed Calendars

To produce mixed output instead of combined, use the same program and change the setting of the OUTPUT= option to OUTPUT=MIX:

```
proc calendar data=well.act
              holidata=well.hol
              caledata=well.cal
              workdata=well.wor
              datetime;
   calid _cal_ / output=mix;
   start date;
   dur dur;
   holistart date;
   holivar holiday;
   outstart Monday;
   outfin Saturday;
   title1 'Well Drilling Work Schedule: Mixed Calendars';
   format cost dollar9.2;
run;
```

Output for Mixed Calendars

Output 5.8 Multiple Schedule Calendar with Atypical Workshifts (Mixed Output)

```
                                    Well Drilling Work Schedule: Mixed Calendars                                            1

-------------------------------------------------------------------------------------------------------------------------------
|                                                                                                                             |
|                                                         July  1996                                                          |
|                                                                                                                             |
|-----------------------------------------------------------------------------------------------------------------------------|
|       Monday       |      Tuesday       |     Wednesday      |      Thursday      |       Friday       |      Saturday      |
|--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
|         1          |         2          |         3          |         4          |         5          |         6          |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |+Assemble Tank/$1,0>|                    |
|                    |                    |+==============================Excavate/$3,500.00=================================>|
|+=================Deliver Material/$500.00===================+|****Independence****|+Lay Power Line/$2,>|                    |
|+===================Drill Well/$1,000.00=====================>|****Independence****|<Drill Well/$1,000.+|                    |
|--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
|         8          |         9          |         10         |         11         |         12         |         13         |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|+===========================Build Pump House/$2,000.00============================+|                    |                    |
|<=============================Assemble Tank/$1,000.00=============================+|                    |                    |
|<=======Lay Power Line/$2,000.00========+|                    |                    |                    |                    |
|<Excavate/$3,500.00>|******Vacation******|<Excavate/$3,500.00+|+=======Pour Foundation/$1,500.00=======>|                    |
|--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
|         15         |         16         |         17         |         18         |         19         |         20         |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|+=========================================Install Pump/$500.00=========================================+|                    |
|<=================Pour Foundation/$1,500.00==================+|                    |+Install Pipe/$1,00>|                    |
|--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
|         22         |         23         |         24         |         25         |         26         |         27         |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|+========================================Erect Tower/$2,500.00=========================================>|                    |
|<========Install Pipe/$1,000.00=========+|                    |                    |                    |                    |
|--------------------+--------------------+--------------------+--------------------+--------------------+--------------------|
|         29         |         30         |         31         |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|                    |                    |                    |                    |                    |                    |
|<Erect Tower/$2,500+|                    |                    |                    |                    |                    |
-------------------------------------------------------------------------------------------------------------------------------
```

Example 5: Schedule Calendar, Blank or with Holidays

Procedure features:

PROC CALENDAR statement options:

- FILL
- HOLIDATA=
- INTERVAL=WORKDAY

DUR statement
HOLIDUR statement
HOLISTART statement
HOLIVAR statement

This example produces a schedule calendar that displays only holidays. You can use this same code to produce a set of blank calendars by removing the HOLIDATA= option and the HOLISTART, HOLIVAR, and HOLIDUR statements from the PROC CALENDAR step.

Program

Create the activities data set. Specify one activity in the first month and one in the last, each with a duration of 0. PROC CALENDAR does not print activities with zero durations in the output.

```
data acts;
   input sta : date7. act $ 11-30 dur;
   datalines;
01JAN97   Start                 0
31DEC97   Finish                0
;
```

Create the holidays data set.

```
data holidays;
   input sta : date7. act $ 11-30 dur;
   datalines;
01JAN97   New Year's            1
28MAR97   Good Friday           1
30MAY97   Memorial Day          1
04JUL97   Independence Day      1
01SEP97   Labor Day             1
27NOV97   Thanksgiving          2
25DEC97   Christmas Break       5
;
```

Set PAGESIZE= and LINESIZE= appropriately. To create larger boxes for each day in the calendar output, increase the value of PAGESIZE=.

```
options nodate pageno=1 linesize=132 pagesize=30;
```

Create the calendar. DATA= identifies the activities data set; HOLIDATA= identifies the holidays data set. FILL displays all months, even those with no activities. By default, only months with activities appear in the report. INTERVAL=WORKDAY specifies that activities and holidays are measured in 8-hour days and that PROC CALENDAR schedules activities only Monday through Friday.

```
proc calendar data=acts holidata=holidays fill interval=workday;
```

The START statement specifies the variable in the activities data set that contains the starting date of the activities; DUR specifies the variable that contains the duration of each activity. Creating a schedule calendar requires START and DUR.

```
   start sta;
   dur dur;
```

The HOLISTART, HOLIVAR, and HOLIDUR statements specify the variables in the holidays data set that contain the start date, name, and duration of each holiday, respectively. When you use a holidays data set, HOLISTART is required. Because at least one holiday lasts more than one day, HOLIDUR (or HOLIFIN) is required.

```
   holistart sta;
   holivar act;
   holidur dur;
   title1 'Calendar of Holidays Only';
run;
```

Output

Output 5.9 Schedule Calendars with Holidays Only (Partial Output).

Without INTERVAL=WORKDAY, the 5-day Christmas break would be scheduled through the weekend.

```
                                        Calendar of Holidays Only                                          1

 --------------------------------------------------------------------------------------------------------------------
 |                                                                                                                  |
 |                                            January  1997                                                         |
 |                                                                                                                  |
 |------------------------------------------------------------------------------------------------------------------|
 |     Sunday     |     Monday     |    Tuesday     |   Wednesday    |    Thursday    |     Friday     |    Saturday    |
 |----------------+----------------+----------------+----------------+----------------+----------------+----------------|
 |                |                |                |       1        |       2        |       3        |       4        |
 |                |                |                |***New Year's****|               |                |                |
 |----------------+----------------+----------------+----------------+----------------+----------------+----------------|
 |       5        |       6        |       7        |       8        |       9        |       10       |       11       |
 |                |                |                |                |                |                |                |
 |----------------+----------------+----------------+----------------+----------------+----------------+----------------|
 |       12       |       13       |       14       |       15       |       16       |       17       |       18       |
 |                |                |                |                |                |                |                |
 |----------------+----------------+----------------+----------------+----------------+----------------+----------------|
 |       19       |       20       |       21       |       22       |       23       |       24       |       25       |
 |                |                |                |                |                |                |                |
 |----------------+----------------+----------------+----------------+----------------+----------------+----------------|
 |       26       |       27       |       28       |       29       |       30       |       31       |                |
 |                |                |                |                |                |                |                |
 --------------------------------------------------------------------------------------------------------------------
```

Calendar of Holidays Only 2

February 1997

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Calendar of Holidays Only 12

December 1997

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25 *Christmas Break*	26 *Christmas Break*	27
28	29 *Christmas Break*	30 *Christmas Break*	31 *Christmas Break*			

Example 6: Calculating a Schedule Based on Completion of Predecessor Tasks

Procedure features:

PROC CALENDAR statement

CALID statement

FIN statement

VAR statement

Other features:

PROC CPM step
PROC SORT step

Automating Your Scheduling Task with SAS/OR Software

When changes occur to a schedule, you have to adjust the activity starting dates manually if you use PROC CALENDAR to produce a schedule calendar. Alternatively, you can use PROC CPM in SAS/OR software to reschedule work when dates change. Even more important, you can provide only an initial starting date for a project and let PROC CPM calculate starting dates for activities, based on identified successor tasks, that is, tasks that cannot begin until their predecessors end.

In order to use PROC CPM, you must

1. create an activities data set that contains activities with durations. (You can indicate nonwork days, weekly work schedules, and workshifts with holidays, calendar, and workshift data sets.)
2. indicate which activities are successors to others (precedence relationships).
3. define resource limitations *if* you want them considered in the schedule.
4. provide an initial starting date.

PROC CPM can process your data to generate a data set that contains the start and end dates for each activity. PROC CPM schedules the activities, based on the duration information, weekly work patterns, workshifts, as well as holidays and nonwork days that interrupt the schedule. You can generate several views of the schedule that is computed by PROC CPM, from a simple listing of start and finish dates to a calendar, a Gantt chart, or a network diagram.

Highlights of This Example

This example

- calculates a project schedule containing multiple calendars (PROC CPM)
- produces a listing of the PROC CPM output data set (PROC PRINT)
- displays the schedule in calendar format (PROC CALENDAR).

This example features PROC CPM's ability to calculate a schedule that

- is based on an initial starting date
- applies different non-work periods to different calendars, such as personal vacation days to each employee's schedule
- includes milestones (activities with a duration of 0).

See Also

This example introduces users of PROC CALENDAR to more advanced SAS scheduling tools. For an introduction to project management tasks and tools and several examples, see *Project Management Using the SAS System*. For more examples, see *SAS/OR Software: Project Management Examples*. For complete reference documentation, see *SAS/OR User's Guide: Project Management, Version 6, First Edition*.

Program

Set appropriate options. If the linesize is not long enough to print the variable values, PROC CALENDAR either truncates the values or produces no calendar output. A longer linesize also makes it easier to view a listing of a PROC CPM output data set.

```
options nodate pageno=1 linesize=132 pagesize=60;
```

Create the activities data set and identify separate calendars. These data identify two calendars: the professor's (the value of _CAL_ is **Prof.**) and the student's (the value of _CAL_ is **Student**). The Succ1 variable identifies which activity cannot begin until the current one ends. For example **Analyze Exp 1** cannot begin until **Run Exp 1** is completed. The DAYS value of **0** for JOBNUM **3, 6,** and **8** indicates that these are milestones.

```
data grant;
   input jobnum Task $ 4-22 Days Succ1 $ 27-45 aldate : date7. altype $
         _cal_ $;
   format aldate date7.;
   datalines;
1  Run Exp 1          11  Analyze Exp 1       .      .   Student
2  Analyze Exp 1       5  Send Report 1       .      .   Prof.
3  Send Report 1       0  Run Exp 2           .      .   Prof.
4  Run Exp 2          11  Analyze Exp 2       .      .   Student
5  Analyze Exp 2       4  Send Report 2       .      .   Prof.
6  Send Report 2       0  Write Final Report  .      .   Prof.
7  Write Final Report  4  Send Final Report   .      .   Prof.
8  Send Final Report   0                      .      .   Student
9  Site Visit          1                    18jul96 ms  Prof.
;
```

Create the holidays data set and identify which calendar a nonwork day belongs to. The two holidays are listed twice, once for the professor's calendar and once for the student's. Because each person is associated with a separate calendar, PROC CPM can apply the personal vacation days to the appropriate calendars.

```
data nowork;
   format holista date7. holifin date7.;
   input holista : date7. holifin : date7. name $ 17-32 _cal_ $;
   datalines;
04jul96 04jul96 Independence Day Prof.
02sep96 02sep96 Labor Day        Prof.
04jul96 04jul96 Independence Day Student
02sep96 02sep96 Labor Day        Student
15jul96 16jul96 PROF Vacation    Prof.
15aug96 16aug96 STUDENT Vacation Student
;
```

Calculate the schedule with PROC CPM. PROC CPM uses information supplied in the activities and holidays data sets to calculate start and finish dates for each activity. The DATE= option supplies the starting date of the project. The CALID statement is not required, even though this example includes two calendars, because the calendar identification variable has the special name _CAL_.

```
proc cpm data=grant
         date='01jul96'd
         interval=weekday
         out=gcpm1
         holidata=nowork;
   activity task;
   successor succ1;
   duration days;
   calid _cal_;
   id task;
   aligndate aldate;
   aligntype altype;
   holiday holista / holifin=holifin;
run;
```

Print the output data set created with PROC CPM. This step is not required. PROC PRINT is a useful way to view the calculations produced by PROC CPM. See Output 5.10 on page 132.

```
proc print data=gcpm1;
   title 'Data Set GCPM1, Created with PROC CPM';
run;
```

Sort GCPM1 by the variable that contains the activity start dates before using it with PROC CALENDAR.

```
proc sort data=gcpm1;
   by e_start;
run;
```

Create the schedule calendar. GCPM1 is the activity data set. PROC CALENDAR uses the S_START and S_FINISH dates, calculated by PROC CPM, to print the schedule. The VAR statement selects only the variable TASK to display on the calendar output. See Output 5.11 on page 132.

```
proc calendar data=gcpm1
              holidata=nowork
              interval=workday;
   start e_start;
   fin   e_finish;
   calid _cal_ / output=combine;
   holistart holista;
   holifin   holifin;
   holivar name;
   var task;
   title 'Schedule for Experiment X-15';
   title2 'Professor and Student Schedule';
run;
```

Output

Output 5.10 The Data Set GCPM1

PROC PRINT displays the observations in GCPM1, showing the scheduling calculations created by PROC CPM.

Data Set GCPM1, Created with PROC CPM 1

Obs	Task	Succ1	Days	_cal_	E_START	E_FINISH	L_START	L_FINISH	T_FLOAT	F_FLOAT
1	Run Exp 1	Analyze Exp 1	11	Student	01JUL96	16JUL96	01JUL96	16JUL96	0	0
2	Analyze Exp 1	Send Report 1	5	Prof.	17JUL96	23JUL96	17JUL96	23JUL96	0	0
3	Send Report 1	Run Exp 2	0	Prof.	24JUL96	24JUL96	24JUL96	24JUL96	0	0
4	Run Exp 2	Analyze Exp 2	11	Student	24JUL96	07AUG96	24JUL96	07AUG96	0	0
5	Analyze Exp 2	Send Report 2	4	Prof.	08AUG96	13AUG96	08AUG96	13AUG96	0	0
6	Send Report 2	Write Final Report	0	Prof.	14AUG96	14AUG96	14AUG96	14AUG96	0	0
7	Write Final Report	Send Final Report	4	Prof.	14AUG96	19AUG96	14AUG96	19AUG96	0	0
8	Send Final Report		0	Student	20AUG96	20AUG96	20AUG96	20AUG96	0	0
9	Site Visit		1	Prof.	18JUL96	18JUL96	18JUL96	18JUL96	0	0

Output 5.11 Schedule Calendar Based on Output from PROC CPM

PROC CALENDAR created this schedule calendar by using the S_START and S_FINISH dates that were calculated by PROC CPM. The activities on July 24th and August 14th, because they are milestones, do not delay the start of a successor activity. Note that Site Visit occurs on July 18, the same day that Analyze Exp 1 occurs. To prevent this overallocation of resources, you can use **resource constrained scheduling**, available in SAS/OR software.

```
                                          Schedule for Experiment X-15                                                  2
                                         Professor and Student Schedule

          -------------------------------------------------------------------------------------------------------------------------
          |                                                                                                                       |
          |                                                   July  1996                                                          |
          |                                                                                                                       |
          |-----------------------------------------------------------------------------------------------------------------------|
          |     Sunday     |     Monday     |    Tuesday     |   Wednesday    |    Thursday    |     Friday     |    Saturday    |
----------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |                |       1        |       2        |       3        |       4        |       5        |       6        |
|.........|................|................|................|................|................|................|................|
| PROF.   |                |                |                |                |Independence Day|                |                |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |+===================Run Exp 1====================>|Independence Day|<==Run Exp 1===>|                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       7        |       8        |       9        |       10       |       11       |       12       |       13       |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |<====================================Run Exp 1=====================================>|                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       14       |       15       |       16       |       17       |       18       |       19       |       20       |
|.........|................|................|................|................|................|................|................|
| PROF.   |                |*PROF Vacation**|*PROF Vacation**|                |+==Site Visit==+|                |                |
|         |                |                |                |+=================Analyze Exp 1==================>|                |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |<===========Run Exp 1===========+|                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       21       |       22       |       23       |       24       |       25       |       26       |       27       |
|.........|................|................|................|................|................|................|................|
| PROF.   |                |<=========Analyze Exp 1=========+|+Send Report 1=+|                |                |                |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |                |                |+===================Run Exp 2====================>|                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       28       |       29       |       30       |       31       |                |                |                |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |<===================Run Exp 2====================>|                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
----------------------------------------------------------------------------------------------------------------------------------
```

```
                                    Schedule for Experiment X-15                                                3
                                   Professor and Student Schedule

          -------------------------------------------------------------------------------------------------------------------------
          |                                                                                                                       |
          |                                                    August  1996                                                       |
          |                                                                                                                       |
          |-----------------------------------------------------------------------------------------------------------------------|
          |     Sunday     |     Monday     |    Tuesday     |   Wednesday    |    Thursday    |     Friday     |    Saturday    |
----------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |                |                |                |                |       1        |       2        |       3        |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |                |                |                |<===========Run Exp 2===========>|                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       4        |       5        |       6        |       7        |       8        |       9        |       10       |
|.........|................|................|................|................|................|................|................|
| PROF.   |                |                |                |                |+=========Analyze Exp 2=========>|                |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |<===================Run Exp 2====================+|                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       11       |       12       |       13       |       14       |       15       |       16       |       17       |
|.........|................|................|................|................|................|................|................|
| PROF.   |                |                |                |+===============Write Final Report===============>|                |
|         |                |<=========Analyze Exp 2=========+|+Send Report 2=+|                |                |                |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |                |                |                |STUDENT Vacation|STUDENT Vacation|                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       18       |       19       |       20       |       21       |       22       |       23       |       24       |
|.........|................|................|................|................|................|................|................|
| PROF.   |                |<Write Final Re+|                |                |                |                |                |
|.........|................|................|................|................|................|................|................|
| STUDENT |                |                |+Send Final Rep+|                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|---------+----------------+----------------+----------------+----------------+----------------+----------------+----------------|
|         |       25       |       26       |       27       |       28       |       29       |       30       |       31       |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
|         |                |                |                |                |                |                |                |
----------------------------------------------------------------------------------------------------------------------------------
```

Example 7: Summary Calendar with MEAN Values By Observation

Procedure features:

- CALID statement:
  - _CAL_ variable
  - OUTPUT=SEPARATE option
- FORMAT statement
- LABEL statement

MEAN statement
SUM statement

Other features:

PROC FORMAT:
PICTURE statement

This example

- produces a summary calendar
- displays holidays
- produces sum and mean values by business day (observation) for three variables
- prints a legend and uses variable labels
- uses picture formats to display values.

MEAN Values by Number of Days

To produce MEAN values based on *the number of days in the calendar month*, use MEANTYPE=NDAYS. By default, MEANTYPE=NOBS, which calculates the MEAN values according to *the number of days for which data exist*.

Program

Create the activities data set. MEALS records how many meals were served for breakfast, lunch, and dinner on the days that the cafeteria was open for business.

```
data meals;
   input date : date7. Brkfst Lunch Dinner;
   datalines;
02Dec96       123 234 238
03Dec96       188 188 198
04Dec96       123 183 176
05Dec96       200 267 243
06Dec96       176 165 177
09Dec96       178 198 187
10Dec96       165 176 187
11Dec96       187 176 231
12Dec96       176 187 222
13Dec96       187 187 123
16Dec96       176 165 177
17Dec96       156   . 167
18Dec96       198 143 167
19Dec96       178 198 187
20Dec96       165 176 187
23Dec96       187 187 123
;
```

Create the holidays data set.

```
data closed;
   input date date. holiday $ 11-25;
```

```
   datalines;
26DEC96   Repairs
27DEC96   Repairs
30DEC96   Repairs
31DEC96   Repairs
24DEC96   Christmas Eve
25DEC96   Christmas
;
```

Sort the activities data set by the activity starting date. You are not required to sort the holidays data set.

```
proc sort data=meals;
   by date;
run;
```

Create picture formats for the variables that indicate how many meals were served.

```
proc format;
   picture bfmt other = '000 Brkfst';
   picture lfmt other = '000 Lunch ';
   picture dfmt other = '000 Dinner';
run;
```

Set PAGESIZE= and LINESIZE= appropriately. The legend box prints on the next page if PAGESIZE= is not set large enough. LINESIZE= controls the width of the cells in the calendar.

```
options nodate pageno=1 linesize=132 pagesize=60;
```

Create the summary calendar. DATA= identifies the activities data set; HOLIDATA= identifies the holidays data set. The START statement specifies the variable in the activities data set that contains the activity starting date; START is required.

```
proc calendar data=meals holidata=closed;
   start date;
```

The HOLISTART and HOLIVAR statements specify the variables in the holidays data set that contain the start date and the name of each holiday, respectively. HOLISTART is required when you use a holidays data set.

```
   holistart date;
   holiname holiday;
```

The SUM and MEAN statements calculate sum and mean values for three variables and print them with the specified format. The LABEL statement prints a legend and uses labels instead of variable names. The FORMAT statement associates picture formats with three variables.

```
   sum brkfst lunch dinner / format=4.0;
   mean brkfst lunch dinner / format=6.2;
   label brkfst = 'Breakfasts Served'
         lunch  = '   Lunches Served'
         dinner = '   Dinners Served';
   format brkfst bfmt.
          lunch lfmt.
          dinner dfmt.;
   title 'Meals Served in Company Cafeteria';
   title2 'Mean Number by Business Day';
run;
```

Output

Output 5.12 Summary Calendar with MEAN Values by Observation

```
                                       Meals Served in Company Cafeteria                                          1
                                          Mean Number by Business Day

 ----------------------------------------------------------------------------------------------
 |                                                                                            |
 |                                       December  1996                                       |
 |                                                                                            |
 |--------------------------------------------------------------------------------------------|
 |   Sunday   |   Monday   |  Tuesday   | Wednesday  |  Thursday  |   Friday   |  Saturday  |
 |------------+------------+------------+------------+------------+------------+------------|
 |     1      |     2      |     3      |     4      |     5      |     6      |     7      |
 |            |            |            |            |            |            |            |
 |            | 123 Brkfst | 188 Brkfst | 123 Brkfst | 200 Brkfst | 176 Brkfst |            |
 |            | 234 Lunch  | 188 Lunch  | 183 Lunch  | 267 Lunch  | 165 Lunch  |            |
 |            | 238 Dinner | 198 Dinner | 176 Dinner | 243 Dinner | 177 Dinner |            |
 |------------+------------+------------+------------+------------+------------+------------|
 |     8      |     9      |     10     |     11     |     12     |     13     |     14     |
 |            |            |            |            |            |            |            |
 |            | 178 Brkfst | 165 Brkfst | 187 Brkfst | 176 Brkfst | 187 Brkfst |            |
 |            | 198 Lunch  | 176 Lunch  | 176 Lunch  | 187 Lunch  | 187 Lunch  |            |
 |            | 187 Dinner | 187 Dinner | 231 Dinner | 222 Dinner | 123 Dinner |            |
 |------------+------------+------------+------------+------------+------------+------------|
 |     15     |     16     |     17     |     18     |     19     |     20     |     21     |
 |            |            |            |            |            |            |            |
 |            | 176 Brkfst | 156 Brkfst | 198 Brkfst | 178 Brkfst | 165 Brkfst |            |
 |            | 165 Lunch  |          . | 143 Lunch  | 198 Lunch  | 176 Lunch  |            |
 |            | 177 Dinner | 167 Dinner | 167 Dinner | 187 Dinner | 187 Dinner |            |
 |------------+------------+------------+------------+------------+------------+------------|
 |     22     |     23     |     24     |     25     |     26     |     27     |     28     |
 |            |            |Christmas Ev|*Christmas**|**Repairs***|**Repairs***|            |
 |            | 187 Brkfst |            |            |            |            |            |
 |            | 187 Lunch  |            |            |            |            |            |
 |            | 123 Dinner |            |            |            |            |            |
 |------------+------------+------------+------------+------------+------------+------------|
 |     29     |     30     |     31     |            |            |            |            |
 |            |**Repairs***|**Repairs***|            |            |            |            |
 |            |            |            |            |            |            |            |
 |            |            |            |            |            |            |            |
 |            |            |            |            |            |            |            |
 ----------------------------------------------------------------------------------------------


                              -------------------------------------
                              |                   | Sum  |  Mean  |
                              |                   |      |        |
                              | Breakfasts Served | 2763 | 172.69 |
                              |    Lunches Served | 2830 | 188.67 |
                              |    Dinners Served | 2990 | 186.88 |
                              -------------------------------------
```

Example 8: Multiple Summary Calendars with Atypical Workshifts (Separated Output)

Procedure features:

- PROC CALENDAR statementoptions:
  - DATETIME
  - LEGEND
- CALID statement:
  - _CAL_ variable
  - OUTPUT=SEPARATE option

OUTSTART statement
OUTFIN statement
SUM statement

Data sets:
WELL.ACT on page 116 and WELL.HOL on page 117.

This example

- □ produces a summary calendar for multiple calendars in a single PROC step
- □ prints the calendars on separate pages
- □ displays holidays
- □ uses separate work patterns, work shifts, and holidays for each calendar

Producing Different Output for Multiple Calendars

This example produces separate output for multiple calendars. To produce combined or mixed output for these data, you need to change only two things:

- □ how the activities data set is sorted
- □ how the OUTPUT= option is set.

To print . . .	Sort the activities data set by . . .	And set OUTPUT= to	See Example
Separate pages for each calendar	calendar id and starting date	SEPARATE	3, 8
All activities on the same page and identify each calendar	starting date	COMBINE	4, 2
All activities on the same page and NOT identify each calendar	starting date	MIX	4

Program

Specify the SAS data library where the activities data set is stored.

```
libname well 'SAS-data-library';
run;
```

Sort the activities data set by the variables containing the calendar identification and the starting date, respectively.

```
proc sort data=well.act;
   by _cal_ date;
```

```
run;
```

Set PAGESIZE= and LINESIZE= appropriately. The legend box prints on the next page if PAGESIZE= is not set large enough. LINESIZE= controls the width of the boxes.

```
options nodate pageno=1 linesize=132 pagesize=60;
```

Create the summary calendar. DATA= identifies the activities data set; HOLIDATA= identifies the holidays data set; CALDATA= identifies the calendar data set; WORKDATA= identifies the workdays data set. DATETIME specifies that the variable specified with the START statement contains a SAS datetime value. LEGEND prints text that identifies the variables.

```
proc calendar data=well.act
              holidata=well.hol
              datetime legend;
```

The CALID statement specifies that the _CAL_ variable identifies calendars. OUTPUT=SEPARATE prints information for each calendar on separate pages.

```
   calid _cal_ / output=separate;
```

The START statement specifies the variable in the activities data set that contains the activity starting date. The HOLISTART and HOLIVAR statements specify the variables in the holidays data set that contain the start date and name of each holiday, respectively. These statements are required when you use a holidays data set.

```
   start date;
   holistart date;
   holivar holiday;
```

The SUM statement totals the COST variable for all observations in each calendar.

```
   sum cost / format=dollar10.2;
```

Display a 6-day week. OUTSTART and OUTFIN specify that the calendar display a 6-day week, Monday through Saturday.

```
   outstart Monday;
   outfin Saturday;
   title 'Well Drilling Cost Summary';
   title2 'Separate Calendars';
   format cost dollar10.2;
run;
```

Output

Output 5.13 Separated Output for Multiple Summary Calendars

```
                                    Well Drilling Cost Summary                                         1
                                        Separate Calendars

............................................................ _cal_=CAL1 ............................................................

        --------------------------------------------------------------------------------------------------------------------
        |                                                                                                                  |
        |                                                  July  1996                                                      |
        |                                                                                                                  |
        |------------------------------------------------------------------------------------------------------------------|
        |      Monday      |     Tuesday      |    Wednesday     |     Thursday     |      Friday      |     Saturday     |
        |------------------+------------------+------------------+------------------+------------------+------------------|
        |        1         |        2         |        3         |        4         |        5         |        6         |
        |                  |                  |                  |***Independence***|                  |                  |
        | Drill Well       |                  |                  | Lay Power Line   | Assemble Tank    |                  |
        |              3.5 |                  |                  |                3 |                4 |                  |
        |        $1,000.00 |                  |                  |        $2,000.00 |        $1,000.00 |                  |
        |------------------+------------------+------------------+------------------+------------------+------------------|
        |        8         |        9         |       10         |       11         |       12         |       13         |
        |                  |                  |                  |                  |                  |                  |
        | Build Pump House |                  |                  | Pour Foundation  |                  |                  |
        |                3 |                  |                  |                4 |                  |                  |
        |        $2,000.00 |                  |                  |        $1,500.00 |                  |                  |
        |------------------+------------------+------------------+------------------+------------------+------------------|
        |       15         |       16         |       17         |       18         |       19         |       20         |
        |                  |                  |                  |                  |                  |                  |
        | Install Pump     |                  |                  |                  | Install Pipe     | Erect Tower      |
        |                4 |                  |                  |                  |                2 |                6 |
        |          $500.00 |                  |                  |                  |        $1,000.00 |        $2,500.00 |
        |------------------+------------------+------------------+------------------+------------------+------------------|
        |       22         |       23         |       24         |       25         |       26         |       27         |
        |                  |                  |                  |                  |                  |                  |
        |                  |                  |                  |                  |                  |                  |
        |                  |                  |                  |                  |                  |                  |
        |                  |                  |                  |                  |                  |                  |
        |------------------+------------------+------------------+------------------+------------------+------------------|
        |       29         |       30         |       31         |                  |                  |                  |
        |                  |                  |                  |                  |                  |                  |
        |                  |                  |                  |                  |                  |                  |
        |                  |                  |                  |                  |                  |                  |
        |                  |                  |                  |                  |                  |                  |
        --------------------------------------------------------------------------------------------------------------------

                                    -------------------------
                                    |  Legend  |    Sum     |
                                    |          |            |
                                    | task     |            |
                                    | dur      |            |
                                    | cost     | $11,500.00 |
                                    -------------------------
```

```
                              Well Drilling Cost Summary                                    2
                                  Separate Calendars

............................................ _cal_=CAL2 ............................................

  ----------------------------------------------------------------------------------------------------------------
  |                                                                                                              |
  |                                                July  1996                                                    |
  |                                                                                                              |
  |--------------------------------------------------------------------------------------------------------------|
  |     Monday      |     Tuesday     |    Wednesday    |    Thursday     |     Friday      |    Saturday     |
  |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
  |        1        |        2        |        3        |        4        |        5        |        6        |
  |                 |                 |                 |                 |                 |                 |
  | Deliver Material|                 | Excavate        |                 |                 |                 |
  |               2 |                 |            4.75 |                 |                 |                 |
  |         $500.00 |                 |       $3,500.00 |                 |                 |                 |
  |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
  |        8        |        9        |       10        |       11        |       12        |       13        |
  |                 |*****Vacation*****|                |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
  |       15        |       16        |       17        |       18        |       19        |       20        |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
  |       22        |       23        |       24        |       25        |       26        |       27        |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |-----------------+-----------------+-----------------+-----------------+-----------------+-----------------|
  |       29        |       30        |       31        |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  |                 |                 |                 |                 |                 |                 |
  ----------------------------------------------------------------------------------------------------------------

                                          -------------------------
                                          |  Legend  |    Sum     |
                                          |          |            |
                                          | task     |            |
                                          | dur      |            |
                                          | cost     |  $4,000.00 |
                                          -------------------------
```

CHAPTER 6

The CATALOG Procedure

Overview: CATALOG Procedure

The CATALOG procedure manages entries in SAS catalogs. PROC CATALOG is an interactive, statement-driven procedure that enables you to

- create a listing of the contents of a catalog
- copy a catalog or selected entries within a catalog
- rename, exchange, or delete entries within a catalog
- change the name of a catalog entry
- modify, by changing or deleting, the description of a catalog entry.

For more information on SAS data libraries and catalogs, refer to *SAS Language Reference: Concepts*.

To learn how to use the SAS windowing environment to manage entries in a SAS catalog, see the SAS online Help for the SAS Explorer window. You may prefer to use the Explorer window instead of using PROC CATALOG. The window can do most of what the procedure does.

Syntax: PROC CATALOG

Tip: Supports RUN-group processing.

Tip: Supports the Output Delivery System. See “Output Delivery System” on page 32 for details.

Reminder: You can perform similar functions with the SAS Explorer window and with dictionary tables in the SQL procedure. For information on the Explorer window, see the online Help. For information on PROC SQL, see Chapter 40, “The SQL Procedure,” on page 1113.

PROC CATALOG CATALOG=<*libref.*>*catalog* <ENTRYTYPE=*etype*> <FORCE> <KILL>;

CONTENTS <OUT=*SAS-data-set*> <FILE=*fileref*>;

COPY OUT=<*libref.*>*catalog* <*options*>;
SELECT *entry(s)* </ ENTRYTYPE=*etype*>;
EXCLUDE *entry(s)* </ ENTRYTYPE=*etype*>;

CHANGE *old-name-1=new-name-1*
<*…old-name-n=new-name-n*>
</ ENTRYTYPE=*etype*>;

EXCHANGE *name-1=other-name-1*
<*…name-n=other-name-n*>
</ ENTRYTYPE=*etype*>;

DELETE *entry(s)* </ ENTRYTYPE=*etype*>;

MODIFY *entry* (DESCRIPTION=<<’>*entry-description*<’>>)</ ENTRYTYPE=*etype*>;

SAVE *entry(s)* </ ENTRYTYPE=*etype*>;

To do this	Use this statement
Copy entries from one SAS catalog to another	
Copy or move all entries	COPY (with MOVE option)
Copy entries to a new catalog (overwriting the catalog if it already exists)	COPY (with NEW option)
Copy only selected entries	COPY, SELECT
Copy all *except* the entries specified	COPY, EXCLUDE
Delete entries from a SAS catalog	
Delete *all* entries	PROC CATALOG (with KILL option)
Delete specified entries	DELETE
Delete all *except* the entries specified	SAVE
Alter names and descriptions	

To do this	Use this statement
Change the names of catalog entries	CHANGE
Switch the names of two catalog entries	EXCHANGE
Change the description of a catalog entry	MODIFY
Print	
Print the contents of a catalog	CONTENTS

PROC CATALOG Statement

PROC CATALOG CATALOG=<*libref.*>*catalog* <ENTRYTYPE=*etype*> <FORCE>
<KILL>;

To do this	Use this option
Restrict processing to one entry type	ENTRYTYPE=
Delete all catalog entries	KILL
Force certain statements to execute on a catalog opened by another process	FORCE

Required Arguments

CATALOG=<*libref.*>*catalog*
specifies the SAS catalog to process.

Alias: CAT=, C=

Default: If ENTRYTYPE= is not specified, PROC CATALOG processes all entries in the catalog.

Options

ENTRYTYPE=*etype*
restricts processing of the current PROC CATALOG step to one entry type.

Alias: ET=

Default: If you omit ENTRYTYPE=, PROC CATALOG processes all entries in a catalog.

Interaction: The specified entry type applies to any one-level entry names used in a subordinate statement. You cannot override this specification in a subordinate statement.

Interaction: ENTRYTYPE= does not restrict the effects of the KILL option.

Tip: In order to process multiple entry types in a single PROC CATALOG step, use ENTRYTYPE= in a subordinate statement, not in the PROC CATALOG statement.

See also: "Specifying an Entry Type" on page 155.

Featured in: Example 1 on page 158 and Example 2 on page 162

FORCE
forces statements to execute on a catalog opened by another process.

Some CATALOG statements require exclusive access to the catalog they operate on if the statement can radically change the contents of a catalog. If exclusive access cannot be obtained, the action fails. The statements and the catalogs that are affected are

KILL	affects the specified catalog
COPY	affects the OUT= catalog
COPY MOVE	affects the IN= and the OUT= catalogs
SAVE	affects the specified catalog.

Tip: Use FORCE to execute the statement, even if exclusive access cannot be obtained.

KILL
deletes all entries in a SAS catalog.

Interaction: The KILL option deletes all catalog entries even when ENTRYTYPE= is specified.

Interaction: The SAVE statement has no effect because the KILL option deletes all entries in a SAS catalog before any other statements are processed.

Tip: KILL deletes all entries but does not remove an empty catalog from the SAS data library. You must use another method, such as PROC DATASETS or the DIR window to delete an empty SAS catalog.

CAUTION:
Do not attempt to limit the effects of the KILL option. This option deletes all entries in a SAS catalog before any option or other statement takes effect. △

CHANGE Statement

Renames one or more catalog entries.

Tip: You can change multiple names in a single CHANGE statement or use multiple CHANGE statements.

Featured in: Example 2 on page 162

CHANGE *old-name-1=new-name-1*
<*...old-name-n=new-name-n*>
</ ENTRYTYPE=*etype*>;

Required Arguments

old-name=new-name

specifies the current name of a catalog entry and the new name you want to assign to it. Specify any valid SAS name.

Restriction: You must designate the type of the entry, either with the name (*ename.etype*) or with the ENTRYTYPE= option.

Options

ENTRYTYPE=*etype*
restricts processing to one entry type.

See: "The ENTRYTYPE= Option" on page 156

See also: "Specifying an Entry Type" on page 155

CONTENTS Statement

Lists the contents of a catalog in the procedure output or writes a list of the contents to a SAS data set, an external file, or both.

Featured in: Example 2 on page 162

CONTENTS <OUT=*SAS-data-set*> <FILE=*fileref*>;

Without Options

The output is sent to the procedure output.

Options

Note: The ENTRYTYPE= (ET=) option is not available for the CONTENTS statement. △

CATALOG=*<libref.>catalog*
specifies the SAS catalog to process.

Alias: CAT=, C=

Default: None

FILE=*fileref*
sends the contents to an external file, identified with a SAS fileref.

Interaction: If *fileref* has not been previously assigned to a file, then the file is created and named according to operating environment-dependent rules for external files.

OUT=*SAS-data-set*
sends the contents to a SAS data set. When the statement executes, a message on the SAS log reports that a data set has been created. The data set contains six variables in this order:

LIBNAME	the libref
MEMNAME	the catalog name
NAME	the names of entries
TYPE	the types of entries
DESC	the descriptions of entries
DATE	the dates entries were last modified.

COPY Statement

Copies some or all of the entries in one catalog to another catalog.

Restriction: A COPY statement's effect ends at a RUN statement or at the beginning of a statement other than the SELECT or EXCLUDE statement.

Tip: Use SELECT or EXCLUDE statements, but not both, after the COPY statement to limit which entries are copied.

Tip: You can copy entries from multiple catalogs in a single PROC step, not just the one specified in the PROC CATALOG statement.

Tip: The ENTRYTYPE= option does not require a forward slash (/) in this statement.

Featured in: Example 1 on page 158

COPY OUT=<*libref.*>*catalog* <*options*>;

To do this	Use this option
Restrict processing to one type of entry	ENTRYTYPE=
Copy from a different catalog in the same step	IN=
Move (copy and then delete) a catalog entry	MOVE
Copy entries to a new catalog (overwriting the catalog if it already exists)	NEW
Protect several types of SAS/AF entries from being edited with PROC BUILD	NOEDIT
Not copy source lines from a PROGRAM, FRAME, or SCL entry	NOSOURCE

Required Arguments

OUT=<*libref.*>*catalog*
names the catalog to which entries are copied.

Options

ENTRYTYPE=*etype*
restricts processing to one entry type for the current COPY statement and any subsequent SELECT or EXCLUDE statements.

See: "The ENTRYTYPE= Option" on page 156

See also: "Specifying an Entry Type" on page 155

IN=<*libref.*>*catalog*
specifies the catalog to copy.

Interaction: The IN= option overrides a CATALOG= argument that was specified in the PROC CATALOG statement.

Featured in: Example 1 on page 158

MOVE
deletes the original catalog or entries after the new copy is made.

Interaction: When MOVE removes all entries from a catalog, the procedure deletes the catalog from the library.

NEW
overwrites the destination (specified by OUT=) if it already exists. If you omit NEW, PROC CATALOG updates the destination. For information about using the NEW option with concatenated catalogs, see "Catalog Concatenation" on page 157.

NOEDIT
prevents the copied version of the following SAS/AF entry types from being edited by the BUILD procedure:

CBT	PROGRAM
FRAME	SCL
HELP	SYSTEM
MENU	

Restriction: If you specify the NOEDIT option for an entry that is not one of these types, it is ignored.

Tip: When creating SAS/AF applications for other users, use NOEDIT to protect the application by preventing certain catalog entries from being altered.

Featured in: Example 1 on page 158

NOSOURCE
omits copying the source lines when you copy a SAS/AF PROGRAM, FRAME, or SCL entry.

Alias: NOSRC

Restriction: If you specify this option for an entry other than a PROGRAM, FRAME, or SCL entry, it is ignored.

DELETE Statement

Deletes entries from a SAS catalog.

Tip: Use DELETE to delete only a few entries; use SAVE when it is more convenient to specify which entries *not* to delete.

Tip: You can specify multiple entries. You can also use multiple DELETE statements.

See also: "SAVE Statement" on page 152

Featured in: Example 1 on page 158

DELETE *entry(s)* </ ENTRYTYPE=*etype*>;

Required Arguments

entry(s)
specifies the name of one or more SAS catalog entries.

Restriction: You must designate the type of the entry, either with the name (*ename.etype*) or with the ENTRYTYPE= option.

Options

ENTRYTYPE=*etype*
restricts processing to one entry type.

See: "The ENTRYTYPE= Option" on page 156

See also: "Specifying an Entry Type" on page 155

EXCHANGE Statement

Switches the name of two catalog entries.

Restriction: The catalog entries must be of the same type.

EXCHANGE *name-1=other-name-1*
<*…name-n=other-name-n*>
</ ENTRYTYPE=*etype*>;

Required Arguments

name=other-name
specifies two catalog entry names that the procedure will switch.

Interaction: You can specify only the entry name without the entry type if you use the ENTRYTYPE= option on either the PROC CATALOG statement or the EXCHANGE statement.

See also: "Specifying an Entry Type" on page 155

Options

ENTRYTYPE=*etype*
restricts processing to one entry type.

See: "The ENTRYTYPE= Option" on page 156

See also: "Specifying an Entry Type" on page 155

EXCLUDE Statement

Specifies entries that the COPY statement does *not* copy.

Restriction: Requires the COPY statement.

Restriction: Do not use the EXCLUDE statement with the SELECT statement.

Tip: You can specify multiple entries in a single EXCLUDE statement.

Tip: You can use multiple EXCLUDE statements with a single COPY statement within a RUN group.

See also: "COPY Statement" on page 148 and "SELECT Statement" on page 153

Featured in: Example 1 on page 158

EXCLUDE *entry(s)* </ ENTRYTYPE=*etype*>;

Required Arguments

entry(s)
specifies the name of one or more SAS catalog entries.

Restriction: You must designate the type of the entry, either when you specify the name (*ename.etype*) or with the ENTRYTYPE= option.

See also: "Specifying an Entry Type" on page 155

Options

ENTRYTYPE=*etype*
restricts processing to one entry type.

See: "The ENTRYTYPE= Option" on page 156

See also: "Specifying an Entry Type" on page 155

MODIFY Statement

Changes the description of a catalog entry.

Featured in: Example 2 on page 162

MODIFY *entry* (DESCRIPTION=<<'>*entry-description*<'>>) </ ENTRYTYPE=*etype*>;

Required Arguments

entry
specifies the name of one SAS catalog entry. Optionally, you can specify the entry type with the name.

Restriction: You must designate the type of the entry, either when you specify the name (*ename.etype*) or with the ENTRYTYPE= option.

See also: "Specifying an Entry Type" on page 155

DESCRIPTION=<<'>*entry-description*<'>>
changes the description of a catalog entry by replacing it with a new description, up to 256 characters long, or by removing it altogether. Optionally, you can enclose the description in single or double quotes.

Alias: DESC

Tip: Use DESCRIPTION= with no text to remove the current description.

Options

ENTRYTYPE=*etype*
restricts processing to one entry type.

See: "The ENTRYTYPE= Option" on page 156

See also: "Specifying an Entry Type" on page 155

SAVE Statement

Specify entries *not* to delete from a SAS catalog.

Restriction: Cannot limit the effects of the KILL option.

Tip: Use SAVE to delete all but a few entries in a catalog. Use DELETE when it is more convenient to specify which entries to delete.

Tip: You can specify multiple entries and use multiple SAVE statements.

See also: "DELETE Statement" on page 150

SAVE *entry(s)* </ ENTRYTYPE=*etype*>;

Required Arguments

entry(s)
specifies the name of one or more SAS catalog entries.

Restriction: You must designate the type of the entry, either with the name (*ename.etype*) or with the ENTRYTYPE= option.

Options

ENTRYTYPE=*etype*
restricts processing to one entry type.

See: "The ENTRYTYPE= Option" on page 156

See also: "Specifying an Entry Type" on page 155

SELECT Statement

Specifies entries that the COPY statement will copy.

Restriction: Requires the COPY statement.

Restriction: Cannot be used with an EXCLUDE statement.

Tip: You can specify multiple entries in a single SELECT statement.

Tip: You can use multiple SELECT statements with a single COPY statement within a RUN group.

See also: "COPY Statement" on page 148 and "EXCLUDE Statement" on page 151

Featured in: Example 1 on page 158

SELECT *entry(s)* </ ENTRYTYPE=*etype*>;

Required Arguments

entry(s)
specifies the name of one or more SAS catalog entries.

Restriction: You must designate the type of the entry, either when you specify the name (*ename.etype*) or with the ENTRYTYPE= option.

Options

ENTRYTYPE=*etype*
restricts processing to one entry type.

See: "The ENTRYTYPE= Option" on page 156.

See also: "Specifying an Entry Type" on page 155.

Concepts: CATALOG Procedure

Interactive Processing with RUN Groups

Definition

The CATALOG procedure is interactive. Once you submit a PROC CATALOG statement, you can continue to submit and execute statements or groups of statements without repeating the PROC CATALOG statement.

A set of procedure statements ending with a RUN statement is called a *RUN group*. The changes specified in a given group of statements take effect when a RUN statement is encountered.

How to End a PROC CATALOG Step

In the DATA step and most SAS procedures, a RUN statement is a step boundary and ends the step. A simple RUN statement does not, however, end an interactive procedure. To terminate a PROC CATALOG step, you can

- submit a QUIT statement
- submit a RUN statement with the CANCEL option
- submit another DATA or PROC statement
- end your SAS session.

Note: When you enter a QUIT, DATA, or PROC statement, any statements following the last RUN group execute before the CATALOG procedure terminates. If you enter a RUN statement with the CANCEL option, however, the remaining statements *do not execute* before the procedure ends. △

See Example 2 on page 162.

Error Handling and RUN Groups

Error handling is based in part on the division of statements into RUN groups. If a syntax error is encountered, *none* of the statements in the current RUN group execute, and execution proceeds to the next RUN group.

For example, the following statements contain a misspelled DELETE statement:

```
proc catalog catalog=misc entrytype=help;
   copy out=drink;
      select coffee tea;
   del juices;        /* INCORRECT!!! */
   exchange glass=plastic;
run;
   change calstats=nutri;
run;
```

Because the DELETE statement is incorrectly specified as DEL, no statements in that RUN group execute, *except* the PROC CATALOG statement itself. The CHANGE statement does execute, however, because it is in a different RUN group.

CAUTION:

Be careful when setting up batch jobs in which one RUN group's statements depend on the effects of a previous RUN group, especially when deleting and renaming entries. △

Specifying an Entry Type

Four Ways to Supply an Entry Type

There is no default entry type, so if you do not supply one, PROC CATALOG generates an error. You can supply an entry type in one of four ways. See Table 6.1 on page 155.

Table 6.1 Supplying an Entry Type

You can supply an entry type with...	Example
the entry name	`delete test1.program test1.log test2.log;`
ET= in parentheses	`delete test1 (et=program);`
ET= *after* a slash[1]	`delete test1 (et=program) test1 test2 / et=log;`
ENTRYTYPE= *without* a slash[2]	`proc catalog catalog=mycat et=log; delete test1 test2;`

1 in a subordinate statement
2 in the PROC CATALOG or the COPY statement

Note: All statements, except the CONTENTS statement, accept the ENTRYTYPE= (alias ET=) option. △

Why Use the ENTRYTYPE= Option?

ENTRYTYPE= can save keystrokes when you are processing multiple entries of the same type.

To create a default for entry type for all statements in the current step, use ENTRYTYPE= in the PROC CATALOG statement. To set the default for only the current statement, use ENTRYTYPE= in a subordinate statement.

If many entries are of one type, but a few are of other types, you can use ENTRYTYPE= to specify a default and then override that for individual entries with (ENTRYTYPE=) *in parentheses* after those entries.

Avoid a Common Error

You cannot specify the ENTRYTYPE= option in both the PROC CATALOG statement and a subordinate statement. For example, these statements generate an error and do not delete any entries because the ENTRYTYPE= specifications contradict each other:

```
/* THIS IS INCORRECT CODE. */
proc catalog cat=sample et=help;
```

```
   delete a b c / et=program;
run;
```

The ENTRYTYPE= Option

The ENTRYTYPE= option is available in every statement in the CATALOG procedure except CONTENTS.

ENTRYTYPE=*etype*

not in parentheses, sets a default entry type for the entire PROC step when used in the PROC CATALOG statement. In all other statements, this option sets a default entry type for the *current* statement.

Alias: ET=

Default: If you omit ENTRYTYPE=, PROC CATALOG processes all entries in the catalog.

Interaction: If you specify ENTRYTYPE= in the PROC CATALOG statement, do not specify either ENTRYTYPE= or (ENTRYTYPE=) in a subordinate statement.

Interaction: (ENTRYTYPE=*etype*) *in parentheses* immediately following an entry name overrides ENTRYTYPE= *in that same statement*.

Tip: On all statements *except* the PROC CATALOG and COPY statements, this option follows a slash.

Tip: To process multiple entry types in a single PROC CATALOG step, use ENTRYTYPE= in a subordinate statement, not in the PROC CATALOG statement.

See also: "Specifying an Entry Type" on page 155.

Featured in: Example 1 on page 158

(ENTRYTYPE=*etype*)

in parentheses, identifies the type of the entry just preceding it.

Alias: (ET=)

Restriction: (ENTRYTYPE=*etype*) immediately following an entry name in a subordinate statement *cannot override* an ENTRYTYPE= option *in the PROC CATALOG statement*. It generates a syntax error.

Interaction: (ENTRYTYPE=*etype*) immediately following an entry name overrides ENTRYTYPE= *in that same statement*.

Tip: This form is useful mainly for specifying exceptions to an ENTRYTYPE= option used in a subordinate statement. The following statement deletes A.HELP, B.FORMAT, and C.HELP:

```
delete a b (et=format) c / et=help;
```

Tip: For the CHANGE and EXCHANGE statements, specify (ENTRYTYPE=) *in parentheses* only once for each pair of names following the second name in the pair. For example,

```
change old1=new1 (et=log)
       old1=new2 (et=help);
```

See also: "Specifying an Entry Type" on page 155

Featured in: Example 1 on page 158 and Example 2 on page 162

Catalog Concatenation

The CATALOG procedure supports both implicit and explicit concatenation of catalogs. All statements and options that can be used on single (unconcatenated) catalogs can be used on catalog concatenations.

Restrictions

When you use the CATALOG procedure to copy concatenated catalogs and you use the NEW option, the following rules apply:

1 If the input catalog is a concatenation and if the output catalog exists in any level of the input concatenation, the copy is not allowed.

2 If the output catalog is a concatenation and if the input catalog exists in the first level of the output concatenation, the copy is not allowed.

For example, the following code demonstrates these two rules, and the copy fails:

```
libname first 'path-name1';
libname second 'path-name2';
/* create contat.x */
libname concat (first second);

/* fails rule #1 */
proc catalog c=concat.x;
   copy out=first.x new;
run;
quit;

/* fails rule #2 */
proc catalog c=first.x;
   copy out=concat.x new;
run;
quit;
```

In summary, the following table shows when copies are allowed. In the table, A and B are libraries, and each contains catalog X. Catalog C is an implicit concatenation of A and B, and catalog D is an implicit concatenation of B and A.

Input catalog	Output catalog	Copy allowed?
C.X	B.X	No
C.X	D.X	No
D.X	C.X	No
A.X	A.X	No
A.X	B.X	Yes
B.X	A.X	Yes
C.X	A.X	No
B.X	C.X	Yes
A.X	C.X	No

Examples: CATALOG Procedure

Example 1: Copying, Deleting, and Moving Catalog Entries from Multiple Catalogs

Procedure features:

PROC CATALOG statement:
- CATALOG= argument

COPY statement options:
- IN=
- MOVE
- NOEDIT

DELETE statement options:
- ENTRYTYPE= or ET=

EXCLUDE statement options:
- ENTRYTYPE= or ET=
- (ENTRYTYPE=) or (ET=)

QUIT statement

RUN statement

SELECT statement options:
- ENTRYTYPE= or ET=

This example

- copies entries by excluding a few entries
- copies entries by specifying a few entries
- protects entries from being edited
- moves entries
- deletes entries
- processes entries from multiple catalogs
- processes entries in multiple run groups.

Input Catalogs

The SAS catalog PERM.SAMPLE contains the following entries:

```
DEFAULT     FORM        Default form for printing
FSLETTER    FORM        Standard form for letters (HP Laserjet)
LOAN        FRAME       Loan analysis application
LOAN        HELP        Information about the application
BUILD       KEYS        Function Key Definitions
LOAN        KEYS        Custom key definitions for application
CREDIT      LOG         credit application log
TEST1       LOG         Inventory program
```

```
TEST2       LOG         Inventory program
TEST3       LOG         Inventory program
LOAN        PMENU       Custom menu definitions for applicaticm
CREDIT      PROGRAM     credit application pgm
TEST1       PROGRAM     testing budget applic.
TEST2       PROGRAM     testing budget applic.
TEST3       PROGRAM     testing budget applic.
LOAN        SCL         SCL code for loan analysis application
PASSIST     SLIST       User profile
PRTINFO     KPRINTER    Printing Parameters
```

The SAS catalog PERM.FORMATS contains the following entries:

```
REVENUE    FORMAT      FORMAT:MAXLEN=16,16,12
DEPT       FORMATC     FORMAT:MAXLEN=1,1,14
```

Program

Set the SAS system options. Write the source code to the log by specifying the SOURCE SAS system option.

```
options nodate pageno=1 linesize=80 pagesize=60 source;
```

Assign a library reference to a SAS data library. The LIBNAME statement assigns the libref PERM to the SAS data library that contains a permanent SAS catalog.

```
libname perm 'SAS-data-library';
```

Delete two entries from the PERM.SAMPLE catalog.

```
proc catalog cat=perm.sample;
   delete credit.program credit.log;
run;
```

Copy all entries in the PERM.SAMPLE catalog to the WORK.TCATALL catalog.

```
   copy out=tcatall;
run;
```

Copy everything except three LOG entries and PASSIST.SLIST from PERM.SAMPLE to WORK.TESTCAT. The EXCLUDE statement specifies which entries not to copy. ET= specifies a default type. (ET=) specifies an exception to the default type.

```
   copy out=testcat;
      exclude test1 test2 test3  passist (et=slist) / et=log;
run;
```

Move three LOG entries from PERM.SAMPLE to WORK.LOGCAT. The SELECT statement specifies which entries to move. ET= restricts processing to LOG entries.

```
   copy out=logcat move;
      select test1 test2 test3 / et=log;
run;
```

Copy five SAS/AF software entries from PERM.SAMPLE to PERM.FINANCE. The NOEDIT option protects these entries in PERM.FINANCE from further editing with PROC BUILD.

```
   copy out=perm.finance noedit;
      select loan.frame loan.help loan.keys loan.pmenu;
run;
```

Copy two formats from PERM.FORMATS to PERM.FINANCE. The IN= option enables you to copy from a different catalog than the one specified in the PROC CATALOG statement. Note the entry types for numeric and character formats: REVENUE.FORMAT is a numeric format and DEPT.FORMATC is a character format. The COPY and SELECT statements execute before the QUIT statement ends the PROC CATALOG step.

```
   copy in=perm.formats out=perm.finance;
      select revenue.format dept.formatc;
quit;
```

Log

```
1    libname perm 'SAS-data-library';
NOTE: Directory for library PERM contains files of mixed engine types.
NOTE: Libref PERM was successfully assigned as follows:
      Engine:        V9
      Physical Name: 'SAS-data-library'
2    options nodate pageno=1 linesize=80 pagesize=60 source;
3   proc catalog cat=perm.sample;
4      delete credit.program credit.log;
5   run;
NOTE: Deleting entry CREDIT.PROGRAM in catalog PERM.SAMPLE.
NOTE: Deleting entry CREDIT.LOG in catalog PERM.SAMPLE.
6      copy out=tcatall;
7   run;
NOTE: Copying entry DEFAULT.FORM from catalog PERM.SAMPLE to catalog
      WORK.TCATALL.
NOTE: Copying entry FSLETTER.FORM from catalog PERM.SAMPLE to catalog
      WORK.TCATALL.
NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry BUILD.KEYS from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST1.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST2.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST3.LOG from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry TEST1.PROGRAM from catalog PERM.SAMPLE to catalog
      WORK.TCATALL.
NOTE: Copying entry TEST2.PROGRAM from catalog PERM.SAMPLE to catalog
      WORK.TCATALL.
NOTE: Copying entry TEST3.PROGRAM from catalog PERM.SAMPLE to catalog
      WORK.TCATALL.
NOTE: Copying entry LOAN.SCL from catalog PERM.SAMPLE to catalog WORK.TCATALL.
NOTE: Copying entry PASSIST.SLIST from catalog PERM.SAMPLE to catalog
      WORK.TCATALL.
NOTE: Copying entry PRTINFO.XPRINTER from catalog PERM.SAMPLE to catalog
      WORK.TCATALL.
```

```
8      copy out=testcat;
9         exclude test1 test2 test3  passist (et=slist) / et=log;
10   run;
NOTE: Copying entry DEFAULT.FORM from catalog PERM.SAMPLE to catalog
      WORK.TESTCAT.
NOTE: Copying entry FSLETTER.FORM from catalog PERM.SAMPLE to catalog
      WORK.TESTCAT.
NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry BUILD.KEYS from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry TEST1.PROGRAM from catalog PERM.SAMPLE to catalog
      WORK.TESTCAT.
NOTE: Copying entry TEST2.PROGRAM from catalog PERM.SAMPLE to catalog
      WORK.TESTCAT.
NOTE: Copying entry TEST3.PROGRAM from catalog PERM.SAMPLE to catalog
      WORK.TESTCAT.
NOTE: Copying entry LOAN.SCL from catalog PERM.SAMPLE to catalog WORK.TESTCAT.
NOTE: Copying entry PRTINFO.XPRINTER from catalog PERM.SAMPLE to catalog
      WORK.TESTCAT.
11      copy out=logcat move;
12         select test1 test2 test3 / et=log;
13   run;
NOTE: Moving entry TEST1.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.
NOTE: Moving entry TEST2.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.
NOTE: Moving entry TEST3.LOG from catalog PERM.SAMPLE to catalog WORK.LOGCAT.
14      copy out=perm.finance noedit;
15         select loan.frame loan.help loan.keys loan.pmenu;
16   run;
NOTE: Copying entry LOAN.FRAME from catalog PERM.SAMPLE to catalog PERM.FINANCE.
NOTE: Copying entry LOAN.HELP from catalog PERM.SAMPLE to catalog PERM.FINANCE.
NOTE: Copying entry LOAN.KEYS from catalog PERM.SAMPLE to catalog PERM.FINANCE.
NOTE: Copying entry LOAN.PMENU from catalog PERM.SAMPLE to catalog PERM.FINANCE.
17      copy in=perm.formats out=perm.finance;
18         select revenue.format dept.formatc;
19   quit;
NOTE: Copying entry REVENUE.FORMAT from catalog PERM.FORMATS to catalog
      PERM.FINANCE.
NOTE: Copying entry DEPT.FORMATC from catalog PERM.FORMATS to catalog
      PERM.FINANCE.
```

Example 2: Displaying Contents, Changing Names, and Changing a Description

Procedure features:

PROC CATALOG statement
CHANGE statement options:
  (ENTRYTYPE=) or (ET=)
CONTENTS statement options:
  FILE=
MODIFY statement
RUN statement
QUIT statement

This example

- lists the entries in a catalog and routes the output to a file

- changes entry names
- changes entry descriptions
- processes entries in multiple run groups.

Program

Set the SAS system options. The system option SOURCE writes the source code to the log.

```
options nodate pageno=1 linesize=80 pagesize=60 source;
```

Assign a library reference. The LIBNAME statement assigns a libref to the SAS data library that contains a permanent SAS catalog.

```
libname perm 'SAS-data-library';
```

List the entries in a catalog and route the output to a file. The CONTENTS statement creates a listing of the contents of the SAS catalog PERM.FINANCE and routes the output to a file.

```
proc catalog catalog=perm.finance;
   contents;
title1 'Contents of PERM.FINANCE before changes are made';
run;
```

Change entry names. The CHANGE statement changes the name of an entry that contains a user-written character format. (ET=) specifies the entry type.

```
   change dept=deptcode (et=formatc);
run;
```

Process entries in multiple run groups. The MODIFY statement changes the description of an entry. The CONTENTS statement creates a listing of the contents of PERM.FINANCE after all the changes have been applied. QUIT ends the procedure.

```
   modify loan.frame (description='Loan analysis app. - ver1');
   contents;
title1 'Contents of PERM.FINANCE after changes are made';
run;
quit;
```

Output

Output 6.1

```
                  Contents of PERM.FINANCE before changes are made            1

                          Contents of Catalog PERM.FINANCE

# Name    Type            Create Date       Modified Date Description
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
1 REVENUE FORMAT   16OCT1996:13:48:11  16OCT1996:13:48:11 FORMAT:MAXLEN=16,16,12
2 DEPT    FORMATC  30OCT1996:13:40:42  30OCT1996:13:40:42 FORMAT:MAXLEN=1,1,14
3 LOAN    FRAME    30OCT1996:13:40:43  30OCT1996:13:40:43 Loan analysis
                                                          application
4 LOAN    HELP     16OCT1996:13:48:10  16OCT1996:13:48:10 Information about
                                                          the application
5 LOAN    KEYS     16OCT1996:13:48:10  16OCT1996:13:48:10 Custom key definitions
                                                          for application
6 LOAN    PMENU    16OCT1996:13:48:10  16OCT1996:13:48:10 Custom menu
                                                          definitions for
                                                          application
7 LOAN    SCL      16OCT1996:13:48:10  16OCT1996:13:48:10 SCL code for loan
                                                          analysis application
```

```
                  Contents of PERM.FINANCE after changes are made             2

                          Contents of Catalog PERM.FINANCE

# Name     Type            Create Date       Modified Date Description
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
1 REVENUE  FORMAT   16OCT1996:13:48:11  16OCT1996:13:48:11 FORMAT:MAXLEN=
                                                           16,16,12
2 DEPTCODE FORMATC  30OCT1996:13:40:42  30OCT1996:13:40:42 FORMAT:MAXLEN=1,1,14
3 LOAN     FRAME    30OCT1996:13:40:43  11FEB2002:13:20:50 Loan analysis
                                                           app. - ver1
4 LOAN     HELP     16OCT1996:13:48:10  16OCT1996:13:48:10 Information about
                                                           the application
5 LOAN     KEYS     16OCT1996:13:48:10  16OCT1996:13:48:10 Custom key
                                                           definitions for
                                                           application
6 LOAN     PMENU    16OCT1996:13:48:10  16OCT1996:13:48:10 Custom menu
                                                           definitions for
                                                           application
7 LOAN     SCL      16OCT1996:13:48:10  16OCT1996:13:48:10 SCL code for loan
                                                           analysis application
```

CHAPTER

7

The CHART Procedure

Overview: CHART Procedure

The CHART procedure produces vertical and horizontal bar charts, block charts, pie charts, and star charts. These types of charts graphically display values of a variable or a statistic associated with those values. The charted variable can be numeric or character.

PROC CHART is a useful tool to visualize data quickly, but if you need to produce presentation-quality graphics that include color and various fonts, you can use SAS/GRAPH software. The GCHART procedure in SAS/GRAPH software produces the same types of charts as PROC CHART does. In addition, PROC GCHART can produce donut charts.

The following sections explain the different types of charts that PROC CHART can produce. All of the charts illustrate the results from a multiple-choice survey of 568 people, with five possible responses that range from “always” to “never.”

About Bar Charts

Horizontal and vertical bar charts display the magnitude of data with bars, each of which represents a category of data. The length or height of the bars represents the value of the chart statistic for each category.

Output 7.1 on page 166 shows a vertical bar chart that displays the number of responses for the five categories from the survey data. The following statements produce the output:

```
options nodate pageno=1 linesize=80
        pagesize=30;

proc chart data=survey;
   vbar response / sumvar=count
        midpoints='Always' 'Usually'
           'Sometimes' 'Rarely' 'Never';
run;
```

Output 7.1 Vertical Bar Chart

```
                             The SAS System                                1

Count Sum

200 +                   *****
    |                   *****
    |                   *****
    |                   *****
    |                   *****
150 +                   *****
    |                   *****
    |                   *****
    |                   *****       *****
    |       *****       *****       *****
100 +       *****       *****       *****       *****
    |       *****       *****       *****       *****
    |       *****       *****       *****       *****
    |       *****       *****       *****       *****
    |       *****       *****       *****       *****
 50 +       *****       *****       *****       *****
    |       *****       *****       *****       *****       *****
    |       *****       *****       *****       *****       *****
    |       *****       *****       *****       *****       *****
    |       *****       *****       *****       *****       *****
    ---------------------------------------------------------------------
           Always      Usually    Sometimes    Rarely       Never

                                  Response
```

Output 7.2 on page 167 shows the same data presented in a horizontal bar chart. The two types of bar charts have essentially the same characteristics, except that horizontal bar charts by default display a table of statistic values to the right of the bars. The following statements produce the output:

```
options nodate pageno=1 linesize=80
        pagesize=60;

proc chart data=survey;
   hbar response / sumvar=count
        midpoints='Always' 'Usually'
           'Sometimes' 'Rarely' 'Never';
run;
```

Output 7.2 Horizontal Bar Chart

```
                              The SAS System                              1

  Response                                                    Count
                                                                Sum
             |
  Always     |*********************                         106.0000
             |
  Usually    |****************************************      202.0000
             |
  Sometimes  |************************                      119.0000
             |
  Rarely     |*******************                            97.0000
             |
  Never      |*********                                      44.0000
             |
             ----+---+---+---+---+---+---+---+---+---+
                 20  40  60  80 100 120 140 160 180 200

                              Count Sum
```

About Block Charts

Block charts display the relative magnitude of data by using blocks of varying height, each set in a square that represents a category of data. Output 7.3 on page 167 shows the number of each survey response in the form of a block chart.

```
options nodate pageno=1 linesize=80
        pagesize=30;

proc chart data=survey;
   block response / sumvar=count
         midpoints='Always' 'Usually'
            'Sometimes' 'Rarely' 'Never';
run;
```

Output 7.3 Block Chart

```
                                The SAS System                                1

                           Sum of Count by Response

                        ___
                       /_ /|
                      |**| |
                      |**| |
                      |**| |          ___
          ___         |**| |         /_ /|          ___
         /_ /|        |**| |        |**| |         /_ /|
       -|**| |--------|**| |--------|**| |--------|**| |---------------------
      / |**| |      / |**| |      / |**| |      / |**| |      /   ___       /
     /  |**| |     /  |**| |     /  |**| |     /  |**| |     /   /_ /|     /
    /   |**| |    /   |**| |    /   |**| |    /   |**| |    /   |**| |    /
   /    |**|/    /    |**|/    /    |**|/    /    |**|/    /    |**|/    /
  /             /             /             /             /             /
 /      106    /      202    /      119    /       97    /       44    /
/-------------/-------------/-------------/-------------/-------------/

   Always        Usually      Sometimes      Rarely         Never

                              Response
```

About Pie Charts

Pie charts represent the relative contribution of parts to the whole by displaying data as wedge-shaped slices of a circle. Each slice represents a category of the data. Output 7.4 on page 168 shows the survey results divided by response into five pie slices. The following statements produce the output:

```
options nodate pageno=1 linesize=80
        pagesize=35;

proc chart data=survey;
   pie response / sumvar=count;
run;
```

Output 7.4 Pie Chart

```
                      The SAS System                                1

                 Sum of Count by Response


                             Never
                       ***********
          Rarely   ****    .      ****
                 **        .      .   **
               **          . 44   .     **
              *            .7.75%.        *   Always
            **      97     .    ..         **
           **     17.08%   .   .            **
           * ..            .   .     106     *
          *     ..          . .    18.66%     *
          *        .. .     ..                *
          *            . .                    *
          *                 +  . . .. . .. . .*
          *     119                           *
          *   20.95%     ..                   *
Sometimes *             .                     *
           *           .                     *
           **         .        202          **
             *      ..       35.56%        *
              *    .                      *
               ** .                     **
                 **                   **
                   ****           ****
                       ***********    Usually
```

About Star Charts

With PROC CHART, you can produce star charts that show group frequencies, totals, or mean values. A star chart is similar to a vertical bar chart, but the bars on a star chart radiate from a center point, like spokes in a wheel. Star charts are commonly used for cyclical data, such as measures taken every month or day or hour, or for data like these in which the categories have an inherent order ("always" meaning more frequent than "usually" which means more frequent than "sometimes"). Output 7.5 on page 169 shows the survey data displayed in a star chart. The following statements produce the output:

```
options nodate pageno=1 linesize=80
        pagesize=60;

proc chart data=survey;
   star response / sumvar=count;
run;
```

Output 7.5 Star Chart

```
                                 The SAS System                                1

Center = 0                  Sum of Count by Response               Outside = 202


                                                 Never
                                 *************    44
                            *****             *****
                         ***                       ***
                      ***                             ***
                    **                                   **
                   *                                       *
         Rarely  **                                         **
           97   *                                             *
              **                                               **
              *                                                 *
             *                                                   *
            **                                                   **
            *               *......                               *
           **                ". ..  ......*.                       **
           *               ..   ..      .. ...                     *
           *               .      ..    .     ....                 *
           *               .        .. ..         ...              *
           *               .          .+..............*            *  Always
           *              .         .. ..             .            *   106
           *              .      ...    .            .             *
           *              .     .        .           .             *
           **            ..  ...         .          ..           **
            *            . ..             .         .             *
            **           *.               ..        .            **
             *             ..              .       ..            *
              *              ..             .      .            *
               *               ..           .     ..           *
                *                ...        ..    .           *
      Sometimes  **                 ..       .    .         **
         119       *                  ...     .  ..        *
                    **                   .    .  .       **
                      ***                 ...  . .    ***
                         ***                 ....  ***
                            *****             *.***
                                 *************   Usually
                                                   202
```

Syntax: CHART Procedure

Requirement: You must use at least one of the chart-producing statements.

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

PROC CHART <*option(s)*>;

BLOCK *variable(s)* </ *option(s)*>;

BY <DESCENDING> *variable-1*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

HBAR *variable(s)* </ *option(s)*>;

PIE *variable(s)* </ *option(s)*>;

STAR *variable(s)* </ *option(s)*>;

VBAR *variable(s)* </ *option(s)*>;

PROC CHART Statement

PROC CHART <*option(s)*>;

Options

DATA=*SAS-data-set*
identifies the input SAS data set.

Main discussion: "Input Data Sets" on page 19

Restriction: You cannot use PROC CHART with an engine that supports concurrent access if another user is updating the data set at the same time.

FORMCHAR <(*position(s)*)>='*formatting-character(s)*'
defines the characters to use for constructing the horizontal and vertical axes, reference lines, and other stuctural parts of a chart. It also defines the symbols to use to create the bars, blocks, or sections in the output.

position(s)
identifies the position of one or more characters in the SAS formatting-character string. A space or a comma separates the positions.

Default: Omitting (*position(s)*), is the same as specifying all 20 possible SAS formatting characters, in order.

Range: PROC CHART uses 6 of the 20 formatting characters that SAS provides. Table 7.1 on page 172 shows the formatting characters that PROC CHART uses. Figure 7.1 on page 172 illustrates the use of formatting characters commonly used in PROC CHART.

formatting-character(s)
lists the characters to use for the specified positions. PROC CHART assigns characters in *formatting-character(s)* to *position(s)*, in the order that they are listed. For instance, the following option assigns the asterisk (*) to the second formatting character, the pound sign (#) to the seventh character, and does not alter the remaining characters:

```
formchar(2,7)='*#'
```

Interaction: The SAS system option FORMCHAR= specifies the default formatting characters. The system option defines the entire string of formatting characters. The FORMCHAR= option in a procedure can redefine selected characters.

Tip: You can use any character in *formatting-characters*, including hexadecimal characters. If you use hexadecimal characters, you must put an **x** after the closing quote. For instance the following option assigns the hexadecimal character 2D to the second formatting character, the hexadecimal character 7C to the seventh character, and does not alter the remaining characters:

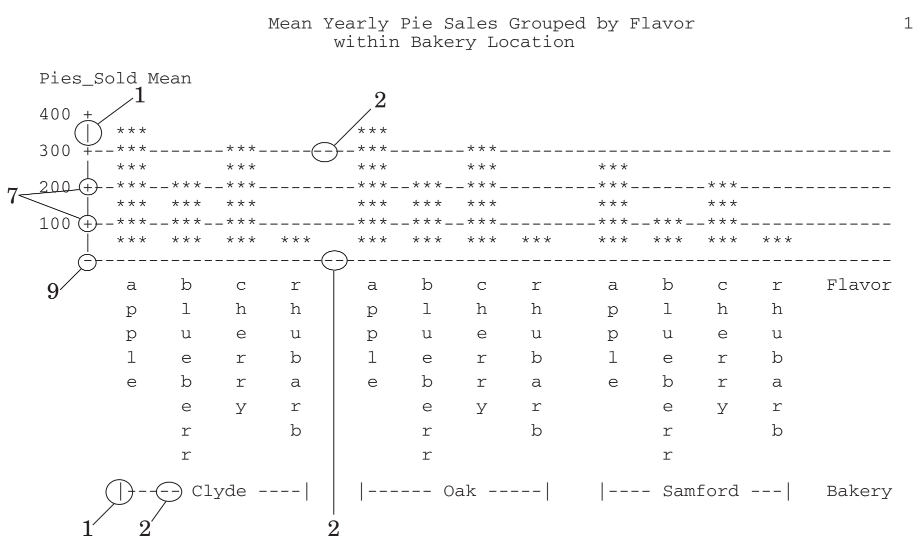
```
formchar(2,7)='2D7C'x
```

See also: For information on which hexadecimal codes to use for which characters, consult the documentation for your hardware.

Table 7.1 Formatting Characters Used by PROC CHART

Position ...	Default	Used to draw
1	|	Vertical axes in bar charts, the sides of the blocks in block charts, and reference lines in horizontal bar charts. In side-by-side bar charts, the first and second formatting characters appear around each value of the group variable (below the chart) to indicate the width of each group.
2	-	Horizontal axes in bar charts, the horizontal lines that separate the blocks in a block chart, and reference lines in vertical bar charts. In side-by-side bar charts, the first and second formatting characters appear around each value of the group variable (below the chart) to indicate the width of each group.
7	+	Tick marks in bar charts and the centers in pie and star charts.
9	-	Intersection of axes in bar charts.
16	/	Ends of blocks and the diagonal lines that separate blocks in a block chart.
20	*	Circles in pie and star charts.

Figure 7.1 Formatting Characters Commonly Used in PROC CHART Output



LPI=*value*

specifies the proportions of PIE and STAR charts. The *value* is determined by

$$(\text{lines per inch} / \text{columns per inch}) * 10$$

For example, if you have a printer with 8 lines per inch and 12 columns per inch, specify LPI=6.6667.

Default: 6

BLOCK Statement

Produces a block chart.

Featured in: Example 6 on page 194

BLOCK *variable(s)* </ *option(s)*>;

Required Arguments

variable(s)
specifies the variables for which PROC CHART produces a block chart, one chart for each variable.

Options

The options available on the BLOCK, HBAR, PIE, STAR, and VBAR statements are documented in "Customizing All Types of Charts" on page 177.

Statement Results

Because each block chart must fit on one output page, you may have to adjust the SAS system options LINESIZE= and PAGESIZE= if you have a large number of charted values for the BLOCK variable and for the variable specified in the GROUP= option.

Table 7.2 on page 173 shows the maximum number of charted values of BLOCK variables for selected LINESIZE= (LS=) specifications that can fit on a 66-line page.

Table 7.2 Maximum Number of Bars of BLOCK Variables

GROUP= Value	LS= 132	LS= 120	LS= 105	LS= 90	LS= 76	LS= 64
0,1	9	8	7	6	5	4
2	8	8	7	6	5	4
3	8	7	6	5	4	3
4	7	7	6	5	4	3
5,6	7	6	5	4	3	2

If the value of any GROUP= level is longer than three characters, the maximum number of charted values for the BLOCK variable that can fit may be reduced by one.

BLOCK level values truncate to 12 characters. If you exceed these limits, PROC CHART produces a horizontal bar chart instead.

BY Statement

Produces a separate chart for each BY group.

Main discussion: "BY" on page 54

Featured in: Example 6 on page 194

BY <DESCENDING> *variable-1*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are grouped in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

HBAR Statement

Produces a horizontal bar chart.

Tip: HBAR charts can print either the name or the label of the chart variable.

Featured in: Example 5 on page 192

HBAR *variable(s)* </ *option(s)*>;

Required Argument

variable(s)
specifies the variables for which PROC CHART produces a horizontal bar chart, one chart for each variable.

Options

The options available on the BLOCK, HBAR, PIE, STAR, and VBAR statements are documented in "Customizing All Types of Charts" on page 177.

Statement Results

Each chart occupies one or more output pages, depending on the number of bars; each bar occupies one line, by default.

By default, for horizontal bar charts of TYPE=FREQ, CFREQ, PCT, or CPCT, PROC CHART prints the following statistics: frequency, cumulative frequency, percentage, and cumulative percentage. If you use one or more of the statistics options, PROC CHART prints only the statistics that you request, plus the frequency.

PIE Statement

Produces a pie chart.

PIE *variable(s)* </ *option(s)*>;

Required Argument

variable(s)
specifies the variables for which PROC CHART produces a pie chart, one chart for each variable.

Options

The options available on the BLOCK, HBAR, PIE, STAR, and VBAR statements are documented in "Customizing All Types of Charts" on page 177.

Statement Results

PROC CHART determines the number of slices for the pie in the same way that it determines the number of bars for vertical bar charts. Any slices of the pie accounting for less than three print positions are grouped together into an "OTHER" category.

The pie's size is determined only by the SAS system options LINESIZE= and PAGESIZE=. By default, the pie looks elliptical if your printer does not print 6 lines per inch and 10 columns per inch. To make a circular pie chart on a printer that does not print 6 lines and 10 columns per inch, use the LPI= option on the PROC CHART statement. See the decription of LPI= on page 172 for the formula that gives you the proper LPI= value for your printer.

If you try to create a PIE chart for a variable with more than 50 levels, PROC CHART produces a horizontal bar chart instead.

STAR Statement

Produces a star chart.

STAR *variable(s)* </ *option(s)*>;

Required Argument

variable(s)

specifies the variables for which PROC CHART produces a star chart, one chart for each variable.

Options

The options available on the BLOCK, HBAR, PIE, STAR, and VBAR statements are documented in "Customizing All Types of Charts" on page 177.

Statement Results

The number of points in the star is determined in the same way as the number of bars for vertical bar charts.

If all the data have positive values, the center of the star represents zero and the outside circle represents the maximum value. If the data contain negative values, the center represents the minimum. See the description of the AXIS= option on page 178 for more information about how to specify maximum and minimum values. For information about how to specify the proportion of the chart, see the description of the LPI= option on page 172.

If you try to create a star chart for a variable with more than 24 levels, PROC CHART produces a horizontal bar chart instead.

VBAR Statement

Produces a vertical bar chart.

Featured in: Example 1 on page 184, Example 2 on page 186, Example 3 on page 187, Example 4 on page 190

VBAR *variable(s)* </ *option(s)*>;

Required Argument

variable(s)
specifies the variables for which PROC CHART produces a vertical bar chart, one chart for each variable.

Options

The options available on the BLOCK, HBAR, PIE, STAR, and VBAR statements are documented in "Customizing All Types of Charts" on page 177.

Statement Results

PROC CHART prints one page per chart. Along the vertical axis, PROC CHART describes the chart frequency, the cumulative frequency, the chart percentage, the cumulative percentage, the sum, or the mean. At the bottom of each bar, PROC CHART prints a value according to the value of the TYPE= option, if specified. For character variables or discrete numeric variables, this value is the actual value represented by the bar. For continuous numeric variables, the value gives the midpoint of the interval represented by the bar.

PROC CHART can automatically scale the vertical axis, determine the bar width, and choose spacing between the bars. However, by using options, you can choose bar intervals and the number of bars, include missing values in the chart, produce side-by-side charts, and subdivide the bars. If the number of characters per line (LINESIZE=) is not sufficient to display all vertical bars, PROC CHART produces a horizontal bar chart instead.

Customizing All Types of Charts

Many options in PROC CHART are valid in more than one statement. This section describes the options that you can use on the chart-producing statements.

To do this	Use this option
Specify that numeric variables are discrete	DISCRETE
Specify a frequency variable	FREQ=
Specify that missing values are valid levels	MISSING
Specify the variable for which values or means are displayed	SUMVAR=
Specify the statistic represented in the chart	TYPE=
Specify groupings	
Group the bars in side-by-side charts	GROUP=
Specify that group percentages sum to 100	G100
Group the bars in side-by-side charts	GROUP=
Specify the number of bars for continuous variables	LEVELS=

To do this	Use this option
Define ranges for continuous variables	MIDPOINTS=
Divide the bars into categories	SUBGROUP=
Compute statistics	
Compute the cumulative frequency for each bar	CFREQ
Compute the cumulative percentage for each bar	CPERCENT
Compute the frequency for each bar	FREQ
Compute the mean of the observations for each bar	MEAN
Compute the percentage of total observations for each bar	PERCENT
Compute the total number of observations for each bar	SUM
Control output format	
Print the bars in ascending order of size	ASCENDING
Specify the values for the response axis	AXIS=
Print the bars in descending order of size	DESCENDING
Specify extra space between groups of bars	GSPACE=
Suppress the default header line	NOHEADER
Allow no space between vertical bars	NOSPACE
Suppress the statistics	NOSTATS
Suppress the subgroup legend or symbol table	NOSYMBOL
Suppress the bars with zero frequency	NOZEROS
Draw reference lines	REF=
Specify the spaces between bars	SPACE=
Specify the symbols within bars or blocks	SYMBOL=
Specify the width of bars	WIDTH=

Options

ASCENDING
prints the bars and any associated statistics in ascending order of size within groups.

Alias: ASC

Restriction: Available only on the HBAR and VBAR statements

AXIS=*value-expression*
specifies the values for the response axis, where *value-expression* is a list of individual values, each separated by a space, or a range with a uniform interval for the values. For example, the following range specifies tick marks on a bar chart from 0 to 100 at intervals of 10:

```
hbar x / axis=0 to 100 by 10;
```

Restriction: Not available on the PIE statement

Restriction: Values must be uniformly spaced, even if you specify them individually.

Restriction: For frequency charts, values must be integers.

Interaction: For BLOCK charts, AXIS= sets the scale of the tallest block. To set the scale, PROC CHART uses the maximum value from the AXIS= list. If no value is greater than 0, PROC CHART ignores the AXIS= option.

Interaction: For HBAR and VBAR charts, AXIS= determines tick marks on the response axis. If the AXIS= specification contains only one value, the value determines the minimum tick mark if the value is less than 0, or determines the maximum tick mark if the value is greater than 0.

Interaction: For STAR charts, a single AXIS= value sets the minimum (the center of the chart) if the value is less than zero, or sets the maximum (the outside circle) if the value is greater than zero. If the AXIS= specification contains more than one value, PROC CHART uses the minimum and maximum values from the list.

Interaction: If you use AXIS= and the BY statement, PROC CHART produces uniform axes over BY groups.

CAUTION:

Values in *value-expression* override the range of the data. For example, if the data range is 1 to 10 and you specify a range of 3 to 5, only the data in the range 3 to 5 appear on the chart. Values out of range produce a warning message in the SAS log. △

CFREQ

prints the cumulative frequency.

Restriction: Available only on the HBAR statement

CPERCENT

prints the cumulative percentages.

Restriction: Available only on the HBAR statement

DESCENDING

prints the bars and any associated statistics in descending order of size within groups.

Alias: DESC

Restriction: Available only on the HBAR and VBAR statements

DISCRETE

specifies that a numeric chart variable is discrete rather than continuous. Without DISCRETE, PROC CHART assumes that all numeric variables are continuous and automatically chooses intervals for them unless you use MIDPOINTS= or LEVELS=.

FREQ

prints the frequency of each bar to the side of the chart.

Restriction: Available only on the HBAR statement

FREQ=*variable*

specifies a data set variable that represents a frequency count for each observation. Normally, each observation contributes a value of one to the frequency counts. With FREQ=, each observation contributes its value of the FREQ= value.

Restriction: If the FREQ= values are not integers, PROC CHART truncates them.

Interaction: If you use SUMVAR=, PROC CHART multiplies the sums by the FREQ= value.

GROUP=*variable*

produces side-by-side charts, with each chart representing the observations that have a common value for the GROUP= variable. The GROUP= variable can be character or numeric and is assumed to be discrete. For example, the following statement produces a frequency bar chart for men and women in each department:

```
vbar gender / group=dept;
```

Missing values for a GROUP= variable are treated as valid levels.

Restriction: Available only on the BLOCK, HBAR, and VBAR statements

Featured in: Example 4 on page 190, Example 5 on page 192, Example 6 on page 194

GSPACE=*n*

specifies the amount of extra space between groups of bars. Use GSPACE=0 to leave no extra space between adjacent groups of bars.

Restriction: Available only on the HBAR and VBAR statements

Interaction: PROC CHART ignores GSPACE= if you omit GROUP=

G100

specifies that the sum of percentages for each group equals 100. By default, PROC CHART uses 100 percent as the total sum. For example, if you produce a bar chart that separates males and females into three age categories, the six bars, by default, add to 100 percent; however, with G100, the three bars for females add to 100 percent, and the three bars for males add to 100 percent.

Restriction: Available only on the BLOCK, HBAR, and VBAR statements

Interaction: PROC CHART ignores G100 if you omit GROUP=.

LEVELS=*number-of-midpoints*

specifies the number of bars that represent each chart variable when the variables are continuous.

MEAN

prints the mean of the observations represented by each bar.

Restriction: Available only on the HBAR statement and only when you use SUMVAR= and TYPE=

Restriction: Not available when TYPE=CFREQ, CPERCENT, FREQ, or PERCENT

MIDPOINTS=*midpoint-specification* | OLD

defines the range of values that each bar, block, or section represents by specifying the range midpoints.

The value for MIDPOINTS= is one of the following:

midpoint-specification

specifies midpoints, either individually, or across a range at a uniform interval. For example, the following statement produces a chart with five bars; the first bar represents the range of values of X with a midpoint of 10, the second bar represents the range with a midpoint of 20, and so on:

```
vbar x / midpoints=10 20 30 40 50;
```

Here is an example of a midpoint specification for a character variable:

```
vbar x / midpoints='JAN' 'FEB' 'MAR';
```

Here is an example of specifying midpoints across a range at a uniform interval:

```
vbar x / midpoints=10 to 100 by 5;
```

OLD

specifies an algorithm that PROC CHART used in previous versions of SAS to choose midpoints for continuous variables. The old algorithm was based on the work of Nelder (1976). The current algorithm that PROC CHART uses if you omit OLD is based on the work of Terrell and Scott (1985).

Default: Without MIDPOINTS=, PROC CHART displays the values in the SAS System's normal sorted order.

Restriction: When the VBAR variables are numeric, the midpoints must be given in ascending order.

MISSING
specifies that missing values are valid levels for the chart variable.

NOHEADER
suppresses the default header line printed at the top of a chart.

Alias: NOHEADING

Restriction: Available only on the BLOCK, PIE, and STAR statements

Featured in: Example 6 on page 194

NOSTATS
suppresses the statistics on a horizontal bar chart.

Alias: NOSTAT

Restriction: Available only on the HBAR statement

NOSYMBOL
suppresses printing of the subgroup symbol or legend table.

Alias: NOLEGEND

Restriction: Available only on the BLOCK, HBAR, and VBAR statements

Interaction: PROC CHART ignores NOSYMBOL if you omit SUBGROUP=.

NOZEROS
suppresses any bar with zero frequency.

Restriction: Available only on the HBAR and VBAR statements

PERCENT
prints the percentages of observations having a given value for the chart variable.

Restriction: Available only on the HBAR statement

REF=*value(s)*
draws reference lines on the response axis at the specified positions.

Restriction: Available only on the HBAR and VBAR statements

Tip: The REF= values should correspond to values of the TYPE= statistic.

Featured in: Example 4 on page 190

SPACE=*n*
specifies the amount of space between individual bars.

Restriction: Available only on the HBAR and VBAR statements

Tip: Use SPACE=0 to leave no space between adjacent bars.

Tip: Use the GSPACE= option to specify the amount of space between the bars within each group.

SUBGROUP=*variable*
subdivides each bar or block into characters that show the contribution of the values of *variable* to that bar or block. PROC CHART uses the first character of each value to fill in the portion of the bar or block that corresponds to that value, unless more than one value begins with the same first character. In that case, PROC CHART uses the letters A, B, C, and so on to fill in the bars or blocks. If the variable is formatted, PROC CHART uses the first character of the formatted value.

The characters used in the chart and the values that they represent are given in a legend at the bottom of the chart. The subgroup symbols are ordered A through Z and 0 through 9 with the characters in ascending order.

PROC CHART calculates the height of a bar or block for each subgroup individually and then rounds the percentage of the total bar up or down. So the total height of the bar may be higher or lower than the same bar without the SUBGROUP= option.

Restriction: Available only on the BLOCK, HBAR, and VBAR statements

Interaction: If you use both TYPE=MEAN and SUBGROUP=, PROC CHART first calculates the mean for each variable listed in the SUMVAR= option, then subdivides the bar into the percentages contributed by each subgroup.

Featured in: Example 3 on page 187

SUM
prints the total number of observations that each bar represents.

Restriction: Available only on the HBAR statement and only when you use both SUMVAR= and TYPE=

Restriction: Not available when TYPE=CFREQ, CPERCENT, FREQ, or PERCENT

SUMVAR=*variable*
specifies the variable for which either values or means (depending on the value of TYPE=) PROC CHART displays in the chart.

Interaction: If you use SUMVAR= and you use TYPE= with a value other than MEAN or SUM, TYPE=SUM overrides the specified TYPE= value.

Tip: Both HBAR and VBAR charts can print labels for SUMVAR= variables if you use a LABEL statement.

Featured in: Example 3 on page 187, Example 4 on page 190, Example 5 on page 192, Example 6 on page 194

SYMBOL=*character(s)*
specifies the character or characters that PROC CHART uses in the bars or blocks of the chart when you do not use the SUBGROUP= option.

Default: asterisk (*)

Restriction: Available only on the BLOCK, HBAR, and VBAR statements

Interaction: If the SAS system option OVP is in effect and if your printing device supports overprinting, you can specify up to three characters to produce overprinted charts.

Featured in: Example 6 on page 194

TYPE=*statistic*
specifies what the bars or sections in the chart represent. The *statistic* is one of the following:

CFREQ
specifies that each bar, block, or section represent the cumulative frequency.

CPERCENT
specifies that each bar, block, or section represent the cumulative percentage.

Alias: CPCT

FREQ
specifies that each bar, block, or section represent the frequency with which a value or range occurs for the chart variable in the data.

MEAN
specifies that each bar, block, or section represent the mean of the SUMVAR= variable across all observations belonging to that bar, block, or section.

Interaction: With TYPE=MEAN, you can only compute MEAN and FREQ statistics.

Featured in: Example 4 on page 190

PERCENT

specifies that each bar, block, or section represent the percentage of observations that have a given value or that fall into a given range of the chart variable.

Alias: PCT

Featured in: Example 2 on page 186

SUM
specifies that each bar, block, or section represent the sum of the SUMVAR= variable for the observations corresponding to each bar, block, or section.

Default: FREQ (unless you use SUMVAR=, which causes a default of SUM)

Interaction: With TYPE=SUM, you can only compute SUM and FREQ statistics.

WIDTH=*n*
specifies the width of the bars on bar charts.

Restriction: Available only on the HBAR and VBAR statements

Concepts: CHART Procedure

Variable Characteristics

- Character variables and formats cannot exceed a length of 16.
- For continuous numeric variables, PROC CHART automatically selects display intervals, although you can explicitly define interval midpoints.
- For character variables and discrete numeric variables, which contain several distinct values rather than a continuous range, the data values themselves define the intervals.

Results: CHART Procedure

Missing Values

- Missing values are not considered as valid levels for the chart variable when you use the MISSING option.
- Missing values for a GROUP= or SUBGROUP= variable are treated as valid levels.
- PROC CHART ignores missing values for the FREQ= option and the SUMVAR= option.
- If the value of the FREQ= variable is missing, zero, or negative, the observation is excluded from the calculation of the chart statistic.
- If the value of the SUMVAR= variable is missing, the observation is excluded from the calculation of the chart statistic.

Examples: CHART Procedure

With PROC CHART, you can produce several types of charts within a single PROC step, but in this chapter, each example shows only one chart.

Example 1: Producing a Simple Frequency Count

Procedure features:
VBAR statement

This example produces a vertical bar chart that shows a frequency count for the values of the chart variable.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the input data set SHIRTS. The data set SHIRTS contains the sizes of a particular shirt that is sold during a week at a clothing store, with one observation for each shirt sold.

```
data shirts;
   input Size $ @@;
   datalines;
medium    large
large     large
large     medium
medium    small
small     medium
medium    large
small     medium
large     large
large     small
medium    medium
medium    medium
medium    large
small     small
;
```

Create a vertical bar chart with frequency counts. The VBAR statement produces a vertical bar chart for the frequency counts of the Size values.

```
proc chart data=shirts;
   vbar size;
   title 'Number of Each Shirt Size Sold';
run;
```

Output

The frequency chart shows the store's sales of the shirt for the week: 9 large shirts, 11 medium shirts, and 6 small shirts.

```
         Number of Each Shirt Size Sold                        1

Frequency

11 +                   *****
   |                   *****
   |                   *****
   |                   *****
10 +                   *****
   |                   *****
   |                   *****
   |                   *****
 9 +       *****       *****
   |       *****       *****
   |       *****       *****
   |       *****       *****
 8 +       *****       *****
   |       *****       *****
   |       *****       *****
   |       *****       *****
 7 +       *****       *****
   |       *****       *****
   |       *****       *****
   |       *****       *****
 6 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
 5 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
 4 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
 3 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
 2 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
 1 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   --------------------------------------------
           large      medium       small

                       Size
```

Example 2: Producing a Percentage Bar Chart

Procedure features:
VBAR statement option:
TYPE=
Data set: SHIRTS on page 184

This example produces a vertical bar chart. The chart statistic is the percentage for each category of the total number of shirts sold.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create a vertical bar chart with percentages. The VBAR statement produces a vertical bar chart. TYPE= specifies percentage as the chart statistic for the variable Size.

```
proc chart data=shirts;
   vbar size / type=percent;
   title 'Percentage of Total Sales for Each Shirt Size';
run;
```

Output

The chart shows the percentage of total sales for each shirt size. Of all the shirts sold, about 42.3 percent were medium, 34.6 were large, and 23.1 were small.

```
 Percentage of Total Sales for Each Shirt Size                1

Percentage

   |                   *****
   |                   *****
40 +                   *****
   |                   *****
   |                   *****
   |                   *****
   |                   *****
35 +       *****       *****
   |       *****       *****
   |       *****       *****
   |       *****       *****
   |       *****       *****
30 +       *****       *****
   |       *****       *****
   |       *****       *****
   |       *****       *****
   |       *****       *****
25 +       *****       *****
   |       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
20 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
15 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
10 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
 5 +       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   |       *****       *****       *****
   --------------------------------------------
           large      medium       small

                       Size
```

Example 3: Subdividing the Bars into Categories

Procedure features:

VBAR statement options:

SUBGROUP=
SUMVAR=

This example

- □ produces a vertical bar chart for categories of one variable with bar lengths that represent the values of another variable.
- □ subdivides each bar into categories based on the values of a third variable.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the input data set PIESALES. The PIESALES data set contains the number of each flavor of pie sold for two years at three bakeries owned by the same company – one bakery on Samford Avenue, one on Oak Street, and one on Clyde Drive.

```
data piesales;
   input Bakery $ Flavor $ Year Pies_Sold;
   datalines;
Samford  apple      1995  234
Samford  apple      1996  288
Samford  blueberry  1995  103
Samford  blueberry  1996  143
Samford  cherry     1995  173
Samford  cherry     1996  195
Samford  rhubarb    1995   26
Samford  rhubarb    1996   28
Oak      apple      1995  319
Oak      apple      1996  371
Oak      blueberry  1995  174
Oak      blueberry  1996  206
Oak      cherry     1995  246
Oak      cherry     1996  311
Oak      rhubarb    1995   51
Oak      rhubarb    1996   56
Clyde    apple      1995  313
Clyde    apple      1996  415
Clyde    blueberry  1995  177
Clyde    blueberry  1996  201
Clyde    cherry     1995  250
Clyde    cherry     1996  328
Clyde    rhubarb    1995   60
Clyde    rhubarb    1996   59
;
```

Create a vertical bar chart with the bars that are subdivided into categories. The VBAR statement produces a vertical bar chart with one bar for each pie flavor. SUBGROUP= divides each bar into sales for each bakery.

```
proc chart data=piesales;
   vbar flavor / subgroup=bakery
```

Specify the bar length variable. SUMVAR= specifies Pies_Sold as the variable whose values are represented by the lengths of the bars.

```
                  sumvar=pies_sold;
   title 'Pie Sales by Flavor Subdivided by Bakery Location';
run;
```

Output

The bar that represents the sales of apple pies, for example, shows 1,940 total pies across both years and all three bakeries. The symbol for the Samford Avenue bakery represents the 522 pies at the top, the symbol for the Oak Street bakery represents the 690 pies in the middle, and the symbol for the Clyde Drive bakery represents the 728 pies at the bottom of the bar for apple pies. By default, the labels along the horizontal axis are truncated to eight characters.

```
      Pie Sales by Flavor Subdivided by Bakery Location               1

Pies_Sold Sum

     |       SSSSS
     |       SSSSS
     |       SSSSS
1800 +       SSSSS
     |       SSSSS
     |       SSSSS
     |       SSSSS
1600 +       SSSSS
     |       SSSSS
     |       SSSSS                   SSSSS
     |       OOOOO                   SSSSS
1400 +       OOOOO                   SSSSS
     |       OOOOO                   SSSSS
     |       OOOOO                   SSSSS
     |       OOOOO                   SSSSS
1200 +       OOOOO                   SSSSS
     |       OOOOO                   OOOOO
     |       OOOOO                   OOOOO
     |       OOOOO       SSSSS       OOOOO
1000 +       OOOOO       SSSSS       OOOOO
     |       OOOOO       SSSSS       OOOOO
     |       OOOOO       SSSSS       OOOOO
     |       OOOOO       SSSSS       OOOOO
 800 +       OOOOO       OOOOO       OOOOO
     |       CCCCC       OOOOO       OOOOO
     |       CCCCC       OOOOO       OOOOO
     |       CCCCC       OOOOO       OOOOO
 600 +       CCCCC       OOOOO       CCCCC
     |       CCCCC       OOOOO       CCCCC
     |       CCCCC       OOOOO       CCCCC
     |       CCCCC       OOOOO       CCCCC
 400 +       CCCCC       CCCCC       CCCCC
     |       CCCCC       CCCCC       CCCCC
     |       CCCCC       CCCCC       CCCCC
     |       CCCCC       CCCCC       CCCCC       SSSSS
 200 +       CCCCC       CCCCC       CCCCC       OOOOO
     |       CCCCC       CCCCC       CCCCC       OOOOO
     |       CCCCC       CCCCC       CCCCC       CCCCC
     |       CCCCC       CCCCC       CCCCC       CCCCC
     --------------------------------------------------------
             apple     blueberr     cherry      rhubarb

                              Flavor


     Symbol Bakery      Symbol Bakery      Symbol Bakery

        C   Clyde          O   Oak            S   Samford
```

Example 4: Producing Side-by-Side Bar Charts

Procedure features:

VBAR statement options:

GROUP=
REF=
SUMVAR=
TYPE=

Data set: PIESALES on page 188

This example

- charts the mean values of a variable for the categories of another variable
- creates side-by-side bar charts for the categories of a third variable
- draws reference lines across the charts.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create a side-by-side vertical bar chart. The VBAR statement produces a side-by-side vertical bar chart to compare the sales across values of Bakery, specified by GROUP=. Each Bakery group contains a bar for each Flavor value.

```
proc chart data=piesales;
   vbar flavor / group=bakery
```

Create reference lines. REF= draws reference lines to mark pie sales at 100, 200, and 300.

```
                 ref=100 200 300
```

Specify the bar length variable. SUMVAR= specifies Pies_Sold as the variable that is represented by the lengths of the bars.

```
                 sumvar=pies_sold
```

Specify the statistical variable. TYPE= averages the sales for 1995 and 1996 for each combination of bakery and flavor.

```
                 type=mean;
   title  'Mean Yearly Pie Sales Grouped by Flavor';
   title2 'within Bakery Location';
run;
```

Output

The side-by-side bar charts compare the sales of apple pies, for example, across bakeries. The mean for the Clyde Drive bakery is 364, the mean for the Oak Street bakery is 345, and the mean for the Samford Avenue bakery is 261.

```
                   Mean Yearly Pie Sales Grouped by Flavor                    1
                            within Bakery Location

Pies_Sold Mean

    |  ***
350 +  ***                   ***
    |  ***                   ***
    |  ***                   ***
    |  ***                   ***
    |  ***                   ***
300 +--***-------------------***----------------------------------------------
    |  ***       ***         ***
    |  ***       ***         ***       ***
    |  ***       ***         ***       ***
    |  ***       ***         ***       ***         ***
250 +  ***       ***         ***       ***         ***
    |  ***       ***         ***       ***         ***
    |  ***       ***         ***       ***         ***
    |  ***       ***         ***       ***         ***
    |  ***       ***         ***       ***         ***
200 +--***-------***---------***-------***---------***------------------------
    |  ***  ***  ***         ***  ***  ***         ***
    |  ***  ***  ***         ***  ***  ***         ***       ***
    |  ***  ***  ***         ***  ***  ***         ***       ***
    |  ***  ***  ***         ***  ***  ***         ***       ***
150 +  ***  ***  ***         ***  ***  ***         ***       ***
    |  ***  ***  ***         ***  ***  ***         ***       ***
    |  ***  ***  ***         ***  ***  ***         ***       ***
    |  ***  ***  ***         ***  ***  ***         ***  ***  ***
    |  ***  ***  ***         ***  ***  ***         ***  ***  ***
100 +--***--***--***---------***--***--***---------***--***--***--------------
    |  ***  ***  ***         ***  ***  ***         ***  ***  ***
    |  ***  ***  ***         ***  ***  ***         ***  ***  ***
    |  ***  ***  ***         ***  ***  ***         ***  ***  ***
    |  ***  ***  ***  ***    ***  ***  ***         ***  ***  ***
 50 +  ***  ***  ***  ***    ***  ***  ***  ***    ***  ***  ***
    |  ***  ***  ***  ***    ***  ***  ***  ***    ***  ***  ***
    |  ***  ***  ***  ***    ***  ***  ***  ***    ***  ***  ***  ***
    |  ***  ***  ***  ***    ***  ***  ***  ***    ***  ***  ***  ***
    |  ***  ***  ***  ***    ***  ***  ***  ***    ***  ***  ***  ***
    --------------------------------------------------------------------------
        a    b    c    r      a    b    c    r      a    b    c    r    Flavor
        p    l    h    h      p    l    h    h      p    l    h    h
        p    u    e    u      p    u    e    u      p    u    e    u
        l    e    r    b      l    e    r    b      l    e    r    b
        e    b    r    a      e    b    r    a      e    b    r    a
             e    y    r           e    y    r           e    y    r
             r         b           r         b           r         b
             r                     r                     r

       |----- Clyde ----|    |------ Oak -----|    |---- Samford ---|   Bakery
```

Example 5: Producing a Horizontal Bar Chart for a Subset of the Data

Procedure features:

HBAR statement options:

GROUP=
SUMVAR=

Other features:
WHERE= data set option

Data set: PIESALES on page 188

This example

- produces horizontal bar charts only for observations with a common value
- charts the values of a variable for the categories of another variable
- creates side-by-side bar charts for the categories of a third variable.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Specify the variable value limitation for the horizontal bar chart. WHERE= limits the chart to only the 1995 sales totals.

```
proc chart data=piesales(where=(year=1995));
```

Create a side-by-side horizontal bar chart. The HBAR statement produces a side-by-side horizontal bar chart to compare sales across values of Flavor, specified by GROUP=. Each Flavor group contains a bar for each Bakery value.

```
   hbar bakery / group=flavor
```

Specify the bar length variable. SUMVAR= specifies Pies_Sold as the variable whose values are represented by the lengths of the bars.

```
   sumvar=pies_sold;
   title '1995 Pie Sales for Each Bakery According to Flavor';
run;
```

Output

```
                1995 Pie Sales for Each Bakery According to Flavor            1

  Flavor     Bakery                                                    Pies_Sold
                                                                             Sum
                       |
  apple      Clyde     |******************************************     313.0000
             Oak       |*******************************************    319.0000
             Samford   |*******************************                234.0000
                       |
  blueberr   Clyde     |************************                       177.0000
             Oak       |***********************                        174.0000
             Samford   |**************                                 103.0000
                       |
  cherry     Clyde     |*********************************              250.0000
             Oak       |*********************************              246.0000
             Samford   |***********************                        173.0000
                       |
  rhubarb    Clyde     |********                                        60.0000
             Oak       |*******                                         51.0000
             Samford   |***                                             26.0000
                       |
                       ----+---+---+---+---+---+---+---+---+---+---
                          30  60  90 120 150 180 210 240 270 300

                                      Pies_Sold Sum
```

Example 6: Producing Block Charts for BY Groups

Procedure features:

BLOCK statement options:

GROUP=
NOHEADER=
SUMVAR=
SYMBOL=

BY statement

Other features:

PROC SORT

SAS System options:

NOBYLINE
OVP

TITLE statement:

#BYVAL specification

Data set: PIESALES on page 188

This example

- □ sorts the data set
- □ produces a block chart for each BY group
- □ organizes the blocks into a three-dimensional chart

- prints BY group-specific titles.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Sort the input data set PIESALES. PROC SORT sorts PIESALES by year. This is required to produce a separate chart for each year.

```
proc sort data=piesales out=sorted_piesales;
   by year;
run;
```

Suppress BY lines and allow overprinted characters in the block charts. NOBYLINE suppresses the usual BY lines in the output. OVP allows overprinted characters in the charts.

```
options nobyline ovp;
```

Specify the BY group for multiple block charts. The BY statement produces one chart for 1995 sales and one for 1996 sales.

```
proc chart data=sorted_piesales;
   by year;
```

Create a block chart. The BLOCK statement produces a block chart for each year. Each chart contains a grid (Bakery values along the bottom, Flavor values along the side) of cells that contain the blocks.

```
   block bakery / group=flavor
```

Specify the bar length variable. SUMVAR= specifies Pies_Sold as the variable whose values are represented by the lengths of the blocks.

```
                  sumvar=pies_sold
```

Suppress the default header line. NOHEADER suppresses the default header line.

```
                  noheader
```

Specify the block symbols and create the chart titles. SYMBOL= specifies the symbols in the blocks.

```
                  symbol='OX';
   title  'Pie Sales for Each Bakery and Flavor';
```

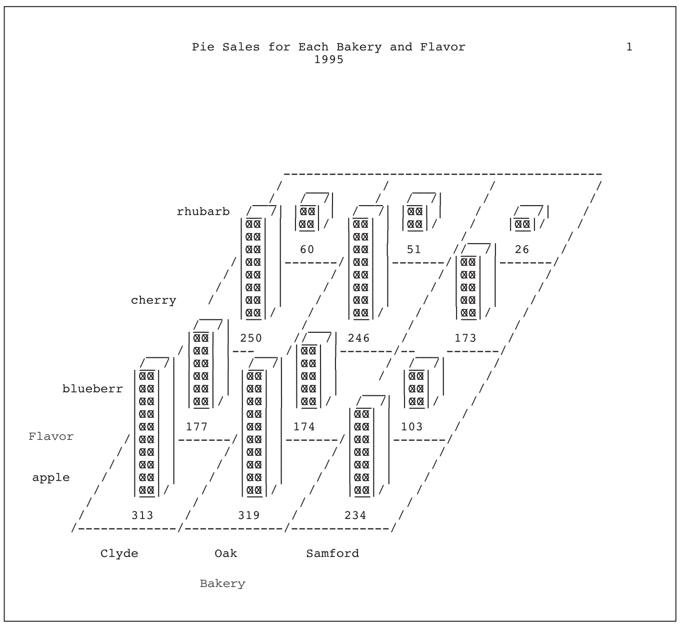
Create the second line of the title with an input variable. The #BYVAL specification inserts the year into the second line of the title.

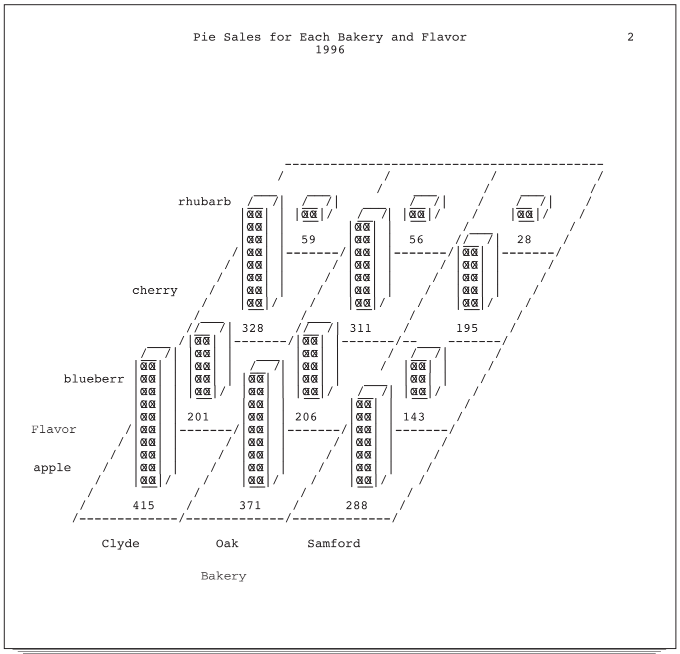
```
   title2 '#byval(year)';
run;
```

Reset the printing of the default BY line. The SAS system option BYLINE resets the printing of the default BY line.

```
options byline;
```

Output





References

Nelder, J.A. (1976), "A Simple Algorithm for Scaling Graphs," *Applied Statistics*, Volume 25, Number 1, London: The Royal Statistical Society.

Terrell, G.R. and Scott, D.W. (1985), "Oversmoothed Nonparametric Density Estimates," *Journal of the American Statistical Association*, 80, 389, 209–214.

CHAPTER 8

The CIMPORT Procedure

Overview: CIMPORT Procedure

The CIMPORT procedure *imports* a transport file that was created (*exported*) by the CPORT procedure. PROC CIMPORT restores the transport file to its original form as a SAS catalog, SAS data set, or SAS data library. *Transport files* are sequential files that each contain a SAS data library, a SAS catalog, or a SAS data set in transport format. The transport format that PROC CPORT writes is the same for all environments and for many releases of SAS.

PROC CIMPORT can read only transport files that PROC CPORT creates. For information on the transport files that the transport engine creates, see the section on SAS files in *SAS Language Reference: Concepts*.

PROC CIMPORT also *converts* SAS files, which means that it changes the format of a SAS file from the format appropriate for one version of SAS to the format appropriate for another version. For example, you can use PROC CPORT and PROC CIMPORT to move files from earlier releases of SAS to more recent releases. In such cases, PROC CIMPORT automatically converts the contents of the transport file as it imports it.

PROC CIMPORT produces no output, but it does write notes to the SAS log.

To export and import files, follow these steps:

1. Use PROC CPORT to export the SAS files that you want to transport.
2. If you are changing operating environments, move the transport file to the new machine by using either communications software or a magnetic medium.

   Note: If you use communications software to move the transport file, be sure that it treats the transport file as a *binary* file and that it modifies neither the attributes nor the contents of the file. △
3. Use PROC CIMPORT to translate the transport file into the format appropriate for the new operating environment or release.

Syntax: PROC CIMPORT

PROC CIMPORT *destination=libref* | <*libref.*>*member-name* <*option(s)*>;
EXCLUDE *SAS file(s)* | *catalog entry(s)*</ MEMTYPE=*mtype*></ ENTRYTYPE=*entry-type*>;
SELECT *SAS file(s)* | *catalog entry(s)*</ MEMTYPE=*mtype*></ ENTRYTYPE=*entry-type*>;

PROC CIMPORT Statement

PROC CIMPORT *destination=libref* | <*libref.*> *member-name*<*option(s)*>;

To do this	Use this option
Identify the input transport file	
Specify a previously defined fileref or the filename of the transport file to read	INFILE=
Read the input transport file from a tape	TAPE
Select files to import	
Exclude specified entry types from the import process	EET=
Specify entry types to import	ET=
Control the contents of the transport file	
Import a SAS file without changing the created and modified date and time	DATECOPY
Specify whether to extend by 1 byte the length of short numerics (less than 8 bytes) when you import them	EXTENDSN=
Specify that only data sets, only catalogs, or both, be moved when a library is imported	MEMTYPE=
Enable access to a locked catalog	FORCE
Create a new catalog for the imported transport file, and delete any existing catalog with the same name	NEW
Import SAS/AF PROGRAM and SCL entries without edit capability	NOEDIT
Suppress the importing of source code for SAS/AF entries that contain compiled SCL code	NOSRC

Required Arguments

destination=libref* | < *libref.* >*member-name

identifies the type of file to import and specifies the specific catalog, SAS data set, or SAS data library to import.

destination

identifies the file or files in the transport file as a single catalog, as a single SAS data set, or as the members of a SAS data library. The *destination* argument can be one of the following:

CATALOG | CAT | C

DATA | DS | D

LIBRARY | LIB | L

libref | <*libref.* > *member-name*

specifies the specific catalog, SAS data set, or SAS data library as the destination of the transport file. If the *destination* argument is CATALOG or DATA, you can specify both a *libref* and a member name. If the *libref* is omitted, PROC CIMPORT uses the default library as the *libref*, which is usually the WORK library. If the *destination* argument is LIBRARY, specify only a *libref*.

Options

DATECOPY

copies the SAS internal date and time when the SAS file was created and the date and time when it was last modified to the resulting destination file. Note that the operating environment date and time are not preserved.

Restriction: DATECOPY can be used only when the destination file uses the V8 or V9 engine.

Tip: You can alter the file creation date and time with the DTC= option on the MODIFY statement"MODIFY Statement" on page 366 in a PROC DATASETS step.

EET=(*etype(s)*)

excludes specified entry types from the import process. If the *etype* is a single entry type, then you can omit the parentheses. Separate multiple values with spaces.

Interaction: You cannot specify both the EET= option and the ET= option in the same PROC CIMPORT step.

ET=(*etype(s)*)

specifies the entry types to import. If the *etype* is a single entry type, then you can omit the parentheses. Separate multiple values with spaces.

Interaction: You cannot specify both the EET= option and the ET= option in the same PROC CIMPORT step.

EXTENDSN=YES | NO

specifies whether to extend by 1 byte the length of short numerics (fewer than 8 bytes) when you import them. You can avoid a loss of precision when you transport a short numeric in IBM format to IEEE format if you extend its length. You cannot extend the length of an 8-byte short numeric.

Default: YES

Restriction: This option applies only to data sets.

Tip: Do not store fractions as short numerics.

FORCE

enables access to a locked catalog. By default, PROC CIMPORT locks the catalog that it is updating to prevent other users from accessing the catalog while it is being updated. The FORCE option overrides this lock, which allows other users to access the catalog while it is being imported, or allows you to import a catalog that is currently being accessed by other users.

CAUTION:

The FORCE option can lead to unpredictable results. The FORCE option allows multiple users to access the same catalog entry simultaneously. △

INFILE=*fileref* | *'filename'*

specifies a previously defined fileref or the filename of the transport file to read. If you omit the INFILE= option, then PROC CIMPORT attempts to read from a transport file with the fileref SASCAT. If a fileref SASCAT does not exist, then PROC CIMPORT attempts to read from a file named SASCAT.DAT.

Alias: FILE=

Featured in: Example 1 on page 205.

MEMTYPE=*mtype*

specifies that only data sets, only catalogs, or both, be moved when a SAS library is imported. Values for *mtype* can be

ALL
both catalogs and data sets

CATALOG | CAT
catalogs

DATA | DS
SAS data sets

NEW

creates a new catalog to contain the contents of the imported transport file when the destination you specify has the same name as an existing catalog. NEW deletes any existing catalog with the same name as the one you specify as a destination for the import. If you do not specify NEW, and the destination you specify has the same name as an existing catalog, PROC CIMPORT appends the imported transport file to the existing catalog.

NOEDIT

imports SAS/AF PROGRAM and SCL entries without edit capability.

You obtain the same results if you create a new catalog to contain SCL code by using the MERGE statement with the NOEDIT option in the BUILD procedure of SAS/AF software.

Note: The NOEDIT option affects only SAS/AF PROGRAM and SCL entries. It does not affect FSEDIT SCREEN and FSVIEW FORMULA entries. △

Alias: NEDIT

NOSRC

suppresses the importing of source code for SAS/AF entries that contain compiled SCL code.

You obtain the same results if you create a new catalog to contain SCL code by using the MERGE statement with the NOSOURCE option in the BUILD procedure of SAS/AF software.

Alias: NSRC

Interaction: PROC CIMPORT ignores the NOSRC option if you use it with an entry type other than FRAME, PROGRAM, or SCL.

TAPE
reads the input transport file from a tape.

Default: PROC CIMPORT reads from disk.

EXCLUDE Statement

Excludes specified files or entries from the import process.

Tip: There is no limit to the number of EXCLUDE statements you can use in one invocation of PROC CIMPORT.

Interaction: You can use either EXCLUDE statements or SELECT statements in a PROC CIMPORT step, but not both.

EXCLUDE *SAS file(s)* | *catalog entry(s)*</ MEMTYPE=*mtype*></ ENTRYTYPE=*entry-type*>;

Required Arguments

SAS file(s) | catalog entry(s)
specifies either the name(s) of one or more SAS files or the name(s) of one or more catalog entries to be excluded from the import process. Specify SAS filenames if you import a data library; specify catalog entry names if you import an individual SAS catalog. Separate multiple filenames or entry names with a space. You can use shortcuts to list many like-named files in the EXCLUDE statement. For more information, see "Shortcuts for Specifying Lists of Variable Names" on page 24.

Options

ENTRYTYPE=*entry-type*
specifies a single entry type for the catalog entry(s) listed in the EXCLUDE statement. See *SAS Language Reference: Concepts* for a complete list of catalog entry types.

Restriction: ENTRYTYPE= is valid only when you import an individual SAS catalog.

Alias: ETYPE=, ET=

MEMTYPE=*mtype*
specifies a single member type for the SAS file(s) listed in the EXCLUDE statement. Values for *mtype* can be

ALL
both catalogs and data sets

CATALOG
catalogs

DATA
SAS data sets.

You can also specify the MEMTYPE= option, enclosed in parentheses, immediately after the name of a file. In parentheses, MEMTYPE= identifies the type of the filename that just precedes it. When you use this form of the option, it overrides the MEMTYPE= option that follows the slash in the EXCLUDE statement, but it must match the MEMTYPE= option in the PROC CIMPORT statement.

Restriction: MEMTYPE= is valid only when you import a SAS data library.

Alias: MTYPE=, MT=

Default: ALL

SELECT Statement

Specifies individual files or entries to import.

Tip: There is no limit to the number of SELECT statements you can use in one invocation of PROC CIMPORT.

Interaction: You can use either EXCLUDE statements or SELECT statements in a PROC CIMPORT step, but not both.

Featured in: Example 2 on page 206

SELECT *SAS file(s)* | *catalog entry(s)*</ MEMTYPE=*mtype*></ ENTRYTYPE=*entry-type*>;

Required Arguments

SAS file(s) | catalog entry(s)

specifies either the name(s) of one or more SAS files or the name(s) of one or more catalog entries to import. Specify SAS filenames if you import a data library; specify catalog entry names if you import an individual SAS catalog. Separate multiple filenames or entry names with a space. You can use shortcuts to list many like-named files in the SELECT statement. For more information, see “Shortcuts for Specifying Lists of Variable Names” on page 24.

Options

ENTRYTYPE=*entry-type*

specifies a single entry type for the catalog entry(s) listed in the SELECT statement. See *SAS Language Reference: Concepts* for a complete list of catalog entry types.

Restriction: ENTRYTYPE= is valid only when you import an individual SAS catalog.

Alias: ETYPE=, ET=

MEMTYPE=*mtype*

specifies a single member type for the SAS file(s) listed in the SELECT statement. Valid values are CATALOG or CAT, DATA, or ALL.

You can also specify the MEMTYPE= option, enclosed in parentheses, immediately after the name of a file. In parentheses, MEMTYPE= identifies the type of the filename that just precedes it. When you use this form of the option, it overrides the MEMTYPE= option that follows the slash in the SELECT statement, but it must match the MEMTYPE= option in the PROC CIMPORT statement.

Restriction: MEMTYPE= is valid only when you import a SAS data library.

Alias: MTYPE=, MT=

Default: ALL

Results: CIMPORT Procedure

Data Control Block Characteristics for Mainframe Environments

A common problem when you create or import a transport file under the OS/390 environment is a failure to specify the correct Data Control Block (DCB) characteristics. When you reference a transport file you must specify the following DCB characteristics:

LRECL: 80

BLKSIZE: 8000

RECFM: FB

Note: A BLKSIZE value of less than 8000 may be more efficient for your storage device in some cases. The BLKSIZE value must be an exact multiple of the LRECL value. △

Another common problem can occur if you use communications software to move files from another environment to OS/390. In some cases, the transport file does not have the proper DCB characteristics when it arrives on OS/390. If the communications software does not allow you to specify file characteristics, try the following approach for OS/390:

1 Create a file under OS/390 with the correct DCB characteristics and initialize the file.
2 Move the transport file from the other environment to the newly created file under OS/390 using binary transfer.

Examples: CIMPORT Procedure

Example 1: Importing an Entire Data Library

Procedure features:

PROC CIMPORT statement option:

INFILE=

This example shows how to use PROC CIMPORT to read from disk a transport file, named TRANFILE, that PROC CPORT created from a SAS data library in another operating environment. The transport file was moved to the new operating environment by means of communications software or magnetic medium. PROC CIMPORT imports the transport file to a SAS data library, called NEWLIB, in the new operating environment.

Program

Specify the library name and filename. The LIBNAME statement specifies a libname for the new SAS data library. The FILENAME statement specifies the filename of the transport file that PROC CPORT created and enables you to specify any operating environment options for file characteristics.

```
libname newlib 'SAS-data-library';
filename tranfile 'transport-file'
                    host-option(s)-for-file-characteristics;
```

Import the SAS data library in the NEWLIB library. PROC CIMPORT imports the SAS data library into the library named NEWLIB.

```
proc cimport library=newlib infile=tranfile;
run;
```

SAS Log

```
 NOTE: Proc CIMPORT begins to create/update catalog NEWLIB.FINANCE
NOTE: Entry LOAN.FRAME has been imported.
NOTE: Entry LOAN.HELP has been imported.
NOTE: Entry LOAN.KEYS has been imported.
NOTE: Entry LOAN.PMENU has been imported.
NOTE: Entry LOAN.SCL has been imported.
NOTE: Total number of entries processed in catalog NEWLIB.FINANCE: 5

NOTE: Proc CIMPORT begins to create/update catalog NEWLIB.FORMATS
NOTE: Entry REVENUE.FORMAT has been imported.
NOTE: Entry DEPT.FORMATC has been imported.
NOTE: Total number of entries processed in catalog NEWLIB.FORMATS: 2
```

Example 2: Importing Individual Catalog Entries

Procedure features:

PROC CIMPORT statement options:
- INFILE=

SELECT statement

This example shows how to use PROC CIMPORT to import the individual catalog entries LOAN.PMENU and LOAN.SCL from the transport file TRANS2, which was created from a single SAS catalog.

Program

Specify the library name, filename, and operating environment options. The LIBNAME statement specifies a libname for the new SAS data library. The FILENAME statement specifies the filename of the transport file that PROC CPORT created and enables you to specify any operating environment options for file characteristics.

```
libname newlib 'SAS-data-library';
filename trans2 'transport-file'
                 host-option(s)-for-file-characteristics;
```

Import the specified catalog entries to the new SAS catalog. PROC CIMPORT imports the individual catalog entries from the TRANS2 transport file and stores them in a new SAS catalog called NEWLIB.FINANCE. The SELECT statement selects only the two specified entries from the transport file to be imported into the new catalog.

```
proc cimport catalog=newlib.finance infile=trans2;
   select loan.pmenu loan.scl;
run;
```

SAS Log

```
NOTE: Proc CIMPORT begins to create/update catalog NEWLIB.FINANCE
NOTE: Entry LOAN.PMENU has been imported.
NOTE: Entry LOAN.SCL has been imported.
NOTE: Total number of entries processed in catalog NEWLIB.FINANCE: 2
```

Example 3: Importing a Single Indexed SAS Data Set

Procedure features:

PROC CIMPORT statement option:

INFILE=

This example shows how to use PROC CIMPORT to import an indexed SAS data set from a transport file that was created by PROC CPORT from a single SAS data set.

Program

Specify the library name, filename, and operating environment options. The LIBNAME statement specifies a libname for the new SAS data library. The FILENAME statement specifies the filename of the transport file that PROC CPORT created and enables you to specify any operating environment options for file characteristics.

```
libname newdata 'SAS-data-library';
filename trans3 'transport-file'
                 host-option(s)-for-file-characteristics;
```

Import the SAS data set. PROC CIMPORT imports the single SAS data set that you identify with the DATA= specification in the PROC CIMPORT statement. PROC CPORT exported the data set NEWDATA.TIMES in the transport file TRANS3.

```
proc cimport data=newdata.times infile=trans3;
run;
```

SAS Log

```
NOTE: Proc CIMPORT begins to create/update data set NEWDATA.TIMES
NOTE: The data set index x is defined.
NOTE: Data set contains 2 variables and 2 observations.
      Logical record length is 16
```

CHAPTER

9

The COMPARE Procedure

Overview: COMPARE Procedure

The COMPARE procedure compares the contents of two SAS data sets, selected variables in different data sets, or variables within the same data set.

PROC COMPARE compares two data sets: the *base data set* and the *comparison data set*. The procedure determines matching variables and matching observations. *Matching variables* are variables with the same name or variables that you explicitly pair by using the VAR and WITH statements. Matching variables must be of the same type. *Matching observations* are observations that have the same values for all ID variables that you specify or, if you do not use the ID statement, that occur in the same position in the data sets. If you match observations by ID variables, both data sets must be sorted by all ID variables.

When you compare data sets using PROC COMPARE, you receive the following type of information:

- whether matching variables have different values
- whether one data set has more observations than the other
- what variables the two data sets have in common
- how many variables are in one data set but not in the other
- whether matching variables have different formats, labels, or types.
- a comparison of the values of matching observations.

Further, PROC COMPARE creates two kinds of output data sets that give detailed information about the differences between observations of variables it is comparing.

The following example compares the data sets PROCLIB.ONE and PROCLIB.TWO, which contain similar data about students:

```
data proclib.one(label='First Data Set');
   input student year $ state $ gr1 gr2;
   label year='Year of Birth';
   format gr1 4.1;
   datalines;
1000 1970 NC 85 87
1042 1971 MD 92 92
1095 1969 PA 78 72
1187 1970 MA 87 94
;

data proclib.two(label='Second Data Set');
   input student $ year $ state $ gr1
         gr2 major $;
   label state='Home State';
   format gr1 5.2;
   datalines;
1000 1970 NC 84 87 Math
1042 1971 MA 92 92 History
1095 1969 PA 79 73 Physics
1187 1970 MD 87 74 Dance
1204 1971 NC 82 96 French
;
```

PROC COMPARE produces lengthy output. You can use one or more options to determine the kinds of comparisons to make and the degree of detail in the report. For example, in the following PROC COMPARE step, the NOVALUES option suppresses the part of the output that shows the differences in the values of matching variables:

```
proc compare base=proclib.one
             compare=proclib.two novalues;
run;
```

Output 9.1 Comparison of Two Data Sets

```
                              The SAS System                              1

                             COMPARE Procedure
                  Comparison of PROCLIB.ONE with PROCLIB.TWO
                               (Method=EXACT)

                             Data Set Summary

Dataset               Created          Modified  NVar    NObs  Label

PROCLIB.ONE  13MAY98:15:01:42  13MAY98:15:01:42     5       4  First Data Set
PROCLIB.TWO  13MAY98:15:01:44  13MAY98:15:01:44     6       5  Second Data Set


                             Variables Summary

              Number of Variables in Common: 5.
              Number of Variables in PROCLIB.TWO but not in PROCLIB.ONE: 1.
              Number of Variables with Conflicting Types: 1.
              Number of Variables with Differing Attributes: 3.


              Listing of Common Variables with Conflicting Types

                     Variable  Dataset      Type  Length

                     student   PROCLIB.ONE  Num        8
                               PROCLIB.TWO  Char       8


            Listing of Common Variables with Differing Attributes

          Variable  Dataset      Type  Length  Format  Label

          year      PROCLIB.ONE  Char       8          Year of Birth
                    PROCLIB.TWO  Char       8
          state     PROCLIB.ONE  Char       8
                    PROCLIB.TWO  Char       8          Home State
```

```
                             The SAS System                              2

                            COMPARE Procedure
                 Comparison of PROCLIB.ONE with PROCLIB.TWO
                              (Method=EXACT)

           Listing of Common Variables with Differing Attributes

         Variable  Dataset      Type  Length  Format  Label

         gr1       PROCLIB.ONE  Num        8  4.1
                   PROCLIB.TWO  Num        8  5.2


                           Observation Summary

                      Observation      Base  Compare

                      First Obs           1        1
                      First Unequal       1        1
                      Last  Unequal       4        4
                      Last  Match         4        4
                      Last  Obs           .        5

      Number of Observations in Common: 4.
      Number of Observations in PROCLIB.TWO but not in PROCLIB.ONE: 1.
      Total Number of Observations Read from PROCLIB.ONE: 4.
      Total Number of Observations Read from PROCLIB.TWO: 5.

      Number of Observations with Some Compared Variables Unequal: 4.
      Number of Observations with All Compared Variables Equal: 0.
```

```
                             The SAS System                              3

                            COMPARE Procedure
                 Comparison of PROCLIB.ONE with PROCLIB.TWO
                              (Method=EXACT)

                        Values Comparison Summary

      Number of Variables Compared with All Observations Equal: 1.
      Number of Variables Compared with Some Observations Unequal: 3.
      Total Number of Values which Compare Unequal: 6.
      Maximum Difference: 20.


                      Variables with Unequal Values

            Variable  Type  Len   Compare Label  Ndif   MaxDif

            state     CHAR    8   Home State        2
            gr1       NUM     8                     2    1.000
            gr2       NUM     8                     2   20.000
```

"Procedure Output" on page 230 shows the default output for these two data sets. Example 1 on page 239 shows the complete output for these two data sets.

Syntax: COMPARE Procedure

Restriction: You must use the VAR statement when you use the WITH statement.

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can use the LABEL, ATTRIB, FORMAT, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

PROC COMPARE <*option(s)*>;
  BY <DESCENDING> *variable-1*
    <…<DESCENDING> *variable-n*>
    <NOTSORTED>;
  ID <DESCENDING> *variable-1*
    <…<DESCENDING> *variable-n*>
    <NOTSORTED>;
  VAR *variable(s)*;
  WITH *variable(s)*;

To do this	Use this statement
Produce a separate comparison for each BY group	BY
Identify variables to use to match observations	ID
Restrict the comparison to values of specific variables	VAR
Compare variables of different names	WITH and VAR
Compare two variables in the same data set	WITH and VAR

PROC COMPARE Statement

Restriction: If you omit COMPARE=, you must use the WITH and VAR statements.

Restriction: PROC COMPARE reports errors differently if one or both of the compared data sets are not RADIX addressable. Version 6 compressed files are not RADIX addressable, while, beginning with Version 7, compressed files are RADIX addressable. (The integrity of the data is not compromised; the procedure simply numbers the observations differently.)

Reminder: You can use data set options with the BASE= and COMPARE= options.

PROC COMPARE <*option(s)*>;

To do this	Use this option
Specify the data sets to compare	
Specify the base data set	BASE=
Specify the comparison data set	COMPARE=
Control the output data set	
Create an output data set	OUT=
Write an observation for each observation in the BASE= and COMPARE= data sets	OUTALL
Write an observation for each observation in the BASE= data set	OUTBASE
Write an observation for each observation in the COMPARE= data set	OUTCOMP
Write an observation that contains the differences for each pair of matching observations	OUTDIF
Suppress the writing of observations when all values are equal	OUTNOEQUAL
Write an observation that contains the percent differences for each pair of matching observations	OUTPERCENT
Create an output data set that contains summary statistics	OUTSTATS=
Specify how the values are compared	
Specify the criterion for judging the equality of numeric values	CRITERION=
Specify the method for judging the equality of numeric values	METHOD=
Judge missing values equal to any value	NOMISSBASE and NOMISSCOMP
Control the details in the default report	
Include the values for all matching observations	ALLOBS
Print a table of summary statistics for all pairs of matching variables	ALLSTATS and STATS
Include in the report the values and differences for all matching variables	ALLVARS
Print only a short comparison summary	BRIEFSUMMARY
Change the report for numbers between 0 and 1	FUZZ=
Restrict the number of differences to print	MAXPRINT=
Suppress the print of creation and last-modified dates	NODATE
Suppress all printed output	NOPRINT
Suppress the summary reports	NOSUMMARY
Suppress the value comparison results.	NOVALUES
Produce a complete listing of values and differences	PRINTALL

To do this	Use this option
Print the value differences by observation, not by variable	TRANSPOSE
Control the listing of variables and observations	
List all variables and observations found in only one data set	LISTALL
List all variables and observations found only in the base data set	LISTBASE
List all observations found only in the base data set	LISTBASEOBS
List all variables found only in the base data set	LISTBASEVAR
List all variables and observations found only in the comparison data set	LISTCOMP
List all observations found only in the comparison data set	LISTCOMPOBS
List all variables found only in the comparison data set	LISTCOMPVAR
List variables whose values are judged equal	LISTEQUALVAR
List all observations found in only one data set	LISTOBS
List all variables found in only one data set	LISTVAR

Options

ALLOBS
includes in the report of value comparison results the values and, for numeric variables, the differences for all matching observations, even if they are judged equal.

Default: If you omit ALLOBS, PROC COMPARE prints values only for observations that are judged unequal.

Interaction: When used with the TRANSPOSE option, ALLOBS invokes the ALLVARS option and displays the values for all matching observations and variables.

ALLSTATS
prints a table of summary statistics for all pairs of matching variables.

See also: "Table of Summary Statistics" on page 233 for information on the statistics produced

ALLVARS
includes in the report of value comparison results the values and, for numeric variables, the differences for all pairs of matching variables, even if they are judged equal.

Default: If you omit ALLVARS, PROC COMPARE prints values only for variables that are judged unequal.

Interaction: When used with the TRANSPOSE option, ALLVARS displays unequal values in context with the values for other matching variables. If you omit the TRANSPOSE option, ALLVARS invokes the ALLOBS option and displays the values for all matching observations and variables.

BASE=*SAS-data-set*
specifies the data set to use as the base data set.

Alias: DATA=

Default: the most recently created SAS data set

Tip: You can use the WHERE= data set option with the BASE= option to limit the observations that are available for comparison.

BRIEFSUMMARY
produces a short comparison summary and suppresses the four default summary reports (data set summary report, variables summary report, observation summary report, and values comparison summary report).

Alias: BRIEF

Tip: By default, a listing of value differences accompanies the summary reports. To suppress this listing, use the NOVALUES option.

Featured in: Example 4 on page 246

COMPARE=*SAS-data-set*
specifies the data set to use as the comparison data set.

Aliases: COMP=, C=

Default: If you omit COMPARE=, the comparison data set is the same as the base data set, and PROC COMPARE compares variables within the data set.

Restriction: If you omit COMPARE=, you must use the WITH statement.

Tip: You can use the WHERE= data set option with COMPARE= to limit the observations that are available for comparison.

CRITERION=γ
specifies the criterion for judging the equality of numeric values. Normally, the value of γ (gamma) is positive, in which case the number itself becomes the equality criterion. If you use a negative value for γ, PROC COMPARE uses an equality criterion proportional to the precision of the computer on which SAS is running.

Default: 0.00001

See also: "The Equality Criterion" on page 226 for more information

ERROR
displays an error message in the SAS log when differences are found.

Interaction: This option overrides the WARNING option.

FUZZ=*number*
alters the values comparison results for numbers less than *number*. PROC COMPARE prints

- 0 for any variable value that is less than *number*
- a blank for difference or percent difference if it is less than *number*
- 0 for any summary statistic that is less than *number*.

Default 0

Range: 0 - 1

Tip: A report that contains many trivial differences is easier to read in this form.

LISTALL
lists all variables and observations that are found in only one data set.

Alias LIST

Interaction: using LISTALL is equivalent to using the following four options: LISTBASEOBS, LISTCOMPOBS, LISTBASEVAR, and LISTCOMPVAR.

LISTBASE
lists all observations and variables that are found in the base data set but not in the comparison data set.

Interaction: Using LISTBASE is equivalent to using the LISTBASEOBS and LISTBASEVAR options.

LISTBASEOBS
lists all observations that are found in the base data set but not in the comparison data set.

LISTBASEVAR
lists all variables that are found in the base data set but not in the comparison data set.

LISTCOMP
lists all observations and variables that are found in the comparison data set but not in the base data set.

Interaction: Using LISTCOMP is equivalent to using the LISTCOMPOBS and LISTCOMPVAR options.

LISTCOMPOBS
lists all observations that are found in the comparison data set but not in the base data set.

LISTCOMPVAR
lists all variables that are found in the comparison data set but not in the base data set.

LISTEQUALVAR
prints a list of variables whose values are judged equal at all observations in addition to the default list of variables whose values are judged unequal.

LISTOBS
lists all observations that are found in only one data set.

Interaction: Using LISTOBS is equivalent to using the LISTBASEOBS and LISTCOMPOBS options.

LISTVAR
lists all variables that are found in only one data set.

Interaction: Using LISTVAR is equivalent to using both the LISTBASEVAR and LISTCOMPVAR options.

MAXPRINT=*total* | (*per-variable, total*)
specifies the maximum number of differences to print, where

total
is the maximum total number of differences to print. The default value is 500 unless you use the ALLOBS option (or both the ALLVAR and TRANSPOSE options), in which case the default is 32000.

per-variable
is the maximum number of differences to print for each variable within a BY group. The default value is 50 unless you use the ALLOBS option (or both the ALLVAR and TRANSPOSE options), in which case the default is 1000.

The MAXPRINT= option prevents the output from becoming extremely large when data sets differ greatly.

METHOD=ABSOLUTE | EXACT | PERCENT | RELATIVE<(δ)>
specifies the method for judging the equality of numeric values. The constant δ (delta) is a number between 0 and 1 that specifies a value to add to the denominator when calculating the equality measure. By default, δ is 0.

Unless you use the CRITERION= option, the default method is EXACT. If you use CRITERION=, the default method is RELATIVE(ϕ), where ϕ (phi) is a small number that depends on the numerical precision of the computer on which SAS is running and on the value of CRITERION=.

See also: "The Equality Criterion" on page 226

NODATE

suppresses the display in the data set summary report of the creation dates and the last modified dates of the base and comparison data sets.

NOMISSBASE

judges a missing value in the base data set equal to any value. (By default, a missing value is equal only to a missing value of the same kind, that is .=., .^=.A, .A=.A, .A^=.B, and so on.)

You can use this option to determine the changes that would be made to the observations in the comparison data set if it were used as the master data set and the base data set were used as the transaction data set in a DATA step UPDATE statement. For information on the UPDATE statement, see the chapter on SAS language statements in *SAS Language Reference: Dictionary*.

NOMISSCOMP

judges a missing value in the comparison data set equal to any value. (By default, a missing value is equal only to a missing value of the same kind, that is .=., .^=.A, .A=.A, .A^=.B, and so on.)

You can use this option to determine the changes that would be made to the observations in the base data set if it were used as the master data set and the comparison data set were used as the transaction data set in a DATA step UPDATE statement. For information on the UPDATE statement, see the chapter on SAS language statements in *SAS Language Reference: Dictionary*.

NOMISSING

judges missing values in both the base and comparison data sets equal to any value. By default, a missing value is only equal to a missing value of the same kind, that is .=., .^=.A, .A=.A, .A^=.B, and so on.

Alias: NOMISS

Interaction: Using NOMISSING is equivalent to using both NOMISSBASE and NOMISSCOMP.

NOPRINT

suppresses all printed output.

Tip: You may want to use this option when you are creating one or more output data sets.

Featured in: Example 6 on page 251

NOSUMMARY

suppresses the data set, variable, observation, and values comparison summary reports.

Tips: NOSUMMARY produces no output if there are no differences in the matching values.

Featured in: Example 2 on page 243

NOTE

displays notes in the SAS log describing the results of the comparison, whether or not differences were found.

NOVALUES

suppresses the report of the value comparison results.

Featured in: “Overview: COMPARE Procedure” on page 209

OUT=*SAS-data-set*
names the output data set. If *SAS-data-set* does not exist, PROC COMPARE creates it. *SAS-data-set* contains the differences between matching variables.

See also: “Output Data Set (OUT=)” on page 236

Featured in: Example 6 on page 251

OUTALL
writes an observation to the output data set for each observation in the base data set and for each observation in the comparison data set. The option also writes observations to the output data set containing the differences and percent differences between the values in matching observations.

Tip: Using OUTALL is equivalent to using the following four options: OUTBASE, OUTCOMP, OUTDIF, and OUTPERCENT.

See also: “Output Data Set (OUT=)” on page 236

OUTBASE
writes an observation to the output data set for each observation in the base data set, creating observations in which _TYPE_=BASE.

See also: “Output Data Set (OUT=)” on page 236

Featured in: Example 6 on page 251

OUTCOMP
writes an observation to the output data set for each observation in the comparison data set, creating observations in which _TYPE_=COMP.

See also: “Output Data Set (OUT=)” on page 236

Featured in: Example 6 on page 251

OUTDIF
writes an observation to the output data set for each pair of matching observations. The values in the observation include values for the differences between the values in the pair of observations. The value of _TYPE_ in each observation is DIF.

Default: The OUTDIF option is the default unless you specify the OUTBASE, OUTCOMP, or OUTPERCENT option. If you use any of these options, you must explicitly specify the OUTDIF option to create _TYPE_=DIF observations in the output data set.

See also: “Output Data Set (OUT=)” on page 236

Featured in: Example 6 on page 251

OUTNOEQUAL
suppresses the writing of an observation to the output data set when all values in the observation are judged equal. In addition, in observations containing values for some variables judged equal and others judged unequal, the OUTNOEQUAL option uses the special missing value ".E" to represent differences and percent differences for variables judged equal.

See also: “Output Data Set (OUT=)” on page 236

Featured in: Example 6 on page 251

OUTPERCENT
writes an observation to the output data set for each pair of matching observations. The values in the observation include values for the percent differences between the values in the pair of observations. The value of _TYPE_ in each observation is PERCENT.

See also: “Output Data Set (OUT=)” on page 236

OUTSTATS=*SAS-data-set*
writes summary statistics for all pairs of matching variables to the specified *SAS-data-set*.

Tip: If you want to print a table of statistics in the procedure output, use the STATS, ALLSTATS, or PRINTALL option.

See also: "Output Statistics Data Set (OUTSTATS=)" on page 237 and "Table of Summary Statistics" on page 233.

Featured in: Example 7 on page 253

PRINTALL
invokes the following options: ALLVARS, ALLOBS, ALLSTATS, LISTALL, and WARNING.

Featured in: Example 1 on page 239

STATS
prints a table of summary statistics for all pairs of matching numeric variables that are judged unequal.

See also: "Table of Summary Statistics" on page 233 for information on the statistics produced.

TRANSPOSE
prints the reports of value differences by observation instead of by variable.

Interaction: If you also use the NOVALUES option, the TRANSPOSE option lists only the *names* of the variables whose values compare as unequal for each observation, not the values and differences.

See also: "Comparison Results for Observations (Using the TRANSPOSE Option)" on page 235.

WARNING
displays a warning message in the SAS log when differences are found.

Interaction: The ERROR option overrides the WARNING option.

BY Statement

Produces a separate comparison for each BY group.

Main discussion: "BY" on page 54

BY <DESCENDING> *variable-1*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must be sorted by all the variables that you specify. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are grouped in another way, for example, chronological order.

The requirement for ordering observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

BY Processing with PROC COMPARE

To use a BY statement with PROC COMPARE, you must sort both the base and comparison data sets by the BY variables. The nature of the comparison depends on whether all BY variables are in the comparison data set and, if they are, whether their attributes match those of the BY variables in the base data set. The following table shows how PROC COMPARE behaves under different circumstances:

Condition	Behavior of PROC COMPARE
All BY variables are in the comparison data set and all attributes match exactly	Compares corresponding BY groups
None of the BY variables are in the comparison data set	Compares each BY group in the base data set with the entire comparison data set
Some BY variables are not in the comparison data set	Writes an error message to the SAS log and terminates
Some BY variables have different types in the two data sets	Writes an error message to the SAS log and terminates

ID Statement

Lists variables to use to match observations.

See also: "A Comparison with an ID Variable" on page 225
Featured in: Example 5 on page 248

ID <DESCENDING> *variable-1*
<…<DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to match observations. You can specify more than one variable, but the data set must be sorted by the variable or variables you specify. These variables are *ID variables*. ID variables also identify observations on the printed reports and in the output data set.

Options

DESCENDING
specifies that the data set is sorted in descending order by the variable that immediately follows the word DESCENDING in the ID statement.

If you use the DESCENDING option, you must sort the data sets. SAS does not use an index to process an ID statement with the DESCENDING option. Further, the use of DESCENDING for ID variables must correspond to the use of the DESCENDING option in the BY statement in the PROC SORT step that was used to sort the data sets.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The data are grouped in another way, for example, chronological order.

See also: "Comparing Unsorted Data" on page 222

Requirements for ID Variables

- ID variables must be in the BASE= data set or PROC COMPARE stops processing.
- If an ID variable is not in the COMPARE= data set, PROC COMPARE prints a warning to the SAS log and does not use that variable to match observations in the comparison data set (but does write it to the OUT= data set).
- ID variables must be of the same type in both data sets.
- You should sort both data sets by the common ID variables (within the BY variables, if any) unless you specify the NOTSORTED option.

Comparing Unsorted Data

If you do not want to sort the data set by the ID variables, you can use the NOTSORTED option. When you specify the NOTSORTED option, or if the ID statement is omitted, PROC COMPARE matches the observations one-to-one. That is, PROC COMPARE matches the first observation in the base data set with the first observation in the comparison data set, the second with the second, and so on. If you use NOTSORTED, and the ID values of corresponding observations are not the same, PROC COMPARE prints an error message and stops processing.

If the data sets are not sorted by the common ID variables and you do not specify the NOTSORTED option, PROC COMPARE prints a warning message and continues to process the data sets as if you had specified NOTSORTED.

Avoiding Duplicate ID Values

The observations in each data set should be uniquely labeled by the values of the ID variables. If PROC COMPARE finds two successive observations with the same ID values in a data set, it

- prints the warning **`Duplicate Observations`** for the first occurrence for that data set

- prints the total number of duplicate observations found in the data set in the observation summary report
- uses the first observation with the duplicate value for the comparison.

When the data sets are not sorted, PROC COMPARE detects only those duplicate observations that occur in succession.

VAR Statement

Restricts the comparison of the values of variables to those named in the VAR statement.

Featured in: Example 2 on page 243, Example 3 on page 244, and Example 4 on page 246

VAR *variable(s)*;

Required Arguments

variable(s)
one or more variables that appear in the BASE= and COMPARE= data sets or only in the BASE= data set.

Details

- If you do not use the VAR statement, PROC COMPARE compares the values of all matching variables except those appearing in BY and ID statements.
- If a variable in the VAR statement does not exist in the COMPARE= data set, PROC COMPARE writes a warning to the SAS log and ignores the variable.
- If a variable in the VAR statement does not exist in the BASE= data set, PROC COMPARE stops processing and gives an error message.
- The VAR statement restricts only the comparison of values of matching variables. PROC COMPARE still reports on the total number of matching variables and compares their attributes. However, it produces neither error nor warning messages about these variables.

WITH Statement

Compares variables in the base data set with variables that have different names in the comparison data set, and compares different variables that are in the same data set.

Restriction: You must use the VAR statement when you use the WITH statement.

Featured in: Example 2 on page 243, Example 3 on page 244, and Example 4 on page 246

WITH *variable(s)*;

Required Arguments

variable(s)
one or more variables to compare with variables in the VAR statement.

Comparing Selected Variables

If you want to compare variables in the base data set with variables with different names in the comparison data set, specify the names of the variables in the base data set in the VAR statement and the names of the matching variables in the WITH statement. The first variable that you list in the WITH statement corresponds to the first variable that you list in the VAR statement, the second with the second, and so on. If the WITH statement list is shorter than the VAR statement list, PROC COMPARE assumes that the extra variables in the VAR statement have the same names in the comparison data set as they do in the base data set. If the WITH statement list is longer than the VAR statement list, PROC COMPARE ignores the extra variables.

A variable name can appear any number of times in the VAR statement or the WITH statement. By selecting VAR and WITH statement lists, you can compare the variables in any permutation.

If you omit the COMPARE= option in the PROC COMPARE statement, you must use the WITH statement. In this case, PROC COMPARE compares the values of variables with different names in the BASE= data set.

Concepts: COMPARE Procedure

PROC COMPARE first compares the following:

- data set attributes (set by the data set options TYPE= and LABEL=).
- variables. PROC COMPARE checks each variable in one data set to determine whether it matches a variable in the other data set.
- attributes (type, length, labels, formats, and informats) of matching variables.
- observations. PROC COMPARE checks each observation in one data set to determine whether it matches an observation in the other data set. PROC COMPARE either matches observations by their position in the data sets or by the values of the ID variable.

After making these comparisons, PROC COMPARE compares the values in the parts of the data sets that match. PROC COMPARE either compares the data by the position of observations or by the values of an ID variable.

A Comparison by Position of Observations

Figure 9.1 on page 225 shows two data sets. The data inside the shaded boxes show the part of the data sets that the procedure compares. Assume that variables with the same names have the same type.

Figure 9.1 Comparison by the Positions of Observations

Data Set ONE

IDNUM	NAME	GENDER	GPA
2998	Bagwell	f	3.722
9866	Metcalf	m	3.342
2118	Gray	f	3.177
3847	Baglione	f	4.000
2342	Hall	m	3.574

Data Set TWO

IDNUM	NAME	GENDER	GPA	YEAR
2998	Bagwell	f	3.722	2
9866	Metcalf	m	3.342	2
2118	Gray	f	3.177	3
3847	Baglione	f	4.000	4
2342	Hall	m	3.574	4
7565	Gold	f	3.609	2
1755	Syme	f	3.883	3

When you use PROC COMPARE to compare data set TWO with data set ONE, the procedure compares the first observation in data set ONE with the first observation in data set TWO, and it compares the second observation in the first data set with the second observation in the second data set, and so on. In each observation that it compares, the procedure compares the values of the IDNUM, NAME, GENDER, and GPA.

The procedure does not report on the values of the last two observations or the variable YEAR in data set TWO because there is nothing to compare them with in data set ONE.

A Comparison with an ID Variable

In a simple comparison, PROC COMPARE uses the observation number to determine which observations to compare. When you use an ID variable, PROC COMPARE uses the values of the ID variable to determine which observations to compare. ID variables should have unique values and must have the same type.

For the two data sets shown in Figure 9.2 on page 226, assume that IDNUM is an ID variable and that IDNUM has the same type in both data sets. The procedure compares the observations that have the same value for IDNUM. The data inside the shaded boxes show the part of the data sets that the procedure compares.

Figure 9.2 Comparison by the Value of the ID Variable

Data Set ONE

IDNUM	NAME	GENDER	GPA
2998	Bagwell	f	3.722
9866	Metcalf	m	3.342
2118	Gray	f	3.177
3847	Baglione	f	4.000
2342	Hall	m	3.574

Data Set TWO

IDNUM	NAME	GENDER	GPA	YEAR
2998	Bagwell	f	3.722	2
9866	Metcalf	m	3.342	2
2118	Gray	f	3.177	3
3847	Baglione	f	4.000	4
2342	Hall	m	3.574	4
7565	Gold	f	3.609	2
1755	Syme	f	3.883	3

The data sets contain three matching variables: NAME, GENDER, and GPA. They also contain five matching observations: the observations with values of **2998**, **9866**, **2118**, **3847**, and **2342** for IDNUM.

Data Set TWO contains two observations (IDNUM=**7565** and IDNUM=**1755**) for which data set ONE contains no matching observations. Similarly, no variable in data set ONE matches the variable YEAR in data set TWO.

See Example 5 on page 248 for an example that uses an ID variable.

The Equality Criterion

The COMPARE procedure judges numeric values unequal if the magnitude of their difference, as measured according to the METHOD= option, is greater than the value of the CRITERION= option. PROC COMPARE provides four methods for applying CRITERION=:

- The EXACT method tests for exact equality.
- The ABSOLUTE method compares the absolute difference to the value specified by CRITERION=.
- The RELATIVE method compares the absolute relative difference to the value specified by CRITERION=.
- The PERCENT method compares the absolute percent difference to the value specified by CRITERION=.

For a numeric variable compared, let x be its value in the base data set and let y be its value in the comparison data set. If both x and y are nonmissing, the values are judged unequal according to the value of METHOD= and the value of CRITERION= (γ) as follows:

- If METHOD=EXACT, the values are unequal if y does not equal x.
- If METHOD=ABSOLUTE, the values are unequal if

$$\mathrm{ABS}\,(y-x) > \gamma$$

- If METHOD=RELATIVE, the values are unequal if

$$\mathrm{ABS}\,(y-x)\,/\,((\mathrm{ABS}\,(x)+\mathrm{ABS}\,(y))\,/2+\delta) > \gamma$$

  The values are equal if *x*=*y*=0.
- If METHOD=PERCENT, the values are unequal if

$$100\,(\mathrm{ABS}\,(y-x)\,/\mathrm{ABS}\,(x)) > \gamma \;\; \mathrm{for} \;\; x \neq 0$$

  or

$$y \neq 0 \;\; \mathrm{for} \;\; \mathrm{x} = 0 \;\; .$$

If *x* or *y* is missing, then the comparison depends on the NOMISSING option. If NOMISSING is in effect, a missing value will always compare equal to anything. Otherwise, a missing value is judged equal only to a missing value of the same type, (that is, .=., .^=.A, .A=.A, .A^=.B, and so on).

If the value specified for CRITERION= is negative, the actual criterion used is made equal to the absolute value of γ times a very small number ϵ (epsilon) that depends on the numerical precision of the computer. This number ϵ is defined as the smallest positive floating-point value such that, using machine arithmetic, $1-\epsilon<1<1+\epsilon$. Round-off or truncation error in floating-point computations is typically a few orders of magnitude larger than ϵ. This means that CRITERION=–1000 often provides a reasonable test of the equality of computed results at the machine level of precision.

The value δ added to the denominator in the RELATIVE method is specified in parentheses after the method name: METHOD=RELATIVE(δ). If not specified in METHOD=, δ defaults to 0. The value of δ can be used to control the behavior of the error measure when both *x* and *y* are very close to 0. If δ is not given and *x* and *y* are very close to 0, any error produces a large relative error (in the limit, 2).

Specifying a value for δ avoids this extreme sensitivity of the RELATIVE method for small values. If you specify METHOD=RELATIVE(δ) CRITERION=γ when both *x* and *y* are much smaller than δ in absolute value, the comparison is as if you had specified METHOD=ABSOLUTE CRITERION=$\delta\gamma$. However, when either *x* or *y* is much larger than δ in absolute value, the comparison is like METHOD=RELATIVE CRITERION=γ. For moderate values of *x* and *y*, METHOD=RELATIVE(δ) CRITERION=γ is, in effect, a compromise between METHOD=ABSOLUTE CRITERION=$\delta\ \gamma$ and METHOD=RELATIVE CRITERION=γ.

For character variables, if one value has a greater length than the other, the shorter value is padded with blanks for the comparison. Nonblank character values are judged equal only if they agree at each character. If NOMISSING is in effect, blank character values compare equal to anything.

Definition of Difference and Percent Difference

In the reports of value comparisons and in the OUT= data set, PROC COMPARE displays difference and percent difference values for the numbers compared. These quantities are defined using the value from the base data set as the reference value.

For a numeric variable compared, let x be its value in the base data set and let y be its value in the comparison data set. If x and y are both nonmissing, the difference and percent difference are defined as follows:

Difference = $y - x$

Percent Difference = $(y - x)/x * 100 \text{ for } x \neq 0$

Percent Difference = missing for $x = 0$.

How PROC COMPARE Handles Variable Formats

PROC COMPARE compares unformatted values. If you have two matching variables that are formatted differently, PROC COMPARE lists the formats of the variables.

Results: COMPARE Procedure

PROC COMPARE reports the results of its comparisons in the following ways:

- the SAS log
- return codes stored in the automatic macro SYSINFO
- procedure output
- output data sets.

SAS Log

When you use the WARNING, PRINTALL, or ERROR option, PROC COMPARE writes a description of the differences to the SAS log.

Macro Return Codes (SYSINFO)

PROC COMPARE stores a return code in the automatic macro variable SYSINFO. The value of the return code provides information about the result of the comparison. By checking the value of SYSINFO after PROC COMPARE has run and before any other step begins, SAS macros can use the results of a PROC COMPARE step to determine what action to take or what parts of a SAS program to execute.

Table 9.1 on page 228 is a key for interpreting the SYSINFO return code from PROC COMPARE. For each of the conditions listed, the associated value is added to the return code if the condition is true. Thus, the SYSINFO return code is the sum of the codes listed in Table 9.1 on page 228 for the applicable conditions:

Table 9.1 Macro Return Codes

Bit	Condition	Code	Hex	Description
1	DSLABEL	1	0001X	Data set labels differ
2	DSTYPE	2	0002X	Data set types differ
3	INFORMAT	4	0004X	Variable has different informat
4	FORMAT	8	0008X	Variable has different format
5	LENGTH	16	0010X	Variable has different length

Bit	Condition	Code	Hex	Description
6	LABEL	32	0020X	Variable has different label
7	BASEOBS	64	0040X	Base data set has observation not in comparison
8	COMPOBS	128	0080X	Comparison data set has observation not in base
9	BASEBY	256	0100X	Base data set has BY group not in comparison
10	COMPBY	512	0200X	Comparison data set has BY group not in base
11	BASEVAR	1024	0400X	Base data set has variable not in comparison
12	COMPVAR	2048	0800X	Comparison data set has variable not in base
13	VALUE	4096	1000X	A value comparison was unequal
14	TYPE	8192	2000X	Conflicting variable types
15	BYVAR	16384	4000X	BY variables do not match
16	ERROR	32768	8000X	Fatal error: comparison not done

These codes are ordered and scaled to allow a simple check of the degree to which the data sets differ. For example, if you want to check that two data sets contain the same variables, observations, and values, but you do not care about differences in labels, formats, and so forth, use the following statements:

```
proc compare base=SAS-data-set
             compare=SAS-data-set;
run;

%if &sysinfo >= 64 %then
   %do;
      handle error;
   %end;
```

You can examine individual bits in the SYSINFO value by using DATA step bit-testing features to check for specific conditions. For example, to check for the presence of observations in the base data set that are not in the comparison data set, use the following statements:

```
proc compare base=SAS-data-set
             compare=SAS-data-set;
run;

%let rc=&sysinfo;
data _null_;
   if &rc='1......'b then
      put 'Observations in Base but not
           in Comparison Data Set';
run;
```

PROC COMPARE must run before you check SYSINFO and you must obtain the SYSINFO value before another SAS step starts because every SAS step resets SYSINFO.

Procedure Output

The following sections show and describe the default output of the two data sets shown in "Overview: COMPARE Procedure" on page 209. Because PROC COMPARE produces lengthy output, the output is presented in seven pieces.

Data Set Summary

This report lists the attributes of the data sets being compared. These attributes include the following:

- the data set names
- the data set types, if any
- the data set labels, if any
- the dates created and last modified
- the number of variables in each data set
- the number of observations in each data set.

Output 9.2 on page 230 shows the Data Set Summary.

Output 9.2 Partial Output

```
                          COMPARE Procedure
             Comparison of PROCLIB.ONE with PROCLIB.TWO
                           (Method=EXACT)

                          Data Set Summary

Dataset              Created          Modified  NVar    NObs  Label


PROCLIB.ONE  11SEP97:15:11:07  11SEP97:15:11:09     5       4  First Data Set
PROCLIB.TWO  11SEP97:15:11:10  11SEP97:15:11:10     6       5  Second Data Set
```

Variables Summary

This report compares the variables in the two data sets. The first part of the report lists the following:

- the number of variables the data sets have in common
- the number of variables in the base data set that are not in the comparison data set and vice versa
- the number of variables in both data sets that have different types
- the number of variables that differ on other attributes (length, label, format, or informat)
- the number of BY, ID, VAR, and WITH variables specified for the comparison.

The second part of the report lists matching variables with different attributes and shows how the attributes differ. (The COMPARE procedure omits variable labels if the line size is too small for them.)

Output 9.3 on page 231 shows the Variables Summary.

Output 9.3 Partial Output

```
                         Variables Summary

              Number of Variables in Common: 5.
              Number of Variables in PROCLIB.TWO but not in PROCLIB.ONE: 1.
              Number of Variables with Conflicting Types: 1.
              Number of Variables with Differing Attributes: 3.


              Listing of Common Variables with Conflicting Types

                     Variable  Dataset      Type  Length

                     student   PROCLIB.ONE  Num        8
                               PROCLIB.TWO  Char       8


            Listing of Common Variables with Differing Attributes

          Variable  Dataset      Type  Length  Format  Label

          year      PROCLIB.ONE  Char       8          Year of Birth
                    PROCLIB.TWO  Char       8
          state     PROCLIB.ONE  Char       8
                    PROCLIB.TWO  Char       8          Home State
          gr1       PROCLIB.ONE  Num        8  4.1
                    PROCLIB.TWO  Num        8  5.2
```

Observation Summary

This report provides information about observations in the base and comparison data sets. First of all, the report identifies the first and last observation in each data set, the first and last matching observations, and the first and last differing observations. Then, the report lists the following:

- the number of observations that the data sets have in common
- the number of observations in the base data set that are not in the comparison data set and vice versa
- the total number of observations in each data set
- the number of matching observations for which PROC COMPARE judged some variables unequal
- the number of matching observations for which PROC COMPARE judged all variables equal.

Output 9.4 on page 231 shows the Observation Summary.

Output 9.4 Partial Output

```
                        Observation Summary

                     Observation      Base  Compare

                     First Obs           1        1
                     First Unequal       1        1
                     Last  Unequal       4        4
                     Last  Match         4        4
                     Last  Obs           .        5

    Number of Observations in Common: 4.
    Number of Observations in PROCLIB.TWO but not in PROCLIB.ONE: 1.
    Total Number of Observations Read from PROCLIB.ONE: 4.
    Total Number of Observations Read from PROCLIB.TWO: 5.

    Number of Observations with Some Compared Variables Unequal: 4.
    Number of Observations with All Compared Variables Equal: 0.
```

Values Comparison Summary

This report first lists the following:

- the number of variables compared with all observations equal
- the number of variables compared with some observations unequal
- the number of variables with differences involving missing values, if any
- the total number of values judged unequal
- the maximum difference measure between unequal values for all pairs of matching variables (for differences not involving missing values).

In addition, for the variables for which some matching observations have unequal values, the report lists

- the name of the variable
- other variable attributes
- the number of times PROC COMPARE judged the variable unequal
- the maximum difference measure found between values (for differences not involving missing values)
- the number of differences caused by comparison with missing values, if any.

Output 9.5 on page 232 shows the Values Comparison Summary.

Output 9.5 Partial Output

```
                     Values Comparison Summary

    Number of Variables Compared with All Observations Equal: 1.
    Number of Variables Compared with Some Observations Unequal: 3.
    Total Number of Values which Compare Unequal: 6.
    Maximum Difference: 20.

                    Variables with Unequal Values

          Variable  Type  Len   Compare Label  Ndif   MaxDif

          state     CHAR    8   Home State        2
          gr1       NUM     8                     2    1.000
          gr2       NUM     8                     2   20.000
```

Value Comparison Results

This report consists of a table for each pair of matching variables judged unequal at one or more observations. When comparing character values, PROC COMPARE displays only the first 20 characters. When you use the TRANSPOSE option, it displays only the first 12 characters. Each table shows

- the number of the observation or, if you use the ID statement, the values of the ID variables
- the value of the variable in the base data set
- the value of the variable in the comparison data set
- the difference between these two values (numeric variables only)
- the percent difference between these two values (numeric variables only).

Output 9.6 on page 233 shows the Value Comparison Results for Variables.

Output 9.6 Partial Output

```
                   Value Comparison Results for Variables

          __________________________________________________________
                     ||  Home State
                     ||  Base Value           Compare Value
                Obs  ||  state                 state
           ________  ||  ________              ________
                     ||
                  2  ||  MD                    MA
                  4  ||  MA                    MD
          __________________________________________________________


          __________________________________________________________
                     ||       Base    Compare
                Obs  ||        gr1        gr1      Diff.     % Diff
           ________  ||  _________  _________  _________  _________
                     ||
                  1  ||       85.0      84.00    -1.0000    -1.1765
                  3  ||       78.0      79.00     1.0000     1.2821
          __________________________________________________________


          __________________________________________________________
                     ||       Base    Compare
                Obs  ||        gr2        gr2      Diff.     % Diff
           ________  ||  _________  _________  _________  _________
                     ||
                  3  ||    72.0000    73.0000     1.0000     1.3889
                  4  ||    94.0000    74.0000   -20.0000   -21.2766
          __________________________________________________________
```

You can suppress the value comparison results with the NOVALUES option. If you use both the NOVALUES and TRANSPOSE options, PROC COMPARE lists for each observation the names of the variables with values judged unequal but does not display the values and differences.

Table of Summary Statistics

If you use the STATS, ALLSTATS, or PRINTALL options, the Value Comparison Results for Variables section contains summary statistics for the numeric variables being compared. The STATS option generates these statistics for only the numeric variables whose values are judged unequal. The ALLSTATS and PRINTALL options generate these statistics for all numeric variables, even if all values are judged equal.

Note: In all cases PROC COMPARE calculates the summary statistics based on all matching observations that do not contain missing values, not just on those containing unequal values. △

Output 9.7 on page 234 shows the following summary statistics for base data set values, comparison data set values, differences, and percent differences:

N
: the number of nonmissing values

MEAN
: the mean, or average, of the values

STD
: the standard deviation

MAX
: the maximum value

MIN
: the minimum value

STDERR
: the standard error of the mean

T
: the T ratio (MEAN/STDERR)

PROB> | T |
: the probability of a greater absolute T value if the true population mean is 0.

NDIF
: the number of matching observations judged unequal, and the percent of the matching observations that were judged unequal.

DIFMEANS
: the difference between the mean of the base values and the mean of the comparison values. This line contains three numbers. The first is the mean expressed as a percentage of the base values mean. The second is the mean expressed as a percentage of the comparison values mean. The third is the difference in the two means (the comparison mean minus the base mean).

R
: the correlation of the base and comparison values for matching observations that are nonmissing in both data sets.

RSQ
: the square of the correlation of the base and comparison values for matching observations that are nonmissing in both data sets.

Output 9.7 on page 234 is from the ALLSTATS option using the two data sets shown in "Overview":

Output 9.7 Partial Output

```
                 Value Comparison Results for Variables

            __________________________________________________________
                       ||       Base    Compare
                   Obs ||        gr1        gr1      Diff.     % Diff
             ________  ||  _________  _________  _________  _________
                       ||
                     1 ||       85.0      84.00    -1.0000    -1.1765
                     3 ||       78.0      79.00     1.0000     1.2821
             ________  ||  _________  _________  _________  _________
                       ||
              N        ||          4          4          4          4
              Mean     ||    85.5000    85.5000          0     0.0264
              Std      ||     5.8023     5.4467     0.8165     1.0042
              Max      ||    92.0000    92.0000     1.0000     1.2821
              Min      ||    78.0000    79.0000    -1.0000    -1.1765
              StdErr   ||     2.9011     2.7234     0.4082     0.5021
              t        ||    29.4711    31.3951     0.0000     0.0526
              Prob>|t| ||     <.0001     <.0001     1.0000     0.9614
                       ||
              Ndif     ||          2    50.000%
              DifMeans ||     0.000%     0.000%          0
              r, rsq   ||      0.991      0.983
            __________________________________________________________


            __________________________________________________________
                       ||       Base    Compare
                   Obs ||        gr2        gr2      Diff.     % Diff
             ________  ||  _________  _________  _________  _________
                       ||
                     3 ||    72.0000    73.0000     1.0000     1.3889
                     4 ||    94.0000    74.0000   -20.0000   -21.2766
             ________  ||  _________  _________  _________  _________
                       ||
              N        ||          4          4          4          4
              Mean     ||    86.2500    81.5000    -4.7500    -4.9719
              Std      ||     9.9457     9.4692    10.1776    10.8895
              Max      ||    94.0000    92.0000     1.0000     1.3889
              Min      ||    72.0000    73.0000   -20.0000   -21.2766
              StdErr   ||     4.9728     4.7346     5.0888     5.4447
              t        ||    17.3442    17.2136    -0.9334    -0.9132
              Prob>|t| ||     0.0004     0.0004     0.4195     0.4285
                       ||
              Ndif     ||          2    50.000%
              DifMeans ||    -5.507%    -5.828%    -4.7500
              r, rsq   ||      0.451      0.204
            __________________________________________________________
```

Note: If you use a wide line size with PRINTALL, PROC COMPARE prints the value comparison result for character variables next to the result for numeric variables. In that case, PROC COMPARE calculates only NDIF for the character variables. △

Comparison Results for Observations (Using the TRANSPOSE Option)

The TRANSPOSE option prints the comparison results by observation instead of by variable. The comparison results precede the observation summary report. By default, the source of the values for each row of the table is indicated by the following label:

_OBS_1=*number-1*   _OBS_2=*number-2*

where *number-1* is the number of the observation in the base data set for which the value of the variable is shown, and *number-2* is the number of the observation in the comparison data set.

Output 9.8 on page 236 shows the differences in PROCLIB.ONE and PROCLIB.TWO by observation instead of by variable.

Output 9.8 Partial Output

```
                  Comparison Results for Observations

        _OBS_1=1 _OBS_2=1:
        Variable   Base Value       Compare         Diff.        % Diff
             gr1         85.0         84.00     -1.000000     -1.176471

        _OBS_1=2 _OBS_2=2:
        Variable   Base Value       Compare
           state           MD            MA

        _OBS_1=3 _OBS_2=3:
        Variable   Base Value       Compare         Diff.        % Diff
             gr1         78.0         79.00      1.000000      1.282051
             gr2    72.000000     73.000000      1.000000      1.388889

        _OBS_1=4 _OBS_2=4:
        Variable   Base Value       Compare         Diff.        % Diff
             gr2    94.000000     74.000000    -20.000000    -21.276596
           state           MA            MD
```

If you use an ID statement, the identifying label has the following form:

ID-1=ID-value-1 ... ID-n=ID-value-n

where *ID* is the name of an ID variable and *ID-value* is the value of the ID variable.

Note: When you use the TRANSPOSE option, PROC COMPARE prints only the first 12 characters of the value. △

Output Data Set (OUT=)

By default, the OUT= data set contains an observation for each pair of matching observations. The OUT= data set contains the following variables from the data sets you are comparing:

- all variables named in the BY statement
- all variables named in the ID statement
- all matching variables or, if you use the VAR statement, all variables listed in the VAR statement.

In addition, the data set contains two variables created by PROC COMPARE to identify the source of the values for the matching variables: _TYPE_ and _OBS_.

TYPE
: is a character variable of length 8. Its value indicates the source of the values for the matching (or VAR) variables in that observation. (For ID and BY variables, which are not compared, the values are the values from the original data sets.) _TYPE_ has the label **Type of Observation**. The four possible values of this variable are as follows:

  BASE

The values in this observation are from an observation in the base data set. PROC COMPARE writes this type of observation to the OUT= data set when you specify the OUTBASE option.

COMPARE
: The values in this observation are from an observation in the comparison data set. PROC COMPARE writes this type of observation to the OUT= data set when you specify the OUTCOMP option.

DIF
: The values in this observation are the differences between the values in the base and comparison data sets. For character variables, PROC COMPARE uses a period (.) to represent equal characters and an X to represent unequal characters. PROC COMPARE writes this type of observation to the OUT= data set by default. However, if you request any other type of observation with the OUTBASE, OUTCOMP, or OUTPERCENT option, you must specify the OUTDIF option to generate observations of this type in the OUT= data set.

PERCENT
: The values in this observation are the percent differences between the values in the base and comparison data sets. For character variables the values in observations of type PERCENT are the same as the values in observations of type DIF.

OBS
: is a numeric variable containing a number further identifying the source of the OUT= observations.

For observations with _TYPE_ equal to BASE, _OBS_ is the number of the observation in the base data set from which the values of the VAR variables were copied. Similarly, for observations with _TYPE_ equal to COMPARE, _OBS_ is the number of the observation in the comparison data set from which the values of the VAR variables were copied.

For observations with _TYPE_ equal to DIF or PERCENT, _OBS_ is a sequence number that counts the matching observations in the BY group.

OBS has the label **`Observation Number`**.

The COMPARE procedure takes variable names and attributes for the OUT= data set from the base data set except for the lengths of ID and VAR variables, for which it uses the longer length regardless of which data set that length is from. This behavior has two important repercussions:

- If you use the VAR and WITH statements, the names of the variables in the OUT= data set come from the VAR statement. Thus, observations with _TYPE_ equal to **`BASE`** contain the values of the VAR variables, while observations with _TYPE_ equal to **`COMPARE`** contain the values of the WITH variables.
- If you include a variable more than once in the VAR statement in order to compare it with more than one variable, PROC COMPARE can include only the first comparison in the OUT= data set because each variable must have a unique name. Other comparisons produce warning messages.

For an example of the OUT= option, see Example 6 on page 251.

Output Statistics Data Set (OUTSTATS=)

When you use the OUTSTATS= option, PROC COMPARE calculates the same summary statistics as the ALLSTATS option for each pair of numeric variables compared (see “Table of Summary Statistics” on page 233). The OUTSTATS= data set contains an observation for each summary statistic for each pair of variables. The data set also contains the BY variables used in the comparison and several variables created by PROC COMPARE:

VAR
: is a character variable containing the name of the variable from the base data set for which the statistic in the observation was calculated.

WITH
: is a character variable containing the name of the variable from the comparison data set for which the statistic in the observation was calculated. The _WITH_ variable is not included in the OUTSTATS= data set unless you use the WITH statement.

TYPE
: is a character variable containing the name of the statistic contained in the observation. Values of the _TYPE_ variable are **N**, **MEAN**, **STD**, **MIN**, **MAX**, **STDERR**, **T**, **PROBT**, **NDIF**, **DIFMEANS**, and **R**, **RSQ**.

BASE
: is a numeric variable containing the value of the statistic calculated from the values of the variable named by _VAR_ in the observations in the base data set with matching observations in the comparison data set.

COMP
: is a numeric variable containing the value of the statistic calculated from the values of the variable named by the _VAR_ variable (or by the _WITH_ variable if you use the WITH statement) in the observations in the comparison data set with matching observations in the base data set.

DIF
: is a numeric variable containing the value of the statistic calculated from the differences of the values of the variable named by the _VAR_ variable in the base data set and the matching variable (named by the _VAR_ or _WITH_ variable) in the comparison data set.

PCTDIF
: is a numeric variable containing the value of the statistic calculated from the percent differences of the values of the variable named by the _VAR_ variable in the base data set and the matching variable (named by the _VAR_ or _WITH_ variable) in the comparison data set.

Note: For both types of output data sets, PROC COMPARE assigns one of the following data set labels:

```
Comparison of base-SAS-data-set
with comparison-SAS-data-set

Comparison of variables in base-SAS-data-set
```

△

Labels are limited to 40 characters.

See Example 7 on page 253 for an example of an OUTSTATS= data set.

Examples: COMPARE Procedure

Example 1: Producing a Complete Report of the Differences

Procedure features:

PROC COMPARE statement options

BASE=
PRINTALL
COMPARE=

Data sets:

PROCLIB.ONE, PROCLIB.TWO on page 210

This example shows the most complete report that PROC COMPARE produces as procedure output.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Create a complete report of the differences between two data sets. BASE= and COMPARE= specify the data sets to compare. PRINTALL prints a full report of the differences.

```
proc compare base=proclib.one compare=proclib.two printall;
   title 'Comparing Two Data Sets: Full Report';
run;
```

Output

A > in the output marks information that is in the full report but not in the default report. The additional information includes a listing of variables found in one data set but not the other, a listing of observations found in one data set but not the other, a listing of variables with all equal values, and summary statistics. For an explanation of the statistics, see “Table of Summary Statistics” on page 233.

```
                         Comparing Two Data Sets: Full Report                    1

                                  COMPARE Procedure
                      Comparison of PROCLIB.ONE with PROCLIB.TWO
                                   (Method=EXACT)

                                  Data Set Summary

Dataset                Created          Modified  NVar    NObs  Label

PROCLIB.ONE  11SEP97:16:19:59  11SEP97:16:20:01     5       4  First Data Set
PROCLIB.TWO  11SEP97:16:20:01  11SEP97:16:20:01     6       5  Second Data Set


                                  Variables Summary

              Number of Variables in Common: 5.
              Number of Variables in PROCLIB.TWO but not in PROCLIB.ONE: 1.
              Number of Variables with Conflicting Types: 1.
              Number of Variables with Differing Attributes: 3.


          Listing of Variables in PROCLIB.TWO but not in PROCLIB.ONE

                                Variable  Type  Length

>                                major     Char       8


              Listing of Common Variables with Conflicting Types

                       Variable  Dataset      Type  Length

                       student   PROCLIB.ONE  Num        8
                                 PROCLIB.TWO  Char       8
```

```
                    Comparing Two Data Sets: Full Report                    2

                               COMPARE Procedure
                  Comparison of PROCLIB.ONE with PROCLIB.TWO
                                (Method=EXACT)

               Listing of Common Variables with Differing Attributes

             Variable  Dataset      Type  Length  Format  Label

             year      PROCLIB.ONE  Char       8          Year of Birth
                       PROCLIB.TWO  Char       8
             state     PROCLIB.ONE  Char       8
                       PROCLIB.TWO  Char       8          Home State
             gr1       PROCLIB.ONE  Num        8  4.1
                       PROCLIB.TWO  Num        8  5.2


                    Comparison Results for Observations

>        Observation 5 in PROCLIB.TWO not found in PROCLIB.ONE.


                            Observation Summary

                       Observation      Base  Compare

                       First Obs           1        1
                       First Unequal       1        1
                       Last  Unequal       4        4
                       Last  Match         4        4
                       Last  Obs           .        5

          Number of Observations in Common: 4.
          Number of Observations in PROCLIB.TWO but not in PROCLIB.ONE: 1.
          Total Number of Observations Read from PROCLIB.ONE: 4.
          Total Number of Observations Read from PROCLIB.TWO: 5.

          Number of Observations with Some Compared Variables Unequal: 4.
          Number of Observations with All Compared Variables Equal: 0.
```

```
                    Comparing Two Data Sets: Full Report                    3

                               COMPARE Procedure
                  Comparison of PROCLIB.ONE with PROCLIB.TWO
                                (Method=EXACT)

                          Values Comparison Summary

          Number of Variables Compared with All Observations Equal: 1.
          Number of Variables Compared with Some Observations Unequal: 3.
          Total Number of Values which Compare Unequal: 6.
          Maximum Difference: 20.


                      Variables with All Equal Values

>                   Variable  Type  Len   Label

                   year      CHAR    8   Year of Birth

                       Variables with Unequal Values

                Variable  Type  Len   Compare Label  Ndif   MaxDif

                state     CHAR    8   Home State        2
                gr1       NUM     8                     2    1.000
                gr2       NUM     8                     2   20.000
```

```
                Comparing Two Data Sets: Full Report                    4

                         COMPARE Procedure
              Comparison of PROCLIB.ONE with PROCLIB.TWO
                           (Method=EXACT)

               Value Comparison Results for Variables

           __________________________________________________________
                      ||  Year of Birth
                      ||  Base Value           Compare Value
                 Obs  ||  year                  year
            ________  ||  ________              ________
                      ||
                   1  ||  1970                  1970
                   2  ||  1971                  1971
                   3  ||  1969                  1969
                   4  ||  1970                  1970
           __________________________________________________________


           __________________________________________________________
                      ||  Home State
                      ||  Base Value           Compare Value
                 Obs  ||  state                 state
            ________  ||  ________              ________
                      ||
                   1  ||  NC                    NC
                   2  ||  MD                    MA
                   3  ||  PA                    PA
                   4  ||  MA                    MD
           __________________________________________________________
```

```
                Comparing Two Data Sets: Full Report                    5

                         COMPARE Procedure
              Comparison of PROCLIB.ONE with PROCLIB.TWO
                           (Method=EXACT)

               Value Comparison Results for Variables

           __________________________________________________________
                      ||       Base    Compare
                 Obs  ||       gr1         gr1       Diff.     % Diff
            ________  ||  _________  _________  _________  _________
                      ||
                   1  ||      85.0       84.00    -1.0000    -1.1765
                   2  ||      92.0       92.00          0          0
                   3  ||      78.0       79.00     1.0000     1.2821
                   4  ||      87.0       87.00          0          0
            ________  ||  _________  _________  _________  _________
  >                   ||
               N      ||         4           4          4          4
              Mean    ||   85.5000     85.5000          0     0.0264
              Std     ||    5.8023      5.4467     0.8165     1.0042
              Max     ||   92.0000     92.0000     1.0000     1.2821
              Min     ||   78.0000     79.0000    -1.0000    -1.1765
             StdErr   ||    2.9011      2.7234     0.4082     0.5021
               t      ||   29.4711     31.3951     0.0000     0.0526
            Prob>|t|  ||    <.0001      <.0001     1.0000     0.9614
                      ||
             Ndif     ||         2     50.000%
            DifMeans  ||    0.000%      0.000%          0
             r, rsq   ||     0.991       0.983
           __________________________________________________________
```

```
                 Comparing Two Data Sets: Full Report                  6

                           COMPARE Procedure
               Comparison of PROCLIB.ONE with PROCLIB.TWO
                             (Method=EXACT)

                  Value Comparison Results for Variables

          ___________________________________________________________
                   ||       Base    Compare
               Obs ||        gr2        gr2      Diff.     % Diff
           ________ ||  _________  _________  _________  _________
                   ||
                 1 ||    87.0000    87.0000          0          0
                 2 ||    92.0000    92.0000          0          0
                 3 ||    72.0000    73.0000     1.0000     1.3889
                 4 ||    94.0000    74.0000   -20.0000   -21.2766
           ________ ||  _________  _________  _________  _________
>                  ||
             N     ||          4          4          4          4
            Mean   ||    86.2500    81.5000    -4.7500    -4.9719
            Std    ||     9.9457     9.4692    10.1776    10.8895
            Max    ||    94.0000    92.0000     1.0000     1.3889
            Min    ||    72.0000    73.0000   -20.0000   -21.2766
           StdErr  ||     4.9728     4.7346     5.0888     5.4447
             t     ||    17.3442    17.2136    -0.9334    -0.9132
          Prob>|t| ||     0.0004     0.0004     0.4195     0.4285
                   ||
           Ndif    ||          2    50.000%
          DifMeans ||    -5.507%    -5.828%    -4.7500
           r, rsq  ||      0.451      0.204
          ___________________________________________________________
```

Example 2: Comparing Variables in Different Data Sets

Procedure features:
PROC COMPARE statement option
NOSUMMARY
VAR statement
WITH statement

Data sets:
PROCLIB.ONE, PROCLIB.TWO on page 210.

This example compares a variable from the base data set with a variable in the comparison data set. All summary reports are suppressed.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Suppress all summary reports of the differences between two data sets. BASE= specifies the base data set and COMPARE= specifies the comparison data set. NOSUMMARY suppresses all summary reports.

```
proc compare base=proclib.one compare=proclib.two nosummary;
```

Specify one variable from the base data set to compare with one variable from the comparison data set. The VAR and WITH statements specify the variables to compare. This example compares GR1 from the base data set with GR2 from the comparison data set.

```
   var gr1;
   with gr2;
   title 'Comparison of Variables in Different Data Sets';
run;
```

Output

```
                   Comparison of Variables in Different Data Sets                1

                                 COMPARE Procedure
                     Comparison of PROCLIB.ONE with PROCLIB.TWO
                                   (Method=EXACT)

NOTE: Data set PROCLIB.TWO contains 1 observations not in PROCLIB.ONE.
NOTE: Values of the following 1 variables compare unequal: gr1^=gr2


                       Value Comparison Results for Variables

             __________________________________________________________
                        ||       Base    Compare
                    Obs ||        gr1        gr2      Diff.     % Diff
              ________  ||  _________  _________  _________  _________
                        ||
                      1 ||       85.0    87.0000     2.0000     2.3529
                      3 ||       78.0    73.0000    -5.0000    -6.4103
                      4 ||       87.0    74.0000   -13.0000   -14.9425
             __________________________________________________________
```

Example 3: Comparing a Variable Multiple Times

Procedure features:

VAR statement

WITH statement

Data sets:

PROCLIB.ONE, PROCLIB.TWO on page 210.

This example compares one variable from the base data set with two variables in the comparison data set.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Suppress all summary reports of the differences between two data sets. BASE= specifies the base data set and COMPARE= specifies the comparison data set. NOSUMMARY suppresses all summary reports.

```
proc compare base=proclib.one compare=proclib.two nosummary;
```

Specify one variable from the base data set to compare with two variables from the comparison data set. The VAR and WITH statements specify the variables to compare. This example compares GR1 from the base data set with GR1 and GR2 from the comparison data set.

```
   var gr1 gr1;
   with gr1 gr2;
   title 'Comparison of One Variable with Two Variables';
run;
```

Output

The Value Comparison Results section shows the result of the comparison.

```
                  Comparison of One Variable with Two Variables                1

                                COMPARE Procedure
                    Comparison of PROCLIB.ONE with PROCLIB.TWO
                                 (Method=EXACT)

NOTE: Data set PROCLIB.TWO contains 1 observations not in PROCLIB.ONE.
NOTE: Values of the following 2 variables compare unequal: gr1^=gr1 gr1^=gr2


                      Value Comparison Results for Variables

            __________________________________________________________
                        ||       Base    Compare
                   Obs  ||        gr1        gr1      Diff.     % Diff
             ________   ||  _________  _________  _________  _________
                        ||
                     1  ||       85.0      84.00    -1.0000    -1.1765
                     3  ||       78.0      79.00     1.0000     1.2821
            __________________________________________________________


            __________________________________________________________
                        ||       Base    Compare
                   Obs  ||        gr1        gr2      Diff.     % Diff
             ________   ||  _________  _________  _________  _________
                        ||
                     1  ||       85.0    87.0000     2.0000     2.3529
                     3  ||       78.0    73.0000    -5.0000    -6.4103
                     4  ||       87.0    74.0000   -13.0000   -14.9425
            __________________________________________________________
```

Example 4: Comparing Variables That Are in the Same Data Set

Procedure features:

PROC COMPARE statement options

ALLSTATS
BRIEFSUMMARY

VAR statement

WITH statement

Data set:

PROCLIB.ONE on page 210.

This example shows that PROC COMPARE can compare two variables that are in the same data set.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Create a short summary report of the differences within one data set. ALLSTATS prints summary statistics. BRIEFSUMMARY prints only a short comparison summary.

```
proc compare base=proclib.one allstats briefsummary;
```

Specify two variables from the base data set to compare. The VAR and WITH statements specify the variables in the base data set to compare. This example compares GR1 with GR2. Because there is no comparison data set, the variables GR1 and GR2 must be in the base data set.

```
   var gr1;
   with gr2;
   title 'Comparison of Variables in the Same Data Set';
run;
```

Output

```
                Comparison of Variables in the Same Data Set                1

                             COMPARE Procedure
                  Comparisons of variables in PROCLIB.ONE
                              (Method=EXACT)

NOTE: Values of the following 1 variables compare unequal: gr1^=gr2


                  Value Comparison Results for Variables

            __________________________________________________________
                       ||       Base    Compare
                   Obs ||        gr1        gr2      Diff.     % Diff
             ________  ||  _________  _________  _________  _________
                       ||
                     1 ||       85.0    87.0000     2.0000     2.3529
                     3 ||       78.0    72.0000    -6.0000    -7.6923
                     4 ||       87.0    94.0000     7.0000     8.0460
             ________  ||  _________  _________  _________  _________
                       ||
               N       ||          4          4          4          4
              Mean     ||    85.5000    86.2500     0.7500     0.6767
               Std     ||     5.8023     9.9457     5.3774     6.5221
               Max     ||    92.0000    94.0000     7.0000     8.0460
               Min     ||    78.0000    72.0000    -6.0000    -7.6923
              StdErr   ||     2.9011     4.9728     2.6887     3.2611
               t       ||    29.4711    17.3442     0.2789     0.2075
             Prob>|t|  ||     <.0001     0.0004     0.7984     0.8489
                       ||
              Ndif     ||          3    75.000%
             DifMeans  ||     0.877%     0.870%     0.7500
              r, rsq   ||     0.898      0.807
            __________________________________________________________
```

Example 5: Comparing Observations with an ID Variable

Procedure features:
ID statement

In this example, PROC COMPARE compares only the observations that have matching values for the ID variable.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Create the PROCLIB.EMP95 and PROCLIB.EMP96 data sets. PROCLIB.EMP95 and PROCLIB.EMP96 contain employee data. IDNUM works well as an ID variable because it has unique values. A DATA step on page 1643 creates PROCLIB.EMP95. A DATA step on page 1644 creates PROCLIB.EMP96.

```
data proclib.emp95;
   input #1 idnum $4. @6 name $15.
         #2 address $42.
         #3 salary 6.;
   datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
... more data lines...
3888 Kim Siu
5662 Magnolia Blvd Southeast Cary NC 27513
77558
;

data proclib.emp96;
   input #1 idnum $4. @6 name $15.
         #2 address $42.
         #3 salary 6.;
   datalines;
2388 James Schmidt
100 Apt. C Blount St. SW Raleigh NC 27693
92100
2457 Fred Williams
99 West Lane  Garner NC 27509
33190
...more data lines...
```

```
6544 Roger Monday
3004 Crepe Myrtle Court Raleigh NC 27604
47007
;
```

Sort the data sets by the ID variable. Both data sets must be sorted by the variable that will be used as the ID variable in the PROC COMPARE step. OUT= specifies the location of the sorted data.

```
proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
   by idnum;
run;
```

Create a summary report that compares observations with matching values for the ID variable. The ID statement specifies IDNUM as the ID variable.

```
proc compare base=emp95_byidnum compare=emp96_byidnum;
   id idnum;
   title 'Comparing Observations that Have Matching IDNUMs';
run;
```

Output

PROC COMPARE identifies specific observations by the value of IDNUM. In the **Value Comparison Results for Variables** section, PROC COMPARE prints the nonmatching addresses and nonmatching salaries. For salaries, PROC COMPARE computes the numerical difference and the percent difference. Because ADDRESS is a character variable, PROC COMPARE displays only the first 20 characters. For addresses where the observation has an IDNUM of **0987**, **2776**, or **3888**, the differences occur after the 20th character and the differences do not appear in the output. The plus sign in the output indicates that the full value is not shown. To see the entire value, create an output data set. See Example 6 on page 251.

```
               Comparing Observations that Have Matching IDNUMs               1

                              COMPARE Procedure
           Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
                                (Method=EXACT)

                              Data Set Summary

     Dataset                      Created          Modified  NVar    NObs

     WORK.EMP95_BYIDNUM  13MAY98:16:03:36  13MAY98:16:03:36     4      10
     WORK.EMP96_BYIDNUM  13MAY98:16:03:36  13MAY98:16:03:36     4      12


                              Variables Summary

                    Number of Variables in Common: 4.
                    Number of ID Variables: 1.


                             Observation Summary

                   Observation      Base  Compare  ID

                   First Obs           1        1  idnum=0987
                   First Unequal       1        1  idnum=0987
                   Last  Unequal      10       12  idnum=9857
                   Last  Obs          10       12  idnum=9857

Number of Observations in Common: 10.
Number of Observations in WORK.EMP96_BYIDNUM but not in WORK.EMP95_BYIDNUM: 2.
Total Number of Observations Read from WORK.EMP95_BYIDNUM: 10.
Total Number of Observations Read from WORK.EMP96_BYIDNUM: 12.

Number of Observations with Some Compared Variables Unequal: 5.
Number of Observations with All Compared Variables Equal: 5.
               Comparing Observations that Have Matching IDNUMs               2

                              COMPARE Procedure
           Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
                                (Method=EXACT)

                          Values Comparison Summary

       Number of Variables Compared with All Observations Equal: 1.
       Number of Variables Compared with Some Observations Unequal: 2.
       Total Number of Values which Compare Unequal: 8.
       Maximum Difference: 2400.
```

```
                    Variables with Unequal Values

                 Variable  Type  Len  Ndif   MaxDif

                 address   CHAR   42     4
                 salary    NUM     8     4     2400



                Value Comparison Results for Variables

       _______________________________________________________
               ||  Base Value           Compare Value
        idnum  ||  address               address
        _____  ||  ___________________+  ___________________+
               ||
        0987   ||  2344 Persimmons Bran  2344 Persimmons Bran
        2776   ||  12988 Wellington Far  12988 Wellington Far
        3888   ||  5662 Magnolia Blvd S  5662 Magnolia Blvd S
        9857   ||  1000 Taft Ave. Morri  100 Taft Ave. Morris
       _______________________________________________________

           Comparing Observations that Have Matching IDNUMs             3

                          COMPARE Procedure
       Comparison of WORK.EMP95_BYIDNUM with WORK.EMP96_BYIDNUM
                            (Method=EXACT)

                Value Comparison Results for Variables

       _______________________________________________________
               ||       Base    Compare
        idnum  ||     salary     salary       Diff.     % Diff
        _____  ||  _________  _________   _________  _________
               ||
        0987   ||      44010      45110        1100     2.4994
        3286   ||      87734      89834        2100     2.3936
        3888   ||      77558      79958        2400     3.0945
        9857   ||      38756      40456        1700     4.3864
       _______________________________________________________
```

Example 6: Comparing Values of Observations Using an Output Data Set (OUT=)

Procedure features:

PROC COMPARE statement options:

- NOPRINT
- OUT=
- OUTBASE
- OUTBASE
- OUTCOMP
- OUTDIF
- OUTNOEQUAL

Other features: PRINT procedure

Data sets: PROCLIB.EMP95 and PROCLIB.EMP96 on page 248

This example creates and prints an output data set that shows the differences between matching observations.

In Example 5 on page 248, the output does not show the differences past the 20th character. The output data set in this example shows the full values. Further, it shows the observations that occur in only one of the data sets.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=120 pagesize=40;
```

Sort the data sets by the ID variable. Both data sets must be sorted by the variable that will be used as the ID variable in the PROC COMPARE step. OUT= specifies the location of the sorted data.

```
proc sort data=proclib.emp95 out=emp95_byidnum;

 by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
   by idnum;
run;
```

Specify the data sets to compare. BASE= and COMPARE= specify the data sets to compare.

```
proc compare base=emp95_byidnum compare=emp96_byidnum
```

Create the output data set RESULT and include all unequal observations and their differences. OUT= names and creates the output data set. NOPRINT suppresses the printing of the procedure output. OUTNOEQUAL includes only observations that are judged unequal. OUTBASE writes an observation to the output data set for each observation in the base data set. OUTCOMP writes an observation to the output data set for each observation in the comparison data set. OUTDIF writes an observation to the output data set that contains the differences between the two observations.

```
             out=result outnoequal outbase outcomp outdif
     noprint;
```

Specify the ID variable. The ID statement specifies IDNUM as the ID variable.

```
   id idnum;
run;
```

Print the output data set RESULT and use the BY and ID statements with the ID variable. PROC PRINT prints the output data set. Using the BY and ID statements with the same variable makes the output easy to read. See Chapter 32, “The PRINT Procedure,” on page 817 for more information on this technique.

```
proc print data=result noobs;
   by idnum;
   id idnum;
   title 'The Output Data Set RESULT';
run;
```

Output

The differences for character variables are noted with an X or a period (.). An X shows that the characters do not match. A period shows that the characters do match. For numeric variables, an E means that there is no difference. Otherwise, the numeric difference is shown. By default, the output data set shows that two observations in the comparison data set have no matching observation in the base data set. You do not have to use an option to make those observations appear in the output data set.

```
                                The Output Data Set RESULT                                    1

idnum    _TYPE_     _OBS_    name              address                                      salary

0987     BASE          1     Dolly Lunford     2344 Persimmons Branch  Apex NC 27505         44010
         COMPARE       1     Dolly Lunford     2344 Persimmons Branch Trail Apex NC 27505    45110
         DIF           1     ...............   .......................XXXXX.XXXXXXXXXXXXX     1100

2776     BASE          5     Robert Jones      12988 Wellington Farms Ave. Cary NC 27512     29025
         COMPARE       5     Robert Jones      12988 Wellington Farms Ave. Cary NC 27511     29025
         DIF           5     ...............   ....................................X.            E

3278     COMPARE       6     Mary Cravens      211 N. Cypress St. Cary NC 27512              35362

3286     BASE          6     Hoa Nguyen        2818 Long St. Cary NC 27513                   87734
         COMPARE       7     Hoa Nguyen        2818 Long St. Cary NC 27513                   89834
         DIF           6     ...............   ..........................................     2100

3888     BASE          7     Kim Siu           5662 Magnolia Blvd Southeast Cary NC 27513    77558
         COMPARE       8     Kim Siu           5662 Magnolia Blvd Southwest Cary NC 27513    79958
         DIF           7     ...............   ........................XX................     2400

6544     COMPARE       9     Roger Monday      3004 Crepe Myrtle Court Raleigh NC 27604      47007

9857     BASE         10     Kathy Krupski     1000 Taft Ave. Morrisville NC 27508           38756
         COMPARE      12     Kathy Krupski     100 Taft Ave. Morrisville NC 27508            40456
         DIF          10     ...............   ...XXXXXXXXXXXXXX.XXXXX.XXXXXXXXXXX.......     1700
```

Example 7: Creating an Output Data Set of Statistics (OUTSTATS=)

Procedure features:

PROC COMPARE statement options:

NOPRINT

OUTSTATS=

Data sets: PROCLIB.EMP95, PROCLIB.EMP96 on page 248

This example creates an output data set that contains summary statistics for the numeric variables that are compared.

Program

```
libname proclib 'SAS-data-library';
```

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Sort the data sets by the ID variable. Both data sets must be sorted by the variable that will be used as the ID variable in the PROC COMPARE step. OUT= specifies the location of the sorted data.

```
proc sort data=proclib.emp95 out=emp95_byidnum;
   by idnum;
run;

proc sort data=proclib.emp96 out=emp96_byidnum;
   by idnum;
run;
```

Create the output data set of statistics and compare observations that have matching values for the ID variable. BASE= and COMPARE= specify the data sets to compare. OUTSTATS= creates the output data set DIFFSTAT. NOPRINT suppresses the procedure output. The ID statement specifies IDNUM as the ID variable. PROC COMPARE uses the values of IDNUM to match observations.

```
proc compare base=emp95_byidnum compare=emp96_byidnum
             outstats=diffstat noprint;
   id idnum;
run;
```

Print the output data set DIFFSTAT. PROC PRINT prints the output data set DIFFSTAT.

```
proc print data=diffstat noobs;
   title 'The DIFFSTAT Data Set';
run;
```

Output

The variables are described in “Output Statistics Data Set (OUTSTATS=)” on page 237.

```
                          The DIFFSTAT Data Set                              1

     _VAR_      _TYPE_         _BASE_       _COMP_       _DIF_     _PCTDIF_

     salary     N               10.00        10.00       10.00      10.0000
     salary     MEAN         52359.00     53089.00      730.00       1.2374
     salary     STD          24143.84     24631.01      996.72       1.6826
     salary     MAX          92100.00     92100.00     2400.00       4.3864
     salary     MIN          29025.00     29025.00        0.00       0.0000
     salary     STDERR        7634.95      7789.01      315.19       0.5321
     salary     T                6.86         6.82        2.32       2.3255
     salary     PROBT            0.00         0.00        0.05       0.0451
     salary     NDIF             4.00        40.00         .           .
     salary     DIFMEANS         1.39         1.38      730.00         .
     salary     R,RSQ            1.00         1.00         .           .
```

CHAPTER

10

The CONTENTS Procedure

Overview: CONTENTS Procedure

The CONTENTS procedure shows the contents of a SAS data set and prints the directory of the SAS data library.

Generally, the CONTENTS procedure functions the same as the CONTENTS statement in the DATASETS procedure. The differences between the CONTENTS procedure and the CONTENTS statement in PROC DATASETS are as follows:

- The default for *libref* in the DATA= option in PROC CONTENTS is either WORK or USER. For the CONTENTS statement, the default is the libref of the procedure input library.
- PROC CONTENTS can read sequential files. The CONTENTS statement cannot.

Syntax: PROC CONTENTS

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can use the ATTRIB, FORMAT, and LABEL statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

Reminder: You can use data set options with the DATA= and OUT= options. See "Data Set Options" on page 17 for a list.

Reminder: Complete documentation for the CONTENTS statement and the CONTENTS procedure is in "CONTENTS Statement" on page 344.

PROC CONTENTS <*option(s)*>;

To do this	Use this option
Print centiles information for indexed variables	CENTILES
Specify the input data set	DATA=
Include information in the output about the number of observations, number of variables, and data set labels	DETAILS | NODETAILS
Print a list of the SAS files in the SAS data library	DIRECTORY
Print the length of a variable's informat or format	FMTLEN
Restrict processing to one or more types of SAS file	MEMTYPE=
Suppress the printing of individual files	NODS
Suppress the printing of the output	NOPRINT
Specify the output data set	OUT=
Specify an output data set that contains information about constraints	OUT2=
Print abbreviated output	SHORT
Print a list of the variables by their logical position in the data set	VARNUM

CHAPTER

11

The COPY Procedure

Overview: COPY Procedure

The COPY procedure copies one or more SAS files from a SAS data library.

Generally, the COPY procedure functions the same as the COPY statement in the DATASETS procedure. The two differences are as follows:

- The IN= argument is required with PROC COPY. In the COPY statement, IN= is optional. If IN= is omitted, the default value is the libref of the procedure input library.
- PROC DATASETS cannot work with libraries that allow only sequential data access.

Syntax: PROC COPY

Reminder: See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

Reminder: Complete documentation for the COPY statement and the COPY procedure is in "COPY Statement" on page 347.

Restriction: PROC COPY ignors explicit concatenations with catalogs. Use PROC CATALOG COPY to copy concatenated catalogs.

PROC COPY OUT=*libref-1* IN=*libref-2*
<CLONE | NOCLONE>
<CONSTRAINT=YES | NO>
<DATECOPY>
<INDEX=YES | NO>
<MEMTYPE=(*mtype(s)*)>
<MOVE <ALTER=*alter-password*>>;
EXCLUDE *SAS-file(s)* </ MEMTYPE=*mtype*>;
SELECT *SAS-file(s)* </ <MEMTYPE=*mtype*>

<ALTER=*alter-password*>>;

Concepts: COPY Procedure

Transporting SAS Data Sets between Hosts

The COPY procedure, along with the XPORT engine and the XML engine, can create and read transport files that can be moved from one host to another. PROC COPY can create transport files only with SAS data sets, not with catalogs or other types of SAS files.

Transporting is a three-step process:

1. Use PROC COPY to copy one or more SAS data sets to a file that is created with either the transport (XPORT) engine or the XML engine. This file is referred to as a *transport file* and is always a sequential file.
2. After the file is created, you can move it to another operating environment via communications software, such as FTP, or tape. If you use communications software, be sure to move the file in binary format to avoid any type of conversion. If you are moving the file to a mainframe, the file must have certain attributes. Consult the SAS documentation for your operating environment and the SAS Technical Support Web page for more information.
3. After you have successfully moved the file to the receiving host, use PROC COPY to copy the data sets from the transport file to a SAS data library.

For an example, see Example 1 on page 260.

For details on transporting files, see *Moving and Accessing SAS Files across Operating Environments*.

The CPORT and CIMPORT procedures also provide a way to transport SAS files. For information, see Chapter 8, "The CIMPORT Procedure," on page 199 and Chapter 13, "The CPORT Procedure," on page 307.

Example: COPY Procedure

Example 1: Copying SAS Data Sets between Hosts

Features:

PROC COPY statement options:

IN=
MEMTYPE=
OUT=

Other features: XPORT engine

This example illustrates how to create a transport file on a host and read it on another host.

In order for this example to work correctly, the transport file must have certain characteristics, as described in the SAS documentation for your operating environment.

In addition, the transport file must be moved to the receiving operating system in binary format.

Program

Assign library references. Assign a libref, such as SOURCE, to the SAS data library that contains the SAS data set that you want to transport. Also, assign a libref to the transport file and use the XPORT keyword to specify the XPORT engine.

```
libname source 'SAS-data-library-on-sending-host';
libname xptout xport 'filename-on-sending-host';
```

Copy the SAS data sets to the transport file. Use PROC COPY to copy the SAS data sets from the IN= library to the transport file. MEMTYPE=DATA specifies that only SAS data sets are copied. SELECT selects the data sets that you want to copy.

```
proc copy in=source out=xptout memtype=data;
   select bonus budget salary;
run;
```

SAS Log

SAS Log on Sending Host

```
1    libname source 'SAS-data-library-on-sending-host ';
NOTE: Libref SOURCE was successfully assigned as follows:
      Engine:        V9
      Physical Name: SAS-data-library-on-sending-host
2    libname xptout xport 'filename-on-sending-host';
NOTE: Libref XPTOUT was successfully assigned as follows:
      Engine:        XPORT
      Physical Name: filename-on-sending-host
3    proc copy in=source out=xptout memtype=data;
4    select bonus budget salary;
5    run;

NOTE: Copying SOURCE.BONUS to XPTOUT.BONUS (memtype=DATA).
NOTE: The data set XPTOUT.BONUS has 1 observations and 3 variables.
NOTE: Copying SOURCE.BUDGET to XPTOUT.BUDGET (memtype=DATA).
NOTE: The data set XPTOUT.BUDGET has 1 observations and 3 variables.
NOTE: Copying SOURCE.SALARY to XPTOUT.SALARY (memtype=DATA).
NOTE: The data set XPTOUT.SALARY has 1 observations and 3 variables.
```

Enable the procedure to read data from the transport file. The XPORT engine in the LIBNAME statement enables the procedure to read the data from the transport file.

```
libname insource xport 'filename-on-receiving-host';
```

Copy the SAS data sets to the receiving host. After you copy the files (for example, by using FTP in binary mode to the Windows NT host), use PROC COPY to copy the SAS data sets to the WORK data library on the receiving host.

```
proc copy in=insource out=work;
run;
```

SAS Log on Receiving Host

```
1     libname insource xport 'filename-on-receiving-host';
NOTE: Libref INSOURCE was successfully assigned as follows:
       Engine:      XPORT
       Physical Name: filename-on-receiving-host
2     proc copy in=insource out=work;
3     run;
NOTE: Input library INSOURCE is sequential.
NOTE: Copying INSOURCE.BUDGET to WORK.BUDGET (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set WORK.BUDGET has 1 observations and 3 variables.
NOTE: Copying INSOURCE.BONUS to WORK.BONUS (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set WORK.BONUS has 1 observations and 3 variables.
NOTE: Copying INSOURCE.SALARY to WORK.SALARY (memtype=DATA).
NOTE: BUFSIZE is not cloned when copying across different engines.
      System Option for BUFSIZE was used.
NOTE: The data set WORK.SALARY has 1 observations and 3 variables.
```

CHAPTER 12

The CORR Procedure

Overview: CORR Procedure

The CORR procedure is a statistical procedure for numeric random variables that computes Pearson correlation coefficients, three nonparametric measures of association, and the probabilities associated with these statistics. The correlation statistics include

- Pearson product-moment and weighted product-moment correlation
- Spearman rank-order correlation

- □ Kendall's tau-b
- □ Hoeffding's measure of dependence, D
- □ Pearson, Spearman, and Kendall partial correlation.

PROC CORR also computes Cronbach's coefficient alpha for estimating reliability.

The default correlation analysis includes descriptive statistics, Pearson correlation statistics, and probabilities for each analysis variable. You can save the correlation statistics in a SAS data set for use with other statistical and reporting procedures.

Output 12.1 on page 264 is the simplest form of PROC CORR output. Pearson correlation statistics are computed for all numeric variables from a study investigating the effect of exercise on physical fitness. The statements that produce the output follow:

```
options pagesize=60;
proc corr data=fitness;
run;
```

Output 12.1 Simple Correlation Analysis for a Fitness Study Using PROC CORR

```
                              The SAS System                              1

                            The CORR Procedure

            4  Variables:    Age      Weight   Runtime  Oxygen


                            Simple Statistics

Variable          N         Mean      Std Dev          Sum      Minimum      Maximum

Age              30     47.56667      5.26330         1427     38.00000     57.00000
Weight           30     77.70500      8.34152         2331     59.08000     91.63000
Runtime          29     10.61448      1.41655    307.82000      8.17000     14.03000
Oxygen           29     47.06445      5.32129         1365     37.38800     60.05500


                    Pearson Correlation Coefficients
                       Prob > |r| under H0: Rho=0
                         Number of Observations

                       Age         Weight        Runtime         Oxygen

       Age          1.00000       -0.21777        0.19528       -0.32899
                                    0.2477         0.3100         0.0814
                         30             30             29             29

       Weight      -0.21777        1.00000        0.15155       -0.19900
                     0.2477                        0.4326         0.3007
                         30             30             29             29

       Runtime      0.19528        0.15155        1.00000       -0.78346
                     0.3100         0.4326                        <.0001
                         29             29             29             28

       Oxygen      -0.32899       -0.19900       -0.78346        1.00000
                     0.0814         0.3007         <.0001
                         29             29             28             29
```

Output 12.2 on page 265 and Output 12.3 on page 266 illustrate the use of PROC CORR to calculate partial correlation statistics for the fitness study and to store the results in an output data set. The statements that produce the analysis also

- □ suppress the descriptive statistics

- □ select and label analysis variables
- □ exclude all observations with missing values
- □ calculate the partial covariance matrix
- □ calculate three types of partial correlation coefficients
- □ generate an output data set that contains Pearson correlation statistics and print the output data set.

For an explanation of the program that produces the following output, see Example 4 on page 302.

Output 12.2 Customized Correlation Analysis with Partial Covariances and Correlation Statistics

```
                Partial Correlations for a Fitness and Exercise Study                           1

                                 The CORR Procedure

                   1 Partial Variables:    Age
                   3         Variables:    Weight   Oxygen   Runtime


                          Partial Covariance Matrix, DF = 26

                                           Weight            Oxygen           Runtime

Weight       Wt in kg              72.43742055      -12.75113194        2.06766763
Oxygen       O2 use               -12.75113194       27.01654904       -5.59370556
Runtime      1.5 mi in minutes      2.06766763       -5.59370556        1.94512451


                    Pearson Partial Correlation Coefficients, N = 28
                           Prob > |r| under H0: Partial Rho=0

                                      Weight        Oxygen       Runtime

            Weight                   1.00000      -0.28824       0.17419
            Wt in kg                                0.1448        0.3849

            Oxygen                  -0.28824       1.00000      -0.77163
            O2 use                    0.1448                     <.0001

            Runtime                  0.17419      -0.77163       1.00000
            1.5 mi in minutes         0.3849        <.0001


                   Spearman Partial Correlation Coefficients, N = 28
                           Prob > |r| under H0: Partial Rho=0

                                      Weight        Oxygen       Runtime

            Weight                   1.00000      -0.16407       0.08708
            Wt in kg                                0.4135        0.6658

            Oxygen                  -0.16407       1.00000      -0.67112
            O2 use                    0.4135                      0.0001

            Runtime                  0.08708      -0.67112       1.00000
            1.5 mi in minutes         0.6658        0.0001


            Kendall Partial Tau b Correlation Coefficients, N = 28

                                      Weight        Oxygen       Runtime

            Weight                   1.00000      -0.09021       0.02854
            Wt in kg

            Oxygen                  -0.09021       1.00000      -0.52158
            O2 use

            Runtime                  0.02854      -0.52158       1.00000
            1.5 mi in minutes
```

Output 12.3 Output Data Set with Pearson Partial Correlation Statistics

```
          Pearson Correlation Statistics Using the PARTIAL Statement         2
                        Output Data Set from PROC CORR

            _TYPE_    _NAME_       Weight      Oxygen     Runtime

             COV      Weight      72.4374    -12.7511      2.0677
             COV      Oxygen     -12.7511     27.0165     -5.5937
             COV      Runtime      2.0677     -5.5937      1.9451
             MEAN                  0.0000      0.0000      0.0000
             STD                   8.5110      5.1977      1.3947
             N                    28.0000     28.0000     28.0000
             CORR     Weight       1.0000     -0.2882      0.1742
             CORR     Oxygen      -0.2882      1.0000     -0.7716
             CORR     Runtime      0.1742     -0.7716      1.0000
```

Syntax: CORR Procedure

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

PROC CORR <*option(s)*>;
  BY <DESCENDING> *variable-1*<...<DESCENDING> *variable-n*>
    <NOTSORTED>;
  FREQ *frequency-variable*;
  PARTIAL *variable(s)*;
  VAR *variable(s)*;
  WEIGHT *weight-variable*;
  WITH *variable(s)*;

To do this	Use this statement
Produce separate correlation analyses for each BY group	BY
Identify a variable whose values represent the frequency of each observation	FREQ
Identify controlling variables to compute Pearson, Spearman, or Kendall partial correlation coefficients	PARTIAL
Identify variables to correlate and their order in the correlation matrix	VAR

To do this	Use this statement
Identify a variable whose values weight each observation to compute Pearson weight product-moment correlation	WEIGHT
Compute correlations for specific combinations of variables	WITH

PROC CORR Statement

PROC CORR <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Create output data sets	
Specify an output data set to contain Hoeffding's D statistics	OUTH=
Specify an output data set to contain Kendall correlations	OUTK=
Specify an output data set to contain Pearson correlations	OUTP=
Specify an output data set to contain Spearman correlations	OUTS=
Control statistical analysis	
Exclude observations with nonpositive weight values from the analysis	EXCLNPWGT
Request Hoeffding's measure of dependence, D	HOEFFDING
Request Kendall's tau-b	KENDALL
Request Pearson product-moment correlation	PEARSON
Request Spearman rank-order correlation	SPEARMAN
Control Pearson correlation statistics	
Compute Cronbach's coefficient alpha	ALPHA
Compute covariances	COV
Compute corrected sums of squares and crossproducts	CSSCP
Exclude missing values	NOMISS
Specify singularity criterion	SINGULAR=
Compute sums of squares and crossproducts	SSCP
Specify the divisor for variance calculations	VARDEF=
Control printed output	
Specify the number and order of correlation coefficients	BEST=
Suppress Pearson correlations	NOCORR
Suppress all printed output	NOPRINT

To do this	Use this option
Suppress significance probabilities	NOPROB
Suppress descriptive statistics	NOSIMPLE
Change the order of correlation coefficients	RANK

Options

ALPHA

calculates and prints Cronbach's coefficient alpha. PROC CORR computes separate coefficients using raw and standardized values (scaling the variables to a unit variance of 1). For each VAR statement variable, PROC CORR computes the correlation between the variable and the total of the remaining variables. It also computes Cronbach's coefficient alpha using only the remaining variables.

Main discussion: "Cronbach's Coefficient Alpha" on page 284

Restriction: If you use a WITH statement, ALPHA is invalid.

Interaction: ALPHA invokes PEARSON.

Interaction: If you specify OUTP=, the output data set also contains six observations with Cronbach's coefficient alpha.

Interaction: When you use the PARTIAL statement, PROC CORR calculates Cronbach's coefficient alpha for partialled variables.

See also: OUTP= option

Featured in: Example 3 on page 299

BEST=*n*

prints *n* correlation coefficients for each variable. Correlations are ordered from highest to lowest in absolute value. Otherwise, PROC CORR prints correlations in a rectangular table using the variable names as row and column labels.

Interaction: When you specify HOEFFDING, PROC CORR prints the D statistics in order from highest to lowest.

Range: 1 to the maximum number of variables

COV

calculates and prints covariances.

Interaction: COV invokes PEARSON.

Interaction: If you specify OUTP=, the output data set contains the covariance matrix and the _TYPE_ variable value is COV.

Interaction: When you use the PARTIAL statement, PROC CORR computes a partial covariance matrix.

See also: OUTP= option

Featured in: Example 2 on page 295 and Example 4 on page 302

CSSCP

prints the corrected sums of squares and crossproducts.

Interaction: CSSCP invokes PEARSON.

Interaction: If you specify OUTP=, the output data set contains a CSSCP matrix and the _TYPE_ variable value is CSSCP. If you use a PARTIAL statement, the output data set contains a partial CSSCP matrix.

Interaction: When you use a PARTIAL statement, PROC CORR prints both an unpartial and a partial CSSCP matrix.

See also: OUTP= option

DATA=*SAS-data-set*
specifies the input SAS data set.

Main discussion: "Input Data Sets" on page 19

EXCLNPWGT
excludes observations with nonpositive weight values (zero or negative) from the analysis. By default, PROC CORR treats observations with negative weights like those with zero weights and counts them in the total number of observations.

Requirement: You must use a WEIGHT statement.

See also: "WEIGHT Statement" on page 275

HOEFFDING
calculates and prints Hoeffding's D statistics. This D statistic is 30 times larger than the usual definition and scales the range between -0.5 and 1 so that only large positive values indicate dependence.

Main discussion: "Hoeffding's Measure of Dependence, D" on page 281

Restriction: When you use a WEIGHT or PARTIAL statement, HOEFFDING is invalid.

Featured in: Example 1 on page 291

KENDALL
calculates and prints Kendall tau-b coefficients based on the number of concordant and discordant pairs of observations. Kendall's tau-b ranges from -1 to 1.

Main discussion: "Kendall's tau-b" on page 280

Restriction: When you use a WEIGHT statement, KENDALL is invalid.

Interactions: When you use a PARTIAL statement, probability values for Kendall's partial tau-b are not available.

Featured in: Example 4 on page 302

NOCORR
suppresses calculating and printing of Pearson correlations.

Interaction: If you specify OUTP=, the data set type remains CORR. To change the data set type to COV, CSSCP, or SSCP, use the TYPE= data set option.

See also: "Output Data Sets" on page 290

Featured in: Example 3 on page 299

NOMISS
excludes observations with missing values from the analysis. Otherwise, PROC CORR computes correlation statistics using all the nonmissing pairs of variables.

Main discussion: "Missing Values" on page 287

Tip: Using NOMISS is computationally more efficient.

Featured in: Example 3 on page 299

NOPRINT
suppresses all printed output.

Tip: Use NOPRINT when you want to create an output data set only.

NOPROB
suppresses printing the probabilities associated with each correlation coefficient.

NOSIMPLE

suppresses printing simple descriptive statistics for each variable. However, if you request an output data set, the output data set still contains simple descriptive statistics for the variables.

Featured in: Example 2 on page 295

OUTH=*output-data-set*
creates an output data set containing Hoeffding's D statistics. The contents of the output data set are similar to the OUTP= data set.

Main discussion: "Output Data Sets" on page 290

Interaction: OUTH= invokes HOEFFDING.

OUTK=*output-data-set*
creates an output data set containing Kendall correlation statistics. The contents of the output data set are similar to the OUTP= data set.

Main discussion: "Output Data Sets" on page 290

Interaction: OUTK= option invokes KENDALL.

OUTP=*output-data-set*
creates an output data set containing Pearson correlation statistics. This data set also includes means, standard deviations, and the number of observations. The value of the _TYPE_ variable is CORR.

Main discussion: "Output Data Sets" on page 290

Interaction: OUTP= invokes PEARSON.

Interaction: If you specify ALPHA, the output data set also contains six observations with Cronbach's coefficient alpha.

Featured in: Example 4 on page 302

OUTS=*SAS-data-set*
creates an output data set containing Spearman correlation statistics. The contents of the output data set are similar to the OUTP= data set.

Main discussion: "Output Data Sets" on page 290

Interaction: OUTS= invokes SPEARMAN.

PEARSON
calculates and prints Pearson product-moment correlations when you use the HOEFFDING, KENDALL, or SPEARMAN option. If you omit the correlation type, PROC CORR automatically produces Pearson correlations. The correlations range from -1 to 1.

Main discussion: "Pearson Product-Moment Correlation" on page 279

Featured in: Example 1 on page 291

RANK
prints the correlation coefficients for each variable. Correlations are ordered from highest to lowest in absolute value. Otherwise, PROC CORR prints correlations in a rectangular table using the variable names as row and column labels.

Interaction: If you use HOEFFDING, PROC CORR prints the D statistics in order from highest to lowest.

SINGULAR=*p*
specifies the criterion for determining the singularity of a variable when you use a PARTIAL statement. A variable is considered singular if its corresponding diagonal element after Cholesky decomposition has a value less than p times the original unpartialed corrected sum of squares of that variable.

Main discussion: "Partial Correlation" on page 282

Default: 1E-8

Range: between 0 and 1

SPEARMAN

calculates and prints Spearman correlation coefficients based on the ranks of the variables. The correlations range from -1 to 1.

Main discussion: "Spearman Rank-Order Correlation" on page 280

Restriction: When you specify a WEIGHT statement, SPEARMAN is invalid.

Featured in: Example 1 on page 291

SSCP

prints the sums of squares and crossproducts.

Interaction: SSCP invokes PEARSON.

Interaction: When you specify OUTP=, the output data set contains a SSCP matrix and the _TYPE_ variable value is SSCP. If you use a PARTIAL statement, the output data set does not contain an SSCP matrix.

Interaction: When you use a PARTIAL statement, PROC CORR prints the unpartial SSCP matrix.

Featured in: Example 2 on page 295

VARDEF=*divisor*

specifies the divisor to use in the calculation of variances, standard deviations, and covariances.

Table 12.1 on page 272 shows the possible values for *divisor* and associated divisors where k is the number of PARTIAL statement variables.

Table 12.1 Possible Values for VARDEF=

Value	Divisor	Formula
DF	degrees of freedom	n - k - 1
N	number of observations	n
WDF	sum of weights minus one	$(\Sigma_i w_i)$ - k - 1
WEIGHT|WGT	sum of weights	$\Sigma_i w_i$

The procedure computes the variance as $CSS/divisor$, where CSS is the corrected sums of squares and equals $\sum (x_i - \overline{x})^2$. When you weight the analysis variables, CSS equals $\sum w_i (x_i - \overline{x}_w)^2$, where $\overline{x}_w$ is the weighted mean.

Default: DF

Tip: When you use the WEIGHT statement and VARDEF=DF, the variance is an estimate of σ^2, where the variance of the ith observation is $var(x_i) = \sigma^2/w_i$ and w_i is the weight for the ith observation. This yields an estimate of the variance of an observation with unit weight.

Tip: When you use the WEIGHT statement and VARDEF=WGT, the computed variance is asymptotically (for large n) an estimate of $\sigma^2/\overline{w}$, where $\overline{w}$ is the average weight. This yields an asymptotic estimate of the variance of an observation with average weight.

Main discussion: Weighted statistics "Weighted Statistics Example" on page 60.

BY Statement

Calculates separate correlation statistics for each BY group.

Main discussion: “BY” on page 54

BY <DESCENDING> *variable-1* <…<DESCENDING> *variable-n*><NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are grouped in another way, such as chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

FREQ Statement

Treats observations as if they appear multiple times in the input data set.

Tip: The effects of the FREQ and WEIGHT statements are similar except when calculating degrees of freedom.

See also: For an example that uses the FREQ statement, see “FREQ” on page 56

FREQ *variable*;

Required Arguments

variable
specifies a numeric variable whose value represents the frequency of the observation. If you use the FREQ statement, the procedure assumes that each observation represents *n* observations, where *n* is the value of *variable*. If *n* is not an integer, SAS truncates it. If *n* is less than 1 or is missing, the procedure does not use that observation to calculate statistics.

The sum of the frequency variable represents the total number of observations.

PARTIAL Statement

Computes Pearson partial correlation, Spearman partial rank-order correlation, or Kendall's partial tau-b.

Restriction: Not valid with the HOEFFDING option.

Interaction: Invokes the NOMISS option to exclude all observations with missing values.

Main discussion: "Partial Correlation" on page 282

Featured in: Example 4 on page 302

PARTIAL *variable(s)*;

Required Arguments

variable(s)
identifies one or more variables to use in the calculation of partial correlation statistics.

Using PROC CORR Statement Options with the PARTIAL Statement

- If you use the PEARSON option, PROC CORR also prints the partial variance and standard deviation for each VAR or WITH statement variable.
- If you use the KENDALL option, PROC CORR cannot compute probability values for Kendall's partial tau-b.

VAR Statement

Specifies the variables to use to calculate correlation statistics.

Default: If you omit this statement, PROC CORR computes correlations for all numeric variables not listed in the other statements.

Featured in: Example 1 on page 291 and Example 2 on page 295

VAR *variable(s)*;

Required Arguments

variable(s)
identifies one or more variables to use in the calculation of correlation coefficients.

WEIGHT Statement

Specifies weights for the analysis variables in the calculation of Pearson weighted product-moment correlation.

Restriction: Not valid with the HOEFFDING, KENDALL, or SPEARMAN option.

See also: For information about calculating weighted correlations, see "Pearson Product-Moment Correlation" on page 279.

WEIGHT *variable*;

Required Arguments

variable
specifies a numeric variable to use to compute weighted product-moment correlation coefficients. The variable does not have to be an integer. If the value of the weight variable is

Weight value…	PROC CORR…
0	counts the observation in the total number of observations
less than 0	converts the value to zero and counts the observation in the total number of observations
missing	excludes the observation

To exclude observations that contain negative and zero weights from the analysis, use EXCLNPWGT. Note that most SAS/STAT procedures, such as PROC GLM, exclude negative and zero weights by default.

Tip: When you use the WEIGHT statement, consider which value of the VARDEF= option is appropriate. See the discussion of the VARDEF= option on page 272 for more information.

Note: Prior to Version 8 of SAS, the procedure did not exclude the observations with missing weights from the count of observations. △

WITH Statement

Determines the variables to use in conjunction with the VAR statement variables to calculate limited combinations of correlation coefficients.

Restriction: Not valid with the ALPHA option.

Featured in: Example 2 on page 295

WITH *variable(s)*;

Required Argument

variable(s)

lists one or more variables to obtain correlations for specific combinations of variables. The WITH statement variables appear down the side of the correlation matrix and the VAR statement variables appear across the top of the correlation matrix. PROC CORR computes the following correlations for the VAR statement variables A and B and the WITH statement variables X, Y, and Z:

X and A	X and B
Y and A	Y and B
Z and A	Z and B

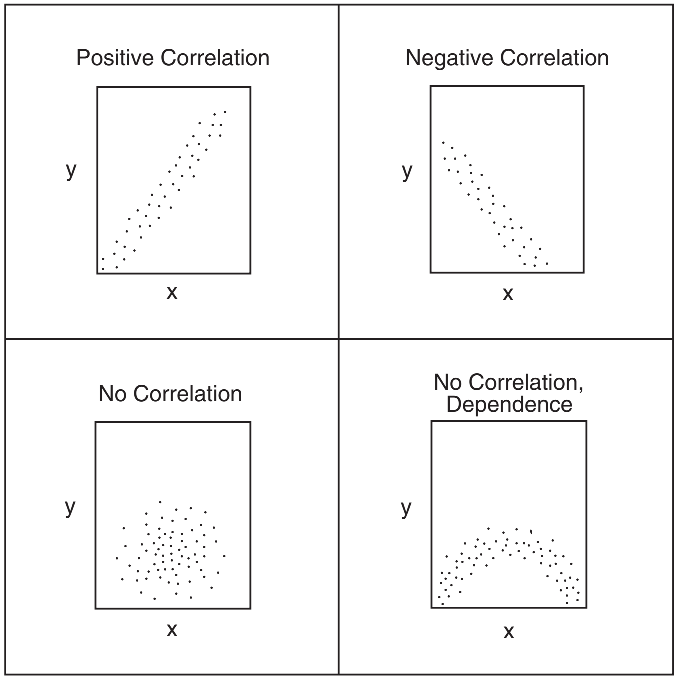
Concepts: CORR Procedure

Interpreting Correlation Coefficients

Correlation coefficients contain information on both the strength and direction of a linear relationship between two numeric random variables. If one variable x is an exact linear function of another variable y, a positive relationship exists when the correlation is 1 and an inverse relationship exists when the correlation is -1. If there is no linear predictability between the two variables, the correlation is 0. If the variables are normal and correlation is 0, the two variables are independent. However, correlation does not imply causality because, in some cases, an underlying causal relationship may exist.

The scatterplots in Figure 12.1 on page 277 depict the relationship between two numeric random variables.

Figure 12.1 Examining Correlations Using Scatterplots



When the relationship between two variables is nonlinear or when outliers are present, the correlation coefficient incorrectly estimates the strength of the relationship. Plotting the data before computing a correlation coefficient enables you to verify the linear relationship and to identify the potential outliers.

Determining Computer Resources

The only factor limiting the number of variables that you can analyze is the amount of available memory. The computer resources that PROC CORR requires depend on which statements and options you specify. To determine the computer resources that you need, use

N	number of observations in the data set.
C	number of correlation types (1 to 4).
V	number of VAR statement variables.
W	number of WITH statement variables.
P	number of PARTIAL statement variables.

so that

T=	V+W+P	
K=	V*W	when W>0
	V*(V+1)/2	when W=0
L=	K	when P=0
	T*(T+1)/2	when P>0

For small N and large K, the CPU time varies as K for all types of correlations. For large N, the CPU time depends on the type of correlation. To calculate CPU time use

K*N	with PEARSON (default)
T*N*log N	with SPEARMAN
K*N*log N	with HOEFFDING or KENDALL

You can reduce CPU time by specifying NOMISS. Without NOMISS, processing is much faster when most observations do not contain missing values.

The options and statements you use in the procedure require different amounts of storage to process the data. For Pearson correlations, the amount of temporary storage in bytes (M) is

40T+16L	with NOMISS and NOSIMPLE
40T+16L+56T	with NOMISS
40T+16L+56K	with NOSIMPLE
40T+16L+56K+56T	with no options

Using a PARTIAL statement increases the amount of temporary storage by 12T bytes. Using the ALPHA option increases the amount of temporary storage by 32V+16 bytes.

The following example uses a PARTIAL statement, which invokes NOMISS.

```
proc corr;
   var x1 x2;
   with y1 y2 y3;
   partial z1;
```

Therefore, using 40T+16L+56T+12T, the minimum temporary storage equals 984 bytes (T=2+3+1 and L=T(T+1)/2).

Using the SPEARMAN, KENDALL, or HOEFFDING option requires additional temporary storage for each observation. For the most time-efficient processing, the amount of temporary storage in bytes is

40T+8K+8L*C+12T*N+28N+QS+QP+QK

where

QS=	0	with NOSIMPLE
	68T	otherwise
QP=	56K	with PEARSON and without NOMISS
	0	otherwise
QK =	32N	with KENDALL or HOEFFDING
	0	otherwise.

The following example uses KENDALL:

```
proc corr kendall;
   var x1 x2 x3;
```

Therefore, the minimum temporary storage in bytes is

40*3+8*6+8*6*1+12*3N+28N+3*68+32N = 420+96N

where N is the number of observations.

If M bytes are not available, PROC CORR must process the data multiple times to compute all the statistics. This reduces the minimum temporary storage you need by 12(T−2)N bytes. When this occurs, PROC CORR prints a note suggesting a larger memory region.

Statistical Computations: CORR Procedure

PROC CORR computes several parametric and nonparametric correlation statistics as measures of association. The formulas for computing these measures and the associated probabilities follow.

Pearson Product-Moment Correlation

The Pearson product-moment correlation is a parametric measure of association for two continuous random variables. The formula for the true Pearson product-moment correlation, denoted ρ_{xy}, is

$$\begin{aligned}\rho_{xy} &= \frac{\mathrm{cov}\left(x,y\right)}{\sqrt{\mathrm{var}\left(x\right)\mathrm{var}\left(y\right)}} \\ &= \frac{\mathrm{E}\left(\left(x-\mathrm{E}x\right)\left(y-\mathrm{E}y\right)\right)}{\sqrt{\mathrm{E}\left(x-\mathrm{E}x\right)^2\mathrm{E}\left(y-\mathrm{E}y\right)^2}}\end{aligned}$$

The sample correlation, such as a Pearson product-moment correlation or weighted product-moment correlation, estimates the true correlation. The formula for the Pearson product-moment correlation is

$$r_{xy} = \frac{\sum\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)}{\sqrt{\sum\left(x_i-\bar{x}\right)^2\sum\left(y_i-\bar{y}\right)^2}}$$

where $\bar{x}$ is the sample mean of x and $\bar{y}$ is the sample mean of y.

The formula for a weighted Pearson product-moment correlation is

$$r_{xy} = \frac{\sum w_i\left(x_i-\bar{x}_w\right)\left(y_i-\bar{y}_w\right)}{\sqrt{\sum w_i\left(x_i-\bar{x}_w\right)^2\sum w_i\left(y_i-\bar{y}_w\right)^2}}$$

where

$$\bar{x}_w = \sum w_i x_i / \sum w_i$$
$$\bar{y}_w = \sum w_i y_i / \sum w_i$$

Note that $\bar{x}_w$ is the weighted mean of x, $\bar{y}_w$ is the weighted mean of y, and w_i is the weight.

When one variable is dichotomous (0,1) and the other variable is continuous, a Pearson correlation is equivalent to a point biserial correlation. When both variables are dichotomous, a Pearson correlation coefficient is equivalent to the phi coefficient.

Spearman Rank-Order Correlation

Spearman rank-order correlation is a nonparametric measure of association based on the rank of the data values. The formula is

$$\theta = \frac{\sum \left(\mathrm{R}_i - \bar{\mathrm{R}}\right)\left(\mathrm{S}_i - \bar{\mathrm{S}}\right)}{\sqrt{\sum \left(\mathrm{R}_i - \bar{\mathrm{R}}\right)^2 \sum \left(\mathrm{S}_i - \bar{\mathrm{S}}\right)^2}}$$

where R_i is the rank of the $i\mathrm{th}$ x value, S_i is the rank of the $i\mathrm{th}$ y value, $\bar{\mathrm{R}}$ is the mean of the R_i values, and $\bar{\mathrm{S}}$ is the mean of the S_i values.

PROC CORR computes the Spearman's correlation by ranking the data and using the ranks in the Pearson product-moment correlation formula. In case of ties, the averaged ranks are used.

Kendall's tau-b

Kendall's tau-b is a nonparametric measure of association based on the number of concordances and discordances in paired observations. Concordance occurs when paired observations vary together, and discordance occurs when paired observations vary differently. The formula for Kendall's tau-b is

$$\tau = \frac{\sum_{i<j} \mathrm{sgn}\left(x_i - x_j\right) \mathrm{sgn}\left(y_i - y_j\right)}{\sqrt{\left(\mathrm{T}_0 - \mathrm{T}_1\right)\left(\mathrm{T}_0 - \mathrm{T}_2\right)}}$$

where

$$\mathrm{T}_0 = n\left(n-1\right)/2$$
$$\mathrm{T}_1 = \sum t_i\left(t_i - 1\right)/2$$
$$\mathrm{T}_2 = \sum u_i\left(u_i - 1\right)/2$$

and where t_i is the number of tied x values in the $i\mathrm{th}$ group of tied x values, u_i is the number of tied y values in the $i\mathrm{th}$ group of tied y values, f is the number of observations, and sgn(z) is defined as

$$\mathrm{sgn}(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z = 0 \\ -1 & \text{if } z < 0 \end{cases}$$

PROC CORR computes Kendall's correlation by ranking the data and using a method similar to Knight (1966). The data are double sorted by ranking observations according to values of the first variable and reranking the observations according to values of the second variable. PROC CORR computes Kendall's tau-b from the number of interchanges of the first variable and corrects for tied pairs (pairs of observations with equal values of X or equal values of Y).

Hoeffding's Measure of Dependence, D

Hoeffding's measure of dependence, D, is a nonparametric measure of association that detects more general departures from independence. The statistic approximates a weighted sum over observations of chi-square statistics for two-by-two classification tables (Hoeffding 1948). Each set of (x, y) values are cut points for the classification. The formula for Hoeffding's D is

$$\mathrm{D} = 30\,\frac{(n-2)(n-3)\mathrm{D}_1 + \mathrm{D}_2 - 2(n-2)\mathrm{D}_3}{n(n-1)(n-2)(n-3)(n-4)}$$

where

$$\begin{aligned} \mathrm{D}_1 &= \sum_i (\mathrm{Q}_i - 1)(\mathrm{Q}_i - 2) \\ \mathrm{D}_2 &= \sum_i (\mathrm{R}_i - 1)(\mathrm{R}_i - 2)(\mathrm{S}_i - 1)(\mathrm{S}_i - 2) \\ \mathrm{D}_3 &= \sum_i (\mathrm{R}_i - 2)(\mathrm{S}_i - 2)(\mathrm{Q}_i - 1) \end{aligned}$$

R_i is the rank of x_i, S_i is the rank of y_i, and Q_i (also called the bivariate rank) is 1 plus the number of points with both x and y values less than the $i\mathrm{th}$ point. A point that is tied on only the x value or y value contributes 1/2 to Q_i if the other value is less than the corresponding value for the $i\mathrm{th}$ point. A point that is tied on both x and y contributes 1/4 to Q_i.

PROC CORR obtains the Q_i values by first ranking the data. The data are then double sorted by ranking observations according to values of the first variable and reranking the observations according to values of the second variable. Hoeffding's D statistic is computed using the number of interchanges of the first variable.

When no ties occur among data set observations, the D statistic values are between -0.5 and 1, with 1 indicating complete dependence. However, when ties occur, the D statistic may result in a smaller value. That is, for a pair of variables with identical values, the Hoeffding's D statistic may be less than 1. With a large number of ties in a small data set, the D statistic may be less than -0.5 . For more information about Hoeffding's D, see Hollander and Wolfe (1973, p. 228).

Partial Correlation

A partial correlation measures the strength of a relationship between two variables, while controlling the effect of one or more additional variables. The Pearson partial correlation for a pair of variables may be defined as the correlation of errors after regression on the controlling variables. Let $\mathbf{y}= (y_1, y_2, \ldots, y_v)$ be the set of variables to correlate. Also let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be sets of regression parameters and $\mathbf{z}$ be the set of controlling variables, where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_v)$, $\boldsymbol{\beta}$ is the slope, and $\mathbf{z}= (z_1, z_2, \ldots, z_p)$. Suppose

$$\mathrm{E}(\mathbf{y}) = \boldsymbol{\alpha} + \mathbf{z}\boldsymbol{\beta}$$

is a regression model for $\mathbf{y}$ given $\mathbf{z}$. The population Pearson partial correlation between the $i\text{th}$ and the $j\text{th}$ variables of $\mathbf{y}$ given $\mathbf{z}$ is defined as the correlation between errors $(y_i - \mathrm{E}(y_i))$ and $(y_j - \mathrm{E}(y_j))$.

If the exact values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are unknown, you can use a sample Pearson partial correlation to estimate the population Pearson partial correlation. For a given sample of observations, you estimate the sets of unknown parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ using the least-squares estimators $\widehat{\boldsymbol{\alpha}}$ and $\widehat{\boldsymbol{\beta}}$. Then the fitted least-squares regression model is

$$\widehat{\mathbf{y}} = \widehat{\boldsymbol{\alpha}} + \mathbf{z}\widehat{\boldsymbol{\beta}}$$

The partial corrected sums of squares and crossproducts (CSSCP) of $\mathbf{y}$ given $\mathbf{z}$ are the corrected sums of squares and crossproducts of the residuals $\mathbf{y} - \widehat{\mathbf{y}}$. Using these partial corrected sums of squares and crossproducts, you can calculate the partial variances, partial covariances, and partial correlations.

PROC CORR derives the partial corrected sums of squares and crossproducts matrix by applying the Cholesky decomposition algorithm to the CSSCP matrix. For Pearson partial correlations, let $\mathbf{S}$ be the partitioned CSSCP matrix between two sets of variables, $\mathbf{z}$ and $\mathbf{y}$:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{\mathbf{zz}} & \mathbf{S}_{\mathbf{zy}} \\ \mathbf{S}'_{\mathbf{zy}} & \mathbf{S}_{\mathbf{yy}} \end{bmatrix}$$

PROC CORR calculates $\mathbf{S}_{\mathbf{yy}\cdot\mathbf{z}}$, the partial CSSCP matrix of $\mathbf{y}$ after controlling for $\mathbf{z}$, by applying the Cholesky decomposition algorithm sequentially on the rows associated with $\mathbf{z}$, the variables being partialed out.

After applying the Cholesky decomposition algorithm to each row associated with variables $\mathbf{z}$, PROC CORR checks all higher numbered diagonal elements associated with $\mathbf{z}$ for singularity. After the Cholesky decomposition, a variable is considered singular if the value of the corresponding diagonal element is less than p times the original unpartialed corrected sum of squares of that variable. You can specify the singularity criterion p using the SINGULAR= option. For Pearson partial correlations, a controlling variable $\mathbf{z}$ is considered singular if the R^2 for predicting this variable from the variables that are already partialed out exceeds $1 - p$. When this happens, PROC CORR excludes the variable from the analysis. Similarly, a variable is considered singular if the R^2 for predicting this variable from the controlling variables exceeds $1 - p$. When this happens, its associated diagonal element and all higher numbered elements in this row or column are set to zero.

After the Cholesky decomposition algorithm is performed on all rows associated with $\mathbf{z}$, the resulting matrix has the form

$$\begin{bmatrix} \mathbf{T}_{\mathbf{zz}} & \mathbf{T}_{\mathbf{zy}} \\ \mathbf{0} & \mathbf{S}_{\mathbf{yy}\cdot\mathbf{z}} \end{bmatrix}$$

where $\mathbf{T}_{\mathbf{zz}}$ is an upper triangular matrix with

$$\begin{aligned} \mathbf{T}'_{\mathbf{zz}}\mathbf{T}_{\mathbf{zz}} &= \mathbf{S}_{\mathbf{zz}'} \\ \mathbf{T}'_{\mathbf{zz}}\mathbf{T}_{\mathbf{zy}} &= \mathbf{S}_{\mathbf{zy}'} \\ \mathbf{S}_{\mathbf{yy}\cdot\mathbf{z}} &= \mathbf{S}_{\mathbf{yy}} - \mathbf{T}'_{\mathbf{zy}}\mathbf{T}_{\mathbf{zy}}. \end{aligned}$$

If $\mathbf{S}_{\mathbf{zz}}$ is positive definite, then the partial CSSCP matrix $\mathbf{S}_{\mathbf{yy}\cdot\mathbf{z}}$ is identical to the matrix derived from the formula

$$\mathbf{S}_{\mathbf{yy}\cdot\mathbf{z}} = \mathbf{S}_{\mathbf{yy}} - \mathbf{S}'_{\mathbf{zy}}\ \mathbf{S}^{-1}_{\mathbf{zz}}\ \mathbf{S}_{\mathbf{zy}}$$

The partial variance-covariance matrix is calculated with the variance divisor (VARDEF= option). PROC CORR can then use the standard Pearson correlation formula on the partial variance-covariance matrix to calculate the Pearson partial correlation matrix. Another way to calculate Pearson partial correlation is by applying the Cholesky decomposition algorithm directly to the correlation matrix and by using the correlation formula on the resulting matrix.

To derive the corresponding Spearman partial rank-order correlations and Kendall partial tau-b correlations, PROC CORR applies the Cholesky decomposition algorithm to the Spearman rank-order correlation matrix and Kendall tau-b correlation matrix and uses the correlation formula. The singularity criterion for nonparametric partial correlations is identical to Pearson partial correlation except that PROC CORR uses a matrix of nonparametric correlations and sets a singular variable's associated correlations to missing. The partial tau-b correlations range from –1 to 1. However, the sampling distribution of this partial tau-b is unknown; therefore, the probability values are not available.

When a correlation matrix (Pearson, Spearman, or Kendall tau-b correlation matrix) is positive definite, the resulting partial correlation between variables x and y after adjusting for a single variable z is identical to that obtained from the first-order partial correlation formula

$$r_{xy\cdot z} = \frac{r_{xy} - r_{xz}r_{yz}}{\sqrt{\left(1 - r_{xz}^2\right)\left(1 - r_{yz}^2\right)}}$$

where r_{xy}, r_{xz}, and r_{yz} are the appropriate correlations.

The formula for higher-order partial correlations is a straightforward extension of the above first-order formula. For example, when the correlation matrix is positive definite, the partial correlation between x and y controlling for both z_1 and z_2 is identical to the second-order partial correlation formula

$$r_{xy\cdot z_1 z_2} = \frac{r_{xy\cdot z_1} - r_{xz_2\cdot z_1} r_{yz_2\cdot z_1}}{\sqrt{\left(1 - r^2_{xz_2\cdot z_1}\right)\left(1 - r^2_{yz_2\cdot z_1}\right)}}$$

where $r_{xy\cdot z_1}$, $r_{xz_2\cdot z_1}$, and $r_{yz_2\cdot z_1}$ are first-order partial correlations among variables x, y, and z_2 given z_1.

Cronbach's Coefficient Alpha

Analyzing latent constructs such as job satisfaction, motor ability, sensory recognition, or customer satisfaction requires instruments to accurately measure the constructs. Interrelated items may be summed to obtain an overall score for each participant. Cronbach's coefficient alpha estimates the reliability of this type of scale by determining the internal consistency of the test or the average correlation of items within the test (Cronbach 1951).

When a value is recorded, the observed value contains some degree of measurement error. Two sets of measurements on the same variable for the same individual may not have identical values. However, repeated measurements for a series of individuals will show some consistency. Reliability measures internal consistency from one set of measurements to another. The observed value Y is divided into two components, a true value T and a measurement error E. The measurement error is assumed to be independent of the true value, that is,

$$\mathrm{Y} = \mathrm{T} + \mathrm{E}\ , \quad \mathrm{cov}\left(\mathrm{T}, \mathrm{E}\right) = 0$$

The reliability coefficient of a measurement test is defined as the squared correlation between the observed value Y and the true value T, that is,

$$\rho^2\left(\mathrm{Y}, \mathrm{T}\right) = \frac{\mathrm{cov}\left(\mathrm{Y}, \mathrm{T}\right)^2}{\mathrm{var}\left(\mathrm{Y}\right)\mathrm{var}\left(\mathrm{T}\right)} = \frac{\mathrm{var}\left(\mathrm{T}\right)^2}{\mathrm{var}\left(\mathrm{Y}\right)\mathrm{var}\left(\mathrm{T}\right)} = \frac{\mathrm{var}\left(\mathrm{T}\right)}{\mathrm{var}\left(\mathrm{Y}\right)}$$

which is the proportion of the observed variance due to true differences among individuals in the sample. If Y is the sum of several observed variables measuring the same feature, you can estimate var(T). Cronbach's coefficient alpha, based on a lower bound for var(T), is an estimate of the reliability coefficient.

Suppose p variables are used with $\mathrm{Y}_j = \mathrm{T}_j + \mathrm{E}_j$ for $j = 1, 2, \ldots, p$, where Y_j is the observed value, T_j is the true value, and E_j is the measurement error. The measurement errors (E_j) are independent of the true values (T_j) and are also independent of each other. Let $\mathrm{Y}_0 = \sum \mathrm{Y}_j$ be the total observed score and $\mathrm{T}_0 = \sum \mathrm{T}_j$ be the total true score. Because

$$(p-1)\sum \mathrm{var}\left(\mathrm{T}_j\right) \geq \sum_{i \neq j} \mathrm{cov}\left(\mathrm{T}_i, \mathrm{T}_j\right)\ ,$$

a lower bound for $\text{var}(\text{T}_0)$ is given by

$$\frac{p}{p-1} \sum_{i \neq j} \text{cov}(\text{T}_i, \text{T}_j)$$

With $\text{cov}(\text{Y}_i, \text{Y}_j) = \text{cov}(\text{T}_i, \text{T}_j)$ for $i \neq j$, a lower bound for the reliability coefficient is then given by the Cronbach's coefficient alpha:

$$\boldsymbol{\alpha} = \left(\frac{p}{p-1}\right) \frac{\sum_{i \neq j} \text{cov}(\text{Y}_i, \text{Y}_j)}{\text{var}(\text{Y}_0)}$$

$$= \left(\frac{p}{p-1}\right) \left(1 - \frac{\sum_{j} \text{var}(\text{Y}_j)}{\text{var}(\text{Y}_0)}\right)$$

If the variances of the items vary widely, you can standardize the items to a standard deviation of 1 before computing the coefficient alpha. If the variables are dichotomous (0,1), the coefficient alpha is equivalent to the Kuder-Richardson 20 (KR-20) reliability measure.

When the correlation between each pair of variables is 1, the coefficient alpha has a maximum value of 1. With negative correlations between some variables, the coefficient alpha can have a value less than zero. The larger the overall alpha coefficient, the more likely that items contribute to a reliable scale. Nunnally (1978) suggests .70 as an acceptable reliability coefficient; smaller reliability coefficients are seen as inadequate. However, this varies by discipline.

To determine how each item reflects the reliability of the scale, you calculate a coefficient alpha after deleting each variable independently from the scale. The Cronbach's coefficient alpha from all variables except the $k\text{th}$ variable is given by

$$\boldsymbol{\alpha}_k = \left(\frac{p-1}{p-2}\right) \left(1 - \frac{\sum_{i \neq k} \text{var}(\text{Y}_i)}{\text{var}\left(\sum_{i \neq k} \text{Y}_i\right)}\right)$$

If the reliability coefficient increases after deleting an item from the scale, you can assume that the item is not correlated highly with other items in the scale. Conversely, if the reliability coefficient decreases you can assume that the item is highly correlated with other items in the scale. See *SAS Communications*, 4th Quarter 1994, for more information on how to interpret Cronbach's coefficient alpha.

Listwise deletion of observations with missing values is necessary to correctly calculate Cronbach's coefficient alpha. PROC CORR does not automatically use listwise deletion when you specify ALPHA. Therefore, use the NOMISS option if the data set contains missing values. Otherwise, PROC FREQ prints a warning message in the SAS log indicating the need to use NOMISS with ALPHA.

Probability Values

Probability values for the Pearson and Spearman correlations are computed by treating

$$\frac{(n-2)^{1/2}\,r}{(1-r^2)^{1/2}}$$

as coming from a t distribution with $n-2$ degrees of freedom, where r is the appropriate correlation.

Probability values for the Pearson and Spearman partial correlations are computed by treating

$$\frac{(n-k-2)^{1/2}\,r}{(1-r^2)^{1/2}}$$

as coming from a t distribution with $n-k-2$ degrees of freedom, where r is the appropriate partial correlation and k is the number of variables being partialed out.

Probability values for Kendall correlations are computed by treating

$$\frac{\mathrm{s}}{\sqrt{\mathrm{var}\,(\mathrm{s})}}$$

as coming from a normal distribution when

$$\mathrm{s} = \sum_{i<j} \mathrm{sgn}\,(x_i - x_j)\,\mathrm{sgn}\,(y_i - y_j)$$

and where x_i are the values of the first variable, y_i are the values of the second variable, and the function sgn(z) is defined as

$$\mathrm{sgn}\,(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z = 0 \\ -1 & \text{if } z < 0 \end{cases}$$

The formula for the variance of s, var(s), is computed as

$$\mathrm{var}\,(\mathrm{s}) = \frac{\mathrm{v}_0 - \mathrm{v}_t - \mathrm{v}_u}{18} + \frac{\mathrm{v}_1}{2n\,(n-1)} + \frac{\mathrm{v}_2}{9n\,(n-1)\,(n-2)}$$

where

$$\mathrm{v}_0 = n\,(n-1)\,(2n+5)$$
$$\mathrm{v}_t = \sum t_i\,(t_i - 1)\,(2t_i + 5)$$

$$v_u = \sum u_i (u_i - 1)(2u_i + 5)$$
$$v_1 = \left(\sum t_i (t_i - 1)\right)\left(\sum u_i (u_i - 1)\right)$$
$$v_2 = \left(\sum t_i (t_i - 1)(t_i - 2)\right)\left(\sum u_i (u_i - 1)(u_i - 2)\right)$$

The sums are over tied groups of values where t_i is the number of tied x values and u_i is the number of tied y values (Noether 1967). The sampling distribution of Kendall's partial tau-b is unknown; therefore, the probability values are not available.

The probability values for Hoeffding's D statistic are computed using the asymptotic distribution computed by Blum, Kiefer, and Rosenblatt (1961). The formula is

$$\frac{(n-1)\pi^4}{60}\mathrm{D} + \frac{\pi^4}{72}$$

which comes from the asymptotic distribution. When the sample size is less than 10, see the tables for the distribution of D in Hollander and Wolfe (1973).

Results: CORR Procedure

Missing Values

By default, PROC CORR uses *pairwise deletion* when observations contain missing values. PROC CORR includes all nonmissing pairs of values for each pair of variables in the statistical computations. Therefore, the correlations statistics may be based on different numbers of observations.

If you specify the NOMISS option, PROC CORR uses *listwise deletion* when a value of the BY, FREQ, VAR, WEIGHT, or WITH statement variable is missing. PROC CORR excludes all observations with missing values from the analysis. Therefore, the number of observations for each pair of variables is identical. The PARTIAL statement always excludes the observations with missing values by automatically invoking NOMISS. Listwise deletion is needed to correctly calculate Cronbach's coefficient alpha when data are missing. If a data set contains missing values, when you specify ALPHA use the NOMISS option

There are two reasons to specify NOMISS and, thus, to avoid pairwise deletion. First, NOMISS is computationally more efficient, so you use fewer computer resources. Second, if you use the correlations as input to regression or other statistical procedures, a pairwise-missing correlation matrix leads to several statistical difficulties. Pairwise correlation matrices may not be nonnegative definite, and the pattern of missing values may bias the results.

ODS Table Names

PROC CORR assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. For more information, see *SAS Output Delivery System User's Guide*.

Table 12.2 ODS Tables Produced with the PROC CORR Statement

ODS Name	Description	Option
Cov	Covariances Row/Column variable variance, DF (missing values)	COV
CronbachAlpha	Coefficient Alpha	ALPHA
CronbachAlphaDel	Coefficient Alpha with Deleted Variable	ALPHA
Csscp	Corrected sums of squares and crossproducts Row/Column variable corrected sums of squares (missing values)	CSSCP
HoeffdingCorr	Hoeffding's D statistics p values (NOPROB is not specified) number of observations (missing values)	HOEFFDING
KendallCorr	Kendall tau-b coefficients p values (NOPROB is not specified) number of observations (missing values)	KENDALL
PearsonCorr	Pearson correlations p-value (NOPROB is not specified) number of observations (missing values)	omit NOCORR or PEARSON
SimpleStats	Simple descriptive statistics	omit NOSIMPLE
SpearmanCorr	Spearman correlations p values (NOPROB is not specified) number of observations (missing values)	SPEARMAN
Sscp	Sums of squares and crossproducts Row/Column variable sums of squares (missing values)	SSCP
VarInformation	Variable Information	default

Table 12.3 ODS Tables Produced with the PARTIAL Statement

ODS Name	Description	PROC CORR statement Option
PartialCsscp	Partial corrected sums of squares and crossproduct	CSSCP
PartialCov	Partial covariances	COV
PartialKendallCorr	Partial Kendall tau-b coefficients	KENDALL
PartialPearsonCorr	Partial Pearson correlations p values (NOPROB is not specified)	default
PartialSpearmanCorr	Partial Spearman correlations p values (NOPROB is not specified)	SPEARMAN

Output

By default, PROC CORR prints a report that includes descriptive statistics and correlation statistics for each variable. The descriptive statistics include the number of observations with nonmissing values, the mean, the standard deviation, the minimum, and the maximum. PROC CORR reports the following additional descriptive statistics when you request various correlation statistics:

sum
: for Pearson correlation only

median
: for nonparametric measures of association

partial variance
: for Pearson partial correlation

partial standard deviation
: for Pearson partial correlation.

If variable labels are available, PROC CORR labels the variables.

When you specify the CSSCP, SSCP, or COV option, the appropriate sum-of-squares and crossproducts and covariance matrix appears at the top of the correlation report. If the data set contains missing values, PROC CORR prints additional statistics for each pair of variables. These statistics, calculated from the observations with nonmissing row and column variable values, may include

SSCP(W','V')
: uncorrected sum-of-squares and crossproducts

USS(W')
: uncorrected sum-of-squares for the row variable

USS(V')
: uncorrected sum-of-squares for the column variable

CSSCP(W','V')
: corrected sum-of-squares and crossproducts

CSS(W')
: corrected sum-of-squares for the row variable

CSS(V')
: corrected sum-of-squares for the column variable

COV (W','V')
: covariance

VAR (W')
: variance for the row variable

VAR (V')
: variance for the column variable

DF(W',V')
: divisor for calculating covariance and variances.

For each pair of variables, PROC CORR always prints the correlation coefficients, the number of observations used to calculate the coefficient, and the significance probability. When you specify the ALPHA option, PROC CORR prints Cronbach's coefficient alpha, the correlation between the variable and the total of the remaining variables, and Cronbach's coefficient alpha using the remaining variables for the raw variables and the standardized variables.

Output Data Sets

When you specify the OUTP=, OUTS=, OUTK=, or OUTH= option, PROC CORR creates an output data set containing statistics for Pearson correlation, Spearman correlation, Kendall correlation, or Hoeffding's D, respectively. By default, the output data set is a special data set type (TYPE=CORR) that many SAS/STAT procedures recognize, including PROC REG and PROC FACTOR. When you specify the NOCORR option and the COV, CSSCP, or SSCP option, use the TYPE= data set option to change the data set type to COV, CSSCP, or SSCP. For example, the following statement

```
proc corr nocorr cov outp=b(type=cov);
```

specifies the output data set type as COV.

PROC CORR does not print the output data set. Use PROC PRINT, PROC REPORT, or another SAS reporting tool to print the output data set.

The output data set includes the following variables

BY variables
: identifies the BY group when using a BY statement.

TYPE variable
: identifies the type of observation.

NAME variable
: identifies the variable that corresponds to a given row of the correlation matrix.

INTERCEP variable
: identifies variable sums when specifying the SSCP option.

VAR variables
: identifies the variables listed in the VAR statement.

You can use a combination of the _TYPE_ and _NAME_ variables to identify the contents of an observation. The _NAME_ variable indicates which row of the correlation matrix the observation corresponds to. The values of the _TYPE_ variable are

SSCP
: uncorrected sums of squares and crossproducts

CSSCP
: corrected sums of squares and crossproducts

COV
: covariances

MEAN
: mean of each variable

STD
: standard deviation of each variable

N
: number of nonmissing observations for each variable

SUMWGT
: sum of the weights for each variable when using a WEIGHT statement

CORR
: correlation statistics for each variable.

When you specify the SSCP option, the OUTP= data set includes an additional observation that contains intercept values. When you specify the ALPHA option, the OUTP= data set also includes observations with the following _TYPE_ values:

RAWALPHA
: Cronbach's coefficient alpha for raw variables

STDALPHA
: Cronbach's coefficient alpha for standardized variables

RAWALDEL
: Cronbach's coefficient alpha for raw variables after deleting one variable

STDALDEL
: Cronbach's coefficient alpha for standardized variables after deleting one variable

RAWCTDEL
: correlation between a raw variable and the total of the remaining raw variables

STDCTDEL
: correlation between a standardized variable and the total of the remaining standardized variables.

When you use a PARTIAL statement, the previous statistics are calculated after the variables are partialed. If PROC CORR computes Pearson correlation statistics, MEAN equals zero and STD equals the partial standard deviation associated with the partial variance for the OUTP=, OUTK=, or OUTS= data set. Otherwise, PROC CORR assigns missing values to MEAN and STD. Output 12.4 on page 291 lists the observations in an OUTP= data set when the COV option and PARTIAL statement are used to compute Pearson partial correlations. The _TYPE_ variable identifies COV, MEAN, STD, N, and CORR as the statistical values for the variables Weight, Oxygen, and Runtime. MEAN always equals 0, while STD is a partial standard deviation.

Output 12.4 OUTP= Data Set with Pearson Partial Correlations

```
   Pearson Correlation Statistics Using the PARTIAL Statement  1
                 Output Data Set from PROC CORR

     _TYPE_    _NAME_       Weight      Oxygen     Runtime

      COV      Weight      72.4374    -12.7511      2.0677
      COV      Oxygen     -12.7511     27.0165     -5.5937
      COV      Runtime      2.0677     -5.5937      1.9451
      MEAN                  0.0000      0.0000      0.0000
      STD                   8.5110      5.1977      1.3947
      N                    28.0000     28.0000     28.0000
      CORR     Weight       1.0000     -0.2882      0.1742
      CORR     Oxygen      -0.2882      1.0000     -0.7716
      CORR     Runtime      0.1742     -0.7716      1.0000
```

Examples: CORR Procedure

Example 1: Computing Pearson Correlations and Other Measures of Association

Procedure features:

PROC CORR statement options:
- HOEFFDING
- PEARSON
- SPEARMAN

VAR statement

This example
- □ produces a correlation analysis with descriptive statistics, Pearson product-moment correlation, Spearman rank-order correlation, and Hoeffding's measure of dependence, D
- □ selects the analysis variables.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the FITNESS data set. This data set contains measurements from a study of physical fitness of 30 participants between the ages 38 and 57. Each observation represents one person. Two observations contain missing values.

```
data fitness;
   input Age Weight Runtime Oxygen @@;
   datalines;
57 73.37 12.63 39.407   54 79.38 11.17 46.080
52 76.32 9.63  45.441   50 70.87 8.92    .
51 67.25 11.08 45.118   54 91.63 12.88 39.203
51 73.71 10.47 45.790   57 59.08 9.93  50.545
49 76.32  .    48.673   48 61.24 11.5  47.920
52 82.78 10.5  47.467   44 73.03 10.13 50.541
45 87.66 14.03 37.388   45 66.45 11.12 44.754
47 79.15 10.6  47.273   54 83.12 10.33 51.855
49 81.42 8.95  40.836   51 77.91 10.00 46.672
48 91.63 10.25 46.774   49 73.37 10.08 50.388
44 89.47 11.37 44.609   40 75.07 10.07 45.313
44 85.84 8.65  54.297   42 68.15 8.17  59.571
38 89.02 9.22  49.874   47 77.45 11.63 44.811
40 75.98 11.95 45.681   43 81.19 10.85 49.091
44 81.42 13.08 39.442   38 81.87 8.63  60.055
;
```

Generate the correlation statistics. PEARSON, SPEARMAN, and HOEFFDING compute correlation statistics. When you request nonparametric correlations, specify PEARSON to compute Pearson correlations.

```
proc corr data=fitness pearson spearman hoeffding;
```

Specify the analysis variables. The VAR statement specifies that Weight, Oxygen, and Runtime are the analysis variables and specifies the order in which to print them.

```
   var weight oxygen runtime;
```

Specify the title. The TITLE statement specifies a title for the report.

```
   title 'Measures of Association for';
   title2 'a Physical Fitness Study';
run;
```

Output

The correlation report includes descriptive statistics, Pearson's rho, Spearman's rho, and Hoeffding's D. The report uses the median, instead of the sum, as a descriptive measure when PROC CORR computes nonparametric measures of association.

Because missing data are excluded pairwise, the number of observations PROC CORR uses to calculate the correlation coefficients varies.

```
                         Measures of Association for                       1
                           a Physical Fitness Study

                              The CORR Procedure

                      3  Variables:  Weight   Oxygen   Runtime


                              Simple Statistics

Variable          N        Mean     Std Dev      Median     Minimum     Maximum

Weight           30    77.70500     8.34152    77.68000    59.08000    91.63000
Oxygen           29    47.06445     5.32129    46.67200    37.38800    60.05500
Runtime          29    10.61448     1.41655    10.47000     8.17000    14.03000


                       Pearson Correlation Coefficients
                          Prob > |r| under H0: Rho=0
                            Number of Observations

                            Weight        Oxygen       Runtime

              Weight       1.00000      -0.19900       0.15155
                                          0.3007        0.4326
                                30            29            29

              Oxygen      -0.19900       1.00000      -0.78346
                            0.3007                      <.0001
                                29            29            28

              Runtime      0.15155      -0.78346       1.00000
                            0.4326        <.0001
                                29            28            29


                      Spearman Correlation Coefficients
                          Prob > |r| under H0: Rho=0
                            Number of Observations

                            Weight        Oxygen       Runtime

              Weight       1.00000      -0.13110       0.10546
                                          0.4979        0.5861
                                30            29            29

              Oxygen      -0.13110       1.00000      -0.68363
                            0.4979                      <.0001
                                29            29            28

              Runtime      0.10546      -0.68363       1.00000
                            0.5861        <.0001
                                29            28            29
```

```
                    Measures of Association for                          2
                      a Physical Fitness Study

                         The CORR Procedure

                  Hoeffding Dependence Coefficients
                        Prob > D under H0: D=0
                        Number of Observations

                          Weight         Oxygen        Runtime

             Weight      0.97559       -0.01789       -0.02418
                          <.0001         0.9775         1.0000
                              30             29             29

             Oxygen     -0.01789        1.00000        0.16554
                          0.9775                        <.0001
                              29             29             28

             Runtime    -0.02418        0.16554        1.00000
                          1.0000         <.0001
                              29             28             29
```

Example 2: Computing Rectangular Correlation Statistics with Missing Data

Procedure features:

PROC CORR statement options:

- COV
- NOSIMPLE
- SSCP

VAR statement

WITH statement

This example

- suppresses descriptive statistics
- prints uncorrected sum-of-squares and crossproducts
- calculates a rectangular covariance matrix
- calculates a rectangular correlation matrix
- excludes observations with missing values using pairwise deletion (default method).

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the SETOSA data set. This data set contains measurements for four iris parts: sepal length, sepal width, petal length, and petal width based on Fisher's iris data (1936). Fifty iris specimens from the species *Iris setosa* are used. Each observation represents one specimen. Three observations contain missing values. The LABEL statement associates a label with each variable.

```
data setosa;
   input SepalLength SepalWidth PetalLength PetalWidth @@;
   label sepallength='Sepal Length in mm.'
         sepalwidth='Sepal Width in mm.'
         petallength='Petal Length in mm.'
         petalwidth='Petal Width in mm.';
   datalines;
50 33 14 02   46 34 14 03   46 36 .  02
51 33 17 05   55 35 13 02   48 31 16 02
52 34 14 02   49 36 14 01   44 32 13 02
50 35 16 06   44 30 13 02   47 32 16 02
48 30 14 03   51 38 16 02   48 34 19 02
50 30 16 02   50 32 12 02   43 30 11 .
58 40 12 02   51 38 19 04   49 30 14 02
51 35 14 02   50 34 16 04   46 32 14 02
57 44 15 04   50 36 14 02   54 34 15 04
52 41 15 .    55 42 14 02   49 31 15 02
54 39 17 04   50 34 15 02   44 29 14 02
47 32 13 02   46 31 15 02   51 34 15 02
50 35 13 03   49 31 15 01   54 37 15 02
54 39 13 04   51 35 14 03   48 34 16 02
48 30 14 01   45 23 13 03   57 38 17 03
51 38 15 03   54 34 17 02   51 37 15 04
52 35 15 02   53 37 15 02
;
```

Generate the correlation statistics but suppress descriptive statistics. SSCP displays the uncorrected sum-of-squares and crossproducts matrix and invokes PEARSON. COV calculates the covariance matrix. NOSIMPLE suppresses descriptive statistics.

```
proc corr data=setosa sscp cov nosimple;
```

Generate a rectangular correlation matrix. The WITH statement together with the VAR statement produces a rectangular correlation matrix. The matrix rows are PetalLength and PetalWidth while the matrix columns are SepalLength and SepalWidth.

```
   var sepallength sepalwidth;
   with petallength petalwidth;
```

Specify the title. The TITLE statement specifies a title for the report.

```
   title 'Fisher (1936) Iris Setosa Data';
run;
```

Output

The correlation report includes rectangular sum-of-squares and crossproducts, covariances, and the correlation matrix using the two WITH variables and two VAR variables. The descriptive statistics do not appear. PROC CORR uses variable labels to label matrix rows (WITH variables).

PROC CORR calculates sum-of-squares and crossproducts and covariances statistics for each pair of variables by using observations with nonmissing row and column variable values.

Because missing data are excluded pairwise, the number of observations PROC CORR uses to calculate the correlation coefficients changes.

```
                      Fisher (1936) Iris Setosa Data                       1

                            The CORR Procedure

                  2 With Variables:  PetalLength PetalWidth
                  2      Variables:  SepalLength SepalWidth


                     Sums of Squares and Crossproducts
                       SSCP / Row Var SS / Col Var SS

                                   SepalLength        SepalWidth

              PetalLength          36214.00000       24756.00000
              Petal Length in mm.  10735.00000       10735.00000
                                   123793.0000        58164.0000

              PetalWidth            6113.00000        4191.00000
              Petal Width in mm.     355.00000         355.00000
                                   121356.0000        56879.0000


                         Variances and Covariances
               Covariance / Row Var Variance / Col Var Variance / DF

                                   SepalLength        SepalWidth

              PetalLength          1.270833333       1.363095238
              Petal Length in mm.  2.625000000       2.625000000
                                   12.33333333       14.60544218
                                            48                48

              PetalWidth           0.911347518       1.048315603
              Petal Width in mm.   1.063386525       1.063386525
                                   11.80141844       13.62721631
                                            47                47


                      Pearson Correlation Coefficients
                         Prob > |r| under H0: Rho=0
                           Number of Observations

                                         Sepal         Sepal
                                        Length         Width

                  PetalLength          0.22335       0.22014
                  Petal Length in mm.   0.1229        0.1285
                                            49            49

                  PetalWidth           0.25726       0.27539
                  Petal Width in mm.    0.0775        0.0582
                                            48            48
```

Example 3: Computing Cronbach's Coefficient Alpha

Procedure features:
PROC CORR statement options:
ALPHA
NOCORR
NOMISS

This example

- computes Cronbach's coefficient alpha for a multiple-item mixed-rating scale
- suppresses Pearson correlation statistics
- excludes observations with missing values using listwise deletion.

This example does not examine the correlation matrix but assumes that all items are positively correlated. Normally, you want to examine the correlation and covariance matrices to make sure that all variables are positively correlated. Positive correlation is needed because items measure a common entity. You exclude negatively correlated items from the analysis because they do not measure the same construct.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the PYSCHDAT data set. This data set contains responses to a questionnaire assessing the mental stability of 30 randomly selected female psychiatric patients.* Each observation represents one patient. The scale includes seven items. The LABEL statement provides a label for each item. Seven observations contain missing values.

```
data psychdat;
   input Age Anxiety Depression Sleep Sex Life WeightChange @@;
   label age             = 'age in years'
         anxiety         = 'anxiety level'
         depression      = 'depression level'
         sleep           = 'normal sleep (1=y 2=n)'
         sex             = 'sexual (1=n 2=y)'
         life            = 'suicidal (1=n 2=y)'
         weightchange    = 'recent weight change';
    datalines;
39  2   2   2   2   2  4.9 41  2   2   2   2   2  2.2
```

* Data are from *Assignments in Applied Statistics* by Simon Conrad. Copyright © 1989 by John Wiley & Sons, Inc. Reprinted with permission from the publisher.

```
42  3  3  .  2  2  4.0 30  2  2  2  2  2 -2.6
35  2  1  1  2  1 -0.3 44  .  1  2  1  1  0.9
31  2  2  .  2  2 -1.5 39  3  2  2  2  1  3.5
35  3  2  2  2  2 -1.2 33  2  2  2  2  2  0.8
38  2  1  1  1  1 -1.9 31  2  2  2  .  1  5.5
40  3  2  2  2  1  2.7 44  2  2  2  2  2  4.4
43  3  2  2  2  2  3.2 32  1  1  1  2  1 -1.5
32  1  2  2  .  1 -1.9 43  4  3  2  2  2  8.3
46  3  2  2  2  2  3.6 30  2  2  2  2  1  1.4
34  3  3  .  2  2   .  37  3  2  2  2  1   .
35  2  1  2  2  1 -1.0 45  2  2  2  2  2  6.5
35  2  2  2  2  1 -2.1 31  2  2  2  2  1 -0.4
32  2  2  2  2  1 -1.9 44  2  2  2  2  2  3.7
40  3  3  2  2  2  4.5 42  3  3  2  2  2  4.2
;
```

Generate Cronbach's alpha for all the analysis variables. Suppress Pearson correlation statistics. ALPHA computes Cronbach's alpha and invokes PEARSON. NOCORR suppresses Pearson correlation statistics. NOMISS excludes observations with missing values. Omitting a VAR statement causes PROC CORR to use all numeric variables.

```
proc corr data=psychdat alpha nocorr nomiss;
```

Specify the title. The TITLE statement specifies a title for the report.

```
   title1 'Mental Stability Scale for Female Psychiatric Patients';
run;
```

Output

The correlation report includes descriptive statistics and Cronbach's coefficient alpha, the correlation between the variable and the total of the remaining variables, and Cronbach's coefficient alpha using the remaining variables for both the raw variables and the standardized variables. These calculations use the 23 observations without missing values.

Because the variances of some variables vary widely, you use the standardized scores to estimate reliability. The overall standardized alpha of .85 is an acceptable reliability coefficient. This is greater than Nunnally's suggested value of .70.

The standardized alpha provides information on how each item reflects the reliability of the scale. Notice that the standardized alpha decreases after removing Depression from the construct. Therefore, this variable appears strongly correlated with other items in the scale. The standardized alpha increases slightly after removing Sex from the construct. Thus, removing this variable from the scale makes the construct more reliable.

```
             Mental Stability Scale for Female Psychiatric Patients           1

                               The CORR Procedure

    7  Variables:    Age          Anxiety      Depression   Sleep        Sex
                     Life         WeightChange


                               Simple Statistics

       Variable               N          Mean       Std Dev           Sum

       Age                   23      37.91304       5.13378     872.00000
       Anxiety               23       2.34783       0.64728      54.00000
       Depression            23       1.95652       0.56232      45.00000
       Sleep                 23       1.86957       0.34435      43.00000
       Sex                   23       1.95652       0.20851      45.00000
       Life                  23       1.56522       0.50687      36.00000
       WeightChange          23       1.78261       3.06381      41.00000

                               Simple Statistics

       Variable           Minimum       Maximum    Label

       Age               30.00000      46.00000    age in years
       Anxiety            1.00000       4.00000    anxiety level
       Depression         1.00000       3.00000    depression level
       Sleep              1.00000       2.00000    normal sleep (1=y 2=n)
       Sex                1.00000       2.00000    sexual (1=n 2=y)
       Life               1.00000       2.00000    suicidal (1=n 2=y)
       WeightChange      -2.60000       8.30000    recent weight change


                           Cronbach Coefficient Alpha

                          Variables              Alpha
                          ----------------------------
                          Raw                 0.627754
                          Standardized        0.845339
```

```
   Cronbach Coefficient Alpha with Deleted Variable

                     Raw Variables            Standardized Variables

Deleted          Correlation                     Correlation
Variable          with Total            Alpha      with Total            Alpha
----------------------------------------------------------------------------
Age                 0.742614         0.557515        0.546856         0.832207


              Cronbach Coefficient Alpha with Deleted Variable

                   Deleted
                   Variable        Label
                   --------------------------------------
                   Age             age in years
```

```
  Mental Stability Scale for Female Psychiatric Patients           2

                            The CORR Procedure

              Cronbach Coefficient Alpha with Deleted Variable

                     Raw Variables            Standardized Variables

Deleted          Correlation                     Correlation
Variable          with Total            Alpha      with Total            Alpha
----------------------------------------------------------------------------
Anxiety             0.577129         0.600944        0.590851         0.825643
Depression          0.554983         0.608273        0.770956         0.797610
Sleep               0.378930         0.630242        0.618367         0.821482
Sex                 0.155115         0.642017        0.333368         0.862537
Life                0.622207         0.607333        0.625338         0.820421
WeightChange        0.843939         0.341006        0.749261         0.801087


              Cronbach Coefficient Alpha with Deleted Variable

                   Deleted
                   Variable        Label
                   --------------------------------------
                   Anxiety         anxiety level
                   Depression      depression level
                   Sleep           normal sleep (1=y 2=n)
                   Sex             sexual (1=n 2=y)
                   Life            suicidal (1=n 2=y)
                   WeightChange    recent weight change
```

Example 4: Storing Partial Correlations in an Output Data Set

Procedure features:

PROC CORR statement options:

- COV
- KENDALL
- NOSIMPLE
- OUTP=
- SPEARMAN

PARTIAL statement
VAR statement

Data set: FITNESS on page 292

This example

- suppresses descriptive statistics
- calculates three types of partial correlation coefficients
- calculates a partial covariance matrix
- excludes observations with missing values using listwise deletion
- selects the analysis variables
- creates an output data set with Pearson correlation statistics.

See “Output Data Sets” on page 290 for a listing of the output data set.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=120 pagesize=60;
```

Generate the correlation statistics and create the output data set FITCORR. SPEARMAN and KENDALL request correlation statistics. COV calculates the covariance matrix and invokes PEARSON. NOSIMPLE suppresses descriptive statistics. OUT= creates the FITCORR data set that contains Pearson correlation statistics.

```
proc corr data=fitness spearman kendall cov nosimple
          outp=fitcorr;
```

Specify the analysis variable. The VAR statement specifies that Weight, Oxygen, and Runtime are the analysis variables and specifies the order in which to print them.

```
   var weight oxygen runtime;
```

Generate the partial correlations. The PARTIAL statement calculates partial correlations using Age as the controlling variable.

```
   partial age;
```

Specify the labels for the report. The LABEL statement associates a label with each variable for the duration of the PROC step.

```
   label age      = 'Age of subject'
         weight   = 'Wt in kg'
         runtime  = '1.5 mi in minutes'
         oxygen   = 'O2 use';
```

Specify the title. The TITLE statement specifies a title for the report.

```
   title1 'Partial Correlations for a Fitness and Exercise Study';
run;
```

Output

The report includes a partial covariance matrix and partial correlations for Pearson's rho, Spearman's rho, and Kendall's tau-b. The *p*-values for Kendall's tau-b are not available. Because observations with missing data are excluded, PROC CORR uses 28 observations to calculate correlation coefficients.

```
                       Partial Correlations for a Fitness and Exercise Study                          1

                                         The CORR Procedure

                          1 Partial Variables:    Age
                          3         Variables:    Weight   Oxygen   Runtime


                                Partial Covariance Matrix, DF = 26

                                                  Weight            Oxygen           Runtime

          Weight       Wt in kg               72.43742055      -12.75113194        2.06766763
          Oxygen       O2 use                -12.75113194       27.01654904       -5.59370556
          Runtime      1.5 mi in minutes       2.06766763       -5.59370556        1.94512451


                        Pearson Partial Correlation Coefficients, N = 28
                               Prob > |r| under H0: Partial Rho=0

                                             Weight        Oxygen       Runtime

                    Weight                  1.00000      -0.28824       0.17419
                    Wt in kg                               0.1448        0.3849

                    Oxygen                 -0.28824       1.00000      -0.77163
                    O2 use                   0.1448                      <.0001

                    Runtime                 0.17419      -0.77163       1.00000
                    1.5 mi in minutes        0.3849        <.0001


                       Spearman Partial Correlation Coefficients, N = 28
                               Prob > |r| under H0: Partial Rho=0

                                             Weight        Oxygen       Runtime

                    Weight                  1.00000      -0.16407       0.08708
                    Wt in kg                               0.4135        0.6658

                    Oxygen                 -0.16407       1.00000      -0.67112
                    O2 use                   0.4135                      0.0001

                    Runtime                 0.08708      -0.67112       1.00000
                    1.5 mi in minutes        0.6658        0.0001


                    Kendall Partial Tau b Correlation Coefficients, N = 28

                                             Weight        Oxygen       Runtime

                    Weight                  1.00000      -0.09021       0.02854
                    Wt in kg

                    Oxygen                 -0.09021       1.00000      -0.52158
                    O2 use

                    Runtime                 0.02854      -0.52158       1.00000
                    1.5 mi in minutes
```

References

Blum, J.R., Kiefer, J., and Rosenblatt, M. (1961), "Distribution Free Tests of Independence Based on the Sample Distribution Function," *Annals of Mathematical Statistics*, 32, 485–498.

Conover, W.J. (1998), *Practical Nonparametric Statistics, Third Edition*, New York: John Wiley & Sons, Inc.

Cronbach, L.J. (1951), "Coefficient Alpha and the Internal Structure of Tests," *Psychometrika*, 16, 297–334.

Fisher, R.A. (1936), "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics*, 7, 179–188.

Hoeffding, W. (1948), "A Non-Parametric Test of Independence," *Annals of Mathematical Statistics*, 19, 546–557.

Hollander, M. and Wolfe, D. (1999), *Nonparametric Statistical Methods, Second Edition*, New York: John Wiley & Sons, Inc.

Knight, W.E. (1966), "A Computer Method for Calculating Kendall's Tau with Ungrouped Data," *Journal of the American Statistical Association*, 61, 436–439.

Moore, D.S. (2000), *Statistics: Concepts and Controversies, 5th Edition*, New York: W.H. Freeman & Company.

Noether, G.E. (1967), *Elements of Nonparametric Statistics*, New York: John Wiley & Sons, Inc.

Novick, M.R. (1967), "Coefficient Alpha and the Reliability of Composite Measurements," *Psychometrika*, 32, 1–13.

Nunnally, J. C. and Bernstein, I.H. (1994), *Psychometric theory, Third Edition*, New York: McGraw-Hill Companies.

Ott, R. L. and Longnecker, M.T. (2000), *An Introduction to Statistical Methods and Data Analysis, 5th Edition*, Belmont: Wadsworth Publishing Company.

SAS Institute Inc., "Measuring the Internal Consistency of a Test, Using PROC CORR to Compute Cronbach's Coefficient Alpha," *SAS Communications*, 20:4, TT2–TT5.

Spector, P.E. (1992). *Summated Rating Scale Construction: An Introduction*, Newbury Park: Sage.

CHAPTER
13

The CPORT Procedure

Overview: CPORT Procedure

The CPORT procedure writes SAS data sets, SAS catalogs, or SAS data libraries to sequential file formats (transport files). Use PROC CPORT with the CIMPORT procedure to move files from one environment to another. *Transport files* are sequential files that each contain a SAS data library, a SAS catalog, or a SAS data set in transport format. The transport format that PROC CPORT writes is the same for all environments and for many releases of SAS. In PROC CPORT, *export* means to put a SAS data library, a SAS catalog, or a SAS data set into transport format. PROC CPORT exports catalogs and data sets, either singly or as a SAS data library. PROC CIMPORT restores (*imports*) the transport file to its original form as a SAS catalog, SAS data set, or SAS data library.

Only PROC CIMPORT can read the transport files that PROC CPORT creates. For information on the transport files that the transport engine creates, see the section on SAS files in *SAS Language Reference: Concepts*.

PROC CPORT also *converts* SAS files, which means that it changes the format of a SAS file from the format appropriate for one version of SAS to the format appropriate for another version. For example, you can use PROC CPORT and PROC CIMPORT to move files from earlier releases of SAS to more recent releases. In such cases, PROC CIMPORT automatically converts the contents of the transport file as it imports it.

PROC CPORT produces no output (other than the transport files), but it does write notes to the SAS log.

To export and import files, follow these steps:

1 Use PROC CPORT to export the SAS files that you want to transport.

2 If you are changing operating environments, move the transport file to the new machine by using either communications software or a magnetic medium.

Note: If you use communications software to move the transport file, be sure that it treats the transport file as a *binary* file and that it modifies neither the attributes nor the contents of the file. △

3 Use PROC CIMPORT to translate the transport file into the format appropriate for the new operating environment or release.

Syntax: PROC CPORT

PROC CPORT *source-type=libref* | <*libref.*>*member-name*<*option(s)*>;

EXCLUDE *SAS file(s)* | *catalog entry(s)*</ MEMTYPE=*mtype*></ ENTRYTYPE=*entry-type*>;

SELECT *SAS file(s)* | *catalog entry(s)* </ MEMTYPE=*mtype*></ ENTRYTYPE=*entry-type*>;

TRANTAB NAME=*translation-table-name* <*option(s)*>;

PROC CPORT Statement

PROC CPORT *source-type=libref* | <*libref.*>*member-name*<*option(s)*>;

To do this	Use this option
Identify the transport file	
Specify the transport file to write to	FILE=
Direct the output from PROC CPORT to a tape	TAPE
Select files to export	
Export copies of all data sets or catalog entries that have a modification date equal to or later than the date you specify	AFTER=
Exclude specified entry types from the transport file	EET=
Include specified entry types in the transport file	ET=
Specify whether to export all generations of a data set	GENERATION=

To do this	Use this option
Specify that only data sets, only catalogs, or both, be moved when a library is exported	MEMTYPE=
Control the contents of the transport file	
Suppress the conversion of displayed character data to transport format	ASIS
Control the exportation of integrity constraints	CONSTRAINT
Copy the created and modified date and time to the transport file	DATECOPY
Control the exportation of indexes with indexed SAS data sets	INDEX
Suppress the compression of binary zeros and blanks in the transport file	NOCOMPRESS
Write all alphabetic characters to the transport file in uppercase	OUTTYPE= UPCASE
Translate specified characters from one ASCII or EBCDIC value to another	TRANSLATE
Export SAS/AF PROGRAM and SCL entries without edit capability when you import them	NOEDIT
Specify that exported catalog entries contain compiled SCL code, but not the source code	NOSRC
Specify a libref associated with a SAS data library	OUTLIB=

Required Arguments

source-type=libref* | < *libref.*>*member-name

identifies the type of file to export and specifies the catalog, SAS data set, or SAS data library to export.

source-type

identifies the file(s) to export as a single catalog, as a single SAS data set, or as the members of a SAS data library. The *source-type* argument can be one of the following:

CATALOG | CAT | C

DATA | DS | D

LIBRARY | LIB | L

libref | <*libref.*>*member-name*

specifies the specific catalog, SAS data set, or SAS data library to export. If *source-type* is CATALOG or DATA, you can specify both a libref and a member name. If the *libref* is omitted, PROC CPORT uses the default library as the *libref*, which is usually the WORK library. If the *source-type* argument is LIBRARY, specify only a *libref*. If you specify a library, PROC CPORT exports only data sets and catalogs from that library. You cannot export other types of files.

Options

AFTER=*date*

exports copies of all data sets or catalog entries that have a modification date later than or equal to the date you specify. The modification date is the most recent date when the contents of the data set or catalog entry changed. Specify date as a SAS date literal or as a numeric SAS date value.

Tip: You can determine the modification date of a catalog entry by using the CATALOG procedure.

Featured in: Example 5 on page 321.

ASIS

suppresses the conversion of displayed character data to transport format. Use this option when you move files that contain DBCS (double-byte character set) data from one operating environment to another if both operating environments use the same type of DBCS data.

Interaction: The ASIS option invokes the NOCOMPRESS option.

Interaction: You cannot use both the ASIS option and the OUTTYPE= options in the same PROC CPORT step.

CONSTRAINT=YES | NO

controls the exportation of integrity constraints that have been defined on a data set. When you specify CONSTRAINT=YES, all types of integrity constraints are exported for a library; only general integrity constraints are exported for a single data set. When you specify CONTRAINT=NO, indexes created without integrity constraints are ported, but neither integrity constraints nor any indexes created with integrity constraints are ported. For more information on integrity constraints, see the section on SAS files in *SAS Language Reference: Concepts*.

Alias: CON=

Default: YES

Interaction: You cannot specify both CONSTRAINT= and INDEX= in the same PROC CPORT step.

Interaction: If you specify INDEX=NO, no integrity constraints are exported.

DATECOPY

copies the SAS internal date and time when the SAS file was created and the date and time when it was last modified to the resulting transport file. Note that the operating environment date and time are not preserved.

Restriction: DATECOPY can be used only when the destination file uses the V8 or V9 engine.

Tip: You can alter the file creation date and time with the DTC= option on the MODIFY statement“MODIFY Statement” on page 366 in a PROC DATASETS step.

EET=(*etype(s)*)

excludes specified entry types from the transport file. If *etype* is a single entry type, then you can omit the parentheses. Separate multiple values with a space.

Interaction: You cannot use both the EET= option and the ET= option in the same PROC CPORT step.

ET=(*etype(s)*)

includes specified entry types in the transport file. If *etype* is a single entry type, then you can omit the parentheses. Separate multiple values with a space.

Interaction: You cannot use both the EET= option and the ET= option in the same PROC CPORT step.

FILE=*fileref* | *'filename'*
specifies a previously defined fileref or the filename of the transport file to write to. If you omit the FILE= option, then PROC CPORT writes to the fileref SASCAT, if defined. If the fileref SASCAT is not defined, PROC CPORT writes to SASCAT.DAT in the current directory.

Note: The behavior of PROC CPORT when SASCAT is undefined varies from one operating environment to another. For details, see the SAS documentation for your operating environment. △

Featured in: All examples.

GENERATION=YES | NO
specifies whether to export all generations of a SAS data set. To export only the base generation of a data set, specify GENERATION=NO in the PROC CPORT statement. To export a specific generation number, use the GENNUM= data set option when you specify a data set in the PROC CPORT statement. For more information on generation data sets, see *SAS Language Reference: Concepts*.

Note: PROC CIMPORT imports all generations of a data set that are present in the transport file. It deletes any previous generation set with the same name and replaces it with the imported generation set, even if the number of generations does not match. △

Alias: GEN=

Default: YES for libraries; NO for single data sets

INDEX=YES | NO
specifies whether to export indexes with indexed SAS data sets.

Default: YES

Interaction: You cannot specify both INDEX= and CONSTRAINT= in the same PROC CPORT step.

Interaction: If you specify INDEX=NO, no integrity constraints are exported.

INTYPE=*DBCS-type*
specifies the type of DBCS data stored in the SAS files to be exported. Double-byte character set (DBCS) data uses up to two bytes for each character in the set. *DBCS-type* must be one of the following values:

IBM | HITAC | FACOM	for OS/390
IBM	for VSE
DEC | SJIS	for OpenVMS
PCIBM | SJIS	for OS/2

Restriction The INTYPE= option is allowed only if SAS is built with Double-Byte Character Set (DBCS) extensions. Because these extensions require significant computing resources, there is a special distribution for those sites that require it. An error is reported if this option is used at a site for which DBCS extensions are not enabled.

Default: If the INTYPE= option is not used, the DBCS type defaults to the value of the SAS system option DBCSTYPE=.

Interaction: Use the INTYPE= option in conjunction with the OUTTYPE= option to change from one type of DBCS data to another.

Interaction: The INTYPE= option invokes the NOCOMRPESS option.

Interaction: You cannot use the INTYPE= option and the ASIS option in the same PROC CPORT step.

Tip: You can set the value of the SAS system option DBCSTYPE= in your configuration file.

MEMTYPE=*mtype*
restricts the type of SAS file that PROC CPORT writes to the transport file. MEMTYPE= restricts processing to one member type. Values for *mtype* can be

ALL
both catalogs and data sets

CATALOG | CAT
catalogs

DATA | DS
SAS data sets

Alias: MT=

Default: ALL

Featured in: Example 1 on page 317.

NOCOMPRESS
suppresses the compression of binary zeros and blanks in the transport file.

Alias: NOCOMP

Default: By default, PROC CPORT compresses binary zeros and blanks to conserve space.

Interaction: The ASIS, INTYPE=, and OUTTYPE= options invoke the NOCOMPRESS option.

Note: Compression of the transport file does not alter the flag in each catalog and data set that indicates whether the original file was compressed. △

NOEDIT
exports SAS/AF PROGRAM and SCL entries without edit capability when you import them.

The NOEDIT option produces the same results as when you create a new catalog to contain SCL code by using the MERGE statement with the NOEDIT option in the BUILD procedure of SAS/AF software.

Note: The NOEDIT option affects only SAS/AF PROGRAM and SCL entries. It does not affect FSEDIT SCREEN or FSVIEW FORMULA entries. △

Alias: NEDIT

NOSRC
specifies that exported catalog entries contain compiled SCL code but not the source code.

The NOSRC option produces the same results as when you create a new catalog to contain SCL code by using the MERGE statement with the NOSOURCE option in the BUILD procedure of SAS/AF software.

Alias: NSRC

OUTLIB=*libref*
specifies a libref associated with a SAS data library. If you specify the OUTLIB= option, PROC CIMPORT is invoked automatically to re-create the input data library, data set, or catalog in the specified library.

Alias: OUT=

Tip: Use the OUTLIB= option when you change SAS files from one DBCS type to another within the same operating environment if you want to keep the original data intact.

OUTTYPE=UPCASE
writes all displayed characters to the transport file and to the OUTLIB= file in uppercase.

Interaction: The OUTTYPE= option invokes the NOCOMPRESS option.

TAPE
directs the output from PROC CPORT to a tape.

Default: The output from PROC CPORT is sent to disk.

TRANSLATE=(*translation-list*)
translates specified characters from one ASCII or EBCDIC value to another. Each element of *translation-list* has the form

ASCII-value-1 TO *ASCII-value-2*

EBCDIC-value-1 TO *EBCDIC-value-2*

You can use hexadecimal or decimal representation for ASCII values. If you use the hexadecimal representation, values must begin with a digit and end with an x. Use a leading zero if the hexadecimal value begins with an alphabetic character.

For example, to translate all left brackets to left braces, specify the TRANSLATE= option as follows (for ASCII characters):

```
translate=(5bx to 7bx)
```

The following example translates all left brackets to left braces and all right brackets to right braces:

```
translate=(5bx to 7bx 5dx to 7dx)
```

EXCLUDE Statement

Excludes specified files or entries from the transport file.

Tip: There is no limit to the number of EXCLUDE statements you can use in one invocation of PROC CPORT.

Interaction: You can use either EXCLUDE statements or SELECT statements in a PROC CPORT step, but not both.

EXCLUDE *SAS file(s)* | *catalog entry(s)*</ MEMTYPE=*mtype*></ ENTRYTYPE=*entry-type*>;

Required Arguments

SAS file(s) | catalog entry(s)
specifies either the name(s) of one or more SAS files or the names of one or more catalog entries to be excluded from the transport file. Specify SAS filenames when you export a SAS data library; specify catalog entry names when you export an individual SAS catalog. Separate multiple filenames or entry names with a space. You can use shortcuts to list many like-named files in the EXCLUDE statement. For more information, see “Shortcuts for Specifying Lists of Variable Names” on page 24.

Options

ENTRYTYPE=*entry-type*
specifies a single entry type for the catalog entries listed in the EXCLUDE statement. See *SAS Language Reference: Concepts* for a complete list of catalog entry types.

Restriction: ENTRYTYPE= is valid only when you export an individual SAS catalog.

Alias: ETYPE=, ET=

MEMTYPE=*mtype*
specifies a single member type for the SAS file(s) listed in the EXCLUDE statement. Valid values are CATALOG or CAT, DATA, or ALL. If you do not specify the MEMTYPE= option in the EXCLUDE statement, then processing is restricted to those member types specified in the MEMTYPE= option in the PROC CPORT statement.

You can also specify the MEMTYPE= option, enclosed in parentheses, immediately after the name of a file. In parentheses, MEMTYPE= identifies the type of the file name that just precedes it. When you use this form of the option, it overrides the MEMTYPE= option that follows the slash in the EXCLUDE statement, but it must match the MEMTYPE= option in the PROC CPORT statement:

Restriction: MEMTYPE= is valid only when you export a SAS data library.

Restriction: If you specify a member type for MEMTYPE= in the PROC CPORT statement, it must agree with the member type that you specify for MEMTYPE= in the EXCLUDE statement.

Alias: MTYPE=, MT=

Default: If you do not specify MEMTYPE= in the PROC CPORT statement or in the EXCLUDE statement, the default is MEMTYPE=ALL.

SELECT Statement

Includes specified files or entries in the transport file.

Tip: There is no limit to the number of SELECT statements you can use in one invocation of PROC CPORT.

Interaction: You can use either EXCLUDE statements or SELECT statements in a PROC CPORT step, but not both.

Featured in: Example 2 on page 318

SELECT *SAS file(s)* | *catalog entry(s)*</ MEMTYPE=*mtype*> </ ENTRYTYPE=*entry-type*> ;

Required Arguments

SAS file(s) | catalog entry(s)
specifies either the name(s) of one or more SAS files or the names of one or more catalog entries to be included in the transport file. Specify SAS filenames when you export a SAS data library; specify catalog entry names when you export an individual SAS catalog. Separate multiple filenames or entry names with a space. You can use shortcuts to list many like-named files in the SELECT statement. For more information, see "Shortcuts for Specifying Lists of Variable Names" on page 24.

Options

ENTRYTYPE=*entry-type*
specifies a single entry type for the catalog entries listed in the SELECT statement. See *SAS Language Reference: Concepts* for a complete list of catalog entry types.

Restriction: ENTRYTYPE= is valid only when you export an individual SAS catalog.

Alias: ETYPE=, ET=

MEMTYPE=*mtype*
specifies a single member type for the SAS file(s) listed in the SELECT statement. Valid values are CATALOG or CAT, DATA, or ALL. If you do not specify the MEMTYPE= option in the SELECT statement, then processing is restricted to those member types specified in the MEMTYPE= option in the PROC CPORT statement.

You can also specify the MEMTYPE= option, enclosed in parentheses, immediately after the name of a member. In parentheses, MEMTYPE= identifies the type of the member name that just precedes it. When you use this form of the option, it overrides the MEMTYPE= option that follows the slash in the SELECT statement, but it must match the MEMTYPE= option in the PROC CPORT statement.

Restriction: MEMTYPE= is valid only when you export a SAS data library.

Restriction: If you specify a member type for MEMTYPE= in the PROC CPORT statement, it must agree with the member type that you specify for MEMTYPE= in the SELECT statement.

Alias: MTYPE=, MT=

Default: If you do not specify MEMTYPE= in the PROC CPORT statement or in the SELECT statement, the default is MEMTYPE=ALL.

TRANTAB Statement

Specifies translation tables for characters in catalog entries you export.

Tip: You can specify only one table for each TRANTAB statement, but there is no limit to the number of TRANTAB statements you can use in one invocation of PROC CPORT.

Featured in: Example 4 on page 320.

See also: Chapter 47, "The TRANTAB Procedure," on page 1409

TRANTAB NAME=*translation-table-name*
<*option(s)*>;

Required Arguments

NAME=*translation-table-name*
specifies the name of the translation table to apply to the character data in the SAS file you export. The *translation-table-name* is the name of a catalog entry in either the SASUSER.PROFILE catalog or the SASHELP.HOST catalog. PROC CPORT prints an error message in the SAS log if it cannot find the translation table.

Note: The translation takes place before PROC CPORT writes to the transport file. △

Options

OPT=
specifies how to apply the translation table. Use one of the following values for the OPT= option:

DISP
applies the translation table to all the DISPLAY window text.

SRC
applies the translation table to all the SCL text.

(DISP SRC)
applies the translation table to all the DISPLAY window text and SOURCE window text.

Default: PROC CPORT applies all options to the specified translation table.

TYPE=(*target-list*)
applies the translation table only to the specified targets. If the *target-list* is a single target, then you can omit the parentheses. The *target-list* can be one of the following types:

etype-list
applies the translation table only to the entries with the catalog entry type you specify.

CATDESC
applies the translation table to the description of each exported catalog entry.

DATASET
applies the translation table to the observations, the data set label, and the variable labels in each exported data set.

Default: PROC CPORT applies the translation table to all entries and data sets in the specified catalog.

Featured in: Example 4 on page 320.

Concepts: CPORT Procedure

Transporting Password-Protected Data Sets

For password-protected data sets, the password(s) are applied to the destination data set when it is imported. If the data set is transported as part of a library, it is not necessary to supply the password. If the data set is transported singly, you must supply the read password. If you omit the password in the PROC CPORT step, SAS prompts you for the password. If the target SAS engine does not support passwords, then the import will fail. For example, the following SAS code transports a password-protected data set called WORK.ONE:

```
proc cport data=one(read=hithere) file='bin';
```

Results: CPORT Procedure

Data Control Block Characteristics for Mainframe Environments

A common problem when you create or import a transport file under the OS/390 environment is a failure to specify the correct Data Control Block (DCB) characteristics. When you reference a transport file, you must specify the following DCB characteristics:

LRECL	80
BLKSIZE	8000
RECFM	FB

Note: A BLKSIZE value of less than 8000 may be more efficient for your storage device in some cases. The BLKSIZE value should be an exact multiple of the LRECL value. △

Another common problem can occur if you use communications software to move files from another environment to OS/390. In some cases, the transport file does not have the proper DCB characteristics when it arrives on OS/390. If the communications software does not allow you to specify file characteristics, try the following approach for OS/390:

1 Create a file under OS/390 with the correct DCB characteristics and initialize the file.

2 Move the transport file from the other environment to the newly created file under OS/390 using binary transfer.

Examples: CPORT Procedure

Example 1: Exporting Multiple Catalogs

Procedure features:

PROC CPORT statement options:

FILE=
MEMTYPE=

This example shows how to use PROC CPORT to export entries from all of the SAS catalogs in the SAS data library you specify.

Program

Specify the library reference for the SAS data library that contains the source files to be exported and the file reference to which the output transport file is written. The LIBNAME statement assigns a libref for the SAS data library. The FILENAME statement assigns a fileref and any operating environment options for file characteristics for the transport file that PROC CPORT creates.

```
libname source 'SAS-data-library';
filename tranfile 'transport-file'
                  host-option(s)-for-file-characteristics;
```

Create the transport file. The PROC CPORT step executes on the operating environment where the source library is located. MEMTYPE=CATALOG writes all SAS catalogs in the source library to the transport file.

```
proc cport library=source file=tranfile memtype=catalog;
run;
```

SAS Log

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FINANCE
NOTE: The catalog has 5 entries and its maximum logical record length is 866.
NOTE: Entry LOAN.FRAME has been transported.
NOTE: Entry LOAN.HELP has been transported.
NOTE: Entry LOAN.KEYS has been transported.
NOTE: Entry LOAN.PMENU has been transported.
NOTE: Entry LOAN.SCL has been transported.

NOTE: Proc CPORT begins to transport catalog SOURCE.FORMATS
NOTE: The catalog has 2 entries and its maximum logical record length is 104.
NOTE: Entry REVENUE.FORMAT has been transported.
NOTE: Entry DEPT.FORMATC has been transported.
```

Example 2: Exporting Individual Catalog Entries

Procedure features:

PROC CPORT statement options:

FILE=

SELECT statement

This example shows how to use PROC CPORT to export individual catalog entries, rather than all of the entries in a catalog.

Program

Assign library references. The LIBNAME and FILENAME statements assign a libref for the source library and a fileref for the transport file, respectively.

```
libname source 'SAS-data-library';
filename tranfile 'transport-file'
                   host-option(s)-for-file-characteristics;
```

Write an entry to the transport file. SELECT writes only the LOAN.SCL entry to the transport file for export.

```
proc cport catalog=source.finance file=tranfile;
select loan.scl;
run;
```

SAS Log

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FINANCE
NOTE: The catalog has 5 entries and its maximum logical record length is 866.
NOTE: Entry LOAN.SCL has been transported.
```

Example 3: Exporting a Single SAS Data Set

Procedure features:

PROC CPORT statement option:

FILE=

This example shows how to use PROC CPORT to export a single SAS data set.

Program

Assign library references. The LIBNAME and FILENAME statements assign a libref for the source library and a fileref for the transport file, respectively.

```
libname source 'SAS-data-library';
filename tranfile 'transport-file'
                   host-option(s)-for-file-characteristics;
```

Specify the type of file that you are exporting. The DATA= specification in the PROC CPORT statement tells the procedure that you are exporting a SAS data set rather than a library or a catalog.

```
proc cport data=source.times file=tranfile;
run;
```

SAS Log

```
NOTE: Proc CPORT begins to transport data set SOURCE.TIMES
NOTE: The data set contains 2 variables and 2 observations.
      Logical record length is 16.
NOTE: Transporting data set index information.
```

Example 4: Applying a Translation Table

Procedure features:

PROC CPORT statement option:

FILE=

TRANTAB statement option:

TYPE=

This example shows how to apply a customized translation table to the transport file before PROC CPORT exports it. For this example, assume that you have already created a customized translation table called TTABLE1.

Program

Assign library references. The LIBNAME and FILENAME statements assign a libref for the source library and a fileref for the transport file, respectively.

```
libname source 'SAS-data-library';
filename tranfile 'transport-file'
                   host-option(s)-for-file-characteristics;
```

Apply the translation specifics. The TRANTAB statement applies the translation that you specify with the customized translation table TTABLE1. TYPE= limits the translation to FORMAT entries.

```
proc cport catalog=source.formats file=tranfile;
   trantab name=ttable1 type=(format);
run;
```

SAS Log

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FORMATS
NOTE: The catalog has 2 entries and its maximum logical record length is 104.
NOTE: Entry REVENUE.FORMAT has been transported.
NOTE: Entry DEPT.FORMATC has been transported.
```

Example 5: Exporting Entries Based on Modification Date

Procedure features:

PROC CPORT statement options:

AFTER=
FILE=

This example shows how to use PROC CPORT to transport only the catalog entries with modification dates equal to or later than the date you specify in the AFTER= option.

Program

Assign library references. The LIBNAME and FILENAME statements assign a libref for the source library and a fileref for the transport file, respectively.

```
libname source 'SAS-data-library';
filename tranfile 'transport-file'
                   host-option(s)-for-file-characteristics;
```

Specify the catalog entries to be written to the transport file. AFTER= specifies that only catalog entries with modification dates on or after September 9, 1996, should be written to the transport file.

```
proc cport catalog=source.finance file=tranfile
           after='09sep1996'd;
run;
```

SAS Log

PROC CPORT writes messages to the SAS log to inform you that it began the export process for all the entries in the specified catalog. However, PROC CPORT wrote only the entries LOAN.FRAME and LOAN.HELP in the FINANCE catalog to the transport file because only those two entries had a modification date equal to or later than September 9, 1996. That is, of all the entries in the specified catalog, only two met the requirement of the AFTER= option.

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FINANCE
NOTE: The catalog has 5 entries and its maximum logical record length is 866.
NOTE: Entry LOAN.FRAME has been transported.
NOTE: Entry LOAN.HELP has been transported.
```

CHAPTER 14

The CV2VIEW Procedure

Information about the CV2VIEW Procedure

See: For complete documentation of the CV2VIEW procedure, see *SAS/ACCESS for Relational Databases: Reference*.

CHAPTER
15

The DATASETS Procedure

Overview: DATASETS Procedure

The DATASETS procedure is a utility procedure that manages your SAS files. With PROC DATASETS, you can

- copy SAS files from one SAS library to another
- rename SAS files
- repair SAS files
- delete SAS files
- list the SAS files that are contained in a SAS library
- list the attributes of a SAS data set, such as the date when the data was last modified, whether the data is compressed, whether the data is indexed, and so on
- manipulate passwords on SAS files
- append SAS data sets
- modify attributes of SAS data sets and variables within the data sets
- create and delete indexes on SAS data sets
- create and manage audit files for SAS data sets
- create and delete integrity constraints on SAS data sets.

For example, the following DATASETS procedure

1. copies all data sets from the CONTROL library to the HEALTH library
2. lists the contents of the HEALTH library
3. deletes the SYNDROME data set from the HEALTH library
4. changes the name of the PRENAT data set to INFANT.

The SAS log is shown in Output 15.1 on page 327.

```
libname control 'SAS-data-library-1';
libname health 'SAS-data-library-2';
```

```
proc datasets memtype=data;
   copy in=control out=health;
run;

proc datasets library=health memtype=data details;
   delete syndrome;
   change prenat=infant;
run;
quit;
```

Output 15.1 Log from PROC DATASETS

```
59   proc datasets library=health memtype=data details;
                                                     Directory

                                 Libref         HEALTH
                                 Engine         V9
                                 Physical Name  external-file
                                 File Name      external-file


                         Member  Obs, Entries                                  File
                 #  Name      Type     or Indexes   Vars  Label                       Size  Last Modified

                 1  ALL       DATA         23        17                          13312   29JAN2002:08:06:46
                 2  BODYFAT   DATA          1         2                           5120   29JAN2002:08:06:46
                 3  CONFOUND  DATA          8         4                           5120   29JAN2002:08:06:46
                 4  CORONARY  DATA         39         4                           5120   29JAN2002:08:06:46
                 5  DRUG1     DATA          6         2  JAN95 Data               5120   29JAN2002:08:06:46
                 6  DRUG2     DATA         13         2  MAY95 Data               5120   29JAN2002:08:06:46
                 7  DRUG3     DATA         11         2  JUL95 Data               5120   29JAN2002:08:06:46
                 8  DRUG4     DATA          7         2  JAN92 Data               5120   29JAN2002:08:06:46
                 9  DRUG5     DATA          1         2  JUL92 Data               5120   29JAN2002:08:06:46
                10  GROUP     DATA        148        11                          25600   29JAN2002:08:06:46
                11  MLSCL     DATA         32         4  Multiple Sclerosis Data  5120   29JAN2002:08:06:46
                12  NAMES     DATA          7         4                           5120   29JAN2002:08:06:46
                13  OXYGEN    DATA         31         7                           9216   29JAN2002:08:06:46
                14  PERSONL   DATA        148        11                          25600   29JAN2002:08:06:46
                15  PHARM     DATA          6         3  Sugar Study              5120   29JAN2002:08:06:46
                16  POINTS    DATA          6         6                           5120   29JAN2002:08:06:46
                17  PRENAT    DATA        149         6                          17408   29JAN2002:08:06:46
                18  RESULTS   DATA         10         5                           5120   29JAN2002:08:06:46
                19  SLEEP     DATA        108         6                           9216   29JAN2002:08:06:46
                20  SYNDROME  DATA         46         8                           9216   29JAN2002:08:06:46
                21  TENSION   DATA          4         3                           5120   29JAN2002:08:06:46
                22  TEST2     DATA         15         5                           5120   29JAN2002:08:06:46
                23  TRAIN     DATA          7         2                           5120   29JAN2002:08:06:47
                24  VISION    DATA         16         3                           5120   29JAN2002:08:06:47
                25  WEIGHT    DATA         83        13  California Results      13312   29JAN2002:08:06:47
                26  WGHT      DATA         83        13  California Results      13312   29JAN2002:08:06:47
60      delete syndrome;
61      change prenat=infant;
62   run;
NOTE: Deleting HEALTH.SYNDROME (memtype=DATA).
NOTE: Changing the name HEALTH.PRENAT to HEALTH.INFANT (memtype=DATA).
63   quit;
```

Notes

- Although the DATASETS procedure can perform some operations on catalogs, generally the CATALOG procedure is the best utility to use for managing catalogs. For documentation of PROC CATALOG, see "Overview: CATALOG Procedure" on page 143.
- The term *member* often appears as a synonym for *SAS file*. If you are unfamiliar with SAS files and SAS libraries, see "SAS Files Concepts" in *SAS Language Reference: Concepts*.
- PROC DATASETS cannot work with sequential data libraries.

Syntax: PROC DATASETS

Tip: Supports RUN-group processing.

Tip: Supports the Output Delivery System. See “Output Delivery System” on page 32 for details.

Reminder: See Chapter 3, “Statements with the Same Function in Multiple Procedures,” on page 53 for details. You can also use any global statements as well. See “Global Statements” on page 18 for a list.

PROC DATASETS <*option(s)*>;

AGE *current-name related-SAS-file(s)*
</ <ALTER=*alter-password*>
<MEMTYPE=*mtype*>>;

APPEND BASE=<*libref.*>*SAS-data-set*
<APPENDVER=V6>
<DATA=<*libref.*>*SAS-data-set*>
<FORCE>;

AUDIT *SAS-file* <(*SAS-password*)>;
INITIATE;
<LOG < BEFORE_IMAGE=YES | NO>
<DATA_IMAGE=YES | NO>
<ERROR_IMAGE=YES | NO>>;
<USER_VAR *variable-1* <... *variable-n*>>;

AUDIT *SAS-file* <(<*SAS-password*> <GENNUM= *integer*>)>;
SUSPEND | RESUME | TERMINATE;

CHANGE *old-name-1=new-name-1*
<*...old-name-n=new-name-n* >
</ <ALTER=*alter-password*>
<GENNUM=ALL | *integer*>
<MEMTYPE=*mtype*>>;

CONTENTS<*option(s)*>;

COPY OUT=*libref-1*
<CLONE | NOCLONE>
<CONSTRAINT=YES | NO>
<DATECOPY>
<FORCE>
<IN=*libref-2*>
<INDEX=YES | NO>
<MEMTYPE=(*mtype(s)*)>
<MOVE <ALTER=*alter-password*>>;
EXCLUDE *SAS-file(s)* < / MEMTYPE=*mtype*>;
SELECT *SAS-file(s)*
</ <ALTER=*alter-password*>
<MEMTYPE= *mtype*>>;

DELETE *SAS-file(s)*
</ <ALTER=*alter-password*>
<GENNUM=ALL | HIST | REVERT | *integer*>
<MEMTYPE=*mtype*>>;

EXCHANGE *name-1=other-name-1*
<*...name-n=other-name-n*>

</ <ALTER=*alter-password*>
<MEMTYPE=*mtype*> >;
MODIFY *SAS-file* <(*option(s)*)>
</ <DTC=*SAS-date-time*>
<GENNUM=*integer*>
<MEMTYPE=*mtype*>>;
FORMAT *variable-list-1* <*format-1*>
<*…variable-list-n* <*format-n*>>;
IC CREATE <*constraint-name*=> *constraint*
<MESSAGE='*message-string*' <MSGTYPE=USER>>;
IC DELETE *constraint-name(s)* | _ALL_;
IC REACTIVATE *foreign-key-name* REFERENCES *libref*;
INDEX CENTILES *index(s)*
</ <REFRESH>
<UPDATECENTILES= ALWAYS|NEVER|*integer*>>;
INDEX CREATE *index-specification(s)*
</ <NOMISS>
<UNIQUE>
<UPDATECENTILES=ALWAYS|NEVER|*integer*>>;
INDEX DELETE *index(s)* | _ALL_;
INFORMAT *variable-list-1* <*informat-1*>
<*…variable-list-n* <*informat-n*>>;
LABEL *variable-1*=<*'label-1'*|' '>
<*…variable-n*=<*'label-n'*|' ' >>;
RENAME *old-name-1*=*new-name-1*
<*…old-name-n*=*new-name-n*>;
REPAIR *SAS-file(s)*
</ <ALTER=*alter-password*>
<GENNUM=*integer*>
<MEMTYPE=*mtype*>>;
SAVE *SAS-file(s)* </ MEMTYPE=*mtype*>;

PROC DATASETS Statement

PROC DATASETS <*option(s)*>;

To do this	Use this option
Specify the procedure input library	LIBRARY=
Provide alter access to any alter-protected SAS file in the SAS data library	ALTER=
Include information in the log about the number of observations, number of variables, number of indexes, and data set labels	DETAILS|NODETAILS
Force a RUN group to execute even when there are errors	FORCE

To do this	Use this option
Force an append operation	FORCE
Restrict processing for generation data sets	GENNUM=
Delete SAS files	KILL
Restrict processing to a certain type of SAS file	MEMTYPE=
Suppress the printing of the directory	NOLIST
Suppress error processing	NOWARN
Provide read, write, or alter access	PW=
Provide read access	READ=

Options

ALTER=*alter-password*
provides the alter password for any alter-protected SAS files in the SAS data library.

See also: "Using Passwords with the DATASETS Procedure" on page 377

DETAILS|NODETAILS
determines whether the following columns are written to the log:

Obs, Entries, or Indexes
gives the number of observations for SAS files of type AUDIT, DATA, and VIEW; the number of entries for type CATALOG; and the number of files of type INDEX that are associated with a data file, if any. If SAS cannot determine the number of observations in a SAS data set, the value in this column is set to missing. For example, in a very large data set, if the number of observations or deleted observations exceeds the number that can be stored in a double-precision integer, the count will show as missing. The value for type CATALOG is the total number of entries. For other types, this column is blank.

Tip: The value for files of type INDEX includes both user-defined indexes and indexes created by integrity constraints. To view index ownership and attribute information, use PROC DATASETS with the CONTENTS statement and the OUT2 option.

Vars
gives the number of variables for types AUDIT, DATA and VIEW. If SAS cannot determine the number of variables in the SAS data set, the value in this column is set to missing. For other types, this column is blank.

Label
contains the label associated with the SAS data set. This column prints a label only for the type DATA.

The DETAILS option affects output only when a directory is specified and requires read access to all read-protected SAS files in the SAS data library. If you do not supply the read password, the directory listing contains missing values for the columns produced by the DETAILS option.

Default: If neither DETAILS or NODETAILS is specified, the default is the system option setting. The default system option setting is NODETAILS.

Tip: If you are using the SAS windowing environment and specify the DETAILS option for a library that contains read-protected SAS files, a requestor window prompts you for each read password that you do not specify in the PROC DATASETS statement. Therefore, you may want to assign the same read password to all SAS files in the same SAS data library.

Featured in: Example 1 on page 392

FORCE

performs two separate actions:

- forces a RUN group to execute even if errors are present in one or more statements in the RUN group. See "RUN-Group Processing" on page 375 for a discussion of RUN-group processing and error handling.
- forces all APPEND statements to concatenate two data sets even when the variables in the data sets are not exactly the same. The APPEND statement drops the extra variables and issues a warning message. Without the FORCE option, the procedure issues an error message and stops processing if you try to perform an append operation with two SAS data sets whose variables are not exactly the same. Refer to "APPEND Statement" on page 335 for more information on the FORCE option.

GENNUM=ALL | HIST | REVERT | *integer*

restricts processing for generation data sets. Valid values are as follows:

ALL

for subordinate CHANGE and DELETE statements, refers to the base version and all historical versions in a generation group.

HIST

for a subordinate DELETE statement, refers to all historical versions, but excludes the base version in a generation group.

REVERT | 0

for a subordinate DELETE statement, refers to the base version in a generation group and changes the most current historical version, if it exists, to the base version.

integer

for subordinate AUDIT, CHANGE, MODIFY, DELETE, and REPAIR statements, refers to a specific version in a generation group. Specifying a positive number is an absolute reference to a specific generation number that is appended to a data set name; that is, `gennum=2` specifies MYDATA#002. Specifying a negative number is a relative reference to a historical version in relation to the base version, from the youngest to the oldest; that is, `gennum=-1` refers to the youngest historical version.

See also: "Restricting Processing for Generation Data Sets" on page 380

See also: "Understanding Generation Data Sets" in *SAS Language Reference: Concepts*

KILL

deletes *all* SAS files in the SAS data library that are available for processing. The MEMTYPE= option subsets the member types that the statement deletes.

CAUTION:

The KILL option deletes the SAS files immediately after you submit the statement. △

LIBRARY=*libref*

names the library that the procedure processes. This library is the *procedure input library*.

Aliases: DDNAME=, DD=, LIB=

Default: WORK or USER. See "Temporary and Permanent SAS Data Sets" on page 16 for more information on the WORK and USER libraries.

Restriction: A SAS library that is accessed via a sequential engine (such as a tape format engine) cannot be specified as the value of the LIBRARY= option.

Featured in: Example 1 on page 392

MEMTYPE=(*mtype(s)*)
restricts processing to one or more member types and restricts the listing of the data library directory to SAS files of the specified member types. For example, the following PROC DATASETS statement limits processing to SAS data sets in the default data library and limits the directory listing in the SAS log to SAS files of member type DATA:

```
proc datasets memtype=data;
```

Aliases: MTYPE=, MT=

Default: ALL

See also: "Restricting Member Types for Processing" on page 378

NODETAILS
See the description of DETAILS on page 331.

NOLIST
suppresses the printing of the directory of the SAS files in the SAS log.

Featured in: Example 3 on page 398

Note: If you specify the ODS RTF destination, PROC DATASETS output will go to both the SAS log and the ODS output area. The NOLIST option will suppress output to both. To see the output only in the SAS log, use the ODS EXCLUDE statement by specifying the member directory as the exclusion. △

NOWARN
suppresses the error processing that occurs when a SAS file that is specified in a SAVE, CHANGE, EXCHANGE, REPAIR, DELETE, or COPY statement or listed as the first SAS file in an AGE statement is not in the procedure input library. When an error occurs and the NOWARN option is in effect, PROC DATASETS continues processing that RUN group. If NOWARN is not in effect, PROC DATASETS stops processing that RUN group and issues a warning for all operations except DELETE, for which it does not stop processing.

PW= *password*
provides the password for any protected SAS files in the SAS data library. PW= can act as an alias for READ=, WRITE=, or ALTER=.

See also: "Using Passwords with the DATASETS Procedure" on page 377

READ=*read-password*
provides the read-password for any read-protected SAS files in the SAS data library.

See also: "Using Passwords with the DATASETS Procedure" on page 377

AGE Statement

Renames a group of related SAS files in a library.

Featured in: Example 6 on page 405

AGE *current-name related-SAS-file(s)*

</ <ALTER=*alter-password*>
<MEMTYPE=*mtype*>>;

Required Arguments

current-name
is a SAS file that the procedure renames. *current-name* receives the name of the first name in *related-SAS-file(s)*.

related-SAS-file(s)
is one or more SAS files in the SAS data library.

Options

ALTER=*alter-password*
provides the alter password for any alter-protected SAS files named in the AGE statement. Because an AGE statement renames and deletes SAS files, you need alter access to use the AGE statement. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

See also: "Using Passwords with the DATASETS Procedure" on page 377

MEMTYPE=*mtype*
restricts processing to one member type. All of the SAS files that you name in the AGE statement must be the same member type. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

Aliases: MTYPE=, MT=

Default: If you do not specify MEMTYPE= in the PROC DATASETS statement, the default is DATA.

See also: "Restricting Member Types for Processing" on page 378

Details

- The AGE statement renames *current-name* to the name of the first name in *related-SAS-file(s)*, renames the first name in *related-SAS-file(s)* to the second name in *related-SAS-file(s)*, and so on until it changes the name of the next-to-last SAS file in *related-SAS-file(s)* to the last name in *related-SAS-file(s)*. The AGE statement then deletes the last file in *related-SAS-file(s)*.
- If the first SAS file named in the AGE statement does not exist in the SAS data library, PROC DATASETS stops processing the RUN group containing the AGE statement and issues an error message. The AGE statement does not age any of the *related-SAS-file(s)*. To override this behavior, use the NOWARN option in the PROC DATASETS statement.

  If one of the *related-SAS-file(s)* does not exist, the procedure prints a warning message to the SAS log but continues to age the SAS files that it can.
- If you age a data set that has an index, the index continues to correspond to the data set.
- You can age only entire generation groups. For example, if data sets A and B have generation groups, then the following statement deletes generation group B and ages (renames) generation group A to the name B:

```
age a b;
```

For example, suppose the generation group for data set A has 3 historical versions and the generation group for data set B has 2 historical versions. Then aging A to B has this effect:

Old Name	Version	New Name	Version
A	base	B	base
A	1	B	1
A	2	B	2
A	3	B	3
B	base	is deleted	
B	1	is deleted	
B	2	is deleted	

APPEND Statement

Adds the observations from one SAS data set to the end of another SAS data set.

Reminder: You can use data set options with the BASE= and DATA= options. See “Data Set Options” on page 17 for a list. You can also use any global statements as well. See “Global Statements” on page 18.

Requirement: The BASE= data set must be a member of a SAS library that supports update processing.

Default: If the BASE= data set is accessed through a SAS server and if no other user has the data set open at the time the APPEND statement begins processing, the BASE= data set defaults to CNTLLEV=MEMBER (member-level locking). When this happens, no other user can update the file while the data set is processed.

Tip: If a failure occurs during processing, the data set is marked as damaged and is reset to its pre-append condition at the next REPAIR statement. If the data set has an index, the index is not updated with each observation but is updated once at the end. (This is Version 7 and later behavior, as long as APPENDVER=V6 is not set.)

Featured in: Example 5 on page 403

APPEND BASE=<*libref.*>*SAS-data-set*
<APPENDVER=V6>
<DATA=<*libref.*>*SAS-data-set*>
<FORCE>;

Required Arguments

BASE=<*libref.*> *SAS-data-set*
names the data set to which you want to add observations.

libref

specifies the library that contains the SAS data set. If you omit the *libref*, the default is the libref for the procedure input library. If you are using PROC APPEND, the default for *libref* is either WORK or USER.

SAS-data-set

names a SAS data set. If the APPEND statement cannot find an existing data set with this name, it creates a new data set in the library. That is, you can use the APPEND statement to create a data set by specifying a new data set name in the BASE= argument.

The BASE= data set is the current SAS data set after all append operations regardless of whether you are creating a new data set or appending to an existing data set.

Alias: OUT=

Featured in: Example 5 on page 403

Options

APPENDVER=V6

uses the Version 6 behavior for appending observations to the BASE= data set, which is to append one observation at a time. Beginning in Version 7, to improve performance, the default behavior changed so that all observations are appended after the data set is processed.

See also: "Appending to an Indexed Data Set" on page 338

DATA=<*libref.*> *SAS-data-set*

names the SAS data set containing observations that you want to append to the end of the SAS data set specified in the BASE= argument.

libref

specifies the library that contains the SAS data set. If you omit *libref*, the default is the libref for the procedure input library. The DATA= data set can be from any SAS data library, but you must use the two-level name if the data set resides in a library other than the procedure input library.

SAS-data-set

names a SAS data set. If the APPEND statement cannot find an existing data set with this name, it stops processing.

Alias: NEW=

Default: the most recently created SAS data set, from any SAS data library

See also: "Appending with Generation Groups" on page 339

Featured in: Example 5 on page 403

FORCE

forces the APPEND statement to concatenate data sets when the DATA= data set contains variables that either

- □ are not in the BASE= data set
- □ do not have the same type as the variables in the BASE= data set
- □ are longer than the variables in the BASE= data set.

See also: "Appending to Data Sets with Different Variables" on page 339

See also: "Appending to Data Sets That Contain Variables with Different Attributes" on page 339

Featured in: Example 5 on page 403

Tip: You can use the GENNUM= data set option to append to or from a specific version in a generation group. Here are some examples:

```
/* appends historical version to base A */
proc datasets;
   append base=a
      data=a (gennum=2);

/* appends current version of A to historical version */
proc datasets;
   append base=a (gennum=1)
      data=a;
```

Restricting the Observations That Are Appended

You can use the WHERE= data set option with the DATA= data set in order to restrict the observations that are appended. Likewise, you can use the WHERE statement in order to restrict the observations from the DATA= data set. The WHERE statement has no effect on the BASE= data set. If you use the WHERE= data set option with the BASE= data set, WHERE= has no effect.

CAUTION:
For an existing BASE= data set: If there is a WHERE statement on the BASE= data set, it will take effect only if the WHEREUP= option is set to YES. △

CAUTION:
For the non-existent BASE= data set: If there is a WHERE statement on the non-existent BASE= data set, regardless of the WHEREUP option setting, you use the WHERE statement. △

Note: You cannot append a data set to itself by using the WHERE= data set option. △

Choosing between the SET Statement and the APPEND Statement

If you use the SET statement in a DATA step to concatenate two data sets, SAS must process all the observations in both data sets to create a new one. The APPEND statement bypasses the processing of data in the original data set and adds new observations directly to the end of the original data set. Using the APPEND statement can be more efficient than using a SET statement if

- the BASE= data set is large
- all variables in the BASE= data set have the same length and type as the variables in the DATA= data set and if all variables exist in both data sets.

  Note: You can use the CONTENTS statement to see the variable lengths and types. △

The APPEND statement is especially useful if you frequently add observations to a SAS data set (for example, in production programs that are constantly appending data to a journal-type data set).

Appending Password-Protected SAS Data Sets

In order to use the APPEND statement, you need read access to the DATA= data set and write access to the BASE= data set. To gain access, use the READ= and WRITE= data set options in the APPEND statement the way you would use them in any other SAS statement, which is in parentheses immediately after the data set name. When you are appending password-protected data sets, use the following guidelines:

- If you do not give the read password for the DATA= data set in the APPEND statement, by default the procedure looks for the read password for the DATA= data set in the PROC DATASETS statement. However, the procedure does not look for the write password for the BASE= data set in the PROC DATASETS statement. Therefore, you must specify the write password for the BASE= data set in the APPEND statement.
- If the BASE= data set is read-protected only, you must specify its read password in the APPEND statement.

Appending to a Compressed Data Set

You can concatenate compressed SAS data sets. Either or both of the BASE= and DATA= data sets can be compressed. If the BASE= data set allows the reuse of space from deleted observations, the APPEND statement may insert the observations into the middle of the BASE= data set to make use of available space.

For information on the COMPRESS= and REUSE= data set and system options, see *SAS Language Reference: Dictionary*.

Appending to an Indexed Data Set

Beginning with Version 7, the behavior of appending to an indexed data set changed to improve performance.

- In Version 6, when you appended to an indexed data set, the index was updated for each added observation. Index updates tend to be random; therefore, disk I/O could have been high.
- Currently, SAS does not update the index until all observations are added to the data set. After the append, SAS internally sorts the observations and inserts the data into the index in sequential order, which reduces most of the disk I/O and results in a faster append method.

The current method is used by default when the following requirements are met; otherwise, the Version 6 method is used:

- The BASE= data set is open for member-level locking.
- The BASE= data set does not contain referential integrity constraints.
- The BASE= data set is not accessed using the Cross Environment Data Access (CEDA) facility.
- The BASE= data set is not using a WHERE= data set option.

To display information in the SAS log about the append method that is being used, you can specify the MSGLEVEL= system option as follows:

```
options msglevel=i;
```

Either a message displays if the fast-append method is in use or a message or messages display as to why the fast-append method is not in use.

The current append method initially adds observations to the BASE= data set regardless of the restrictions that are determined by the index. For example, a variable that has an index that was created with the UNIQUE option does not have its values validated for uniqueness until the index is updated. Then, if a nonunique value is detected, the offending observation is deleted from the data set. This means that after observations are appended, some of them may subsequently be deleted.

For a simple example, consider that the BASE= data set has ten observations numbered from 1 to 10 with a UNIQUE index for the variable ID. You append a data set that contains five observations numbered from 1 to 5, and observations 3 and 4 both contain the same value for ID. The following occurs

1 After the observations are appended, the BASE= data set contains 15 observations numbered from 1 to 15.

2 SAS updates the index for ID, validates the values, and determines that observations 13 and 14 contain the same value for ID.

3 SAS deletes one of the observations from the BASE= data set, resulting in 14 observations that are numbered from 1 to 15. For example, observation 13 is deleted. Note that you cannot predict which observation will be deleted, because the internal sort may place either observation first. (In Version 6, you could predict that observation 13 would be added and observation 14 would be rejected.)

If you do not want the current behavior (which could result in deleted observations) or if you want to be able to predict which observations are appended, request the Version 6 append method by specifying the APPENDVER=V6 option:

```
proc datasets;
   append base=a data=b appendver=v6;
run;
```

Note: In Version 6, deleting the index and then recreating it after the append could improve performance. The current method may eliminate the need to do that. However, the performance depends on the nature of your data. △

Appending to Data Sets with Different Variables

If the DATA= data set contains variables that are not in the BASE= data set, use the FORCE option in the APPEND statement to force the concatenation of the two data sets. The APPEND statement drops the extra variables and issues a warning message.

If the BASE= data set contains a variable that is not in the DATA= data set, the APPEND statement concatenates the data sets, but the observations from the DATA= data set have a missing value for the variable that was not present in the DATA= data set. The FORCE option is not necessary in this case.

Appending to Data Sets That Contain Variables with Different Attributes

If a variable has different attributes in the BASE= data set than it does in the DATA= data set, the attributes in the BASE= data set prevail.

If the length of a variable is longer in the DATA= data set than in the BASE= data set, or if the same variable is a character variable in one data set and a numeric variable in the other, use the FORCE option. Using FORCE has these consequences:

- The length of the variables in the BASE= data set takes precedence. SAS truncates values from the DATA= data set to fit them into the length that is specified in the BASE= data set.
- The type of the variables in the BASE= data set takes precedence. The APPEND statement replaces values of the wrong type (all values for the variable in the DATA= data set) with missing values.

Appending Data Sets That Contain Integrity Constraints

If the DATA= data set contains integrity constraints and the BASE= data set does not exist, the APPEND statement copies the general constraints. Note that the referential constraints are not copied. If the BASE= data set exists, the APPEND action copies only observations.

Appending with Generation Groups

You can use the GENNUM= data set option to append to a specific version in a generation group. Here are examples:

SAS Statements	Result

`proc datasets;`
`   append base=a`
`      data=b(gennum=2);`
	appends historical version B#002 to base A

`proc datasets;`
`   append base=a(gennum=2)`
`      data=b(gennum=2);`
	appends historical version B#002 to historical version A#002

Using the APPEND Procedure instead of the APPEND Statement

The only difference between the APPEND procedure and the APPEND statement in PROC DATASETS, is the default for *libref* in the BASE= and DATA= arguments. For PROC APPEND, the default is either WORK or USER. For the APPEND statement, the default is the libref of the procedure input library.

System Failures

If a system failure or some other type of interruption occurs while the procedure is executing, the append operation may not be successful; it is possible that not all, perhaps none, of the observations will be added to the BASE= data set. In addition, the BASE= data set may suffer damage. The APPEND operation performs an update in place, which means that it does not make a copy of the original data set before it begins to append observations. If you want to be able to restore the original observations, you can initiate an audit trail for the base data file and select to store a before-update image of the observations. Then you can write a DATA step to extract and reapply the original observations to the data file. For information about initiating an audit trail, see the PROC DATASETS “AUDIT Statement” on page 340.

AUDIT Statement

Initiates and controls event logging to an audit file as well as suspends, resumes, or terminates event logging in an audit file.

See also: “Understanding an Audit Trail” in *SAS Language Reference: Concepts*

Tip: The AUDIT statement takes one of two forms, depending on whether you are initiating the audit trail or suspending, resuming, or terminating event logging in an audit file.

AUDIT *SAS-file* <(*SAS-password*)>;

INITIATE;
<LOG <BEFORE_IMAGE=YES | NO>
<DATA_IMAGE=YES | NO>
<ERROR_IMAGE=YES | NO>>;
<USER_VAR *variable-1* <... *variable-n*>>;

AUDIT *SAS-file* <(<*SAS-password*> <GENNUM= *integer*>)>;

SUSPEND | RESUME | TERMINATE;

Required Arguments and Statements

SAS-file
specifies the SAS data file in the procedure input library that you want to audit.

INITIATE
creates an audit file that has the same name as the SAS data file and a data set type of AUDIT. The audit file logs additions, deletions, and updates to the SAS data file. You must initiate an audit trail before you can suspend, resume, or terminate it.

Options

SAS-password
specifies the password for the SAS data file, if one exists. The parentheses are required.

GENNUM=*integer*
specifies that the SUSPEND, RESUME, or TERMINATE action be performed on the audit trail of a generation file. You cannot initiate an audit trail on a generation file. Valid values for GENNUM= are *integer*, which is a number that references a specific version from a generation group. Specifying a positive number is an absolute reference to a specific generation number that is appended to a data set's name; that is, `gennum=2` specifies MYDATA#002. Specifying a negative number is a relative reference to a historical version in relation to the base version, from the youngest to the oldest; that is, `gennum=-1` refers to the youngest historical version. Specifying 0, which is the default, refers to the base version. The parentheses are required.

LOG
specifies the audit settings. The audit settings are

BEFORE_IMAGE=YES|NO
controls the storage of before-update record images.

DATA_IMAGE=YES|NO
controls the storage of after-update record images.

ERROR_IMAGE=YES|NO
controls the storage of unsuccessful after-update record images.
When the LOG statement is omitted, the default behavior is to log all images.

USER_VAR *variable-1* < ... *variable-n*>
defines optional variables to be logged in the audit file with each update to an observation. The syntax for defining variables is

USER_VAR *variable-name-1* <$> <*length*> <LABEL='*variable-label*' >
<... *variable-name-n* <$> <*length*> <LABEL='*variable-label*'> >

where

variable-name
is a name for the variable.

$
indicates that the variable is a character variable.

length
specifies the length of the variable. If a length is not specified, the default is 8.

LABEL='*variable-label*'

specifies a label for the variable.

You can define attributes such as format and informat for the user variables in the data file by using the PROC DATASETS MODIFY statement.

SUSPEND
suspends event logging to the audit file, but does not delete the audit file.

RESUME
resumes event logging to the audit file, if it was suspended.

TERMINATE
terminates event logging and deletes the audit file.

Creating an Audit File

The following example creates the audit file MYLIB.MYFILE.AUDIT to log updates to the data file MYLIB.MYFILE.DATA, storing all available record images:

```
proc datasets library=MyLib;
   audit MyFile (alter=MyPassword);
   initiate;
run;
```

The following example creates the same audit file but stores only error record images:

```
proc datasets library=MyLib;
   audit MyFile (alter=MyPassword);
   initiate;
   log data_image=NO before_image=NO;
run;
```

CHANGE Statement

Renames one or more SAS files in the same SAS data library.

Featured in: Example 1 on page 392

CHANGE *old-name-1=new-name-1*
<*…old-name-n=new-name-n* >
</ <ALTER=*alter-password*>
<GENNUM=ALL|*integer*>
<MEMTYPE=*mtype*>>;

Required Arguments

old-name=new-name
changes the name of a SAS file in the input data library. *old-name* must be the name of an existing SAS file in the input data library.

Featured in: Example 1 on page 392

Options

ALTER=*alter-password*
provides the alter password for any alter-protected SAS files named in the CHANGE statement. Because a CHANGE statement changes the names of SAS files, you need alter access to use the CHANGE statement for *new-name*. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

See also: "Using Passwords with the DATASETS Procedure" on page 377

GENNUM=ALL|*integer*
restricts processing for generation data sets. You can use the option either in parentheses after the name of each SAS file or after a forward slash. Valid values are

ALL | 0
refers to the base version and all historical versions of a generation group.

integer
refers to a specific version from a generation group. Specifying a positive number is an absolute reference to a specific generation number that is appended to a data set's name; that is, **gennum=2** specifies MYDATA#002. Specifying a negative number is a relative reference to a historical version in relation to the base version, from the youngest to the oldest; that is, **gennum=-1** refers to the youngest historical version.
For example, the following statements change the name of version A#003 to base B:

```
proc datasets;
   change A=B / gennum=3;

proc datasets;
   change A(gennum=3)=B;
```

The following CHANGE statement produces an error:

```
proc datasets;
   change A(gennum=3)=B(gennum=3);
```

See also: "Restricting Processing for Generation Data Sets" on page 380

See also: "Understanding Generation Data Sets" in *SAS Language Reference: Concepts*

MEMTYPE=*mtype*
restricts processing to one member type. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

Aliases: MTYPE=, MT=

Default: If you do not specify MEMTYPE= in the PROC DATASETS statement, the default is MEMTYPE=ALL.

See also: "Restricting Member Types for Processing" on page 378

Details

- The CHANGE statement changes names by the order that the *old-names* occur in the directory listing, not in the order that you list the changes in the CHANGE statement.
- If the *old-name* SAS file does not exist in the SAS data library, PROC DATASETS stops processing the RUN group containing the CHANGE statement and issues an error message. To override this behavior, use the NOWARN option in the PROC DATASETS statement.
- If you change the name of a data set that has an index, the index continues to correspond to the data set.

CONTENTS Statement

Describes the contents of one or more SAS data sets and prints the directory of the SAS data library.

Reminder: You can use data set options with the DATA=, OUT=, and OUT2= options. See "Data Set Options" on page 17 for a list. You can use any global statements as well. See "Global Statements" on page 18.

Featured in: Example 4 on page 400

CONTENTS <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Specify the name for an output data set	OUT=
Specify the name of an output data set to contain information about indexes and integrity constraints	OUT2=
Include information in the output about the number of observations, number of variables, number of indexes, and data set labels	DETAILS | NODETAILS
Print a list of the SAS files in the SAS data library	DIRECTORY
Print the length of a variable's informat or format	FMTLEN
Restrict processing to one or more types of SAS files	MEMTYPE=
Suppress the printing of individual files	NODS
Suppress the printing of the output	NOPRINT
Print a list of the variables by their position in the data set	VARNUM
Print abbreviated output	SHORT
Print centiles information for indexed variables	CENTILES

Options

CENTILES

prints centiles information for indexed variables.

The following additional fields are printed in the default report of PROC CONTENTS when the CENTILES option is selected and an index exists on the data set. Note that the additional fields depend on whether the index is simple or complex.

#	number of the index on the data set.
Index	name of the index.

Update Centiles	percent of the data values that must be changed before the CENTILES for the indexed variables are automatically updated.
Current Update Percent	percent of index updated since CENTILES were refreshed.
# of Unique Values	number of unique indexed values.
Variables	names of the variables used to make up the index. Centile information is listed below the variables.

DATA=*SAS-file-specification*
specifies an entire library or a specific SAS data set within a library. *SAS-file-specification* can take one of the following forms:

<libref.>SAS-data-set
names one SAS data set to process. The default for *libref* is the libref of the procedure input library. For example, to obtain the contents of the SAS data set HTWT from the procedure input library, use the following CONTENTS statement:

```
contents data=HtWt;
```

To obtain the contents of a specific version from a generation group, use the GENNUM= data set option as shown in the following CONTENTS statement:

```
contents data=HtWt(gennum=3);
```

*<libref.>*_ALL_
gives you information about all SAS data sets that have the type or types specified by the MEMTYPE= option. *libref* refers to the SAS data library. The default for *libref* is the libref of the procedure input library.

- If you are using the _ALL_ keyword, you need read access to all read-protected SAS data sets in the SAS data library.
- DATA=_ALL_ automatically prints a listing of the SAS files that are contained in the SAS library. Note that for SAS views, all librefs that are associated with the views must be assigned in the current session in order for them to be processed for the listing.

Default: most recently created data set in your job or session, from any SAS data library.

Tip: If you specify a read-protected data set in the DATA= option but do not give the read password, by default the procedure looks in the PROC DATASETS statement for the read password. However, if you do not specify the DATA= option and the default data set (last one created in the session) is read protected, the procedure does not look in the PROC DATASETS statement for the read password.

Featured in: Example 4 on page 400

DETAILS|NODETAILS
DETAILS includes these additional columns of information in the output, but only if DIRECTORY is also specified.

Default: If neither DETAILS or NODETAILS is specified, the defaults are as follows: for the CONTENTS procedure, the default is the system option setting, which is NODETAILS; for the CONTENTS statement, the default is whatever is specified on the PROC DATASETS statement, which also defaults to the system option setting.

See also: description of the additional columns in “Options” in “PROC DATASETS Statement” on page 330

DIRECTORY

prints a list of all SAS files in the specified SAS data library. If DETAILS is also specified, using DIRECTORY causes the additional columns described in DETAILS|NODETAILS on page 331 to be printed.

FMTLEN

prints the length of the informat or format. If you do not specify a length for the informat or format when you associate it with a variable, the length does not appear in the output of the CONTENTS statement unless you use the FMTLEN option. The length also appears in the FORMATL or INFORML variable in the output data set.

MEMTYPE=(*mtype(s)*)

restricts processing to one or more member types. The CONTENTS statement produces output only for member types DATA, VIEW, and ALL, which includes DATA and VIEW.

MEMTYPE= in the CONTENTS statement differs from MEMTYPE= in most of the other statements in the DATASETS procedure in the following ways:

- A slash does not precede the option.
- You cannot enclose the MEMTYPE= option in parentheses to limit its effect to only the SAS file immediately preceding it.

Specifying the MEMTYPE= option in the PROC DATASETS statement affects the CONTENTS statement only if you specify the _ALL_ keyword in the DATA= option. For example, the following statements produce the contents of only the SAS data sets with member type DATA:

```
proc datasets memtype=data;
   contents data=_all_;
run;
```

Aliases: MT=, MTYPE=

Default: DATA

NODS

suppresses printing the contents of individual files when you specify _ALL_ in the DATA= option. The CONTENTS statement prints only the SAS data library directory. You cannot use the NODS option when you specify only one SAS data set in the DATA= option.

NODETAILS

See the description of DETAILS|NODETAILS.

NOPRINT

suppresses printing the output of the CONTENTS statement.

OUT=*SAS-data-set*

names an output SAS data set.

Tip: OUT= does not suppress the printed output from the statement. If you want to suppress the printed output, you must use the NOPRINT option.

See also: "The OUT= Data Set" on page 386 for a description of the variables in the OUT= data set.

OUT2=*SAS-data-set*

names the output data set to contain information about indexes and integrity constraints.

Tip: OUT2= does not suppress the printed output from the statement. To suppress the printed output, use the NOPRINT option.

See also: "The OUT2= Data Set" on page 391 for a description of the variables in the OUT2= data set.

SHORT
prints only the list of variable names, the index information, and the sort information for the SAS data set.

VARNUM
prints a list of the variable names in the order of their logical position in the data set. By default, the CONTENTS statement lists the variables alphabetically. The physical position of the variable in the data set is engine-dependent.

Using the CONTENTS Procedure instead of the CONTENTS Statement

The only difference between the CONTENTS procedure and the CONTENTS statement in PROC DATASETS is the default for *libref* in the DATA= option. For PROC CONTENTS, the default is either WORK or USER. For the CONTENTS statement, the default is the libref of the procedure input library.

COPY Statement

Copies all or some of the SAS files in a SAS library.

Featured in: Example 1 on page 392

COPY OUT=*libref-1*
<CLONE | NOCLONE>
<CONSTRAINT=YES | NO>
<DATECOPY>
<FORCE>
<IN=*libref-2*>
<INDEX=YES | NO>
<MEMTYPE=(*mtype(s)*)>
<MOVE <ALTER=*alter-password*>> ;

Required Arguments

OUT=*libref-1*
names the SAS library to copy SAS files to.

Aliases: OUTLIB= and OUTDD=

Featured in: Example 1 on page 392

Options

ALTER=*alter-password*
provides the alter password for any alter-protected SAS files that you are moving from one data library to another. Because the MOVE option deletes the SAS file from the original data library, you need alter access to move the SAS file.

See also: "Using Passwords with the DATASETS Procedure" on page 377

CLONE | NOCLONE

specifies whether to copy the following data set attributes:

- size of input/output buffers
- whether the data set is compressed
- whether free space is reused
- data representation of input data set, library, or operating environment.

You specify these attributes with either data set options, SAS system options, or LIBNAME statement options:

- BUFSIZE= value for the size of the input/output buffers
- COMPRESS= value for whether the data set is compressed
- REUSE= value for whether free space is reused
- OUTREP= value for data representation.

For the BUFSIZE= attribute, the following table summarizes how the COPY statement works:

Table 15.1 CLONE and the BUFSIZE= Attribute

If you use...	the COPY statement...
CLONE	uses the BUFSIZE= value from the input data set for the output data set.
NOCLONE	uses the current setting of the SAS system option BUFSIZE= for the output data set.
neither	determines the type of access method, sequential or random, used by the engine for the input data set and the engine for the output data set. If both engines use the same type of access, the COPY statement uses the BUFSIZE= value from the input data set for the output data set. If the engines do not use the same type of access, the COPY statement uses the setting of SAS system option BUFSIZE= for the output data set.

For the COMPRESS= and REUSE= attributes, the following table summarizes how the COPY statement works:

Table 15.2 CLONE and the COMPRESS= and REUSE= Attributes

If you use...	the COPY statement...
CLONE	uses the values from the input data set for the output data set. If the engine for the input data set does not support the COMPRESS= or REUSE= attribute, the COPY statement uses the current setting of the corresponding SAS system option.
NOCLONE	uses the current setting of the SAS system options COMPRESS= or REUSE= for the output data set.
neither	defaults to CLONE.

For data representation:

CLONE
: results in a copy with the data representation of the input data set.

NOCLONE results in a copy with the data representation of the data library (if specified) or the native data representation of the operating environment.

Data representation is the format in which data is represented on a computer architecture or in an operating environment. For example, on an IBM PC, character data is represented by its ASCII encoding and byte-swapped integers. Native data representation refers to an environment for which the data representation compares with the CPU that is accessing the file. For example, a file that is in Windows data representation is native to the Windows operating environment.

CONSTRAINT=YES|NO
specifies whether to copy all integrity constraints when copying a data set.

Default: NO

DATECOPY
copies the SAS internal date and time when the SAS file was created and the date and time when it was last modified to the resulting copy of the file. Note that the operating environment date and time are not preserved.

Restriction: DATECOPY cannot be used with encrypted files or catalogs.

Restriction: DATECOPY can be used only when the resulting SAS file uses the V8 or V9 engine.

Tip: You can alter the file creation date and time with the DTC= option on the MODIFY statement. See “MODIFY Statement” on page 366.

Tip: If the file that you are copying has attributes that require additional processing, the last modified date is changed to the current date. For example, when you copy a data set that has an index, the index must be rebuilt, and this changes the last modified date to the current date. Other attributes that require additional processing and that could affect the last modified date include integrity constraints and a sort indicator.

FORCE
allows you to use the MOVE option for a SAS data set on which an audit trail exists.

Note: The AUDIT file is not moved with the audited data set. △

IN=*libref-2*
names the SAS library containing SAS files to copy.

Aliases: INLIB= and INDD=

Default: the libref of the procedure input library

To copy only selected members, use the SELECT or EXCLUDE statements.

INDEX=YES|NO
specifies whether to copy all indexes for a data set when copying the data set to another SAS data library.

Default: YES

MEMTYPE=(*mtype(s)*)
restricts processing to one or more member types.

Aliases: MT=, MTYPE=

Default: If you omit MEMTYPE= in the PROC DATASETS statement, the default is MEMTYPE=ALL.

See also: “Specifying Member Types When Copying or Moving SAS Files” on page 350

See also: “Member Types” on page 379

Featured in: Example 1 on page 392

MOVE
moves SAS files from the input data library (named with the IN= option) to the output data library (named with the OUT= option) and deletes the original files from the input data library.

Restriction: The MOVE option can be used to delete a member of a SAS library only if the IN= engine supports the deletion of tables. A tape format engine does not support table deletion. If you use a tape format engine, SAS suppresses the MOVE operation and prints a warning.

Featured in: Example 1 on page 392

NOCLONE
See the description of CLONE.

Copying an Entire Library

To copy an entire SAS data library, simply specify an input data library and an output data library following the COPY statement. For example, the following statements copy all the SAS files in the SOURCE data library into the DEST data library:

```
proc datasets library=source;
   copy out=dest;
run;
```

Copying Selected SAS Files

To copy selected SAS files, use a SELECT or EXCLUDE statement. For more discussion of using the COPY statement with a SELECT or an EXCLUDE statement, see "Specifying Member Types When Copying or Moving SAS Files" on page 350and see Example 1 on page 392 for an example. Also, see "EXCLUDE Statement" on page 357 and "SELECT Statement" on page 374.

You can also select or exclude an abbreviated list of members. For example, the following statement selects members TABS, TEST1, TEST2, and TEST3:

```
select tabs test1-test3;
```

Also, you can select a group of members whose names begin with the same letter or letters by entering the common letters followed by a colon (:). For example, you can select the four members in the previous example and all other members having names that begin with the letter T by specifying the following statement:

```
select t:;
```

You specify members to exclude in the same way that you specify those to select. That is, you can list individual member names, use an abbreviated list, or specify a common letter or letters followed by a colon (:). For example, the following statement excludes the members STATS, TEAMS1, TEAMS2, TEAMS3, TEAMS4 and all the members that begin with the letters RBI from the copy operation:

```
exclude stats teams1-teams4 rbi:;
```

Note that the MEMTYPE= option affects which types of members are available to be selected or excluded.

Specifying Member Types When Copying or Moving SAS Files

The MEMTYPE= option in the COPY statement differs from the MEMTYPE= option in other statements in the procedure in several ways:

- □ A slash does not precede the option.

- You cannot limit its effect to the member immediately preceding it by enclosing the MEMTYPE= option in parentheses.
- The SELECT and EXCLUDE statements and the IN= option (in the COPY statement) affect the behavior of the MEMTYPE= option in the COPY statement according to the following rules:

  1. MEMTYPE= in a SELECT or EXCLUDE statement takes precedence over the MEMTYPE= option in the COPY statement. The following statements copy only VISION.CATALOG and NUTR.DATA from the default data library to the DEST data library; the MEMTYPE= value in the first SELECT statement overrides the MEMTYPE= value in the COPY statement.

```
proc datasets;
   copy out=dest memtype=data;
      select vision(memtype=catalog) nutr;
run;
```

  2. If you do not use the IN= option, or you use it to specify the library that happens to be the procedure input library, the value of the MEMTYPE= option in the PROC DATASETS statement limits the types of SAS files that are available for processing. The procedure uses the order of precedence described in rule 1 to further subset the types available for copying. The following statements do not copy any members from the default data library to the DEST data library; instead, the procedure issues an error message because the MEMTYPE= value specified in the SELECT statement is not one of the values of the MEMTYPE= option in the PROC DATASETS statement.

```
      /* This step fails! */
proc datasets memtype=(data program);
   copy out=dest;
      select apples / memtype=catalog;
run;
```

  3. If you specify an input data library in the IN= option other than the procedure input library, the MEMTYPE= option in the PROC DATASETS statement has no effect on the copy operation. Because no subsetting has yet occurred, the procedure uses the order of precedence described in rule 1 to subset the types available for copying. The following statements successfully copy BODYFAT.DATA to the DEST data library because the SOURCE library specified in the IN= option in the COPY statement is not effected by the MEMTYPE= option in the PROC DATASETS statement.

```
proc datasets library=work memtype=catalog;
   copy in=source out=dest;
      select bodyfat / memtype=data;
run;
```

Copying Password-Protected SAS Files

You can copy a password-protected SAS file without specifying the password. In addition, because the password continues to correspond to the SAS file, you must know the password in order to access and manipulate the SAS file after you copy it.

Copying Data Sets with Long Variable Names

If the VALIDVARNAME=V6 system option is set and the data set has long variable names, the long variable names are truncated, unique variables names are generated, and the copy succeeds. The same is true for index names. If VALIDVARNAME=ANY or

MIXEDCASE, the copy fails with an error if the OUT= engine does not support long variable names.

When a variable name is truncated, the variable name is shortened to eight bytes. If this name has already been defined in the data set, the name is shortened and a digit is added, starting with the number 2. The process of truncation and adding a digit continues until the variable name is unique. For example, a variable named LONGVARNAME becomes LONGVARN, provided that a variable with that name does not already exist in the data set. In that case, the variable name becomes LONGVAR2.

CAUTION:

Truncated variable names can collide with names already defined in the input data set. This is possible when the variable name that is already defined is exactly eight bytes long and ends in a digit. In that case, the truncated name is defined in the output data set and the name from the input data set is changed. For example,

```
options validvarname=mixedcase;
data test;
   lonvar10='aLongVariableName';
   retain longvar1-longvar5 0;
run;

options validvarname=v6;
proc copy in=work out=sasuser;
   select test;
run;
```

In this example, LONGVAR10 is truncated to LONVAR1 and placed in the output data set. Next, the original LONGVAR1 is copied. Its name is no longer unique and so it is renamed LONGVAR2. The other variables in the input data set are also renamed according to the renaming algorithm. △

Using the COPY Procedure instead of the COPY Statement

Generally, the COPY procedure functions the same as the COPY statement in the DATASETS procedure. The differences are

- The IN= argument is required with PROC COPY. In the COPY statement, IN= is optional. If omitted, the default value is the libref of the procedure input library.
- PROC DATASETS cannot work with libraries that allow only sequential data access.
- The COPY statement honors the NOWARN option but PROC COPY does not.

Copying Generation Groups

You can use the COPY statement to copy an entire generation group. However, you cannot copy a specific version in a generation group.

Transporting SAS Data Sets between Hosts

Typically, you use PROC COPY to transport SAS data sets between hosts. See Chapter 11, "The COPY Procedure," on page 259 for more information and an example.

DELETE Statement

Deletes SAS files from a SAS data library.

Featured in: Example 1 on page 392

DELETE *SAS-file(s)*
</ <ALTER=*alter-password*>
<GENNUM=ALL|HIST|REVERT|*integer*>
<MEMTYPE=*mtype*>>;

Required Arguments

SAS-file(s)
specifies one or more SAS files that you want to delete.

Options

ALTER=*alter-password*
provides the alter password for any alter-protected SAS files that you want to delete. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

See also: "Using Passwords with the DATASETS Procedure" on page 377

GENNUM=ALL|HIST|REVERT|*integer*
restricts processing for generation data sets. You can use the option either in parentheses after the name of each SAS file or after a forward slash. Valid values are

ALL
refers to the base version and all historical versions in a generation group.

HIST
refers to all historical versions, but excludes the base version in a generation group.

REVERT|0
deletes the base version and changes the most current historical version, if it exists, to the base version.

integer
is a number that references a specific version from a generation group. Specifying a positive number is an absolute reference to a specific generation number that is appended to a data set's name; that is, **`gennum=2`** specifies MYDATA#002. Specifying a negative number is a relative reference to a historical version in relation to the base version, from the youngest to the oldest; that is, **`gennum=-1`** refers to the youngest historical version.

See also: "Restricting Processing for Generation Data Sets" on page 380

See also: "Understanding Generation Data Sets" in *SAS Language Reference: Concepts*

MEMTYPE=*mtype*
restricts processing to one member type. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

Aliases: MT=, MTYPE=

Default: DATA

See also: "Restricting Member Types for Processing" on page 378

Featured in: Example 1 on page 392

Details

- SAS immediately deletes SAS files when the RUN group executes. You do not have an opportunity to verify the delete operation before it begins.
- If you attempt to delete a SAS file that does not exist in the procedure input library, PROC DATASETS issues a message and continues processing. If NOWARN is used, no message is issued.
- When you use the DELETE statement to delete a data set that has indexes associated with it, the statement also deletes the indexes.
- You cannot use the DELETE statement to delete a data file that has a foreign key integrity constraint or a primary key with foreign key references. For data files that have foreign keys, you must remove the foreign keys before you delete the data file. For data files that have primary keys with foreign key references, you must remove the foreign keys that reference the primary key before you delete the data file.

Working with Generation Groups

When you are working with generation groups, you can use the DELETE statement to

- delete the base version and all historical versions
- delete the base version and rename the youngest historical version to the base version
- delete an absolute version
- delete a relative version
- delete all historical versions and leave the base version.

Deleting the Base Version and All Historical Versions

The following statements delete the base version and all historical versions where the data set name is A:

```
proc datasets;
   delete A(gennum=all);

proc datasets;
   delete A / gennum=all;

proc datasets gennum=all;
   delete A;
```

The following statements delete the base version and all historical versions where the data set name begins with the letter A:

```
proc datasets;
   delete A:(gennum=all);

proc datasets;
   delete A: / gennum=all;

proc datasets gennum=all;
```

```
   delete A:;
```

Deleting the Base Version and Renaming the Youngest Historical Version to the Base Version

The following statements delete the base version and rename the youngest historical version to the base version, where the data set name is A:

```
proc datasets;
   delete A(gennum=revert);

proc datasets;
   delete A / gennum=revert;

proc datasets gennum=revert;
   delete A;
```

The following statements delete the base version and rename the youngest historical version to the base version, where the data set name begins with the letter A:

```
proc datasets;
   delete A:(gennum=revert);

proc datasets;
   delete A: / gennum=revert;

proc datasets gennum=revert;
   delete A:;
```

Deleting a Version with an Absolute Number

The following statements use an absolute number to delete the first historical version:

```
proc datasets;
   delete A(gennum=1);

proc datasets;
   delete A / gennum=1;

proc datasets gennum=1;
   delete A;
```

The following statements delete a specific historical version, where the data set name begins with the letter A:

```
proc datasets;
   delete A:(gennum=1);

proc datasets;
   delete A: / gennum=1;

proc datasets gennum=1;
   delete A:;
```

Deleting a Version with a Relative Number

The following statements use a relative number to delete the youngest historical version, where the data set name is A:

```
proc datasets;
   delete A(gennum=-1);
```

```
proc datasets;
   delete A / gennum=-1;

proc datasets gennum=-1;
   delete A;
```

The following statements use a relative number to delete the youngest historical version, where the data set name begins with the letter A:

```
proc datasets;
   delete A:(gennum=-1);

proc datasets;
   delete A: / gennum=-1;

proc datasets gennum=-1;
   delete A:;
```

Deleting All Historical Versions and Leaving the Base Version

The following statements delete all historical versions and leave the base version, where the data set name is A:

```
proc datasets;
   delete A(gennum=hist);

proc datasets;
   delete A / gennum=hist;

proc datasets gennum=hist;
   delete A;
```

The following statements delete all historical versions and leave the base version, where the data set name begins with the letter A:

```
proc datasets;
   delete A:(gennum=hist);

proc datasets;
   delete A: / gennum=hist;

proc datasets gennum=hist;
   delete A:;
```

EXCHANGE Statement

Exchanges the names of two SAS files in a SAS library.

Featured in: Example 1 on page 392

EXCHANGE *name-1=other-name-1*
<*...name-n=other-name-n*>
</ <ALTER=*alter-password*>

<MEMTYPE=*mtype*>>;

Required Arguments

name=other-name
exchanges the names of SAS files in the procedure input library. Both *name* and *other-name* must already exist in the procedure input library.

Options

ALTER=*alter-password*
provides the alter password for any alter-protected SAS files whose names you want to exchange. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

See also: "Using Passwords with the DATASETS Procedure" on page 377

MEMTYPE=*mtype*
restricts processing to one member type. You can only exchange the names of SAS files of the same type. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

Default: If you do not specify MEMTYPE= in the PROC DATASETS statement, the default is ALL.

See also: "Restricting Member Types for Processing" on page 378

Details

- When you exchange more than one pair of names in one EXCHANGE statement, PROC DATASETS performs the exchanges in the order that the names of the SAS files occur in the directory listing, not in the order that you list the exchanges in the EXCHANGE statement.
- If the *name* SAS file does not exist in the SAS data library, PROC DATASETS stops processing the RUN group that contains the EXCHANGE statement and issues an error message. To override this behavior, specify the NOWARN option in the PROC DATASETS statement.
- The EXCHANGE statement also exchanges the associated indexes so that they correspond with the new name.
- The EXCHANGE statement only allows two existing generation groups to exchange names. You cannot exchange a specific generation number with either an existing base version or another generation number.

EXCLUDE Statement

Excludes SAS files from copying.

Restriction: Must follow a COPY statement

Restriction: Cannot appear in the same COPY step with a SELECT statement

Featured in: Example 1 on page 392

EXCLUDE *SAS-file(s)* </ MEMTYPE=*mtype*>;

Required Arguments

SAS-file(s)
specifies one or more SAS files to exclude from the copy operation. All SAS files you name in the EXCLUDE statement must be in the library that is specified in the IN= option in the COPY statement. If the SAS files are generation groups, the EXCLUDE statement allows only selection of the base versions.

Options

MEMTYPE=*mtype*
restricts processing to one member type. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

Aliases: MTYPE=, MT=

Default: If you do not specify MEMTYPE= in the PROC DATASETS statement, the COPY statement, or in the EXCLUDE statement, the default is MEMTYPE=ALL.

See also: "Restricting Member Types for Processing" on page 378

See also: "Specifying Member Types When Copying or Moving SAS Files" on page 350

Excluding Many Like-Named Files

You can use shortcuts for listing many SAS files in the EXCLUDE statement. For more information, see "Shortcuts for Specifying Lists of Variable Names" on page 24.

FORMAT Statement

Permanently assigns, changes, and removes variable formats in the SAS data set specified in the MODIFY statement.

Restriction: Must appear in a MODIFY RUN group

Featured in: Example 3 on page 398

FORMAT *variable-list-1 <format-1>*
<...variable-list-n <format-n>>;

Required Arguments

variable-list
specifies one or more variables whose format you want to assign, change, or remove. If you want to disassociate a format with a variable, list the variable last in the list with no format following. For example:

```
format x1-x3 4.1 time hhmm2.2 age;
```

Options

format
specifies a format to apply to the variable or variables listed before it. If you do not specify a format, the FORMAT statement removes any format associated with the variables in *variable-list*.

Note: You can use shortcut methods for specifying variables, such as the keywords _NUMERIC, _CHARACTER_, and _ALL_. See “Shortcuts for Specifying Lists of Variable Names” on page 24 for more information. △

IC CREATE Statement

Creates an integrity constraint.

Restriction: Must be in a MODIFY RUN group

See also: “Understanding Integrity Constraints” in *SAS Language Reference: Concepts*

IC CREATE <*constraint-name*=> *constraint* <MESSAGE=’*message-string*’
<MSGTYPE=USER>>;

Required Arguments

constraint
is the type of constraint. Valid values are as follows:

NOT NULL (*variable*)
specifies that *variable* does not contain a SAS missing value, including special missing values.

UNIQUE (*variables*)
specifies that the values of *variables* must be unique. This constraint is identical to DISTINCT.

DISTINCT (*variables*)
specifies that the values of *variables* must be unique. This constraint is identical to UNIQUE.

CHECK (WHERE-*expression*)
limits the data values of variables to a specific set, range, or list of values. This is accomplished with a WHERE expression.

PRIMARY KEY (*variables*)
specifies a primary key, that is, a set of variables that do not contain missing values and whose values are unique.

Note: A primary key affects the values of an individual data file until it has a foreign key referencing it. △

FOREIGN KEY (*variables*) REFERENCES *table-name*
<ON DELETE *referential-action*> <ON UPDATE *referential-action*>

specifies a foreign key, that is, a set of variables whose values are linked to the values of the primary key variables in another data file. The referential actions are enforced when updates are made to the values of a primary key variable that is referenced by a foreign key.

There are three types of referential actions: RESTRICT, SET NULL, and CASCADE. For a RESTRICT referential action,

a delete operation
: deletes the primary key row, but only if no foreign key values match the deleted value.

an update operation
: updates the primary key value, but only if no foreign key values match the current value to be updated.

For a SET NULL referential action,

a delete operation
: deletes the primary key row and sets the corresponding foreign key values to NULL.

an update operation
: modifies the primary key value and sets all matching foreign key values to NULL.

For a CASCADE referential action,

an update operation
: modifies the primary key value, and additionally modifies any matching foreign key values to the same value. CASCADE is not supported for delete operations.

Note: RESTRICT is the default action if no referential action is specified. Before it will enforce a SET NULL or CASCADE referential action, SAS checks to see if there are other foreign keys that reference the primary key and that specify RESTRICT for the intended operation. If RESTRICT is specified, or if the constraint reverts to the default values, then RESTRICT is enforced for all foreign keys, unless no foreign key values match the values to updated or deleted. △

Options

<constraint-name=>
is an optional name for the constraint. The name must be a valid SAS name. When you do not supply a constraint name, a default name is generated. This default constraint name has the following form

Default name	Constraint type
NM*xxxx*	Not Null
UN*xxxx*	Unique
CK*xxxx*	Check
PK*xxxx*	Primary key
FK*xxxx*	Foreign key

where *xxxx* is a counter beginning at 0001.

Note: The names PRIMARY, FOREIGN, MESSAGE, UNIQUE, DISTINCT, CHECK, and NOT cannot be used as values for *constraint-name*. △

<MESSAGE='*message-string*' <MSGTYPE=USER>>
message-string is the text of an error message to be written to the log when the data fails the constraint. For example,

```
ic create not null(socsec)
   message='Invalid Social Security number';
```

Length: The maximum length of the message is 250 characters.

<MSGTYPE=USER> controls the format of the integrity constraint error message. By default when the MESSAGE= option is specified, the message you define is inserted into the SAS error message for the constraint, separated by a space. MSGTYPE=USER suppresses the SAS portion of the message.

The following examples show how to create integrity constraints:

```
ic create a = not null(x);
ic create Unique_D = unique(d);
ic create Distinct_DE = distinct(d e);
ic create E_less_D = check(where=(e < d or d = 99));
ic create primkey = primary key(a b);
ic create forkey = foreign key (a b) references table-name
  on update cascade on delete set null;
ic create not null (x);
```

Note that for a referential constraint to be established, the foreign key must specify the same number of variables as the primary key, in the same order, and the variables must be of the same type (character/numeric) and length.

IC DELETE Statement

Deletes an integrity constraint.

Restriction: Must be in a MODIFY RUN group

See also: "Understanding Integrity Constraints" in *SAS Language Reference: Concepts*

IC DELETE *constraint-name(s)* | _ALL_;

Arguments

constraint-name(s)
names one or more constraints to delete. For example, to delete the constraints Unique_D and Unique_E, use this statement:

```
ic delete Unique_D Unique_E;
```

ALL
deletes all constraints for the SAS data file specified in the preceding MODIFY statement.

IC REACTIVATE Statement

Reactivates a foreign key integrity constraint that is inactive.

Restriction: Must be in a MODIFY RUN group

See also: “Understanding Integrity Constraints” in *SAS Language Reference: Concepts*

IC REACTIVATE *foreign-key-name* REFERENCES *libref*;

Arguments

foreign-key-name
is the name of the foreign key to reactivate.

libref
refers to the SAS library containing the data set that contains the primary key that is referenced by the foreign key.

For example, suppose that you have the foreign key FKEY defined in data set MYLIB.MYOWN and that FKEY is linked to a primary key in data set MAINLIB.MAIN. If the integrity constraint is inactivated by a copy or move operation, you can reactivate the integrity constraint by using the following code:

```
proc datasets library=mylib;
   modify myown;
   ic reactivate fkey references mainlib;
run;
```

INDEX CENTILES

Updates centiles statistics for indexed variables.

Restriction: Must be in a MODIFY RUN group

See also: “Understanding SAS Indexes” in *SAS Language Reference: Concepts*

INDEX CENTILES *index(s)*
</ <REFRESH>
<UPDATECENTILES= ALWAYS | NEVER | *integer*>>;

Required Arguments

index(s)
names one or more indexes.

Options

REFRESH
updates centiles immediately, regardless of the value of UPDATECENTILES.

UPDATECENTILES=ALWAYS | NEVER | *integer*
specifies when centiles are to be updated. It is not practical to update centiles after every data set update. Therefore, you can specify as the value of UPDATECENTILES the percent of the data values that can be changed before centiles for the indexed variables are updated.

Valid values for UPDATECENTILES are

ALWAYS | 0
updates centiles when the data set is closed if any changes have been made to the data set index.

NEVER | 101
does not update centiles.

integer
is the percent of values for the indexed variable that can be updated before centiles are refreshed.

Alias: UPDCEN

Default 5 (percent)

INDEX CREATE Statement

Creates simple or composite indexes for the SAS data set specified in the MODIFY statement.

Restriction: Must be in a MODIFY RUN group

See also: "Understanding SAS Indexes" in *SAS Language Reference: Concepts*

Featured in: Example 3 on page 398

INDEX CREATE *index-specification(s)*
</ <NOMISS>
<UNIQUE>
<UPDATECENTILES= ALWAYS | NEVER | *integer*>>;

Required Arguments

index-specification(s)
can be one or both of the following forms:

variable
creates a simple index on the specified variable.

index=(variables)
creates a composite index. The name you specify for *index* is the name of the composite index. It must be a valid SAS name and cannot be the same as any variable name or any other composite index name. You must specify at least two variables.

Options

NOMISS
excludes from the index all observations with missing values for all index variables.

When you create an index with the NOMISS option, SAS uses the index only for WHERE processing and only when missing values fail to satisfy the WHERE expression. For example, if you use the following WHERE statement, SAS does not use the index, because missing values satisfy the WHERE expression:

```
where dept ne '01';
```

Refer to *SAS Language Reference: Concepts*.

Note: BY-group processing ignores indexes that are created with the NOMISS option. △

Featured in: Example 3 on page 398

UNIQUE
specifies that the combination of values of the index variables must be unique. If you specify UNIQUE and multiple observations have the same values for the index variables, the index is not created.

Featured in: Example 3 on page 398

UPDATECENTILES=ALWAYS | NEVER | *integer*
specifies when centiles are to be updated. It is not practical to update centiles after every data set update. Therefore, you can specify the percent of the data values that can be changed before centiles for the indexed variables are updated. Valid values for UPDATECENTILES are as follows:

ALWAYS | 0
updates centiles when the data set is closed if any changes have been made to the data set index.

NEVER | 101
does not update centiles.

integer
specifies the percent of values for the indexed variable that can be updated before centiles are refreshed.

Alias: UPDCEN

Default: 5% (percent)

INDEX DELETE Statement

Deletes one or more indexes associated with the SAS data set specified in the MODIFY statement.

Restriction: Must appear in a MODIFY RUN group

INDEX DELETE *index(s)* | _ALL_;

Required Arguments

index(s)

names one or more indexes to delete. The index(es) must be for variables in the SAS data set that is named in the preceding MODIFY statement. You can delete both simple and composite indexes.

ALL
deletes all indexes, except for indexes that are owned by an integrity constraint. When an index is created, it is marked as owned by the user, by an integrity constraint, or by both. If an index is owned by both a user and an integrity constraint, the index is not deleted until both an IC DELETE statement and an INDEX DELETE statement are processed.

Note: You can use the CONTENTS statement to produce a list of all indexes for a data set. △

INFORMAT Statement

Permanently assigns, changes, and removes variable informats in the data set specified in the MODIFY statement.

Restriction: Must appear in a MODIFY RUN group

Featured in: Example 3 on page 398

INFORMAT *variable-list-1 <informat-1>*
<…variable-list-n <informat-n>>;

Required Arguments

variable-list
specifies one or more variables whose informats you want to assign, change, or remove. If you want to disassociate an informat with a variable, list the variable last in the list with no informat following. For example:

```
informat a b 2. x1-x3 4.1 c;
```

Options

informat
specifies an informat for the variables immediately preceding it in the statement. If you do not specify an informat, the INFORMAT statement removes any existing informats for the variables in *variable-list*.

Note: You can use shortcut methods for specifying variables, such as the keywords _NUMERIC, _CHARACTER_, and _ALL_. See “Shortcuts for Specifying Lists of Variable Names” on page 24 for more information. △

LABEL Statement

Assigns, changes, and removes variable labels for the SAS data set specified in the MODIFY statement.

Restriction: Must appear in a MODIFY RUN group

Featured in: Example 3 on page 398

LABEL *variable-1*=<*'label-1'*|' '>
<*...variable-n*=<*'label-n'*|' ' >>;

Required Arguments

variable=<'label'>

assigns a label to a variable. If a single quotation mark appears in the label, write it as two single quotation marks in the LABEL statement. Specifying *variable*= or *variable*=' 'removes the current label.

Range: 1-256 characters

MODIFY Statement

Changes the attributes of a SAS file and, through the use of subordinate statements, the attributes of variables in the SAS file.

Featured in: Example 3 on page 398

MODIFY *SAS-file* <(*option(s)*)>
</ <DTC=*SAS-date-time*>
<GENNUM=*integer*>
<MEMTYPE=*mtype*>>;

To do this	**Use this option**
Restrict processing to a certain type of SAS file	MEMTYPE=
Specify attributes	
Assign or change a data set label	LABEL=
Assign or change a special data set type	TYPE=
Specify how the data are currently sorted	SORTEDBY=
Specify a creation date and time	DTC=
Modify passwords	
Modify an alter password	ALTER=

To do this	Use this option
Modify a read, write, or alter password	PW=
Modify a read password	READ=
Modify a write password	WRITE=
Modify generation groups	
Modify the maximum number of versions for a generation group	GENMAX=
Modify a historical version	GENNUM=

Required Arguments

SAS-file
specifies a SAS file that exists in the procedure input library.

Options

ALTER=*password-modification*
assigns, changes, or removes an alter password for the SAS file named in the MODIFY statement. *password-modification* is one of the following:

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

See also: "Manipulating Passwords" on page 369

DTC=*SAS-date-time*
specifies a date and time to substitute for the date and time stamp placed on a SAS file at the time of creation. You cannot use this option in parentheses after the name of each SAS file; you must specify DTC= after a forward slash. For example:

```
modify mydata / dtc='03MAR00:12:01:00'dt;
```

Tip: Use DTC= to alter a SAS file's creation date and time prior to using the DATECOPY option in the CIMPORT procedure, COPY procedure, CPORT procedure, SORT procedure, and the COPY statement in the DATASETS procedure.

Restriction: A SAS file's creation date and time cannot be set later than the date and time the file was actually created.

Restriction: DTC= cannot be used with encrypted files or sequential files.

Restriction: DTC= can be used only when the resulting SAS file uses the V8 or V9 engine.

GENMAX=*number-of-generations*
specifies the maximum number of versions. You can use this option either in parentheses after the name of each SAS file or after a forward slash.

Range: 0 to 1,000

Default: 0

GENNUM=*integer*
restricts processing for generation data sets. You can specify GENNUM= either in parentheses after the name of each SAS file or after a forward slash. Valid value is *integer*, which is a number that references a specific version from a generation group. Specifying a positive number is an absolute reference to a specific generation number that is appended to a data set's name; that is, `gennum=2` specifies MYDATA#002. Specifying a negative number is a relative reference to a historical version in relation to the base version, from the youngest to the oldest; that is, `gennum=-1` refers to the youngest historical version. Specifying 0, which is the default, refers to the base version.

See also: "Understanding Generation Data Sets" in *SAS Language Reference: Concepts*

LABEL=*'data-set-label'* | ''
assigns, changes, or removes a data set label for the SAS data set named in the MODIFY statement. If a single quotation mark appears in the label, write it as two single quotation marks. LABEL= or LABEL=' 'removes the current label.

Range: 1-40 characters

Featured in: Example 3 on page 398

MEMTYPE=*mtype*
restricts processing to one member type. You cannot specify MEMTYPE= in parentheses after the name of each SAS file; you must specify MEMTYPE= after a forward slash.

Aliases: MTYPE= and MT=

Default: If you do not specify the MEMTYPE= option in the PROC DATASETS statement or in the MODIFY statement, the default is MEMTYPE=DATA.

PW=*password-modification*
assigns, changes, or removes a read, write, or alter password for the SAS file named in the MODIFY statement. *password-modification* is one of the following:

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

See also: "Manipulating Passwords" on page 369

READ=*password-modification*
assigns, changes, or removes a read password for the SAS file named in the MODIFY statement. *password-modification* is one of the following:

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

See also: "Manipulating Passwords" on page 369

Featured in: Example 3 on page 398

SORTEDBY=*sort-information*

specifies how the data are currently sorted. SAS stores the sort information with the file but does not verify that the data are sorted the way you indicate.
sort-information can be one of the following:

by-clause </ *collate-name*>
: indicates how the data are currently sorted. Values for *by-clause* are the variables and options you can use in a BY statement in a PROC SORT step. *collate-name* names the collating sequence used for the sort. By default, the collating sequence is that of your host operating environment.

NULL
: removes any existing sort information.

Restriction: The data must be sorted in the order that you specify. If the data is not in the specified order, SAS will not sort it for you.

Featured in: Example 3 on page 398

TYPE=*special-type*
assigns or changes the special data set type of a SAS data set. SAS does *not* verify

- the SAS data set type you specify in the TYPE= option (except to check if it has a length of eight or fewer characters).
- that the SAS data set's structure is appropriate for the type you have designated.

Note: Do not confuse the TYPE= option with the MEMTYPE= option. The TYPE= option specifies a type of special SAS data set. The MEMTYPE= option specifies one or more types of SAS files in a SAS data library. △

Tip: Most SAS data sets have no special type. However, certain SAS procedures, like the CORR procedure, can create a number of special SAS data sets. In addition, SAS/STAT software and SAS/EIS software support special data set types.

WRITE=*password-modification*
assigns, changes, or removes a write password for the SAS file named in the MODIFY statement. *password-modification* is one of the following:

- *new-password*
- *old-password* / *new-password*
- / *new-password*
- *old-password* /
- /

See also: "Manipulating Passwords" on page 369

Manipulating Passwords

In order to assign, change, or remove a password, you must specify the password for the highest level of protection that currently exists on that file.

Assigning Passwords

```
/* assigns a password to an unprotected file */
modify colors (pw=green);
```

```
/* assigns an alter password to an already read-protected SAS data set */
modify colors (read=green alter=red);
```

Changing Passwords

```
/* changes the write password from YELLOW to BROWN */
modify cars (write=yellow/brown);
```

```
/* uses alter access to change unknown read password to BLUE */
modify colors (read=/blue alter=red);
```

Removing Passwords

```
/* removes the alter password RED from STATES */
modify states (alter=red/);
```

```
/* uses alter access to remove the read password */
modify zoology (read=green/ alter=red);
```

```
/* uses PW= as an alias for either WRITE= or ALTER= to remove unknown
   read password */
modify biology (read=/ pw=red);
```

Working with Generation Groups

Changing the Number of Generations

```
/* changes the number of generations on data set A to 99 */
modify A (genmax=99);
```

Removing Passwords

```
/* removes the alter password RED from STATES#002 */
modify states (alter=red/) / gennum=2;
```

RENAME Statement

Renames variables in the SAS data set specified in the MODIFY statement.

Restriction: Must appear in a MODIFY RUN group

Featured in: Example 3 on page 398

RENAME *old-name-1=new-name-1*
<*...old-name-n=new-name-n*>;

Required Arguments

old-name=new-name
changes the name of a variable in the data set specified in the MODIFY statement. *old-name* must be a variable that already exists in the data set. *new-name*, which must be a valid SAS name, cannot be the name of a variable that already exists in the data set or the name of an index.

Details

- If *old-name* does not exist in the SAS data set or *new-name* already exists, PROC DATASETS stops processing the RUN group containing the RENAME statement and issues an error message.
- When you use the RENAME statement to change the name of a variable for which there is a simple index, the statement also renames the index.
- If the variable that you are renaming is used in a composite index, the composite index automatically references the new variable name. However, if you attempt to rename a variable to a name that has already been used for a composite index, you receive an error message.

REPAIR Statement

Attempts to restore damaged SAS data sets or catalogs to a usable condition.

REPAIR *SAS-file(s)*
</ <ALTER=*alter-password*>
<GENNUM=*integer*>
<MEMTYPE=*mtype*>>;

Required Arguments

SAS-file(s)
specifies one or more SAS data sets or catalogs in the procedure input library.

Options

ALTER=*alter-password*
provides the alter password for any alter-protected SAS files that are named in the REPAIR statement. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

See also: "Using Passwords with the DATASETS Procedure" on page 377

GENNUM=*integer*
restricts processing for generation data sets. You can use the option either in parentheses after the name of each SAS file or after a forward slash. Valid value is *integer*, which is a number that references a specific version from a generation group. Specifying a positive number is an absolute reference to a specific generation number that is appended to a data set's name; that is, `gennum=2` specifies MYDATA#002. Specifying a negative number is a relative reference to a historical version in relation to the base version, from the youngest to the oldest; that is, `gennum=-1` refers to the

youngest historical version. Specifying 0, which is the default, refers to the base version.

See also: "Restricting Processing for Generation Data Sets" on page 380

See also: "Understanding Generation Data Sets" in *SAS Language Reference: Concepts*

MEMTYPE=*mtype*
restricts processing to one member type.

Aliases: MT=, MTYPE=

Default: If you do not specify the MEMTYPE= option in the PROC DATASETS statement or in the REPAIR statement, the default is MEMTYPE=ALL.

See also: "Restricting Member Types for Processing" on page 378

Details

The most common situations that require the REPAIR statement are as follows:

- A system failure occurs while you are updating a SAS data set or catalog.
- The device on which a SAS data set or an associated index resides is damaged. In this case, you can restore the damaged data set or index from a backup device, but the data set and index no longer match.
- The disk that stores the SAS data set or catalog becomes full before the file is completely written to disk. You may need to free some disk space. PROC DATASETS requires free space when repairing SAS data sets with indexes and when repairing SAS catalogs.
- An I/O error occurs while you are writing a SAS data set or catalog entry.

When you use the REPAIR statement for SAS data sets, it recreates all indexes for the data set. It also attempts to restore the data set to a usable condition, but the restored data set may not include the last several updates that occurred before the system failed. You cannot use the REPAIR statement to recreate indexes that were destroyed by using the FORCE option in a PROC SORT step.

When you use the REPAIR statement for a catalog, you receive a message stating whether the REPAIR statement restored the entry. If the entire catalog is potentially damaged, the REPAIR statement attempts to restore all the entries in the catalog. If only a single entry is potentially damaged, for example when a single entry is being updated and a disk-full condition occurs, on most systems only the entry that is open when the problem occurs is potentially damaged. In this case, the REPAIR statement attempts to repair only that entry. Some entries within the restored catalog may not include the last updates that occurred before a system crash or an I/O error. The REPAIR statement issues warning messages for entries that may have truncated data.

To repair a damaged catalog, the version of SAS that you use must be able to update the catalog. Whether a SAS version can update a catalog (or just read it) is determined by the SAS version that created the catalog:

- A damaged Version 6 catalog can be repaired with Version 6 only.
- A damaged Version 8 catalog can be repaired with either Version 8 or Version 9, but not with Version 6.
- A damaged Version 9 catalog can be repaired with Version 9 only.

If the REPAIR operation is not successful, try to restore the SAS data set or catalog from your system's backup files.

If you issue a REPAIR statement for a SAS file that does not exist in the specified library, PROC DATASETS stops processing the run group that contains the REPAIR statement, and issues an error message. To override this behavior and continue processing, use the NOWARN option in the PROC DATASETS statement.

If you are using Cross-Environment Data Access (CEDA) to process a damaged foreign SAS data set, CEDA cannot repair it. CEDA does not support update processing, which is required in order to repair a damaged data set. To repair the foreign file, you must move it back to its native environment. Note that observations may be lost during the repair process. For more information about CEDA, see "Processing Data Using Cross-Environment Data Access" in *SAS Language Reference: Concepts*.

SAVE Statement

Deletes all the SAS files in a library except the ones listed in the SAVE statement.

Featured in: Example 2 on page 397

SAVE *SAS-file(s)* </ MEMTYPE=*mtype*>;

Required Arguments

SAS-file(s)
specifies one or more SAS files that you do not want to delete from the SAS data library.

Options

MEMTYPE=*mtype*
restricts processing to one member type. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

Aliases: MTYPE= and MT=

Default: If you do not specify the MEMTYPE= option in the PROC DATASETS statement or in the SAVE statement, the default is MEMTYPE=ALL.

See also: "Restricting Member Types for Processing" on page 378

Featured in: Example 2 on page 397

Details

- If one of the SAS files in *SAS-file* does not exist in the procedure input library, PROC DATASETS stops processing the RUN group containing the SAVE statement and issues an error message. To override this behavior, specify the NOWARN option in the PROC DATASETS statement.
- When the SAVE statement deletes SAS data sets, it also deletes any indexes associated with those data sets.

CAUTION:
SAS immediately deletes libraries and library members when you submit a RUN group. You are not asked to verify the delete operation before it begins. Because the SAVE statement deletes many SAS files in one operation, be sure that you

understand how the MEMTYPE= option affects which types of SAS files are saved and which types are deleted. △

- When you use the SAVE statement with generation groups, the SAVE statement treats the base version and all historical versions as a unit. You cannot save a specific version.

SELECT Statement

Selects SAS files for copying.

Restriction: Must follow a COPY statement

Restriction: Cannot appear with an EXCLUDE statement in the same COPY step

Featured in: Example 1 on page 392

SELECT *SAS-file(s)*
</ <ALTER=*alter-password*>
<MEMTYPE= *mtype*>>;

Required Arguments

SAS-file(s)
specifies one or more SAS files that you want to copy. All of the SAS files that you name must be in the data library that is referenced by the libref named in the IN= option in the COPY statement. If the SAS files have generation groups, the SELECT statement allows only selection of the base versions.

Options

ALTER=*alter-password*
provides the alter password for any alter-protected SAS files that you are moving from one data library to another. Because you are moving and thus deleting a SAS file from a SAS data library, you need alter access. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

See also: "Using Passwords with the DATASETS Procedure" on page 377

MEMTYPE=*mtype*
restricts processing to one member type. You can use the option either in parentheses after the name of each SAS file or after a forward slash.

Aliases: MTYPE= and MT=

Default: If you do not specify the MEMTYPE= option in the PROC DATASETS statement, in the COPY statement, or in the SELECT statement, the default is MEMTYPE=ALL.

See also: "Specifying Member Types When Copying or Moving SAS Files" on page 350

See also: "Restricting Member Types for Processing" on page 378

Featured in: Example 1 on page 392

Selecting Many Like-Named Files

You can use shortcuts for listing many SAS files in the SELECT statement. For more information, see "Shortcuts for Specifying Lists of Variable Names" on page 24.

Concepts: DATASETS Procedure

Procedure Execution

When you start the DATASETS procedure, you specify the procedure input library in the PROC DATASETS statement. If you omit a procedure input library, the procedure processes the current default SAS data library (usually the WORK data library). To specify a new procedure input library, issue the DATASETS procedure again.

Statements execute in the order they are written. For example, if you want to see the contents of a data set, copy a data set, and then visually compare the contents of the second data set with the first, the statements that perform those tasks must appear in that order (that is, CONTENTS, COPY, CONTENTS).

RUN-Group Processing

PROC DATASETS supports RUN-group processing. RUN-group processing enables you to submit RUN groups without ending the procedure.

The DATASETS procedure supports four types of RUN groups. Each RUN group is defined by the statements that compose it and by what causes it to execute.

Some statements in PROC DATASETS act as *implied* RUN statements because they cause the RUN group preceding them to execute.

The following list discusses what statements compose a RUN group and what causes each RUN group to execute:

- The PROC DATASETS statement always executes immediately. No other statement is necessary to cause the PROC DATASETS statement to execute. Therefore, the PROC DATASETS statement alone is a RUN group.
- The MODIFY statement, and any of its subordinate statements, form a RUN group. These RUN groups always execute immediately. No other statement is necessary to cause a MODIFY RUN group to execute.
- The APPEND, CONTENTS, and COPY statements (including EXCLUDE and SELECT, if present), form their own separate RUN groups. Every APPEND statement forms a single-statement RUN group; every CONTENTS statement forms a single-statement RUN group; and every COPY step forms a RUN group. Any other statement in the procedure, except those that are subordinate to either the COPY or MODIFY statement, causes the RUN group to execute.
- One or more of the following statements form a RUN group:
  - AGE
  - CHANGE
  - DELETE
  - EXCHANGE
  - REPAIR
  - SAVE

If any of these statements appear in sequence in the PROC step, the sequence forms a RUN group. For example, if a REPAIR statement appears immediately after a SAVE statement, the REPAIR statement does not force the SAVE statement to execute; it becomes part of the same RUN group. To execute the RUN group, submit one of the following statements:

- PROC DATASETS
- APPEND
- CONTENTS
- COPY
- MODIFY
- QUIT
- RUN
- another DATA or PROC step.

SAS reads the program statements that are associated with one task until it reaches a RUN statement or an implied RUN statement. It executes all of the preceding statements immediately, then continues reading until it reaches another RUN statement or implied RUN statement. To execute the last task, you must use a RUN statement or a statement that stops the procedure.

The following PROC DATASETS step contains five RUN groups:

```
libname dest 'SAS-data-library';
   /* RUN group */

proc datasets;
      /* RUN group */
   change nutr=fatg;
   delete bldtest;
   exchange xray=chest;
      /* RUN group */
   copy out=dest;
      select report;
      /* RUN group */
   modify bp;
      label dias='Taken at Noon';
      rename weight=bodyfat;
      /* RUN group */
   append base=tissue data=newtiss;
quit;
```

Note: If you are running in interactive line mode, you can receive messages that statements have already executed before you submit a RUN statement. Plan your tasks carefully if you are using this environment for running PROC DATASETS. △

Error Handling

Generally, if an error occurs in a statement, the RUN group containing the error does not execute. RUN groups preceding or following the one containing the error execute normally. The MODIFY RUN group is an exception. If a syntax error occurs in a statement subordinate to the MODIFY statement, only the statement containing the error fails. The other statements in the RUN group execute.

Note that if the first word of the statement (the statement name) is in error and the procedure cannot recognize it, the procedure treats the statement as part of the preceding RUN group.

Password Errors

If there is an error involving an incorrect or omitted password in a statement, the error affects only the statement containing the error. The other statements in the RUN group execute.

Forcing a RUN Group with Errors to Execute

The FORCE option in the PROC DATASETS statement forces execution of the RUN group even if one or more of the statements contain errors. Only the statements that are error-free execute.

Ending the Procedure

To stop the DATASETS procedure, you must issue a QUIT statement, a RUN CANCEL statement, a new PROC statement, or a DATA statement. Submitting a QUIT statement executes any statements that have not executed. Submitting a RUN CANCEL statement cancels any statements that have not executed.

Using Passwords with the DATASETS Procedure

Several statements in the DATASETS procedure support options that manipulate passwords on SAS files. These options, ALTER=, PW=, READ=, and WRITE=, are also data set options.* If you do not know how passwords affect SAS files, refer to *SAS Language Reference: Concepts*.

When you are working with password-protected SAS files in the AGE, CHANGE, DELETE, EXCHANGE, REPAIR, or SELECT statement, you can specify password options in the PROC DATASETS statement or in the subordinate statement.

Note: The ALTER= option works slightly different for the COPY (when moving a file) and MODIFY statements. Refer to "COPY Statement" on page 347 and "MODIFY Statement" on page 366. △

SAS searches for passwords in the following order:

1 in parentheses after the name of the SAS file in a subordinate statement. When used in parentheses, the option only refers to the name immediately preceding the option. If you are working with more than one SAS file in a data library and each SAS file has a different password, you must specify password options in parentheses after individual names.

   In the following statement, the ALTER= option provides the password RED for the SAS file BONES only:

   ```
   delete xplant bones(alter=red);
   ```

2 after a forward slash (/) in a subordinate statement. When you use a password option following a slash, the option refers to all SAS files named in the statement unless the same option appears in parentheses after the name of a SAS file. This

* In the APPEND and CONTENTS statements, you use these options just as you use any SAS data set option, in parentheses after the SAS data set name.

method is convenient when you are working with more than one SAS file and they all have the same password.

In the following statement, the ALTER= option in parentheses provides the password RED for the SAS file CHEST, and the ALTER= option after the slash provides the password BLUE for the SAS file VIRUS:

```
delete chest(alter=red) virus / alter=blue;
```

3 in the PROC DATASETS statement. Specifying the password in the PROC DATASETS statement can be useful if all the SAS files you are working with in the library have the same password. Do not specify the option in parentheses.

In the following PROC DATASETS step, the PW= option provides the password RED for the SAS files INSULIN and ABNEG:

```
proc datasets pw=red;
   delete insulin;
   contents data=abneg;
run;
```

Note: For the password for a SAS file in a SELECT statement, SAS looks in the COPY statement before it looks in the PROC DATASETS statement. △

Restricting Member Types for Processing

In the PROC DATASETS Statement

If you name a member type or several member types in the PROC DATASETS statement, in most subsequent statements (except the CONTENTS and COPY statements), you can name only a subset of the list of member types included in the PROC DATASETS statement. The directory listing that the PROC DATASETS statement writes to the SAS log includes only those SAS files of the type specified in the MEMTYPE= option.

In Subordinate Statements

Use the MEMTYPE= option in the following subordinate statements to limit the member types that are available for processing:

AGE

CHANGE

DELETE

EXCHANGE

EXCLUDE

REPAIR

SAVE

SELECT

Note: The MEMTYPE= option works slightly differently for the CONTENTS, COPY, and MODIFY statements. Refer to “CONTENTS Statement” on page 344, “COPY Statement” on page 347, and “MODIFY Statement” on page 366 for more information. △

The procedure searches for MEMTYPE= in the following order:

1 in parentheses immediately after the name of a SAS file. When used in parentheses, the MEMTYPE= option refers only to the SAS file immediately preceding the option. For example, the following statement deletes HOUSE.DATA, LOT.CATALOG, and SALES.DATA because the default member type for the DELETE statement is DATA. (Refer to Table 15.3 on page 380 for the default types for each statement.)

```
delete house lot(memtype=catalog) sales;
```

2 after a slash (/) at the end of the statement. When used following a slash, the MEMTYPE= option refers to all SAS files named in the statement unless the option appears in parentheses after the name of a SAS file. For example, the following statement deletes LOTPIX.CATALOG, REGIONS.DATA, and APPL.CATALOG:

```
delete lotpix regions(memtype=data) appl / memtype=catalog;
```

3 in the PROC DATASETS statement. For example, this DATASETS procedure deletes APPL.CATALOG:

```
proc datasets memtype=catalog;
   delete appl;
run;
```

Note: When you use the EXCLUDE and SELECT statements, the procedure looks in the COPY statement for the MEMTYPE= option before it looks in the PROC DATASETS statement. For more information, see "Specifying Member Types When Copying or Moving SAS Files" on page 350. △

4 for the default value. If you do not specify a MEMTYPE= option in the subordinate statement or in the PROC DATASETS statement, the default value for the subordinate statement determines the member type available for processing.

Member Types

The following list gives the possible values for the MEMTYPE= option:

ACCESS
access descriptor files (created by SAS/ACCESS software)

ALL
all member types

CATALOG
SAS catalogs

DATA
SAS data files

FDB
financial database

MDDB
multidimensional database

PROGRAM
stored compiled SAS programs

VIEW
SAS views

Table 15.3 on page 380 shows the member types that you can use in each statement:

Table 15.3 Subordinate Statements and Appropriate Member Types

Statement	Appropriate member types	Default member type
AGE	ACCESS, CATALOG, DATA, FDB, MDDB, PROGRAM, VIEW	DATA
CHANGE	ACCESS, ALL, CATALOG, DATA, FDB, MDDB, PROGRAM, VIEW	ALL
CONTENTS	ALL, DATA, VIEW	DATA[1]
COPY	ACCESS, ALL, CATALOG, DATA, FDB, MDDB, PROGRAM, VIEW	ALL
DELETE	ACCESS, ALL, CATALOG, DATA, FDB, MDDB, PROGRAM, VIEW	DATA
EXCHANGE	ACCESS, ALL, CATALOG, DATA, FDB, MDDB, PROGRAM, VIEW	ALL
EXCLUDE	ACCESS, ALL, CATALOG, DATA, FDB, MDDB, PROGRAM, VIEW	ALL
MODIFY	ACCESS, DATA, VIEW	DATA
REPAIR	ALL, CATALOG, DATA	ALL[2]
SAVE	ACCESS, ALL, CATALOG, DATA, FDB, MDDB, PROGRAM, VIEW	ALL
SELECT	ACCESS, ALL, CATALOG, DATA, FDB, MDDB, PROGRAM, VIEW	ALL

1 When DATA=_ALL_ in the CONTENTS statement, the default is ALL. ALL includes only DATA and VIEW.
2 ALL includes only DATA and CATALOG.

Restricting Processing for Generation Data Sets

Several statements in the DATASETS procedure support the GENNUM= option to restrict processing for generation data sets. GENNUM= is also a data set option.* If you do not know how to request and use generation data sets, see "Generation Data Sets" in *SAS Language Reference: Concepts*.

When you are working with a generation group for the AUDIT, CHANGE, DELETE, MODIFY, and REPAIR statements, you can restrict processing in the PROC DATASETS statement or in the subordinate statement to a specific version.

Note: The GENNUM= option works slightly different for the MODIFY statement. See "MODIFY Statement" on page 366. △

Note: You cannot restrict processing to a specific version for the AGE, COPY, EXCHANGE, and SAVE statements. These statements apply to the entire generation group. △

SAS searches for a generation specification in the following order:

1 in parentheses after the name of the SAS data set in a subordinate statement. When used in parentheses, the option only refers to the name immediately

* For the APPEND and CONTENTS statements, use GENNUM= just as you use any SAS data set option, in parentheses after the SAS data set name.

preceding the option. If you are working with more than one SAS data set in a data library and you want a different generation version for each SAS data set, you must specify GENNUM= in parentheses after individual names.

In the following statement, the GENNUM= option specifies the version of a generation group for the SAS data set BONES only:

```
delete xplant bones (gennum=2);
```

2 after a forward slash (/) in a subordinate statement. When you use the GENNUM= option following a slash, the option refers to all SAS data sets named in the statement unless the same option appears in parentheses after the name of a SAS data set. This method is convenient when you are working with more than one file and you want the same version for all files.

In the following statement, the GENNUM= option in parentheses specifies the generation version for SAS data set CHEST, and the GENNUM= option after the slash specifies the generation version for SAS data set VIRUS:

```
delete chest (gennum=2) virus / gennum=1;
```

3 in the PROC DATASETS statement. Specifying the generation version in the PROC DATASETS statement can be useful if you want the same version for all of the SAS data sets you are working with in the library. Do not specify the option in parentheses.

In the following PROC DATASETS step, the GENNUM= option specifies the generation version for the SAS files INSULIN and ABNEG:

```
proc datasets gennum=2;
   delete insulin;
   contents data=abneg;
run;
```

Note: For the generation version for a SAS file in a SELECT statement, SAS looks in the COPY statement before it looks in the PROC DATASETS statement. △

Results: DATASETS Procedure

Directory Listing to the SAS Log

The PROC DATASETS statement lists the SAS files in the procedure input library unless the NOLIST option is specified. The NOLIST option prevents the creation of the procedure results that go to the log. If you specify the MEMTYPE= option, only specified types are listed. If you specify the DETAILS option, PROC DATASETS prints these additional columns of information: **`Obs, Entries or Indexes`**, **`Vars`**, and **`Label`**.

Directory Listing as SAS Output

The CONTENTS statement lists the directory of the procedure input library if you use the DIRECTORY option or specify DATA=_ALL_.

If you want only a directory, use the NODS option and the _ALL_ keyword in the DATA= option. The NODS option suppresses the description of the SAS data sets; only the directory appears in the output.

Note: The CONTENTS statement does not put a directory in an output data set. If you try to create an output data set using the NODS option, you receive an empty output data set. Use the SQL procedure to create a SAS data set that contains information about a SAS data library. △

Note: If you specify the ODS RTF destination, the PROC DATASETS output will go to both the SAS log and the ODS output area. The NOLIST option will suppress output to both. To see the output only in the SAS log, use the ODS EXCLUDE statement by specifying the member directory as the exclusion. △

PROC DATASETS and the Output Delivery System (ODS)

Most SAS procedures send their messages to the SAS log and their procedure results to the listing. PROC DATASETS is unique because it sends procedure results to both the SAS log and the listing. When the interface to ODS was created, it was decided that all procedure results (from both the log and the listing) should be available to the new ODS destination. In order to implement this feature and maintain compatibility with earlier releases, the interface to ODS had to be slightly different from the usual interface.

By default, the PROC DATASETS statement itself produces two output objects: Members and Directory. These objects are routed to the log. The CONTENTS statement produces three output objects by default: Attributes, EngineHost, and Variables. (The use of various options adds other output objects.) These objects are routed to the listing. If you open one of the new ODS destinations (like HTML, RTF, or PRINTER), all of these objects are, by default, routed to that destination.

You can use ODS SELECT and ODS EXCLUDE statements to control which objects go to which destination, just as you can for any other procedure. However, because of the unique interface between PROC DATASETS and ODS, when you use the keyword LISTING in an ODS SELECT or ODS EXCLUDE statement, you affect both the log and the listing.

Procedure Output

The only statement in PROC DATASETS that produces procedure output is the CONTENTS statement. This section shows the output from the CONTENTS statement for the GROUP data set, which is shown in Output 15.2 on page 383.

Only the items in the output that require explanation are discussed.

Data Set Attributes

Here are descriptions of selected fields shown in Output 15.2 on page 383:

`Member Type`
: is the type of library member (DATA or VIEW).

`Protection`
: indicates whether the SAS data set is READ, WRITE, or ALTER password protected.

`Data Set Type`
: names the special data set type (such as CORR, COV, SSPC, EST, or FACTOR), if any.

Observations
is the total number of observations currently in the file. Note that for a very large data set, if the number of observations exceeds the number that can be stored in a double-precision integer, the count will show as missing.

Deleted Observations
is the number of observations marked for deletion. These observations are not included in the total number of observations, shown in the **Observations** field. Note that for a very large data set, if the number of deleted observations exceeds the number that can be stored in a double-precision integer, the count will show as missing.

Compressed
indicates whether the data set is compressed. If the data set is compressed, the output includes an additional item, **Reuse Space** (with a value of YES or NO), that indicates whether to reuse space that is made available when observations are deleted.

Sorted
indicates whether the data set is sorted. If you sort the data set with PROC SORT, PROC SQL, or specify sort information with the SORTEDBY= data set option, a value of YES appears here, and there is an additional section to the output. See "Sort Information" on page 385 for details.

Data Representation
is the format in which data is represented on a computer architecture or in an operating environment. For example, on an IBM PC, character data is represented by its ASCII encoding and byte-swapped integers. Native data representation refers to an environment for which the data representation compares with the CPU that is accessing the file. For example, a file that is in Windows data representation is native to the Windows operating environment.

Encoding
is the encoding value. Encoding is a set of characters (letters, logograms, digits, punctuation, symbols, control characters, and so on) that have been mapped to numeric values (called code points) that can be used by computers. The code points are assigned to the characters in the character set when you apply an encoding method.

Output 15.2 Data Set Attributes for the GROUP Data Set

```
                                The SAS System                              1

                            The DATASETS Procedure

Data Set Name        HEALTH.GROUP                      Observations          148
Member Type          DATA                              Variables             11
Engine               V9                                Indexes               1
Created              8:06 Tuesday, January 29, 2002    Observation Length    96
Last Modified        9:13 Tuesday, January 29, 2002    Deleted Observations  0
Protection           READ                              Compressed            NO
Data Set Type                                          Sorted                YES
Label                Test Subjects
Data Representation  WINDOWS
Encoding             wlatin1  Western (Windows)
```

Engine and Operating Environment-Dependent Information

The CONTENTS statement produces operating environment-specific and engine-specific information. This information differs depending on the operating environment. The following output is from the Windows operating environment.

Output 15.3 Engine and Operating Environment Dependent Information Section of CONTENTS Output

```
                        Engine/Host Dependent Information

Data Set Page Size          8192
Number of Data Set Pages    4
First Data Page             1
Max Obs per Page            84
Obs in First Data Page      62
Index File Page Size        4096
Number of Index File Pages  2
Number of Data Set Repairs  0
File Name                   c:\Myfiles\health\group.sas7bdat
Release Created             9.0000A0
Host Created                WIN_NT
```

Alphabetic List of Variables and Attributes

Here are descriptions of selected columns in Output 15.4 on page 384:

#
: is the logical position of each variable in the observation. This is the number that is assigned to the variable when it is defined.

Variable
: is the name of each variable. By default, variables appear alphabetically.

Note: Variable names are sorted such that X1, X2, and X10 appear in that order and not in the true collating sequence of X1, X10, and X2. Variable names that contain an underscore and digits may appear in a nonstandard sort order. For example, P25 and P75 appear before P2_5. △

Type
: specifies the type of variable: character or numeric.

Note: If none of the variables in the SAS data set has a format, informat, or label associated with it, the column for that attribute does not appear. △

Output 15.4 Variable Attributes Section

```
                    Alphabetic List of Variables and Attributes

 #   Variable   Type   Len   Format    Informat   Label

 9   BIRTH      Num      8   DATE7.    DATE7.
 4   CITY       Char    15   $.        $.
 3   FNAME      Char    15   $.        $.
10   HIRED      Num      8   DATE7.    DATE7.
11   HPHONE     Char    12   $.        $.
 1   IDNUM      Char     4   $.        $.
 7   JOBCODE    Char     3   $.        $.
 2   LNAME      Char    15   $.        $.
 8   SALARY     Num      8   COMMA8.              current salary excluding bonus
 6   SEX        Char     1   $.        $.
 5   STATE      Char     2   $.        $.
```

Alphabetic List of Indexes and Attributes

The section shown in Output 15.5 on page 385 appears only if the data set has indexes associated with it.

#
: indicates the number of each index. The indexes are numbered sequentially as they are defined.

Index
: displays the name of each index. For simple indexes, the name of the index is the same as a variable in the data set.

Unique Option
: indicates whether the index must have unique values. If the column contains YES, the combination of values of the index variables is unique for each observation.

Nomiss Option
: indicates whether the index excludes missing values for all index variables. If the column contains YES, the index does not contain observations with missing values for all index variables.

# of Unique Values
: gives the number of unique values in the index.

Variables
: names the variables in a composite index.

Output 15.5 Index Attributes Section

```
                Alphabetic List of Indexes and Attributes

                                              # of
                    Unique    NoMiss    Unique
         #    Index    Option    Option    Values    Variables

         1    vital    YES       YES          148    BIRTH SALARY
```

Sort Information

The section shown in Output 15.6 on page 386 appears only if the **Sorted** field has a value of YES.

Sortedby
: indicates how the data are currently sorted. This field contains either the variables and options you use in the BY statement in PROC SORT, the column name in PROC SQL, or the values you specify in the SORTEDBY= option.

Validated
: indicates whether PROC SORT or PROC SQL sorted the data. If PROC SORT or PROC SQL sorted the data set, the value is YES. If you assigned the sort information with the SORTEDBY= data set option, the value is NO.

Character Set
: is the character set used to sort the data. The value for this field can be ASCII, EBCDIC, or PASCII.

Collating Sequence
: is the collating sequence used to sort the data set. This field does not appear if you do not specify a specific collating sequence that is different from the character set. (not shown)

Sort Option
: indicates whether PROC SORT used the NODUPKEY or NODUPREC option when sorting the data set. This field does not appear if you did not use one of these options in a PROC SORT statement. (not shown)

Output 15.6 Sort Information Section

```
                              The SAS System                              2

                          The DATASETS Procedure

                             Sort Information

                           Sortedby       LNAME
                           Validated      NO
                           Character Set  ANSI
```

Output Data Sets

The CONTENTS statement is the only statement in the DATASETS procedure that generates output data sets.

The OUT= Data Set

The OUT= option in the CONTENTS statement creates an output data set. Each variable in each DATA= data set has one observation in the OUT= data set. These are the variables in the output data set:

CHARSET
: the character set used to sort the data set. The value is ASCII, EBCDIC, or PASCII. A blank appears if the data set does not have sort information stored with it.

COLLATE
: the collating sequence used to sort the data set. A blank appears if the sort information for the input data set does not include a collating sequence.

COMPRESS
: indicates whether the data set is compressed.

CRDATE
: date the data set was created.

DELOBS
: number of observations marked for deletion in the data set. (Observations can be marked for deletion but not actually deleted when you use the FSEDIT procedure of SAS/FSP software.)

ENCRYPT
: indicates whether the data set is encrypted.

ENGINE
: name of the method used to read from and write to the data set.

FLAGS
: indicates whether an SQL view is protected (**P**) or contributes (**C**) to a derived variable.

P indicates the variable is protected. The value of the variable can be displayed but not updated.

C indicates whether the variable contributes to a derived variable.

The value of FLAG is blank if **P** or **C** does not apply to an SQL view or if it is a data set view.

FORMAT
: variable format. The value of FORMAT is a blank if you do not associate a format with the variable.

FORMATD
: number of decimals you specify when you associate the format with the variable. The value of FORMATD is 0 if you do not specify decimals in the format.

FORMATL
: format length. If you specify a length for the format when you associate the format with a variable, the length you specify is the value of FORMATL. If you do not specify a length for the format when you associate the format with a variable, the value of FORMATL is the default length of the format if you use the FMTLEN option and 0 if you do not use the FMTLEN option.

GENMAX
: maximum number of versions for the generation group.

GENNEXT
: the next generation number for a generation group.

GENNUM
: the version number.

IDXCOUNT
: number of indexes for the data set.

IDXUSAGE
: use of the variable in indexes. Possible values are

NONE
: the variable is not part of an index.

SIMPLE
: the variable has a simple index. No other variables are included in the index.

COMPOSITE
: the variable is part of a composite index.

BOTH
: the variable has a simple index and is part of a composite index.

INFORMAT
: variable informat. The value is a blank if you do not associate an informat with the variable.

INFORMD
: number of decimals you specify when you associate the informat with the variable. The value is 0 if you do not specify decimals when you associate the informat with the variable.

INFORML

informat length. If you specify a length for the informat when you associate the informat with a variable, the length you specify is the value of INFORML. If you do not specify a length for the informat when you associate the informat with a variable, the value of INFORML is the default length of the informat if you use the FMTLEN option and 0 if you do not use the FMTLEN option.

JUST
: justification (0=left, 1=right).

LABEL
: variable label (blank if none given).

LENGTH
: variable length.

LIBNAME
: libref used for the data library.

MEMLABEL
: label for this SAS data set (blank if no label).

MEMNAME
: SAS data set that contains the variable.

MEMTYPE
: library member type (DATA or VIEW).

MODATE
: date the data set was last modified.

NAME
: variable name.

NOBS
: number of observations in the data set.

NODUPKEY
: indicates whether the NODUPKEY option was used in a PROC SORT statement to sort the input data set.

NODUPREC
: indicates whether the NODUPREC option was used in a PROC SORT statement to sort the input data set.

NPOS
: physical position of the first character of the variable in the data set.

POINTOBS
: indicates if the data set can be addressed by observation.

PROTECT
: the first letter of the level of protection. The value for PROTECT is one or more of the following:

A	indicates the data set is alter-protected.
R	indicates the data set is read-protected.
W	indicates the data set is write-protected.

REUSE
: indicates whether the space made available when observations are deleted from a compressed data set should be reused. If the data set is not compressed, the REUSE variable has a value of NO.

SORTED
the value depends on the sorting characteristics of the input data set. Possible values are

. (period)	for not sorted.
0	for sorted but not validated.
1	for sorted and validated.

SORTEDBY
the value depends on that variable's role in the sort. Possible values are

. (period)
if the variable was not used to sort the input data set.

n
where *n* is an integer that denotes the position of that variable in the sort. A negative value of *n* indicates that the data set is sorted by the descending order of that variable.

TYPE
type of the variable (1=numeric, 2=character).

TYPEMEM
special data set type (blank if no TYPE= value is specified).

VARNUM
variable number in the data set. Variables are numbered in the order they appear.

The output data set is sorted by the variables LIBNAME and MEMNAME.

Note: The variable names are sorted so that the values X1, X2, and X10 are listed in that order, not in the true collating sequence of X1, X10, X2. Therefore, if you want to use a BY statement on MEMNAME in subsequent steps, run a PROC SORT step on the output data set first or use the NOTSORTED option in the BY statement. △

Output 15.7 on page 389 is an example of an output data set created from the GROUP data set, which is shown in Example 4 on page 400 and in "Procedure Output" on page 382.

Output 15.7 The Data Set HEALTH.GRPOUT

```
                         An Example of an Output Data Set                       1

OBS LIBNAME  MEMNAME    MEMLABEL     TYPEMEM  NAME     TYPE  LENGTH  VARNUM

  1 HEALTH    GROUP   Test Subjects           BIRTH      1      8       9
  2 HEALTH    GROUP   Test Subjects           CITY       2     15       4
  3 HEALTH    GROUP   Test Subjects           FNAME      2     15       3
  4 HEALTH    GROUP   Test Subjects           HIRED      1      8      10
  5 HEALTH    GROUP   Test Subjects           HPHONE     2     12      11
  6 HEALTH    GROUP   Test Subjects           IDNUM      2      4       1
  7 HEALTH    GROUP   Test Subjects           JOBCODE    2      3       7
  8 HEALTH    GROUP   Test Subjects           LNAME      2     15       2
  9 HEALTH    GROUP   Test Subjects           SALARY     1      8       8
 10 HEALTH    GROUP   Test Subjects           SEX        2      1       6
 11 HEALTH    GROUP   Test Subjects           STATE      2      2       5

OBS             LABEL                FORMAT  FORMATL  FORMATD  INFORMAT  INFORML

  1                                  DATE       7        0       DATE       7
  2                                  $          0        0       $          0
  3                                  $          0        0       $          0
  4                                  DATE       7        0       DATE       7
  5                                  $          0        0       $          0
  6                                  $          0        0       $          0
  7                                  $          0        0       $          0
  8                                  $          0        0       $          0
  9 current salary excluding bonus   COMMA      8        0                  0
 10                                  $          0        0       $          0
 11                                  $          0        0       $          0
```

```
                    An Example of an Output Data Set                       2

Obs INFORMD JUST NPOS NOBS ENGINE           CRDATE           MODATE DELOBS

  1    0      1    8   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
  2    0      0   58   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
  3    0      0   43   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
  4    0      1   16   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
  5    0      0   79   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
  6    0      0   24   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
  7    0      0   76   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
  8    0      0   28   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
  9    0      1    0   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
 10    0      0   75   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0
 11    0      0   73   148   V9   29JAN02:08:06:46 29JAN02:09:13:36    0

OBS IDXUSAGE  MEMTYPE IDXCOUNT PROTECT FLAGS COMPRESS REUSE SORTED SORTEDBY

  1 COMPOSITE  DATA       1      R--    ---     NO     NO      0       .
  2 NONE       DATA       1      R--    ---     NO     NO      0       .
  3 NONE       DATA       1      R--    ---     NO     NO      0       .
  4 NONE       DATA       1      R--    ---     NO     NO      0       .
  5 NONE       DATA       1      R--    ---     NO     NO      0       .
  6 NONE       DATA       1      R--    ---     NO     NO      0       .
  7 NONE       DATA       1      R--    ---     NO     NO      0       .
  8 NONE       DATA       1      R--    ---     NO     NO      0       1
  9 COMPOSITE  DATA       1      R--    ---     NO     NO      0       .
 10 NONE       DATA       1      R--    ---     NO     NO      0       .
 11 NONE       DATA       1      R--    ---     NO     NO      0       .
```

An Example of an Output Data Set 3

OBS	CHARSET	COLLATE	NODUPKEY	NODUPREC	ENCRYPT	POINTOBS	GENMAX	GENNUM	GENNEXT
1	ASCII		NO	NO	NO	YES	0	.	0
2	ASCII		NO	NO	NO	YES	0	.	0
3	ASCII		NO	NO	NO	YES	0	.	0
4	ASCII		NO	NO	NO	YES	0	.	0
5	ASCII		NO	NO	NO	YES	0	.	0
6	ASCII		NO	NO	NO	YES	0	.	0
7	ASCII		NO	NO	NO	YES	0	.	0
8	ASCII		NO	NO	NO	YES	0	.	0
9	ASCII		NO	NO	NO	YES	0	.	0
10	ASCII		NO	NO	NO	YES	0	.	0
11	ASCII		NO	NO	NO	YES	0	.	0

The OUT2= Data Set

The OUT2= option in the CONTENTS statement creates an output data set that contains information about indexes and integrity constraints. These are the variables in the output data set:

IC_OWN
: contains YES if the index is owned by the integrity constraint.

INACTIVE
: contains YES if the integrity constraint is inactive.

LIBNAME
: libref used for the data library.

MEMNAME
: SAS data set that contains the variable.

MG
: the value of MESSAGE=, if it is used, in the IC CREATE statement.

MSGTYPE
: the value will be blank unless an integrity constraint is violated and you specified a message.

NAME
: the name of the index or integrity constraint.

NOMISS
: contains YES if the NOMISS option is defined for the index.

NUMVALS
: the number of distinct values in the index (displayed for centiles).

NUMVARS
: the number of variables involved in the index or integrity constraint.

ONDELETE
: for a foreign key integrity constraint, contains RESTRICT or SET NULL if applicable (the ON DELETE option in the IC CREATE statement).

ONUPDATE
: for a foreign key integrity constraint, contains RESTRICT or SET NULL if applicable (the ON UPDATE option in the IC CREATE statement).

RECREATE
: the SAS statement necessary to recreate the index or integrity constraint.

REFERENCE

for a foreign key integrity constraint, contains the name of the referenced data set.

TYPE
: the type. For an index, the value is “Index” while for an integrity constraint, the value is the type of integrity constraint (Not Null, Check, Primary Key, etc.).

UNIQUE
: contains YES if the UNIQUE option is defined for the index.

UPERC
: the percentage of the index that has been updated since the last refresh (displayed for centiles).

UPERCMX
: the percentage of the index update that triggers a refresh (displayed for centiles).

WHERE
: for a check integrity constraint, contains the WHERE statement.

Examples: DATASETS Procedure

Example 1: Manipulating SAS Files

Procedure features:

PROC DATASETS statement options:
- DETAILS
- LIBRARY=

CHANGE statement

COPY statement options:
- MEMTYPE
- MOVE
- OUT=

DELETE statement option:
- MEMTYPE=

EXCHANGE statement

EXCLUDE statement

SELECT statement option:
- MEMTYPE=

This example
- changes the names of SAS files
- copies SAS files between SAS data libraries
- deletes SAS files
- selects SAS files to copy
- exchanges the names of SAS files
- excludes SAS files from a copy operation.

Program

Write the programming statements to the SAS log. The SOURCE system option accomplishes this.

```
options pagesize=60 linesize=80 nodate pageno=1 source;
```

```
libname dest1 'SAS-data-library-1';
libname dest2 'SAS-data-library-2';
libname health 'SAS-data-library-3';
```

Specify the procedure input library, and add more details to the directory. DETAILS prints these additional columns in the directory: **`Obs, Entries or Indexes`**, **`Vars`**, and **`Label`**. All member types are available for processing because the MEMTYPE= option does not appear in the PROC DATASETS statement.

```
proc datasets library=health details;
```

Delete two files in the library, and modify the names of a SAS data set and a catalog. The DELETE statement deletes the TENSION data set and the A2 catalog. MT=CATALOG applies only to A2 and is necessary because the default member type for the DELETE statement is DATA. The CHANGE statement changes the name of the A1 catalog to POSTDRUG. The EXCHANGE statement exchanges the names of the WEIGHT and BODYFAT data sets. MEMTYPE= is not necessary in the CHANGE or EXCHANGE statement because the default is MEMTYPE=ALL for each statement.

```
   delete tension a2(mt=catalog);
   change a1=postdrug;
   exchange weight=bodyfat;
```

Restrict processing to one member type and delete and move data views. MEMTYPE=VIEW restricts processing to SAS data views. MOVE specifies that all SAS data views named in the SELECT statements in this step be deleted from the HEALTH data library and moved to the DEST1 data library.

```
   copy out=dest1 move memtype=view;
```

Move the SAS data view SPDATA from the HEALTH data library to the DEST1 data library.

```
      select spdata;
```

Move the catalogs to another data library. The SELECT statement specifies that the catalogs ETEST1 through ETEST5 be moved from the HEALTH data library to the DEST1 data library. MEMTYPE=CATALOG overrides the MEMTYPE=VIEW option in the COPY statement.

```
      select etest1-etest5 / memtype=catalog;
```

Exclude all files with a specified criteria from processing. The EXCLUDE statement excludes from the COPY operation all SAS files that begin with the letter D and the other SAS files listed. All remaining SAS files in the HEALTH data library are copied to the DEST2 data library.

```
   copy out=dest2;
      exclude d: mlscl oxygen test2 vision weight;
quit;
```

SAS Log

```
1    options pagesize=60 linesize=80 nodate pageno=1 source;
2    libname dest1 'c:\Myfiles\dest1';
NOTE: Libref DEST1 was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\Myfiles\dest1
3    libname dest2 'c:\Myfiles\dest2';
NOTE: Libref DEST2 was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\Myfiles\dest2
4    libname health 'c:\Myfiles\health';
NOTE: Libref HEALTH was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\Myfiles\health
5    proc datasets library=health details;
                                   Directory

              Libref         HEALTH
              Engine         V9
              Physical Name  c:\Myfiles\health
              File Name      c:\Myfiles\health


              Member   Obs, Entries                    File
 #  Name      Type      or Indexes   Vars  Label       Size  Last Modified

 1  A1        CATALOG       23                        62464   19FEB2002:14:41:15
 2  A2        CATALOG        1                        17408   19FEB2002:14:41:15
 3  ALL       DATA          23         17             13312   19FEB2002:14:41:19
 4  BODYFAT   DATA           1          2              5120   19FEB2002:14:41:19
 5  CONFOUND  DATA           8          4              5120   19FEB2002:14:41:19
 6  CORONARY  DATA          39          4              5120   19FEB2002:14:41:20
 7  DRUG1     DATA           6          2  JAN95       5120   19FEB2002:14:41:20
                                           Data
 8  DRUG2     DATA          13          2  MAY95       5120   19FEB2002:14:41:20
                                           Data
 9  DRUG3     DATA          11          2  JUL95       5120   19FEB2002:14:41:20
                                           Data
10  DRUG4     DATA           7          2  JAN92       5120   19FEB2002:14:41:20
                                           Data
11  DRUG5     DATA           1          2  JUL92       5120   19FEB2002:14:41:20
                                           Data
12  ETEST1    CATALOG        1                        17408   19FEB2002:14:41:20
13  ETEST2    CATALOG        1                        17408   19FEB2002:14:41:20
14  ETEST3    CATALOG        1                        17408   19FEB2002:14:41:20
15  ETEST4    CATALOG        1                        17408   19FEB2002:14:41:20
16  ETEST5    CATALOG        1                        17408   19FEB2002:14:41:20
17  ETESTS    CATALOG        1                        17408   19FEB2002:14:41:21
18  FORMATS   CATALOG        6                        17408   19FEB2002:14:41:21
19  GROUP     DATA         148         11             25600   19FEB2002:14:41:21
20  INFANT    DATA         149          6             17408   05FEB2002:12:52:30
21  MLSCL     DATA          32          4  Multiple    5120   19FEB2002:14:41:21
                                           Sclerosi
                                           s Data
22  NAMES     DATA           7          4              5120   19FEB2002:14:41:21
23  OXYGEN    DATA          31          7              9216   19FEB2002:14:41:21
24  PERSONL   DATA         148         11             25600   19FEB2002:14:41:21
25  PHARM     DATA           6          3  Sugar       5120   19FEB2002:14:41:21
                                           Study
26  POINTS    DATA           6          6              5120   19FEB2002:14:41:21
27  PRENAT    DATA         149          6             17408   19FEB2002:14:41:22
28  RESULTS   DATA          10          5              5120   19FEB2002:14:41:22
29  SLEEP     DATA         108          6              9216   19FEB2002:14:41:22
30  SPDATA    VIEW           .          2              5120   19FEB2002:14:41:29
31  SYNDROME  DATA          46          8              9216   19FEB2002:14:41:22
32  TENSION   DATA           4          3              5120   19FEB2002:14:41:22
33  TEST2     DATA          15          5              5120   19FEB2002:14:41:22
34  TRAIN     DATA           7          2              5120   19FEB2002:14:41:22
35  VISION    DATA          16          3              5120   19FEB2002:14:41:22
36  WEIGHT    DATA          83         13  Californ   13312   19FEB2002:14:41:22
                                           ia
                                           Results
37  WGHT      DATA          83         13  Californ   13312   19FEB2002:14:41:23
                                           ia
                                           Results
```

```
6       delete tension a2(mt=catalog);
7       change a1=postdrug;
8       exchange weight=bodyfat;
NOTE: Deleting HEALTH.TENSION (memtype=DATA).
NOTE: Deleting HEALTH.A2 (memtype=CATALOG).
NOTE: Changing the name HEALTH.A1 to HEALTH.POSTDRUG (memtype=CATALOG).
NOTE: Exchanging the names HEALTH.WEIGHT and HEALTH.BODYFAT (memtype=DATA).
9       copy out=dest1 move memtype=view;
10         select spdata;
11         select etest1-etest5 / memtype=catalog;
NOTE: Moving HEALTH.SPDATA to DEST1.SPDATA (memtype=VIEW).
NOTE: Moving HEALTH.ETEST1 to DEST1.ETEST1 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST2 to DEST1.ETEST2 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST3 to DEST1.ETEST3 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST4 to DEST1.ETEST4 (memtype=CATALOG).
NOTE: Moving HEALTH.ETEST5 to DEST1.ETEST5 (memtype=CATALOG).
12      copy out=dest2;
13         exclude d: mlscl oxygen test2 vision weight;
14   quit;

NOTE: Copying HEALTH.ALL to DEST2.ALL (memtype=DATA).
NOTE: There were 23 observations read from the data set HEALTH.ALL.
NOTE: The data set DEST2.ALL has 23 observations and 17 variables.
NOTE: Copying HEALTH.BODYFAT to DEST2.BODYFAT (memtype=DATA).
NOTE: There were 83 observations read from the data set HEALTH.BODYFAT.
NOTE: The data set DEST2.BODYFAT has 83 observations and 13 variables.
NOTE: Copying HEALTH.CONFOUND to DEST2.CONFOUND (memtype=DATA).
NOTE: There were 8 observations read from the data set HEALTH.CONFOUND.
NOTE: The data set DEST2.CONFOUND has 8 observations and 4 variables.
NOTE: Copying HEALTH.CORONARY to DEST2.CORONARY (memtype=DATA).
NOTE: There were 39 observations read from the data set HEALTH.CORONARY.
NOTE: The data set DEST2.CORONARY has 39 observations and 4 variables.
NOTE: Copying HEALTH.ETESTS to DEST2.ETESTS (memtype=CATALOG).
NOTE: Copying HEALTH.FORMATS to DEST2.FORMATS (memtype=CATALOG).
NOTE: Copying HEALTH.GROUP to DEST2.GROUP (memtype=DATA).
NOTE: There were 148 observations read from the data set HEALTH.GROUP.
NOTE: The data set DEST2.GROUP has 148 observations and 11 variables.
NOTE: Copying HEALTH.INFANT to DEST2.INFANT (memtype=DATA).
NOTE: There were 149 observations read from the data set HEALTH.INFANT.
NOTE: The data set DEST2.INFANT has 149 observations and 6 variables.
NOTE: Copying HEALTH.NAMES to DEST2.NAMES (memtype=DATA).
NOTE: There were 7 observations read from the data set HEALTH.NAMES.
NOTE: The data set DEST2.NAMES has 7 observations and 4 variables.
NOTE: Copying HEALTH.PERSONL to DEST2.PERSONL (memtype=DATA).
NOTE: There were 148 observations read from the data set HEALTH.PERSONL.
NOTE: The data set DEST2.PERSONL has 148 observations and 11 variables.
NOTE: Copying HEALTH.PHARM to DEST2.PHARM (memtype=DATA).
NOTE: There were 6 observations read from the data set HEALTH.PHARM.
NOTE: The data set DEST2.PHARM has 6 observations and 3 variables.
NOTE: Copying HEALTH.POINTS to DEST2.POINTS (memtype=DATA).
NOTE: There were 6 observations read from the data set HEALTH.POINTS.
NOTE: The data set DEST2.POINTS has 6 observations and 6 variables.
NOTE: Copying HEALTH.POSTDRUG to DEST2.POSTDRUG (memtype=CATALOG).
NOTE: Copying HEALTH.PRENAT to DEST2.PRENAT (memtype=DATA).
NOTE: There were 149 observations read from the data set HEALTH.PRENAT.
NOTE: The data set DEST2.PRENAT has 149 observations and 6 variables.
NOTE: Copying HEALTH.RESULTS to DEST2.RESULTS (memtype=DATA).
NOTE: There were 10 observations read from the data set HEALTH.RESULTS.
NOTE: The data set DEST2.RESULTS has 10 observations and 5 variables.
NOTE: Copying HEALTH.SLEEP to DEST2.SLEEP (memtype=DATA).
NOTE: There were 108 observations read from the data set HEALTH.SLEEP.
NOTE: The data set DEST2.SLEEP has 108 observations and 6 variables.
NOTE: Copying HEALTH.SYNDROME to DEST2.SYNDROME (memtype=DATA).
NOTE: There were 46 observations read from the data set HEALTH.SYNDROME.
NOTE: The data set DEST2.SYNDROME has 46 observations and 8 variables.
NOTE: Copying HEALTH.TRAIN to DEST2.TRAIN (memtype=DATA).
NOTE: There were 7 observations read from the data set HEALTH.TRAIN.
NOTE: The data set DEST2.TRAIN has 7 observations and 2 variables.
NOTE: Copying HEALTH.WGHT to DEST2.WGHT (memtype=DATA).
NOTE: There were 83 observations read from the data set HEALTH.WGHT.
NOTE: The data set DEST2.WGHT has 83 observations and 13 variables.
```

Example 2: Saving SAS Files from Deletion

Procedure features:
SAVE statement option:
MEMTYPE=

This example uses the SAVE statement to save some SAS files from deletion and to delete other SAS files.

Program

Write the programming statements to the SAS log. The SAS system option SOURCE writes all programming statements to the log.

```
options pagesize=40 linesize=80 nodate pageno=1 source;
```

```
libname elder 'SAS-data-library';
```

Specify the procedure input library to process.

```
proc datasets lib=elder;
```

Save the data sets CHRONIC, AGING, and CLINICS, and delete all other SAS files (of all types) in the ELDER library. MEMTYPE=DATA is necessary because the ELDER library has a catalog named CLINICS and a data set named CLINICS.

```
      save chronic aging clinics / memtype=data;
   run;
```

SAS Log

```
1    options pagesize=40 linesize=80 nodate pageno=1 source;
2    libname elder 'c:\Myfiles\elder';
NOTE: Libref ELDER was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\Myfiles\elder
3    proc datasets lib=elder;
                                   Directory

               Libref         ELDER
               Engine         V9
               Physical Name  c:\Myfiles\elder
               File Name      c:\Myfiles\elder


                            Member     File
               #  Name      Type       Size  Last Modified

               1  AGING     DATA       5120   04FEB2002:16:07:35
               2  ALCOHOL   DATA       5120   04FEB2002:16:07:35
               3  BACKPAIN  DATA       5120   04FEB2002:16:07:35
               4  CHRONIC   DATA       5120   04FEB2002:16:07:36
               5  CLINICS   CATALOG   17408   04FEB2002:16:07:36
               6  CLINICS   DATA       5120   04FEB2002:16:07:36
               7  DISEASE   DATA       5120   04FEB2002:16:07:36
               8  GROWTH    DATA       5120   04FEB2002:16:07:36
               9  HOSPITAL  CATALOG   17408   04FEB2002:16:07:36
4       save chronic aging clinics / memtype=data;
5    run;

NOTE: Saving ELDER.CHRONIC (memtype=DATA).
NOTE: Saving ELDER.AGING (memtype=DATA).
NOTE: Saving ELDER.CLINICS (memtype=DATA).
NOTE: Deleting ELDER.ALCOHOL (memtype=DATA).
NOTE: Deleting ELDER.BACKPAIN (memtype=DATA).
NOTE: Deleting ELDER.CLINICS (memtype=CATALOG).
NOTE: Deleting ELDER.DISEASE (memtype=DATA).
NOTE: Deleting ELDER.GROWTH (memtype=DATA).
NOTE: Deleting ELDER.HOSPITAL (memtype=CATALOG).
```

Example 3: Modifying SAS Data Sets

Procedure features:

PROC DATASETS statement option:

- NOLIST

FORMAT statement

INDEX CREATE statement options:

- NOMISS
- UNIQUE

INFORMAT statement

LABEL statement

MODIFY statement options:

- LABEL=
- READ=
- SORTEDBY=

RENAME statement

This example modifies two SAS data sets using the MODIFY statement and statements subordinate to it. Example 4 on page 400 shows the modifications to the GROUP data set.

Tasks include

- modifying SAS files
- labeling a SAS data set
- adding a READ password to a SAS data set
- indicating how a SAS data set is currently sorted
- creating an index for a SAS data set
- assigning informats and formats to variables in a SAS data set
- renaming variables in a SAS data set
- labeling variables in a SAS data set.

Program

Write the programming statements to the SAS log. The SAS system option SOURCE writes the programming statements to the log.

```
options pagesize=40 linesize=80 nodate pageno=1 source;
```

```
libname health 'SAS-data-library';
```

Specify HEALTH as the procedure input library to process. NOLIST suppresses the directory listing for the HEALTH data library.

```
proc datasets library=health nolist;
```

Add a label to a data set, assign a READ password, and specify how to sort the data. LABEL= adds a data set label to the data set GROUP. READ= assigns GREEN as the read password. The password appears as Xs in the SAS log. SAS issues a warning message if you specify a level of password protection on a SAS file that does not include alter protection. SORTEDBY= specifies how the data is sorted.

```
   modify group (label='Test Subjects' read=green sortedby=lname);
```

Create the composite index VITAL on the variables BIRTH and SALARY for the GROUP data set. NOMISS excludes all observations that have missing values for BIRTH and SALARY from the index. UNIQUE specifies that the index is created only if each observation has a unique combination of values for BIRTH and SALARY.

```
      index create vital=(birth salary) / nomiss unique;
```

Assign an informat and format, respectively, to the BIRTH variable.

```
      informat birth date7.;
      format birth date7.;
```

Assign a label to the variable SALARY.

```
      label salary='current salary excluding bonus';
```

Rename a variable, and assign a label. Modify the data set OXYGEN by renaming the variable OXYGEN to INTAKE and assigning a label to the variable INTAKE.

```
   modify oxygen;
      rename oxygen=intake;
      label intake='Intake Measurement';
quit;
```

SAS Log

```
6    options pagesize=40 linesize=80 nodate pageno=1 source;
7    libname health 'c:\Myfiles\health';
NOTE: Libref HEALTH was successfully assigned as follows:
      Engine:        V9
      Physical Name: c:\Myfiles\health

8    proc datasets library=health nolist;
9       modify group (label='Test Subjects' read=XXXXX sortedby=lname);
WARNING: The file HEALTH.GROUP.DATA is not ALTER protected.  It could be
         deleted or replaced without knowing the password.
10      index create vital=(birth salary) / nomiss unique;
NOTE: Composite index vital has been defined.
11      informat birth date7.;
12      format birth date7.;
13      label salary='current salary excluding bonus';
14      modify oxygen;
15      rename oxygen=intake;
NOTE: Renaming variable oxygen to intake.
16      label intake='Intake Measurement';
17   quit;
```

Example 4: Describing a SAS Data Set

Procedure features:

CONTENTS statement option:

DATA=

Other features:

SAS data set option:

READ=

This example shows the output from the CONTENTS statement for the GROUP data set. The output shows the modifications made to the GROUP data set in Example 3 on page 398.

Program

```
options pagesize=40 linesize=132 nodate pageno=1;
```

```
libname health 'SAS-data-library';
```

Specify HEALTH as the procedure input library, and suppress the directory listing.

```
proc datasets library=health nolist;
```

Create the output data set GRPOUT from the data set GROUP. Specify GROUP as the data set to describe, give read access to the GROUP data set, and create the output data set GRPOUT, which appears in "The OUT= Data Set" on page 386.

```
   contents data=group (read=green) out=grpout;
   title  'The Contents of the GROUP Data Set';
run;
```

Output

Output 15.8 The Contents of the GROUP Data Set

```
                         The Contents of the GROUP Data Set                              1

                               The DATASETS Procedure

  Data Set Name        HEALTH.GROUP                      Observations          148
  Member Type          DATA                              Variables             11
  Engine               V9                                Indexes               1
  Created              8:06 Tuesday, January 29, 2002    Observation Length    96
  Last Modified        9:13 Tuesday, January 29, 2002    Deleted Observations  0
  Protection           READ                              Compressed            NO
  Data Set Type                                          Sorted                YES
  Label                Test Subjects
  Data Representation  WINDOWS
  Encoding             wlatin1  Western (Windows)


                         Engine/Host Dependent Information

   Data Set Page Size          8192
   Number of Data Set Pages    4
   First Data Page             1
   Max Obs per Page            84
   Obs in First Data Page      62
   Index File Page Size        4096
   Number of Index File Pages  2
   Number of Data Set Repairs  0
   File Name                   c:\Myfiles\health\group.sas7bdat
   Release Created             9.0000A0
   Host Created                WIN_NT


                     Alphabetic List of Variables and Attributes

 #    Variable    Type    Len    Format     Informat    Label

 9    BIRTH       Num       8    DATE7.     DATE7.
 4    CITY        Char     15    $.         $.
 3    FNAME       Char     15    $.         $.
10    HIRED       Num       8    DATE7.     DATE7.
11    HPHONE      Char     12    $.         $.
```

```
                    The Contents of the GROUP Data Set                            2

                          The DATASETS Procedure

                 Alphabetic List of Variables and Attributes

 #    Variable    Type    Len    Format     Informat    Label

 1    IDNUM       Char      4    $.         $.
 7    JOBCODE     Char      3    $.         $.
 2    LNAME       Char     15    $.         $.
 8    SALARY      Num       8    COMMA8.                current salary excluding bonus
 6    SEX         Char      1    $.         $.
 5    STATE       Char      2    $.         $.


                  Alphabetic List of Indexes and Attributes

                                                # of
                          Unique    NoMiss    Unique
               #    Index    Option    Option    Values    Variables

               1    vital    YES       YES          148    BIRTH SALARY


                             Sort Information

                            Sortedby       LNAME
                            Validated      NO
                            Character Set  ANSI
```

Example 5: Concatenating Two SAS Data Sets

Procedure features:

APPEND statement options:

BASE=
DATA=
FORCE=

This example appends one data set to the end of another data set.

Input Data Sets

The BASE= data set, EXP.RESULTS.

```
                 The EXP.RESULTS Data Set                    1

          ID    TREAT    INITWT    WT3MOS    AGE

           1    Other    166.28    146.98     35
           2    Other    214.42    210.22     54
           3    Other    172.46    159.42     33
           5    Other    175.41    160.66     37
           6    Other    173.13    169.40     20
           7    Other    181.25    170.94     30
          10    Other    239.83    214.48     48
          11    Other    175.32    162.66     51
          12    Other    227.01    211.06     29
          13    Other    274.82    251.82     31
```

The data set EXP.SUR contains the variable WT6MOS, but the EXP.RESULTS data set does not.

```
                 The EXP.SUR Data Set                    2

    id     treat     initwt    wt3mos    wt6mos    age

    14    surgery    203.60    169.78    143.88     38
    17    surgery    171.52    150.33    123.18     42
    18    surgery    207.46    155.22       .       41
```

Program

```
options pagesize=40 linesize=64 nodate pageno=1;
```

```
libname exp 'SAS-data-library';
```

Suppress the printing of the EXP library. LIBRARY= specifies EXP as the procedure input library. NOLIST suppresses the directory listing for the EXP library.

```
proc datasets library=exp nolist;
```

Append the data set EXP.SUR to the EXP.RESULTS data set. The APPEND statement appends the data set EXP.SUR to the data set EXP.RESULTS. FORCE causes the APPEND statement to carry out the append operation even though EXP.SUR has a variable that EXP.RESULTS does not. APPEND does not add the WT6MOS variable to EXP.RESULTS.

```
   append base=exp.results data=exp.sur force;
run;
```

Print the data set.

```
proc print data=exp.results noobs;
   title 'The EXP.RESULTS Data Set';
run;
```

Output

Output 15.9

```
                  The EXP.RESULTS Data Set                  1

            ID     TREAT     INITWT    WT3MOS    AGE

             1    Other      166.28    146.98     35
             2    Other      214.42    210.22     54
             3    Other      172.46    159.42     33
             5    Other      175.41    160.66     37
             6    Other      173.13    169.40     20
             7    Other      181.25    170.94     30
            10    Other      239.83    214.48     48
            11    Other      175.32    162.66     51
            12    Other      227.01    211.06     29
            13    Other      274.82    251.82     31
            14    surgery    203.60    169.78     38
            17    surgery    171.52    150.33     42
            18    surgery    207.46    155.22     41
```

Example 6: Aging SAS Data Sets

Procedure features:
AGE statement

This example shows how the AGE statement ages SAS files.

Program

Write the programming statements to the SAS log. The SAS system option SOURCE writes the programming statements to the log.

```
options pagesize=40 linesize=80 nodate pageno=1 source;
```

```
libname daily 'SAS-data-library';
```

Specify DAILY as the procedure input library and suppress the directory listing.

```
proc datasets library=daily nolist;
```

Delete the last SAS file in the list, DAY7, and then age (or rename) DAY6 to DAY7, DAY5 to DAY6, and so on, until it ages TODAY to DAY1.

```
   age today day1-day7;
run;
```

SAS Log

```
6  options pagesize=40 linesize=80 nodate pageno=1 source;
7
8     proc datasets library=daily nolist;
9
10        age today day1-day7;
11     run;
NOTE: Deleting DAILY.DAY7 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY6 to DAILY.DAY7 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY5 to DAILY.DAY6 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY4 to DAILY.DAY5 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY3 to DAILY.DAY4 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY2 to DAILY.DAY3 (memtype=DATA).
NOTE: Ageing the name DAILY.DAY1 to DAILY.DAY2 (memtype=DATA).
NOTE: Ageing the name DAILY.TODAY to DAILY.DAY1 (memtype=DATA).
```

CHAPTER

16

The DBCSTAB Procedure

Overview: DBCSTAB Procedure

The DBCSTAB procedure produces conversion tables for the double-byte character sets that SAS supports.

Syntax: DBCSTAB Procedure

PROC DBCSTAB TABLE=*table-name*
<BASETYPE=*base-type*> <CATALOG=<*libref.*>*catalog-name*>
<DATA=<*libref.*>*table-name* > <DBCSLANG=*language*>
<DESC='*description*'> <FORCE> <VERIFY>;

PROC DBCSTAB Statement

PROC DBCSTAB TABLE=*table-name*
<*option(s)*>;

Required Arguments

TABLE=*table-name*
specifies the name of the double-byte code table to produce. This table name becomes an entry of type DBCSTAB in the catalog that is specified with the CATALOG= option. By default, the catalog name is SASUSER.DBCS.

Alias: NAME=, N=

Options

BASETYPE=*base-type*
specifies a base type for the double-byte code table conversion. If you use this option, you reduce the number of tables that are produced.

If you specify BASETYPE=, then all double-byte codes are first converted to the base code, and then converted to the required code. If you have *n* codes, then there are $n(n\text{-}1)$ conversions that must be made.

Alias: BTYPE=

CATALOG=*<libref.>catalog-name*
specifies the name of the catalog in which the table is to be stored. If the catalog does not exist, it is created.

Default: SASUSER.DBCS

DATA=*<libref.>table-name*
specifies the data for producing the double-byte code table. Several double-byte character variables are required to produce the table. Use variable names that are equivalent to the value of the DBCSTYPE system option and are recognized by the KCVT function.

DBCSLANG=*language*
specifies the language that the double-byte code table uses. The value of this option should match the value of the DBCSLANG system option.

Alias: DBLANG

DESC='*description*'
specifies a text string to put in the DESCRIPTION field for the entry.

FORCE
produces the conversion tables even if errors are present.

VERIFY
checks the data range of the input table per code. This option is used to check for invalid double-byte code.

Details: When Do I Use the DBCSTAB Procedure?

Use the DBCSTAB procedure to modify an existing DBCS table when

- the DBCS encoding system that you are using is not supported by SAS
- the DBCS encoding system that you are using has a nonstandard translation table.

A situation where you would be likely to use the DBCSTAB procedure is when a valid DBCSTYPE= value is not available. These values are operating environment dependent. In such cases, you can use the DBCSTAB procedure to modify a similar translation table, then specify the use of the new table with the TRANTAB option.

Examples: DBCSTAB Procedure

Example 1: Creating a Conversion Table with the DBCSTAB Procedure

Procedure features:

PROC DBCSTAB statement options:

- CATALOG=
- DBLANG=
- BASETYPE=
- VERIFY

The following example creates a Japanese translation table called CUSTAB and demonstrates how the TRANTAB option can be used to specify this new translation table.

Note: The DBCS, DBCSLANG, and DBCSTYPE options are specified at startup. △

The TRANTAB data set is created as follows:

```
data trantab;
      pcms='8342'x; dec='b9b3'x;
run;

proc dbcstab
      /* name of the new translate table */
          name=custtab
      /* based on pcibm encoding */
          basetype=pcms
      /* data to create the new table */
          data=trantab
      /* japanese language */
          dbcslang=japanese
      /* catalog descriptor */
          desc='Modified Japanese Trantab'
      /* where the table is stored */
          catalog=sasuser.dbcs
      /* checks for invalid DBCS in the new data */
          verify;
run;
```

To specify the translate table, use the TRANTAB option:

```
options trantab=(,,,,,,,,custtab);
```

Translate tables are generally used for DBCS conversion with SAS/CONNECT software, PROC CPORT and PROC CIMPORT, and the DATA step function, KCVT. The TRANTAB= option may be used to specify DBCS translate tables. The ninth argument specifies the DBCS system table:

```
options trantab=(,,,,,,,,systab); /* ninth argument */
```

Japanese, Korean, Chinese, and Taiwanese are acceptable for the systab name.

The tenth argument specifies the DBCS user table:

```
options trantab=(,,,,,,,,,usrtab); /* tenth argument */
```

Example 2: Producing Japanese Conversion Tables with the DBCSTAB Procedure

Procedure features:

PROC DBCSTAB statement options:

- TABLE=
- DATA=
- DBLANG=
- BASETYPE=
- VERIFY

Program

```
data ja_jpn;
   length ibm jis euc pcibm $2.;
   ibm='4040'x;
   jis='2121'x;
   euc='a1a1'x;
   pcibm='8140'x;
run;

proc dbcstab
   table=japanese
   data=ja_jpn
   dblang=japanese
   basetype=jis
   verify;
run;
```

Log

```
1   proc dbcstab
2   table=ja_jpn
3   data=work.ja_jpn
4   dblang=japanese
5   basetype=jis
6   verify;
7   run;

NOTE: Base table for JIS created.
NOTE: IBM table for JIS created.
NOTE: PCIBM table for JIS created.
NOTE: EUC table for JIS created.
NOTE: Base table for IBM created.
NOTE: JIS table for IBM created.
NOTE: Base table for PCIBM created.
NOTE: JIS table for PCIBM created.
NOTE: Base table for EUC created.
NOTE: JIS table for EUC created.
NOTE: 10 DBCS tables are generated. Each table has 1 DBCS characters.
NOTE: Each table is 2 bytes in size.
NOTE: Required table memory size is 612.
NOTE: There were 1 observations read from the dataset WORK.JA_JPN.
```

See Also

For an overview of local language support in SAS, see *SAS Language Reference: Concepts*.

Information about the following SAS system options and functions can be found in *SAS Language Reference: Dictionary*:

- KCVT function
- TRANTAB= option.

Also, see Chapter 47, “The TRANTAB Procedure,” on page 1409.

Also, see

- DBCS= system option (refer to the *SAS Companion* for your operating environment).
- DBCSLANG= system option (refer to the *SAS Companion* for your operating environment).
- DBCSTYPE= system option (refer to the *SAS Companion* for your operating environment).

CHAPTER

17

The DISPLAY Procedure

Overview: DISPLAY Procedure

The DISPLAY procedure executes SAS/AF applications. These applications are composed of a variety of entries that are stored in a SAS catalog and that have been built with the BUILD procedure in SAS/AF software. For complete documentation on building SAS/AF applications, see *SAS Guide to Applications Development*.

You can use the DISPLAY procedure to execute an application that runs in NODMS batch mode. Be aware that any SAS programming statements that you submit with the DISPLAY procedure through the SUBMIT block in SCL are not submitted for processing until PROC DISPLAY has executed.

If you use the SAS windowing environment, you can use the AF command to execute an application. SUBMIT blocks execute immediately when you use the AF command. You can use the AFA command to execute multiple applications concurrently.

Syntax: DISPLAY Procedure

PROC DISPLAY CATALOG=*libref.catalog.entry.type* <BATCH>;

PROC DISPLAY Statement

Featured in: Example 1 on page 414

PROC DISPLAY CATALOG=*libref.catalog.entry.type* <BATCH>;

Required Argument

CATALOG=*libref.catalog.entry.type*
specifies a four-level name for the catalog entry.

libref
specifies the SAS data library where the catalog is stored.

catalog
specifies the name of the catalog.

entry
specifies the name of the entry.

type
specifies the entry's type, which is one of the following. For details, see the description of catalog entry types in the BUILD procedure in online help.

CBT

FRAME

HELP

MENU

PROGRAM

SCL

Options

BATCH
runs PROGRAM and SCL entries in batch mode. If a PROGRAM entry contains a display, then it will not run, and you will receive the following error message:

```
ERROR: Cannot allocate window.
```

Restriction: PROC DISPLAY cannot pass arguments to a PROGRAM, a FRAME, or an SCL entry.

Example: DISPLAY Procedure

Example 1: Executing a SAS/AF Application

Procedure features:
PROC DISPLAY statement:
CATALOG = argument

Suppose that your company has developed a SAS/AF application that compiles statistics from an invoice database. Further, suppose that this application is stored in the SASUSER data library, as a FRAME entry in a catalog named INVOICES.WIDGETS. You can execute this application using the following SAS code:

Program

```
proc display catalog=sasuser.invoices.widgets.frame;
run;
```

CHAPTER

18

The DOCUMENT Procedure

Information about the DOCUMENT Procedure

See: For complete documentation of the DOCUMENT procedure, go to **`http://www.sas.com/service/library/onlinedoc`**. Select Base SAS from the Product-Specific Documentation list.

CHAPTER

19

The EXPLODE Procedure

Overview: EXPLODE Procedure

The EXPLODE procedure produces printed output with oversized text by expanding each letter into a matrix of characters. You can use the EXPLODE procedure to generate posters, flip charts, and header pages for computer output.

Note: PROC EXPLODE with a PARMCARDS statement cannot be included in a macro. △

Output 19.1 on page 419 shows the results of the most basic form of a PROC EXPLODE step with only one line of text. The following statements produce the output:

```
options nodate pageno=1 linesize=80
        pagesize=60;

proc explode;
   parmcards;
 TOP SECRET
;
```

Output 19.1 A Line of Expanded Text

```
                          The SAS System                      1
*****   ***   ****       ***   *****   ***   ****   *****  *****
  *    *   *  *   *     *   *  *      *   *  *   *  *        *
  *    *   *  *   *      *     *      *      *   *  *        *
  *    *   *  ****        *    ****   *      ****   ****     *
  *    *   *  *            *   *      *      * *    *        *
  *    *   *  *         *   *  *      *   *  *  *   *        *
  *     ***   *          ***   *****   ***   *   *  *****    *
```

Through options you can control spacing, the density of the text, and underlining.

Syntax: EXPLODE Procedure

Requirements: PARMCARDS or PARMCARDS4
Message line(s)
Null statement

Reminder: You can use global statements with PROC EXPLODE. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC EXPLODE;
PARMCARDS | **PARMCARDS4**;
message-line(s)
; | ;;;;

PROC EXPLODE Statement

PROC EXPLODE;

PARMCARDS or PARMCARDS4 Statement

Signals the beginning of the message lines.

Requirement: If any part of the message contains a semicolon, you must use PARMCARDS4.

See also: "Null Statement" on page 422

Featured in: Example 1 on page 423 and Example 2 on page 424

PARMCARDS | **PARMCARDS4**;

Message Lines

Specifies the block of text (one or more lines) and any special characters that control the appearance of the text.

Featured in: Example 1 on page 423 and Example 2 on page 424

Message line(s)

<D | L>

<S*n* | P>

<spacing-control>

text

<U *character-1 <...character-n>*>

. . . more blocks of option specifications and text lines . . .

<D | L>

<S*n* | P>

<spacing-control>

<U *character-1 <...character-n>*>

Required Argument

text

specifies the line of printed text. It can contain only the following characters:

ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890

. − + , = * $ / _ ( )> < | & '? ! ; # ¬ " % @ blank

The not symbol (¬) can also appear as either a hat (ˆ)or a tilde (~) depending on your keyboard. PROC EXPLODE ignores lowercase characters.

The EXPLODE procedure reproduces horizontal spacing as it appears in the program, except for column 1, which is reserved for the *spacing-control* option.

Restriction: *text* can begin in any column except the first.

Options

To do this	Use this option
Control vertical spacing	S*n* or *spacing-control*
Control the text density	
Specify dark characters	D
Specify light characters	L
Underline text	U
Begin a new page	P

D | L

controls the density of printed characters. Specify D to produce dark characters that are formed by overprinting the characters H, T, and Q. Specify L to produce light characters that are formed of asterisks.

Default: L initially, then for each line of text the value is carried over from the previous line if you do not specify a value.

Requirement: Must appear in column 1, and must be the only character on that line.

Requirement: To produce overprinting, the SAS system option OVP must be in effect, and your printer must support overprinting.

Featured in: Example 2 on page 424

L
See D | L.

P
See S*n* | P.

S*n* | P
controls the amount of space before the next line of text.

S*n*
skips *n* lines before the next line of text.

Range: 1–9

See also: *spacing-control*

Featured in: Example 1 on page 423

P
begins a new page before the next line of text.

Featured in: Example 2 on page 424

Default: 0

Requirement: Must begin in column 1 and must be the only characters(s) on that line.

spacing-control
specifies the number of lines to skip before the next line of text.

Default: 0

Range: 1–9

Requirement: Must appear in column 1.

Restriction: Spacing control does not work at the top of the page.

See also: S*n* option

<U *character-1* <...*character-n*>>
underlines the *text* on the previous line with asterisks. The *character* values can be anything. The nonblank characters determine where the underline appears. PROC EXPLODE skips two lines before printing the underline.

Featured in: Example 2 on page 424

Null Statement

Ends the PROC EXPLODE step.

Requirement: The Null statement must begin in the first column. If any part of the message contains a semicolon, use four semicolons instead of one.

See also: “PARMCARDS or PARMCARDS4 Statement” on page 420

;|;;;;

Examples: EXPLODE Procedure

Example 1: Controlling Spacing

Procedure features: PARMSCARDS statement

Message lines options: S

spacing-control

This example

- controls horizontal spacing in the output by shifting the starting point of the text lines in the program
- controls vertical spacing with an initial gap of two lines and another gap of two lines before the second line of text.

Program

```
options nodate pageno=1 linesize=88 pagesize=60;
```

Specify the file to which the text is written. PARMCARDS= specifies the file reference, EXTFILE, of the file, PARMFILE, to which PROC EXPLODE writes the text in the message lines.

```
options parmcards=extfile;
filename extfile 'parmfile';


proc explode;
   title 'Cover Page';
```

Specify the spacing control. The numeral 6 before **WORDS** specifies the spacing control. S2 skips two lines before the next line of text.

```
   parmcards;
 THESE
6 WORDS
S2
     ARE BIG
;
```

Output

```
                                      Cover Page                                      1

*****  *   *  *****   ***   *****
  *    *   *  *      *   *  *
  *    *   *  *       *     *
  *    *****  ****     *    ****
  *    *   *  *         *   *
  *    *   *  *      *   *  *
  *    *   *  *****   ***   *****








       *   *   ***   ****   ****    ***
       *   *  *   *  *   *   *  *  *   *
       *   *  *   *  *   *   *  *   *
       *   *  *   *  ****    *  *    *
       * * *  *   *  * *     *  *     *
       ** **  *   *  *  *    *  *  *   *
       *   *   ***   *   *  ****    ***






                              *    ****   *****         ****   *****   ***
                             * *   *   *  *              *  *    *    *   *
                            *   *  *   *  *              *  *    *    *
                            *****  ****   ****           ***     *    * ***
                            *   *  * *    *              *  *    *    *   *
                            *   *  *  *   *              *  *    *    *   *
                            *   *  *   *  *****         ****   *****   ***
```

Example 2: Darkening and Underlining Text

Procedure features: PARMSCARDS4 statement

Message lines options: D

L

P

U

SAS system option: OVP

This example

- □ prints dark text and then returns to light text
- □ specifies a page break
- □ underlines text.

Program

Put overprinted characters in the text. OVP allows overprinted characters in the text.

```
options nodate pageno=1 linesize=88 pagesize=60 ovp;
```

Specify the file that will contain the procedure output. PARMCARDS= specifies the file reference, EXTFILE, of the file, PARMFILE, to which PROC EXPLODE writes the text in the message lines.

```
options parmcards=extfile;
filename extfile 'parmfile';
```

```
proc explode;
   title 'Important Message';
```

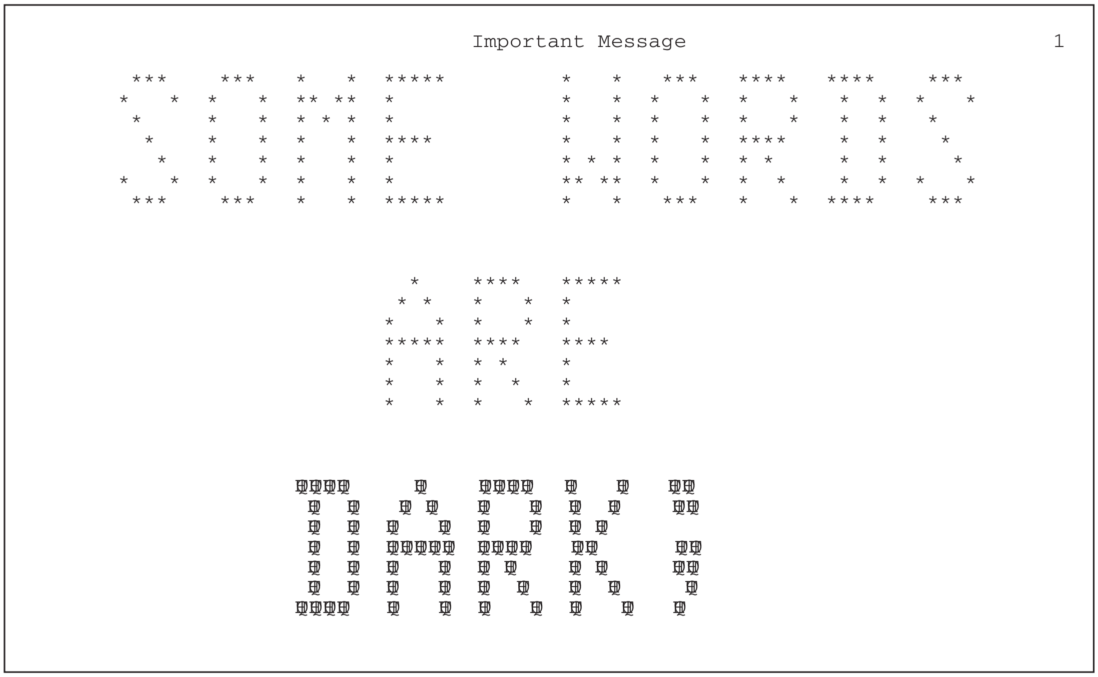
Customize the text in the output. D overprints the line of text to make it darker, P begins a new page, and L returns to regular printing. U with the line of asterisks creates the underline.

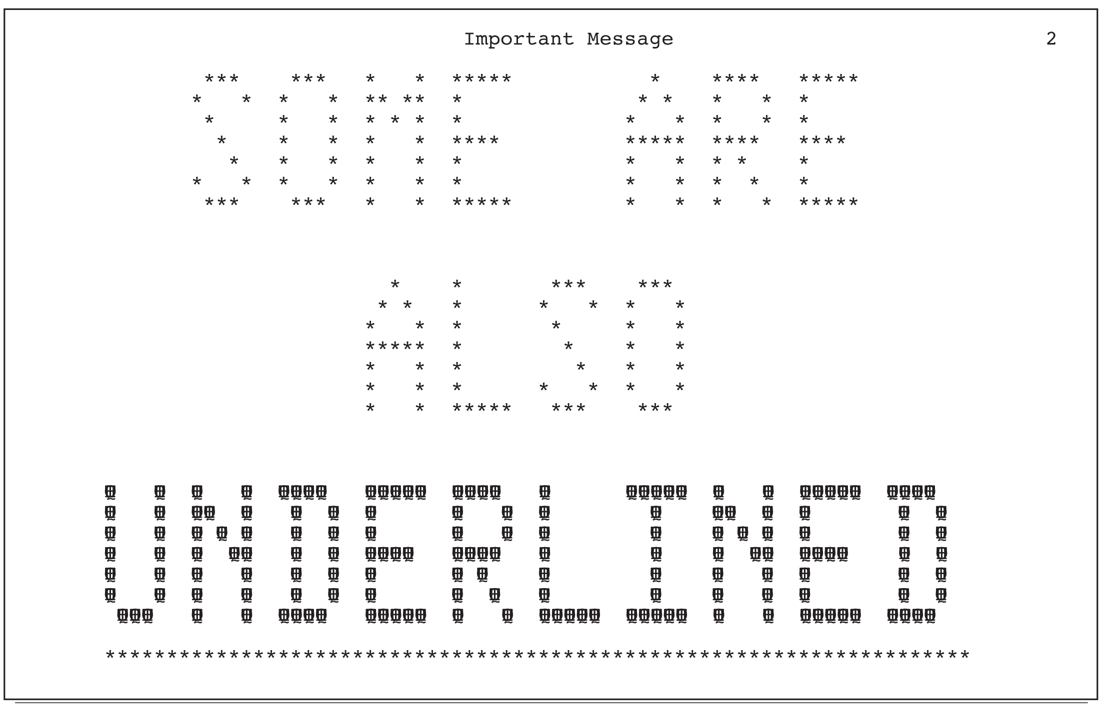
```
   parmcards4;
  SOME WORDS
     ARE
D
    DARK;
P
L
   SOME ARE
     ALSO
```

The Null statement uses four semicolons because the message contains a semicolon.

```
D
  UNDERLINED
U **********
;;;;
```

Output





CHAPTER
20

The EXPORT Procedure

Overview: EXPORT Procedure

The EXPORT procedure reads data from a SAS data set and writes it to an external data source. External data sources can include Microsoft Access Database, Excel files, Lotus spreadsheets, and delimited external files (in which columns of data values are separated by a delimiter such as a blank, comma, or tab).

When you execute PROC EXPORT, the procedure reads the input data set and writes the data to the external data source. PROC EXPORT exports the data by one of the following methods:

- generated DATA step code
- generated SAS/ACCESS code
- translation engines.

You control the results with options and statements that are specific to the output data source. PROC EXPORT produces the specified output file and writes information about the export to the SAS log. In the log, you see the DATA step or the SAS/ACCESS code that is generated by PROC EXPORT. If a translation engine is used, then no code is submitted.

Note: To export data, you can also use the Export Wizard, which is a windowing tool that guides you through the steps to export a SAS data set. You can request the Export Wizard to generate EXPORT procedure statements, which you can save to a file for subsequent use. To invoke the Export Wizard, from the SAS windowing environment select

File ▶ Export Data

△

Syntax: PROC EXPORT

Restriction: PROC EXPORT is available for the following operating environments:

- OpenVMS Alpha
- UNIX
- Microsoft Windows.

PROC EXPORT DATA=<*libref.*>*SAS-data-set* <(*SAS-data-set-options*)>
OUTFILE="*filename*" | OUTTABLE="*tablename*"
<DBMS=*identifier*> <REPLACE>;
<*data-source-statement(s)*;>

PROC EXPORT Statement

Featured in: All examples

PROC EXPORT DATA=<*libref.*>*SAS-data-set* <(*SAS-data-set-options*)>
OUTFILE="*filename*" | OUTTABLE="*tablename*"
<DBMS=*identifier*> <REPLACE>;

Required Arguments

DATA=<*libref.*>*SAS-data-set*
identifies the input SAS data set with either a one- or two-level SAS name (library and member name). If you specify a one-level name, by default, PROC EXPORT uses either the USER library (if assigned) or the WORK library (if USER not assigned).

Default: If you do not specify a SAS data set, PROC EXPORT uses the most recently created SAS data set, which SAS keeps track of with the system variable _LAST_. However, in order to be certain that PROC EXPORT uses the correct data set, you should identify the SAS data set.

Restriction: PROC EXPORT can export data only if the format of the data is supported by the data source or the amount of data is within the limitations of the data source. For example, some data sources have a maximum number of rows or columns, and some data sources cannot support SAS user-defined formats and informats. If the data that you want to export exceeds the limits of the data source, PROC EXPORT may not be able to export it correctly. When incompatible formats are encountered, the procedure formats the data to the best of its ability.

Restriction: PROC EXPORT does not support writing labels as column names. However, SAS does support column names up to 32 characters.

Featured in: All examples

(*SAS-data-set-options*)

specifies SAS data set options. For example, if the data set that you are exporting has an assigned password, you can use the ALTER=, PW=, READ=, or WRITE= data set option, or to export only data that meets a specified condition, you can use the WHERE= data set option. For information about SAS data set options, see "Data Set Options" in *SAS Language Reference: Dictionary*.

Restriction: You cannot specify data set options when exporting delimited, comma-separated, or tab-delimited external files.

Featured in: Example 2 on page 437

OUTFILE="*filename*"

specifies the complete path and filename or a fileref for the output PC file, spreadsheet, or delimited external file. If you specify a fileref or if the complete path and filename does not include special characters (such as the backslash in a path), lowercase characters, or spaces, you can omit the quotation marks. A fileref is a SAS name that is associated with the physical location of the output file. To assign a fileref, use the FILENAME statement. For more information about PC file formats, see *SAS/ACCESS for PC Files: Reference*.

Featured in Example 1 on page 434, Example 2 on page 437, and Example 3 on page 438

Restriction: PROC EXPORT does not support device types or access methods for the FILENAME statement except for DISK. For example, PROC EXPORT does not support the TEMP device type, which creates a temporary external file.

OUTTABLE="*tablename*"

specifies the table name of the output DBMS table. If the name does not include special characters (such as question marks), lowercase characters, or spaces, you can omit the quotation marks. Note that the DBMS table name may be case sensitive.

Requirement: When you export a DBMS table, you must specify the DBMS= option.

Featured in: Example 4 on page 438

Options

DBMS=*identifier*

specifies the type of data to export. To export a DBMS table, you must specify DBMS= by using a valid database identifier. For example, DBMS=ACCESS specifies to export a table into a Microsoft Access 2000 or 2002 database. To export PC files, spreadsheets, and delimited external files, you do not have to specify DBMS= if the filename that is specified in OUTFILE= contains a valid extension so that PROC EXPORT can recognize the type of data. For example, PROC EXPORT recognizes the filename ACCOUNTS.WK1 as a Lotus 1-2-3 Release 2 spreadsheet and the filename MYDATA.CSV as an external file that contains comma-separated data values; therefore, a DBMS= specification is not necessary.

The following values are valid for the DBMS= option:

Identifier	Output Data Source	Extension	Host Availability	Version of File Created
ACCESS	Microsoft Access 2000 or 2002 table	.mdb	Microsoft Windows *	2000
ACCESS97	Microsoft Access 97 table	.mdb	Microsoft Windows *	97
ACCESS2000	Microsoft Access 2000 table	.mdb	Microsoft Windows *	2000

Identifier	Output Data Source	Extension	Host Availability	Version of File Created
ACCESS2002	Microsoft Access 2002 table	.mdb	Microsoft Windows *	2000
CSV	delimited file (comma-separated values)	.csv	OpenVMS Alpha, UNIX, Microsoft Windows	
DBF	dBASE 5.0, IV, III+, and III files	.dbf	UNIX, Microsoft Windows	5.0
DLM	delimited file (default delimiter is a blank)	.*	OpenVMS Alpha, UNIX, Microsoft Windows	
EXCEL	Excel 97 or 2000 or 2002 spreadsheet	.xls	Microsoft Windows *	97
EXCEL4	Excel 4.0 spreadsheet	.xls	Microsoft Windows	4.0
EXCEL5	Excel 5.0 or 7.0 (95) spreadsheet	.xls	Microsoft Windows	5.0
EXCEL97	Excel 97 spreadsheet	.xls	Microsoft Windows *	97
EXCEL2000	Excel 2000 spreadsheet	.xls	Microsoft Windows *	97
EXCEL2002	Excel 2002 spreadsheet	.xls	Microsoft Windows *	97
TAB	delimited file (tab-delimited values)	.txt	OpenVMS Alpha, UNIX, Microsoft Windows	
WK1	Lotus 1-2-3 Release 2 spreadsheet	.wk1	Microsoft Windows	
WK3	Lotus 1-2-3 Release 3 spreadsheet	.wk3	Microsoft Windows	
WK4	Lotus 1-2-3 Release 4 and 5 spreadsheet	.wk4	Microsoft Windows	

* Not available for Microsoft Windows 64-Bit Edition.

Restriction: The availability of an output data source depends on

- the operating environment, and in some cases the platform, as specified in the previous table.

- whether your site has a license to the SAS/ACCESS software for PC file formats. If you do not have a license, only delimited files are available.

Featured in: Example 1 on page 434 and Example 4 on page 438

When you specify a value for DBMS=, consider the following for specific data sources:

- To export to an existing Microsoft Access database, PROC EXPORT can write to Access 97, Access 2000, or Access 2002 regardless of your specification. For example, if you specify DBMS=ACCESS2000 and the database is in Access 97 format, PROC EXPORT exports the table, and the database remains in Access 97 format. However, if you specify OUTFILE= for an Access database that does not exist, a new database is created using the format specified in DBMS=. For example to create a new Access database, specifying DBMS=ACCESS (which defaults to Access 2000 or 2002 format) creates an MDB file that can be read by Access 2000 or Access 2002, not by Access 97.

  The following table lists the DBMS= specifications and indicates which version of Microsoft Access can open the resulting database:

Specification	Access 2002	Access 2000	Access 97
ACCESS	yes	yes	no
ACCESS2002	yes	yes	no
ACCESS2000	yes	yes	no
ACCESS97	yes	yes	yes

- To export a Microsoft Excel spreadsheet, PROC EXPORT creates an XLS file for the version specified. The following table lists the DBMS= specifications and indicates which version of Microsoft Excel can open the resulting spreadsheet:

Specification	Excel 2002	Excel 2000	Excel 97	Excel 5.0	Excel 4.0
EXCEL	yes	yes	yes	no	no
EXCEL2002	yes	yes	yes	no	no
EXCEL2000	yes	yes	yes	no	no
EXCEL97	yes	yes	yes	no	no
EXCEL5	yes	yes	yes	yes	no
EXCEL4	yes	yes	yes	yes	yes

  Note: Later versions of Excel can open and update files in earlier formats. △

- When exporting a SAS data set to a dBASE file (DBF), if the data set contains missing values (for either character or numeric values), the missing values are translated to blanks.
- When exporting a SAS data set to a dBASE file (DBF), values for a character variable that are longer than 255 characters are truncated in the resulting dBASE file because of dBASE limitations.

REPLACE

overwrites an existing file. Note that for a Microsoft Access database or an Excel workbook, REPLACE overwrites the target table or spreadsheet. If you do not specify REPLACE, PROC EXPORT does not overwrite an existing file.

Featured in: Example 2 on page 437 and Example 4 on page 438

Data Source Statements

PROC EXPORT provides a variety of statements that are specific to the output data source.

Statements for PC Files, Spreadsheets, or Delimited Files

The following statement is available when you export delimited external files:

DELIMITER='*char*' | '*nn*'*x*;

specifies the delimiter to separate columns of data in the output file. You can specify the delimiter as a single character or as a hexadecimal value. For example, if you want columns of data to be separated by an ampersand, specify DELIMITER='&'. If you do not specify DELIMITER=, PROC EXPORT assumes that the delimiter is a blank. You can replace the equal sign with a blank.

Interaction: You do not have to specify DELIMITER= if you specify DBMS=CSV, DBMS=TAB, or if the output filename has an extension of .CSV or .TXT.

Featured in: Example 1 on page 434

SHEET=*spreadsheet-name*;

identifies a particular spreadsheet name to load into a workbook. You use this statement for Microsoft Excel 97, 2000, or 2002 only. If the SHEET= statement is not specified, PROC EXPORT uses the SAS data set name as the spreadsheet name to load the data.

For Excel data access, a spreadsheet name is treated as a special case of a range name with a dollar sign ($) appended. For example, if you export a table and specify `sheet=Invoice`, you will see a range (table) name INVOICE and another range (table) name 'INVOICES$' created. Excel appends a dollar sign ($) to a spreadsheet name in order to distinguish it from the corresponding range name.

Note: You should not append the dollar sign ($) when you specify the spreadsheet name. For example, SHEET= 'Invoice$' is not allowed. △

You should avoid using special characters for spreadsheet names when exporting a table to an Excel file. Special characters such as a space or a hyphen are replaced with an underscore. For example, if you export a table and specify `sheet='Sheet Number 1'`, PROC EXPORT creates the range names `Sheet_Number_1` and `Sheet_Number_1$`.

Featured in: Example 3 on page 438

Statements for DBMS Tables

The following statements are available to establish a connection to the DBMS when you are exporting to a DBMS table:

DATABASE="*database*";
: specifies the complete path and filename of the database to contain the specified DBMS table. If the database name does not contain lowercase characters, special characters, or national characters ($, #, or @), you can omit the quotation marks. You can replace the equal sign with a blank.

Note: A default may be configured in the DBMS client software; SAS does not generate a default value. △

Featured in: Example 4 on page 438

DBPWD="*database-password*";
: specifies a password that allows access to a database. You can replace the equal sign with a blank.

PWD="*password*";
: specifies the user password used by the DBMS to validate a specific userid. If the password does not contain lowercase characters, special characters, or national characters, you can omit the quotation marks. You can replace the equal sign with a blank.

Note: The DBMS client software may default to the userid and password that was used to log in to the operating environment; SAS does not generate a default value. △

UID="*userid*";
: identifies the user to the DBMS. If the userid does not contain lowercase characters, special characters, or national characters, you can omit the quotation marks. You can replace the equal sign with a blank.

Note: The DBMS client software may default to the userid and password that were used to log in to the operating environment; SAS does not generate a default value. △

WGDB="*workgroup-database-name*";
: specifies the workgroup (security) database name that contains the USERID and PWD data for the DBMS. If the workgroup database name does not contain lowercase characters, special characters, or national characters, you can omit the quotation marks. You can replace the equal sign with a blank.

Note: A default workgroup database may be used by the DBMS; SAS does not generate a default value. △

Security Levels for Microsoft Access Tables

Microsoft Access tables have the following levels of security, for which specific combinations of security statements must be used:

None
: Do not specify DBPWD=, PWD=, UID=, or WGDB=.

Password
: Specify only DBPWD=.

User-level
: Specify only PWD=, UID=, and WGDB=.

Full
: Specify DBPWD=, PWD=, UID=, and WGDB=.

Each statement has a default value; however, you may find it necessary to provide a value for each statement explicitly.

Examples: PROC EXPORT

Example 1: Exporting a Delimited External File

Procedure features:
PROC EXPORT statement arguments:
DATA=
DBMS=
OUTFILE=
Data source statement:
DELIMITER=

This example exports the following SAS data set named SASHELP.CLASS and creates a delimited external file:

Output 20.1 PROC PRINT of SASHELP.CLASS

```
                              The SAS System                                1

     Obs    Name       Sex           Age         Height         Weight

       1    Alfred      M             14           69            112.5
       2    Alice       F             13           56.5           84
       3    Barbara     F             13           65.3           98
       4    Carol       F             14           62.8          102.5
       5    Henry       M             14           63.5          102.5
       6    James       M             12           57.3           83
       7    Jane        F             12           59.8           84.5
       8    Janet       F             15           62.5          112.5
       9    Jeffrey     M             13           62.5           84
      10    John        M             12           59             99.5
      11    Joyce       F             11           51.3           50.5
      12    Judy        F             14           64.3           90
      13    Louise      F             12           56.3           77
      14    Mary        F             15           66.5          112
      15    Philip      M             16           72            150
      16    Robert      M             12           64.8          128
      17    Ronald      M             15           67            133
      18    Thomas      M             11           57.5           85
      19    William     M             15           66.5          112
```

Program

Identify the input SAS data set, specify the output filename, and specify the type of file. Note that the filename does not contain an extension. DBMS=DLM specifies that the output file is a delimited external file.

```
proc export data=sashelp.class
   outfile='c:\myfiles\class'
```

```
   dbms=dlm;
```

Specify the delimiter. The DELIMITER= option specifies that an & (ampersand) will delimit data fields in the output file. The delimiter separates the columns of data in the output file.

```
   delimiter='&';
run;
```

SAS Log

The SAS log displays the following information about the successful export. Notice the generated SAS DATA step.

```
47     /*****************************************************************
48     *   PRODUCT:   SAS
49     *   VERSION:   9.00
50     *   CREATOR:   External File Interface
51     *   DATE:      07FEB02
52     *   DESC:      Generated SAS Datastep Code
53     *   TEMPLATE SOURCE:  (None Specified.)
54     ******************************************************************/
55        data _null_;
56        set  SASHELP.CLASS                                end=EFIEOD;
57        %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
58        %let _EFIREC_ = 0;     /* clear export record count macro variable */
59        file 'c:\myfiles\class' delimiter='&' DSD DROPOVER
59 ! lrecl=32767;
60           format Name $8. ;
61           format Sex $1. ;
62           format Age best12. ;
63           format Height best12. ;
64           format Weight best12. ;
65        if _n_ = 1 then        /* write column names */
66         do;
67           put
68           'Name'
69           '&'
70           'Sex'
71           '&'
72           'Age'
73           '&'
74           'Height'
75           '&'
76           'Weight'
77           ;
78         end;
79         do;
80           EFIOUT + 1;
81           put Name $ @;
82           put Sex $ @;
83           put Age @;
84           put Height @;
85           put Weight ;
86           ;
87         end;
88        if _ERROR_ then call symput('_EFIERR_',1);  /* set ERROR detection
88 ! macro variable */
89        If EFIEOD then
90           call symput('_EFIREC_',EFIOUT);
91        run;

NOTE: Numeric values have been converted to character
      values at the places given by: (Line):(Column).
      88:44   90:31
NOTE: The file 'c:\myfiles\class' is:
      File Name=c:\myfiles\class,
      RECFM=V,LRECL=32767

NOTE: 20 records were written to the file 'c:\myfiles\class'.
      The minimum record length was 17.
      The maximum record length was 26.
NOTE: There were 19 observations read from the data set SASHELP.CLASS.
NOTE: DATA statement used (Total process time):
      real time           0.13 seconds
      cpu time            0.05 seconds


19 records created in c:\myfiles\class from SASHELP.CLASS
                  .


NOTE: c:\myfiles\class was successfully created.
```

Output

The external file produced by PROC EXPORT follows.

```
Name&Sex&Age&Height&Weight
Alfred&M&14&69&112.5
Alice&F&13&56.5&84
Barbara&F&13&65.3&98
Carol&F&14&62.8&102.5
Henry&M&14&63.5&102.5
James&M&12&57.3&83
Jane&F&12&59.8&84.5
Janet&F&15&62.5&112.5
Jeffrey&M&13&62.5&84
John&M&12&59&99.5
Joyce&F&11&51.3&50.5
Judy&F&14&64.3&90
Louise&F&12&56.3&77
Mary&F&15&66.5&112
Philip&M&16&72&150
Robert&M&12&64.8&128
Ronald&M&15&67&133
Thomas&M&11&57.5&85
William&M&15&66.5&112
```

Example 2: Exporting a Subset of Observations to an Excel Spreadsheet

Procedure features:

PROC EXPORT statement arguments:

DATA=
DBMS=
OUTFILE=
REPLACE

This example exports the SAS data set SASHELP.CLASS, shown in Output 20.1 on page 434. PROC EXPORT creates an Excel file named Femalelist.xsl, and by default, creates a spreadsheet named Class. Since the SHEET= data source statement is not specified, PROC EXPORT uses the name of the SAS data set as the spreadsheet name. The WHERE= SAS data set option is specified in order to export a subset of the observations, which results in the spreadsheet containing only the female students.

Program

Identify the input SAS data set, request a subset of the observations, specify the output data source, specify the output file, and overwrite the target spreadsheet if it exists. The output file is an Excel 2000 spreadsheet.

```
proc export data=sashelp.class (where=(sex='F'))
   outfile='c:\myfiles\Femalelist.xls'
```

```
   dbms=excel
   replace;
run;
```

Example 3: Exporting to a Specific Spreadsheet in an Excel Workbook

Procedure features:

PROC EXPORT statement arguments:

DATA=
DBMS=
OUTFILE=

Data Source Statement:

SHEET=

This example exports a SAS data set named MYFILES.GRADES1 and creates an Excel 2000 workbook named Grades.xsl. MYFILES.GRADES1 becomes one spreadsheet in the workbook named Grades1.

Program

Identify the input SAS data set, specify the output data source, and specify the output file.

```
proc export data=myfiles.grades1
   dbms=excel2000
   outfile='c:\Myfiles\Grades.xls';
```

Identify a particular spreadsheet to write to in a workbook.

```
   sheet=Grades1;
run;
```

Example 4: Exporting a Microsoft Access Table

Procedure features:

PROC EXPORT statement arguments:

DATA=
DBMS=
OUTTABLE=
REPLACE

Data Source Statement:

DATABASE=

This example exports a SAS data set named SASUSER.CUST, the first five observations of which follow, and creates a Microsoft Access 97 table. The security level for this Access table is none, so it is not necessary to specify any of the database security statements.

```
Obs     Name                    Street                    Zipcode

 1      David Taylor            124 Oxbow Street          72511
 2      Theo Barnes             2412 McAllen Avenue       72513
 3      Lydia Stirog            12550 Overton Place       72516
 4      Anton Niroles           486 Gypsum Street         72511
 5      Cheryl Gaspar           36 E. Broadway            72515
```

Program

Identify the input SAS data set, specify the output DBMS table name and the output data source, and overwrite the output file if it exists. The output file is a Microsoft Access 97 table. The option REPLACE overwrites an existing file. If you do not specify REPLACE, PROC EXPORT does not overwrite an existing file.

```
proc export data=sasuser.cust
   outtable="customers"
   dbms=access97
   replace;
```

Specify the path and filename of the database to contain the table.

```
   database="c:\myfiles\mydatabase.mdb";
run;
```

CHAPTER 21

The FORMAT Procedure

Overview: FORMAT Procedure

The FORMAT procedure enables you to define your own informats and formats for variables. In addition, you can print the parts of a catalog that contain informats or formats, store descriptions of informats or formats in a SAS data set, and use a SAS data set to create informats or formats.

Informats determine how raw data values are read and stored. *Formats* determine how variable values are printed. For simplicity, this section uses the terminology *the informat converts* and *the format prints*.

Informats and formats tell the SAS System the data's type (character or numeric) and form (such as how many bytes it occupies; decimal placement for numbers; how to handle leading, trailing, or embedded blanks and zeros; and so forth). The SAS System provides informats and formats for reading and writing variables. For a thorough description of informats and formats that SAS provides, see the sections on formats and informats in *SAS Language Reference: Dictionary*.

With informats, you can

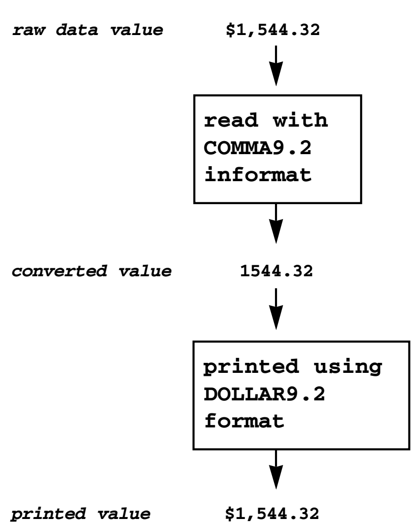
- convert a number to a character string (for example, convert 1 to **YES**)
- convert a character string to a different character string (for example, convert **'YES'** to **'OUI'**)
- convert a character string to a number (for example, convert **YES** to 1)
- convert a number to another number (for example, convert 0 through 9 to 1, 10 through 100 to 2, and so forth.

With formats, you can

- print numeric values as character values (for example, print 1 as **MALE** and 2 as **FEMALE**)
- print one character string as a different character string (for example, print **YES** as **OUI**)
- print numeric values using a template (for example, print 9458763450 as **945-876-3450**).

The following figure summarizes what occurs when you associate an informat and format with a variable. The COMMA*w.d* informat and the DOLLAR*w.d* format are provided by SAS.

Display 21.1 Associating an informat and a format with a variable



In the figure, SAS reads the raw data value that contains the dollar sign and comma. The COMMA9.2 informat ignores the dollar sign and comma and converts the value to 1544.32. The DOLLAR9.2 format prints the value, adding the dollar sign and comma. For more information about associating informats and formats with variables, see "Associating Informats and Formats with Variables" on page 465.

Syntax: FORMAT Procedure

Restriction: You cannot use a SELECT statement and an EXCLUDE statement within the same PROC FORMAT step.

Reminder: You can also use appropriate global statements with this procedure. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC FORMAT <*option(s)*>;
  EXCLUDE *entry(s)*;
  INVALUE <$>*name* <(*informat-option(s)*)>
    value-range-set(s);
  PICTURE *name* <(*format-option(s)*)>
    value-range-set-1 <(*picture-1-option(s)*)>
    <*...value-range-set-n* <(*picture-n-option(s)*)>>;
  SELECT *entry(s)*;
  VALUE <$>*name* <(*format-option(s)*)>
    value-range-set(s);

To do this	Use this statement
Exclude catalog entries from processing by the FMTLIB and CNTLOUT= options	EXCLUDE
Create an informat for reading and converting raw data values	INVALUE
Create a template for printing numbers	PICTURE
Select catalog entries from processing by the FMTLIB and CNTLOUT= options	SELECT
Create a format that specifies character strings to use to print variable values	VALUE

PROC FORMAT Statement

Reminder: You can use data set options with the CNTLIN= and CNTLOUT= data set options. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," for a list.

PROC FORMAT <*option(s)*>;

To do this	Use this option
Specify a SAS data set from which PROC FORMAT builds informats or formats	CNTLIN=
Create a SAS data set that stores information about informats or formats	CNTLOUT=
Print information about informats or formats	FMTLIB
Specify a SAS library or catalog that will contain the informats or formats that you are creating in the PROC FORMAT step	LIBRARY=
Specify the number of characters of the informatted or formatted value that appear in PROC FORMAT output	MAXLABLEN=
Specify the number of characters of the start and end values that appear in the PROC FORMAT output	MAXSELEN=
Prevent a new informat or format from replacing an existing one of the same name	NOREPLACE
Print information about each format and informat on a separate page[1]	PAGE

1 Used in conjunction with FMTLIB. If PAGE is specified, FMTLIB is invoked (or assumed).

Options

CNTLIN=*input-control-SAS-data-set*

specifies a SAS data set from which PROC FORMAT builds informats and formats. CNTLIN= builds formats and informats without using a VALUE, PICTURE, or INVALUE statement. If you specify a one-level name, then the procedure searches only the default data library (either the WORK data library or USER data library) for the data set, regardless of whether you specify the LIBRARY= option.

Note: LIBRARY= can point to either a data library or a catalog. If only a libref is specified, a catalog name of FORMATS is assumed. △

Tip: A common source for an input control data set is the output from the CNTLOUT= option of another PROC FORMAT step.

See also: "Input Control Data Set" on page 471

Featured in: Example 5 on page 482

CNTLOUT=*output-control-SAS-data-set*

creates a SAS data set that stores information about informats and formats that are contained in the catalog specified in the LIBRARY= option.

Note: LIBRARY= can point to either a data library or a catalog. If only a libref is specified, then a catalog name of FORMATS is assumed. △

If you are creating an informat or format in the same step that the CNTLOUT= option appears, then the informat or format that you are creating is included in the CNTLOUT= data set.

If you specify a one-level name, then the procedure stores the data set in the default data library (either the WORK data library or the USER data library), regardless of whether you specify the LIBRARY= option.

Tip: You can use an output control data set as an input control data set in subsequent PROC FORMAT steps.

See also: “Output Control Data Set” on page 468

FMTLIB

prints information about all the informats and formats in the catalog that is specified in the LIBRARY= option. To get information only about specific informats or formats, subset the catalog using the SELECT or EXCLUDE statement.

Interaction: The PAGE option invokes FMTLIB.

Tip: If your output from FMTLIB is not formatted correctly, then try increasing the value of the LINESIZE= system option.

Tip: If you use the SELECT or EXCLUDE statement and omit the FMTLIB and CNTLOUT= options, then the procedure invokes the FMTLIB option and you receive FMTLIB option output.

Featured in: Example 6 on page 486

LIBRARY=*libref<.catalog>*

specifies a catalog to contain informats or formats that you are creating in the current PROC FORMAT step. The procedure stores these informats and formats in the catalog that you specify so that you can use them in subsequent SAS sessions or jobs.

Note: LIBRARY= can point to either a data library or a catalog. If only a libref is specified, then a catalog name of FORMATS is assumed. △

Alias: LIB=

Default: If you omit the LIBRARY= option, then formats and informats are stored in the WORK.FORMATS catalog. If you specify the LIBRARY= option but do not specify a name for *catalog*, then formats and informats are stored in the *libref*.FORMATS catalog.

Tip: SAS automatically searches LIBRARY.FORMATS. You might want to use the LIBRARY libref for your format catalog. You can control the order in which SAS searches for format catalogs with the FMTSEARCH= system option. For further information about FMTSEARCH=, see the section on SAS system options in *SAS Language Reference: Dictionary*.

See also: “Storing Informats and Formats” on page 466

Featured in: Example 1 on page 474

MAXLABLEN=*number-of-characters*

specifies the number of characters in the informatted or formatted value that you want to appear in the CNTLOUT= data set or in the output of the FMTLIB option. The FMTLIB option prints a maximum of 40 characters for the informatted or formatted value.

MAXSELEN=*number-of-characters*

specifies the number of characters in the start and end values that you want to appear in the CNTLOUT= data set or in the output of the FMTLIB option. The FMTLIB option prints a maximum of 16 characters for start and end values.

NOREPLACE

prevents a new informat or format that you are creating from replacing an existing informat or format of the same name. If you omit NOREPLACE, then the procedure warns you that the informat or format already exists and replaces it.

Note: You can have a format and an informat of the same name. △

PAGE

prints information about each format and informat (that is, each entry) in the catalog on a separate page.

Tip: The PAGE option activates the FMTLIB option.

EXCLUDE Statement

Excludes entries from processing by the FMTLIB and CNTLOUT= options.

Restriction: Only one EXCLUDE statement can appear in a PROC FORMAT step.

Restriction: You cannot use a SELECT statement and an EXCLUDE statement within the same PROC FORMAT step.

EXCLUDE *entry(s)*;

Required Arguments

entry(s)

specifies one or more catalog entries to exclude from processing. Catalog entry names are the same as the name of the informat or format that they store. Because informats and formats can have the same name, and because character and numeric informats or formats can have the same name, you must use certain prefixes when specifying informats and formats in the EXCLUDE statement. Follow these rules when specifying entries in the EXCLUDE statement:

- Precede names of entries that contain character formats with a dollar sign ($).
- Precede names of entries that contain character informats with an at sign and a dollar sign (for example, @$*entry-name*).
- Precede names of entries that contain numeric informats with an at sign (@).
- Specify names of entries that contain numeric formats without a prefix.

Shortcuts to Specifying Names

You can use the colon (:) and hyphen (-) wildcard characters to exclude entries. For example, the following EXCLUDE statement excludes all formats or informats that begin with the letter **a**.

```
exclude a:;
```

In addition, the following EXCLUDE statement excludes all formats or informats that occur alphabetically between **apple** and **pear**, inclusive:

```
exclude apple-pear;
```

FMTLIB Output

If you use the EXCLUDE statement without either FMTLIB or CNTLOUT= in the PROC FORMAT statement, then the procedure invokes FMTLIB.

INVALUE Statement

Creates an informat for reading and converting raw data values.

Featured in: Example 4 on page 480.

See also: The section on informats in *SAS Language Reference: Dictionary* for documentation on informats supplied by SAS.

INVALUE <$>*name* <(*informat-option(s)*)>
<*value-range-set(s)*>;

To do this	Use this option
Specify the default length of the informat	DEFAULT=
Specify a fuzz factor for matching values to a range	FUZZ=
Specify a maximum length for the informat	MAX=
Specify a minimum length for the informat	MIN=
Store values or ranges in the order that you define them	NOTSORTED
Left-justify all input strings before they are compared to ranges	JUST
Uppercase all input strings before they are compared to ranges	UPCASE

Required Arguments

name

names the informat that you are creating.

Requirement: The name must be a valid SAS name. A numeric informat name can be up to 31 characters in length; a character informat name can be up to 30 characters in length and cannot end in a number. If you are creating a character informat, then use a dollar sign ($) as the first character; this is why a character informat is limited to 30 characters.

Restriction: A user-defined informat name cannot be the same as an informat name that is supplied by SAS.

Interaction: The maximum length of an informat name is controlled by the VALIDFMTNAME= SAS system option. See *SAS Language Reference: Dictionary* for details on VALIDFMTNAME=.

Tip: Refer to the informat later by using the name followed by a period. However, do not use a period after the informat name in the INVALUE statement.

Tip: When SAS prints messages that refer to a user-written informat, the name is prefixed by an at sign (@). When the informat is stored, the at sign is prefixed to the name that you specify for the informat; this is why the name is limited to 31 or 30 characters. You need to use the at sign *only* when you are using the name in an EXCLUDE or SELECT statement; do not prefix the name with an at sign when you are associating the informat with a variable.

Options

The following options are common to the INVALUE, PICTURE, and VALUE statements and are described in “Informat and Format Options” on page 462:

DEFAULT=*length*

FUZZ= *fuzz-factor*

MAX=*length*

MIN=*length*

NOTSORTED

In addition, you can use the following options:

JUST
left-justifies all input strings before they are compared to the ranges.

UPCASE
converts all raw data values to uppercase before they are compared to the possible ranges. If you use UPCASE, then make sure the values or ranges you specify are in uppercase.

value-range-set(s)
specifies raw data and values that the raw data will become. The *value-range-set(s)* can be one or more of the following:

value-or-range-1 <..., value-or-range-n>=informatted-value | [*existing-informat*]
The informat converts the raw data to the values of *informatted-value* on the right side of the equal sign.

informatted-value
is the value you want the raw data in *value-or-range* to become. Use one of the following forms for *informatted-value*:

'character-string'
is a character string up to 32,767 characters long. Typically, *character-string* becomes the value of a character variable when you use the informat to convert raw data. Use *character-string* for *informatted-value* only when you are creating a character informat. If you omit the single or double quotation marks around *character-string*, then the INVALUE statement assumes that the quotation marks are there.

For hexadecimal literals, you can use up to 32,767 typed characters, or up to 16,382 represented characters at 2 hexadecimal characters per represented character.

number
is a number that becomes the informatted value. Typically, *number* becomes the value of a numeric variable when you use the informat to convert raw data. Use *number* for *informatted-value* when you are creating a numeric informat. The maximum for *number* depends on the host operating environment.

ERROR
treats data values in the designated range as invalid data. SAS assigns a missing value to the variable, prints the data line in the SAS log, and issues a warning message.

SAME
prevents the informat from converting the raw data as any other value. For example, the following GROUP. informat converts values 01 through 20 and assigns the numbers 1 through 20 as the result. All other values are assigned a missing value.

```
invalue group 01-20= _same_
                 other= .;
```

existing-informat

is an informat that is supplied by SAS or a user-defined informat. The informat you are creating uses the existing informat to convert the raw data that match *value-or-range* on the left side of the equals sign. If you use an existing informat, then enclose the informat name in square brackets (for example, [date9.]) or with parentheses and vertical bars, for example, (|date9.|). *Do not enclose the name of the existing informat in single quotation marks.*

value-or-range

See “Specifying Values or Ranges” on page 464.

Consider the following examples:

- The $GENDER. character informat converts the raw data values **F** and **M** to character values **'1'** and **'2'**:

  ```
  invalue $gender 'F'='1'
                  'M'='2';
  ```

  The dollar sign prefix indicates that the informat converts character data.

- When you are creating numeric informats, you can specify character strings or numbers for *value-or-range*. For example, the TRIAL. informat converts any character string that sorts between **A** and **M** to the number 1 and any character string that sorts between **N** and **Z** to the number 2. The informat treats the unquoted range 1–3000 as a numeric range, which includes all numeric values between 1 and 3000:

  ```
  invalue trial 'A'-'M'=1
                'N'-'Z'=2
                 1-3000=3;
  ```

  If you use a numeric informat to convert character strings that do not correspond to any values or ranges, then you receive an error message.

- The CHECK. informat uses _ERROR_ and _SAME_ to convert values of 1 through 4 and 99. All other values are invalid:

  ```
  invalue check 1-4=_same_
                 99=.
              other=_error_;
  ```

PICTURE Statement

Creates a template for printing numbers.

Featured in: Example 1 on page 474 and Example 9 on page 492

See also: The section on formats in *SAS Language Reference: Dictionary* for documentation on formats supplied by SAS.

PICTURE *name* <(*format-option(s)*)>
  <*value-range-set-1* <(*picture-1-option(s)*)>
  <*…value-range-set-n* <(*picture-n-option(s)*)>>>;

To do this	Use this option
Control the attributes of the format	
Specify the default length of the format	DEFAULT=
Specify a fuzz factor for matching values to a range	FUZZ=
Specify a maximum length for the format	MAX=
Specify a minimum length for the format	MIN=
Specify multiple pictures for a given value or range and for overlapping ranges	MULTILABEL
Store values or ranges in the order that you define them	NOTSORTED
Round the value to the nearest integer before formatting	ROUND
Control the attributes of each picture in the format	
Specify a character that completes the formatted value	FILL=
Specify a number to multiply the variable's value by before it is formatted	MULTIPLIER=
Specify that numbers are message characters rather than digit selectors	NOEDIT
Specify a character prefix for the formatted value	PREFIX=

Required Arguments

name
names the format you are creating.

Requirement: The name must be a valid SAS name. A numeric format name can be up to 32 characters in length; a character format name can be up to 31 characters in length, not ending in a number. If you are creating a character format, then use a dollar sign ($) as the first character, which is why a character informat is limited to 30 characters.

Restriction: A user-defined format cannot be the name of a format supplied by SAS.

Interaction: The maximum length of a format name is controlled by the VALIDFMTNAME= SAS system option. See *SAS Language Reference: Dictionary* for details on VALIDFMTNAME=.

Tip: Refer to the format later by using the name followed by a period. However, do not put a period after the format name in the VALUE statement.

Options

The following options are common to the INVALUE, PICTURE, and VALUE statements and are described in "Informat and Format Options" on page 462:

DEFAULT= *length*

FUZZ= *fuzz-factor*

MAX=*length*

MIN=*length*

NOTSORTED

In addition, you can use the following arguments:

DATATYPE=DATE | TIME | DATETIME
specifies that you can use *directives* in the picture as a template to format date, time, or datetime values. See the definition of directives on page 453 for a list.

DECSEP='*character*'
specifies the separator character for the fractional part of a number.

Default: . (a decimal point)

DIG3SEP='*character*'
specifies the three-digit separator character for a number.

Default: , (a comma)

FILL='*character*'
specifies a character that completes the formatted value. If the number of significant digits is less than the length of the format, then the format must complete, or fill, the formatted value:

- The format uses *character* to fill the formatted value if you specify zeros as digit selectors.
- The format uses zeros to fill the formatted value if you specify nonzero digit selectors. The FILL= option has no effect.

If the picture includes other characters, such as a comma, which appear to the left of the digit selector that maps to the last significant digit placed, then the characters are replaced by the fill character or leading zeros.

Default: ' '(a blank)

Interaction: If you use the FILL= and PREFIX= options in the same picture, then the format places the prefix and then the fill characters.

Featured in: Example 9 on page 492

MULTILABEL
allows the assignment of multiple labels or external values to internal values. The following PICTURE statements show the two uses of the MULTILABEL option. In each case, number formats are assigned as labels. The first PICTURE statement assigns multiple labels to a single internal value. Multiple labels may also be assigned to a single range of internal values. The second PICTURE statement assigns labels to overlapping ranges of internal values. The MULTILABEL option allows the assignment of multiple labels to the overlapped internal values.

```
picture abc (multilabel)
   1000='9,999'
   1000='9999';

picture overlap (multilabel)
   /* without decimals */
   0-999='999'
   1000-9999='9,999'

   /* with decimals */
   0-9='9.999'
   10-99='99.99'
   100-999='999.9';
```

Only multilabel-enabled procedures such as PROC MEANS, PROC SUMMARY, and PROC TABULATE can use multiple labels. All other procedures and the DATA step recognize only the primary label. The *primary label* for a given entry is the external value that is assigned to the first internal value or range of internal values that matches or contains the entry when all internal values are ordered sequentially. For example, in the first PICTURE statement, the primary label for 1000 is 1,000 because the format 9,999 is the first external value that is assigned to 1000. The secondary label for 1000 is 1000, based on the 9999 format.

In the second PICTURE statement, the primary label for 5 is 5.000 based on the 9.999 format that is assigned to the range 0–9 because 0–9 is sequentially the first range of internal values containing 5. The secondary label for 5 is 005 because the range 0–999 occurs in sequence after the range 0–9. Consider carefully when you assign multiple labels to an internal value. Unless you use the NOTSORTED option when you assign variables, the SAS System stores the variables in sorted order. This may produce unexpected results when variables with the MULTILABEL format are processed. For example, in the second PICTURE statement, the primary label for 15 is 015, and the secondary label for 15 is 15.00 because the range 0–999 occurs in sequence before the range 10–99. If you want the primary label for 15 to use the 99.99 format, then you might want to change the range 10–99 to 0–99 in the PICTURE statement. The range 0–99 occurs in sequence before the range 0–999 and will produce the desired result.

MULTIPLIER=*n*

specifies a number that the variable's value is to be multiplied by before it is formatted. For example, the following PICTURE statement creates the MILLION. format, which formats the variable value 1600000 as **`$1.6M`**:

```
picture million low-high='00.0M'
        (prefix='$' mult=.00001);
```

Alias: MULT=

Default: 10^n, where *n* is the number of digits after the first decimal point in the picture. For example, suppose your data contains a value 123.456 and you want to print it using a picture of '999.999'. The format multiplies 123.456 by 10^3 to obtain a value of 123456, which results in a formatted value of **`123.456`**.

Example: Example 1 on page 474

NOEDIT

specifies that numbers are message characters rather than digit selectors; that is, the format prints the numbers as they appear in the picture. For example, the following PICTURE statement creates the MILES. format, which formats any variable value greater than 1000 as **`>1000 miles`**:

```
picture miles 1-1000='0000'
           1000<-high='>1000 miles'(noedit);
```

PREFIX='*prefix*'

specifies a character prefix to place in front of the value's first significant digit. You must use zero digit selectors or the prefix will not be used.

The picture must be wide enough to contain both the value and the prefix. If the picture is not wide enough to contain both the value and the prefix, then the format truncates or omits the prefix. Typical uses for PREFIX= are printing leading currency symbols and minus signs. For example, the PAY. format prints the variable value 25500 as **`$25,500.00`**:

```
picture pay low-high='000,009.99'
                     (prefix='$');
```

Default: no prefix

Interaction: If you use the FILL= and PREFIX= options in the same picture, then the format places the prefix and then the fill characters.

Featured in: Example 1 on page 474 and Example 9 on page 492

ROUND
rounds the value to the nearest integer *before* formatting. Without the ROUND option, the format multiplies the variable value by the multiplier, truncates the decimal portion (if any), and prints the result according to the template that you define. With the ROUND option, the format multiplies the variable value by the multiplier, rounds that result to the nearest integer, and then formats the value according to the template. Note that if the FUZZ= option is also specified, the rounding takes place after SAS has used the fuzz factor to determine which range the value belongs to.

Tip: Note that the ROUND option rounds a value of .5 to the next highest integer.

value-range-set
specifies one or more variable values and a template for printing those values. The *value-range-set* is the following:

value-or-range-1 <..., value-or-range-n>='picture'

picture
specifies a template for formatting values of numeric variables. The picture is a sequence of characters in single quotation marks. The maximum length for a picture is 40 characters. Pictures are specified with three types of characters: digit selectors, message characters, and directives. You can have a maximum of 16 digit selectors in a picture.

Digit selectors are numeric characters (0 through 9) that define positions for numeric values. A picture format with nonzero digit selectors prints any leading zeros in variable values; picture digit selectors of 0 do not print leading zeros in variable values. If the picture format contains digit selectors, then a digit selector must be the first character in the picture.

Note: This chapter uses 9's as nonzero digit selectors. △

Message characters are nonnumeric characters that print as specified in the picture. The following PICTURE statement contains both digit selectors (99) and message characters (**`illegal day value`**). Because the DAYS. format has nonzero digit selectors, values are printed with leading zeros. The special range OTHER prints the message characters for any values that do not fall into the specified range (1 through 31).

```
picture days 01-31='99'
       other='99-illegal day value';
```

For example, the values 02 and 67 print as

```
                     02
   67-illegal day value
```

Directives are special characters that you can use in the picture to format date, time, or datetime values.

Restriction: You can only use directives when you specify the DATATYPE= option in the PICTURE statement.

The permitted directives are

%a	Locale's abbreviated weekday name
%A	Locale's full weekday name
%b	Locale's abbreviated month name
%B	Locale's full month name

%d	Day of the month as a decimal number (1–31), with no leading zero
%H	Hour (24-hour clock) as a decimal number (0–23), with no leading zero
%I	Hour (12-hour clock) as a decimal number (1–12), with no leading zero
%j	Day of the year as a decimal number (1–366), with no leading zero
%m	Month as a decimal number (1–12), with no leading zero
%M	Minute as a decimal number (0–59), with no leading zero
%p	Locale's equivalent of either AM or PM
%S	Second as a decimal number (0–59), with no leading zero
%U	Week number of the year (Sunday as the first day of the week) as a decimal number (0,53), with no leading zero
%w	Weekday as a decimal number (1= Sunday, 7=Saturday)
%y	Year without century as a decimal number (0–99), with no leading zero
%Y	Year with century as a decimal number
%%	%

Any directive that generates numbers can produce a leading zero, if desired, by adding a 0 before the directive. This applies to %d, %H, %I, %j, %m, %M, %S, %U, and %y. For example, if you specify %y in the picture, then 2001 would be formatted as '1', but if you specify %0y, then 2001 would be formatted as '01'.

value-or-range

See "Specifying Values or Ranges" on page 464.

Building a Picture Format: Step by Step

This section shows how to write a picture format for formatting numbers with leading zeros. In the SAMPLE data set, the default printing of the variable Amount has leading zeros on numbers between 1 and –1:

```
options nodate pageno=1 linesize=64
        pagesize=60;
data sample;
   input Amount;
   datalines;
-2.05
-.05
-.01
  0
 .09
 .54
 .55
6.6
14.63
;
```

```
             Default Printing of the Variable Amount           1

                          Obs    Amount

                           1      -2.05
                           2      -0.05
                           3      -0.01
                           4       0.00
                           5       0.09
                           6       0.54
                           7       0.55
                           8       6.60
                           9      14.63
```

The following PROC FORMAT step creates the NOZEROS. format, which eliminates leading zeros in the formatted values:

```
libname library 'SAS-data-library';


proc format library=library;
   picture nozeros
           low -  -1  =  '00.00'
              (prefix='-')
           -1 <-<  0  =     '99'
              (prefix='-.' mult=100)
            0  -< 1   =     '99'
              (prefix='.'  mult=100)
            1  - high =  '00.00';
run;
```

The following table explains how one value from each range is formatted. Figure 21.1 on page 457 provides an illustration of each step. The circled numbers in the figure correspond to the step numbers in the table.

Table 21.1 Building a Picture Format

Step	Rule	In this example
1	Determine into which range the value falls and use that picture.	In the second range, the exclusion operator < appears on both sides of the hyphen and excludes –1 and 0 from the range.
2	Take the absolute value of the numeric value.	Because the absolute value is used, you need a separate range and picture for the negative numbers in order to prefix the minus sign.

Step	Rule	In this example
3	Multiply the number by the MULT= value. If you do not specify the MULT= option, then the PICTURE statement uses the default. The default is 10^n , where *n* is the number of digit selectors to the right of the decimal[1] in the picture. (Step 6 discusses digit selectors further.)	Specifying a MULT= value is necessary for numbers between 0 and 1 and numbers between 0 and –1 because no decimal appears in the pictures for those ranges. Because MULT= defaults to 1, truncation of the significant digits results without a MULT= value specified. (Truncation is explained in the next step.) For the two ranges that do not have MULT= values specified, the MULT= value defaults to 100 because the corresponding picture has two digit selectors to the right of the decimal. After the MULT= value is applied, all significant digits are moved to the left of the decimal.
4	Truncate the number after the decimal. If the ROUND option is in effect, then the format rounds the number after the decimal to the next highest integer if the number after the decimal is greater than or equal to .5.	Because the example uses MULT= values that ensured that all of the significant digits were moved to the left of the decimal, no significant digits are lost. The zeros are truncated.
5	Turn the number into a character string. If the number is shorter than the picture, then the length of the character string is equal to the number of digit selectors in the picture. Pad the character string with leading zeros. (The results are equivalent to using the Z*w*. format. Z*w*. is explained in the section on SAS formats in *SAS Language Reference: Dictionary*.	The numbers 205, 5, and 660 become the character strings **0205**, **05**, and **0660**, respectively. Because each picture is longer than the numbers, the format adds a leading zero to each value. The format does not add leading zeros to the number 55 because the corresponding picture only has two digit selectors.

Step	Rule	In this example
6	Apply the character string to the picture. The format only maps the rightmost *n* characters in the character string, where *n* is the number of digit selectors in the picture. Thus, it is important to make sure that the picture has enough digit selectors to accommodate the characters in the string. After the format takes the rightmost *n* characters, it then maps those characters to the picture from left to right. Choosing a zero or nonzero digit selector is important if the character string contains leading zeros. If one of the leading zeros in the character string maps to a nonzero digit selector, then it and all subsequent leading zeros become part of the formatted value. If all of the leading zeros map to zero digit selectors, then none of the leading zeros become part of the formatted value; the format replaces the leading zeros in the character string with blanks.[2]	The leading zero is dropped from each of the character strings **`0205`** and **`0660`** because the leading zero maps to a zero digit selector in the picture.
7	Prefix any characters that are specified in the PREFIX= option. You need the PREFIX= option because when a picture contains any digit selectors, the picture must begin with a digit selector. Thus, you cannot begin your picture with a decimal point, minus sign, or any other character that is not a digit selector.	The PREFIX= option reclaims the decimal point and the negative sign, as shown with the formatted values **`-.05`** and **`.55`**.

1 A decimal in a PREFIX= option is not part of the picture.
2 You can use the FILL= option to specify a character other than a blank to become part of the formatted value.

Figure 21.1 Formatting One Value in Each Range

		-2.05	-.05	.55	6.6
❶	range	`low - -1`	`-1 <-< 0`	`0 -< 1`	`1 - high`
❶	picture	00.00	99	99	00.00
❷	absolute value	2.05	.05	.55	6.6
❸	MULT=	$2.05 \times 10^2=$ 205.000	.05 X 100= 5.000	.55 X 100= 55.000	$6.6 \times 10^2=$ 660.000
❹	truncation	205	5	55	660
❺	character string	`0205`	`05`	`55`	`0660`
❻	template	[ ][2].[0][5]	[0][5]	[5][5]	[ ][6].[6][0]
❼	prefix	prefix = '-'	prefix = '-.'	prefix = '.'	none
	formatted result	-2.05	-.05	.55	6.60

The following PROC PRINT step associates the NOZEROS. format with the AMOUNT variable in SAMPLE:

```
proc print data=sample noobs;
   format amount nozeros.;
   title 'Formatting the Variable Amount';
   title2 'with the NOZEROS. Format';
run;
```

```
                 Formatting the Variable Amount              1
                    with the NOZEROS. Format

                             Amount

                             -2.05
                              -.05
                              -.01
                               .00
                               .09
                               .54
                               .55
                              6.60
                             14.63
```

CAUTION:

The picture must be wide enough for the prefix and the numbers. In this example, if the value –45.00 were formatted with NOZEROS. then the result would be 45.00 because it falls into the first range, low - –1, and the picture for that range is not wide enough to accommodate the prefixed minus sign and the number. △

Specifying No Picture

This PICTURE statement creates a *picture-name* format that has no picture:

picture *picture-name*;

Using this format has the effect of applying the default SAS format to the values.

SELECT Statement

Selects entries from processing by the FMTLIB and CNTLOUT= options.

Restriction: Only one SELECT statement can appear in a PROC FORMAT step.

Restriction: You cannot use a SELECT statement and an EXCLUDE statement within the same PROC FORMAT step.

Featured in: Example 6 on page 486.

SELECT *entry(s)*;

Required Arguments

entry(s)
specifies one or more catalog entries for processing. Catalog entry names are the same as the name of the informat or format that they store. Because informats and formats can have the same name, and because character and numeric informats or formats can have the same name, you must use certain prefixes when specifying informats and formats in the SELECT statement. Follow these rules when specifying entries in the SELECT statement:

- Precede names of entries that contain character formats with a dollar sign ($).
- Precede names of entries that contain character informats with an at sign and a dollar sign, for example, @$*entry-name*.
- Precede names of entries that contain numeric informats with an at sign (@).
- Specify names of entries that contain numeric formats without a prefix.

Shortcuts to Specifying Names

You can use the colon (:) and hyphen (-) wildcard characters to select entries. For example, the following SELECT statement selects all formats or informats that begin with the letter **a**.

```
select a:;
```

In addition, the following SELECT statement selects all formats or informats that occur alphabetically between **apple** and **pear**, inclusive:

```
select apple-pear;
```

FMTLIB Output

If you use the SELECT statement without either FMTLIB or CNTLOUT= in the PROC FORMAT statement, then the procedure invokes FMTLIB.

VALUE Statement

Creates a format that specifies character strings to use to print variable values.

Featured in: Example 2 on page 476.

See also: The chapter on formats in *SAS Language Reference: Dictionary* for documentation on formats supplied by SAS.

VALUE <$>*name* <(*format-option(s)*)>
<*value-range-set(s)*>;

To do this	Use this option
Specify the default length of the format	DEFAULT=
Specify a fuzz factor for matching values to a range	FUZZ=
Specify a maximum length for the format	MAX=

To do this	Use this option
Specify a minimum length for the format	MIN=
Specify multiple values for a given range, or for overlapping ranges	MULTILABEL
Store values or ranges in the order that you define them.	NOTSORTED

Required Arguments

name
names the format that you are creating.

Requirement: The name must be a valid SAS name. A numeric format name can be up to 32 characters in length; a character format name can be up to 31 characters in length and cannot end in a number. If you are creating a character format, then use a dollar sign ($) as the first character; this is why a character informat is limited to 30 characters.

Restriction: The name of a user-defined format cannot be the same as the name of a format that is supplied by SAS.

Interaction: The maximum length of a format name is controlled by the VALIDFMTNAME= SAS system option. See *SAS Language Reference: Dictionary* for details on VALIDFMTNAME=.

Tip: Refer to the format later by using the name followed by a period. However, do not use a period after the format name in the VALUE statement.

Options

The following options are common to the INVALUE, PICTURE, and VALUE statements and are described in "Informat and Format Options" on page 462:

DEFAULT=*length*

FUZZ= *fuzz-factor*

MAX=*length*

MIN=*length*

NOTSORTED

In addition, you can use the following options:

MULTILABEL
allows the assignment of multiple labels or external values to internal values. The following VALUE statements show the two uses of the MULTILABEL option. The first VALUE statement assigns multiple labels to a single internal value. Multiple labels may also be assigned to a single range of internal values. The second VALUE statement assigns labels to overlapping ranges of internal values. The MULTILABEL option allows the assignment of multiple labels to the overlapped internal values.

```
value one (multilabel)
   1='ONE'
   1='UNO'
   1='UN'

value agefmt (multilabel)
   15-29='below 30 years'
```

```
30-50='between 30 and 50'
51-high='over 50 years'
15-19='15 to 19'
20-25='20 to 25'
25-39='25 to 39'
40-55='40 to 55'
56-high='56 and above';
```

Only multilabel-enabled procedures such as PROC MEANS, PROC SUMMARY, and PROC TABULATE can use multiple labels. All other procedures and the data step recognize only the primary label. The *primary label* for a given entry is the external value that is assigned to the first internal value or range of internal values that matches or contains the entry when all internal values are ordered sequentially. For example, in the first VALUE statement, the primary label for 1 is ONE because ONE is the first external value that is assigned to 1. The secondary labels for 1 are UNO and UN. In the second VALUE statement, the primary label for 33 is **`25 to 39`** because the range 25–39 is sequentially the first range of internal values that contains 33. The secondary label for 33 is **`between 30 and 50`** because the range 30–50 occurs in sequence after the range 25–39.

value-range-set(s)

specifies one or more variable values and a character string or an existing format. The *value-range-set(s)* can be one or more of the following:

value-or-range-1 <..., value-or-range-n>='formatted-value' | [*existing-format*]

The variable values on the left side of the equals sign print as the character string on the right side of the equals sign.

formatted-value

specifies a character string that becomes the printed value of the variable value that appears on the left side of the equals sign. Formatted values are always character strings, regardless of whether you are creating a character or numeric format.

Formatted values can be up to 32,767 characters. For hex literals, you can use up to 32,767 typed characters, or up to 16,382 represented characters at 2 hex characters per represented character. Some procedures, however, use only the first 8 or 16 characters of a formatted value.

If you omit the single quotation marks around *formatted-value*, then the VALUE statement assumes them to be there.

If a formatted value contains a single quotation mark, then enclose the value in double quotation marks:

```
value sect 1="Smith's class"
           2="Leung's class";
```

Tip: Formatting numeric variables does not preclude the use of those variables in arithmetic operations. SAS uses stored values for arithmetic operations.

existing-format

specifies a format supplied by SAS or an existing user-defined format. The format you are creating uses the existing format to convert the raw data that match *value-or-range* on the left side of the equals sign.

If you use an existing format, then enclose the format name in square brackets (for example, [date9.]) or with parentheses and vertical bars, for example, (|date9.|). *Do not enclose the name of the existing format in single quotation marks.*

Using an existing format can be thought of as *nesting* formats. A nested level of one means that if you are creating the format A with the format B as a formatted value, then the procedure has to use only one existing format to create A.

Tip: Avoid nesting formats more than one level. The resource requirements can increase dramatically with each additional level.

value-or-range

For details on how to specify *value-or-range*, see "Specifying Values or Ranges" on page 464.

Consider the following examples:

- The $STATE. character format prints the postal code for selected states:

```
value $state 'Delaware'='DE'
             'Florida'='FL'
                'Ohio'='OH';
```

The variable value **Delaware** prints as **DE**, the variable value **Florida** prints as **FL**, and the variable value **Ohio** prints as **OH**. Note that the $STATE. format begins with a dollar sign.

Note: Range specifications are case sensitive. In the $STATE. format above, the value **OHIO** would not match any of the specified ranges. If you are not certain what case the data values are in, then one solution is to use the UPCASE function on the data values and specify all uppercase characters for the ranges. △

- The numeric format ANSWER.writes the values 1 and 2 as **yes** and **no**:

```
value answer 1='yes'
             2='no';
```

Specifying No Ranges

This VALUE statement creates a *format-name* format that has no ranges:

```
value format-name;
```

Using this format has the effect of applying the default SAS format to the values.

Informat and Format Options

This section discusses options that are valid in the INVALUE, PICTURE, and VALUE statements. These options appear in parentheses after the informat or format name. They affect the entire informat or format that you are creating.

DEFAULT=*length*

specifies the default length of the informat or format. The value for DEFAULT= becomes the length of the informat or format if you do not give a specific length when you associate the informat or format with a variable.

The default length of a format is the length of the longest formatted value.

The default length of an informat depends on whether the informat is character or numeric. The default length of character informats is the length of the longest informatted value. The default of a numeric informat is 12 if you have numeric data to the left of the equals sign. If you have a quoted string to the left of the equals sign, then the default length is the length of the longest string.

FUZZ=*fuzz-factor*

specifies a fuzz factor for matching values to a range. If a number does not match or fall in a range exactly but comes within *fuzz-factor*, then the format considers it a match. For example, the following VALUE statement creates the LEVELS. format, which uses a fuzz factor of .2:

```
value levels (fuzz=.2) 1='A'
                       2='B'
                       3='C';
```

FUZZ=.2 means that if a variable value falls within .2 of a value on either end of the range, then the format uses the corresponding formatted value to print the variable value. So the LEVELS. format formats the value 2.1 as `B`.

If a variable value matches one value or range without the fuzz factor, and also matches another value or range with the fuzz factor, then the format assigns the variable value to the value or range that it matched without the fuzz factor.

Default: 1E–12 for numeric formats and 0 for character formats.

Tip: Specify FUZZ=0 to save storage space when you use the VALUE statement to create numeric formats.

Tip: A value that is excluded from a range using the < operator does not receive the formatted value, even if it falls into the range when you use the fuzz factor.

MAX=*length*

specifies a maximum length for the informat or format. When you associate the format with a variable, you cannot specify a width greater than the MAX= value.

Default: 40

Range: 1–40

MIN=*length*

specifies a minimum length for the informat or format.

Default: 1

Range: 1–40

NOTSORTED

stores values or ranges for informats or formats in the order in which you define them. If you do not specify NOTSORTED, then values or ranges are stored in sorted order by default, and SAS uses a binary searching algorithm to locate the range that a particular value falls into. If you specify NOTSORTED, then SAS searches each range in the order in which you define them until a match is found.

Use NOTSORTED if

- □ you know the likelihood of certain ranges occurring, and you want your informat or format to search those ranges first to save processing time.
- □ you want to preserve the order that you define ranges when you print a description of the informat or format using the FMTLIB option.
- □ you want to preserve the order that you define ranges when you use the ORDER=DATA option and the PRELOADFMT option to analyze class variables in PROC MEANS, PROC SUMMARY, or PROC TABULATE.

Do not use NOTSORTED if the distribution of values is uniform or unknown, or if the number of values is relatively small. The binary searching algorithm that SAS uses when NOTSORTED is not specified optimizes the performance of the search under these conditions.

Note: SAS automatically sets the NOTSORTED option when you use the CPORT and the CIMPORT procedures to transport informats or formats between operating environments with different standard collating sequences. This automatic setting of NOTSORTED can occur when you transport informats or formats between ASCII and EBCDIC operating environments. If this situation is undesirable, then do the following:

1. Use the CNTLOUT= option in the PROC FORMAT statement to create an output control data set.

2 Use the CPORT procedure to create a transport file for the control data set.
3 Use the CIMPORT procedure in the target operating environment to import the transport file.
4 In the target operating environment, use PROC FORMAT with the CNTLIN= option to build the formats and informats from the imported control data set.

△

Specifying Values or Ranges

As the syntax of the INVALUE, PICTURE, and VALUE statements indicates, you must specify values as *value-range-sets*. On the left side of the equals sign you specify the values that you want to convert to other values. On the right side of the equals sign, you specify the values that you want the values on the left side to become. This section discusses the different forms that you can use for *value-or-range*, which represents the values on the left side of the equals sign. For details on how to specify values for the right side of the equals sign, see the "Required Arguments" section for the appropriate statement.

The INVALUE, PICTURE, and VALUE statements accept numeric values on the left side of the equals sign. INVALUE and VALUE also accept character strings on the left side of the equals sign.

As the syntax shows, you can have multiple occurrences of *value-or-range* in each *value-range-set*, with commas separating the occurrences. Each occurrence of *value-or-range* is either one of the following:

value
: a single value, such as 12 or **'CA'**. For character formats and informats, enclose the character values in single quotation marks. If you omit the quotation marks around *value*, then PROC FORMAT assumes the quotation marks to be there.

  You can use the keyword OTHER as a single value. OTHER matches all values that do not match any other value or range.

range
: a list of values, for example, 12–68 or **'A'-'Z'**. For ranges with character strings, be sure to enclose each string in single quotation marks. For example, if you want a range that includes character strings from A to Z, then specify the range as **'A'-'Z'**, with single quotation marks around the **A** and around the **Z**.

  If you specify **'A-Z'**, then the procedure interprets it as a three-character string with **A** as the first character, a hyphen (-) as the second character, and a **Z** as the third character.

  If you omit the quotation marks, then the procedure assumes quotation marks around each string. For example, if you specify the range **abc-zzz**, then the procedure interprets it as **'abc'-'zzz'**.

  You can use LOW or HIGH as one value in a range, and you can use the range LOW-HIGH to encompass all values. For example, these are valid ranges:

```
low-'ZZ'
35-high
low-high
```

  You can use the less than (<) symbol to exclude values from ranges. If you are excluding the first value in a range, then put the < after the value. If you are excluding the last value in a range, then put the < before the value. For example, the following range does not include 0:

```
0<-100
```

Likewise, the following range does not include 100:

```
0-<100
```

If a value at the high end of one range also appears at the low end of another range, and you do not use the < noninclusion notation, then PROC FORMAT assigns the value to the first range. For example, in the following ranges, the value **AJ** is part of the first range:

```
'AA'-'AJ'=1 'AJ'-'AZ'=2
```

In this example, to include the value **AJ** in the second range, use the noninclusive notation on the first range:

```
'AA'-<'AJ'=1 'AJ'-'AZ'=2
```

If you overlap values in ranges, then PROC FORMAT returns an error message unless, for the VALUE statement, the MULTILABEL option is specified. For example, the following ranges will cause an error:

```
'AA'-'AK'=1 'AJ'-'AZ=2
```

Each *value-or-range* can be up to 32,767 characters. If *value-or-range* has more than 32,767 characters, then the procedure truncates the value after it processes the first 32,767 characters.

Note: You do not have to account for every value on the left side of the equals sign. Those values are converted using the default informat or format. For example, the following VALUE statement creates the TEMP. format, which prints all occurrences of 98.6 as **NORMAL**:

```
value temp 98.6='NORMAL';
```

If the value were 96.9, then the printed result would be **96.9**. △

Concepts: FORMAT Procedure

Associating Informats and Formats with Variables

Table 21.2 on page 466 summarizes the different methods for associating informats and formats with variables.

Table 21.2 Associating Informats and Formats with Variables

Step	Informats	Formats
In a DATA step	Use the ATTRIB or INFORMAT statement to permanently associate an informat with a variable. Use the INPUT function or INPUT statement to associate the informat with the variable only for the duration of the DATA step.	Use the ATTRIB or FORMAT statement to permanently associate a format with a variable. Use the PUT function or PUT statement to associate the format with the variable only for the duration of the DATA step.
In a PROC step	The ATTRIB and INFORMAT statements are valid in base SAS procedures. However, in base SAS software, typically you do not assign informats in PROC steps because the data has already been read into SAS variables.	Use the ATTRIB statement or the FORMAT statement to associate formats with variables. If you use either statement in a procedure that produces an output data set, then the format is permanently associated with the variable in the output data set. If you use either statement in a procedure that does not produce an output data set or modify an existing data set, the statement associates the format with the variable only for the duration of the PROC step.

Tips

- Do not confuse the FORMAT statement with the FORMAT procedure. The FORMAT and INFORMAT statements associate an existing format or informat (either standard SAS or user-defined) with one or more variables. PROC FORMAT creates user-defined formats or informats. Assigning your own format or informat to a variable is a two-step process: creating the format or informat with the FORMAT procedure, and then assigning the format or informat with the FORMAT, INFORMAT, or ATTRIB statement.
- It is often useful to assign informats in the FSEDIT procedure in SAS/FSP software and in the BUILD procedure in SAS/AF software.

See Also

- For complete documentation on the ATTRIB, INFORMAT, and FORMAT statements, see the section on statements in *SAS Language Reference: Dictionary*.
- For complete documentation on the INPUT and PUT functions, see the section on functions in *SAS Language Reference: Dictionary*.
- See “Formatted Values” on page 25 for more information and examples of using formats in base SAS procedures.

Storing Informats and Formats

Format Catalogs

PROC FORMAT stores user-defined informats and formats as entries in SAS catalogs.* You use the LIBRARY= option in the PROC FORMAT statement to specify the catalog. If you omit the LIBRARY= option, then formats and informats are stored in the WORK.FORMATS catalog. If you specify LIBRARY=*libref* but do not specify a catalog name, then formats and informats are stored in the *libref*.FORMATS catalog. Note that this use of a one-level name differs from the use of a one-level name elsewhere in SAS. With the LIBRARY= option, a one-level name indicates a library; elsewhere in SAS, a one-level name indicates a file in the WORK library.

The name of the catalog entry is the name of the format or informat. The entry types are

- FORMAT for numeric formats
- FORMATC for character formats
- INFMT for numeric informats
- INFMTC for character informats.

Temporary Informats and Formats

Informats and formats are temporary when they are stored in a catalog in the WORK library. If you omit the LIBRARY= option, then PROC FORMAT stores the informats and formats in the temporary catalog WORK.FORMATS. You can retrieve temporary informats and formats only in the same SAS session or job in which they are created. To retrieve a temporary format or informat, simply include the name of the format or informat in the appropriate SAS statement. SAS automatically looks for the format or informat in the WORK.FORMATS catalog.

Permanent Informats and Formats

If you want to use a format or informat that is created in one SAS job or session in a subsequent job or session, then you must permanently store the format or informat in a SAS catalog.

You permanently store informats and formats by using the LIBRARY= option in the PROC FORMAT statement. See the discussion of the LIBRARY= option in “PROC FORMAT Statement” on page 443.

Accessing Permanent Informats and Formats

After you have permanently stored an informat or format, you can use it in later SAS sessions or jobs. If you associate permanent informats or formats with variables in a later SAS session or job, then SAS must be able to access the informats and formats. Thus, you must use a LIBNAME statement to assign a libref to the library that stores the catalog that stores the informats or formats.

SAS uses one of two methods when searching for user-defined formats and informats:

- By default, SAS always searches a library that is referenced by the LIBRARY libref for a FORMATS catalog. If you have only one format catalog, then you should do the following:
    1. Assign the LIBRARY libref to a SAS data library in the SAS session in which you are running the PROC FORMAT step.

* Catalogs are a type of SAS file and reside in a SAS data library. If you are unfamiliar with the types of SAS files or the SAS data library structure, then see the section on SAS files in *SAS Language Reference: Concepts*.

2 Specify LIBRARY=LIBRARY in the PROC FORMAT statement. PROC FORMAT will store the informats and formats that are defined in that step in the LIBRARY.FORMATS catalog.

3 In the SAS program that uses your user-defined formats and informats, include a LIBNAME statement to assign the LIBRARY libref to the library that contains the permanent format catalog.

- If you have more than one format catalog, or if the format catalog is named something other than FORMATS, then you should do the following:
  1. Assign a libref to a SAS data library in the SAS session in which you are running the PROC FORMAT step.
  2. Specify LIBRARY=*libref* or LIBRARY=*libref.catalog* in the PROC FORMAT step, where *libref* is the libref that you assigned in step 1.
  3. In the SAS program that uses your user-defined formats and informats, use the FMTSEARCH= option in an OPTIONS statement, and include *libref* or *libref.catalog* in the list of format catalogs.

The syntax for specifying a list of format catalogs to search is

OPTIONS FMTSEARCH=(*catalog-specification-1*<*… catalog-specification-n*>);

where each *catalog-specification* can be *libref* or *libref.catalog*. If only *libref* is specified, then SAS assumes that the catalog name is FORMATS.

When searching for a format or informat, SAS always searches in WORK.FORMATS first, and then LIBRARY.FORMATS, unless one of them appears in the FMTSEARCH= list. SAS searches the catalogs in the FMTSEARCH= list in the order that they are listed until the format or informat is found.

For further information on FMTSEARCH=, see the section on SAS system options in *SAS Language Reference: Dictionary*. For an example that uses the LIBRARY= and FMTSEARCH= options together, see Example 8 on page 490.

Missing Formats and Informats

If you reference an informat or format that SAS cannot find, then you receive an error message and processing stops unless the SAS system option NOFMTERR is in effect. When NOFMTERR is in effect, SAS uses the *w.* or $*w.* default format to print values for variables with formats that it cannot find. For example, to use NOFMTERR, use this OPTIONS statement:

```
options nofmterr;
```

Refer to the section on SAS system options in *SAS Language Reference: Dictionary* for more information on NOFMTERR.

Results: FORMAT Procedure

Output Control Data Set

The output control data set contains information that describes informats or formats. Output control data sets have a number of uses. For example, an output control data set can be edited with a DATA step to programmatically change value ranges or can be subset with a DATA step to create new formats and informats. Additionally, you can move formats and informats from one operating environment to another by creating an output control data set, using the CPORT procedure to create a transfer file of the data set, and then using the CIMPORT and FORMAT procedures in the target operating environment to create the formats and informats there.

You create an output control data set with the CNTLOUT= option in the PROC FORMAT statement. You use output control data sets, or a set of observations from an output control data set, as an input control data set in a subsequent PROC FORMAT step with the CNTLIN= option.

Output control data sets contain an observation for every value or range in each of the informats or formats in the LIBRARY= catalog. The data set consists of variables that give either global information about each format and informat created in the PROC FORMAT step or specific information about each range and value.

The variables in the output control data set are

DEFAULT
: a numeric variable that indicates the default length for format or informat

END
: a character variable that gives the range's ending value

EEXCL
: a character variable that indicates whether the range's ending value is excluded. Values are

Y	the range's ending value is excluded
N	the range's ending value is not excluded

FILL
: for picture formats, a numeric variable whose value is the value of the FILL= option

FMTNAME
: a character variable whose value is the format or informat name

FUZZ
: a numeric variable whose value is the value of the FUZZ= option

HLO
: a character variable that contains range information about the format or informat in the form of eight different letters that can appear in any combination. Values are

F	standard SAS format or informat used for formatted value or informatted value
H	range's ending value is HIGH
I	numeric informat range (informat defined with unquoted numeric range)
L	range's starting value is LOW
N	format or informat has no ranges, including no OTHER= range
O	range is OTHER
M	MULTILABEL option is in effect
R	ROUND option is in effect
S	NOTSORTED option is in effect

LABEL

a character variable whose value is the informatted or formatted value or the name of an existing informat or format

LENGTH
: a numeric variable whose value is the value of the LENGTH= option

MAX
: a numeric variable whose value is the value of the MAX= option

MIN
: a numeric variable whose value is the value of the MIN= option

MULT
: a numeric variable whose value is the value of the MULT= option

NOEDIT
: for picture formats, a numeric variable whose value indicates whether the NOEDIT option is in effect. Values are

1	NOEDIT option is in effect
0	NOEDIT option is not in effect

PREFIX
: for picture formats, a character variable whose value is the value of the PREFIX= option

SEXCL
: a character variable that indicates whether the range's starting value is excluded. Values are

Y	the range's starting value is excluded
N	the range's starting value is not excluded

START
: a character variable that gives the range's starting value

TYPE
: a character variable that indicates the type of format. Possible values are

C	character format
I	numeric informat
J	character informat
N	numeric format (excluding pictures)
P	picture format

Output 21.1 on page 470 shows an output control data set that contains information on all the informats and formats created in "Examples: FORMAT Procedure" on page 474.

Output 21.1 Output Control Data Set for PROC FORMAT Examples

```
                                           An Output Control Data Set                                               1

                                                                                                               D L
   F                                                                    D                                    D A A
   M                                                                    E  L        P           N          D I T N
   T              S                                   L                 F  E        R           O   S E    E G A G
   N              T                                   A                 A  N    F   E       M F E T E E    C 3 T U
 O A              A                E                  B           M  M  U  G    U   F       U I D Y X X H  S S Y A
 b M              R                N                  E           I  A  L  T    Z   I       L L I P C C L  E E P G
 s E              T                D                  L           N  X  T  H    Z   X       T L T E L L O  P P E E

 1 BENEFIT LOW                          7304 WORDDATE20.          1 40 20 20  1E-12      0.00   0 N N N LF
 2 BENEFIT             7305 HIGH               ** Not Eligible ** 1 40 20 20  1E-12      0.00   0 N N N H
 3 NOZEROS LOW                            -1 00.00                1 40  5  5  1E-12 -  100.00   0 P N N L  . ,
 4 NOZEROS               -1                0 99                   1 40  5  5  1E-12 -. 100.00   0 P Y Y    . ,
 5 NOZEROS                0                1 99                   1 40  5  5  1E-12 .  100.00   0 P N Y    . ,
 6 NOZEROS                1 HIGH             00.00                1 40  5  5  1E-12    100.00   0 P N N H  . ,
 7 PTSFRMT                0                3 0%                   1 40  3  3  1E-12      0.00   0 N N N
 8 PTSFRMT                4                6 3%                   1 40  3  3  1E-12      0.00   0 N N N
 9 PTSFRMT                7                8 6%                   1 40  3  3  1E-12      0.00   0 N N N
10 PTSFRMT                9               10 8%                   1 40  3  3  1E-12      0.00   0 N N N
11 PTSFRMT               11 HIGH             10%                  1 40  3  3  1E-12      0.00   0 N N N H
12 USCURR  LOW              HIGH             000,000              1 40  7  7  1E-12 $    1.61   0 P N N LH . ,
13 CITY    BR1              BR1              Birmingham UK        1 40 14 14      0      0.00   0 C N N
14 CITY    BR2              BR2              Plymouth UK          1 40 14 14      0      0.00   0 C N N
15 CITY    BR3              BR3              York UK              1 40 14 14      0      0.00   0 C N N
16 CITY    US1              US1              Denver USA           1 40 14 14      0      0.00   0 C N N
17 CITY    US2              US2              Miami USA            1 40 14 14      0      0.00   0 C N N
18 CITY    **OTHER**        **OTHER**        INCORRECT CODE       1 40 14 14      0      0.00   0 C N N O
19 EVAL    C                C                                   1 1 40  1  1      0      0.00   0 I N N
20 EVAL    E                E                                   2 1 40  1  1      0      0.00   0 I N N
21 EVAL    N                N                                   0 1 40  1  1      0      0.00   0 I N N
22 EVAL    O                O                                   4 1 40  1  1      0      0.00   0 I N N
23 EVAL    S                S                                   3 1 40  1  1      0      0.00   0 I N N
```

You can use the SELECT or EXCLUDE statement to control which formats and informats are represented in the output control data set. For details, see "SELECT Statement" on page 458 and "EXCLUDE Statement" on page 446.

Input Control Data Set

You specify an input control data set with the CNTLIN= option in the PROC FORMAT statement. The FORMAT procedure uses the data in the input control data set to construct informats and formats. Thus, you can create informats and formats without writing INVALUE, PICTURE, or VALUE statements.

The input control data set must have these characteristics:

- For both numeric and character formats, the data set must contain the variables FMTNAME, START, and LABEL, which are described in "Output Control Data Set" on page 468. The remaining variables are not always required.
- If you are creating a character format or informat, then you must either begin the format or informat name with a dollar sign ($) or specify a TYPE variable with the value **C**.
- If you are creating a PICTURE statement format, then you must specify a TYPE variable with the value **P**.
- If you are creating a format with ranges of input values, then you must specify the END variable. If range values are to be noninclusive, then the variables SEXCL and EEXCL must each have a value of **Y**. Inclusion is the default.

You can create more than one format from an input control data set if the observations for each format are grouped together.

You can use a VALUE, INVALUE, or PICTURE statement in the same PROC FORMAT step with the CNTLIN= option. If the VALUE, INVALUE, or PICTURE statement is creating the same informat or format that the CNTLIN= option is creating, then the VALUE, INVALUE, or PICTURE statement creates the informat or format and the CNTLIN= data set is not used. You can, however, create an informat or format with VALUE, INVALUE, or PICTURE and create a different informat or format with CNTLIN= in the same PROC FORMAT step.

For an example featuring an input control data set, see Example 5 on page 482.

Procedure Output

The FORMAT procedure prints output only when you specify the FMTLIB option or the PAGE option in the PROC FORMAT statement. The printed output is a table for each format or informat entry in the catalog that is specified in the LIBRARY= option. The output also contains global information and the specifics of each value or range that is defined for the format or informat. You can use the SELECT or EXCLUDE statement to control which formats and informats are represented in the FMTLIB output. For details, see “SELECT Statement” on page 458 and “EXCLUDE Statement” on page 446. For an example, see Example 6 on page 486.

The FMTLIB output shown in Output 21.2 on page 472 contains a description of the NOZEROS. format, which is created in “Building a Picture Format: Step by Step” on page 454, and the EVAL. informat, which is created in Example 4 on page 480.

Output 21.2 Output from PROC FORMAT with the FMTLIB Option

```
                  FMTLIB Output for the NOZEROS. Format and the              1
                                EVAL. Informat

   ----------------------------------------------------------------------------
   |       FORMAT NAME: NOZEROS  LENGTH:    5   NUMBER OF VALUES:    4        |
   |   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   5  FUZZ: STD        |
   |--------------------------------------------------------------------------|
   |START           |END             |LABEL  (VER. 7.00    29MAY98:10:00:24)  |
   |----------------+----------------+----------------------------------------|
   |LOW             |              -1|00.00               P-  F  M100         |
   |             -1<                0<99                  P-. F  M100         |
   |              0|                1<99                  P.  F  M100         |
   |              1|HIGH            |00.00                P   F  M100         |
   ----------------------------------------------------------------------------


   ----------------------------------------------------------------------------
   |     INFORMAT NAME: @EVAL    LENGTH:    1   NUMBER OF VALUES:    5        |
   |   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   1  FUZZ:        0   |
   |--------------------------------------------------------------------------|
   |START           |END             |INVALUE(VER. 7.00    29MAY98:10:00:25)  |
   |----------------+----------------+----------------------------------------|
   |C               |C               |                                       1|
   |E               |E               |                                       2|
   |N               |N               |                                       0|
   |O               |O               |                                       4|
   |S               |S               |                                       3|
   ----------------------------------------------------------------------------
```

The fields are described below in the order they appear in the output, from left to right:

INFORMAT NAME

FORMAT NAME
the name of the informat or format. Informat names begin with an at-sign (@).

LENGTH
the length of the informat or format. PROC FORMAT determines the length in the following ways:

- For character informats, the value for LENGTH is the length of the longest raw data value on the left side of the equals sign.
- For numeric informats
  - LENGTH is 12 if all values on the left side of the equals sign are numeric.
  - LENGTH is the same as the longest raw data value on the left side of the equal sign.
- For formats, the value for LENGTH is the length of the longest value on the right side of the equal sign.

In the output for @EVAL., the length is 1 because 1 is the length of the longest raw data value on the left side of the equals sign.

In the output for NOZEROS., the LENGTH is 5 because the longest picture is 5 characters.

NUMBER OF VALUES
the number of values or ranges associated with the informat or format. NOZEROS. has 4 ranges, EVAL. has 5.

MIN LENGTH
the minimum length of the informat or format. The value for MIN LENGTH is 1 unless you specify a different minimum length with the MIN= option.

MAX LENGTH
the maximum length of the informat or format. The value for MAX LENGTH is 40 unless you specify a different maximum length with the MAX= option.

DEFAULT LENGTH
the length of the longest value in the INVALUE or LABEL field, or the value of the DEFAULT= option.

FUZZ
the fuzz factor. For informats, FUZZ always is 0. For formats, the value for this field is STD if you do not use the FUZZ= option. STD signifies the default fuzz value.

START
the beginning value of a range. FMTLIB prints only the first 16 characters of a value in the START and END columns.

END
the ending value of a range. The exclusion sign (<) appears after the values in START and END, if the value is excluded from the range.

INVALUE
LABEL
INVALUE appears only for informats and contains the informatted values. LABEL appears only for formats and contains either the formatted value or picture. The SAS release number and the date on which the format or informat was created are in parentheses after INVALUE or LABEL.

For picture formats, such as NOZEROS., the LABEL section contains the PREFIX=, FILL=, and MULT= values. To note these values, FMTLIB prints the letters **P**, **F**, and **M** to represent each option, followed by the value. For example, in the LABEL section, **P-.** indicates that the prefix value is a dash followed by a period.

FMTLIB prints only 40 characters in the LABEL column.

Examples: FORMAT Procedure

Several examples in this section use the PROCLIB.STAFF data set. In addition, many of the informats and formats that are created in these examples are stored in LIBRARY.FORMATS. The output data set shown in "Output Control Data Set" on page 468 contains a description of these informats and the formats.

```
libname proclib 'SAS-data-library';
```

Create the data set PROCLIB.STAFF. The INPUT statement assigns the names Name, IdNumber, Salary, Site, and HireDate to the variables that appear after the DATALINES statement. The FORMAT statement assigns the standard SAS format DATE7. to the variable HireDate.

```
data proclib.staff;
   input Name & $16. IdNumber $ Salary
         Site $ HireDate date7.;
   format hiredate date7.;
   datalines;
Capalleti, Jimmy  2355 21163 BR1 30JAN79
Chen, Len         5889 20976 BR1 18JUN76
Davis, Brad       3878 19571 BR2 20MAR84
Leung, Brenda     4409 34321 BR2 18SEP74
Martinez, Maria   3985 49056 US2 10JAN93
Orfali, Philip    0740 50092 US2 16FEB83
Patel, Mary       2398 35182 BR3 02FEB90
Smith, Robert     5162 40100 BR5 15APR86
Sorrell, Joseph   4421 38760 US1 19JUN93
Zook, Carla       7385 22988 BR3 18DEC91
;
```

The variables are about a small subset of employees who work for a corporation that has sites in the U.S. and Britain. The data contain the name, identification number, salary (in British pounds), location, and date of hire for each employee.

Example 1: Creating a Picture Format

Procedure features:

PROC FORMAT statement options:

LIBRARY=

PICTURE statement options:

MULT=

PREFIX=
LIBRARY libref
LOW and HIGH keywords

Data set:
PROCLIB.STAFF on page 474.

This example uses a PICTURE statement to create a format that prints the values for the variable Salary in the data set PROCLIB.STAFF in U.S. dollars.

Program

Assign two SAS library references (PROCLIB and LIBRARY). Assigning a library reference LIBRARY is useful in this case because if you use PROC FORMAT, then SAS automatically searches for informats and formats in any library that is referenced with the LIBRARY libref.

```
libname proclib 'SAS-data-library-1 ';
libname library 'SAS-data-library-2';
```

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Specify that user-defined formats will be stored in the catalog LIBRARY.FORMATS. The LIBRARY= option specifies a SAS catalog that will contain the formats or informats that you create with PROC FORMAT. When you create the library named LIBRARY, SAS automatically creates a catalog named FORMATS inside LIBRARY.

```
   proc format library=library;
```

Define the USCurrency. picture format. The PICTURE statement creates a template for printing numbers. LOW-HIGH ensures that all values are included in the range. The MULT= statement option specifies that each value is multiplied by 1.61. The PREFIX= statement adds a US dollar sign to any number that you format. The picture contains six digit selectors, five for the salary and one for the dollar sign prefix.

```
      picture uscurrency low-high='000,000' (mult=1.61 prefix='$');
   run;
```

Print the PROCLIB.STAFF data set. The NOOBS option suppresses the printing of observation numbers. The LABEL option uses variable labels instead of variable names for column headings.

```
proc print data=proclib.staff noobs label;
```

Specify a label and format for the Salary variable. The LABEL statement substitutes the specific label for the variable in the report. In this case, "Salary in US Dollars" is substituted for the variable Salary for this print job only. The FORMAT statement associates the USCurrency. format with the variable name Salary for the duration of this procedure step.

```
    label salary='Salary in U.S. Dollars';
    format salary uscurrency.;
```

Specify the title.

```
    title 'PROCLIB.STAFF with a Format for the Variable Salary';
run;
```

Output

```
               PROCLIB.STAFF with a Format for the Variable Salary           1

                                          Salary in
                              Id          U.S.                      Hire
            Name              Number      Dollars     Site          Date

            Capalleti, Jimmy   2355       $34,072     BR1        30JAN79
            Chen, Len          5889       $33,771     BR1        18JUN76
            Davis, Brad        3878       $31,509     BR2        20MAR84
            Leung, Brenda      4409       $55,256     BR2        18SEP74
            Martinez, Maria    3985       $78,980     US2        10JAN93
            Orfali, Philip     0740       $80,648     US2        16FEB83
            Patel, Mary        2398       $56,643     BR3        02FEB90
            Smith, Robert      5162       $64,561     BR5        15APR86
            Sorrell, Joseph    4421       $62,403     US1        19JUN93
            Zook, Carla        7385       $37,010     BR3        18DEC91
```

Example 2: Creating a Format for Character Values

Procedure features:

VALUE statement

OTHER keyword

Data set:

PROCLIB.STAFF on page 474.

Format: USCurrency on page 475.

This example uses a VALUE statement to create a character format that prints a value of a character variable as a different character string.

Program

Assign two SAS library references (PROCLIB and LIBRARY). Assigning a library reference LIBRARY is useful in this case because if you use PROC FORMAT, then SAS automatically searches for informats and formats in any library that is referenced with the LIBRARY libref.

```
libname proclib 'SAS-data-library-1';
libname library 'SAS-data-library-2';
```

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Create the catalog named LIBRARY.FORMATS, where the user-defined formats will be stored. The LIBRARY= option specifies a permanent storage location for the formats that you create. It also creates a catalog named FORMAT in the specified library. If you do not use LIBRARY=, then SAS temporarily stores formats and informats that you create in a catalog named WORK.FORMATS.

```
   proc format library=library;
```

Define the $CITY. format. The special codes BR1, BR2, and so on, are converted to the names of the corresponding cities. The keyword OTHER specifies that values in the data set that do not match any of the listed city code values are converted to the value **INCORRECT CODE**.

```
      value  $city 'BR1'='Birmingham UK'
                   'BR2'='Plymouth UK'
                   'BR3'='York UK'
                   'US1'='Denver USA'
                   'US2'='Miami USA'
                   other='INCORRECT CODE';
   run;
```

Print the PROCLIB.STAFF data set. The NOOBS option suppresses the printing of observation numbers. The LABEL option uses variable labels instead of variable names for column headings.

```
proc print data=proclib.staff noobs label;
```

Specify a label for the Salary variable. The LABEL statement substitutes the label “Salary in U.S. Dollars” for the name SALARY.

```
   label salary='Salary in U.S. Dollars';
```

Specify formats for Salary and Site. The FORMAT statement temporarily associates the USCurrency. format (created in Example 1 on page 474) with the variable SALARY and also temporarily associates the format $CITY. with the variable SITE.

```
   format salary uscurrency. site $city.;
```

Specify the titles.

```
   title 'PROCLIB.STAFF with a Format for the Variables';
   title2 'Salary and Site';
run;
```

Output

```
                 PROCLIB.STAFF with a Format for the Variables                 1
                                Salary and Site

                                     Salary in
                            Id         U.S.                                Hire
       Name               Number      Dollars     Site                     Date

       Capalleti, Jimmy     2355      $34,072     Birmingham UK     30JAN79
       Chen, Len            5889      $33,771     Birmingham UK     18JUN76
       Davis, Brad          3878      $31,509     Plymouth UK       20MAR84
       Leung, Brenda        4409      $55,256     Plymouth UK       18SEP74
       Martinez, Maria      3985      $78,980     Miami USA         10JAN93
       Orfali, Philip       0740      $80,648     Miami USA         16FEB83
       Patel, Mary          2398      $56,643     York UK           02FEB90
       Smith, Robert        5162      $64,561     INCORRECT CODE    15APR86
       Sorrell, Joseph      4421      $62,403     Denver USA        19JUN93
       Zook, Carla          7385      $37,010     York UK           18DEC91
```

Example 3: Writing a Format for Dates Using a Standard SAS Format

Procedure features:

VALUE statement:

HIGH keyword

Data set:

PROCLIB.STAFF on page 474.

Formats:

USCurrency. on page 475 and $CITY. on page 477.

This example uses an existing format that is supplied by SAS as a formatted value. Tasks include

- creating a numeric format
- nesting formats
- writing a format using a standard SAS format
- formatting dates.

Program

This program defines a format called BENEFIT, which differentiates between employees hired on or before 31DEC1979. The purpose of this program is to indicate any employees who are eligible to receive a benefit, based on a hire date on or prior to December 31, 1979. All other employees with a later hire date are listed as ineligible for the benefit.

Assign two SAS library references (PROCLIB and LIBRARY). Assigning a library reference LIBRARY is useful in this case because if you use PROC FORMAT, then SAS automatically searches for informats and formats in any library that is referenced with the LIBRARY libref.

```
libname proclib 'SAS-data-library-1';
libname library 'SAS-data-library-2';
```

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Store the BENEFIT. format in the catalog LIBRARY.FORMATS. The LIBRARY= option specifies the permanent storage location LIBRARY for the formats that you create. If you do not use LIBRARY=, then SAS temporarily stores formats and informats that you create in a catalog named WORK.FORMATS.

```
   proc format library=library;
```

Define the first range in the BENEFIT. format. This first range differentiates between the employees who were hired on or before 31DEC1979 and those who were hired after that date. The keyword LOW and the SAS date constant '31DEC1979'D create the first range, which includes all date values that occur on or before December 31, 1979. For values that fall into this range, SAS applies the WORDDATE*w*. format.*

```
      value benefit  low-'31DEC1979'd=[worddate20.]
```

Define the second range in the BENEFIT. format. The second range consists of all dates on or after January 1, 1980. The SAS date constant '01JAN1980'D and the keyword HIGH specify the range. Values that fall into this range receive `** Not Eligible **` as a formatted value.

```
                    '01JAN1980'd-high='  ** Not Eligible **';
   run;
```

Print the data set PROCLIB.STAFF. The NOOBS option suppresses the printing of observation numbers. The LABEL option uses variable labels instead of variable names for column headings.

```
proc print data=proclib.staff noobs label;
```

* For more information about SAS date constants, see the section on dates, times, and intervals in *SAS Language Reference: Concepts*. For complete documentation on WORDDATE*w*., see the section on formats in *SAS Language Reference: Dictionary*.

Specify a label for the Salary variable. The LABEL statement substitutes the label "Salary in U.S. Dollars" for the name SALARY.

```
    label salary='Salary in U.S. Dollars';
```

Specify formats for Salary, Site, and Hiredate. The FORMAT statement associates the USCurrency. format (created in Example 1 on page 474) with SALARY, the $CITY. format (created in Example 2 on page 476) with SITE, and the BENEFIT. format with HIREDATE.

```
    format salary uscurrency. site $city. hiredate benefit.;
```

Specify the titles.

```
    title 'PROCLIB.STAFF with a Format for the Variables';
    title2 'Salary, Site, and HireDate';
 run;
```

Output

```
                  PROCLIB.STAFF with a Format for the Variables                1
                            Salary, Site, and HireDate

                                Salary in
                       Id         U.S.
 Name                Number     Dollars     Site                   HireDate

 Capalleti, Jimmy     2355      $34,072     Birmingham UK      January 30, 1979
 Chen, Len            5889      $33,771     Birmingham UK         June 18, 1976
 Davis, Brad          3878      $31,509     Plymouth UK       ** Not Eligible **
 Leung, Brenda        4409      $55,256     Plymouth UK       September 18, 1974
 Martinez, Maria      3985      $78,980     Miami USA         ** Not Eligible **
 Orfali, Philip       0740      $80,648     Miami USA         ** Not Eligible **
 Patel, Mary          2398      $56,643     York UK           ** Not Eligible **
 Smith, Robert        5162      $64,561     INCORRECT CODE    ** Not Eligible **
 Sorrell, Joseph      4421      $62,403     Denver USA        ** Not Eligible **
 Zook, Carla          7385      $37,010     York UK           ** Not Eligible **
```

Example 4: Converting Raw Character Data to Numeric Values

Procedure feature:
INVALUE statement

This example uses an INVALUE statement to create a numeric informat that converts numeric and character raw data to numeric data.

Program

This program converts quarterly employee evaluation grades, which are alphabetic, into numeric values so that reports can be generated that sum the grades up as points.

Set up two SAS library references, one named PROCLIB and the other named LIBRARY.

```
libname proclib 'SAS-data-library-1';
libname library 'SAS-data-library-2';
```

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=64 pagesize=40;
```

Store the Evaluation. informat in the catalog LIBRARY.FORMATS.

```
proc format library=library;
```

Create the numeric informat Evaluation. The INVALUE statement converts the specified values. The letters **O** (Outstanding), **S** (Superior), **E** (Excellent), **C** (Commendable), and **N** (None) correspond to the numbers 4, 3, 2, 1, and 0, respectively.

```
   invalue evaluation 'O'=4
                      'S'=3
                      'E'=2
                      'C'=1
                      'N'=0;
run;
```

Create the PROCLIB.POINTS data set. The instream data, which immediately follows the DATALINES statement, contains a unique identification number (EmployeeId) and bonus evaluations for each employee for each quarter of the year (Q1–Q4). Some of the bonus evaluation values that are listed in the data lines are numbers; others are character values. Where character values are listed in the data lines, the Evaluation. informat converts the value **O** to 4, the value **S** to 3, and so on. The raw data values 0 through 4 are read as themselves because they are not referenced in the definition of the informat. Converting the letter values to numbers makes it possible to calculate the total number of bonus points for each employee for the year. TotalPoints is the total number of bonus points.

```
data proclib.points;
   input EmployeeId $ (Q1-Q4) (evaluation.,+1);
   TotalPoints=sum(of q1-q4);
   datalines;
2355 S O O S
5889 2 2 2 2
3878 C E E E
```

```
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E C C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;
```

Print the PROCLIB.POINTS data set. The NOOBS option suppresses the printing of observation numbers.

```
proc print data=proclib.points noobs;
```

Specify the title.

```
   title 'The PROCLIB.POINTS Data Set';
run;
```

Output

```
                The PROCLIB.POINTS Data Set                 1

        Employee                               Total
           Id      Q1     Q2     Q3     Q4    Points

          2355      3      4      4      3      14
          5889      2      2      2      2       8
          3878      1      2      2      2       7
          4409      0      1      1      1       3
          3985      3      3      3      2      11
          0740      3      2      2      3      10
          2398      2      2      1      1       6
          5162      1      1      1      2       5
          4421      3      2      2      2       9
          7385      1      1      1      0       3
```

Example 5: Creating a Format from a Data Set

Procedure features:

PROC FORMAT statement option:

CNTLIN=

Input control data set

Data set:

WORK.POINTS, created from data lines in the sample code.

This example shows how to create a format from a SAS data set.

Tasks include

- creating a format from an input control data set
- creating an input control data set from an existing SAS data set.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create a temporary data set named scale. The first two variables in the data lines, called BEGIN and END, will be used to specify a range in the format. The third variable in the data lines, called AMOUNT, contains a percentage that will be used as the formatted value in the format. Note that all three variables are character variables as required for PROC FORMAT input control data sets.

```
data scale;
   input begin $ 1-2 end $ 5-8 amount $ 10-12;
   datalines;
0   3    0%
4   6    3%
7   8    6%
9   10   8%
11  16   10%
;
```

Create the input control data set CTRL and set the length of the LABEL variable. The LENGTH statement ensures that the LABEL variable is long enough to accommodate the label **`***ERROR***`**.

```
data ctrl;
   length label $ 11;
```

Rename variables and create an end-of-file flag. The data set CTRL is derived from WORK.SCALE. RENAME= renames BEGIN and AMOUNT as START and LABEL, respectively. The END= option creates the variable LAST, whose value is set to 1 when the last observation is processed.

```
   set scale(rename=(begin=start amount=label)) end=last;
```

Create the variables FMTNAME and TYPE with fixed values. The RETAIN statement is more efficient than an assignment statement in this case. RETAIN retains the value of FMTNAME and TYPE in the program data vector and eliminates the need for the value to be written on every iteration of the DATA step. FMTNAME specifies the name PercentageFormat, which is the format that the input control data set creates. The TYPE variable specifies that the input control data set will create a numeric format.

```
   retain fmtname 'PercentageFormat' type 'n';
```

Write the observation to the output data set.

```
   output;
```

Create an "other" category. Because the only valid values for this application are 0–16, any other value (such as missing) should be indicated as an error to the user. The IF statement executes only after the DATA step has processed the last observation from the input data set. When IF executes, HLO receives a value of **O** to indicate that the range is OTHER, and LABEL receives a value of `***ERROR***`. The OUTPUT statement writes these values as the last observation in the data set. HLO has missing values for all other observations.

```
   if last then do;
      hlo='O';
      label='***ERROR***';
      output;
   end;
run;
```

Print the control data set, CTRL. The NOOBS option suppresses the printing of observation numbers.

```
proc print data=ctrl noobs;
```

Specify the title.

```
   title 'The CTRL Data Set';
run;
```

Output 21.3

Note that although the last observation contains values for START and END, these values are ignored because of the **O** value in the HLO variable.

```
                          The CTRL Data Set                              1

label           start    end        fmtname          type    hlo

0%                0       3      PercentageFormat     n
3%                4       6      PercentageFormat     n
6%                7       8      PercentageFormat     n
8%                9       10     PercentageFormat     n
10%               11      16     PercentageFormat     n
***ERROR***       11      16     PercentageFormat     n       O
```

Store the created format in the catalog WORK.FORMATS and specify the source for the format. The CNTLIN= option specifies that the data set CTRL is the source for the format PTSFRMT.

```
proc format library=work cntlin=ctrl;
run;
```

Create the numeric informat Evaluation. The INVALUE statement converts the specified values. The letters **O** (Outstanding), **S** (Superior), **E** (Excellent), **C** (Commendable), and **N** (None) correspond to the numbers 4, 3, 2, 1, and 0, respectively.

```
proc format;
   invalue evaluation 'O'=4
                      'S'=3
                      'E'=2
                      'C'=1
                      'N'=0;
run;
```

Create the WORK.POINTS data set. The instream data, which immediately follows the DATALINES statement, contains a unique identification number (EmployeeId) and bonus evaluations for each employee for each quarter of the year (Q1–Q4). Some of the bonus evaluation values that are listed in the data lines are numbers; others are character values. Where character values are listed in the data lines, the Evaluation. informat converts the value **O** to 4, the value **S** to 3, and so on. The raw data values 0 through 4 are read as themselves because they are not referenced in the definition of the informat. Converting the letter values to numbers makes it possible to calculate the total number of bonus points for each employee for the year. TotalPoints is the total number of bonus points. The addition operator is used instead of the SUM function so that any missing value will result in a missing value for TotalPoints.

```
data points;
   input EmployeeId $ (Q1-Q4) (evaluation.,+1);
   TotalPoints=q1+q2+q3+q4;
```

```
   datalines;
2355 S O O S
5889 2 . 2 2
3878 C E E E
4409 0 1 1 1
3985 3 3 3 2
0740 S E E S
2398 E E   C
5162 C C C E
4421 3 2 2 2
7385 C C C N
;
```

Generate a report for WORK.POINTS and associate the PTSFRMT. format with the TotalPoints variable. The DEFINE statement performs the association. The column that contains the formatted values of TotalPoints is using the alias Pctage. Using an alias enables you to print a variable twice, once with a format and once with the default format. See Chapter 38, “The REPORT Procedure,” on page 937 for more information about PROC REPORT.

```
proc report data=work.points nowd headskip split='#';
   column employeeid totalpoints totalpoints=Pctage;
   define employeeid / right;
   define totalpoints / 'Total#Points' right;
   define pctage / format=PercentageFormat12. 'Percentage' left;
   title 'The Percentage of Salary for Calculating Bonus';
run;
```

Output

Output 21.4

```
        The Percentage of Salary for Calculating Bonus               1

           Employee      Total
                 Id     Points  Percentage

               2355         14  10%
               5889          .  ***ERROR***
               3878          7  6%
               4409          3  0%
               3985         11  10%
               0740         10  8%
               2398          .  ***ERROR***
               5162          5  3%
               4421          9  8%
               7385          3  0%
```

Example 6: Printing the Description of Informats and Formats

Procedure features:

PROC FORMAT statement option:
  FMTLIB
SELECT statement

Format:
NOZEROS on page 455.

Informat:
Evaluation. on page 481.

This example illustrates how to print a description of an informat and a format. The description shows the values that are input and output.

Program

Set up a SAS library reference named LIBRARY.

```
libname library 'SAS-data-library';
```

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Print a description of Evaluation. and NOZEROS. The FMTLIB option prints information about the formats and informats in the catalog that the LIBRARY= option specifies. LIBRARY=LIBRARY points to the LIBRARY.FORMATS catalog.

```
proc format library=library fmtlib;
```

Select an informat and a format. The SELECT statement selects EVAL and NOZEROS, which were created in previous examples. The at sign (@) in front of EVAL indicates that EVAL. is an informat.

```
   select @evaluation nozeros;
```

Specify the titles.

```
   title 'FMTLIB Output for the NOZEROS. Format and the';
   title2 'Evaluation. Informat';
run;
```

Output

The output is described in "Procedure Output" on page 472.

```
                 FMTLIB Output for the NOZEROS. Format and the                1
                              Evaluation. Informat

 ----------------------------------------------------------------------------
 |       FORMAT NAME: NOZEROS  LENGTH:    5   NUMBER OF VALUES:    4        |
 |   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   5  FUZZ: STD        |
 |--------------------------------------------------------------------------|
 |START           |END             |LABEL  (VER. V7|V8   10APR2002:18:55:08)|
 |----------------+----------------+----------------------------------------|
 |LOW             |              -1|00.00             P-  F  M100         |
 |              -1<               0<99                P-. F  M100         |
 |               0|               1<99                P.  F  M100         |
 |               1|HIGH            |00.00             P   F  M100         |
 ----------------------------------------------------------------------------


 ----------------------------------------------------------------------------
 |                   INFORMAT NAME: @EVALUATION LENGTH: 1                   |
 |   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   1  FUZZ:        0   |
 |--------------------------------------------------------------------------|
 |START           |END             |INVALUE(VER. 9.00    10APR2002:18:55:11)|
 |----------------+----------------+----------------------------------------|
 |C               |C               |                                       1|
 |E               |E               |                                       2|
 |N               |N               |                                       0|
 |O               |O               |                                       4|
 |S               |S               |                                       3|
 ----------------------------------------------------------------------------
```

Example 7: Retrieving a Permanent Format

Procedure features:

PROC FORMAT statement options:

LIBRARY=

Other features:

FMTSEARCH= system option

Data sets:

SAMPLE on page 454.

This example uses the LIBRARY= option and the FMTSEARCH= system option to store and retrieve a format stored in a catalog other than WORK.FORMATS or LIBRARY.FORMATS.

Program

Set up a SAS library reference named PROCLIB.

```
libname proclib 'SAS-data-library';
```

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=64 pagesize=60;
```

Store the NOZEROS. format in the PROCLIB.FORMATS catalog.

```
proc format library=proclib;
```

Create the NOZEROS. format. The PICTURE statement defines the picture format NOZEROS. See "Building a Picture Format: Step by Step" on page 454.

```
   picture nozeros
              low   -     -1  =  '00.00' (prefix='-'          )
               -1  <-<     0  =     '99' (prefix='-.' mult=100)
                0   -<     1  =     '99' (prefix='.'  mult=100)
                1   -   high  =  '00.00';
run;
```

Add the PROCLIB.FORMATS catalog to the search path that SAS uses to find user-defined formats. The FMTSEARCH= system option defines the search path. The FMTSEARCH= system option requires only a libref. FMTSEARCH= assumes that the catalog name is FORMATS if no catalog name appears. Without the FMTSEARCH= option, SAS would not find the NOZEROS. format.*

```
options  fmtsearch=(proclib);
```

Print the SAMPLE data set. The FORMAT statement associates the NOZEROS. format with the Amount variable.

```
proc print data=sample;
   format amount nozeros.;
```

Specify the titles.

```
   title1 'Retrieving the NOZEROS. Format from PROCLIB.FORMATS';
   title2 'The SAMPLE Data Set';
run;
```

* For complete documentation on the FMTSEARCH= system option, see the section on SAS system options in *SAS Language Reference: Dictionary*.

Output

```
      Retrieving the NOZEROS. Format from PROCLIB.FORMATS      1
                     The SAMPLE Data Set

                         Obs    Amount

                          1     -2.05
                          2      -.05
                          3      -.01
                          4       .00
                          5       .09
                          6       .54
                          7       .55
                          8      6.60
                          9     14.63
```

Example 8: Writing Ranges for Character Strings

Data sets:
PROCLIB.STAFF on page 474.

This example creates a format and shows how to use ranges with character strings.

Program

```
libname proclib 'SAS-data-library';
```

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Create the TRAIN data set from the PROCLIB.STAFF data set. PROCLIB.STAFF was created in "Examples: FORMAT Procedure" on page 474.

```
data train;
   set proclib.staff(keep=name idnumber);
run;
```

Print the data set TRAIN without a format. The NOOBS option suppresses the printing of observation numbers.

```
proc print data=train noobs;
```

Specify the title.

```
   title 'The TRAIN Data Set without a Format';
run;
```

```
                  The TRAIN Data Set without a Format                 1

                                           Id
                     Name                Number

                     Capalleti, Jimmy     2355
                     Chen, Len            5889
                     Davis, Brad          3878
                     Leung, Brenda        4409
                     Martinez, Maria      3985
                     Orfali, Philip       0740
                     Patel, Mary          2398
                     Smith, Robert        5162
                     Sorrell, Joseph      4421
                     Zook, Carla          7385
```

Store the format in WORK.FORMATS. Because the LIBRARY= option does not appear, the format is stored in WORK.FORMATS and is available only for the current SAS session.

```
proc format;
```

Create the $SkillTest. format. The $SKILL. format prints each employee's identification number and the skills test that they have been assigned. Employees must take either TEST A, TEST B, or TEST C, depending on their last name. The exclusion operator (<) excludes the last value in the range. Thus, the first range includes employees whose last name begins with any letter from A through D, and the second range includes employees whose last name begins with any letter from E through M. The tilde (~) in the last range is necessary to include an entire string that begins with the letter Z.

```
   value $skilltest  'a'-<'e','A'-<'E'='Test A'
              'e'-<'m','E'-<'M'='Test B'
              'm'-'z~','M'-'Z~'='Test C';
run;
```

Generate a report of the TRAIN data set. The FORMAT= option in the DEFINE statement associates $SkillTest. with the NAME variable. The column that contains the formatted values of NAME is using the alias Test. Using an alias enables you to print a variable twice, once with a format and once with the default format. See Chapter 38, "The REPORT Procedure," on page 937for more information about PROC REPORT.

```
proc report data=train nowd headskip;
   column name name=test idnumber;
   define test / display format=$skilltest. 'Test';
   define idnumber / center;
   title 'Test Assignment for Each Employee';
```

```
run;
```

Output

```
                    Test Assignment for Each Employee                 1

                    Name              Test    IdNumber

                    Capalleti, Jimmy  Test A    2355
                    Chen, Len         Test A    5889
                    Davis, Brad       Test A    3878
                    Leung, Brenda     Test B    4409
                    Martinez, Maria   Test C    3985
                    Orfali, Philip    Test C    0740
                    Patel, Mary       Test C    2398
                    Smith, Robert     Test C    5162
                    Sorrell, Joseph   Test C    4421
                    Zook, Carla       Test C    7385
```

Example 9: Filling a Picture Format

Procedure features:

PICTURE statement options:

FILL=
PREFIX=

This example

- prefixes the formatted value with a specified character
- fills the leading blanks with a specified character
- shows the interaction between the FILL= and PREFIX= options.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=64 pagesize=40;
```

Create the PAY data set. The PAY data set contains the monthly salary for each employee.

```
data pay;
   input Name $ MonthlySalary;
```

```
   datalines;
Liu   1259.45
Lars  1289.33
Kim   1439.02
Wendy 1675.21
Alex  1623.73
;
```

Define the SALARY. picture format and specify how the picture will be filled. When FILL= and PREFIX= PICTURE statement options appear in the same picture, the format places the prefix and then the fill characters. The SALARY. format fills the picture with the fill character because the picture has zeros as digit selectors. The leftmost comma in the picture is replaced by the fill character.

```
proc format;
   picture salary low-high='00,000,000.00' (fill='*' prefix='$');
run;
```

Print the PAY data set. The NOOBS option suppresses the printing of observation numbers. The FORMAT statement temporarily associates the SALARY. format with the variable MonthlySalary.

```
proc print data=pay noobs;
   format monthlysalary salary.;
```

Specify the title.

```
   title 'Printing Salaries for a Check';
run;
```

Output

```
                  Printing Salaries for a Check                1

                      Name     MonthlySalary

                      Liu      ****$1,259.45
                      Lars     ****$1,289.33
                      Kim      ****$1,439.02
                      Wendy    ****$1,675.21
                      Alex     ****$1,623.73
```

See Also

FMTSEARCH= System option
VALIDFMTNAME= System option
FORMAT Statement

CHAPTER

22

The FORMS Procedure

Overview: FORMS Procedure

The FORMS procedure produces labels for envelopes, mailing labels, external tape labels, file cards, and any other printer forms that have a regular pattern.

For each observation in the input SAS data set, PROC FORMS prints data in a rectangular block called a *form unit*. For example, a mailing label is a form unit.

Output 22.1 on page 495 illustrates a simple mailing list produced with PROC FORMS. The statements that produce the output follow. The OBS= data set option limits to six the number of observations that PROC FORMS processes.

```
options pagesize=60 linesize=64 nodate
        pageno=1;


filename labels 'external-file';


proc forms data=list(obs=6) file=labels
           align=0;
   line 1 name;
   line 2 street;
   line 3 city state zip;
run;
```

Output 22.1 Simple Mailing List Produced with PROC FORMS

```
Gabrielli, Theresa
24 Ridgetop Rd.
Westboro        MA 01581

Clayton, Aria
314 Bridge St.
Hanover         NH 03755

Dix, Martin L.
4 Shepherd St.
Norwich         VT 05055

Slater, Emily C.
2009 Cherry St.
York            PA 17407

Ericson, Jane
211 Clancey Court
Chapel Hill     NC 27514

An, Ing
95 Willow Dr.
Charlotte       NC 28211
```

Output 22.2 on page 496 is a customized version of the same mailing list. The statements that create this list

- invert the name so the first name appears first
- eliminate extra spaces between the city and state
- place three form units in each row
- make three copies of each form
- use only observations from the states in New England.

For an explanation of the program that produces these labels, see Example 3 on page 510.

Output 22.2 Customized Mailing List Produced with PROC FORMS

```


Theresa Gabrielli        Theresa Gabrielli        Theresa Gabrielli
24 Ridgetop Rd.          24 Ridgetop Rd.          24 Ridgetop Rd.
Westboro MA 01581        Westboro MA 01581        Westboro MA 01581


Aria Clayton             Aria Clayton             Aria Clayton
314 Bridge St.           314 Bridge St.           314 Bridge St.
Hanover NH 03755         Hanover NH 03755         Hanover NH 03755


Martin L. Dix            Martin L. Dix            Martin L. Dix
4 Shepherd St.           4 Shepherd St.           4 Shepherd St.
Norwich VT 05055         Norwich VT 05055         Norwich VT 05055
```

Syntax: FORMS Procedure

Requirements: At least one LINE statement

Reminder: You can use the ATTRIB, FORMAT, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

PROC FORMS <*option(s)*>;

BY <DESCENDING> *variable-1*
<…<DESCENDING> *variable-n*>
<NOTSORTED>;

FREQ *variable*;

LINE *line-number variable(s)* </ *option(s)*>;

To do this	Use this statement
Produce a separate set of forms for each BY group	BY
Treat observations as if they appear multiple times in the input data set	FREQ
Specify the information to print on a line of the form unit	LINE

PROC FORMS Statement

PROC FORMS <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Identify an external file for PROC FORMS to write to	FILE=
Control the dimensions of a form	
Specify the number of lines in a form unit	LINES=
Specify the number of columns across the form unit	WIDTH=
Control the placement of the forms	
Specify the number of form units to print across the page	ACROSS=
Specify the number of spaces to print between form units	BETWEEN=

To do this	Use this option
Specify the number of lines to skip on a page before printing the first form unit	DOWN=
Specify the number of spaces to indent before printing the first form unit in each row	INDENT=
Specify the number of form units to print down the page	NDOWN=
Specify the number of lines on a page of forms	PAGESIZE=
Specify the number of lines to skip between form units	SKIP=
Control the number of each form unit that PROC FORMS prints	
Specify the number of form units to produce for each observation in each set of form units	COPIES=
Specify the number of sets of form units to produce	SETS=
Control the placement of page eject characters	CC
Specify the number of lines of dummy form units to print	ALIGN=

Options

ACROSS=*form-units-per-line*
specifies the number of form units to print across the page. (See Figure 22.1 on page 504.)

Alias: A=

Default: 1

Range: 1-200

Featured in: Example 1 on page 505

ALIGN=*number*
controls the number of alignment form units that print before the actual data values. Use the alignment form units, which consist solely of Xs, to check printer alignment.

Default: 8 with FILE=; 0 without FILE=

Interaction: If you use ACROSS=, the number of dummy form units that print is the product of the values of ACROSS= and ALIGN=.

Featured in: Example 1 on page 505

BETWEEN=*spaces-between-form-units*
specifies the number of spaces to print between form units. (See Figure 22.1 on page 504.)

Alias: B=

Default: 1

Range: 1-200

Featured in: Example 1 on page 505

CC
in continuous mode, writes a page-eject character at the top of the first page. In page mode, if you also specify FILE=, CC writes a page-eject character at the top of each page. (CC has no effect if you omit FILE=.) For a discussion of page mode and continuous mode, see “Modes of Operation” on page 504.

Tip: If you omit CC, PROC FORMS issues blank lines to go to the next page. We recommend that you always use CC with page-mode operation.

Featured in: Example 2 on page 508

COPIES=*number*
specifies the number of form units to produce for each observation in each set of form units. All copies of an observation appear together.

Alias: C=

Default: 1

Featured in: Example 3 on page 510

DATA=*SAS-data-set*
identifies the input SAS data set.

DOWN=*top-margin*
specifies the number of lines to skip on a page before printing the first form unit. (See Figure 22.1 on page 504.)

Alias: D=

Default: 1

Range: 1-200

Featured in: Example 1 on page 505

Note: When PROC FORMS writes to the procedure output file, it uses one line for each TITLE statement and leaves a blank line beneath the last title. Counting for the top margin begins at the next line. Thus, if you have two TITLE statements and specify DOWN=5, PROC FORMS begins printing the first form unit on each page on line 9. △

FILE=*fileref*
identifies an external file for PROC FORMS to write to. Use the FILENAME statement to associate an external file with a fileref (see *SAS Language Reference: Concepts*).

Alias: DDNAME=, D=

Default: If you omit FILE=, PROC FORMS writes to the procedure output file and selects page mode.

Interaction: If you use FILE= and do not specify the ALIGN= option, PROC FORMS uses ALIGN=8.

Interaction: When you use FILE=, PROC FORMS honors DOWN= only on the first page of form units.

Interaction: If you use FILE= with NDOWN= or PAGESIZE= or both, you select page mode. Otherwise, you select continuous mode.

Featured in: Example 1 on page 505

INDENT=*left-margin*
specifies the number of spaces to indent before printing the first form unit in each row. (See Figure 22.1 on page 504.)

Alias: I=

Default: 0

Range: 0-200

LINES=*form-unit-length*
specifies the number of lines in a form unit. (See Figure 22.1 on page 504.)

Alias: L=

Default: the largest number used with the LINE statement

Range: 1-200

NDOWN=*form-units-per-page*
specifies the number of form units to print down the page and selects page-mode operation. (See Figure 22.1 on page 504.)

Alias: ND=

Default: FLOOR((PAGESIZE–DOWN+SKIP)/(LINES+SKIP)) where FLOOR is a SAS function that returns the largest integer less than or equal to the value of the argument.

Interaction: If NDOWN= specifies a number of form units that is less than PAGESIZE= allows, PROC FORMS honors NDOWN=. If NDOWN= specifies a number of form units that is greater than PAGESIZE= allows, PROC FORMS adjusts the value of NDOWN= downwards to accommodate the page size.

Featured in: Example 2 on page 508

PAGESIZE=*lines-per-page*
specifies the number of lines on a page of forms after allowing for TITLE statements and a blank line following the titles. (See Figure 22.1 on page 504.) It also selects page-mode operation.

Alias: PS=

Default: the system page size (with FILE=); inferred from the characteristics of the procedure output file and from title information (without FILE=)

Range: the value of DOWN= plus the value of LINES=, up to 10,000

Interaction: When you write to the procedure output, if the page size that you specify is greater than the page size specified by the SAS system option PAGESIZE=, PROC FORMS adjusts the PROC FORMS page size to accommodate the system page size.

Interaction: If the page size allows for more form units than NDOWN= specifies, PROC FORMS honors the NDOWN= option. If the page size does not allow for as many form units as NDOWN= specifies, PROC FORMS adjusts the value of NDOWN= to accommodate the page size.

SETS=*number*
specifies the number of sets of form units to produce. In page-mode operation, PROC FORMS starts each set on a new page.

Default: 1

Featured in: Example 2 on page 508

SKIP=*lines-between-form-units*
specifies the number of lines to skip between form units. (See Figure 22.1 on page 504.)

Alias: S=

Default: 1

Range: 1-200

Featured in: Example 1 on page 505

WIDTH=*form-unit-width*
specifies the number of columns across the form unit. PROC FORMS truncates values that do not fit in the specified width. (See Figure 22.1 on page 504.)

Alias: W=

Default: width of the widest line

Range: 1-255

Featured in: Example 1 on page 505

BY Statement

Produces a separate set of forms for each BY group.

Main discussion: "BY" on page 54

BY <DESCENDING> *variable-1*
<…<DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the data set is sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The data are grouped in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

How PROC FORMS Separates BY Groups

In page mode, the forms for each BY group begin on a new page. In continuous mode, BY groups are not separated.

FREQ Statement

Treats observations as if they appear multiple times in the input data set.

FREQ *variable*;

Required Arguments

variable

specifies a numeric variable whose value represents the frequency of each observation. If you use the FREQ statement, the procedure assumes that each observation in the input data set represents *n* observations, where *n* is the value of *variable*. If *n* is not an integer, the SAS System truncates it. If *n* is less than 1 (which includes missing), the procedure does not use that observation.

The sum of the frequency variable represents the total number of observations.

LINE Statement

Specifies the information to print on one line of the form unit. Use one LINE statement for each line of the form unit.

LINE *line-number variable(s)* </ *option(s)*>;

To do this	Use this option
Specify the number of spaces to indent the line within the form unit	INDENT=
Rotate the words in a character variable that contains a comma around the comma and remove the comma	LASTNAME
Remove extra blanks from the line so that one blank separates variables	PACK
Remove periods that represent missing values from a line that contains no other values.	REMOVE

Required Arguments

line-number

identifies the number of the line. You can specify lines in any order. You do not need a LINE statement for a blank line.

Range: An integer between 1 and the value of LINES= in the PROC FORMS statement

variable(s)

specifies one or more variables to print on this line of the form unit. The FORMS procedure inserts one space between each value. By default, the width of a variable's field in the form unit is the formatted length of that variable. Default formats are the length of the variable for character variables and BEST12. for numeric variables.

Interaction: If the length of all values in a line is longer than the value of WIDTH= specified in the PROC FORMS statement, PROC FORMS truncates the values (starting with the rightmost value in the line) to fit the WIDTH= value. For information on squeezing variables onto a line, see PACK on page 503.

Options

INDENT=*margin-within-form-unit*
specifies the number of spaces to indent the line within the form unit. Contrast this option to INDENT= in the PROC FORMS statement, which specifies the size of the left margin preceding the first form unit in each row.

Alias: I=

Featured in: Example 1 on page 505

LASTNAME
rotates the words in a character variable that contains a comma around the comma and removes the comma.

Alias: L

Featured in: Example 1 on page 505

PACK
removes extra blanks from the line so that one blank separates variables.

Alias: P

Tip: PACK can squeeze fields onto a form unit, but if the values for all the variables are long, you may lose an entire field. To avoid this problem, use a FORMAT statement to limit the space for each variable. For example, the following statement sets the field widths of the variables CITY and STATE to 20 and 2 columns, respectively:

```
format city $20. state $2.;
```

Featured in: Example 1 on page 505

REMOVE
removes periods that represent missing values from a line that contains no other values.

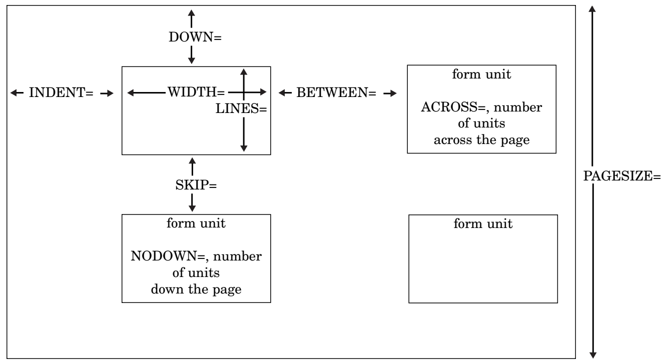
Alias: R

Concepts: FORMS Procedure

Form Layout

The size and spacing of form units are controlled by options in the PROC FORMS statement, as illustrated in Figure 22.1 on page 504. (See also the discussion of these options on page 498.)

Figure 22.1 Sample Placement for Forms



The values of the variables specified in LINE statements are formatted into a form unit that is WIDTH= columns wide and LINES= lines long. Values that do not fit into WIDTH= columns are truncated. ACROSS= form units are printed across the page, with BETWEEN= spaces between adjacent form units. The forms are indented INDENT= spaces from the left margin. SKIP= blank lines are printed between form units down the page.

Modes of Operation

PROC FORMS operates in two modes: continuous mode and page mode. Continuous mode is for forms that feed continuously through a printer, without the printer's needing to perform page ejects. Page mode is for forms that use separate pieces of paper for each form unit or for multiple form units (such as sheets of labels that come with 30 labels per sheet of paper).

By default, PROC FORMS uses page mode. To select continuous mode, you must specify FILE= and must not specify NDOWN= or PAGESIZE=.

In Continuous Mode, PROC FORMS Always Writes to an External File

When it writes in continuous mode, PROC FORMS

1 skips the number of lines specified by DOWN=

2 prints one form unit

3 skips the number of lines specified by SKIP=

4 repeats steps 2 and 3 until it uses all the data.

By default, in continuous mode the first eight form units are dummy form units that consist solely of Xs. These forms give the printer operator a chance to align the printer before real form units begin to print. Use ALIGN= to alter the number of dummy form units. Once the dummy form units are aligned to the physical forms, the file prints correctly. Carriage control characters are unnecessary.

In Page Mode, PROC FORMS Can Write Either to an External File or to the Procedure Output File

In page mode, PROC FORMS

1 goes to the top of a new page

2 skips the number of lines specified by DOWN=

3 prints the number of form units specified by NDOWN= down the page, or if you omit NDOWN=, prints the maximum number of form units allowed by the page size

4 repeats steps 1 to 3 until it uses all the data.

When PROC FORMS has written as many form units as you specified, either it writes a blank line for each line remaining on the page (as determined by the PAGESIZE= option) or it writes a page-eject character. If you are writing to the procedure output file, PROC FORMS always writes the page-eject characters. If you have specified FILE=, PROC FORMS by default writes blank lines, but if you specify the CC option, it writes page eject characters instead.

In page mode, the easiest way to ensure proper alignment is to specify the number of form units to print down the page with the NDOWN= option and to use CC to write a page-eject character at the beginning of each page. If you omit CC, be sure that the page size is set correctly. If it isn't, the number of blank lines that PROC FORMS writes will not take you to the top of the next page.

Note: We recommend that you always use CC when you use page mode with the FILE= option. △

CAUTION:

The procedure output file contains some things that you may not want on your forms. If you omit the FILE= option, the output from PROC FORMS goes to the procedure output file. If the DATE and NUMBER options are in effect, the output will contain dates and page numbers. If any titles or footnotes are defined, they will appear in the output as well. △

Examples: FORMS Procedure

The examples in this chapter assume alignment for the forms that they use. You must experiment to determine how to align your form units with your forms.

Example 1: Printing a Single Form Unit for Each Observation

Procedure features:

PROC FORMS statement options:

ACROSS=
ALIGN=
BETWEEN=
DOWN=
FILE=
SKIP=
WIDTH=

LINE statement options:

INDENT=
LASTNAME
PACK

Other features:

SORT procedure

This example uses PROC FORMS to print one set of mailing labels consisting of one copy of the form unit for each observation.

Program

Create the LIST data set and sort by zip code. The data set LIST contains names and mailing addresses. PROC SORT sorts the data by zip code.

```
options nodate pageno=1 linesize=80 pagesize=60;
data list;
   input Name $ 1-19 Street $ 20-39 City $ 40-54
         State $ 55-56 Zip $ 59-63;
   datalines;
Ericson, Jane      211 Clancey Court   Chapel Hill    NC  27514
Dix, Martin L.     4 Shepherd St.      Norwich        VT  05055
Gabrielli, Theresa 24 Ridgetop Rd.     Westboro       MA  01581
Clayton, Aria      314 Bridge St.      Hanover        NH  03755
Archuleta, Ruby    Box 108             Milagro        NM  87429
Misiewicz, Jeremy  43-C Lakeview Apts. Madison        WI  53704
Ahmadi, Hafez      5203 Marston Way    Boulder        CO  80302
Jacobson, Becky    7 Lincoln St.       Tallahassee    FL  32312
An, Ing            95 Willow Dr.       Charlotte      NC  28211
Slater, Emily C.   2009 Cherry St.     York           PA  17407
;

proc sort data=list;
   by zip;
run;
```

Specify a name for the external file. The FILENAME statement associates the name LABELS with the external file that will receive the output from PROC FORMS.

```
filename labels 'external-file';
```

Send a single form unit for each observation to the LABELS external file. FILE= sends the output to the file associated with the fileref LABELS. Because neither NDOWN= nor PAGESIZE= is specified, PROC FORMS uses continuous mode. WIDTH= sets the width of the form units to 24 to provide enough room for all the variables on each line. ACROSS= writes three form units across each page. BETWEEN= puts four blank characters between adjacent form units. DOWN= skips two lines at the top of the file so that the form units and the forms align correctly. SKIP= skips two lines between form units to maintain the proper alignment. ALIGN= prints two lines of dummy form units.

```
proc forms data=list file=labels
     width=24
     across=3
     between=4
```

```
      down=2
      skip=2
      align=2;
```

Specify the variables to place on each line. The LINE statements specify the variables to place on each line. LASTNAME removes the comma from Name and writes the first name before the last name. PACK removes extra blank characters between City and State. INDENT= indents Zip 15 spaces.

```
   line 1 name / lastname;
   line 2 street;
   line 3 city state / pack;
   line 4 zip / indent=15;
run;
```

Output

```
xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx


xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxxxxxxxxxxxxxxxxxx


Theresa Gabrielli           Aria Clayton                Martin L. Dix
24 Ridgetop Rd.             314 Bridge St.              4 Shepherd St.
Westboro MA                 Hanover NH                  Norwich VT
               01581                       03755                       05055


Emily C. Slater             Jane Ericson                Ing An
2009 Cherry St.             211 Clancey Court           95 Willow Dr.
York PA                     Chapel Hill NC              Charlotte NC
               17407                       27514                       28211


Becky Jacobson              Jeremy Misiewicz            Hafez Ahmadi
7 Lincoln St.               43-C Lakeview Apts.         5203 Marston Way
Tallahassee FL              Madison WI                  Boulder CO
               32312                       53704                       80302


Ruby Archuleta
Box 108
Milagro NM
               87429
```

Example 2: Printing Two Sets of Mailing Labels

Procedure features:

PROC FORMS statement options:

- ALIGN=
- CC
- FILE=
- NDOWN=
- SETS=

Data set:

LIST on page 506

This example uses page mode and SETS= to produce two sets of mailing labels. Each sheet of labels holds four rows of two labels.

Program

Specify a name for the external file. The FILENAME statement associates the name LABELS with the external file that will receive the output from PROC FORMS.

```
filename labels 'external-file';
```

```
options nodate pageno=1 linesize=80 pagesize=60 ;
```

Send two sets of form units to the LABELS external file. FILE= sends the output to the file associated with the fileref LABELS. NDOWN= prints four rows of form units on each page. CC writes carriage control characters to the file specified by FILE=. WIDTH= sets the width of the form units to 24 to provide enough room for the variables on each line. ACROSS= writes two form units across each page. BETWEEN= puts 20 blank characters between adjacent form units. DOWN= skips two lines at the top of each page so that the form units and the forms align correctly. SKIP= skips three lines between form units to maintain the proper alignment. ALIGN= suppresses the printing of dummy form units. SETS= writes two sets of form units. Each set begins on a new page.

```
proc forms data=list file=labels
           ndown=4
           cc
           width=24
           across=2
           between=20
           down=2
           skip=3
           align=0
           sets=2;
```

Specify the variables to place on each line. The LINE statements specify the variables to place on each line. PACK removes extra blank characters between City and State.

```
   line 1 name;
   line 2 street;
   line 3 city state zip / pack;
run;
```

Output

```
Gabrielli, Theresa                          Clayton, Aria
24 Ridgetop Rd.                             314 Bridge St.
Westboro MA 01581                           Hanover NH 03755



Dix, Martin L.                              Slater, Emily C.
4 Shepherd St.                              2009 Cherry St.
Norwich VT 05055                            York PA 17407



Ericson, Jane                               An, Ing
211 Clancey Court                           95 Willow Dr.
Chapel Hill NC 27514                        Charlotte NC 28211



Jacobson, Becky                             Misiewicz, Jeremy
7 Lincoln St.                               43-C Lakeview Apts.
Tallahassee FL 32312                        Madison WI 53704
```

```
Ahmadi, Hafez                               Archuleta, Ruby
5203 Marston Way                            Box 108
Boulder CO 80302                            Milagro NM 87429
```

```
Gabrielli, Theresa                          Clayton, Aria
24 Ridgetop Rd.                             314 Bridge St.
Westboro MA 01581                           Hanover NH 03755



Dix, Martin L.                              Slater, Emily C.
4 Shepherd St.                              2009 Cherry St.
Norwich VT 05055                            York PA 17407



Ericson, Jane                               An, Ing
211 Clancey Court                           95 Willow Dr.
Chapel Hill NC 27514                        Charlotte NC 28211



Jacobson, Becky                             Misiewicz, Jeremy
7 Lincoln St.                               43-C Lakeview Apts.
Tallahassee FL 32312                        Madison WI 53704
```

```
Ahmadi, Hafez                          Archuleta, Ruby
5203 Marston Way                       Box 108
Boulder CO 80302                       Milagro NM 87429
```

Example 3: Writing Multiple Copies of a Label within a Single Set of Labels

Procedure features:
- PROC FORMS statement options:
  - COPIES=
- LINE statement options:
  - LASTNAME
  - PACK

Data set: LIST on page 506

This example writes one set of mailing labels that consists of three copies of each form unit. It selects only those observations with addresses in one of the New England states.

Program

Specify a name for the external file. The FILENAME statement associates the name LABELS with the external file that will receive the output from PROC FORMS.

```
filename labels 'external-file';
```

```
options pagesize=60 pageno=1 nodate linesize=80;
```

Send three copies of each form unit to the LABELS external file. FILE= sends the output to the file associated with the fileref LABELS. NDOWN= prints five rows of form units on each page. CC writes carriage control characters to the file specified by FILE=. ALIGN= suppresses the printing of dummy form units. WIDTH= sets the width of the form units to 24 to provide enough room for the variables on each line. ACROSS= writes three form units across each page. DOWN= skips two lines at the top of each page so that the form units and the forms align correctly. SKIP= skips two lines between form units to maintain the proper alignment. COPIES= writes three copies of each form unit.

```
proc forms data=list file=labels
           ndown=5
           cc
           align=0
           width=24
           across=3
           down=2
           skip=2
```

```
copies=3;
```

Specify the variables to place on each line. The LINE statements specify the variables to place on each line. LASTNAME removes the comma from Name and writes the first name before the last name. PACK removes extra blank characters between City and State.

```
   line 1 name / lastname;
   line 2 street;
   line 3 city state zip / pack;
```

Specify the observations for the external file. The WHERE statement selects observations where State is one of the New England states.

```
   where state in('ME', 'NH', 'VT', 'MA', 'CT', 'RI');
run;
```

Output

```
Theresa Gabrielli        Theresa Gabrielli        Theresa Gabrielli
24 Ridgetop Rd.          24 Ridgetop Rd.          24 Ridgetop Rd.
Westboro MA 01581        Westboro MA 01581        Westboro MA 01581


Aria Clayton             Aria Clayton             Aria Clayton
314 Bridge St.           314 Bridge St.           314 Bridge St.
Hanover NH 03755         Hanover NH 03755         Hanover NH 03755


Martin L. Dix            Martin L. Dix            Martin L. Dix
4 Shepherd St.           4 Shepherd St.           4 Shepherd St.
Norwich VT 05055         Norwich VT 05055         Norwich VT 05055
```

CHAPTER

23

The FREQ Procedure

Overview: FREQ Procedure

The FREQ procedure is a descriptive as well as a statistical procedure that produces one-way to *n*-way frequency and crosstabulation tables. *Frequency tables* concisely describe your data by reporting the distribution of variable values. *Crosstabulation tables*, also known as *contingency tables*, summarize data for two or more classification variables by showing the number of observations for each combination of variable values.

For one-way frequency tables, PROC FREQ can compute statistics to test for equal proportions, specified proportions, or the binomial proportion. For contingency tables, PROC FREQ can compute various statistics to examine the relationships between two classification variables adjusting for any stratification variables. PROC FREQ automatically displays the output in a report and can also save the output in a SAS data set.

For some pairs of variables, you may want to examine the existence or the strength of any association between the variables. To determine the existence of an association, PROC FREQ computes statistics that test the null hypothesis of no association. To determine the strength of an association, PROC FREQ computes measures of association that tend to be close to zero when there is no association and close to their maximums (or minimums) when there is perfect association. The statistics for contingency tables include

- chi-square tests and measures
- measures of association and tests of these measures
- risks (or binomial proportions) and risk differences for 2×2 tables
- odds ratios and relative risks for 2×2 tables
- tests for trend
- tests and measures of agreement
- Cochran-Mantel-Haenszel statistics.

PROC FREQ computes asymptotic standard errors, confidence limits, and tests for measures of association and measures of agreement. Exact *p*-values and confidence limits are available for various test statistics and measures. PROC FREQ also performs stratified analyses that compute statistics within, as well as across, strata for *n*-way tables. The statistics include Cochran-Mantel-Haenszel statistics and measures of agreement.

Output 23.1 on page 515 is the simplest form of PROC FREQ output. The one-way frequency tables of hair and eye color show the distributions of these variables. PROC FREQ lists each variable value along with the frequencies and percentages. The statements that produce the output follow:

```
proc freq data=color;
run;
```

Output 23.1 One-Way Frequency Tables Produced with PROC FREQ

```
                              The SAS System                              1

                            The FREQ Procedure

                                Eye Color

                                             Cumulative    Cumulative
      Eyes     Frequency     Percent     Frequency      Percent
      ---------------------------------------------------------
      blue          222       29.13           222        29.13
      brown         341       44.75           563        73.88
      green         199       26.12           762       100.00


                               Hair Color

                                             Cumulative    Cumulative
     Hair      Frequency     Percent     Frequency      Percent
     ----------------------------------------------------------
     black           22        2.89            22         2.89
     dark           182       23.88           204        26.77
     fair           228       29.92           432        56.69
     medium         217       28.48           649        85.17
     red            113       14.83           762       100.00
```

In addition to listing the frequency distribution separately for each variable, you can create a crosstabulation table to show the joint frequency distribution for the two variables. Output 23.2 on page 516 shows a two-way crosstabulation table and chi-square statistics that test the association between eye and hair color of children from two regions of Europe. The statements that produce this 3×5 table also

- order the variable values according to their appearance in the data set
- exclude the row and column percentages for each cell
- include the expected frequency for each cell
- include each cell's contribution to the total Pearson chi-square statistic.

In addition to displaying the statistics, the program creates an output data set that contains selected chi-square statistics. For an explanation of the program that produces this output, see Example 5 on page 605.

Output 23.2 Chi-Square Statistics Produced with PROC FREQ

```
                 Chi-Square Tests for 3 by 5 Table of Eye and Hair Color                        1

                                   The FREQ Procedure

                                  Table of Eyes by Hair

                  Eyes(Eye Color)     Hair(Hair Color)

                  Frequency      |
                  Expected       |
                  Cell Chi-Square|
                  Percent        |fair    |red     |medium  |dark    |black   |  Total
                  ---------------+--------+--------+--------+--------+--------+
                  blue           |     69 |     28 |     68 |     51 |      6 |    222
                                 | 66.425 | 32.921 |  63.22 | 53.024 | 6.4094 |
                                 | 0.0998 | 0.7357 | 0.3613 | 0.0772 | 0.0262 |
                                 |   9.06 |   3.67 |   8.92 |   6.69 |   0.79 |  29.13
                  ---------------+--------+--------+--------+--------+--------+
                  green          |     69 |     38 |     55 |     37 |      0 |    199
                                 | 59.543 |  29.51 | 56.671 |  47.53 | 5.7454 |
                                 | 1.5019 | 2.4422 | 0.0492 | 2.3329 | 5.7454 |
                                 |   9.06 |   4.99 |   7.22 |   4.86 |   0.00 |  26.12
                  ---------------+--------+--------+--------+--------+--------+
                  brown          |     90 |     47 |     94 |     94 |     16 |    341
                                 | 102.03 | 50.568 | 97.109 | 81.446 | 9.8451 |
                                 | 1.4187 | 0.2518 | 0.0995 |  1.935 | 3.8478 |
                                 |  11.81 |   6.17 |  12.34 |  12.34 |   2.10 |  44.75
                  ---------------+--------+--------+--------+--------+--------+
                  Total               228      113      217      182       22      762
                                    29.92    14.83    28.48    23.88     2.89   100.00


                               Statistics for Table of Eyes by Hair

                         Statistic                     DF       Value      Prob
                         ------------------------------------------------------
                         Chi-Square                     8     20.9248    0.0073
                         Likelihood Ratio Chi-Square    8     25.9733    0.0011
                         Mantel-Haenszel Chi-Square     1      3.7838    0.0518
                         Phi Coefficient                       0.1657
                         Contingency Coefficient               0.1635
                         Cramer's V                            0.1172

                                   Sample Size = 762
```

Output 23.3 An Output Data Set That Contains Chi-Square Statistics

```
              Chi-Square Statistics for Eye and Hair Color              2
                Output Data Set from the FREQ Procedure

  N    NMISS    _PCHI_    DF_PCHI        P_PCHI    _LRCHI_    DF_LRCHI      P_LRCHI

 762     0     20.9248       8      .007349898    25.9733        8      .001061424
```

Several SAS procedures produce frequency counts; only PROC FREQ computes chi-square tests, measures of association, and measures of agreement for contingency tables. Other procedures to consider for counting are PROC TABULATE for more general table layouts; PROC REPORT for tables and customized summaries, PROC CHART for bar charts and other graphical representations; and PROC UNIVARIATE with the FREQ option for one-way frequency tables. When you want to fit models to categorical data, use a SAS/STAT procedure such as CATMOD, GENMOD, LOGISTIC, PHREG, or PROBIT. For more information on selecting the appropriate statistical analyses, refer to *An Introduction to Categorical Data Analysis* (Agresti, 1996) or *Categorical Data Analysis Using the SAS System* (Stokes, et al. 2000).

Syntax: FREQ Procedure

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can use the FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

PROC FREQ <*option(s)*>;
BY <DESCENDING> *variable-1* <…<DESCENDING> *variable-n*> <NOTSORTED>;
EXACT *statistic-keyword(s)* </ *option(s)*>;
OUTPUT *statistic-keyword(s)* <OUT=*SAS-data-set*>;
TABLES *request(s)* </ *option(s)*>;
TEST *statistic-keyword(s)*;
WEIGHT *variable*</ *option*>;

To do this	Use this statement
Calculate separate frequency or crosstabulation tables for each BY group	BY
Request exact tests for specified statistics	EXACT
Create an output data set that contains specified statistics	OUTPUT
Specify frequency or crosstabulation tables and request tests and measures of association	TABLES
Request asymptotic tests for measures of association and agreement	TEST
Identify a variable whose values weight each observation	WEIGHT

PROC FREQ Statement

PROC FREQ <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Control printed output	
Begin the next one-way table on the current page even if the entire table does not fit on that page	COMPRESS
Specify the outline and cell divider characters for the cells of the crosstabulation tables	FORMCHAR=
Suppress all displayed output	NOPRINT
Specify the order to list the variable values	ORDER=
Display one table per page	PAGE

Options

COMPRESS
begins to display the next one-way frequency table on the same page as the preceding one-way table when there is enough space to begin the table. By default, the next one-way table begins on the current page only if the entire table fits on that page.

Restriction: not valid with PAGE

Tip: COMPRESS saves paper and screen space.

DATA=*SAS-data-set*
specifies the input SAS data set.

Main discussion: “Procedure Concepts” on page 19

FORMCHAR <(*position(s)*)>=’*formatting-character(s)*’
defines the characters to use for constructing the outlines and dividers for the cells of crosstabulation tables.

position(s)
identifies the position of one or more characters in the SAS formatting-character string. A space or a comma separates the positions.

Default: Omitting (*position(s)*), is the same as specifying all 20 possible SAS formatting characters, in order.

Range: PROC FREQ uses formatting characters 1, 2, and 7. Table 23.1 on page 520 shows the formatting characters that PROC FREQ uses.

formatting-character(s)
lists the characters to use for the specified positions. PROC FREQ assigns characters in *formatting-character(s)* to *position(s)*, in the order that they are listed. For instance, the following option assigns the asterisk (*) to the second formatting character, the pound sign (#) to the seventh character, and does not alter the remaining characters:

```
formchar(2,7)='*#'
```

Interaction: The SAS system option FORMCHAR= specifies the default formatting characters. The system option defines the entire string of formatting characters. Specifying the FORMCHAR= option in a procedure can redefine selected characters.

Tip: You can use any character in *formatting-characters*, including hexadecimal characters. If you use hexadecimal characters, you must put an **x** after the closing quote. For example the following option assigns the hexadecimal character 2D to the second formatting character, the hexadecimal character 7C to the seventh character, and does not alter the remaining characters:

```
formchar(2,7)='2D7C'x
```

Tip: Assigning a blank space to each *formatting-character* produces tables without any outlines or dividers:

```
formchar (1,2,7)='   '
                  (3 spaces)
```

See also: For information on which hexadecimal codes to use for which characters, consult the documentation for your hardware.

Table 23.1 Formatting Characters Used by PROC FREQ

Position	Default	Used to draw
1	|	Vertical separators
2	-	Horizontal separators
7	+	Intersections of vertical and horizontal separators

NOPRINT

suppresses all displayed output from PROC FREQ.

Interaction: NOPRINT in the PROC statement disables the Output Delivery System for the entire PROC step.

Tip: Use NOPRINT when you want to create only an output data set with the OUTPUT statement or with the OUT= option in the TABLES statement.

Note: NOPRINT is also available in the TABLES statement where it suppresses the tables, but displays the requested statistics. △

ORDER=DATA | FORMATTED | FREQ | INTERNAL

orders the values of the frequency and crosstabulation table variables according to the specified order, where

DATA
: orders values according to their order in the input data set.

FORMATTED
: orders values by their formatted values. This order is operating environment-dependent. By default, the order is ascending.

FREQ
: orders values by descending frequency count.

INTERNAL
: orders values by their unformatted values, which yields the same order as PROC SORT. This order is operating environment-dependent.

Default: INTERNAL

Restriction: ORDER= does not apply to missing values, which always appear first.

Featured in: Example 2 on page 596 and Example 3 on page 599

PAGE

displays only one table per page.

Default: displays multiple tables per page as space permits

Restriction: not valid with COMPRESS

BY Statement

Calculates separate analysis for each BY group.

Main discussion: “Statements” on page 54

Featured in: Example 2 on page 596

BY <DESCENDING> *variable-1* <…<DESCENDING> *variable-n*> <NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are grouped in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

EXACT Statement

Requests exact tests or confidence limits for the specified statistics. Optionally requests Monte Carlo estimates of the exact *p*-values.

Requirements: TABLES statement

Main discussion: “Exact Statistics” on page 581

Featured in: Example 4 on page 601

EXACT *statistic-keyword(s)* </ *option(s)*>;

Required Arguments

statistic-keyword(s)

specifies the statistics for which to provide exact tests or confidence limits. PROC FREQ can compute exact *p*-values for the following hypothesis tests: chi-square goodness-of-fit for one-way tables; Pearson chi-square, likelihood-ratio chi-square, Mantel-Haenszel chi-square, Fisher's exact test, Jonckheere-Terpstra test, Cochran-Armitage test for trend, and McNemar's test for two-way tables. PROC FREQ can also compute exact *p*-values for tests of hypotheses that the following statistics are equal to zero: Pearson correlation coefficient, Spearman correlation coefficient, simple kappa coefficient, and weighted kappa coefficient. PROC FREQ can compute exact *p*-values for the binomial proportion test, as well as exact confidence limits for the binomial proportion. Additionally, PROC FREQ can compute exact confidence limits for odds ratios for 2×2 tables.

The statistic keywords are identical to options in the TABLES statement and keywords in the OUTPUT statement. You can request exact computations for groups of statistics by using keywords that are identical to the following TABLES statement options: CHISQ, MEASURES, and AGREE. For example, when you specify CHISQ in the EXACT statement, PROC FREQ computes exact *p*-values for the available CHISQ statistics (Pearson chi-square, likelihood-ratio chi-square, and Mantel-Haenszel chi-square). You request exact *p*-values for an individual statistic by specifying a keyword shown in Table 23.2 on page 522.

Note: PROC FREQ computes exact tests by using fast and efficient algorithms that are superior to direct enumeration. This technique is appropriate when a data set is small, sparse, skewed, or heavily tied. For some large problems, exact computations may require a large amount of time or memory. Consider using the asymptotic tests for such problems. Alternatively, when asymptotic methods may not be sufficient for such large problems, consider using Monte Carlo estimation of exact *p*-values. See "Exact Statistics" on page 581 for more information. △

Table 23.2 EXACT Statement Statistic-Keywords and Required TABLES Statement Options

Keyword	Exact statistics computed	Required TABLES statement option
AGREE	McNemar's test for 2×2 tables and tests for the simple kappa coefficient and the weighted kappa coefficient	AGREE
BINOMIAL	binomial proportion test for one-way tables	BINOMIAL
CHISQ	chi-square goodness-of-fit test for one-way tables; Pearson chi-square, likelihood-ratio chi-square, and Mantel-Haenszel chi-square tests for two-way tables	ALL, CHISQ
FISHER	Fisher's exact test	ALL*, CHISQ*
JT	Jonckheere-Terpstra test	JT
KAPPA	test for the simple kappa coefficient	AGREE
LRCHI	likelihood-ratio chi-square test	ALL, CHISQ
MCNEM	McNemar's test for 2×2 tables	AGREE

Keyword	Exact statistics computed	Required TABLES statement option
MEASURES	tests for the Pearson correlation coefficient and the Spearman correlation and the odds ratio confidence limits for 2×2 tables	ALL, MEASURES
MHCHI	Mantel-Haenszel chi-square test	ALL, CHISQ
OR	odds ratio confidence limits for 2×2 tables	ALL, MEASURES, RELRISK
PCHI	chi-square goodness-of-fit test for one-way tables, Pearson chi-square test for 2×2 tables	ALL, CHISQ
PCORR	test for the Pearson correlation coefficient	ALL, MEASURES
SCORR	test for the Spearman correlation coefficient	ALL, MEASURES
TREND	Cochran-Armitage test for trend	TREND
WTKAP	test for the weighted kappa coefficient	AGREE

* ALL and CHISQ compute Fisher's exact test only for 2×2 tables.

Options

ALPHA=*p*

specifies the confidence level for the confidence limits for the Monte Carlo *p*-value estimates. A confidence level of *p* results in (1–*p*)×100 percent confidence limits. Using ALPHA=.01 results in 99 percent confidence limits. If *p* is between 0 and 1 but is outside the range, PROC FREQ uses the closest range endpoint. For example, if *p*= 0.000001, PROC FREQ uses 0.0001 to determine confidence limits.

Default: 0.01

Range: 0.000<=*p*<=0.0001

Interaction: ALPHA= invokes the MC option.

MAXTIME=*value*

specifies the maximum clock time (in seconds) that PROC FREQ uses to compute an exact *p*-value directly or with Monte Carlo estimation. If the procedure does not complete the computation within the specified time, the computation terminates.

Range: a positive number

See also: "Computational Resources" on page 583

Featured in: Example 7 on page 611

MC

requests Monte Carlo estimation of exact *p*-values, instead of direct exact *p*-value computation. Monte Carlo estimation can be useful for large problems that require a large amount of time and memory for exact computations, but for which asymptotic approximations may not be sufficient.

Restriction: The MC option is not available with the POINT option.

Restriction: The MC option is available for all statistic keywords except BINOMIAL, MCNEM, and OR. PROC FREQ computes only exact tests or confidence limits for those statistics.

Interaction: ALPHA=, N=, and SEED= automatically invoke the MC option.

Tip: Use MAXTIME= to specify the maximum clock time that PROC FREQ can use for the Monte Carlo estimation.

Main Discussion: "Monte Carlo Estimation" on page 584

N=*n*
specifies the number of samples for Monte Carlo estimation.

Default: 10000

Range: a positive integer

Interaction: N= invokes the MC option.

Tip: Larger values of N= produce more precise estimates of exact *p*-values. Because larger values of N= generate more samples, the computation time increases. If you need more computation time, use MAXTIME= to increase the clock time.

POINT
requests exact point probabilities for the test statistics.

Restriction: The POINT option is available for all statistical keywords except OR, which provides exact confidence limits as opposed to an exact test.

Restriction: The POINT option is not available with the MC option.

Main Discussion: "Exact Statistics" on page 581

SEED=*n*
specifies the initial seed for random number generation for Monte Carlo estimation.

Default: the time of day from the computer's clock

Range: a positive integer

Interaction: SEED= invokes the MC option.

Using TABLES Statement Options with the EXACT Statement

Table 23.2 on page 522 lists the available statistic keywords and the exact statistics that are computed. If you use only one TABLES statement, you do not need to specify options in the TABLES statement to compute the statistics that the EXACT statement requests. PROC FREQ automatically invokes the corresponding TABLES statement option when you request exact computations. However, when you use multiple TABLES statements, and you want exact computations, you must specify options in the TABLES statement to compute the desired statistics. Then PROC FREQ performs exact computations for all statistics that are also specified in the EXACT statement.

OUTPUT Statement

Creates a SAS data set with the statistics that PROC FREQ computes for the last TABLES statement request. The variables contain statistics for each two-way table or stratum, as well as summary statistics across all strata.

Requirements: TABLES statement

Restriction: Only one OUTPUT statement is allowed.

Tip: Use the Output Delivery System to create a SAS data set from any piece of PROC FREQ output.

Main discussion: "Output Data Sets" on page 590

Featured in: Example 5 on page 605

OUTPUT *statistic-keyword(s)* <OUT=*SAS-data-set*>;

Options

OUT=*SAS-data-set*

names the output data set that contains statistics for the last TABLES statement request. If you omit OUT=, the data set is named DATA*n*, where *n* is the smallest integer that makes the name unique.

Default: DATA*n*

statistic-keyword(s)

specifies the statistics that you want in the new data set. Available statistics are those produced by PROC FREQ for each one-way or two-way table, as well as summary statistics across all strata. When you request a statistic, the OUTPUT data set contains that estimate or test statistic, as well as any associated standard error, degrees of freedom, confidence limits, and *p*-values.

You can save statistics by using keywords that are identical to group options in the TABLES statement: AGREE, ALL, CHISQ, CMH, and MEASURES. Alternatively, you can request an individual statistic by specifying a keyword shown in Table 23.3 on page 525.

Using the TABLES Statement with the OUTPUT Statement

In order to specify that the OUTPUT data set contain a particular statistic, you must have PROC FREQ compute the statistic by using the corresponding option in the TABLES statement or the EXACT statement. For example with a 2×2 table, you cannot specify the keyword OR (odds ratio) in the OUTPUT statement without also specifying ALL, MEASURES, or RELRISK in the TABLES statement.

If you use multiple TABLES statements, the contents of the OUTPUT data set correspond to the last TABLES statement. If you use multiple table requests in a TABLES statement, the contents of the OUTPUT data set correspond to the last table request.

Table 23.3 OUTPUT Statement Statistic-Keywords and Required TABLES Statement Options

Keyword	Output data set statistics	Required TABLES statement option
AGREE	McNemar's test for 2×2 tables, simple kappa coefficient, and weighted kappa coefficient. For square tables with more than two response categories, Bowker's test of symmetry. For multiple strata, overall simple and weighted kappa statistics, and tests for equal kappas among strata. For multiple strata with two response categories, Cochran's *Q* test.	AGREE
AJCHI	continuity-adjusted chi-square for 2×2 tables	ALL, CHISQ
ALL	all statistics under CHISQ, MEASURES, CMH, and the number of nonmissing subjects	ALL
BDCHI	Breslow-Day test	ALL, CMH, CMH1, CMH2
BINOMIAL	binomial proportion statistics for one-way tables	BINOMIAL, BINOMIALC
CHISQ	chi-square goodness-of-fit test for one-way tables; for two-way tables, Pearson chi-square, likelihood ratio chi-square, continuity-adjusted chi-square for 2×2 tables, Mantel-Haenszel chi-square, Fisher's exact test for 2×2 tables, phi coefficient, contingency coefficient, and Cramer's *V*	ALL, CHISQ

Keyword	Output data set statistics	Required TABLES statement option
CMH	Cochran-Mantel-Haenszel correlation, row mean scores (*ANOVA*), and general association statistics; for 2×2 tables, logit and Mantel-Haenszel adjusted odds ratios, relative risks, and Breslow-Day test	ALL, CMH
CMH1	same as CMH, but excludes general association and row mean scores (*ANOVA*) statistics	ALL, CMH, CMH1, CMH2
CMH2	same as CMH, but excludes the general association statistic	ALL, CMH, CMH2
CMHCOR	Cochran-Mantel-Haenszel correlation statistic	ALL, CMH, CMH1, CMH2
CMHGA	Cochran-Mantel-Haenszel general association statistic	ALL, CMH
CMHRMS	Cochran-Mantel-Haenszel row mean scores (*ANOVA*) statistic	ALL, CMH, CMH2
COCHQ	Cochran's Q	AGREE
CONTGY	contingency coefficient	ALL, CHISQ
CRAMV	Cramer's V	ALL, CHISQ
EQKAP	test for equal simple kappas	AGREE
EQWKP	test for equal weighted kappas	AGREE
FISHER | EXACT	Fisher's exact test	ALL*, CHISQ*, FISHER, EXACT
GAMMA	gamma	ALL, MEASURES
JT	Jonckheere-Terpstra test	JT
KAPPA	simple kappa coefficient	AGREE
KENTB	Kendall's tau-b	ALL, MEASURES
LAMCR	lambda asymmetric ($C \mid R$)	ALL, MEASURES
LAMDAS	lambda symmetric	ALL, MEASURES
LAMRC	lambda asymmetric ($R \mid C$)	ALL, MEASURES
LGOR	adjusted logit odds ratio	ALL, CMH, CMH1, CMH2
LGRRC1	adjusted logit column 1 relative risk	ALL, CMH, CMH1, CMH2
LGRRC2	adjusted logit column 2 relative risk	ALL, CMH, CMH1, CMH2
LRCHI	likelihood ratio chi-square	ALL, CHISQ
MCNEM	McNemar's test	AGREE
MEASURES	gamma, Kendall's tau-b, Stuart's tau-c, Somers' D ($C \mid R$), Somers' D ($R \mid C$), Pearson correlation coefficient, Spearman correlation coefficient, lambda asymmetric ($C \mid R$), lambda asymmetric ($R \mid C$), lambda symmetric, uncertainty coefficient ($C \mid R$), uncertainty coefficient ($R \mid C$), and symmetric uncertainty coefficient; for 2×2 tables, odds ratio and relative risks	ALL, MEASURES
MHCHI	Mantel-Haenszel chi-square	ALL, CHISQ
MHOR	adjusted Mantel-Haenszel odds ratio	ALL, CMH, CMH1, CMH2
MHRRC1	adjusted Mantel-Haenszel column 1 relative risk	ALL, CMH, CMH1, CMH2
MHRRC2	adjusted Mantel-Haenszel column 2 relative risk	ALL, CMH, CMH1, CMH2

Keyword	Output data set statistics	Required TABLES statement option
N	number of nonmissing subjects for the stratum	
NMISS	number of missing subjects for the stratum	
OR	odds ratio	ALL, MEASURE, RELRISK
PCHI	chi-square goodness-of-fit test for one-way tables; for 2-way tables, Pearson chi-square	ALL, CHISQ
PCORR	Pearson correlation coefficient	ALL, MEASURES
PHI	phi coefficient	ALL, CHISQ
PLCORR	polychoric correlation coefficient	PLCORR
RDIF1	column 1 risk difference (row 1 – row 2)	RISKDIFF, RISKDIFFC
RDIF2	column 2 risk difference (row 1 – row 2)	RISKDIFF, RISKDIFFC
RELRISK	odds ratio and relative risks for 2×2 tables	ALL, MEASURE, RELRISK
RISKDIFF	risks and risk differences	RISKDIFF, RISKDIFFC
RISKDIFF1	column 1 risks and risk difference	RISKDIFF, RISKDIFFC
RISKDIFF2	column 2 risks and risk difference	RISKDIFF, RISKDIFFC
RRC1	column 1 relative risk	ALL, MEASURE, RELRISK
RRC2	column 2 relative risk	ALL, MEASURE, RELRISK
RSK1	column 1 risk (overall)	RISKDIFF, RISKDIFFC
RSK11	column 1 risk, for row 1	RISKDIFF, RISKDIFFC
RSK12	column 2 risk, for row 1	RISKDIFF, RISKDIFFC
RSK2	column 2 risk (overall)	RISKDIFF, RISKDIFFC
RSK21	column 1 risk, for row 2	RISKDIFF, RISKDIFFC
RSK22	column 2 risk, for row 2	RISKDIFF, RISKDIFFC
SCORR	Spearman correlation coefficient	ALL, MEASURES
SMDCR	Somers' D $(C \mid R)$	ALL, MEASURES
SMDRC	Somers' D $(R \mid C)$	ALL, MEASURES
STUTC	Stuart's tau-c	ALL, MEASURES
TREND	Cochran-Armitage test for trend	TREND
TSYMM	Bowker's test of symmetry	AGREE
U	symmetric uncertainty coefficient	ALL, MEASURES
UCR	uncertainty coefficient $(C \mid R)$	ALL, MEASURES
URC	uncertainty coefficient $(R \mid C)$	ALL, MEASURES
WTKAP	weighted kappa coefficient	AGREE

* ALL and CHISQ compute Fisher's exact test only for 2×2 tables. Use the FISHER option to compute Fisher's exact test for general $r \times c$ tables.

TABLES Statement

Requests one-way to *n*-way frequency and crosstabulation tables and computes the statistics for these tables.

Default: If you omit the TABLES statement, PROC FREQ generates one-way frequency tables for all data set variables that are not listed in the other statements.

Featured in: Example 1 on page 592

TABLES *request(s)* </ *option(s)*>;

Required Arguments

request(s)

specifies the frequency and crosstabulation tables to produce. A request is composed of one variable name or several variable names that are separated by asterisks. To request a one-way frequency table, use a single variable. To request a two-way crosstabulation table, use an asterisk between two variables. To request a multiway table (an *n*-way table, where *n*>2), separate the desired variables with asterisks. The unique values of these variables form rows, columns, and strata of the table.

For two-way to multiway tables, the values of the last variable form the crosstabulation table columns while the values of the next-to-last variable form the rows. Each level (or combination of levels) of the other variables forms one stratum. PROC FREQ produces a separate crosstabulation table for each stratum. For example, the TABLES statement request A*B*C*D produces *k* tables, where *k* is the number of different combinations of values for A and B. Each table lists the values for C down the side and the values for D across the top.

You can use multiple TABLES statements in the PROC FREQ step. PROC FREQ builds all the table requests in one pass of the data so that there is essentially no loss of efficiency. You can also specify any number of table requests in a single TABLES statement. To specify multiple table requests quickly, use a grouping syntax by placing parentheses around several variables and joining other variables or variable combinations. For example, the following statements illustrate grouping syntax:

Request	**Equivalent to**
`tables a*(b c);`	`tables a*b a*c;`
`tables (a b)*(c d);`	`tables a*c b*c a*d b*d;`
`tables (a b c)*d;`	`tables a*d b*d c*d;`
`tables a--c;`	`tables a b c;`
`tables (a--c)*d;`	`tables a*d b*d c*d;`

Without Options

If you request a one-way frequency table for a variable without specifying options, PROC FREQ produces frequencies, cumulative frequencies, percentages of the total frequency, and cumulative percentages for each value of the variable. If you request a two-way or *n*-way crosstabulation table without specifying options, PROC FREQ produces crosstabulation tables that include cell frequencies, cell percentages of the total frequency, cell percentages of row frequencies, and cell percentages of column frequencies. The procedure excludes observations with missing values from the table, but displays the total frequency of missing observations below each table.

Options

To do this	Use this option
Control statistical analysis	
Request tests and measures of classification agreement	AGREE
Request tests and measures of association produced by CHISQ, MEASURES, and CMH	ALL
Set the confidence level for confidence limits	ALPHA=
Request Tarone's adjustment in the Breslow-Day test for homogeneity of odds ratios	BDT
Request binomial proportion, confidence limits, and test for one-way tables	BINOMIAL
Request binomial proportion, confidence limits, and test, and include a continuity correction	BINOMIALC
Request BINOMIAL statistics, and include a continuity correction in the asymptotic confidence limits and test	CHISQ
Request confidence limits for the MEASURES statistics	CL
Request all Cochran-Mantel-Haenszel statistics, adjusted relative risks, and odds ratios	CMH
Request adjusted relative risks and odds ratios and CMH correlation statistic	CMH1
Request adjusted relative risks and odds ratios, CMH correlation, and row mean scores (ANOVA) statistic	CMH2
Specify convergence criterion to compute polychoric correlation	CONVERGE=
Request Fisher's exact test for tables larger than 2×2	FISHER
Request Jonckheere-Terpstra test	JT
Specify maximum number of iterations to compute polychoric correlation	MAXITER=
Request measures of association and their asymptotic standard errors	MEASURES
Treat missing values as nonmissing	MISSING
Request polychoric correlation	PLCORR
Request relative risk measures for 2×2 tables	RELRISK
Request risks and risk differences for 2×2 tables	RISKDIFF
Request risks and risk differences and include a continuity correction	RISKDIFFC

To do this	Use this option
Specify the type of row and column scores	SCORES=
Specify expected frequencies for a one-way table chi-square test	TESTF=
Specify expected proportions for a one-way table chi-square test	TESTP=
Request Cochran-Armitage test for trend	TREND
Control additional table information	
Report each cell's contribution to the total Pearson chi-square statistic	CELLCHI2
Display the cumulative column percentage in each cell	CUMCOL
Display the deviation of the cell frequency from the expected value for each cell	DEVIATION
Display the expected cell frequency for each cell	EXPECTED
Display missing value frequencies	MISSPRINT
List all possible combinations of variable levels even when a combination does not occur	SPARSE
Display percentage of total frequency on *n*-way tables when n>2	TOTPCT
Control displayed output	
Specify the HTML contents link for crosstabulation tables	CONTENTS=
Format the frequencies in crosstabulation tables	FORMAT=
Display two-way to *n*-way tables in list format	LIST
Suppress the column percentage for each cell	NOCOL
Suppress the cumulative frequencies and the cumulative percentages in one-way frequency tables and in list format	NOCUM
Suppress the frequency count for each cell	NOFREQ
Suppress the percentage, row total percentage, and column total percentage in crosstabulation tables, or percentages and cumulative percentages in one-way frequency tables and in list format	NOPERCENT
Suppress the display of tables but report the statistics	NOPRINT
Suppress the row percentage for each cell	NOROW
Suppress a log warning message for the asymptotic chi-square test	NOWARN
Display the kappa coefficient weights	PRINTKWT
Display the row and the column scores	SCOROUT
Use a field 8 positions wide to display the cell frequencies between 1.E7 and 1.E8	V5FMT
Create an output data set	
Specify an output data set to contain variable values and frequency counts	OUT=
Include the cumulative frequency and cumulative percent for one-way tables in the output data set	OUTCUM

To do this	Use this option
Include the expected frequency of each cell in the output data set	OUTEXPECT
Include the percentage of column frequency, row frequency, and two-way table frequency in the output data set	OUTPCT

AGREE <(WT=*type*)>

requests tests and measures of classification agreement for square tables. The AGREE option provides McNemar's test for 2×2 tables and Bowker's test of symmetry for tables with more than two response categories. The AGREE option also produces the simple kappa statistic, the weighted kappa statistic, their asymptotic standard errors, and the corresponding confidence limits. When there are multiple strata, PROC FREQ computes overall simple and weighted kappa statistics, as well as tests for equal kappas among strata. When there are multiple strata and two response categories, PROC FREQ computes Cochran's *Q* test.

(WT=*type*)

specifies the type of weights that PROC FREQ uses to compute the weighted kappa coefficient, where *type* is the following:

CA	Cicchetti-Allison weights
FC	Fleiss-Cohen weights

Default: CA

Main discussion: "Weighted Kappa Coefficient" on page 571

Restriction: The table must be square.

Tip: If the table is not square due to observations with zero weights, you can use the ZEROS option in the WEIGHT statement to include these observations. For more details, see "Tables with Zero Rows or Columns" on page 574.

Tip: You can specify PRINTKWT to display the kappa coefficient weights.

Main discussion: "Tests and Measures of Agreement" on page 569

Featured in: Example 9 on page 618

ALL

requests all tests and measures that are computed by the CHISQ, MEASURES, and CMH options.

Interaction: CMH1 and CMH2 control which CMH statistics PROC FREQ computes.

ALPHA=*p*

sets the confidence level for confidence limits. The percentage for the confidence limits is (1–*p*)×100. Using ALPHA=.05 results in 95 percent confidence limits. If *p* is between 0 and 1 but is outside the range, PROC FREQ uses the closest range endpoint. For example, if *p*=0.000001, PROC FREQ uses 0.0001 to determine confidence limits.

Default: 0.05

Range: 0.0001<=*p*<=0.9999

BDT

requests Tarone's adjustment in the Breslow-Day test for homogeneity of odds ratios.

Requirement: You must specify CMH to compute the Breslow-Day test for stratified 2×2 tables.

Main discussion: "Breslow-Day Test for Homogeneity of the Odds Ratios" on page 580

BINOMIAL <(P=*value*) | (LEVEL=*level-number* | *level-value*)>
computes the binomial proportion for one-way tables. BINOMIAL also computes the asymptotic standard error, asymptotic and exact confidence limits, and the asymptotic test for the binomial proportion. To specify the null hypothesis proportion for the test, use P=. By default BINOMIAL computes the proportion of observations for the first variable level that appears in the output. To specify a different level, use LEVEL= *level-number* or LEVEL=*level-value*, where *level-number* is the variable level's number or order in the output, and *level-value* is the formatted value of the variable level.

Default: P=0.5, LEVEL=1

Restriction: for one-way tables

Interaction: To request an exact test for the binomial proportion, specify BINOMIAL in the EXACT statement.

Tip: To include a continuity correction in the binomial asymptotic confidence limits and test, use BINOMIALC instead of BINOMIAL.

Main Discussion: "Binomial Proportion" on page 560

Featured in: Example 3 on page 599

BINOMIALC <(P=*value*) | (LEVEL=*level-number* | *level-value*)>
computes the BINOMIAL option statistics for one-way tables, but includes a continuity correction in the asymptotic confidence limits and asymptotic test. The BINOMIAL option statistics include the binomial proportion, its asymptotic and exact confidence limits, and the asymptotic test for the binomial proportion. To specify the null hypothesis proportion for the test, use P=. By default BINOMIALC computes the proportion of observations for the first variable level that appears in the output. To specify a different level, use LEVEL=*level-number* or LEVEL=*level-value*, where *level-number* is the variable level's number or order in the output, and *level-value* is the formatted value of the variable level.

Alias: BINC

Default: P=0.5, LEVEL=1

Restriction: for one-way tables

Interaction: To request an exact test for the binomial proportion, specify BINOMIAL in the EXACT statement.

Tip: To request binomial statistics without the continuity correction, use BINOMIAL instead of BINOMIALC.

Main Discussion "Binomial Proportion" on page 560

CELLCHI2
displays each cell's contribution to the total Pearson chi-square statistic, which is computed as $(frequency - expected)^2/expected$.

Interaction: CELLCHI2 is valid for contingency tables but has no effect on tables that are produced with LIST.

CHISQ
computes chi-square tests of homogeneity or independence for two-way tables, and computes measures of association based on chi-square for two-way tables. The tests include Pearson chi-square, likelihood-ratio chi-square, and Mantel-Haenszel chi-square. The measures include the phi coefficient, the contingency coefficient, and Cramer's *V*. For 2×2 tables, CHISQ includes Fisher's exact test and the continuity-adjusted chi-square. For one-way tables, CHISQ computes a chi-square goodness-of-fit test for equal proportions. If you specify the null hypothesis

proportions with the TESTP= option, then CHISQ computes a chi-square goodness-of-fit test for the specified proportions. If you specify null hypothesis frequencies with the TESTF= option, CHISQ computes a chi-square goodness-of-fit test for the specified frequencies.

Main discussion: "Chi-Square Tests and Statistics" on page 546

Featured in: Example 4 on page 601 and Example 5 on page 605

CL

requests confidence limits for the MEASURES statistics.

Interaction: If you omit MEASURES, CL invokes MEASURES.

Interaction: PROC FREQ determines the confidence coefficient using ALPHA= , which by default equals 0.05 and produces 95 percent confidence limits.

Main discussion: "Measures of Association" on page 550

Featured in: Example 7 on page 611

CMH

computes Cochran-Mantel-Haenszel statistics, which test for association between the row and column variables after adjusting for the remaining variables in a multiway table. In addition, for 2×2 tables, PROC FREQ computes adjusted Mantel-Haenszel and logit estimates of the odds ratio and relative risks as well as the corresponding confidence limits. For the stratified 2×2 case, PROC FREQ computes the Breslow-Day test for homogeneity of odds ratios.

Interaction: CMH1 and CMH2 control the number of CMH statistics that PROC FREQ computes.

Tip: Use BDT to request Tarone's adjustment in the Breslow-Day test.

Main discussion: "Cochran-Mantel-Haenszel Statistics" on page 574

Featured in: Example 6 on page 609

CMH1

requests the Cochran-Mantel-Haenszel correlation statistic and, for 2×2 tables, adjusted Mantel-Haenszel and logit estimates of the odds ratio and relative risks as well as the corresponding confidence limits. For the stratified 2×2 case, PROC FREQ computes the Breslow-Day test for homogeneity of odds ratios. Except for 2×2 tables, CMH1 requires less memory than CMH, which can require an enormous amount for large tables.

CMH2

requests the Cochran-Mantel-Haenszel correlation statistic, row mean scores (*ANOVA*) statistic and, for 2×2 tables, adjusted Mantel-Haenszel and logit estimates of the odds ratio and relative risks as well as the corresponding confidence limits. For the stratified 2×2 case, PROC FREQ computes the Breslow-Day test for homogeneity of odds ratios. Except for tables with two columns, CMH2 requires less memory than CMH, which can require an enormous amount for large tables.

Featured in: Example 8 on page 615

CONTENTS=

specifies the text for the HTML contents file links to crosstabulation tables. For information on HTML output, see *SAS Output Delivery System User's Guide*. The CONTENTS= option affects only the HTML contents file, and not the HTML body file.

Note: Links to all crosstabulation tables produced by a single TABLES statement use the same text. To specify different text for different crosstabulation table links, request the tables in separate TABLES statements and use the CONTENTS= option in each TABLES statement. △

Default: Cross-Tabular Freq Table

Tip: The CONTENTS= option affects only links to crosstabulation tables. It does not affect links to other PROC FREQ tables. To specify link text for any other PROC FREQ table, you can use PROC TEMPLATE to create a customized table definition. The CONTENTS_LABEL attribute in the DEFINE TABLE statement of PROC TEMPLATE specifies the contents file link for the table. For detailed information, see the discussion of the TEMPLATE procedure in *SAS Output Delivery System User's Guide* .

CONVERGE=*c*
specifies the convergence criterion for computing the polychoric correlation using the PLCORR option. Iterative computation of the polychoric correlation stops when the convergence measure falls below the value of CONVERGE=, or when the number of iterations that is specified by the MAXITER= option is exceeded, whichever happens first.

Alias: CONV=

Default: 0.0001

Range: a positive number

Main discussion: "Polychoric Correlation" on page 558

CUMCOL
displays the cumulative column percentages in cells of the crosstabulation table.

DEVIATION
displays the deviation of the cell frequency from the expected frequency for each cell of the crosstabulation table.

Interaction: DEVIATION is valid for crosstabulation tables but has no effect on tables produced with LIST.

Featured in: Example 5 on page 605

EXPECTED
displays the expected cell frequencies under the hypothesis of independence (or homogeneity).

Interaction: EXPECTED is valid for contingency tables but has no effect on tables produced with LIST.

Featured in: Example 5 on page 605

FISHER
computes Fisher's exact test even when tables are larger than 2×2. You can also request Fisher's exact test by specifying FISHER in the EXACT statement.

Alias: EXACT

Interaction: If you omit CHISQ, FISHER invokes CHISQ.

Interaction: ALL does not invoke this option.

Main discussion: "Fisher's Exact Test" on page 548

CAUTION:
For large tables, PROC FREQ may require a large amount of time or memory to compute exact *p*-values. See "Computational Resources" on page 583 for more information. △

FORMAT=*format-name*
specifies a format for the following crosstabulation table cell values: frequency, expected frequency, and deviation. PROC FREQ also uses this format to display the total row and column frequencies for crosstabulation tables. You can specify any standard SAS numeric format or a numeric format defined with the FORMAT procedure.

Note: To change formats for all other FREQ tables, use PROC TEMPLATE. For detailed information, see the discussion of the TEMPLATE procedure in *SAS Output Delivery System User's Guide*. △

Default: If you omit FORMAT=, PROC FREQ uses the BEST6. format to display frequencies less than 1E6, and the BEST7. format otherwise.

Restriction: You can not specify both FORMAT= and the V5FMT option.

Restriction: The format length must not exceed 24.

See also: For more information on using formats, see *SAS Language Reference: Dictionary*

JT

performs the Jonckheere-Terpstra test.

Main discussion: "Jonckheere-Terpstra Test" on page 567

LIST

displays two-way to *n*-way tables in a list format rather than as crosstabulation tables.

Restriction: PROC FREQ ignores LIST when you request statistical tests or measures of association.

MAXITER=*n*

specifies the maximum number of iterations for computing the polychoric correlation using the PLCORR option. Iterative computation of the polychoric correlation stops when the number of iterations that is specified by MAXITER= is exceeded, or when the convergence measure falls below the value of the CONVERGE= option, whichever happens first.

Default: 20

Range: an integer between 0 and 32767

Main discussion: "Polychoric Correlation" on page 558

MEASURES

requests several measures of association and their asymptotic standard errors (ASE). The measures include gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' *D*, Pearson and Spearman correlation coefficients, lambda (asymmetric and symmetric), uncertainty coefficients (asymmetric and symmetric) and, for 2×2 tables, the odds ratio, column 1 relative risk, column 2 relative risk, and the corresponding confidence limits.

Interaction: CL requests confidence limits.

Main discussion: "Measures of Association" on page 550

Featured in: Example 7 on page 611

MISSING

treats missing values as nonmissing and includes them in calculations of percentages and other statistics.

Main discussion: "Missing Values" on page 585

MISSPRINT

displays missing value frequencies for all tables, even though PROC FREQ does not use the frequencies in the calculation of statistics.

Main discussion: "Missing Values" on page 585

NOCOL

suppresses the column percentages in cells of the crosstabulation table.

Featured in: Example 5 on page 605

NOCUM

suppresses the cumulative frequencies and cumulative percentages for one-way frequency tables and for frequencies in list format.

Featured in: Example 2 on page 596

NOFREQ
suppresses the cell frequencies for a crosstabulation table. This also suppresses frequencies for row totals.

NOPERCENT
suppresses the cell percentages, the row total percentages, and the column total percentages for a crosstabulation table. For one-way frequency tables and frequencies in list format, suppresses the percentages and the cumulative percentages.

NOPRINT
suppresses the frequency and crosstabulation tables, but displays all requested tests and statistics.

Featured in: Example 6 on page 609

NOROW
suppresses the row percentages in cells of the crosstabulation table.

Featured in: Example 5 on page 605

NOWARN
suppresses the log warning message that the asymptotic chi-square test may not be valid when more than 20 percent of the table cells have expected frequencies less than five.

OUT=*SAS-data-set*
names the output data set that contains variable values and frequency counts. The variable COUNT contains the frequencies and the variable PERCENT contains the percentages. If more than one table request appears in the TABLES statement, the contents of the data set correspond to the last table request in the TABLES statement.

Main discussion: "Output Data Sets" on page 590

See also: OUTEXPECT and OUTPCT

Featured in: Example 1 on page 592

OUTCUM
includes the cumulative frequency and the cumulative percent for one-way tables in the output data set when you specify the OUT= option. The variable CUM_FREQ contains the cumulative frequency for each level of the analysis variable, and the variable CUM_PCT contains the cumulative percent for each level.

Requirement: This option is available when you specify the OUT= option.

Interaction: The OUTCUM option has no effect on two-way or multi-way tables.

OUTEXPECT
includes the expected frequency in the output data set when you specify the OUT= option. The variable EXPECTED contains the expected frequency for each table cell.

Requirement: This option is available when you specify the OUT= option.

Main discussion: "Output Data Sets" on page 590

Featured in: Example 1 on page 592

OUTPCT
includes the following additional variables in the output data set when you specify the OUT= option:

PCT_COL

the percentage of column frequency

PCT_ROW
the percentage of row frequency

PCT_TABL
the percentage of stratum frequency, for n-way tables where $n > 2$.

Requirement: This option is available when you specify the OUT= option.

Main discussion: "Output Data Sets" on page 590

PLCORR
computes the polychoric correlation coefficient. For 2×2 tables, this statistic is more commonly known as the tetrachoric correlation coefficient, and is labeled as such in the displayed output.

Interaction: If you omit MEASURES, PLCORR invokes MEASURES.

Main discussion: "Polychoric Correlation" on page 558

See also: CONVERGE= and MAXITER=

PRINTKWT
requests that PROC FREQ display the kappa coefficient weights.

Interaction: You must specify AGREE to compute the kappa coefficients. The WT= option controls how PROC FREQ computes the kappa coefficient weights.

Main discussion: "Weighted Kappa Coefficient" on page 571

RELRISK
requests relative risk measures for 2×2 tables. These measures include the odds ratio, column 1 relative risk, and column 2 relative risk.

Main discussion: "Odds Ratio and Relative Risks for 2×2 Tables" on page 564

Featured in: Example 4 on page 601

RISKDIFF
requests column 1 and 2 risks (or binomial proportions), risk differences, and their confidence limits for 2×2 tables.

Alias: PDIFF, RDIFF

Main discussion: "Risks and Risk Differences" on page 562

RISKDIFFC
requests the RISKDIFF statistics for 2×2 tables, but includes a continuity correction in the asymptotic confidence limits. The RISKDIFF option statistics include the column 1 and column 2 risks (or binomial proportions), risk differences, and their confidence limits.

Main Discussion "Risks and Risk Differences" on page 562

SCORES=*type*
specifies the type of row and column scores that PROC FREQ uses with the Mantel-Haenszel chi-square, Pearson correlation, Cochran-Armitage test for trend, weighted kappa coefficient, and Cochran-Mantel-Haenszel statistics where *type* is

MODRIDIT

RANK

RIDIT

TABLE

By default, the row or column scores are the integers 1,2,… for character variables and the actual variable values for numeric variables. Using other types of scores yields nonparametric analyses.

Default: TABLE

Main discussion: "Scores" on page 545

Featured in: Example 8 on page 615

SCOROUT

displays the row and the column scores. You specify the score type with the SCORES= option. PROC FREQ uses the scores when it calculates the Mantel-Haenszel chi-square, Pearson correlation, Cochran-Armitage test for trend, weighted kappa coefficient, or Cochran-Mantel-Haenszel statistics.

Restriction: SCOROUT displays the row and column scores only when statistics are computed for two-way tables.

Tip: To store the scores in an output data set, use the Output Delivery System.

Main discussion: "Scores" on page 545

See also: SCORES= on page 537

SPARSE

lists all possible combinations of the variable values for an *n*-way table when *n*>1 even if a combination does not occur in the data. SPARSE has no effect unless you use the LIST or OUT= option. When you use SPARSE and LIST, PROC FREQ lists any combination of values with a frequency count of zero. When you use SPARSE and OUT= , PROC FREQ includes empty crosstabulation table cells in the output data set.

See also: "Missing Values" on page 585

Featured in: Example 1 on page 592

TESTF=(*values*)

specifies the null hypothesis frequencies for a one-way chi-square test for specified frequencies. You can separate *values* with blanks or commas.

Range: The sum of the frequency values must equal the total frequency for the one-way table.

Restriction: The number of TESTF= values must equal the number of variable levels in the one-way table. List these values in the order that the corresponding variable levels appear in the output.

Interaction: If you omit CHISQ, TESTF= invokes CHISQ.

Main discussion: "Chi-Square Test for One-Way Tables" on page 546

TESTP=(*values*)

specifies the null hypothesis proportions for a one-way chi-square test for specified proportions. You can separate *values* with blanks or commas.

Range: Specify *values* in probability form as numbers between 0 and 1, where the proportions sum to 1. Or, specify *values* in percentage form as numbers between 0 and 100, where the percentages sum to 100.

Restriction: The number of TESTP= values must equal the number of variable levels in the one-way table. List these values in the order that the corresponding variable levels appear in the output.

Interaction: If you omit CHISQ, TESTP= invokes CHISQ.

Main discussion: "Chi-Square Test for One-Way Tables" on page 546

Featured in: Example 2 on page 596

TOTPCT

displays the percentage of total frequency on crosstabulation tables, for *n*-way tables where *n* > 2. This percentage is also available with the LIST option or as the PERCENT variable in the OUT= output data set.

TREND

performs the Cochran-Armitage test for trend.

Restriction: The table must be $2\times c$ or $r\times 2$.

Main discussion: "Cochran-Armitage Test for Trend" on page 566

Featured in: Example 7 on page 611

V5FMT
uses a field that is 8 positions wide to display the cell frequencies between 1.E7 and 1.E8 so that PROC FREQ does not use scientific notation to display frequencies in this range. By default, PROC FREQ uses a maximum of 7 positions to display cell frequencies. In Version 5 of the SAS System, PROC FREQ used a maximum of 8 positions.

Restriction: You can not specify both V5FMT and the FORMAT= option.

Tip: You can use the FORMAT= option to specify other formats for the crosstabulation cell frequencies.

TEST Statement

Computes asymptotic tests for the specified measures of association and measures of agreement.

Requirement: TABLES statement

Main discussion: "Asymptotic Tests" on page 551

Featured in: Example 7 on page 611

TEST *statistic-keyword(s)*;

Required Arguments

statistic-keyword(s)
specifies the statistics for which to provide asymptotic tests. The available statistics are the measures of association and agreement listed in Table 23.4 on page 540. You can use an individual keyword to request a test, or you can use a group keyword (MEASURES or AGREE) to request all available tests in that group.

For each measure of association or agreement that you specify, the TEST statement provides an asymptotic test that the measure equal zero. When you request an asymptotic test, PROC FREQ gives the asymptotic standard error under the null hypothesis, the test statistic, and the *p*-values. Additionally, PROC FREQ reports the confidence limits for that measure. The ALPHA= option in the TABLES statement determines the confidence level, which by default equals .05 and provides 95 percent confidence limits. In addition to these asymptotic tests, exact tests for selected measures of association and agreement are available with the EXACT statement. See "EXACT Statement" on page 521 for more information.

Table 23.4 TEST Statement Statistic-keywords and Required TABLES Statement Options

Keyword	Asymptotic tests computed	Required TABLES statement option
AGREE	simple kappa coefficient and weighted kappa coefficient	AGREE
GAMMA	gamma	ALL, MEASURES
KAPPA	simple kappa coefficient	AGREE
KENTB	Kendall's tau-*b*	ALL, MEASURES
MEASURES	gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' D $(C \mid R)$, Somers' D $(R \mid C)$, Pearson correlation coefficient, and Spearman correlation coefficient	ALL, MEASURES
PCORR	Pearson correlation coefficient	ALL, MEASURES
SCORR	Spearman correlation coefficient	ALL, MEASURES
SMDCR	Somers' D $(C \mid R)$	ALL, MEASURES
SMDRC	Somers' D $(R \mid C)$	ALL, MEASURES
STUTC	Stuart's tau-*c*	ALL, MEASURES
WTKAP	weighted kappa coefficient	AGREE

WEIGHT Statement

Treats observations as if they appear multiple times in the input data set.

Tip: Use to input the cell counts of an existing table.

Featured in: Example 1 on page 592

WEIGHT *variable*</ *option*>;

Required Arguments

variable

specifies a numeric variable whose value represents the frequency of the observation. If you use the WEIGHT statement, PROC FREQ assumes that an observation represents *n* observations, where *n* is the value of *variable*. The value of the weight variable need not be integer. When a weight value is missing, PROC FREQ ignores the corresponding observation. When a weight value is zero, PROC FREQ ignores the corresponding observation unless you specify the ZEROS option, which includes observations with zero weights. If a WEIGHT statement does not appear, each observation has a default weight of 1. The sum of the weight variable values represents the total number of observations.

Option

ZEROS
includes observations with zero weight values.

Default: PROC FREQ ignores observations with zero weights.

Main discussion: "Using Zero Weights" on page 541 and "Tables with Zero Rows or Columns" on page 574

Using Zero Weights

If you specify the ZEROS option, frequency and crosstabulation tables display any levels corresponding to observations with zero weights. (By default PROC FREQ does not process observations with zero weights, and so does not display levels that contain only observations with zero weights.)

With the ZEROS option, PROC FREQ includes levels with zero weights in the chi-square goodness-of-fit test for one-way tables. Also, PROC FREQ includes any levels with zero weights in binomial computations for one-way tables. This enables computation of binomial estimates and tests when there are no observations with positive weight in the specified level.

For two-way tables, the ZEROS option enables computation of kappa statistics when there are levels containing no observations with positive weight. See "Tables with Zero Rows or Columns" on page 574 for more details.

Note: Even with the ZEROS option, PROC FREQ does not compute the CHISQ or MEASURES statistics for two-way tables when the table has a zero row or zero column, because most of these statistics are undefined in this case. △

Using Negative Weights

If any value of the weight variable is negative, PROC FREQ displays the frequencies (as measured by the weighted values), but does not compute and display percentages and other statistics. If you create an output data set using OUT= in the TABLES statement, PROC FREQ creates the PERCENT variable and assigns a missing value for each observation. PROC FREQ also assigns missing values to the variables that the OUTEXPECT and OUTPCT options create. You cannot create an output data set using the OUTPUT statement since statistics are not computed.

Concepts: FREQ Procedure

Inputting Frequency Counts

PROC FREQ can use either raw data or cell count data to produce frequency and crosstabulation tables. *Raw data*, also known as case-record data, report the data as one record for each subject or sample member. *Cell count data* report the data in tabular form. A table lists all possible combinations of the data values along with the frequency counts. This way of presenting data often appears in published results.

The following DATA step statements store raw data in a SAS data set:

```
data raw;
   input subject $ R C @@;
   datalines;
01 1 1  02 1 1  03 1 1  04 1 1  05 1 1
06 1 2  07 1 2  08 1 2  09 2 1  10 2 1
11 2 1  12 2 1  13 2 2  14 2 2  15 2 2
;
```

You can store the same data as cell counts using the following DATA step statements:

```
data counts;
   input R C CellCount @@;
   datalines;
1 1 5   1 2 3
2 1 4   2 2 3
;
```

The variable R contains the values for the rows and the variable C contains the values for the columns. The variable CellCount contains the cell count for each row and column combination.

Both the RAW data set and COUNTS data set produce identical frequency counts, two-way tables, and statistics. With the COUNTS data set, you must use a WEIGHT statement to specify that CellCount contains cell counts. For example, to create a two-way crosstabulation table submit the following statements:

```
proc freq data=counts;
   weight CellCount;
   tables R*C;
run;
```

Grouping with Formats

PROC FREQ groups a variable's values according to its formatted values. If you assign a format to a variable with a FORMAT statement, PROC FREQ formats the variable values before dividing observations into the levels of a frequency or crosstabulation table.

For example, suppose that a variable X has the values 1.1, 1.4, 1.7, 2.1, and 2.3. Each of these values appears as a level on a frequency table. If you decide to round each value to a single digit, include the statement

```
   format x 1.;
```

in the PROC FREQ step. Now the table lists the frequency count for formatted level 1 as two and formatted level 2 as three.

PROC FREQ treats formatted character variables in the same way. The formatted values are used to group the observations into the levels of a frequency table or crosstabulation table. PROC FREQ uses the entire value of a character format to classify an observation.

You can also use the FORMAT statement to assign formats that were created with PROC FORMAT to the variables. User-written formats determine the number of levels for a variable and provide labels for a table. If you use the same data with different formats, then you can produce frequency counts and statistics for different classifications of the variable values.

When you use PROC FORMAT to create a user-written format that combines missing and nonmissing values into one category, PROC FREQ treats the entire category of formatted values as missing. For example, a questionnaire codes answers as follows: 1 as yes, 2 as no, and 8 as no answer. The following PROC FORMAT step creates a user-written format:

```
proc format;
   value questfmt 1='Yes'
                  2='No'
                .,8='Missing';
run;
```

When you use a FORMAT statement to assign QUESTFMT. to a variable, the variable's frequency table no longer includes a frequency count for the response of 8. You must use MISSING or MISSPRINT in the TABLES statement to list the frequency for no answer. The frequency count for this level will include observations with either a value of 8 or a missing value (.).

The frequency or crosstabulation table lists the values of both character and numeric variables in ascending order based on internal (unformatted) variable values unless you change the order with the ORDER= option. To list the values in ascending order by formatted values, use ORDER=FORMATTED in the PROC FREQ statement.

For more information on the FORMAT statement, see *SAS Language Reference: Dictionary*.

Computational Resources

For each variable in a table request, PROC FREQ stores all of the levels in memory. If all variables are numeric and not formatted, this requires about 84 bytes for each variable level. When there are character variables or formatted numeric variables, the memory that is required depends on the formatted variable lengths, with longer formatted lengths requiring more memory. The number of levels for each variable is limited only by the largest integer that your operating environment can store.

For any single crosstabulation table requested, PROC FREQ builds the entire table in memory, regardless of whether the table has zero cell counts. Thus, if the numeric variables A, B, and C each have 10 levels, PROC FREQ requires 2520 bytes to store the variable levels for the table request A*B*C, as follows:

```
3 variables*10 levels/variable*84 bytes/level
```

In addition , PROC FREQ requires 8000 bytes to store the table cell frequencies

```
1000 cells * 8 bytes/cell
```

even though there may be only 10 observations.

When the variables have many levels or when there are many multiway tables, your computer may not have enough memory to construct the tables. If PROC FREQ runs out of memory while constructing tables, it stops collecting levels for the variable with the most levels and returns the memory that is used by that variable. The procedure then builds the tables that do not contain the disabled variables.

If there is not enough memory for your table request and if increasing the available memory is impractical, you can reduce the number of multiway tables or variable levels. If you are not using CMH or AGREE in the TABLES statement to compute statistics across strata, reduce the number of multiway tables by using PROC SORT to sort the data set by one or more of the variables or use the DATA step to create an index for the variables. Then remove the sorted or indexed variables from the TABLES statement and include a BY statement that uses these variables. You can also reduce memory requirements by using a FORMAT statement in the PROC FREQ step to reduce the number of levels. Additionally, reducing the formatted variable lengths reduces the amount of memory that is needed to store the variable levels. For more information on using formats, see "Grouping with Formats" on page 542.

Statistical Computations: FREQ Procedure

This section gives the formulas PROC FREQ uses to compute the following:

- □ chi-square tests and statistics (CHISQ option)
- □ measures of association (MEASURES option)
- □ binomial proportion (BINOMIAL option)
- □ risks (or binomial proportions) and risk differences for 2×2 tables (RISKDIFF option)
- □ odds ratios and relative risks for 2×2 tables (MEASURES or RELRISK option)
- □ Jonckheere-Terpstra test (JT option)
- □ Cochran-Armitage test for trend (TREND option)
- □ tests and measures of agreement (AGREE option)
- □ Cochran-Mantel-Haenszel statistics (CMH option)

Furthermore, this section describes the computation of exact p-values.

When selecting statistics to analyze your data, consider the study design (which indicates whether the row and column variables are dependent or independent), the measurement scale of the variables (nominal, ordinal, or interval), the type of association that the statistics detect, and the assumptions for valid interpretation of the statistics. For example, the Mantel-Haenszel chi-square statistic requires an ordinal scale for both variables and detects a linear association. On the other hand, the Pearson chi-square is appropriate for all variables and can detect any kind of association, but is less powerful for detecting a linear association. Select tests and measures carefully, choosing those that are appropriate for your data. For more information on when to use a statistic and how to interpret the results, refer to Agresti (1996) and Stokes et al. (1995).

Definitions and Notation

In this chapter, a two-way table represents the crosstabulation of two variables X and Y. Let the rows of the table be labeled by the values $X_i,\ i = 1, 2, \ldots, R$, and the columns by $Y_j,\ j = 1, 2, \ldots, C$. Let n_{ij} denote the cell frequency in the ith row and the jth column and define the following:

$$n_{i\cdot} = \sum_j n_{ij} \qquad \text{(row totals)}$$

$$n_{\cdot j} = \sum_i n_{ij} \qquad \text{(column totals)}$$

$$n = \sum_i \sum_j n_{ij} \qquad \text{(overall total)}$$

$$p_{ij} = n_{ij}/n \qquad \text{(cell percentages)}$$

$$p_{i\cdot} = n_{i\cdot}/n \qquad \text{(row percentages)}$$

$$p_{\cdot j} = n_{\cdot j}/n \qquad \text{(column percentages)}$$

$$R_i = \text{score for row } i$$

$$C_j = \text{score for column } j$$

$$\overline{R} = \sum_i n_{i\cdot} R_i/n \qquad \text{(average row score)}$$

$$A_{ij} = \sum_{k>i} \sum_{l>j} n_{kl} + \sum_{k<i} \sum_{l<j} n_{kl}$$

$$\overline{C} = \sum_{j} n_{\cdot j} C_j / n$$ (average column score)

$$A_{ij} = \sum_{k>i} \sum_{l>j} n_{kl} + \sum_{k<i} \sum_{l<j} n_{kl}$$

$$D_{ij} = \sum_{k>i} \sum_{l<j} n_{kl} + \sum_{k<i} \sum_{l>j} n_{kl}$$

$$P = \sum_{i} \sum_{j} n_{ij} A_{ij}$$ (twice the number of concordances)

$$Q = \sum_{i} \sum_{j} n_{ij} D_{ij}$$ (twice the number of discordances)

Scores

PROC FREQ uses row and column scores when computing the Mantel-Haenszel chi-square, Pearson correlation, Cochran-Armitage test for trend, weighted kappa coefficient, and Cochran-Mantel-Haenszel statistics. The SCORES= option in the TABLES statement specifies the score type that PROC FREQ uses. The available score types are TABLE, RANK, RIDIT, and MODRIDIT scores. The default score type is TABLE.

For numeric variables, TABLE scores are the values of the row and column levels. If the row or column variables are formatted, then the TABLE score is the internal numeric value corresponding to that level. If two or more numeric values are classified into the same formatted level, then the internal numeric value for that level is the smallest of these values. For character variables, TABLE scores are defined as the row numbers and column numbers (that is, 1 for the first row, 2 for the second row, and so on).

RANK scores, which you can use to obtain nonparametric analyses, are defined by

$$\text{Row scores :}$$
$$R1_i = \sum_{k<i} n_{k\cdot} + (n_{i\cdot} + 1)/2 \quad i = 1, 2, \ldots, R$$
$$\text{Column scores :}$$
$$C1_j = \sum_{l<j} n_{\cdot l} + (n_{\cdot j} + 1)/2 \quad j = 1, 2, \ldots, C$$

Note that RANK scores yield midranks for tied values.

RIDIT scores (Bross 1958; Mack and Skillings 1980) also yield nonparametric analyses, but they are standardized by the sample size. RIDIT scores are derived from RANK scores as

$$R2_i = R1_i / n$$
$$C2_j = C1_j / n$$

Modified ridit (MODRIDIT) scores (van Elteren 1960 and Lehmann 1975), which also yield nonparametric analyses, represent the expected values of the order statistics for the uniform distribution on (0,1). Modified ridit scores are derived from RANK scores as

$$R3_i = R1_i / (n + 1)$$
$$C3_j = C1_j / (n + 1)$$

Chi-Square Tests and Statistics

When you specify the CHISQ option in the TABLES statement, PROC FREQ performs the following chi-square tests for each two-way table: Pearson chi-square, continuity-adjusted chi-square for 2×2 tables, likelihood-ratio chi-square, Mantel-Haenszel chi-square, and Fisher's exact test for 2×2 tables. Also, PROC FREQ computes the following statistics derived from the Pearson chi-square: the phi coefficient, the contingency coefficient, and Cramer's *V*. PROC FREQ computes Fisher's exact test for general $R \times C$ tables when you specify the FISHER (or EXACT) option in the TABLES statement, or, equivalently, when you specify the FISHER option in the EXACT statement.

For one-way frequency tables, PROC FREQ performs a chi-square goodness-of-fit test when you specify the CHISQ option. See "Chi-Square Test for One-Way Tables" on page 546 for information. The other chi-square tests and statistics described in this section are defined only for two-way tables, and so are not computed for one-way frequency tables.

All the two-way test statistics described in this section test the null hypothesis of no association between the row variable and the column variable. When the sample size n is large, these test statistics are distributed approximately as chi-square when the null hypothesis is true. When the sample size is not large, exact tests may be useful. PROC FREQ computes exact tests for the following chi-square statistics when you specify the corresponding option in the EXACT statement: Pearson chi-square, likelihood-ratio chi-square, and Mantel-Haenszel chi-square. See "Exact Statistics" on page 581 for more information.

Note that the Mantel-Haenszel chi-square statistic is appropriate only when both variables lie on an ordinal scale. The other chi-square tests and statistics in this section are appropriate for either nominal or ordinal variables. The following sections give the formulas that PROC FREQ uses to compute the chi-square tests and statistics. For further information on the formulas and on the applicability of each statistic, refer to Agresti (1996), Stokes et al. (1995), and the other references cited for each statistic.

Chi-Square Test for One-Way Tables

For one-way frequency tables, the CHISQ option in the TABLES statement computes a chi-square goodness-of-fit test. Let C denote the number of classes, or levels, in the one-way table. Let f_i denote the frequency of class i (or the number of observations in class i), for $i = 1, 2, \ldots, C$. Then PROC FREQ computes the chi-square statistic as

$$Q_P = \sum_{i=1}^{C} \frac{(f_i - e_i)^2}{e_i}$$

where e_i is the expected frequency for class i under the null hypothesis.

In the test for equal proportions, which is the default for the CHISQ option, the null hypothesis specifies equal proportions of the total sample size for each class. Under this null hypothesis, the expected frequency for each class equals the total sample size divided by the number of classes,

$$e_i = n/C \qquad \text{for} \;\; i = 1, 2, \ldots, C$$

In the test for specified frequencies, which PROC FREQ computes when you input null hypothesis frequencies using the TESTF= option, the expected frequencies are those TESTF= values. In the test for specified proportions, which PROC FREQ computes when you input null hypothesis proportions using the TESTP= option, the expected frequencies are determined from the TESTP= proportions p_i, as

$$e_i = p_i \cdot n \qquad \text{for} \;\; i = 1, 2, \ldots, C$$

Under the null hypothesis (of equal proportions, specified frequencies, or specified proportions), this test statistic has an asymptotic chi-square distribution, with $C - 1$ degrees of freedom. In addition to the asymptotic test, PROC FREQ computes the exact one-way chi-square test when you specify the CHISQ option in the EXACT statement.

Chi-Square Test for Two-Way Tables

The Pearson chi-square statistic for two-way tables involves the differences between the observed and expected frequencies, where the expected frequencies are computed under the null hypothesis of independence. The chi-square statistic is computed as

$$Q_P = \sum_i \sum_j \frac{(n_{ij} - e_{ij})^2}{e_{ij}}$$

where

$$e_{ij} = \frac{n_{i\cdot} n_{\cdot j}}{n}$$

When the row and column variables are independent, Q_P has an asymptotic chi-square distribution with $(R{-}1)(C - 1)$ degrees of freedom. For large values of Q_P, this test rejects the null hypothesis in favor of the alternative hypothesis of general association. In addition to the asymptotic test, PROC FREQ computes the exact chi-square test when you specify the PCHI option or CHISQ option in the EXACT statement.

For a 2×2 table, the Pearson chi-square is also appropriate for testing the equality of two binomial proportions or, for $R \times 2$ and $2 \times C$ tables, the homogeneity of proportions. Refer to Fienberg (1980).

Likelihood-Ratio Chi-Square Test

The likelihood-ratio chi-square statistic involves the ratios between the observed and expected frequencies. The statistic is computed as

$$G^2 = 2 \sum_i \sum_j n_{ij} \ln \left(\frac{n_{ij}}{e_{ij}} \right)$$

When the row and column variables are independent, G^2 has an asymptotic chi-square distribution with $(R-1)(C-1)$ degrees of freedom. In addition to the asymptotic test, PROC FREQ computes the exact test when you specify the LRCHI option or the CHISQ option in the EXACT statement.

Continuity-Adjusted Chi-Square Test

The continuity-adjusted chi-square statistic for 2×2 tables is similar to the Pearson chi-square, except that it is adjusted for the continuity of the chi-square distribution. The continuity-adjusted chi-square is most useful for small sample sizes. The use of the continuity adjustment is controversial; this chi-square test is more conservative, and more like Fisher's exact test, when your sample size is small. As the sample size increases, the statistic becomes more and more like the Pearson chi-square. The statistic is computed as

$$Q_C = \sum_i \sum_j \frac{\left[\max\left(0, |n_{ij} - e_{ij}| - 0.5\right)\right]^2}{e_{ij}}$$

Under the null hypothesis of independence, Q_C has an asymptotic chi-square distribution with $(R-1)(C-1)$ degrees of freedom.

Mantel-Haenszel Chi-Square Test

The Mantel-Haenszel chi-square statistic tests the alternative hypothesis that there is a linear association between the row variable and the column variable. Both variables must lie on an ordinal scale. The statistic is computed as

$$Q_{MH} = (n-1)\, r^2$$

where r^2 is the Pearson correlation between the row variable and the column variable. For a description of the Pearson correlation, see "Pearson Correlation Coefficient" on page 555. The Pearson correlation, and thus the Mantel-Haenszel chi-square statistic, use the scores you specify in the SCORES= option in the TABLES statement.

Under the null hypothesis of no association, Q_{MH} has an asymptotic chi-square distribution with 1 degree of freedom. In addition to the asymptotic test, PROC FREQ computes the exact test when you specify the MHCHI option or the CHISQ option in the EXACT statement.

Refer to Mantel and Haenszel (1959) and Landis et al. (1978).

Fisher's Exact Test

Fisher's exact test is another test of association between the row and column variables. This test assumes that the row and column totals are fixed, and then uses the hypergeometric distribution to compute probabilities of possible tables with those row and column totals. Fisher's exact test does not depend on any large-sample distribution assumptions, and so it is appropriate even for small sample sizes and for sparse tables.

PROC FREQ gives the following information for Fisher's exact test for 2×2 tables: table probability, two-sided p-value, left-sided p-value, and right-sided p-value. Where p is the hypergeometric probability of a specific table with the observed row and column totals, p-values are computed by summing these probabilities p over defined sets of tables,

$$PROB = \sum_{A} p$$

The table probability is the hypergeometric probability of the observed table. The two-sided p-value is the sum of all possible table probabilities (for tables having the observed row and column totals) that are less than or equal to the observed table probability. So, for the two-sided p-value, the set A includes all possible tables with hypergeometric probabilities less than or equal to the probability of the observed table. A small two-sided p-value supports the alternative hypothesis of association between the row and column variables.

One-sided tests are defined in terms of the frequency of the cell in the first row and first column of the table, the (1,1) cell. Denoting the observed (1,1) cell frequency by F, the left-sided p-value for Fisher's exact test is the probability that the (1,1) cell frequency is less than or equal to F. So, for the left-sided p-value, the set A includes those tables with a (1,1) cell frequency less than or equal to F. A small left-sided p-value supports the alternative hypothesis that the probability of an observation being in the first cell is less than expected under the null hypothesis of independent row and column variables.

Similarly, for a right-sided alternative hypothesis, A is the set of tables where the frequency of the (1,1) cell is greater than or equal to that in the observed table. A small right-sided p-value supports the alternative that the probability of the first cell is greater than that expected under the null hypothesis.

Because the (1,1) cell frequency completely determines the 2×2 table when the marginal row and column sums are fixed, these one-sided alternatives can be equivalently stated in terms of other cell probabilities or ratios of cell probabilities. The left-sided alternative is equivalent to an odds ratio greater than 1, where the odds ratio equals $(n_{11}n_{22}/n_{12}n_{21})$. Additionally, the left-sided alternative is equivalent to the column 1 risk for row 1 being less than the column 1 risk for row 2, $p_{1|1} < p_{1|2}$. Similarly, the right-sided alternative is equivalent to the column 1 risk for row 1 being greater than the column 1 risk for row 2, $p_{1|1} > p_{1|2}$. Refer to Agresti (1996).

Fisher's exact test was extended to general $R \times C$ tables by Freeman and Halton (1951), and this test is also known as the Freeman-Halton test. For $R \times C$ tables, the two-sided p-value is defined the same as it is for 2×2 tables. A is the set of all tables with p less than or equal to the probability of the observed table. A small p-value supports the alternative hypothesis of association between the row and column variables. For $R \times C$ tables, Fisher's exact test is inherently two-sided. The alternative hypothesis is defined only in terms of general, and not linear, association. Therefore, PROC FREQ does not compute right-sided or left-sided p-values for general $R \times C$ tables.

For $R \times C$ tables, PROC FREQ computes Fisher's exact test using the network algorithm of Mehta and Patel (1983), which provides a faster and more efficient solution than direct enumeration. See "Exact Statistics" on page 581 for more information.

Phi Coefficient

The phi coefficient is a measure of association derived from the Pearson chi-square statistic. It has the range $-1 \leq \phi \leq 1$ for 2×2 tables. Otherwise, the range is $0 \leq \phi \leq \min\left(\sqrt{R-1}, \sqrt{C-1}\right)$ (Liebetrau, 1983). The phi coefficient is computed as

$$\phi = \frac{n_{11}n_{22} - n_{12}n_{21}}{\sqrt{n_{1\cdot}n_{2\cdot}n_{\cdot1}n_{\cdot2}}} \quad \text{for } 2\times2 \text{ tables}$$

$$\phi = \sqrt{Q_P/n} \qquad \text{otherwise.}$$

Refer to Fleiss (1981, pp 59-60).

Contingency Coefficient

The contingency coefficient is a measure of association derived from the Pearson chi-square. It has the range $0 \leq P \leq \sqrt{(m-1)/m}$, where $m = \min(R, C)$ (Liebetrau, 1983). The contingency coefficient is computed as

$$P = \sqrt{\frac{Q_P}{Q_P + n}}$$

Refer to Kendall and Stuart (1979, pp 587-588).

Cramer's *V*

Cramer's *V* is a measure of association derived from the Pearson chi-square. It is designed so that the attainable upper limit is always 1. It has the range $-1 \leq V \leq 1$ for 2×2 tables; otherwise, the range is $0 \leq V \leq 1$. Cramer's *V* is computed as

$$V = \phi \qquad \text{for } 2\times2 \text{ tables}$$

$$V = \sqrt{\frac{Q_P/n}{\min(R-1, C-1)}} \qquad \text{otherwise.}$$

Refer to Kendall and Stuart (1979, p. 588).

Measures of Association

When you specify the MEASURES option in the TABLES statement, PROC FREQ computes several statistics that describe the association between the two variables of the contingency table. The following are measures of ordinal association that consider whether the variable Y tends to increase as X increases: gamma, Kendall's tau-*b*, Stuart's tau-*c*, and Somers' *D*. These measures are appropriate for ordinal variables, and classify pairs of observations as *concordant* or *discordant*. A pair is *concordant* if the observation with the larger value of X also has the larger value of Y. A pair is *discordant* if the observation with the larger value of X has the smaller value of Y. Refer to Agresti (1996) and the other references cited in the discussion of each measure of association.

The Pearson correlation coefficient and the Spearman rank correlation coefficient are also appropriate for ordinal variables. The Pearson correlation describes the strength of the linear association between the row and column variables, and is computed using the row and column scores specified by the SCORES= option in the TABLES statement. The Spearman correlation is computed with rank scores. The polychoric correlation (requested by the PLCORR option) also requires ordinal variables, and assumes that the variables have an underlying bivariate normal distribution. The following measures of association do not require ordinal variables, but are appropriate for nominal variables: lambda asymmetric and symmetric, and the uncertainty coefficients.

PROC FREQ computes estimates of the measures according to the formulas given in the discussion of each measure of association. For each measure, PROC FREQ computes an asymptotic standard error, which is the square root of the asymptotic variance denoted by *var* in the following sections.

Confidence Limits

If you specify the CL option in the TABLES statement, PROC FREQ computes asymptotic confidence limits for all MEASURES statistics. The confidence coefficient is determined according to the value of the ALPHA= option, which by default equals 0.05 and produces 95 percent confidence limits. The confidence limits are computed as

$$est \ \pm \ z_{\alpha/2} \cdot ASE$$

where est is the estimate of the measure, $z_{\alpha/2}$ is the $100\,(1-\alpha/2)$ percentile of the standard normal distribution, and *ASE* is the asymptotic standard error of the estimate.

Asymptotic Tests

For each measure that you specify in the TEST statement, PROC FREQ computes an asymptotic test of the null hypothesis that the measure equals zero. Asymptotic tests are available for the following measures of association: gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(R|C)$, Somers' $D(C|R)$, the Pearson correlation coefficient, and the Spearman rank correlation coefficient. To compute an asymptotic test, PROC FREQ uses a standardized test statistic *z*, which has an asymptotic standard normal distribution under the null hypothesis. The standardized test statistic is computed as

$$z = \frac{est}{\sqrt{var_0\,(est)}}$$

where est is the estimate of the measure, and $var_0\,(est)$ is the variance of the estimate under the null hypothesis. Formulas for $var_0\,(est)$ are given in the discussion of each measure of association.

Note that the ratio of est to $\sqrt{var_0\,(est)}$ is the same for the following measures: gamma, Kendall's tau-*b*, Stuart's tau-*c*, Somers' $D(R|C)$, and Somers' $D(C|R)$. Therefore, the tests for these measures are identical. For example, the *p*-values for the test of H_0: gamma=0 equal the *p*-values for the test of H_0: tau-*b*= 0.

PROC FREQ computes one-sided and two-sided *p*-values for each of these tests. When the test statistic *z* is greater than its null hypothesis expected value of zero, PROC FREQ computes the right-sided *p*-value, which is the probability of a larger value of the statistic occurring under the null hypothesis. A small right-sided *p*-value supports the alternative hypothesis that the true value of the measure is greater than zero. When the test statistic is less than or equal to zero, PROC FREQ computes the left-sided *p*-value, which is the probability of a smaller value of the statistic occurring under the null hypothesis. A small left-sided *p*-value supports the alternative hypothesis that the true value of the measure is less than zero. The one-sided *p*-value P_1 can be expressed as

$$P_1 = \text{Prob}\,(Z > z) \quad \text{if} \quad z > 0$$
$$P_1 = \text{Prob}\,(Z < z) \quad \text{if} \quad z \le 0$$

where Z has a standard normal distribution. The two-sided p-value P_2 is computed as

$$P_2 = \text{Prob}(|Z| > |z|)$$

Exact Tests

Exact tests are available for two measures of association, the Pearson correlation coefficient and the Spearman rank correlation coefficient. If you specify the PCORR option in the EXACT statement, PROC FREQ computes the exact test of the hypothesis that the Pearson correlation equals zero. If you specify the SCORR option in the EXACT statement, PROC FREQ computes the exact test of the hypothesis that the Spearman correlation equals zero. See "Exact Statistics" on page 581 for information on exact tests.

Gamma

The estimator of gamma is based only on the number of concordant and discordant pairs of observations. It ignores tied pairs (that is, pairs of observations that have equal values of X or equal values of Y). Gamma is appropriate only when both variables lie on an ordinal scale. It has the range $-1 \le \Gamma \le 1$. If the two variables are independent, then the estimator of gamma tends to be close to zero. Gamma is estimated by

$$G = \frac{(P - Q)}{(P + Q)}$$

with

$$var = \frac{16}{(P + Q)^4} \sum_i \sum_j n_{ij} \left(QA_{ij} - PD_{ij} \right)^2$$

The variance of the estimator under the null hypothesis that gamma equals zero is computed as

$$var_0\,(G) = \frac{4}{(P+Q)^2} \left(\sum_i \sum_j n_{ij} \left(A_{ij} - D_{ij} \right)^2 - (P-Q)^2/n \right)$$

For 2×2 tables, gamma is equivalent to Yule's Q. Refer to Goodman and Kruskal (1963; 1972), Brown and Benedetti (1977), and Agresti (1990).

Kendall's Tau-*b*

Kendall's tau-*b* is similar to gamma except that tau-*b* uses a correction for ties. Tau-*b* is appropriate only when both variables lie on an ordinal scale. Tau-*b* has the range $-1 \le \tau_b \le 1$. It is estimated by

$$t_b = \frac{(P - Q)}{\sqrt{w_r w_c}}$$

with

$$var = \frac{1}{w^4} \left(\sum_i \sum_j n_{ij} \left(2wd_{ij} + t_b v_{ij} \right)^2 - n^3 t_b^2 \left(w_r + w_c \right)^2 \right)$$

where

$$\begin{aligned} w &= \sqrt{w_r w_c} \\ w_r &= n^2 - \sum_i n_{i\cdot}^2 \\ w_c &= n^2 - \sum_j n_{\cdot j}^2 \\ d_{ij} &= A_{ij} - D_{ij} \\ v_{ij} &= n_{i\cdot} w_c + n_{\cdot j} w_r \end{aligned}$$

The variance of the estimator under the null hypothesis that tau-b equals zero is computed as

$$var_0 \left(t_b \right) = \frac{4}{w_r w_c} \left(\sum_i \sum_j n_{ij} \left(A_{ij} - D_{ij} \right)^2 - \left(P - Q \right)^2 / n \right)$$

Refer to Kendall (1955) and Brown and Benedetti (1977).

Stuart's Tau-*c*

Stuart's tau-c makes an adjustment for table size in addition to a correction for ties. Tau-c is appropriate only when both variables lie on an ordinal scale. Tau-c has the range $-1 \le \tau_c \le 1$. It is estimated by

$$t_c = \frac{m \left(P - Q \right)}{n^2 \left(m - 1 \right)}$$

with

$$var = \frac{4m^2}{\left(m - 1 \right)^2 n^4} \left(\sum_i \sum_j n_{ij} d_{ij}^2 - \left(P - Q \right)^2 / n \right)$$

where

$$m = \min(R, C)$$
$$d_{ij} = A_{ij} - D_{ij}$$

The variance of the estimator under the null hypothesis that tau-*c* equals zero is the same as var in the above equation.

$$var_0(t_c) = var$$

Refer to Brown and Benedetti (1977).

Somers' *D*

Somers' $D(C|R)$ and Somers' $D(R|C)$ are asymmetric modifications of tau-*b*. $C|R$ denotes that the row variable X is regarded as an independent variable, while the column variable Y is regarded as dependent. Similarly, $R|C$ denotes that the column variable Y is regarded as an independent variable, while the row variable X is regarded as dependent. Somers' D differs from tau-*b* in that it uses a correction only for pairs that are tied on the independent variable. Somers' D is appropriate only when both variables lie on an ordinal scale. It has the range $-1 \leq D \leq 1$. Formulas for Somers' $D(R|C)$ are obtained by interchanging the indices:

$$D(C|R) = \frac{(P-Q)}{w_r}$$

with

$$var = \frac{4}{\mathrm{w}_r^4} \sum_i \sum_{\mathrm{j}} n_{ij} \left(w_r d_{ij} - (P-Q)(n - n_{i\cdot}) \right)^2$$

where

$$w_r = n^2 - \sum_i n_{i\cdot}^2$$
$$d_{ij} = A_{ij} - D_{ij}$$

The variance of the estimator under the null hypothesis that tau-*c* equals zero is computed as

$$var_0(D(C|R)) = \frac{4}{w_r^2} \left(\sum_i \sum_j n_{ij} (A_{ij} - D_{ij})^2 - (P-Q)^2 / n \right)$$

Refer to Somers (1962) and Goodman and Kruskal (1972).

Pearson Correlation Coefficient

PROC FREQ computes the Pearson correlation coefficient using the scores specified in the SCORES= option. The Pearson correlation is appropriate only when both variables lie on an ordinal scale. It has the range $-1 \leq \rho \leq 1$. The Pearson correlation coefficient is computed as

$$r = \frac{v}{w} = \frac{ss_{rc}}{\sqrt{ss_r ss_c}}$$

with

$$var = \frac{1}{w^4} \sum_i \sum_j n_{ij} \left(w \left(R_i - \overline{R}\right) \left(C_j - \overline{C}\right) - \frac{b_{ij} v}{2w} \right)^2$$

The row scores R_i and the column scores C_j are determined by the SCORES= option in the TABLES statement. Then

$$ss_r = \sum_i \sum_j n_{ij} \left(R_i - \overline{R}\right)^2$$

$$ss_c = \sum_i \sum_j n_{ij} \left(C_j - \overline{C}\right)^2$$

$$ss_{rc} = \sum_i \sum_j n_{ij} \left(R_i - \overline{R}\right) \left(C_j - \overline{C}\right)$$

$$b_{ij} = \left(R_i - \overline{R}\right)^2 ss_c + \left(C_j - \overline{C}\right)^2 ss_r$$

$$v = ss_{rc}$$

$$w = \sqrt{ss_r ss_c}$$

where $\overline{R}$ and $\overline{C}$ are the average row and columns scores as defined in "Definitions and Notation" on page 544. Refer to Snedecor and Cochran (1989) and Brown and Benedetti (1977).

To compute an asymptotic test for the Pearson correlation, PROC FREQ uses a standardized test statistic r^*, which has an asymptotic standard normal distribution under the null hypothesis. The standardized test statistic is computed as

$$r^* = \frac{r}{\sqrt{var_0\left(r\right)}}$$

where $var_0\left(r\right)$ is the variance of the correlation under the null hypothesis.

$$var_0\left(r\right) = \frac{\sum_i \sum_j n_{ij} \left(R_i - \overline{R}\right)^2 \left(C_j - \overline{C}\right)^2 - ss_{rc}^2 / n}{ss_r ss_c}$$

This asymptotic variance is derived for multinomial sampling in a contingency table framework, and it differs from the form obtained under the assumption that both variables are continuous and normally distributed. Refer to Brown and Benedetti (1977).

PROC FREQ also computes the exact test for the hypothesis that the Pearson correlation equals zero when you specify the PCORR option in the EXACT statement. See “Exact Statistics” on page 581 for more information on exact tests.

Spearman Rank Correlation Coefficient

The Spearman correlation coefficient is computed using rank scores $R1_i$ and $C1_j$, defined in “Scores” on page 545. It is appropriate only when both variables lie on an ordinal scale. It has the range $-1 \leq \rho_s \leq 1$. The Spearman correlation coefficient is computed as

$$r_s = \frac{v}{w}$$

with

$$var = \frac{1}{n^2 w^4} \sum_i \sum_j n_{ij} \left(z_{ij} - \overline{z} \right)^2$$

where

$$v = \sum_i \sum_j n_{ij} R(i) C(j)$$

$$w = \frac{1}{12}\sqrt{FG}$$

$$F = n^3 - \sum_i n_{i\cdot}^3$$
$$G = n^3 - \sum_j n_{\cdot j}^3$$
$$R(i) = R1_i - \frac{n}{2}$$
$$C(j) = C1_j - \frac{n}{2}$$

$$\overline{z} = \frac{1}{n}\sum_i \sum_j n_{ij} z_{ij}$$
$$z_{ij} = wv_{ij} - vw_{ij}$$
$$v_{ij} = n(R(i)C(j) + \frac{1}{2}\sum_l n_{il}C(l) + \frac{1}{2}\sum_k n_{kj}R(k) + \sum_l \sum_{k>i} n_{kl}C(l) + \sum_k \sum_{l>j} n_{kl}R(k))$$

$$w_{ij} = \frac{-n}{96w}\left(Fn_{\cdot j}^2 + Gn_{i\cdot}^2\right)$$

Refer to Snedecor and Cochran (1989) and Brown and Benedetti (1977).

To compute an asymptotic test for the Spearman correlation, PROC FREQ uses a standardized test statistic r_s^*, which has an asymptotic standard normal distribution under the null hypothesis. The standardized test statistic is computed as

$$r_s^* = \frac{r_s}{\sqrt{var_0(r_s)}}$$

where $var_0(r_s)$ is the variance of the correlation under the null hypothesis.

$$var_0(r_s) = \frac{1}{n^2w^2}\sum_i \sum_j n_{ij}\left(v_{ij} - \overline{v}\right)^2$$

where

$$\overline{v} = \sum_i \sum_j n_{ij} v_{ij} / n$$

This asymptotic variance is derived for multinomial sampling in a contingency table framework, and it differs from the form obtained under the assumption that both variables are continuous. Refer to Brown and Benedetti (1977).

PROC FREQ also computes the exact test for the hypothesis that the Spearman rank correlation equals zero when you specify the SCORR option in the EXACT statement. See “Exact Statistics” on page 581 for more information.

Polychoric Correlation

When you specify the PLCORR option in the TABLES statement, PROC FREQ computes the polychoric correlation. This measure of association is based on the assumption that the ordered, categorical variables of the frequency table have an underlying bivariate normal distribution. For 2×2 tables, the polychoric correlation is also known as the tetrachoric correlation. Refer to Drasgow (1986) for an overview of polychoric correlation. The polychoric correlation coefficient is the maximum likelihood estimate of the product-moment correlation between the normal variables, estimating thresholds from the observed table frequencies. Olsson (1979) gives the likelihood equations and an asymptotic covariance matrix for the estimates.

To estimate the polychoric correlation, PROC FREQ iteratively solves the likelihood equations by a Newton-Raphson algorithm. Iteration stops when the convergence measure falls below the convergence criterion, or when the maximum number of iterations is reached, whichever occurs first. The CONVERGE= option sets the convergence criterion, and the default is 0.0001. The MAXITER= option sets the maximum number of iterations, and the default is 20.

Lambda Asymmetric

Asymmetric lambda, $\lambda(C|R)$, is interpreted as the probable improvement in predicting the column variable Y given knowledge of the row variable X. Asymmetric lambda has the range $0 \leq \lambda(C|R) \leq 1$. It is computed as

$$\lambda(C|R) = \frac{\sum_i r_i - r}{n - r}$$

with

$$var = \frac{(n - \sum_i r_i)}{(n-r)^3} \left(\sum_i r_i + r - 2 \sum_i (r_i | l_i = l) \right)$$

where

$$r_i = \max_j (n_{ij})$$
$$r = \max_j (n_{\cdot j})$$

Also, let l_i be the unique value of j such that $r_i = n_{ij}$, and let l be the unique value of j such that $r_i = n_{\cdot j}$.

Because of the uniqueness assumptions, ties in the frequencies or inthe marginal totals must be broken in an arbitrary but consistent manner. In case of ties, l is defined here as the smallest value of j such that $r = n_{\cdot j}$. For a given i, if there is at least one value j such that $n_{ij} = r_i = c_j$ then l_i is defined here to be the smallest such value of j. Otherwise, if $n_{il} = r_i$, then l_i is defined to be equal to l. If neither condition is true, then l_i is taken to be the smallest value of j such that $n_{ij} = r_i$. The formulas for lambda asymmetric $R|C$ can be obtained by interchanging the indices.

Refer to Goodman and Kruskal (1963).

Lambda Symmetric

The nondirectional lambda is the average of the two asymmetric lambdas. Lambda symmetric has the range $0 \leq \lambda \leq 1$. Lambda symmetric is defined as

$$\lambda = \frac{\left(\sum_i r_i + \sum_j c_j - r - c \right)}{(2n - r - c)} = \frac{(w - v)}{w}$$

with

$$var = \frac{1}{w^4} \left(wvy - 2w^2 \left[n - \sum_i \sum_j (n_{ij} | j = l_i, i = k_j) \right] - 2v^2 (n - n_{kl}) \right)$$

where

$$\begin{aligned}
c_j &= \max_i \; (n_{ij}) \\
c &= \max_i \; (n_{i\cdot}) \\
w &= 2n - r - c \\
v &= 2n - \sum_i r_i - \sum_j c_j \\
x &= \sum_i (r_i | l_i = l) + \sum_j (c_j | k_j = k) + r_k + c_l \\
y &= 8n - w - v - 2x
\end{aligned}$$

Refer to Goodman and Kruskal (1963).

Uncertainty Coefficient Asymmetric

The uncertainty coefficient, $U(C|R)$, is the proportion of uncertainty (entropy) in the column variable Y that is explained by the row variable X. It has the range $0 \leq U(C|R) \leq 1$. The formulas for $U(R|C)$ are obtained by interchanging the indices.

$$U(C|R) = \frac{H(X) + H(Y) - H(XY)}{H(Y)} = \frac{v}{w}$$

with

$$var = \frac{1}{n^2 w^4} \sum_i \sum_j n_{ij} \left(H(Y) \ln \left(\frac{n_{ij}}{n_{i\cdot}} \right) + [H(X) - H(XY)] \ln \left(\frac{n_{\cdot j}}{n} \right) \right)^2$$

where

$$\begin{aligned} v &= H(X) + H(Y) - H(XY) \\ w &= H(Y) \\ H(X) &= -\sum_i \left(\frac{n_{i\cdot}}{n} \right) \ln \left(\frac{n_{i\cdot}}{n} \right) \\ H(Y) &= -\sum_j \left(\frac{n_{\cdot j}}{n} \right) \ln \left(\frac{n_{\cdot j}}{n} \right) \\ H(XY) &= -\sum_i \sum_j \left(\frac{n_{ij}}{n} \right) \ln \left(\frac{n_{ij}}{n} \right) \end{aligned}$$

Refer to Theil (1972, pp 115-120) and Goodman and Kruskal (1972).

Uncertainty Coefficient Symmetric

The uncertainty coefficient, U, is the symmetric version of the two asymmetric coefficients. It has the range $0 \le U \le 1$. It is defined as

$$U = \frac{2(H(X) + H(Y) - H(XY))}{H(X) + H(Y)}$$

with

$$var = 4 \sum_i \sum_j \frac{n_{ij} \left(H(XY) \ln \left(\frac{n_{i\cdot} n_{\cdot j}}{n^2} \right) - [H(X) + H(Y)] \ln \left(\frac{n_{ij}}{n} \right) \right)^2}{n^2 (H(X) + H(Y))^4}$$

Refer to Goodman and Kruskal (1972).

Binomial Proportion

When you specify the BINOMIAL option in the TABLES statement, PROC FREQ computes a binomial proportion for one-way tables. By default, this is the proportion of observations for the first variable level, or class, that appears in the output. (To specify a different level, use the LEVEL= option.)

$$\hat{p} = n_1 / n$$

where n_1 is the frequency for the specified level, and n is the total frequency for the one-way table. The standard error for the binomial proportion is computed as

$$se(\hat{p}) = \sqrt{\hat{p}(1-\hat{p})/n}$$

Using the normal approximation to the binomial distribution, PROC FREQ constructs asymptotic confidence limits for p according to

$$\hat{p} \pm z_{\alpha/2} \cdot se(\hat{p})$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$ percentile of the standard normal distribution. The confidence level α is determined by the ALPHA= option, which by default equals .05 and produces 95 percent confidence limits.

If you specify the BINOMIALC option, PROC FREQ includes a continuity correction of $1/2n$ in the asymptotic confidence limits for p. The purpose of this correction is to adjust for the difference between the normal approximation and the binomial distribution, which is a discrete distribution. Refer to Fleiss (1981). With the continuity correction, the asymptotic confidence limits for p are

$$\hat{p} \pm \left(z_{\alpha/2} \cdot se(\hat{p}) + (1/2n)\right)$$

Additionally, PROC FREQ computes exact confidence limits for the binomial proportion using the *F* distribution method given in Collett (1991) and also described by Leemis and Trivedi (1996).

PROC FREQ computes an asymptotic test of the hypothesis that the binomial proportion equals p_0, where the value of p_0 is specified by the P= option in the TABLES statement. If you do not specify a value for P=, PROC FREQ uses $p_0 = 0.5$ by default. The asymptotic test statistic is

$$z = \frac{\hat{p} - p_0}{\sqrt{p_0(1-p_0)/n}}$$

If you specify the BINOMIALC option, PROC FREQ includes a continuity correction in the asymptotic test statistic, towards adjusting for the difference between the normal approximation and the discrete binomial distribution. Refer to Fleiss (1981). The continuity correction of $1/(2n)$ is subtracted from $(\hat{p} - p_0)$ in the numerator of the test statistic if $(\hat{p} - p_0)$ is positive; otherwise, the continuity correction is added to the numerator.

PROC FREQ computes one-sided and two-sided *p*-values for this test. When the test statistic *z* is greater than its null hypothesis expected value of zero, PROC FREQ computes the right-sided *p*-value, which is the probability of a larger value of the statistic occurring under the null hypothesis. A small right-sided *p*-value supports the alternative hypothesis that the true value of the proportion is greater than p_0. When the test statistic is less than or equal to zero, PROC FREQ computes the left-sided *p*-value, which is the probability of a smaller value of the statistic occurring under the null hypothesis. A small left-sided *p*-value supports the alternative hypothesis that the true value of the proportion is less than p_0. The one-sided *p*-value P_1 can be expressed as

$$P_1 = \text{Prob}\ (Z > z) \quad \text{if}\ z > 0$$
$$P_1 = \text{Prob}\ (Z < z) \quad \text{if}\ z \le 0$$

where Z has a standard normal distribution. The two-sided p-value P_2 is computed as

$$P_2 = \text{Prob}\ \ (|Z| > |z|)$$

When you specify the BINOMIAL option in the EXACT statement, PROC FREQ also computes an exact test of the null hypothesis $H_0 :\ p = p_0$. To compute this exact test, PROC FREQ uses the binomial probability function

$$\text{Prob}\,(X{=}x|p_0){=}\binom{n}{x}p_0^x\,(1-p_0)^{(n-x)} \quad x{=}0{,}1{,}2{,}\ldots{,}n$$

where the variable X has a binomial distribution with parameters n and p_0. To compute $\text{Prob}\,(X \le n_1)$, PROC FREQ sums these binomial probabilities over x from zero to n_1. To compute $\text{Prob}\,(X \ge n_1)$, PROC FREQ sums these binomial probabilities over x from n_1 to n. Then the exact one-sided p-value is

$$P_1 = \min\left(\text{Prob}\,(X \le n_1|p_0)\,,\text{Prob}\,(X \ge n_1|p_0)\right)$$

and the exact two-sided p-value is

$$P_2 = 2 \cdot P_1$$

Risks and Risk Differences

The RISKDIFF option in the TABLES statement provides estimates of risks (or binomial proportions) and risk differences for 2×2 tables. This analysis may be appropriate when you are comparing the proportion of some characteristic for two groups, where row 1 and row 2 correspond to the two groups, and the columns correspond to two possible characteristics or outcomes. For example, the row variable might be a treatment or dose, and the column variable might be the response. Refer to Collett (1991), Fleiss (1981), and Stokes et al. (1995).

Let the frequencies of the 2×2 table be represented as follows:

	Column 1	Column 2	Total
Row 1	n_{11}	n_{12}	$n_{1\bullet}$
Row 2	n_{21}	n_{22}	$n_{2\bullet}$
Total	$n_{\bullet 1}$	$n_{\bullet 2}$	n

The column 1 risk for row 1 is the proportion of row 1 observations classified in column 1

$$p_{1|1} = n_{11} / n_{1\cdot}$$

This estimates the conditional probability of the column 1 response, given the first level of the row variable.

The column 1 risk for row 2 is the proportion of row 2 observations classified in column 1,

$$p_{1|2} = n_{21} / n_{2\cdot}$$

and the overall column 1 risk is the proportion of all observations classified in column 1,

$$p_{\cdot 1} = n_{\cdot 1} / n$$

The column 1 risk difference compares the risks for the two rows, and it is computed as the column 1 risk for row 1 minus the column 1 risk for row 2,

$$(pdiff)_1 = p_{1|1} - p_{1|2}$$

The risks and risk difference are defined similarly for column 2.

The standard error of the column 1 risk estimate for row i is computed as

$$se\left(p_{1|i}\right) = \sqrt{p_{1|i}\left(1 - p_{1|i}\right) / n_{i\cdot}}$$

The standard error of the overall column 1 risk estimate is computed as

$$se\left(p_{\cdot 1}\right) = \sqrt{p_{\cdot 1}\left(1 - p_{\cdot 1}\right) / n}$$

If the two rows represent independent binomial samples, the standard error for the column 1 risk difference is computed as

$$se\left((pdiff)_1\right) = \sqrt{var\left(p_{1|1}\right) + var\left(p_{1|2}\right)}$$

The standard errors are computed similarly for the column 2 risks and risk difference.

Using the normal approximation to the binomial distribution, PROC FREQ constructs asymptotic confidence limits for the risks and risk differences according to

$$est \ \pm \ \left(z_{\alpha/2} \cdot \ se\left(est\right)\right)$$

where est is the estimate, $z_{\alpha/2}$ is the $(1 - \alpha/2)$ percentile of the standard normal distribution, and $se\left(est\right)$ is the standard error of the estimate. The confidence level α

is determined from the value of the ALPHA= option, which, by default, equals 0.05 and produces 95 percent confidence limits.

If you specify the RISKDIFFC option, PROC FREQ includes continuity corrections in the asymptotic confidence limits for the risks and risk differences. Continuity corrections adjust for the difference between the normal approximation and the discrete binomial distribution. Refer to Fleiss (1981). Including a continuity correction, the asymptotic confidence limits become

$$est \; \pm \; \left(z_{\alpha/2} \cdot \; se\,(est)\right) + cc$$

where cc is the continuity correction. For the column 1 risk for row 1, $cc = (1/2n_{1\cdot})$; for the column 1 risk for row 2, $cc = (1/2n_{2\cdot})$; for the overall column 1 risk, $cc = (1/2n)$; and for the column 1 risk difference, $cc = ((1/n_{1\cdot} + 1/n_{2\cdot})/2)$. Continuity corrections are computed similarly for thecolumn 2 risks and risk difference.

PROC FREQ computes exact confidence limits for the column 1, column 2, and overall risks using the *F* distribution method given in Collett (1991), and also described by Leemis and Trivedi (1996). PROC FREQ does not provide exact confidence limits for the risk differences. Refer to Agresti (1992) for a discussion of issues involved in constructing exact confidence limits for differences of proportions.

Odds Ratio and Relative Risks for 2×2 Tables

Odds Ratio (Case-Control Studies)

The odds ratio is a useful measure of association for a variety of study designs. For a retrospective design called a *case-control study*, the odds ratio can be used to estimate the relative risk when the probability of positive response is small (Agresti, 1990). In a case-control study, two independent samples are identified based on a binary (yes-no) response variable, and the conditional distribution of a binary explanatory variable is examined within fixed levels of the response variable. Refer to Stokes et al. (1995) and Agresti (1996).

The odds of a positive response (column 1) in row 1 is n_{11}/n_{12}. Similarly, the odds of positive response in row 2 is n_{21}/n_{22}. The odds ratio is formed as the ratio of the row 1 odds to the row 2 odds. The odds ratio for 2×2 tables is defined as

$$\mathrm{OR} = \frac{n_{11}/n_{12}}{n_{21}/n_{22}} = \frac{n_{11}n_{22}}{n_{12}n_{21}}$$

The odds ratio can be any nonnegative number. When the row and column variables are independent, the true value of the odds ratio equals 1. An odds ratio greater than 1 indicates that the odds of a positive response are higher in row 1 than in row 2. Values less than 1 indicate the odds of positive response are higher in row 2. The strength of association increases with the deviation from 1.

The transformation $G = (\mathrm{OR} - 1)\,/\,(\mathrm{OR} + 1)$ transforms the odds ratio to the range $(-1, 1)$ such that $G = 0$ when $\mathrm{OR} = 1$, $G = -1$ when $\mathrm{OR} = 0$, and G is close to 1 for very large values of OR. G is the gamma statistic, which PROC FREQ computes when you specify the MEASURES option.

The asymptotic $100\,(1 - \alpha)$ percent confidence limits for the odd ratio are

$$\left(\mathrm{OR}\cdot\;\exp\left(-z\sqrt{v}\,\right),\;\mathrm{OR}\cdot\;\exp\left(z\sqrt{v}\,\right)\right)$$

where

$$v = var\,(\ln\ \text{OR}) = \frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}$$

and z is the $100\,(1 - \alpha/2)$ percentile of the standard normal distribution. If any of the four cell frequencies are zero, the estimates are not computed.

When you specify the OR option in the EXACT statement PROC FREQ computes exact confidence limits for the odds ratio using an iterative algorithm based on that presented by Thomas (1971). Because this is a discrete problem, the confidence coefficient for these exact confidence limits is not exactly $1 - \alpha$, but is at least $1 - \alpha$. Thus, these confidence limits are conservative. Refer to Agresti (1992).

Relative Risks (Cohort Studies)

These measures of relative risk are useful in *cohort* (prospective) study designs, where two samples are identified based on the presence or absence of an explanatory factor. The two samples are observed in future time for the binary (yes-no) response variable under study. Relative risk measures are also useful in cross-sectional studies, where two variables are observed simultaneously. Refer to Stokes et al. (1995) and Agresti (1996).

The column 1 relative risk is the ratio of the column 1 risks for row 1 to row 2. The column 1 risk for row 1 is the proportion of the row 1 observations classified in column 1,

$$p_{1|1} = n_{11}/n_{1\cdot}$$

Similarly, the column 1 risk for row 2 is

$$p_{1|2} = n_{21}/n_{2\cdot}$$

The column 1 relative risk is then computed as

$$\text{RR}_1 = \frac{p_{1|1}}{p_{1|2}}$$

A relative risk greater than 1 indicates that the probability of positive response is greater in row 1 than in row 2. Similarly, a relative risk that is less than 1 indicates that the probability of positive response is less in row 1 than in row 2. The strength of association increases with the deviation from 1.

The asymptotic $100\,(1 - \alpha)$ percent confidence limits for the column 1 relative risk are

$$\left(\text{RR}_1 \cdot \exp\left(-z\sqrt{v}\right),\ \text{RR}_1 \cdot \exp\left(z\sqrt{v}\right)\right)$$

where

$$v = var\,(\ln\ \text{RR}_1) = \frac{1 - p_{1|1}}{n_{11}} + \frac{1 - p_{1|2}}{n_{21}}$$

and z is the $100\,(1-\alpha/2)$ percentile of the standard normal distribution. If either n_{11} or n_{21} is zero, PROC FREQ does not compute the relative risks.

The column 2 relative risks are computed similarly.

Cochran-Armitage Test for Trend

The TREND option in the TABLES statement requests the Cochran-Armitage test for trend, which tests for trend in binomial proportions across levels of a single factor or covariate. This test is appropriate for a contingency table where one variable has two levels and the other variable is ordinal. The two-level variable represents the response, and the other variable represents an explanatory variable with ordered levels. When the contingency table has two columns and *R* rows, PROC FREQ tests for trend across the *R* levels of the row variable. When the table has two rows and *C* columns, PROC FREQ tests for trend across the *C* levels of the column variable.

The trend test is based upon the regression coefficient for the weighted linear regression of the binomial proportions on the scores of the levels of the explanatory variable. Refer to Margolin (1988) and Agresti (1990). If the contingency table has two columns and *R* rows, the trend test statistic is computed as

$$T = \frac{\sum_{i=1}^{R} n_{i1}\left(R_i - \overline{R}\right)}{\sqrt{p_{\cdot 1}\,(1-p_{\cdot 1})\,s^2}}$$

where

$$s^2 = \sum_{i=1}^{R} n_{i\cdot}\left(R_i - \overline{R}\right)^2$$

The row scores R_i are determined by the value of the SCORES= option in the TABLES statement. By default, PROC FREQ uses TABLE scores. For character variables, the TABLE scores for the row variable are the row numbers (for example, 1 for the first row, 2 for the second row, and so on). For numeric variables, the TABLE score for each row is the numeric value of the row level. When you perform the trend test, the explanatory variable may be numeric (for example, dose of a test substance), and these variable values may be appropriate scores. If the explanatory variable has ordinal levels that are not numeric, you can assign meaningful scores to the variable levels. Sometimes equidistant scores, such as the TABLE scores for a character variable, may be appropriate. For more information on choosing scores for the trend test, refer to Margolin (1988).

The null hypothesis for the Cochran-Armitage test is no trend, which means the binomial proportion $p_{i1} = n_{i1}/n_{i\cdot}$ is the same for all levels of the explanatory variable.Under this null hypothesis, the trend test statistic is asymptotically distributed as a standard normal random variable. In addition to this asymptotic test, PROC FREQ can compute the exact test for trend, which you request by specifying the TREND option in the EXACT statement. See the “EXACT Statement” on page 521 for information on exact tests.

PROC FREQ computes one-sided and two-sided *p*-values for the trend test. When the test statistic is greater than its expected value of zero, PROC FREQ computes the right-sided *p*-value, which is the probability of a larger value of the statistic occurring under the null hypothesis. A small right-sided p-value supports the alternative hypothesis of increasing trend in column 1 probability from row 1 to row R. When the test statistic is less than or equal to zero, PROC FREQ computes the left-sided p-value. A small left-sided p-value supports the alternative of decreasing trend. The one-sided p-value P_1 can be expressed as

$$P_1 = \text{Prob } (\text{Trend Statistic} > T) \quad \text{if } T > 0$$
$$P_1 = \text{Prob } (\text{Trend Statistic} < T) \quad \text{if } T \leq 0$$

The two-sided p-value P_2 is computed as

$$P_2 = \text{Prob } (|\text{Trend Statistic}| > |T|)$$

Jonckheere-Terpstra Test

The JT option in the TABLES statement requests the Jonckheere-Terpstra test, which is a nonparametric test for ordered differences among classes. It tests the null hypothesis that the distribution of the response variable does not differ among classes. It is designed to detect alternatives of ordered class differences, which can be expressed as $\tau_1 \leq \tau_2 \leq \ldots \leq \tau_R$ (or $\tau_1 \geq \tau_2 \geq \ldots \geq \tau_R$) with at least one of the inequalities being strict, where τ_i denotes the effect of class i. For such ordered alternatives, the Jonckheere-Terpstra test can be preferable to tests of more general class difference alternatives, such as the Kruskal-Wallis test (requested by the WILCOXON option in the NPAR1WAY procedure). Refer to Pirie (1983) and Hollander and Wolfe (1973) for more information about the Jonckheere-Terpstra test.

The Jonckheere-Terpstra test is appropriate for a contingency table where an ordinal column variable represents the response. The row variable, which can be nominal or ordinal, represents the classification variable. The levels of the row variable should be ordered according to the ordering you want the test to detect. The order of variable levels is determined by the ORDER= option in the PROC FREQ statement. The default is ORDER=INTERNAL, which orders by unformatted value. If you specify ORDER=DATA, PROC FREQ orders values according to their order in the input data set. For more information on how to order variable levels, see the ORDER= option on page 520.

The Jonckheere-Terpstra test statistic is computed by first forming $R(R-1)/2$ Mann-Whitney counts $M_{i,i'}$, where $i < i'$, for pairs of rows in the contingency table,

$$M_{i,i'} = \{\text{number of times } X_{i,j} < X_{i',j'}, j = 1, \ldots, n_{i\cdot};\ j' = 1, \ldots, n_{i'\cdot}\} + \frac{1}{2}\{\text{number of times } X_{i,j} = X_{i',j'}, j = 1, \ldots, n_{i\cdot};\ j' = 1, \ldots, n_{i'\cdot}\}$$

where $X_{i,j}$ is response j in row i. Then the Jonckheere-Terpstra test statistic is computed as

$$J = \sum_{1 \leq i <} \sum_{i' \leq \text{R}} M_{i,i'}$$

This test rejects the null hypothesis of no difference among classes for large values of J. Asymptotic p-values for the Jonkheere-Terpstra test are obtained by using the normal approximation for the distribution of the standardized test statistic. The standardized test statistic is computed as

$$J^* = \frac{J - E_0(J)}{\sqrt{var_0(J)}}$$

where E_0 and $var_0(J)$ are the expected value and variance of the test statistic under the null hypothesis.

$$E_0(J) = \left(n^2 - \sum_i n_{i\cdot}^2 \right) / 4$$

$$var_0(J) = A/72 + B/\left[36n(n-1)(n-2)\right] + C/\left[8n(n-1)\right]$$

where

$$A = n(n-1)(2n+5) - \sum_i n_{i\cdot}(n_{i\cdot}-1)(2n_{i\cdot}+5) - \sum_j n_{\cdot j}(n_{\cdot j}-1)(2n_{\cdot j}+5)$$

$$B = \left[\sum_i n_{i\cdot}(n_{i\cdot}-1)(n_{i\cdot}-2) \right] \left[\sum_j n_{\cdot j}(n_{\cdot j}-1)(n_{\cdot j}-2) \right]$$

$$C = \left[\sum_i n_{i\cdot}(n_{i\cdot}-1) \right] \left[\sum_j n_{\cdot j}(n_{\cdot j}-1) \right]$$

In addition to this asymptotic test, PROC FREQ can compute the exact Jonckheere-Terpstra test, which you request by specifying the JT option in the EXACT statement. See the “EXACT Statement” on page 521 for information on exact tests.

PROC FREQ computes one-sided and two-sided p-values for the Jonckheere-Terpstra test. When the standardized test statistic is greater than its expected value of 0, PROC FREQ computes the right-sided p-value, which is the probability of a larger value of the statistic occurring under the null hypothesis. A small right-sided p-value supports the alternative hypothesis of increasing order from row 1 to row R. When the standardized test statistic is less than or equal to 0, PROC FREQ computes the left-sided p-value. A small left-sided p-value supports the alternative of decreasing order from row 1 to row R. The one-sided p-value, P_1, can be expressed as

$$P_1 = \text{Prob (Std JT Statistic} > J^*) \quad \text{if } J^* > 0$$
$$P_1 = \text{Prob (Std JT Statistic} < J^*) \quad \text{if } J^* \le 0$$

The two-sided p-value, P_2, is computed as

$$P_2 = \text{Prob}\ (|\text{Std JT Statistic}| \ > |J^*|)$$

Tests and Measures of Agreement

When you specify the AGREE option in the TABLES statement, PROC FREQ computes tests and measures of agreement for square tables (that is, for tables where the number of rows equals the number of columns). For two-way tables, these tests and measures include McNemar's test for 2×2 tables, Bowker's test of symmetry, the simple kappa coefficient, and the weighted kappa coefficient. For multiple strata (n-way tables, where $n > 2$), PROC FREQ computes the overall simple kappa coefficient and the overall weighted kappa coefficient, as well as tests for equal kappas (simple and weighted) among strata. For multiple strata of 2×2 tables, PROC FREQ computes Cochran's Q.

PROC FREQ computes the kappa coefficients (simple and weighted), their asymptotic standard errors, and their confidence limits when you specify the AGREE option in the TABLES statement. If you also specify the KAPPA option in the TEST statement, then PROC FREQ computes the asymptotic test of the hypothesis that simple kappa equals zero. Similarly, if you specify WTKAP in the TEST statement, PROC FREQ computes the asymptotic test for weighted kappa.

In addition to the asymptotic tests that are described in this section, PROC FREQ also computes the exact p-value for McNemar's test when you specify the keyword MCNEM in the EXACT statement. For the kappa statistic, PROC FREQ computes an exact test of the hypothesis that kappa (or weighted kappa) equals zero when you specify KAPPA (or WTKAP) in the EXACT statement. See "Exact Statistics" on page 581 for more information about these tests.

The discussion of each test and measure of agreement provides the formulas that PROC FREQ uses to compute the AGREE statistics. For information about the use and interpretation of these statistics, refer to Agresti (1990), Agresti (1996), Fleiss (1981), and the references that follow.

McNemar's Test

PROC FREQ computes McNemar's test for 2×2 tables when you specify the AGREE option. McNemar's test is appropriate when you are analyzing data from matched pairs of subjects with a dichotomous (yes-no) response. It tests for marginal homogeneity, or a null hypothesis of $p_{1\cdot} = p_{\cdot 1}$. McNemar's test is computed as

$$Q_M = \frac{(n_{12} - n_{21})^2}{n_{12} + n_{21}}$$

Under the null hypothesis, Q_M has an asymptotic chi-square distribution with one degree of freedom. Refer to McNemar (1947), as well as the references cited on page 569 in the preceding section. PROC FREQ also computes an exact p-value for McNemar's test when you specify MCNEM in the EXACT statement.

Bowker's Test of Symmetry

PROC FREQ computes Bowker's test of symmetry for square two-way tables that are larger than 2×2. (For 2×2 tables, Bowker's test is identical to McNemar's test.) For Bowker's test of symmetry, the null hypothesis is that the probabilities in the square table satisfy symmetry, or that $p_{ij} = p_{ji}$ for all pairs of table cells. When there are more than two categories for each variable, Bowker's test of symmetry is calculated as

$$Q_B = \sum_{i<j}\sum \frac{(n_{ij} - n_{ji})^2}{n_{ij} + n_{ji}}$$

For large samples, Q_B has an asymptotic chi-square distribution with $R(R-1)/2$ degrees of freedom under the null hypothesis of symmetryof the expected counts. Refer to Bowker (1948). For two categories, this test of symmetry is identical to McNemar's test.

Simple Kappa Coefficient

The simple kappa coefficient, introduced by Cohen (1960), is a measure of interrater agreement:

$$\widehat{\kappa} = \frac{P_0 - P_e}{1 - P_e}$$

where $P_0{=}\Sigma_i p_{ii}$ and $P_e{=}\Sigma_i p_{i\cdot} p_{\cdot i}$. Viewing the two response variables as two independentratings of the n subjects, the kappa coefficient equals +1 when there is complete agreement of the raters. When the observed agreement exceeds chance agreement, the kappa coefficient is positive, with its magnitude reflecting the strength of agreement. Although unusual in practice, kappa is negative when the observed agreement is less than chance agreement. The minimum value of kappa is between –1 and 0, depending on the marginal proportions.

The asymptotic variance of the simple kappa coefficient is estimated by the following, according to Fleiss et al. (1969):

$$var = \frac{A + B - C}{(1 - P_e)^2\, n}$$

where

$$A = \sum_i p_{ii}\left[1 - (p_{i\cdot} + p_{\cdot i})(1 - \widehat{\kappa})\right]^2$$

$$B = (1 - \widehat{\kappa})^2 \sum_{i \neq j}\sum p_{ij}\,(p_{\cdot i} + p_{j\cdot})^2$$

and

$$C = \left[\widehat{\kappa} - P_e(1 - \widehat{\kappa})\right]^2$$

PROC FREQ computes confidence limits for the simple kappa coefficient according to

$$\widehat{\kappa} \pm z_{\alpha/2} \cdot \sqrt{var}$$

where $z_{\alpha/2}$ is the $100\left(1-\alpha/2\right)$ percentile of the standard normal distribution. The value of α is determined by the value of the ALPHA= option, which by default equals 0.05 and produces 95 percent confidence limits.

To compute an asymptotic test for the kappa coefficient, PROC FREQ uses a standardized test statistic $\widehat{\kappa}^*$, which has an asymptotic standard normal distribution under the null hypothesis that kappa equals zero. The standardized test statistic is computed as

$$\widehat{\kappa}^* = \frac{\widehat{\kappa}}{\sqrt{var_0\left(\widehat{\kappa}\right)}}$$

where $var_0\left(\widehat{\kappa}\right)$ is the variance of the kappa coefficient under the nullhypothesis.

$$var_0\left(\widehat{\kappa}\right) = \frac{P_e + P_e^2 - \sum_i p_{i\cdot}p_{\cdot i}\left(p_{i\cdot} + p_{\cdot i}\right)}{\left(1-P_e\right)^2 n}$$

Refer to Fleiss (1981).

In addition to the asymptotic test for kappa, PROC FREQ computes an exact test when you specify the KAPPA option or the AGREE option in the EXACT statement. See “Exact Statistics” on page 581 for more information on exact tests.

Weighted Kappa Coefficient

The weighted kappa coefficient is a generalization of the simple kappa coefficient, using weights to quantify the relative difference between categories. PROC FREQ computes the weights from the column scores, using either the Cicchetti-Allison weight type or the Fleiss-Cohen weight type, which are described below. The weights w_{ij} are constructed so that $0 \leq w_{ij} < 1$ for all $i \neq j$, $w_{ii} = 1$ for all i, and $w_{ij} = w_{ji}$. The weighted kappa coefficient is defined as

$$\widehat{\kappa}_w = \frac{P_{o(w)} - P_{e(w)}}{1 - P_{e(w)}}$$

where

$$P_{o(w)} = \sum_i \sum_j w_{ij} p_{ij}$$

and

$$P_{e(w)} = \sum_i \sum_j w_{ij} p_{i\cdot} p_{\cdot j}$$

For 2×2 tables, the weighted kappa coefficient is identical to the simple kappa coefficient. Therefore, PROC FREQ displays only the simple kappa coefficient for 2×2 tables. The asymptotic variance of the weighted kappa coefficient is estimated by the following, according to Fleiss et al. (1969):

$$var = \frac{\sum_i \sum_j p_{ij} \left[w_{ij} - (\overline{w}_{i\cdot} + \overline{w}_{\cdot j})(1 - \widehat{\kappa}_w) \right]^2 - \left[\widehat{\kappa}_w - P_{e(w)} (1 - \widehat{\kappa}_w) \right]^2}{\left(1 - P_{e(w)}\right)^2 n}$$

where

$$\overline{w}_{i\cdot} = \sum_j p_{\cdot j} w_{ij}$$

and

$$\overline{w}_{\cdot j} = \sum_i p_{i\cdot} w_{ij}$$

PROC FREQ computes confidence limits for the weighted kappa coefficient according to

$$\widehat{\kappa}_w \pm z_{\alpha/2} \cdot \sqrt{var}$$

where $z_{\alpha/2}$ is the $100\,(1-\alpha/2)$ percentile of the standard normal distribution. The value of α is determined by the value of the ALPHA= option, which by default equals 0.05 and produces 95 percent confidence limits.

To compute an asymptotic test for the weighted kappa coefficient, PROC FREQ uses a standardized test statistic $\widehat{\kappa}_w^*$, which has an asymptotic standard normal distribution underthe null hypothesis. The standardized test statistic is computed as

$$\widehat{\kappa}_w^* = \frac{\widehat{\kappa}_w}{\sqrt{var_0\left(\widehat{\kappa}_w\right)}}$$

where $var_0\left(\widehat{\kappa}_w\right)$ is the variance of the kappa coefficient under the null hypothesis.

$$var_0\left(\widehat{\kappa}_w\right) = \frac{\sum_i \sum_j p_{i\cdot} p_{\cdot j} \left[w_{ij} - (\overline{w}_{i\cdot} + \overline{w}_{\cdot j}) \right]^2 - P_{e(w)}^2}{\left(1 - P_{e(w)}\right)^2 n}$$

Refer to Fleiss (1981).

In addition to the asymptotic test for weighted kappa, PROC FREQ computes the exact test when you specify the WTKAP option or the AGREE option in the EXACT statement. See "Exact Statistics" on page 581 for more information on exact tests.

PROC FREQ computes kappa coefficient weights using the column scores and one of two available weight types. The column scores are determined by the SCORES= option in the TABLES statement. The two available weight types are Cicchetti-Allison and Fleiss-Cohen. By default, PROC FREQ uses the Cicchetti-Allison type. If you specify WT=FC in the AGREE option, then PROC FREQ uses the Fleiss-Cohen weight type to construct kappa weights. To display the kappa weights, specify the PRINTKWT option in the TABLES statement.

PROC FREQ computes Cicchetti-Allison kappa coefficient weights using a form similar to that given by Cicchetti and Allison (1971).

$$w_{ij} = 1 - \frac{|C_i - C_j|}{C_C - C_1}$$

where C_i is the score for column i, and *C* is the number of categories. You can specify the type of score using the SCORES= option in the TABLES statement. If you do not specify the SCORES= option, PROC FREQ uses TABLE scores. For numeric variables, TABLE scores are the numeric values of the variable levels. You can assign numeric values to the categories in a way that reflects their level of similarity. For example, suppose you have four categories and order them according to similarity. If you assign them values of 0, 2, 4, and 10, the following weights are used for computing the weighted kappa coefficient: $w_{12} = .8, w_{13} = .6, w_{14} = 0, w_{23} = .8, w_{24} = .2$, and $w_{34} = .4$.

If you specify (WT=FC) with the AGREE option in the TABLES statement, PROC FREQ computes Fleiss-Cohen kappa coefficient weights using a form similar to that given by Fleiss and Cohen (1973).

$$w_{ij} = 1 - \frac{(C_i - C_j)^2}{(C_C - C_1)^2}$$

Overall Kappa Coefficient

When there are multiple strata, PROC FREQ combines the stratum-level estimates of kappa into an overall estimate of the supposed common value of kappa. Assume there are q strata, indexed by $h = 1, 2, \ldots, q$, and let $var\left(\widehat{\kappa}_h\right)$ denote the variance of $\widehat{\kappa}_h$. Then the estimate of the overall kappa, according to Fleiss (1981), is computed as follows:

$$\widehat{\kappa}_{overall} = \sum_{h=1}^{q} \frac{\widehat{\kappa}_h}{var\left(\widehat{\kappa}_h\right)} \Big/ \sum_{h=1}^{q} \frac{1}{var\left(\widehat{\kappa}_h\right)}$$

An estimate of the overall weighted kappa is computed similarly.

Tests for Equal Kappa Coefficients

The following chi-square statistic, with $q - 1$ degrees of freedom, is used to test whether the values of the kappa are equal among the q strata:

$$Q_\kappa = \sum_{h=1}^{q} \frac{\left(\widehat{\kappa}_h - \widehat{\kappa}_{overall}\right)^2}{var\left(\widehat{\kappa}_h\right)}$$

A similar test is done for weighted kappa coefficients.

Cochran's *Q* Test

When there are multiple strata and two response categories, Cochran's **Q** statistic is used to test the homogeneity of the one-dimensional margins. Let m denote the number of variables and N denote the total number of subjects. Then Cochran's **Q** statistic is computed as follows:

$$Q_C = (m-1)\,\frac{m\sum_{j=1}^{m} T_j^2 - T^2}{mT - \sum_{k=1}^{N} S_k^2}$$

where T_j is the number of positive responses for variable j, T is the total number of positive responses over all variables, and S_k is the number of positive responses for subject k. Under the null hypothesis, Cochran's **Q** is an approximate chi-square statistic with $m-1$ degrees of freedom. Refer to Cochran (1950). When there are two variables ($m=2$), Cochran's **Q** simplifies to McNemar's statistic. When there are more than two response categories, you can test for marginal homogeneity using the repeated measures capabilities of the CATMOD procedure.

Tables with Zero Rows or Columns

The AGREE statistics are defined only for square tables, where the number of rows equals the number of columns. If the table is not square, PROC FREQ does not compute AGREE statistics. In the kappa statistic framework, where two independent raters are assigning ratings to each of the *n* subjects, suppose one of the raters does not use all possible *r* rating levels. If the corresponding table has *r* rows but only $r-1$ columns, then the table is not square, and PROC FREQ does not compute the AGREE statistics. To create a square table in this situation, use the ZEROS option in the WEIGHT statement, which requests that PROC FREQ include observations with zero weights in the analysis. And input zero-weight observations to represent any rating levels that are not used by a rater, so that the input data set has at least one observation for each possible rater and rating combination. This includes all rating levels in the analysis, whether or not all levels are actually assigned by both raters. The resulting table is a square table, $r \times r$, and so all AGREE statistics can be computed.

For more information on the ZEROS option, see "Using Zero Weights" on page 541. By default, PROC FREQ does not process observations that have zero weights, because these observations do not contribute to the total frequency count, and because any resulting zero-weight row or column causes many of the tests and measures of association to be undefined. However, kappa statistics are defined for tables with a zero-weight row or column, and the ZEROS option allows input of zero-weight observations so you can construct the tables needed to compute kappas.

Cochran-Mantel-Haenszel Statistics

For *n*-way crosstabulation tables, consider the following example:

```
proc freq;
   tables a*b*c*d / cmh;
run;
```

The CMH option in the TABLES statement gives a stratified statistical analysis of the relationship between C and D, controlling for A and B. The stratified analysis provides a way to adjust for the possible confounding effects of A and B without being forced to estimate parameters for them. The analysis produces Cochran-Mantel-Haenszel statistics, and for 2×2 tables, it includes estimation of the common odds ratio, common relative risks, and the Breslow-Day test for homogeneity of the odds ratios.

Let the number of strata be denoted by q, indexing the strata by $h = 1, 2, \ldots, q$. Each stratum contains a contingency table with X representing the row variable and Y representing the column variable. For table h, denote the cell frequency in row i and column j by n_{hij}, with corresponding row and column marginal totals denoted by $n_{hi\cdot}$ and $n_{h\cdot j}$ and the overall stratum total by n_h .

Because the formulas for the Cochran-Mantel-Haenszel statistics are more easily defined in terms of matrices, the following notation is used. Vectors are presumed to be column vectors unless they are transposed (′).

$$
\begin{aligned}
\mathbf{n}'_{hi} &= (n_{hi1}, n_{hi2}, \ldots, n_{hi\mathrm{C}}) && (1{\times}C) \\
\mathbf{n}'_{h} &= (\mathbf{n}'_{h1}, \mathbf{n}'_{h2}, \ldots, \mathbf{n}'_{h\mathrm{R}}) && (1{\times}RC) \\
p_{hi\cdot} &= \frac{n_{hi\cdot}}{n_h} && (1{\times}1) \\
p_{h\cdot j} &= \frac{n_{h\cdot j}}{n_h} && (1{\times}1) \\
\mathbf{P}'_{h*\cdot} &= (p_{h1\cdot}, p_{h2\cdot}, \ldots, p_{h\mathrm{R}\cdot}) && (1{\times}R) \\
\mathbf{P}'_{h\cdot *} &= (p_{h\cdot 1}, p_{h\cdot 2}, \ldots, p_{h\cdot \mathrm{C}}) && (1{\times}C)
\end{aligned}
$$

Assume that the strata are independent and that the marginal totals of each stratum are fixed. The null hypothesis, H_0, is that there is no association between X and Y in any of the strata. The corresponding model is the multiple hypergeometric, which implies that under H_0, the expected value and covariance matrix of the frequencies are, respectively,

$$\mathbf{m}_h = \mathbf{E}\left[\mathbf{n}_h | \mathrm{H}_0\right] = n_h \left(\mathbf{P}_{h\cdot *} \otimes \mathbf{P}_{h*\cdot}\right)$$

and

$$\mathbf{var}\left[\mathbf{n}_h | \mathrm{H}_0\right] = c\left[\left(\mathbf{D}_{\mathbf{P}h\cdot *} - \mathbf{P}_{h\cdot *}\mathbf{P}'_{h\cdot *}\right) \otimes \left(\mathbf{D}_{\mathbf{P}h*\cdot} - \mathbf{P}_{h*\cdot}\mathbf{P}'_{h*\cdot}\right)\right]$$

where

$$c = \frac{n_h^2}{n_h - 1}$$

and where $\otimes$ denotes Kronecker product multiplication and $\mathbf{D_a}$ is a diagonal matrix with elements of $\mathbf{a}$ on the main diagonal.

The generalized CMH statistic (Landis, Heyman, and Koch 1978) is defined as

$$Q_{\mathrm{CMH}} = \mathbf{G}'\mathbf{V}_{\mathbf{G}}^{-1}\mathbf{G}$$

where

$$\mathbf{G} = \sum_h \mathbf{B}_h \left(\mathbf{n}_h - \mathbf{m}_h\right)$$
$$\mathbf{V_G} = \sum_h \mathbf{B}_h \left(\mathbf{Var}\left(\mathbf{n}_h | \mathrm{H}_0\right)\right) \mathbf{B}_h'$$

and where

$$\mathbf{B}_h = \mathbf{C}_h \otimes \mathbf{R}_\mathrm{h}$$

is a matrix of fixed constants based on column scores $\mathbf{C}_h$ and row scores $\mathbf{R}_\mathrm{h}$. When the null hypothesis is true, the CMH statistic has an asymptotic chi-square distribution with degrees of freedom equal to the rank of $\mathbf{B}_h$. If $\mathbf{V_G}$ is found to be singular, PROC FREQ displays a message and sets the value of the CMH statistic to missing.

PROC FREQ computes three CMH statistics using this formula for the generalized CMH statistic, with different row and column score definitions for each statistic. The CMH statistics that PROC FREQ computes are the correlation statistic, the ANOVA (row mean scores) statistic, and the general association statistic. These statistics test the null hypothesis of no association against different alternative hypotheses. The following sections describe the computation of these CMH statistics.

CAUTION:

CMH statistics have low power for detecting an association when the patterns of association for some of the strata are in the opposite direction of the patterns displayed by other strata. Thus, a nonsignificant CMH statistic suggests either that there is no association or that no pattern of association has enough strength or consistency to dominate any other pattern. △

Correlation Statistic

The correlation statistic, with one degree of freedom, was popularized by Mantel and Haenszel (1959) and Mantel (1963) and is therefore known as the Mantel-Haenszel statistic.

The alternative hypothesis is that there is a linear association between X and Y in at least one stratum. If either X or Y does not lie on an ordinal (or interval) scale, then this statistic is meaningless.

To compute the correlation statistic, PROC FREQ uses the formula for the generalized CMH statistic with the row and column scores determined by the SCORES= option in the TABLES statement. See "Scores" on page 545 for more information on the available score types. The matrix of row scores $\mathbf{R}_h$ has dimension $1 \times R$, and the matrix of column scores $\mathbf{C}_h$ has dimension $1 \times C$.

When there is only one stratum, this CMH statistic reduces to $(n-1)\,r^2$, where r is the Pearson correlation coefficient between X and Y. When you specify nonparametric (RANK, RIDIT, or MODRIDIT) scores, the statistic reduces to $(n-1)\,r_s^2$, where r_s is the Spearman rank correlation coefficient between X and Y. When there is more than one stratum, then the CMH statistic becomes a stratum-adjusted correlation statistic.

ANOVA (Row Mean Scores) Statistic

The *ANOVA* statistic can be used only when the column variable Y lies on an ordinal (or interval) scale so that the mean score of Y is meaningful. For the ANOVA statistic, the mean score is computed for each row of the table, and the alternative hypothesis is that, for at least one stratum, the mean scores of the R rows are unequal. In other words, the statistic is sensitive to location differences among the R distributions of Y.

The matrix of column scores $\mathbf{C}_h$ has dimension $1 \times C$, and the scores, one for each column, are specified in the SCORES= option. The matrix $\mathbf{R}_h$ has dimension $(R-1) \times R$ which PROC FREQ creates internally as

$$\mathbf{R}_h = [\mathbf{I}_{R-1}, -\mathrm{J}_{R-1}]$$

where $\mathbf{I}_{R-1}$ is an identity matrix of rank $R-1$, and $\mathbf{J}_{R-1}$ is an $(R-1) \times 1$ vector of ones. This matrix has the effect of forming $R-1$ independent contrasts of the R mean scores.

When there is only one stratum, this CMH statistic is essentially an analysis-of-variance (*ANOVA*) statistic in the sense that it is a function of the variance ratio *F* statistic that would be obtained from a one-way *ANOVA* on the dependent variable Y. If nonparametric scores are specified in this case, then the *ANOVA* statistic is a Kruskal-Wallis test.

If there is more than one stratum, then this CMH statistic corresponds to a stratum-adjusted *ANOVA* or Kruskal-Wallis test. In the special case where there is one subject per row and one subject per column in the contingency table of each stratum, then this CMH statistic is identical to Friedman's chi-square. See Example 8 on page 615 for an illustration.

General Association Statistic

The alternative hypothesis for the general association statistic is that, for at least one stratum, there is some kind of association between X and Y. This statistic is always interpretable because it does not require an ordinal scale for either X or Y.

For the general association statistic, the matrix $\mathbf{R}_h$ is the same as the one used for the *ANOVA* statistic. The matrix $\mathbf{C}_h$ is defined similarly as

$$\mathbf{C}_h = [\mathbf{I}_{C-1}, -\mathbf{J}_{C-1}]$$

PROC FREQ generates both score matrices internally. When there is only one stratum, then the general association CMH statistic reduces to $Q_P\,(n-1)\,/n$, where Q_P is the Pearson chi-square statistic. When there is more than one stratum, then the CMH statistic becomes a stratum-adjusted Pearson chi-square statistic. Note that a similar adjustment is made by summing the Pearson chi-squares across the strata. However, the latter statistic requires a large sample size in each stratum to support the resulting chi-square distribution with $q\,(R-1)\,(C-1)$ degrees of freedom. The CMH statistic requires only a large overall sample size because it has only $(R-1)\,(C-1)$ degrees of freedom.

Refer to Cochran (1954); Mantel and Haenszel (1959); Mantel (1963); Birch (1965); and Landis et al. (1978).

Adjusted Odds Ratio and Relative Risk Estimates

The CMH option provides adjusted odds ratio and relative risk estimates for stratified 2×2 tables. For each of these measures, PROC FREQ computes the Mantel-Haenszel estimate and the logit estimate. These estimates apply to n-way table requests in the TABLES statement, when the row and column variables both have only two levels. For example,

```
proc freq;
   tables a*b*c*d / cmh;
run;
```

In this example, if the row and column variables C and D both have two levels, PROC FREQ provides odds ratio and relative risk estimates, adjusting for the confounding variables A and B.

The choice of an appropriate measure depends on the study design. For case-control (retrospective) studies, the odds ratio is appropriate. For cohort (prospective) or cross-sectional studies, the relative risk is appropriate. See "Odds Ratio and Relative Risks for 2×2 Tables" on page 564 for more information on these measures.

Throughout this section, z is the $100\,(1-\alpha/2)$ percentile of the standard normal distribution.

Odds Ratio (Case-Control Studies): Mantel-Haenszel Adjusted

The Mantel-Haenszel adjusted odds ratio estimator is given by

$$\mathrm{OR}_{\mathrm{MH}} = \frac{\sum_h n_{h11} n_{h22} / n_h}{\sum_h n_{h12} n_{h21} / n_h}$$

It is always computed unless the denominator is zero. Refer to Mantel and Haenszel (1959) and Agresti (1990).

Using the estimated variance for $\log\left(\mathrm{OR}_{\mathrm{MH}}\right)$ given by Robins et al. (1986), PROC FREQ computes the corresponding $100\,(1-\alpha)$ percent confidence limits for the odds ratio as

$$\left(\mathrm{OR}_{\mathrm{MH}} \cdot \exp\left(-z\hat{\sigma}\right),\, \mathrm{OR}_{\mathrm{MH}} \cdot \exp\left(z\hat{\sigma}\right)\right)$$

where

$$\begin{aligned}
\hat{\sigma}^2 &= var\ \left[\ln \mathrm{OR}_{\mathrm{MH}}\right] \\
&= \frac{\sum_h \left(n_{h11}+n_{h22}\right)\left(n_{h11}n_{h22}\right)/n_h^2}{2(\sum_h n_{h11}n_{h22}/n_h)^2} \\
&\quad + \frac{\sum_h \left[\left(n_{h11}+n_{h22}\right)\left(n_{h12}n_{h21}\right)+\left(n_{h12}+n_{h21}\right)\left(n_{h11}n_{h22}\right)\right]/n_h^2}{2(\sum_h n_{h11}n_{h22}/n_h)(\sum_h n_{h12}n_{h21}/n_h)} \\
&\quad + \frac{\sum_h \left(n_{h12}+n_{h21}\right)\left(n_{h12}n_{h21}\right)/n_h^2}{2(\sum_h n_{h12}n_{h21}/n_h)^2}
\end{aligned}$$

Note that the Mantel-Haenszel odds ratio estimator is less sensitive to small n_h than the logit estimator.

Odds Ratio (Case-Control Studies): Adjusted Logit

The adjusted logit odds ratio estimator (Woolf 1955) is given by

$$\mathrm{OR}_{\mathrm{L}} = \exp\left(\frac{\sum_h w_h \ln \mathrm{OR}_h}{\sum_h w_h} \right)$$

and the corresponding $100\,(1-\alpha)$ percent confidence limits are

$$\left(\mathrm{OR}_{\mathrm{L}} \cdot \exp\left(-z/\sqrt{\textstyle\sum_h w_h} \right) ,\mathrm{OR}_{\mathrm{L}} \cdot \exp\left(z/\sqrt{\textstyle\sum_h w_h} \right) \right)$$

where OR_h is the odds ratio for stratum h, and

$$w_h = \frac{1}{var\,(\ln \mathrm{OR}_h)}$$

Refer to Woolf (1955)

If any cell frequency in a stratum h is zero, then PROC FREQ adds 0.5 to each cell of the stratum before computing OR_h and w_h (Haldane 1955), and displays a warning.

Relative Risks (Cohort Studies)

The Mantel-Haenszel estimate of the common relative risk for column 1 is computed as

$$\mathrm{RR}_{\mathrm{MH}} = \frac{\sum_h n_{h11} n_{h2\cdot} / n_h}{\sum_h n_{h21} n_{h1\cdot} / n_h}$$

It is always computed unless the denominator is zero. Refer to Mantel and Haenszel (1959) and Agresti(1990).

Using the estimated variance for $\log\,(\mathrm{RR}_{\mathrm{MH}})$ given by Greenland and Robins (1985), PROC FREQ computes the corresponding confidence $100\,(1-\alpha)$ percent limits for the relative risk as

$$(\mathrm{RR}_{\mathrm{MH}} \cdot \exp\,(-z\hat{\sigma}\,),\ \mathrm{RR}_{\mathrm{MH}} \cdot \exp\,(z\hat{\sigma}\,))$$

where

$$\begin{aligned} \hat{\sigma}^2 &= v\hat{a}r^2\,[\ln \mathrm{RR}_{\mathrm{MH}}] \\ &= \frac{\sum_h \left(n_{h1\cdot} n_{h2\cdot} n_{h\cdot 1} - n_{h11} n_{h21} n_h \right) / n_h^2}{\left(\sum_h n_{h11} n_{h2\cdot} / n_h \right) \left(\sum_h n_{h21} n_{h1\cdot} / n_h \right)} \end{aligned}$$

The adjusted logit estimate of the common relative risk for column 1 is computed as

$$\mathrm{RR}_{\mathrm{L}} = \exp\left(\frac{\sum_h w_h \ln \mathrm{RR}_h}{\sum w_h} \right)$$

and the corresponding $100\,(1-\alpha)$ percent confidence limits are

$$\left(\mathrm{RR}_{\mathrm{L}} \cdot \exp\left(-z / \sqrt{\sum_h w_h} \right),\ \mathrm{RR}_{\mathrm{L}} \cdot \exp\left(z / \sqrt{\sum_h w_h} \right) \right)$$

where RR_h is the column 1 relative risk estimator for stratum h, and

$$w_h = \frac{1}{var\,(\ln \mathrm{RR}_h)}$$

If n_{h11} or n_{h21} is zero, then PROC FREQ adds 0.5 to each cell of the stratum before computing RR_h and w_h, and displays a warning.

Refer to Kleinbaum, Kupper, and Morgenstern (1982, Sections 17.4, 17.5) and Breslow and Day (1994).

Breslow-Day Test for Homogeneity of the Odds Ratios

When you specify the CMH option, PROC FREQ computes the Breslow-Day test for the stratified analysis of 2×2 tables. It tests the null hypothesis that the odds ratios from the q strata are all equal. When the null hypothesis is true, the statistic has approximately a chi-square distribution with $q-1$ degrees of freedom.

The Breslow-Day statistic is computed as

$$Q_{\mathrm{BD}} = \sum_h \frac{\left(n_{h11} - \mathrm{E}\,\left(n_{h11} | \mathrm{OR}_{\mathrm{MH}}\right)\right)^2}{var\,\left(n_{h11} | \mathrm{OR}_{\mathrm{MH}}\right)}$$

where *E* and *var* denote expected value and variance, respectively. The summation does not include any tables with a zero row or column. If $\mathrm{OR}_{\mathrm{MH}}$ equals zero or if it is undefined, then PROC FREQ does not compute the statistic, and displays a warning message.

For the Breslow-Day test to be valid, the sample size should be relatively large in each stratum, and at least 80% of the expected cell counts should be greater than 5. Note that this is a stricter sample size requirement than the one for the Cochran-Mantel-Haenszel test for $k \times 2 \times 2$ tables, in that each stratum sample size (not just the overall sample size) must be relatively large. Even when the Breslow-Day test is valid, it may not be very powerful against certain alternatives, as discussed in Breslow and Day (1980).

If you specify the BDT option, PROC FREQ computes the Breslow-Day test with Tarone's adjustment, which subtracts an adjustment factor from Q_{BD} to make the resulting statistic asymptotically chi-square.

$$Q_{\text{BDT}} = Q_{\text{BD}} - \frac{\left(\sum_h \left(n_{h11} - \text{E}\left(n_{h11} | \text{OR}_{\text{MH}} \right) \right) \right)^2}{\sum_h var\left(n_{h11} | \text{OR}_{\text{MH}} \right)}$$

Refer to Tarone (1985), Jones et al. (1989), and Breslow (1996).

Exact Statistics

Exact statistics can be useful in situations where the asymptotic assumptions are not met, and so the asymptotic *p*-values are not close approximations for the true *p*-values. Standard asymptotic methods involve the assumption that the test statistic follows a particular distribution when the sample size is sufficiently large. When the sample size is not large, asymptotic results may not be valid, with the asymptotic *p*-values differing perhaps substantially from the exact *p*-values. Asymptotic results may also be unreliable when the distribution of the data is sparse, skewed, or heavily tied. Refer to Agresti (1996) and Bishop et al. (1975). Exact computations are based on the statistical theory of exact conditional inference for contingency tables, reviewed by Agresti (1992).

In addition to computation of exact *p*-values, PROC FREQ provides the option of estimating exact *p*-values by Monte Carlo simulation. This can be useful for problems that are so large that exact computations require a great amount of time and memory, but for which asymptotic approximations may not be sufficient.

PROC FREQ provides exact *p*-values for the following tests for two-way tables: Pearson chi-square, likelihood-ratio chi-square, Mantel-Haenszel chi-square, Fisher's exact test, Jonckheere-Terpstra test, Cochran-Armitage test for trend, and McNemar's test. PROC FREQ can also compute exact *p*-values for tests of hypotheses that the following statistics are equal to zero: Pearson correlation coefficient, Spearman correlation coefficient, simple kappa coefficient, and weighted kappa coefficient. Additionally, PROC FREQ can compute exact confidence limits for the odds ratio for 2×2 tables. For one-way frequency tables, PROC FREQ provides the exact chi-square goodness-of-fit test (for equal proportions, or for proportions or frequencies that you specify). Also for one-way tables, PROC FREQ provides exact confidence limits for the binomial proportion, and an exact test for the binomial proportion value.

If the procedure does not complete the computation within the specified time, use MAXTIME= to increase the amount of clock time that PROC FREQ can use to compute the exact *p*-values directly or with Monte Carlo estimation.

The following sections summarize the computational algorithms, define the *p*-values that PROC FREQ computes, and discuss the computational resource requirements.

Computational Algorithms

PROC FREQ computes exact *p*-values for general $R\times C$ tables using the network algorithm developed by Mehta and Patel (1983). This algorithm provides a substantial advantage over direct enumeration, which can be very time-consuming and feasible only for small problems. Refer to Agresti (1992) for a review of algorithms for computation of exact *p*-values, and refer to Mehta et al. (1984, 1991) for information on the performance of the network algorithm.

The reference set for a given contingency table is the set of all contingency tables with the observed marginal row and column sums. Corresponding to this reference set, the network algorithm forms a directed acyclic network consisting of nodes in a number of stages. A path through the network corresponds to a distinct table in the reference set. The distances between nodes are defined so that the total distance of a path through the network is the corresponding value of the test statistic. At each node, the algorithm computes the shortest and longest path distances for all the paths that pass through that node. For statistics that can be expressed as a linear combination of cell frequencies multiplied by increasing row and column scores, PROC FREQ computes shortest and longest path distances using the algorithm given in Agresti et al. (1990). For statistics of other forms, PROC FREQ computes an upper limit for the longest path and a lower limit for the shortest path following the approach of Valz and Thompson (1994).

The longest and shortest path distances or limits for a node are compared to the value of the test statistic to determine whether all paths through the node contribute to the *p*-value, none of the paths through the node contribute to the *p*-value, or neither of these situations occur. If all paths through the node contribute, the *p*-value is incremented accordingly, and these paths are eliminated from further analysis. If no paths contribute, these paths are eliminated from the analysis. Otherwise, the algorithm continues, still processing this node and the associated paths. The algorithm finishes when all nodes have been accounted for, incrementing the *p*-value accordingly, or eliminated.

In applying the network algorithm, PROC FREQ uses full precision to represent all statistics, row and column scores, and other quantities involved in the computations. Although it is possible to use rounding to improve the speed and memory requirements of the algorithm, PROC FREQ does not do this because it can result in reduced accuracy of the *p*-values.

PROC FREQ computes exact confidence limits for the odds ratio according to an iterative algorithm based on that presented by Thomas (1971). Refer also to Gart (1971). Because this is a discrete problem, the confidence coefficient is not exactly $1 - \alpha$, but is at least $1 - \alpha$. Thus, these confidence limits are conservative.

For one-way tables, PROC FREQ computes the exact chi-square goodness-of-fit test by the method of Radlow and Alf (1975). PROC FREQ generates all possible one-way tables with the observed total sample size and number of categories. For each possible table, PROC FREQ compares its chi-square value with the value for the observed table. If the table's chi-square value is greater than or equal to the observed chi-square, PROC FREQ increments the exact *p*-value by the probability of that table, which is calculated under the null hypothesis using the multinomial frequency distribution. By default, the null hypothesis states that all categories have equal proportions. If you specify null hypothesis proportions or frequencies using the TESTP= or TESTF= option in the TABLES statement, then PROC FREQ calculates the exact chi-square test based on that null hypothesis.

For binomial proportions in one-way tables, PROC FREQ computes exact confidence limits using the ***F*** distribution method given in Collett (1991) and also described by Leemis and Trivedi (1996). PROC FREQ computes the exact test for a binomial proportion $H_0: \; p = p_0$ by summing binomial probabilities over all alternatives. See "Binomial Proportion" on page 560 for details. By default PROC FREQ uses $p_0 = 0.5$ as the null hypothesis proportion. Alternatively, you can specify the null hypothesis proportion with the P= option in the TABLES statement.

Definition of *p*-Values

For several tests in PROC FREQ, the test statistic is nonnegative, and large values of the test statistic indicate a departure from the null hypothesis. Such tests include the Pearson chi-square, the likelihood-ratio chi-square, the Mantel-Haenszel chi-square, Fisher's exact test for tables larger than 2×2 tables, McNemar's test, and the one-way goodness-of-fit test. The exact *p*-value for these nondirectional tests is the sum of probabilities for those tables having a test statistic greater than or equal to the value of the observed test statistic.

There are other tests where it may be appropriate to test against either a one-sided or a two-sided alternative hypothesis. For example, when you test the null hypothesis that the true parameter value equals zero $(T = 0)$, the alternative of interest may be one-sided $(T < 0, \text{ or } T > 0)$ or two-sided $(T \neq 0)$. Such tests include the Pearson correlation coefficient, Spearman correlation coefficient, Jonckheere-Terpstra test, Cochran-Armitage test for trend, simple kappa coefficient, and weighted kappa coefficient. For these tests, PROC FREQ computes the right-sided *p*-value when the observed value of the test statistic is greater than its expected value. The right-sided *p*-value is the sum of probabilities for those tables having a test statistic greater than or equal to the observed test statistic. Otherwise, when the test statistic is less than or equal to its expected value, PROC FREQ computes the left-sided *p*-value. The left-sided *p*-value is the sum of probabilities for those tables having a test statistic less than or equal to the one observed. The one-sided *p*-value P_1 can be expressed as

$$\begin{array}{ll} P_1 = \text{Prob (Test Statistic } \geq t) & \text{if } t > E_0(T) \\ P_1 = \text{Prob (Test Statistic} \leq t) & \text{if } t \leq E_0(T) \end{array}$$

where t is the observed value of the test statistic, and $E_0(T)$ is the expected value of the test statistic under the null hypothesis. PROC FREQ computes the two-sided *p*-value as the sum of the one-sided *p*-value and the corresponding area in the opposite tail of the distribution of the statistic, equidistant from the expected value. The two-sided *p*-value P_2 can be expressed as

$$P_2 = \text{Prob } (|\text{Test Statistic} - E_0(T)| \geq |t - E_0(T)|)$$

If you specify the POINT option in the EXACT statement, PROC FREQ also displays exact point probabilities for the test statistics. The exact point probability is the exact probability that the test statistic equals the observed value.

Computational Resources

PROC FREQ uses relatively fast and efficient algorithms for exact computations. These recently developed algorithms, together with improvements in computer power, make it feasible now to perform exact computations for data sets where previously only asymptotic methods could be applied. Nevertheless, there are still large problems that may require a prohibitive amount of time and memory for exact computations, depending on the speed and memory available on your computer. For large problems, consider whether exact methods are really needed or whether asymptotic methods might give results quite close to the exact results, while requiring much less computer time and memory. When asymptotic methods may not be sufficient for such large problems, consider using Monte Carlo estimation of exact *p*-values, as described in "Monte Carlo Estimation" on page 584.

A formula does not exist that can determine in advance how much time or memory that PROC FREQ needs to compute an exact *p*-value for a certain problem. The time and memory requirements depend on several factors which include the test that is performed, the total sample size, the number of rows and columns, and the specific arrangement of the observations into table cells. Generally, larger problems (in terms of total sample size, number of rows, and number of columns) tend to require more time and memory. Additionally, for a fixed total sample size, time and memory requirements tend to increase as the number of rows and columns increases, because this corresponds to an increase in the number of tables in the reference set. Also for a fixed sample size, time and memory requirements increase as the marginal row and column totals become more homogeneous. Refer to Agresti et al. (1992) and Gail and Mantel (1977).

At any time while PROC FREQ computes exact p-values, you can terminate the computations by pressing the system interrupt key sequence (refer to the *SAS Companion* for your operating environment) and choosing to stop computations. After you terminate exact computations, PROC FREQ completes all other remaining tasks that the procedure specifies. The procedure produces the requested output, reporting missing values for any exact p-values that were not computed by the time of termination.

You can also use the MAXTIME= option in the EXACT statement to limit the amount of clock time PROC FREQ uses for exact computations. You specify a MAXTIME= value that is the maximum amount of time (in seconds) that PROC FREQ can use to compute an exact p-value. If PROC FREQ does not finish computing an exact p-value within that time, it terminates the computation and completes all other remaining tasks.

Monte Carlo Estimation

If you specify the option MC in the EXACT statement, PROC FREQ computes Monte Carlo estimates of the exact p-values, instead of directly computing the exact p-values. Monte Carlo estimation can be useful for large problems that require a great amount of time and memory for exact computations, but for which asymptotic approximations may not be sufficient. To describe the precision of each Monte Carlo estimate, PROC FREQ provides the asymptotic standard error and $(1-\alpha) \times 100$ percent confidence limits. The confidence level α is determined by the ALPHA= option in the EXACT statement, which by default equals .01 and produces 99 percent confidence limits. The N= option in the EXACT statement specifies the number of samples that PROC FREQ uses for Monte Carlo estimation, and the default is 10000 samples. You can specify a larger value for N= to improve the precision of the Monte Carlo estimates. Because larger values of N= generate more samples, the computation time increases. Alternatively, you can specify a smaller value of N= to reduce the computation time.

To compute a Monte Carlo estimate of an exact p-value, PROC FREQ generates a random sample of tables with the same total sample size, row totals, and column totals as the observed table. PROC FREQ uses the algorithm of Agresti et al. (1979), which generates tables in proportion to their hypergeometric probabilities, conditional on the marginal frequencies. For each sample table, PROC FREQ computes the value of the test statistic and compares it to the value for the observed table. When estimating a right-sided p-value, PROC FREQ counts all sample tables for which the test statistic is greater than or equal to the observed test statistic. Then the p-value estimate equals the number of these tables divided by the total number of tables sampled.

$$
\begin{aligned}
&\hat{P}_{MC} = M/N \\
&M = \text{number of samples with } (\text{Test Statistic} \geq t) \\
&N \;= \text{number of samples} \\
&T = \text{observed Test Statistic}
\end{aligned}
$$

PROC FREQ computes left-sided and two-sided p-value estimates similarly. For left-sided p-values, PROC FREQ evaluates whether the test statistic for each sampled table is less than or equal to the observed test statistic. For two-sided p-values, PROC FREQ examines the sample test statistics according to the expression for P_2 given in "Definition of p-Values" on page 582. The variable M above is a binomially distributed variable with N trials and success probability p. It follows that the asymptotic standard error of the Monte Carlo estimate is

$$se\left(\hat{P}_{MC}\right) = \sqrt{\hat{P}_{MC}\left(1 - \hat{P}_{MC}\right) / (N-1)}$$

PROC FREQ constructs asymptotic confidence limits for the *p*-values according to

$$\hat{P}_{MC} \pm z_{\alpha/2} \cdot se\left(\hat{P}_{MC}\right)$$

where $z_{\alpha/2}$ is the $100\,(1-\alpha/2)$ percentile of the standard normal distribution, and the confidence level α is determined by the ALPHA= option in the EXACT statement.

When the Monte Carlo estimate $\hat{P}_{MC}$ equals 0, then PROC FREQ computes the confidence limits for the *p*-value as

$$\left(0, 1 - \alpha^{(1/N)}\right)$$

When the Monte Carlo estimate $\hat{P}_{MC}$ equals 1, then PROC FREQ computes the confidence limits as

$$\left(\alpha^{(1/N)}, 1\right)$$

Results: FREQ Procedure

Missing Values

By default, PROC FREQ excludes missing values before it constructs the frequency and crosstabulation tables. PROC FREQ also excludes missing values before computing statistics. However, PROC FREQ displays the total frequency of observations with missing values below each table. The following options in the TABLES statement change how PROC FREQ handles missing values:

MISSPRINT
: includes missing value frequencies in frequency or crosstabulation tables.

MISSING
: includes missing values in percentage and statistical calculations.

The OUT= option in the TABLES statement includes an observation in the output data set that contains the frequency of missing values. The NMISS keyword in the OUTPUT statement creates a variable in the output data set that contains the number of missing values.

Output 23.4 on page 585 shows three ways that PROC FREQ handles missing values. The first table uses the default method; the second table uses MISSPRINT; and the third table uses MISSING.

Output 23.4 Missing Values in Frequency Tables

```
                     *** Default ***

                   The FREQ Procedure

                                   Cumulative    Cumulative
A    Frequency     Percent     Frequency      Percent
------------------------------------------------------
1           2       50.00             2        50.00
2           2       50.00             4       100.00

                 Frequency Missing = 2

                *** MISSPRINT Option ***

                   The FREQ Procedure

                                   Cumulative    Cumulative
A    Frequency     Percent     Frequency      Percent
------------------------------------------------------
.           2         .               .            .
1           2       50.00             2        50.00
2           2       50.00             4       100.00

                 Frequency Missing = 2

                *** MISSING Option ***

                   The FREQ Procedure

                                   Cumulative    Cumulative
A    Frequency     Percent     Frequency      Percent
------------------------------------------------------
.           2       33.33             2        33.33
1           2       33.33             4        66.67
2           2       33.33             6       100.00
```

When a combination of variable values for a crosstabulation is missing, PROC FREQ assigns zero to the frequency count for the table cell. By default, PROC FREQ omits missing combinations in list format and in the output data set that is created with a TABLES statement. To include the missing combinations, use SPARSE with LIST or OUT= in the TABLES statement.

PROC FREQ treats missing BY variable values like any other BY variable value. The missing values form a separate BY group. When the value of a WEIGHT variable is missing, PROC FREQ excludes the observation from the analysis.

ODS Table Names

PROC FREQ assigns a name to each table it creates. You can use these names to reference the table when using the Output Delivery System (ODS) to select tables and create output data sets. For more information, see *SAS Output Delivery System User's Guide*.

Table 23.5 ODS Tables Produced with the TABLES Statement

ODS Table Name	Description	Option
BinomialProp	Binomial proportion	BINOMIAL (one-way tables)
BinomialPropTest	Binomial proportion test	BINOMIAL (one-way tables)
BreslowDayTest	Breslow-Day test	CMH (hx2x2tables)
CMH	Cochran-Mantel-Haenszel statistics	CMH
ChiSq	Chi-square tests and measures	CHISQ
CochransQ	Cochran's Q	AGREE (hx2x2 tables)
ColScores	Column scores	SCOROUT
CommonRelRisks	Common relative risks	CMH (hx2x2 tables)
CrossTabFreqs	Crosstabulation table	(n-way table request, n > 1)
EqualKappaTest	Test for equal simple kappas	AGREE (hx2x2 tables)
EqualKappaTests	Tests for equal kappas	AGREE (hxrx r tables, r>2)
FishersExact	Fisher's exact test	FISHER or EXACT CHISQ (2x2 tables)
JTTest	Jonckheere-Terpstra test	JT
KappaStatistics	Kappa statistics	AGREE (rxr tables, r > 2, and no TEST or EXACT requests for kappas)
KappaWeights	Kappa weights	AGREE and PRINTKWT
List	List frequencies	LIST
McNemarsTest	McNemar's test	AGREE (2x2 tables)
Measures	Measures of association	MEASURES
OneWayChiSq	One-way chi-square goodness-of-fit test	CHISQ (one-way tables)
OneWayFreqs	One-way frequencies	(one-way table request)
OverallKappa	Overall simple kappa coefficient	AGREE (hx2x2 tables)
OverallKappas	Overall kappa coefficients	AGREE (hxrxr tables, r>2)
RelativeRisks	Relative risk estimates	RELRISK or MEASURES (2x2 tables)
RiskDiffCol1	Column 1 risk estimates	RISKDIFF (2x2 tables)
RiskDiffCol2	Column 2 risk estimates	RISKDIFF (2x2 tables)
RowScores	Row scores	SCOROUT

SimpleKappaTest	Simple kappa test	AGREE (2x2 tables), AGREE rxr tables, r>2)
SymmetryTest	Test of symmetry	AGREE
TrendTest	Cochran-Armitage test for trend	TREND
WeightedKappa	Weighted kappa coefficient	AGREE (rxr tables, r>2)

Table 23.6 ODS Tables Produced with the EXACT Statement

ODS Table Name	Description	Option
FishersExact	Fisher's exact test	FISHER
FishersExactMC	Monte Carlo estimates for Fisher's exact test	FISHER / MC
JTTestMC	Monte Carlo estimates for the JT exact test	JT / MC
LRChiSq	Likelihood-ratio chi-square exact test	LRCHI
LRChiSqMC	Monte Carlo estimate for the likelihood-ratio chi-square exact test	LRCHI / MC
MHChiSq	Mantel-Haenszel chi-square exact test	MHCHI
MHChiSqMC	Monte Carlo estimate for the Mantel-Haenszel chi-square exact test	MHCHI / MC
OddsRatioCL	Exact confidence limits for the odds ratio	OR
OneWayChiSqMC	Monte Carlo estimates for the one-way chi-square exact test	CHISQ / MC (one-way tables)
PearsonChiSq	Pearson chi-square exact test	PCHI
PearsonChiSqMC	Monte Carlo estimate for the Pearson chi-square exact test	PCHI / MC
PearsonCorr	Pearson correlation coefficient	PCORR
PearsonCorrMC	Monte Carlo estimates for the Pearson correlation exact test	PCORR / MC
PearsonCorrTest	Pearson correlation test	PCORR
SimpleKappa	Simple kappa coefficient	KAPPA
SimpleKappaMC	Monte Carlo estimates for the simple kappa exact test	KAPPA/ MC
SimpleKappaTest	Simple kappa test	KAPPA or WTKAP
SpearmanCorr	Spearman correlation coefficient	SCORR
SpearmanCorrMC	Monte Carlo estimates for the Spearman correlation exact test	SCORR / MC
SpearmanCorrTest	Spearman correlation test	SCORR
TrendTestMC	Monte Carlo estimates for the trend exact test	TREND / MC
WeightedKappa	Weighted kappa coefficient	WTKAP KAPPA

WeightedKappaMC	Monte Carlo estimates for the weighted kappa exact test	WTKAP / MC
WeightedKappaTest	Weighted kappa test	WTKAP

Table 23.7 ODS Tables Produced with the TEST Statement

ODS Table Name	Description	Option
Gamma	Gamma	GAMMA
GammaTest	Gamma test	GAMMA
PearsonCorr	Pearson correlation coefficient	PCORR
PearsonCorrTest	Pearson correlation test	PCORR
SimpleKappa	Simple kappa coefficient	KAPPA
SimpleKappaTest	Simple kappa test	KAPPA or WTKAP
SomersDCR	Somers' D(C | R)	SMDCR
SomersDCRTest	Somers' D(C | R) test	SMDCR
SomersDRC	Somers' D(R | C)	SMDRC
SomersDRCTest	Somers' D(R | C) test	SMDRC
SpearmanCorr	Spearman correlation coefficient	SCORR
SpearmanCorrTest	Spearman correlation test	SCORR
TauB	Kendall's tau-b	KENTB
TauBTest	Kendall's tau-b test	KENTB
TauC	Stuart's tau-c	STUTC
TauCTest	Stuart's tau-c test	STUTC
WeightedKappa	Weighted kappa coefficient	WTKAP KAPPA
WeightedKappaTest	Weighted kappa test	WTKAP

Procedure Output

By default, a one-way table lists the variable name, variable values, frequency counts, percentages, cumulative frequency counts, cumulative percentages, and the number of missing values. Unless you use LIST in the TABLES statement, a two-way table appears as a crosstabulation table. An n-way table appears as multiple crosstabulation tables with one table for each combination of values for the stratification variables. By default, each cell of a crosstabulation table lists the frequency count, percentage of the total frequency count, row percentage, and column percentage.

Use the following TABLES statement options to report additional information for each table cell:

CELLCHI2
: includes the cell's contribution to the total chi-square statistic

CUMCOL
: includes the cumulative column percentage of the cell

DEVIATION

includes the deviation of the cell frequency from the expected value

EXPECTED
: includes the expected cell frequency under the hypothesis of independence.

You can also use the SCOROUT option to display the type of score, row score, and column score for two-way tables.

By default, PROC FREQ displays the next one-way frequency table on the current page when there is enough space to display the entire table. If you use COMPRESS in the PROC FREQ statement, the next one-way table starts to display on the current page even when the entire table will not fit. If you use PAGE in the PROC FREQ statement, each frequency or crosstabulation table always displays on a separate page.

Suppressing the Displayed Output

The NOPRINT option in the PROC FREQ statement and NOPRINT, NOCOL, NOCUM, NOFREQ, NOPERCENT, and NOROW in the TABLES statement suppress displayed output. Use NOPRINT in the PROC FREQ statement to suppress all displayed output as well as the Output Delivery System. Use NOPRINT in the TABLES statement to suppress frequency and crosstabulation tables but still display the requested statistics. Use NOCOL, NOCUM, NOFREQ, NOPERCENT, and NOROW to suppress various frequencies and percentages in the frequency and crosstabulation tables.

CAUTION:

Multiway tables can generate a great deal of displayed output. For example, if the variables A, B, C, D, and E each have ten levels, the table request A*B*C*D*E may generate 1000 or more pages of output. If you are primarily interested in the tests and measures of association, use NOPRINT in the TABLES statement to suppress the tables but display the statistics. Or use NOPRINT in the PROC FREQ statement to suppress all displayed output, and use the OUTPUT statement to store the statistics in an output data set. If you are interested in frequency counts and percentages use LIST in the TABLES statement. △

Output Data Sets

PROC FREQ produces two types of output data sets that you can use with other statistical and reporting procedures. These data sets are produced as follows:

TABLES statement, OUT= option
: creates an output data set that contains frequency or crosstabulation table counts and percentages.

OUTPUT statement
: creates an output data set that contains statistics.

PROC FREQ does not display the output data set. Use PROC PRINT, PROC REPORT, or any other SAS reporting tool to display the output data set.

Contents of the TABLES Statement Output Data Set

The OUT= option in the TABLES statement creates an output data set that contains one observation for each combination of the variable values in the last table request. By default, each observation contains the frequency and percentage for each combination of variable values. When the input data set contains missing values, the output data set contains an observation with the frequency of missing values. The output data set includes the following variables:

- □ BY variables
- □ table request variables, such as A, B, C, and D in the table request A*B*C*D
- □ COUNT variable containing the cell frequency
- □ PERCENT variable containing the cell percentage.

If you use OUTEXPECT and OUTPCT, the output data set also contains expected frequencies and row, column, and table percentages, respectively. The additional variables are

- □ EXPECTED variable containing the expected frequency
- □ PCT_TABL variable containing the percentage of two-way table frequency, for n-way tables where $n > 2$
- □ PCT_ROW variable containing the percentage of row frequency
- □ PCT_COL variable containing the percentage of column frequency.

If you use OUTCUM, the output data set also contains the cumulative frequency and the cumulative percent for one-way tables in the output data set . The additional variables are

- □ CUM_FREQ variable containing the cumulative frequency for each level of the analysis variable
- □ CUM_PCT variable containing the cumulative percent for each level.

When you submit the following statements

```
proc freq;
   tables a a*b / out=d;
run;
```

the output data set D contains frequencies and percentages for the last table request, A*B. If A has two levels (1 and 2), B has three levels (1, 2, and 3), and no table cell count is zero or missing, the output data set D includes six observations, one for each combination of A and B. The first observation corresponds to A=1 and B=1; the second observation corresponds to A=1 and B=2; and so on. The data set also includes the variables COUNT and PERCENT. The value of COUNT is the number of observations that have the given combination of A and B values. The value of PERCENT is the percent of the total number of observations having that A and B combination.

When PROC FREQ combines different variable values into the same formatted level, the output data set contains the smallest internal value for the formatted level. For example, suppose a variable X has the values 1.1, 1.4, 1.7, 2.1, and 2.3. When you submit the statement

```
   format x 1.;
```

in a PROC FREQ step, the formatted levels listed in the frequency table for X are 1 and 2. If you create an output data set with the frequency counts, the internal values of X are 1.1 and 1.7. To report the internal values of X when you display the output data set, use a format of 3.1 with X.

Contents of the OUTPUT Statement Output Data Set

The OUTPUT statement creates a SAS data set that contains the statistics that PROC FREQ computes for the last table request. You specify which statistics to store in the output data set. There is an observation with the specified statistics for each stratum or two-way table. If PROC FREQ computes summary statistics for a stratified table, the output data set also contains a summary observation for these statistics. Additionally, you can output statistics for one-way tables, such as chi-square or binomial proportion statistics. If you use a BY statement, the output data set contains observations for each BY group.

The output data set can include the following variables:

- □ BY variables
- □ variables that identify the stratum such as A and B in the table request A*B*C*D
- □ variables that contain the specified statistics.

The output data set also includes variables with the *p*-value and degrees of freedom, asymptotic standard error (ASE), or confidence limits when PROC FREQ computes these values for a specified statistic.

The variable names for the specified statistics in the output data set are the names of the keywords that are enclosed in underscores. PROC FREQ forms variable names for the corresponding *p*-values, degrees of freedom, or confidence limits by combining the name of the keyword with one of the following prefixes

DF_	degrees of freedom
E_	asymptotic standard error (ASE)
E0_	asymptotic standard error under the null hypothesis
L_	lower confidence limit
P_	*p*-value
P2_	two-sided *p*-value
PL_	left-sided *p*-value
PR_	right-sided *p*-value
U_	upper confidence limit
XP_	exact *p*-value
XP2_	exact two-sided *p*-value
XPR_	exact right-sided *p*-value
XPL_	exact left-sided *p*-value
XPT_	exact point probability
XL_	exact lower confidence limit
XU_	exact upper confidence limit
Z_	standardized value

If the length of the prefix plus the statistic keyword exceeds eight characters, PROC FREQ truncates the keyword so that the name of the new variable is eight characters long.

Examples: FREQ Procedure

Example 1: Creating an Output Data Set with Table Cell Frequencies

Procedure features:

TABLES statement,
  multiple requests
TABLES statement options:
  OUT=
  OUTEXPECT
  SPARSE
WEIGHT statement

Other features:
PRINT procedure

This example

- □ creates two frequency tables and a crosstabulation table using existing cell counts
- □ creates an output data set for the last table request with frequencies, percentages, and expected cell frequencies
- □ includes zero cell counts in the output data set
- □ displays the output data set.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the COLOR data set. This data set contains information on eye and hair color of children from two regions of Europe. The data are recorded as cell counts instead of as one observation per child. Count contains the frequencies of the 15 eye and hair color combinations for each region. Missing eye and hair color combinations are excluded from the data set.

```
data color;
   input Region Eyes $ Hair $ Count @@;
   label eyes='Eye Color'
         hair='Hair Color'
         region='Geographic Region';
   datalines;
1 blue  fair    23  1 blue  red      7  1 blue  medium  24
1 blue  dark    11  1 green fair    19  1 green red      7
1 green medium  18  1 green dark    14  1 brown fair    34
1 brown red      5  1 brown medium  41  1 brown dark    40
1 brown black    3  2 blue  fair    46  2 blue  red     21
2 blue  medium  44  2 blue  dark    40  2 blue  black    6
2 green fair    50  2 green red     31  2 green medium  37
2 green dark    23  2 brown fair    56  2 brown red     42
2 brown medium  53  2 brown dark    54  2 brown black   13
;
```

Generate the frequency tables and a crosstabulation table from existing cell counts. The WEIGHT statement uses Count to weight the observations in the Color data set.

```
proc freq data=color;
   weight count;
```

Specify the variables to use to create the tables. Create the output data set FREQCNT that will contain the table frequencies and expected cell frequencies for the last table request. The TABLES statement requests three tables: Eyes and Hair frequencies and an Eyes by Hair crosstabulation. OUT= creates the FREQCNT data set that contains crosstabulation table frequencies. OUTEXPECT stores expected cell frequencies and SPARSE stores zero cell counts in FREQCNT.

```
   tables eyes hair eyes*hair/out=freqcnt outexpect
                                             sparse;
```

Specify the title.

```
   title 'Eye and Hair Color of European Children';
run;
```

Print the data set. PROC PRINT displays the FREQCNT data set. The TITLE statement specifies a title.

```
proc print data=freqcnt noobs;
   title2 'Output Data Set from PROC FREQ';
run;
```

Output

By default, PROC FREQ lists the variable values in alphabetical order. Because Eyes*Hair requests a crosstabulation table, the table rows are eye color and the table columns are hair color. A zero cell count for green eyes and black hair indicates that this eyes and hair combination does not occur in the data.

```
                  Eye and Hair Color of European Children                    1

                             The FREQ Procedure

                                 Eye Color

                                              Cumulative    Cumulative
           Eyes     Frequency     Percent      Frequency      Percent
           ---------------------------------------------------------
           blue          222       29.13            222        29.13
           brown         341       44.75            563        73.88
           green         199       26.12            762       100.00


                                 Hair Color

                                              Cumulative    Cumulative
          Hair      Frequency     Percent      Frequency      Percent
          ----------------------------------------------------------
          black          22        2.89             22         2.89
          dark          182       23.88            204        26.77
          fair          228       29.92            432        56.69
          medium        217       28.48            649        85.17
          red           113       14.83            762       100.00


                            Table of Eyes by Hair

         Eyes(Eye Color)     Hair(Hair Color)

         Frequency|
         Percent  |
         Row Pct  |
         Col Pct  |black   |dark    |fair    |medium  |red     |  Total
         ---------+--------+--------+--------+--------+--------+
         blue     |      6 |     51 |     69 |     68 |     28 |    222
                  |   0.79 |   6.69 |   9.06 |   8.92 |   3.67 |  29.13
                  |   2.70 |  22.97 |  31.08 |  30.63 |  12.61 |
                  |  27.27 |  28.02 |  30.26 |  31.34 |  24.78 |
         ---------+--------+--------+--------+--------+--------+
         brown    |     16 |     94 |     90 |     94 |     47 |    341
                  |   2.10 |  12.34 |  11.81 |  12.34 |   6.17 |  44.75
                  |   4.69 |  27.57 |  26.39 |  27.57 |  13.78 |
                  |  72.73 |  51.65 |  39.47 |  43.32 |  41.59 |
         ---------+--------+--------+--------+--------+--------+
         green    |      0 |     37 |     69 |     55 |     38 |    199
                  |   0.00 |   4.86 |   9.06 |   7.22 |   4.99 |  26.12
                  |   0.00 |  18.59 |  34.67 |  27.64 |  19.10 |
                  |   0.00 |  20.33 |  30.26 |  25.35 |  33.63 |
         ---------+--------+--------+--------+--------+--------+
         Total          22      182      228      217      113      762
                      2.89    23.88    29.92    28.48    14.83   100.00
```

The output data set contains frequency counts and percentages for the last table. The data set also includes an observation for the zero cell count and a variable with the expected cell frequency for each table cell.

```
                Eye and Hair Color of European Children                2
                     Output Data Set from PROC FREQ

            Eyes     Hair      COUNT    EXPECTED    PERCENT

            blue     black        6       6.409      0.7874
            blue     dark        51      53.024      6.6929
            blue     fair        69      66.425      9.0551
            blue     medium      68      63.220      8.9239
            blue     red         28      32.921      3.6745
            brown    black       16       9.845      2.0997
            brown    dark        94      81.446     12.3360
            brown    fair        90     102.031     11.8110
            brown    medium      94      97.109     12.3360
            brown    red         47      50.568      6.1680
            green    black        0       5.745      0.0000
            green    dark        37      47.530      4.8556
            green    fair        69      59.543      9.0551
            green    medium      55      56.671      7.2178
            green    red         38      29.510      4.9869
```

Example 2: Computing Chi-Square Tests for One-Way Frequency Tables

Procedure features:

PROC FREQ statement option:

ORDER=

BY statement

TABLES statement options:

NOCUM
TESTP=

WEIGHT statement

Other features:

SORT procedure

Data set: COLOR on page 593

This example

- sorts a data set by geographic region
- creates a one-way frequency table for each BY group
- orders the values of the frequency table by their appearance in the input data set
- suppresses the cumulative frequencies and percentages
- computes a chi-square goodness-of-fit test for specified proportions.

The chi-square goodness-of-fit test examines whether the children's hair color has a specified multinomial distribution for two regions. The hypothesized distribution for hair color is 30 percent fair, 12 percent red, 30 percent medium, 25 percent dark, and 3 percent black.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Sort the Color data set. PROC SORT sorts the observations by the variable Region.

```
proc sort data=color;
   by region;
run;
```

Generate the frequency table in a specified order from existing cell counts. ORDER=DATA orders the frequency table values (hair color) by their order in the data set. The WEIGHT statement uses Count to weight the observations.

```
proc freq data=color order=data;
   weight count;
```

Specify the variable to use to create the frequency tables. Compute a chi-square goodness-of-fit test for specified proportions. The TABLES statement requests a frequency table for hair color. NOCUM suppresses the cumulative frequencies and percentages. TESTP= specifies hypothesized percentages for the chi-square test. The number of percentages equals the number of table levels and the percentages sum to 100.

```
   tables hair/nocum testp=(30 12 30 25 3);
```

Create the frequency tables separately for each BY group. The BY statement produces a separate table for each BY group and displays a heading above each one.

```
   by region;
```

Specify a title for the report. The TITLE statement specifies a title.

```
   title 'Hair Color of European Children';
run;
```

Output

The frequency table lists the variable values (hair color) in the order that they appear in the data set. The last column lists the hypothesized percentages for the chi-square test. Always check that you have ordered the TESTP= percentages to correctly match the order of the variable levels.

PROC FREQ computes a chi-square statistic for each region. The chi-square statistic is significant at the .05 level for region 2 (**p**≤.05) but not for region 1, indicating a significant departure from the hypothesized percentages in region 2.

```
                    Hair Color of European Children                      1

----------------------------- Geographic Region=1 ------------------------------

                          The FREQ Procedure

                             Hair Color

                                                       Test
                 Hair      Frequency     Percent     Percent
                 -------------------------------------------
                 fair            76       30.89       30.00
                 red             19        7.72       12.00
                 medium          83       33.74       30.00
                 dark            65       26.42       25.00
                 black            3        1.22        3.00


                           Chi-Square Test
                      for Specified Proportions
                      -------------------------
                      Chi-Square        7.7602
                      DF                     4
                      Pr > ChiSq        0.1008

                          Sample Size = 246
```

```
                    Hair Color of European Children                      2

----------------------------- Geographic Region=2 ------------------------------

                          The FREQ Procedure

                             Hair Color

                                                       Test
                 Hair      Frequency     Percent     Percent
                 -------------------------------------------
                 fair           152       29.46       30.00
                 red             94       18.22       12.00
                 medium         134       25.97       30.00
                 dark           117       22.67       25.00
                 black           19        3.68        3.00


                           Chi-Square Test
                      for Specified Proportions
                      -------------------------
                      Chi-Square       21.3824
                      DF                     4
                      Pr > ChiSq        0.0003

                          Sample Size = 516
```

Example 3: Computing Binomial Proportions for One-Way Frequency Tables

Procedure features:
PROC FREQ statement option:
ORDER=
TABLES statement options:
ALPHA=
BINOMIAL
WEIGHT statement

Data set: COLOR on page 593

This example

- creates a one-way frequency table using existing cell counts
- orders the values of the frequency table by their frequency in the input data set
- computes the binomial proportion and the corresponding test statistic
- specifies the null hypothesis proportion for the asymptotic test of the binomial proportion
- specifies the confidence level for the confidence limits.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=40;
```

Generate the frequency table in a specified order from existing cell counts. ORDER=FREQ orders the frequency table values by their frequency in the data set. The WEIGHT statement uses Count to weight the observations.

```
proc freq data=color order=freq;
   weight count;
```

Specify the variable to use to create the frequency table. Compute the binomial proportion and the corresponding test statistics. Specify the confidence level for confidence limits. The TABLES statement requests a frequency table for eye color. BINOMIAL computes the binomial proportion and confidence limits, and also tests the hypothesis that the proportion for the first eye color level equals 0.5. ALPHA= specifies 90 percent confidence limits.

```
   tables eyes/binomial alpha=.1;
```

Specify the variable to use to create the frequency table. Compute the binomial proportion and the corresponding test statistics. Specify the null hypothesis proportion value for the test. The TABLES statement requests a frequency table for hair color. BINOMIAL computes the binomial proportion and confidence limits, and also tests the hypothesis that the proportion for the first hair color level equals 0.28.

```
   tables hair/binomial(p=.28);
```

Specify a title for the report. The TITLE statement specifies a title.

```
   title 'Hair and Eye Color of European Children';
run;
```

Output

The frequency table lists the variable values in the order of the descending frequency count. PROC FREQ computes the binomial proportion for the first variable level. The report includes the asymptotic standard error (ASE), and asymptotic and exact confidence limits for the binomial proportion. The specified confidence level of .1 results in 90 percent confidence limits.

Because the value of **Z** is less than zero for eye color, PROC FREQ computes a left-sided **p**-value. The small **p**-value supports the alternative hypothesis that the true value of the proportion of children with brown eyes is less than 50 percent.

```
                     Hair and Eye Color of European Children                   1

                                The FREQ Procedure

                                    Eye Color

                                                 Cumulative    Cumulative
            Eyes     Frequency      Percent     Frequency       Percent
            ---------------------------------------------------------
            brown         341       44.75            341         44.75
            blue          222       29.13            563         73.88
            green         199       26.12            762        100.00


                               Binomial Proportion
                                 for Eyes = brown
                        --------------------------------
                        Proportion                0.4475
                        ASE                       0.0180
                        90% Lower Conf Limit      0.4179
                        90% Upper Conf Limit      0.4771

                        Exact Conf Limits
                        90% Lower Conf Limit      0.4174
                        90% Upper Conf Limit      0.4779

                          Test of H0: Proportion = 0.5

                        ASE under H0              0.0181
                        Z                        -2.8981
                        One-sided Pr <  Z         0.0019
                        Two-sided Pr > |Z|        0.0038
```

Because the value of **Z** is greater than zero for hair color, PROC FREQ computes a right-sided **p**-value. The large **p**-value provides insufficient evidence to reject the null hypothesis that the proportion of children with fair hair equals 28 percent.

```
                 Hair and Eye Color of European Children                    2

                            The FREQ Procedure

                               Hair Color

                                              Cumulative    Cumulative
          Hair      Frequency     Percent     Frequency      Percent
          -----------------------------------------------------------
          fair           228       29.92           228        29.92
          medium         217       28.48           445        58.40
          dark           182       23.88           627        82.28
          red            113       14.83           740        97.11
          black           22        2.89           762       100.00


                           Binomial Proportion
                             for Hair = fair
                        --------------------------------
                        Proportion                0.2992
                        ASE                       0.0166
                        95% Lower Conf Limit      0.2667
                        95% Upper Conf Limit      0.3317

                        Exact Conf Limits
                        95% Lower Conf Limit      0.2669
                        95% Upper Conf Limit      0.3331

                         Test of H0: Proportion = 0.28

                        ASE under H0              0.0163
                        Z                         1.1812
                        One-sided Pr >  Z         0.1188
                        Two-sided Pr > |Z|        0.2375
```

Example 4: Analyzing a 2×2 Contingency Table

Procedure features:

- PROC FREQ statement option:
  - ORDER=
- EXACT statement
- TABLES statement options:
  - CHISQ
  - RELRISK
- WEIGHT statement

Other features:

- FORMAT procedure
- SORT procedure

This example

- □ creates a two-way contingency table using existing cell counts
- □ sorts the data in descending order so that the first table cell contains the frequency of positive exposure and positive response
- □ computes chi-square tests, exact Pearson chi-square test, and Fisher's exact test to compare the probability of coronary heart disease for two types of diet
- □ computes estimates of the relative risk and 95 percent exact confidence limits for the odds ratio.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=84 pagesize=64;
```

Assign a character string format to a numeric value. PROC FORMAT creates user-written formats to identify the type of exposure and response with character values.

```
proc format;
   value expfmt 1='High Cholesterol Diet'
                0='Low Cholesterol  Diet';
   value rspfmt 1='Yes'
                0='No';
run;
```

Create the FATCOMP data set. This data set contains hypothetical data for a case-control study of a high-fat diet and the risk of coronary heart disease. The data are recorded as cell counts instead of as one observation per subject. The variable Count contains the frequencies for each exposure and response combination.

```
data fatcomp;
   input Exposure Response Count;
   label response='Heart Disease';
   datalines;
0 0 6
0 1 2
1 0 4
1 1 11
;
```

Sort the FATCOMP data set. PROC SORT sorts the observations in descending order by the variables Exposure and Response.

```
proc sort data=fatcomp;
   by descending exposure descending response;
run;
```

Generate the cross-tabulation table in a specified order from existing cell counts. ORDER=DATA orders the contingency table values by their order in the data set. The WEIGHT statement uses Count to weight the observations.

```
proc freq data=fatcomp order=data;
   weight count;
```

Specify the variables to use to create the contingency tables. Compute chi-square tests, the measures of association based on chi-square, and the measures of relative risk. The TABLES statement requests a two-way table. CHISQ requests chi-square tests. RELRISK requests relative risk measures.

```
   tables exposure*response / chisq relrisk;
```

Request exact tests or confidence limits for the specified statistics. The EXACT statement requests the exact Pearson chi-square test and exact confidence limits for the odds ratio.

```
   exact pchi or;
```

Assign the formats to the variables and specify a title for the report. The FORMAT statement assigns formats to the variables Exposure and Response. The TITLE statement specifies a title.

```
   format exposure expfmt. response rspfmt.;
   title 'Case-Control Study of High Fat/Cholesterol Diet';
run;
```

Output

The contingency table lists the variable values so that the first table cell contains the frequency of positive exposure and response. PROC FREQ does not truncate the formatted variable values that are more than 16 characters but uses multiple lines to show Exposure levels.

PROC FREQ displays a warning message that sample size requirements may not be met for the asymptotic chi-square tests. The exact tests are appropriate when sample size is small.

Because the alternative hypothesis for this analysis states that coronary heart disease was more likely to be associated with a high-fat diet, a one-sided test is needed. Fisher's exact test (right-sided) tests that the probability of heart disease in the high-fat group exceeds the probability of heart disease in the low-fat group.

The odds ratio, which provides an estimate of the relative risk when an event is rare, indicates that the odds of heart disease are 8.25 times higher in the high-fat-diet group. However, the wide confidence limits indicate that this estimate has low precision.

```
                Case-Control Study of High Fat/Cholesterol Diet                1

                               The FREQ Procedure

                         Table of Exposure by Response

                  Exposure          Response(Heart Disease)

                  Frequency        |
                  Percent          |
                  Row Pct          |
                  Col Pct          |Yes     |No      |  Total
                  -----------------+--------+--------+
                  High Cholesterol |     11 |      4 |     15
                   Diet            |  47.83 |  17.39 |  65.22
                                   |  73.33 |  26.67 |
                                   |  84.62 |  40.00 |
                  -----------------+--------+--------+
                  Low Cholesterol  |      2 |      6 |      8
                   Diet            |   8.70 |  26.09 |  34.78
                                   |  25.00 |  75.00 |
                                   |  15.38 |  60.00 |
                  -----------------+--------+--------+
                  Total                  13       10       23
                                      56.52    43.48   100.00


                  Statistics for Table of Exposure by Response

             Statistic                     DF       Value      Prob
             ------------------------------------------------------
             Chi-Square                     1      4.9597    0.0259
             Likelihood Ratio Chi-Square    1      5.0975    0.0240
             Continuity Adj. Chi-Square     1      3.1879    0.0742
             Mantel-Haenszel Chi-Square     1      4.7441    0.0294
             Phi Coefficient                       0.4644
             Contingency Coefficient               0.4212
             Cramer's V                            0.4644

          WARNING: 50% of the cells have expected counts less than 5.
                   (Asymptotic) Chi-Square may not be a valid test.
```

```
          Pearson Chi-Square Test
     ----------------------------------
     Chi-Square                  4.9597
     DF                               1
     Asymptotic Pr >  ChiSq      0.0259
     Exact      Pr >= ChiSq      0.0393


            Fisher's Exact Test
     ----------------------------------
     Cell (1,1) Frequency (F)        11
     Left-sided Pr <= F          0.9967
     Right-sided Pr >= F         0.0367

     Table Probability (P)       0.0334
     Two-sided Pr <= P           0.0393
```

```
              Case-Control Study of High Fat/Cholesterol Diet                  2

                             The FREQ Procedure

                 Statistics for Table of Exposure by Response

                 Estimates of the Relative Risk (Row1/Row2)

 Type of Study                   Value       95% Confidence Limits
 -----------------------------------------------------------------
 Case-Control (Odds Ratio)      8.2500        1.1535       59.0029
 Cohort (Col1 Risk)             2.9333        0.8502       10.1204
 Cohort (Col2 Risk)             0.3556        0.1403        0.9009


                      Odds Ratio (Case-Control Study)
                    -----------------------------------
                    Odds Ratio                   8.2500

                    Asymptotic Conf Limits
                    95% Lower Conf Limit         1.1535
                    95% Upper Conf Limit        59.0029

                    Exact Conf Limits
                    95% Lower Conf Limit         0.8677
                    95% Upper Conf Limit       105.5488

                              Sample Size = 23
```

Example 5: Creating an Output Data Set Containing Chi-Square Statistics

Procedure features:

PROC FREQ statement option:

ORDER=

OUTPUT statement options:

OUT=

statistic-keywords

TABLES statement options:

CHISQ
DEVIATION
EXPECTED
NOCOL
NOROW

WEIGHT statement

Other features:

PRINT procedure

Data set: COLOR on page 593

This example

- creates a 3×5 contingency table showing the joint frequency distribution for two variables
- suppresses the row and column percentages for each cell
- displays the expected frequency for each cell
- displays each cell's contribution to the total Pearson chi-square statistic
- creates an output data set with Pearson chi-square and likelihood-ratio chi-square statistics
- displays the output data set.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 pagesize=60;
```

Generate the crosstabulation table in a specified order from existing cell counts. ORDER=DATA orders the table values (eye and hair color) by their order in the data set. The WEIGHT statement uses Count to weight the observations.

```
proc freq data=color order=data;
   weight count;
```

Specify the variables to use to create the contingency tables. Compute chi-square tests and measures of association based on chi-square. Display the expected frequency and the contribution to the total Pearson chi-square statistic for each cell. The TABLES statement requests a two-way table. CHISQ requests chi-square tests. EXPECTED displays the expected cell frequency, and CELLCHI2 displays the cell contribution to chi-square. NOROW and NOCOL suppress the row and column percents for each cell.

```
   tables eyes*hair /chisq expected cellchi2
                     norow nocol;
```

Create the output data set CHISQDAT. The OUTPUT statement creates the CHISQDAT data set with eight variables. N stores the number of nonmissing observations, NMISS stores the number of missing observations, PCHI stores Pearson chi-square statistics, and LRCHI stores likelihood-ratio chi-square statistics. The TITLE statement specifies a title.

```
   output out=chisqdat pchi lrchi n nmiss;
```

Specify the title.

```
title 'Chi-Square Tests for 3 by 5 Table of Eye and Hair Color';
run;
```

Print the data set. PROC PRINT displays the CHISQDAT data set. The TITLE statement specifies a title.

```
proc print data=chisqdat noobs;
   title 'Chi-Square Statistics for Eye and Hair Color';
   title2 'Output Data Set from the FREQ Procedure';
run;
```

Output

The contingency table lists eye and hair color in the order that they appear in the data set. The first column label explains the contents of each table cell. The Pearson chi-square provides evidence of an association between eye and hair color (p=.007). The cell chi-square values show that most of the association is due to more green-eyed children with fair or red hair and fewer with dark or black hair. Exactly the opposite occurs with the brown-eyed children.

```
                 Chi-Square Tests for 3 by 5 Table of Eye and Hair Color                       1

                                   The FREQ Procedure

                                 Table of Eyes by Hair

           Eyes(Eye Color)     Hair(Hair Color)

           Frequency      |
           Expected       |
           Cell Chi-Square|
           Percent        |fair    |red     |medium  |dark    |black   |  Total
           ---------------+--------+--------+--------+--------+--------+
           blue           |     69 |     28 |     68 |     51 |      6 |    222
                          | 66.425 | 32.921 |  63.22 | 53.024 | 6.4094 |
                          | 0.0998 | 0.7357 | 0.3613 | 0.0772 | 0.0262 |
                          |   9.06 |   3.67 |   8.92 |   6.69 |   0.79 |  29.13
           ---------------+--------+--------+--------+--------+--------+
           green          |     69 |     38 |     55 |     37 |      0 |    199
                          | 59.543 |  29.51 | 56.671 |  47.53 | 5.7454 |
                          | 1.5019 | 2.4422 | 0.0492 | 2.3329 | 5.7454 |
                          |   9.06 |   4.99 |   7.22 |   4.86 |   0.00 |  26.12
           ---------------+--------+--------+--------+--------+--------+
           brown          |     90 |     47 |     94 |     94 |     16 |    341
                          | 102.03 | 50.568 | 97.109 | 81.446 | 9.8451 |
                          | 1.4187 | 0.2518 | 0.0995 |  1.935 | 3.8478 |
                          |  11.81 |   6.17 |  12.34 |  12.34 |   2.10 |  44.75
           ---------------+--------+--------+--------+--------+--------+
           Total               228      113      217      182       22      762
                             29.92    14.83    28.48    23.88     2.89   100.00


                          Statistics for Table of Eyes by Hair

                  Statistic                     DF       Value      Prob
                  ------------------------------------------------------
                  Chi-Square                     8     20.9248    0.0073
                  Likelihood Ratio Chi-Square    8     25.9733    0.0011
                  Mantel-Haenszel Chi-Square     1      3.7838    0.0518
                  Phi Coefficient                       0.1657
                  Contingency Coefficient               0.1635
                  Cramer's V                            0.1172

                                   Sample Size = 762
```

The output data set has one observation that contains the sample size, number of missing observations, and chi-square statistics with the corresponding degrees of freedom and probability values.

```
                    Chi-Square Statistics for Eye and Hair Color                 2
                      Output Data Set from the FREQ Procedure

    N    NMISS    _PCHI_   DF_PCHI       P_PCHI    _LRCHI_   DF_LRCHI      P_LRCHI

   762     0     20.9248      8      .007349898   25.9733       8      .001061424
```

Example 6: Computing Cochran-Mantel-Haenszel Statistics for a Stratified Table

Procedure features:

TABLES statement options:

CMH

NOPRINT

WEIGHT statement

This example

- □ creates stratified two-way contingency tables using existing cell counts
- □ suppresses the display of the contingency tables
- □ computes Cochran-Mantel-Haenszel statistics adjusting for the effects of a stratification variable.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the MIGRAINE data set. This data set contains hypothetical data for a clinical trial of migraine treatment. Subjects of both genders receive either new drug therapy or a placebo. Their response to treatment is coded as better or the same. The data are recorded as cell counts instead of as one observation per subject. The variable Frequency contains the frequencies for each treatment and response combination.

```
data migraine;
   input Gender $ Treatment $ Improve $ Frequency @@;
   datalines;
```

```
female Active  Better 16  female Active  Same 11
female Placebo Better  5  female Placebo Same 20
male   Active  Better 12  male   Active  Same 16
male   Placebo Better  7  male   Placebo Same 19
;
```

Generate the crosstabulation tables from existing cell counts. The WEIGHT statement uses Frequency to weight the observations.

```
proc freq data=migraine;
   weight frequency;
```

Specify the variables to use to create the three-way table. Compute Cochran-Mantel-Haenszel statistics, adjusted relative risks, and odds ratios. Suppress the printing of the tables. The TABLES statement requests a three-way table stratified by Gender where Treatment forms the rows and Improve forms the columns. CMH requests the Cochran-Mantel-Haenszel statistics. NOPRINT suppresses the display of contingency tables.

```
   tables gender*treatment*improve/cmh noprint;
```

Specify the title.

```
   title1 'Clinical Trial for Treatment of Migraine Headaches';
run;
```

Output

PROC FREQ computes Cochran-Mantel-Haenszel statistics, controlling for Gender. For stratified 2×2 contingency tables, these statistics include estimates of the common relative risk and the Breslow-Day test for homogeneity of the odds ratios. For a stratified 2×2 table, the three CMH statistics test the same hypothesis. The significant **p**-value (.004) indicates that the association between treatment and response remains strong after adjusting for gender.

The large **p**-value for the Breslow-Day test (.222) indicates no significant gender difference in the odds ratios. Because this is a prospective study, the relative risk estimate assesses the effectiveness of the new drug. The probability of migraine improvement with the new drug is just over two times the probability of improvement with the placebo.

```
              Clinical Trial for Treatment of Migraine Headaches            1

                             The FREQ Procedure

                  Summary Statistics for Treatment by Improve
                           Controlling for Gender

         Cochran-Mantel-Haenszel Statistics (Based on Table Scores)

       Statistic    Alternative Hypothesis    DF       Value      Prob
       ---------------------------------------------------------------
           1        Nonzero Correlation        1      8.3052    0.0040
           2        Row Mean Scores Differ     1      8.3052    0.0040
           3        General Association        1      8.3052    0.0040


              Estimates of the Common Relative Risk (Row1/Row2)

  Type of Study     Method                 Value     95% Confidence Limits
  -------------------------------------------------------------------------
  Case-Control      Mantel-Haenszel       3.3132       1.4456       7.5934
    (Odds Ratio)    Logit                 3.2941       1.4182       7.6515

  Cohort            Mantel-Haenszel       2.1636       1.2336       3.7948
    (Col1 Risk)     Logit                 2.1059       1.1951       3.7108

  Cohort            Mantel-Haenszel       0.6420       0.4705       0.8761
    (Col2 Risk)     Logit                 0.6613       0.4852       0.9013


                             Breslow-Day Test for
                         Homogeneity of the Odds Ratios
                         ------------------------------
                         Chi-Square              1.4929
                         DF                           1
                         Pr > ChiSq              0.2218


                            Total Sample Size = 106
```

Example 7: Computing the Cochran-Armitage Trend Test

Procedure features:

EXACT statement options:

statistic-keywords
MAXTIME=

TABLES statement options:

CL
MEASURES
TREND

TEST statement

WEIGHT statement

This example

- □ creates a two-way table using existing cell counts
- □ computes measures of association and asymptotic 95% confidence limits
- □ computes asymptotic and exact p-values for the Cochran-Armitage trend test
- □ specifies the maximum time to compute an exact p-value
- □ computes asymptotic tests for Somers' $D(C|R)$.

The Cochran-Armitage test checks for trend in binomial proportions across levels of a single factor. Use this test for a contingency table with a two-level response variable and an explanatory variable with any number of ordered levels. The binomial proportion is defined as the proportion in the first level of the response variable. PROC FREQ uses explanatory variable scores to compute the Cochran-Armitage test, which you can set to meaningful values that reflect the degree of difference among the levels.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=72;
```

Create the PAIN data set. This data set contains hypothetical data for a clinical trial of a drug therapy to control pain. The clinical trial investigates whether adverse responses increase with larger drug doses. Subjects receive either a placebo or one of four drug doses. An adverse response is coded No or Yes. The data are recorded as cell counts instead of as one observation per subject. The variable Count contains the frequencies for each drug dose and response combination.

```
data pain;
   input Dose Adverse $ Count @@;
   cards;
0 No 26 0 Yes  6
1 No 26 1 Yes  7
2 No 23 2 Yes  9
3 No 18 3 Yes 14
4 No  9 4 Yes 23
;
```

Generate the cross-tabulation tables from existing cell counts. The WEIGHT statement uses Count to weight the observations.

```
proc freq data=pain;
   weight count;
```

Specify the variables to use to create the two-way table. Compute measures of association, their confidence limits, and the Cochran-Armitage test for trend. The TABLES statement requests a two-way table. TREND requests the Cochran-Armitage trend test. MEASURES requests measures of associations and CL computes confidence limits.

```
   tables dose*adverse /trend measures cl;
```

Computes asymptotic tests for the specified measures of association. The TEST statement computes an asymptotic test for Somers' **D**(**C**|**R**).

```
   test smdcr;
```

Compute exact tests for the specified statistics. The EXACT statement requests exact trend test. MAXTIME= specifies that PROC FREQ terminate the computations after 60 seconds (1 minute).

```
   exact trend /maxtime=60;
```

Specify the title.

```
   title1 'Clinical Trial for Treatment of Pain';
run;
```

Output

The Row Pct values show the expected increasing trend in the proportion of adverse effects (from 18.75% to 71.88%).

```
                     Clinical Trial for Treatment of Pain                     1

                              The FREQ Procedure

                           Table of Dose by Adverse

                     Dose      Adverse

                     Frequency|
                     Percent  |
                     Row Pct  |
                     Col Pct  |No      |Yes     |  Total
                     ---------+--------+--------+
                            0 |     26 |      6 |     32
                              |  16.15 |   3.73 |  19.88
                              |  81.25 |  18.75 |
                              |  25.49 |  10.17 |
                     ---------+--------+--------+
                            1 |     26 |      7 |     33
                              |  16.15 |   4.35 |  20.50
                              |  78.79 |  21.21 |
                              |  25.49 |  11.86 |
                     ---------+--------+--------+
                            2 |     23 |      9 |     32
                              |  14.29 |   5.59 |  19.88
                              |  71.88 |  28.13 |
                              |  22.55 |  15.25 |
                     ---------+--------+--------+
                            3 |     18 |     14 |     32
                              |  11.18 |   8.70 |  19.88
                              |  56.25 |  43.75 |
                              |  17.65 |  23.73 |
                     ---------+--------+--------+
                            4 |      9 |     23 |     32
                              |   5.59 |  14.29 |  19.88
                              |  28.13 |  71.88 |
                              |   8.82 |  38.98 |
                     ---------+--------+--------+
                     Total         102       59      161
                                 63.35    36.65   100.00


                   Statistics for Table of Dose by Adverse

                                                                   95%
 Statistic                              Value       ASE     Confidence Limits
 ----------------------------------------------------------------------------
 Gamma                                 0.5313    0.0935      0.3480      0.7146
 Kendall's Tau-b                       0.3373    0.0642      0.2114      0.4631
 Stuart's Tau-c                        0.4111    0.0798      0.2547      0.5675

 Somers' D C|R                         0.2569    0.0499      0.1592      0.3547
 Somers' D R|C                         0.4427    0.0837      0.2786      0.6068

 Pearson Correlation                   0.3776    0.0714      0.2378      0.5175
 Spearman Correlation                  0.3771    0.0718      0.2363      0.5178

 Lambda Asymmetric C|R                 0.2373    0.0837      0.0732      0.4014
 Lambda Asymmetric R|C                 0.1250    0.0662      0.0000      0.2547
 Lambda Symmetric                      0.1604    0.0621      0.0388      0.2821

 Uncertainty Coefficient C|R           0.1261    0.0467      0.0346      0.2175
 Uncertainty Coefficient R|C           0.0515    0.0191      0.0140      0.0890
 Uncertainty Coefficient Symmetric     0.0731    0.0271      0.0199      0.1262
```

Somers' **D** (**C|R**)measures the association. The column variable (Adverse) is the response and the row variable (Dose) is a predictor. Because the asymptotic 95% confidence limit does not contain zero, this indicates a strong positive association. Similarly, Pearson and Spearman correlation coefficients show evidence of a strong positive association as hypothesized.

The Cochran-Armitage test supports the trend hypothesis. The small left-sided **p**-values indicate that the probability of the Column 1 level (Adverse=No) decreases as Dose increases, or equivalently, that the probability of the Column 2 level (Adverse=Yes) increases as Dose increases. The two-sided **p**-value tests against either the increasing or the decreasing alternative. This is an appropriate hypothesis when you want to determine whether the drug has progressive effects on the probability of adverse effects, but the direction is unknown.

```
                    Clinical Trial for Treatment of Pain                    2

                             The FREQ Procedure

                  Statistics for Table of Dose by Adverse

                               Somers' D C|R
                       --------------------------------
                       Somers' D C|R             0.2569
                       ASE                       0.0499
                       95% Lower Conf Limit      0.1592
                       95% Upper Conf Limit      0.3547

                        Test of H0: Somers' D C|R = 0

                       ASE under H0              0.0499
                       Z                         5.1511
                       One-sided Pr >  Z         <.0001
                       Two-sided Pr > |Z|        <.0001


                         Cochran-Armitage Trend Test
                       -------------------------------
                       Statistic (Z)           -4.7918

                       Asymptotic Test
                       One-sided Pr <  Z        <.0001
                       Two-sided Pr > |Z|       <.0001

                       Exact Test
                       One-sided Pr <=  Z    7.237E-07
                       Two-sided Pr >= |Z|   1.324E-06

                              Sample Size = 161
```

Example 8: Computing Friedman's Chi-Square Statistic

Procedure features:

TABLES statement, multiple requests

TABLES statement options:

- CMH2
- NOPRINT
- SCORES=
- SCOROUT

This example

- computes the first two Cochran-Mantel-Haenszel statistics
- uses rank scores to compute the Cochran-Mantel-Haenszel statistics
- suppresses the display of contingency tables for each stratum.

Friedman's test is a nonparametric test for treatment differences in a randomized complete block design. Each block of the design may be a subject or a homogeneous group of subjects. If blocks are groups of subjects, the number of subjects in each block must equal the number of treatments. Treatments are randomly assigned to subjects within each block. If there is one subject per block, then the subjects are repeatedly measured once they are under each treatment. The order of treatments is randomized for each subject.

In this setting, Friedman's test is identical to the *ANOVA* (row means scores) CMH statistic when the analysis uses rank scores (SCORES=RANK). The three-way table uses subject (or subject group) as the stratifying variable, treatment as the row variable, and response as the column variable. PROC FREQ handles ties by assigning midranks to tied response values. If there are multiple subjects per treatment in each block, the *ANOVA* CMH statistic is a generalization of Friedman's test.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80;
```

Create the HYPNOSIS data set. This data set contains data for a study investigating whether hypnosis has the same effect on skin potential (measured in millivolts) for four emotions (Lehmann 1975, 264). Eight subjects are asked to display fear, happiness (joy), depression (sadness), and calmness under hypnosis. The data are recorded as one observation per subject for each emotion.

```
data hypnosis;
   length Emotion $ 10;
   input Subject Emotion $ SkinResponse @@;
   datalines;
1 fear 23.1  1 joy 22.7  1 sadness 22.5  1 calmness 22.6
2 fear 57.6  2 joy 53.2  2 sadness 53.7  2 calmness 53.1
3 fear 10.5  3 joy  9.7  3 sadness 10.8  3 calmness  8.3
4 fear 23.6  4 joy 19.6  4 sadness 21.1  4 calmness 21.6
5 fear 11.9  5 joy 13.8  5 sadness 13.7  5 calmness 13.3
6 fear 54.6  6 joy 47.1  6 sadness 39.2  6 calmness 37.0
7 fear 21.0  7 joy 13.6  7 sadness 13.7  7 calmness 14.8
8 fear 20.3  8 joy 23.6  8 sadness 16.3  8 calmness 14.8
;
```

Specify the variables to use to create the three-way table and the two-way table. Compute adjusted relative risks and odds ratios, CMH correlation, and row mean scores (ANOVA) statistic by using rank scores. Suppress the printing of the tables. The TABLES statement requests a three-way table stratified by Subject and a two-way table. Emotion and SkinResponse form the rows and columns of each table. CMH2 requests the first two Cochran-Mantel-Haenszel statistics. SCORES=RANK uses rank scores to compute these statistics. NOPRINT suppresses the display of contingency tables.

```
proc freq data=hypnosis;
   tables subject*emotion*skinresponse emotion*skinresponse
          /cmh2 scores=rank noprint;
```

Specify the title.

```
   title1 'Examining the Effect of Hypnosis on Skin Potential';
run;
```

Output

PROC FREQ computes Cochran-Mantel-Haenszel statistics across strata controlling for Subject. Because CMH statistics are based on rank scores, the Row Mean Scores Differ statistic is identical to Friedman's chi-square (**Q**=6.45). The **p**-value of .09 indicates that differences in skin potential response for different emotions are significant at the 10% level but not at the 5% level.

When you do not stratify by subject, the Row Mean Scores Differ CMH statistic is identical to a Kruskal-Wallis test and is not significant (**p**=.904). Thus, adjusting for subject is critical to reducing the background variation due to subject differences.

```
               Examining the Effect of Hypnosis on Skin Potential            1

                               The FREQ Procedure

                 Summary Statistics for Emotion by SkinResponse
                            Controlling for Subject

           Cochran-Mantel-Haenszel Statistics (Based on Rank Scores)

        Statistic    Alternative Hypothesis    DF       Value      Prob
        ---------------------------------------------------------------
            1        Nonzero Correlation        1      0.2400    0.6242
            2        Row Mean Scores Differ     3      6.4500    0.0917


                            Total Sample Size = 32
```

```
               Examining the Effect of Hypnosis on Skin Potential            2

                              The FREQ Procedure

                 Summary Statistics for Emotion by SkinResponse

           Cochran-Mantel-Haenszel Statistics (Based on Rank Scores)

        Statistic    Alternative Hypothesis    DF       Value      Prob
        ---------------------------------------------------------------
            1        Nonzero Correlation        1      0.0001    0.9933
            2        Row Mean Scores Differ     3      0.5678    0.9038


                            Total Sample Size = 32
```

Example 9: Testing Marginal Homogeneity with Cochran's *Q*

Procedure features:

TABLES statement, multiple requests

TABLES statement options:

AGREE
NOCUM
NOPRINT

WEIGHT statement

Other features:

FORMAT procedure

This example

- creates frequency tables for the analysis variables using existing cell counts
- computes tests and measures of agreement, which include Cochran's *Q* statistic for stratified 2×2 contingency tables
- suppresses the cumulative frequencies and cumulative percentages
- suppresses the display of contingency tables.

When a binary response is measured several times or under different conditions, Cochran's *Q* tests that the marginal probability of a positive response is unchanged across the times or conditions. When there are more than two response categories, you can use PROC CATMOD in SAS/STAT software to fit a repeated-measures model. Data for this example are from *Categorical Data Analysis* by Alan Agresti. Copyright © 1990. Reprinted by permission of John Wiley and Sons, Inc.

Program

Set the SAS system options. The NODATE option specifies to omit the date and time when the SAS job began. The PAGENO= option specifies the page number for the next page of output that SAS produces. The LINESIZE= option specifies the line size. The PAGESIZE= option specifies the number of lines for a page of SAS output.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Assign a character string format to a single character value. PROC FORMAT creates a user-written format to identify the response to treatment.

```
proc format;
   value $responsefmt 'F'='Favorable'
                      'U'='Unfavorable';
run;
```

Create the DRUGS data set. This data set contains data for a study of three drugs to treat a chronic condition (Agresti, 1990). Forty-six subjects receive drugs A, B, and C. The response to each is coded as favorable (`F`) or unfavorable (`U`). The data are recorded as cell counts instead of as one observation per patient. The variable Count contains the cell count.

```
data drugs;
   input Drug_A $ Drug_B $ Drug_C $ Count @@;
   datalines;
F F F 6   F F U 16   F U F 2
F U U 4   U F F  2   U F U 4
U U F 6   U U U  6
;
```

Generate the crosstabulation table from existing cell counts. The WEIGHT statement uses Count to weight the observations.

```
proc freq data=drugs;
   weight count;
```

Specify the variables to use to create the frequency tables. The TABLES statement requests frequency tables of Drug_A, Drug_B, and Drug_C. NOCUM suppresses the cumulative values.

```
   tables drug_a drug_b drug_c/nocum;
```

Specify the variables to use to create the three-way table. Compute tests and measures of classification agreement. Suppress the printing of the tables. The TABLES statement requests a three-way table of Drug_A, Drug_B, and Drug_C. AGREE requests measures of agreement. NOPRINT suppresses the display of contingency tables.

```
   tables drug_a*drug_b*drug_c/agree noprint;
```

Assign a format to a variable and specify a title for the report. The FORMAT statement assigns formats to the levels of Drug_A, Drug_B, and Drug_C. The TITLE statement specifies a title.

```
   format drug_a drug_b drug_c $responsefmt.;
   title 'Study of Three Drug Treatments for a Chronic Disease';
```

```
run;
```

Output

The one-way frequency tables provides the marginal response for each drug. For drugs A and B, 61% of the subjects reported a favorable response while 35% of the subjects reported a favorable response for drug C.

```
           Study of Three Drug Treatments for a Chronic Disease            1

                              The FREQ Procedure

                   Drug_A         Frequency     Percent
                   ------------------------------------
                   Favorable            28       60.87
                   Unfavorable          18       39.13


                   Drug_B         Frequency     Percent
                   ------------------------------------
                   Favorable            28       60.87
                   Unfavorable          18       39.13


                   Drug_C         Frequency     Percent
                   ------------------------------------
                   Favorable            16       34.78
                   Unfavorable          30       65.22
```

McNemar's test shows strong discordance between drugs B and C when the response to drug A is favorable. A small negative value of simple kappa indicates no agreement between the drug B response and the drug C response.

```
            Study of Three Drug Treatments for a Chronic Disease            2

                              The FREQ Procedure

                 Statistics for Table 1 of Drug_B by Drug_C
                      Controlling for Drug_A=Favorable

                               McNemar's Test
                          ------------------------
                          Statistic (S)    10.8889
                          DF                     1
                          Pr > S            0.0010


                          Simple Kappa Coefficient
                      --------------------------------
                      Kappa                    -0.0328
                      ASE                       0.1167
                      95% Lower Conf Limit     -0.2615
                      95% Upper Conf Limit      0.1960

                              Sample Size = 28


                 Statistics for Table 2 of Drug_B by Drug_C
                     Controlling for Drug_A=Unfavorable

                              McNemar's Test
                          -----------------------
                          Statistic (S)    0.4000
                          DF                    1
                          Pr > S           0.5271


                          Simple Kappa Coefficient
                      --------------------------------
                      Kappa                    -0.1538
                      ASE                       0.2230
                      95% Lower Conf Limit     -0.5909
                      95% Upper Conf Limit      0.2832

                              Sample Size = 18
```

In this example, the hypothesis of interest is whether the response to treatment is equal for the three drugs. Cochran's **Q** is statistically significant (**p**=.014), which leads to rejection of the null hypothesis that the probability of favorable response is the same for the three drugs.

```
              Study of Three Drug Treatments for a Chronic Disease             3

                                The FREQ Procedure

                    Summary Statistics for Drug_B by Drug_C
                             Controlling for Drug_A


                          Overall Kappa Coefficient
                       --------------------------------
                       Kappa                    -0.0588
                       ASE                       0.1034
                       95% Lower Conf Limit     -0.2615
                       95% Upper Conf Limit      0.1439


                             Test for Equal Kappa
                                 Coefficients
                             --------------------
                             Chi-Square    0.2314
                             DF                 1
                             Pr > ChiSq    0.6305


                           Cochran's Q, for Drug_A
                             by Drug_B by Drug_C
                           -----------------------
                           Statistic (Q)    8.4706
                           DF                    2
                           Pr > Q           0.0145


                            Total Sample Size = 46
```

References

Agresti, A. (1992), "A Survey of Exact Inference for Contingency Tables," *Statistical Science*, 7(1), 131–177.

Agresti, A. (1996), *An Introduction to Categorical Data Analysis*, New York: John Wiley and Sons, Inc.

Agresti, A. (1990), *Categorical Data Analysis*, New York: John Wiley and Sons, Inc.

Agresti, A., Mehta, C.R. and Patel, N.R. (1990), "Exact Inference for Contingency Tables with Ordered Categories," *Journal of the American Statistical Association*, 85, 453–458.

Agresti, A., Wackerly, D. and Boyett, J.M. (1979), " Exact Conditional Tests for Cross-Classifications: Approximation of Attained Significance Levels," *Psychometrika*, 44, 75-83.

Birch, M.W. (1965), "The Detection of Partial Association, II: The General Case," *Journal of the Royal Statistical Society, B*, 27, 111–124.

Bishop, Y., Fienberg, S.E., and Holland, P.W. (1975), *Discrete Multivariate Analysis: Theory and Practice*, Cambridge, MA: MIT Press.

Bowker, A.H. (1948), "Bowker's Test for Symmetry," *Journal of the American Statistical Association*, 43, 572–574.

Breslow, N.E. (1996), "Statistics in Epidemiology: The Case-Control Study", *Journal of the American Statistical Association*, 91, 14-26.

Breslow, N.E. and Day, N.E. (1980), *Statistical Methods in Cancer Research, Volume I: The Analysis of Case-Control Studies*, IARC Scientific Publications, No. 32, Lyon, International Agency for Research on Cancer.

Breslow, N.E. and Day, N.E. (1980), *Statistical Methods in Cancer Research, Volume II: The Design and Analysis of Cohort Studies*, IARC Scientific Publications, No. 82, Lyon, International Agency for Research on Cancer.

Bross, I.D.J. (1958), "How to Use Ridit Analysis," *Biometrics*, 14, 18–38.

Brown, M.B. and Benedetti, J.K. (1977), "Sampling Behavior of Tests for Correlation in Two-Way Contingency Tables," *Journal of the American Statistical Association* 72, 309–315.

Cicchetti, D.V. and Allison, T. (1971), "A New Procedure for Assessing Reliability of Scoring EEG Sleep Recordings," *American Journal of EEG Technology*, 11, 101–109.

Cochran, W.G. (1950), "The Comparison of Percentages in Matched Samples," *Biometrika*, 37, 256–266.

Cochran, W.G. (1954), "Some Methods for Strengthening the Common χ^2 Tests," *Biometrics*, 10, 417–451.

Collett, D. (1991), *Modelling Binary Data*, London: Chapman and Hall.

Cohen, J. (1960), "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, 20, 37–46.

Drasgow, F. (1986), "Polychoric and Polyserial Correlations" in *Encyclopedia of Statistical Sciences, Volume 7*, eds. S. Kotz and N. L. Johnson, New York: John Wiley and Sons, Inc., 68–74.

Fienberg, S.E. (1980), *The Analysis of Cross-Classified Data,* 2nd Edition, Cambridge, MA: MIT Press.

Fleiss, J.L. (2000), *Statistical Methods for Rates and Proportions*, 2nd Edition, New York: John Wiley and Sons, Inc.

Fleiss, J.L. and Cohen, J. (1973), " The Equivalence of Weighted Kappa and the Intraclass Correlation Coefficient as Measures of Reliability," *Educational and Psychological Measurement*, 33, 613–619.

Fleiss, J.L., Cohen, J., and Everitt, B.S. (1969), "Large-Sample Standard Errors of Kappa and Weighted Kappa," *Psychological Bulletin*, 72, 323–327.

Freeman, G.H. and Halton, J.H. (1951), "Note on an Exact Treatment of Contingency, Goodness of Fit and Other Problems of Significance," *Biometrika*, 38, 141–149.

Gail, M. and Mantel, N. (1977), "Counting the Number of $r \times c$ Contingency Tables with Fixed Margins," *Journal of the American Statistical Association*, 72, 859-862.

Gart, J.J. (1971), "The Comparison of Proportions: A Review of Significance Tests, Confidence Intervals and Adjustments for Stratification," *Review of the International Statistical Institute*, 39(2), 148–169.

Goodman, L.A. and Kruskal, W.H. (1954, 1959, 1963, 1972), "Measures of Association for Cross-Classification I, II, III, and IV," *Journal of the American Statistical Association*, 49, 732–764; 54, 123–163; 58, 310–364; 67, 415–421.

Greenland, S. and Robins, J.M. (1985), "Estimators of the Mantel-Haenszel Variance Consistent in Both Sparse Data and Large-Strata Limiting Models," *Biometrics*, 42, 311-323.

Haldane, J.B.S. (1955), "The Estimation and Significance of the Logarithm of a Ratio of Frequencies," *Annals of Human Genetics*, 20, 309–314.

Hollander, M. and Wolfe, D.A. (1999), *Nonparametric Statistical Methods, Second Edition*, New York: John Wiley and Sons, Inc.

Jones, M.P., O'Gorman, T.W., Lemka, J.H., and Woolson, R.F. (1989), "A Monte Carlo Investigation of Homogeneity Tests of the Odds Ratio Under Various Sample Size Configurations," *Biometrics*, 45, 171–181.

Kendall, M. (1955), *Rank Correlation Methods*, 2nd Edition, London: Charles Griffin and Co.

Kendall, M. and Stuart, A. (1979), *The Advanced Theory of Statistics, Volume 2*, New York: Macmillan Publishing Company, Inc.

Kleinbaum, D.G., Kupper, L.L., and Morgenstern, H. (1982), *Epidemiologic Research: Principles and Quantitative Methods*, Research Methods Series, New York: Van Nostrand Reinhold.

Landis, R.J., Heyman, E.R., and Koch, G.G. (1978), "Average Partial Association in Three-way Contingency Tables: A Review and Discussion of Alternative Tests," *International Statistical Review*, 46, 237–254.

Leemis, L.M. and Trivedi, K.S. (1996), "A Comparison of Approximate Interval Estimators for the Bernoulli Parameter," *The American Statistician*, 50(1), 63–68.

Lehmann, E.L. (1975), *Nonparametrics: Statistical Methods Based on Ranks*, San Francisco: Holden-Day, Inc.

Liebetrau, A.M. (1983), *Measures of Association*, Quantitative Application in the Social Sciences, Vol. 32, Beverly Hills: Sage Publications, Inc.

Mack, G.A. and Skillings, J.H. (1980), "A Friedman-Type Rank Test for Main Effects in a Two-Factor ANOVA," *Journal of the American Statistical Association*, 75, 947–951.

Mantel, N. (1963), "Chi-square Tests with One Degree of Freedom: Extensions of the Mantel-Haenszel Procedure," *Journal of the American Statistical Association*, 58, 690–700.

Mantel, N. and Haenszel, W. (1959), "Statistical Aspects of the Analysis of Data from Retrospective Studies of Disease," *Journal of the National Cancer Institute*, 22, 719–748.

Margolin, B.H. (1988), "Test for Trend in Proportions," *Johnson's Encyclopedia of Statistics, Volume 9*, eds. S. Kotz and N.L. Johnson, New York: John Wiley and Sons, Inc., 334–336.

McNemar, Q. (1947), "Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages," *Psychometrika*, 12, 153–157.

Mehta, C.R. and Patel, N.R. (1983), "A Network Algorithm for Performing Fisher's Exact Test in $r \times c$ Contingency Tables," *Journal of the American Statistical Association*, 78, 427–434.

Mehta, C.R., Patel, N.R., and Senchaudhuri, P. (1991), "Exact Stratified Linear Rank Tests for Binary Data," *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, ed. E.M. Keramidas, 200–207.

Mehta, C.R., Patel, N.R., and Tsiatis, A.A. (1984), "Exact Significance Testing to Establish Treatment Equivalence with Ordered Categorical Data," *Biometrics*, 40, 819–825.

Narayanan, A. and Watts, D. (1996), "Exact Methods in the NPAR1WAY Procedure," in *Proceedings of the Twenty-First Annual SAS Users Group International Conference*, Cary, NC: SAS Institute Inc., 1290–1294.

Olsson, U. (1979), "Maximum Likelihood Estimation of the Polychoric Correlation Coefficient," *Psychometrika*, 12, 443–460.

Pirie, W. (1983), "Jonckheere Tests for Ordered Alternatives," in *Encyclopedia of Statistical Sciences, Volume 4*, eds. S. Kotz and N.L. Johnson, New York: John Wiley and Sons, Inc., 315–318.

Radlow, R. and Alf, E.F. (1975), "An Alternate Multinomial Assessment of the Accuracy of the Chi-Square Test of Goodness of Fit," *Journal of the American Statistical Association*, 70, 811-813.

Robins, J.M., Breslow, N., and Greenland, S. (1986), "Estimators of the Mantel-Haenszel Variance Consistent in Both Sparse Data and Large-Strata Limiting Models," *Biometrics*, 42, 311-323.

Snedecor, G.W. and Cochran, W.G. (1989), *Statistical Methods*, 8th Edition, Ames, IA: Iowa State University Press.

Somers, R.H. (1962), "A New Asymmetric Measure of Association for Ordinal Variables," *American Sociological Review*, 27, 799–811.

Stokes, M.E., Davis, C.S., and Koch, G.G. (2000), *Categorical Data Analysis Using the SAS System, Second edition*, Cary, NC: SAS Institute Inc.

Tarone, R.E. (1985), "On Heterogeneity Tests Based on Efficient Scores," *Biometrika*, 72, 1, 91-95.

Theil, H. (1972), *Statistical Decomposition Analysis*, Amsterdam: North-Holland Publishing Company.

Thomas, D.G. (1971), "Algorithm AS-36. Exact Confidence Limits for the Odds Ratio in a 2×2 Table," *Applied Statistics*, 20, 105–110.

Valz, P.D. and Thompson, M.E. (1994), "Exact Inference for Kendall's S and Spearman's Rho with Extensions to Fisher's Exact Test in $r \times c$ Contingency Tables," *Journal of Computational and Graphical Statistics*, 3(4), 459–472.

van Elteren, P.H. (1960), "On the Combination of Independent Two-Sample Tests of Wilcoxon," *Bulletin of the International Statistical Institute*, 37, 351–361.

Woolf, B. (1955), "On Estimating the Relationship between Blood Group and Disease," *Annals of Human Genetics*, 19, 251–253.

CHAPTER

24

The FSLIST Procedure

Overview: FSLIST Procedure

The FSLIST procedure allows you to browse external files that are not SAS data sets within a SAS session. Because the files are displayed in an interactive window, the procedure provides a highly convenient mechanism for examining file contents. In addition, you can copy text from the FSLIST window into any window that uses the SAS Text Editor.

Note: For complete documentation on PROC FSLIST, see *SAS/FSP Procedures Guide*. △

Syntax: FSLIST Procedure

PROC FSLIST
FILEREF=*file-specification* | UNIT=*nn* <*option(s)*>;

- You must specify either the FILEREF= or the UNIT= argument with the PROC FSLIST statement.
- *Option(s)* can be one or more of the following:

  CAPS | NOCAPS

  CC | FORTCC | NOCC

  HSCROLL=HALF | PAGE | *n*

  NOBORDER

  NUM | NONUM

  OVP | NOOVP

Statement Descriptions

The only statement that the FSLIST procedure supports is the PROC FSLIST statement, which starts the procedure.

Requirements

You must specify an external file for PROC FSLIST to browse.

FSLIST Command

The FSLIST procedure can also be initiated by entering the following command on the command line of any SAS window:

FSLIST <*|?| *file-specification* <*carriage-control-option* <*overprinting-option*>>>

where *carriage-control-option* can be CC, FORTCC, or NOCC and *overprinting-option* can be OVP or NOOVP.

Note: OVP is ignored if NOCC is in effect. △

PROC FSLIST Statement

The PROC FSLIST statement initiates the FSLIST procedure and specifies the external file to browse. Statement options allow you to modify the default behavior of the procedure.

PROC FSLIST Statement Requirements

The PROC FSLIST statement must include one of the following arguments that specifies the external file to browse.

FILEREF=*file-specification*
DDNAME=*file-specification*
DD=*file-specification*

specifies the external file to browse. *file-specification* can be one of the following:

'*external-file*'
: is the complete operating environment file specification (called the fully qualified pathname under some operating enfironments) for the external file. You must enclose the name in quotation marks.

fileref
: is a fileref that has been previously assigned to the external file. You can use the FILENAME statement to associate a fileref with an actual filename. For information about the FILENAME statement, see the section on statements in *SAS Language Reference: Dictionary*.

UNIT=*nn*

defines the FORTRAN-style logical unit number of the external file to browse. This option is useful when the file to browse has a fileref of the form FT*nn*F001, where *nn* is the logical unit number specified in the UNITS= argument. For example, you can specify

```
proc fslist unit=20;
```

instead of

```
proc fslist fileref=ft20f001;
```

This form of fileref was used for a variety of SAS output files in Version 5 SAS software under the MVS, CMS, and VSE operating environments. For example, the SAS log was written to a file with the fileref (DDname) FT11F001.

PROC FSLIST Statement Options

The following options can be used with the PROC FSLIST statement:

CAPS | NOCAPS
controls how search strings for the FIND command are treated:

CAPS
: translates search strings into uppercase unless they are enclosed in quotes. For example, with this option in effect, the command

```
find nc
```

locates occurrences of **NC**, but not **nc**. To locate lowercase characters, enclose the search string in quotes:

```
find 'nc'
```

NOCAPS
: does not perform a translation; the FIND command locates only those text strings that exactly match the search string.

The default is NOCAPS. You can use the CAPS command in the FSLIST window to change the behavior of the procedure while you are browsing a file.

CC | FORTCC | NOCC
indicates whether carriage-control characters are used to format the display. You can specify one of the following values for this option:

CC
: uses the native carriage-control characters of the host operating environment.

FORTCC
: uses FORTRAN-style carriage control. The first column of each line in the external file is not displayed; the character in this column is interpreted as a carriage-control code. The FSLIST procedure recognizes the following carriage-control characters:

+	skip zero lines and print (overprint)
blank	skip one line and print (single space)
0	skip two lines and print (double space)
-	skip three lines and print (triple space)
1	go to new page print.

NOCC
: treats carriage-control characters as regular text.

If the FSLIST procedure can determine from the file's attributes that the file contains-carriage control information, then that carriage-control information is used to format the displayed text (the CC option is the default). Otherwise, the entire contents of the file are treated as text (the NOCC option the default).

Note: Under some operating environments, FORTRAN-style carriage control is the native carriage control. For these environments, the FORTCC and CC options produce the same behavior. △

HSCROLL=*n*|HALF|PAGE
indicates the default horizontal scroll amount for the LEFT and RIGHT commands. The following values are valid:

n sets the default scroll amount to *n* columns.

HALF sets the default scroll amount to half the window width.

PAGE sets the default scroll amount to the full window width.

The default is HSCROLL=HALF. You can use the HSCROLL command in the FSLIST window to change the default scroll amount.

NOBORDER
suppresses the sides and bottom of the FSLIST window's border. When this option is used, text can appear in the columns and row that are normally occupied by the border.

NUM | NONUM
controls the display of line sequence numbers in files that have a record length of 80 and contain sequence numbers in columns 73 through 80. NUM displays the line sequence numbers; NONUM suppresses them. The default is NONUM.

OVP| NOOVP
indicates whether the carriage-control code for overprinting is honored:

OVP causes the procedure to honor the overprint code and print the current line over the previous line when the code is encountered.

NOOVP causes the procedure to ignore the overprint code and print each line from the file on a separate line of the display.

The default is NOOVP. The OVP option is ignored if the NOCC option is in effect.

FSLIST Command

The FSLIST command provides a handy way to initiate an FSLIST session from any SAS window. The command allows you to use either a fileref or a filename to specify the file to browse. It also allows you to specify how carriage-control information is interpreted.

FSLIST Command Syntax

The general form of the FSLIST command is

FSLIST <*|?| *file-specification* <*carriage-control-option* <*overprinting-option*>>>

where *carriage-control-option* can be CC, FORTCC, or NOCC and *overprinting-option* can be OVP or NOOVP.

Note: OVP is ignored if NOCC is in effect. △

FSLIST Command Arguments

You can specify one of the following arguments with the FSLIST command:

*****

opens a dialog window in which you can specify the name of the file to browse, along with various FSLIST procedure options. In the dialog window, you can specify either a physical filename, a fileref, or a directory name. If you specify a directory name, a selection list of the files in the directory is displayed, from which you can choose the desired file.

?

opens a selection window from which you can choose the external file to browse. The selection list in the window includes all external files that are identified in the current SAS session (all files with defined filerefs).

Note: Only filerefs defined within the current SAS session appear in the selection list. Under some operating environments, it is possible to allocate filerefs outside of SAS. Such filerefs do not appear in the selection list that is displayed by the FSLIST command. △

To select a file, position the cursor on the corresponding fileref and press ENTER.

Note: The selection window is not opened if no filerefs have been defined in the current SAS session. Instead, an error message is printed, instructing you to enter a filename with the FSLIST command. △

file-specification

identifies the external file to browse. *file-specification* can be one of the following:

'*external-file*'

the complete operating environment file specification (called the fully qualified pathname under some operating environments) for the external file. You must enclose the name in quotation marks.

If the specified file is not found, a selection window is opened showing all available filerefs.

fileref

a fileref previously assigned to an external file. If you specify a fileref that is not currently defined, a selection window is opened that shows all available filerefs. An error message in the selection window indicates that the specified fileref is not defined.

If you do not specify any of these three arguments, a selection window is opened that shows the available filerefs (as if you had used the ? argument). The selection window is not opened if no filerefs have been defined in the current SAS session. Instead, an error message is printed that instructs you to enter a filename with the FSLIST command.

FSLIST Command Options

If you use a *file-specification* with the FSLIST command, you can also use the following options. These options are not valid with the ? argument, or when no argument is used:

CC | FORTCC | NOCC

indicates whether carriage-control characters are used to format the display. You can specify one of the following values for this option:

CC	uses the native carriage-control characters of the host operating environment.
FORTCC	uses FORTRAN-style carriage control. See the discussion of the PROC FSLIST statement's FORTCC option on page 629 for details.

NOCC treats carriage-control characters as regular text.

If the FSLIST procedure can determine from the file's attributes that the file contains carriage-control information, then that carriage-control information is used to format the displayed text (the CC option is the default). Otherwise, the entire contents of the file are treated as text (the NOCC option is the default).

OVP | NOOVP

indicates whether the carriage-control code for overprinting is honored. OVP causes the overprint code to be honored; NOOVP causes it to be ignored. The default is NOOVP. The OVP option is ignored if NOCC is in effect.

CHAPTER

25

The IMPORT Procedure

Overview: IMPORT Procedure

The IMPORT procedure reads data from an external data source and writes it to a SAS data set. External data sources can include Microsoft Access Database, Excel files, Lotus spreadsheets, and delimited external files (in which columns of data values are separated by a delimiter such as a blank, comma, or tab).

When you execute PROC IMPORT, the procedure reads the input file and writes the data to a SAS data set. The SAS variable definitions are based on the input records. PROC IMPORT imports the data by one of the following methods:

- generated DATA step code
- generated SAS/ACCESS code
- translation engines.

You control the results with statements and options that are specific to the input data source. PROC IMPORT generates the specified output SAS data set and writes information regarding the import to the SAS log. In the log, you see the DATA step or the SAS/ACCESS code that is generated by PROC IMPORT. If a translation engine is used, then no code is submitted.

Note: To import data, you can also use the Import Wizard, which is a windowing tool that guides you through the steps to import an external data source. You can request the Import Wizard to generate IMPORT procedure statements, which you can save to a file for subsequent use. To invoke the Import Wizard, from the SAS windowing environment select

File ► Import Data

△

Syntax: PROC IMPORT

Restriction: PROC IMPORT is available for the following operating environments:

- OpenVMS Alpha
- UNIX
- Microsoft Windows.

PROC IMPORT
DATAFILE="*filename*" | TABLE="*tablename*"
OUT=<*libref.*>*SAS-data-set* <(*SAS-data-set-options*)>
<DBMS=*identifier*><REPLACE> ;
<*data-source-statement(s)*;>

PROC IMPORT Statement

Featured in: All examples

PROC IMPORT
DATAFILE="*filename*" | TABLE="*tablename*"
OUT=<*libref.*>*SAS-data-set* <(*SAS-data-set-options*)>
<DBMS=*identifier*><REPLACE> ;

Required Arguments

DATAFILE="*filename*"
specifies the complete path and filename or a fileref for the input PC file, spreadsheet, or delimited external file. If you specify a fileref or if the complete path and filename does not include special characters (such as the backslash in a path), lowercase characters, or spaces, you can omit the quotation marks. A fileref is a SAS name that is associated with the physical location of the output file. To assign a fileref, use the FILENAME statement. For more information about PC file formats, see *SAS/ACCESS for PC Files: Reference*.

Featured in: Example 1 on page 641, Example 2 on page 645, and Example 3 on page 646

Restriction: PROC IMPORT does not support device types or access methods for the FILENAME statement except for DISK. For example, PROC IMPORT does not support the TEMP device type, which creates a temporary external file.

Interaction: For some input data sources like a Microsoft Excel spreadsheet, in order to determine the data type (numeric or character) for a column, the first eight rows of data are scanned and the most prevalent type of data is used. If most of the data in the first eight rows is missing, SAS defaults to the character data type; any subsequent numeric data for that column becomes missing as well. Mixed data can also create missing values. For example, if the first eight rows contain mostly character data, SAS assigns the column as a character data type; any subsequent numeric data for that column becomes missing.

Restriction: PROC IMPORT can import data only if the data type is supported by SAS. SAS supports numeric and character types of data but not, for example, binary objects. If the data that you want to import is a type not supported by SAS, PROC IMPORT may not be able to import it correctly. In many cases, the procedure attempts to convert the data to the best of its ability; however, for some types, this is not possible.

Tip: For information about how SAS converts data types, see the specific information for the data source that you are importing in *SAS/ACCESS for PC Files: Reference*. For example, see the chapter "Understanding XLS Essentials" for a table that lists XLS data types and the resulting SAS variable data type and formats.

Tip: For a DBF file, if the file was created by Microsoft Visual FoxPro, the file must be exported by Visual FoxPro into an appropriate dBASE format in order to import the file to SAS.

TABLE="*tablename*"
specifies the table name of the input DBMS table. If the name does not include special characters (such as question marks), lowercase characters, or spaces, you can omit the quotation marks. Note that the DBMS table name may be case sensitive.

Requirement: When you import a DBMS table, you must specify the DBMS= option.

Featured in: Example 4 on page 647

OUT=<*libref.*>*SAS-data-set*
identifies the output SAS data set with either a one- or two-level SAS name (library and member name). If the specified SAS data set does not exist, PROC IMPORT creates it. If you specify a one-level name, by default PROC IMPORT uses either the USER library (if assigned) or the WORK library (if USER not assigned).

Featured in: All examples

(*SAS-data-set-options*)
specifies SAS data set options. For example, to assign a password to the resulting SAS data set, you can use the ALTER=, PW=, READ=, or WRITE= data set option, or to import only data that meets a specified condition, you can use the WHERE= data set option. For information about all SAS data set options, see "Data Set Options" in *SAS Language Reference: Dictionary*.

Restriction: You cannot specify data set options when importing delimited, comma-separated, or tab-delimited external files.

Featured in: Example 3 on page 646

Options

DBMS=*identifier*
specifies the type of data to import. To import a DBMS table, you must specify DBMS= using a valid database identifier. For example, DBMS=ACCESS specifies to import a Microsoft Access 2000 or 2002 table. To import PC files, spreadsheets, and delimited external files, you do not have to specify DBMS= if the filename that is specified by DATAFILE= contains a valid extension so that PROC IMPORT can recognize the type of data. For example, PROC IMPORT recognizes the filename ACCOUNTS.WK1 as a Lotus 1-2-3 Release 2 spreadsheet and the filename

MYDATA.CSV as a delimited external file that contains comma-separated data values; therefore, a DBMS= specification is not necessary.

The following values are valid for the DBMS= option:

Identifier	Input Data Source	Extension	Host Availability
ACCESS	Microsoft Access 2000 or 2002 table	.mdb	Microsoft Windows *
ACCESS97	Microsoft Access 97 table	.mdb	Microsoft Windows *
ACCESS2000	Microsoft Access 2000 table	.mdb	Microsoft Windows *
ACCESS2002	Microsoft Access 2002 table	.mdb	Microsoft Windows *
CSV	delimited file (comma-separated values)	.csv	OpenVMS Alpha, UNIX, Microsoft Windows
DBF	dBASE 5.0, IV, III+, and III files	.dbf	UNIX, Microsoft Windows
DLM	delimited file (default delimiter is a blank)	.*	OpenVMS Alpha, UNIX, Microsoft Windows
EXCEL	Excel 2000 or 2002 spreadsheet	.xls	Microsoft Windows *
EXCEL4	Excel 4.0 spreadsheet	.xls	Microsoft Windows
EXCEL5	Excel 5.0 or 7.0 (95) spreadsheet	.xls	Microsoft Windows
EXCEL97	Excel 97 or 7.0 (95) spreadsheet	.xls	Microsoft Windows *
EXCEL2000	Excel 2000 spreadsheet	.xls	Microsoft Windows *
EXCEL2002	Excel 2002 spreadsheet	.xls	Microsoft Windows *
TAB	delimited file (tab-delimited values)	.txt	OpenVMS Alpha, UNIX, Microsoft Windows
WK1	Lotus 1-2-3 Release 2 spreadsheet	.wk1	Microsoft Windows

Identifier	Input Data Source	Extension	Host Availability
WK3	Lotus 1-2-3 Release 3 spreadsheet	.wk3	Microsoft Windows
WK4	Lotus 1-2-3 Release 4 or 5 spreadsheet	.wk4	Microsoft Windows

* Not available for Microsoft Windows 64-Bit Edition.

Restriction: The availability of an input data source depends on

- □ the operating environment, and in some cases the platform, as specified in the previous table.
- □ whether your site has a license to the SAS/ACCESS software for PC file formats. If you do not have a license, only delimited files are supported.

Featured in: Example 1 on page 641 and Example 4 on page 647

When you specify a value for DBMS=, consider the following:

- □ To import a Microsoft Access table, PROC IMPORT can distinguish whether the table is in Access 97, 2000, or 2002 format regardless of your specification. For example, if you specify DBMS=ACCESS and the table is an Access 97 table, PROC IMPORT will import the file.
- □ To import a Microsoft Excel spreadsheet, PROC IMPORT can distinguish some versions regardless of your specification. For example, if you specify DBMS=EXCEL and the spreadsheet is an Excel 97 spreadsheet, PROC IMPORT can import the file. However, if you specify DBMS=EXCEL4 and the spreadsheet is an Excel 2000 spreadsheet, PROC IMPORT cannot import the file. The following table lists the spreadsheets and whether PROC IMPORT can distinguish them based on the DBMS= specification:

Specification	Excel 2002	Excel 2000	Excel 97	Excel 5.0	Excel 4.0
EXCEL	yes	yes	yes	yes	yes
EXCEL2002	yes	yes	yes	yes	yes
EXCEL2000	yes	yes	yes	yes	yes
EXCEL97	yes	yes	yes	yes	yes
EXCEL5	no	no	no	yes	yes
EXCEL4	no	no	no	yes	yes

Note: Although Excel 4.0 and Excel 5.0 spreadsheets are often interchangeable, it is recommended that you specify the exact version. △

REPLACE

overwrites an existing SAS data set. If you do not specify REPLACE, PROC IMPORT does not overwrite an existing data set.

Featured in: Example 1 on page 641

Data Source Statements

Featured in: All examples

PROC IMPORT provides a variety of statements that are specific to the input data source.

Statements for PC Files, Spreadsheets, or Delimited External Files

The following table lists the statements that are available to import PC files, spreadsheets, and delimited external files, and it denotes which statements are valid for a specific data source. For example, Excel spreadsheets have optional statements to indicate whether column names are in the first row of data or which sheet and range of data to import, while a dBASE file (DBF) does not. For more information about PC file formats, see *SAS/ACCESS for PC Files: Reference*.

Data Source	GETNAMES	RANGE	SHEET	DELIMITER	GETDELETED	DATAROW	MEMOSIZE
DBF					X		
WK1	X	X	X				
WK3	X	X	X				
WK4	X	X	X				
EXCEL	X	X	X				
EXCEL4	X	X	X				
EXCEL5	X	X	X				
EXCEL97	X	X	X				X
EXCEL2000	X	X	X				X
EXCEL2002	X	X	X				X
DLM	X			X		X	
CSV	X					X	
TAB	X					X	

DATAROW=*n*;

starts reading data from row number *n* in the external file.

Default:

1	when GETNAMES=NO
2	when GETNAMES=YES (default for GETNAMES=)

Interaction: When GETNAMES=YES, DATAROW= must be equal to or greater than 2. When GETNAMES=NO, DATAROW must be equal to or greater than 1.

DELIMITER='*char*' | '*nn*'*x*;

for a delimited external file, specifies the delimiter that separates columns of data in the input file. You can specify the delimiter as a single character or as a hexadecimal value. For example, if columns of data are separated by an ampersand, specify DELIMITER='&'. If you do not specify DELIMITER=, PROC IMPORT assumes that the delimiter is the blank. You can replace the equal sign with a blank.

Featured in: Example 1 on page 641

GETDELETED=YES | NO;

for a dBASE file (DBF), indicates whether to write records to the SAS data set that are marked for deletion but have not been purged. You can replace the equal sign with a blank.

Default: NO

GETNAMES=YES | NO;

for spreadsheets and delimited external files, determines whether to generate SAS variable names from the column names in the input file's first row of data. You can replace the equal sign with a blank.

If you specify GETNAMES=NO or if the column names are not valid SAS names, PROC IMPORT uses default variable names. For example, for a delimited file, PROC IMPORT uses VAR1, VAR2, VAR3, and so on.

Note that if a column name contains special characters that are not valid in a SAS name, such as a blank, SAS converts the character to an underscore. For example, the column name **`Occupancy Code`** would become the variable name **`Occupancy_Code`**.

Default: YES

Featured in: Example 1 on page 641 and Example 2 on page 645

MEMOSIZE=*field-length*;

specifies the field length for importing Microsoft Excel 97, 2000, or 2002 Memo fields.

Range: 1 - 32,767

Default: 1024

RANGE="*range-name* | *absolute-range*";

subsets a spreadsheet by identifying the rectangular set of cells to import from the specified spreadsheet. The syntax for *range-name* and *absolute-range* is native to the file being read. You can replace the equal sign with a blank.

range-name is a name that has been assigned to represent a range, such as a range of cells within the spreadsheet.

Limitation: SAS supports range names up to 32 characters. If a range name exceeds 32 characters, SAS will notify you that the name is invalid.

Tip: For Microsoft Excel, range names do not contain special characters such as spaces or hyphens.

absolute-range identifies the top left cell that begins the range and the bottom right cell that ends the range. For Excel 4.0, 5.0, and 7.0 (95), the beginning and ending cells are separated by two periods; that is, **`C9..F12`** specifies a cell range that begins at cell C9, ends at cell F12, and includes all the cells in between. For Excel 97, 2000, and 2002, the beginning and ending cells are separated by a colon – that is, **`C9:F12`**.

Tip: For Excel 97, 2000, and 2002, you can include the spreadsheet name with an absolute range, such as `range="North B$a1:d3"`. If you do not include the spreadsheet name, PROC IMPORT uses the first sheet in the workbook or the spreadsheet name specified with SHEET=.

Default: The entire spreadsheet is selected.

Interaction: For Excel 97, 2000, and 2002 spreadsheets, when RANGE= is specified, a spreadsheet name specified with SHEET= is ignored when the conflict occurs.

SHEET=*spreadsheet-name*;

identifies a particular spreadsheet in a group of spreadsheets. Use this statement with spreadsheets that support multiple spreadsheets within a single file. The naming convention for the spreadsheet name is native to the file being read.

Featured in: Example 2 on page 645

Default: The default depends on the type of spreadsheet. For Excel 4.0 and 5.0, PROC IMPORT reads the first spreadsheet in the file. For Excel 97 and later, PROC IMPORT reads the first spreadsheet from an ascending sort of the spreadsheet names. To be certain that PROC IMPORT reads the desired spreadsheet, you should identify the spreadsheet by specifying SHEET=.

Limitation: SAS supports spreadsheet names up to 31 characters. With the $ appended, the maximum length of a spreadsheet name is 32 characters.

Statements for DBMS Tables

The following data source statements are available to establish a connection to the DBMS when you import a DBMS table.

DATABASE="*database*";

specifies the complete path and filename of the database that contains the specified DBMS table. If the database name does not contain lowercase characters, special characters, or national characters ($, #, or @), you can omit the quotation marks. You can replace the equal sign with a blank.

Note: A default may be configured in the DBMS client software; however, SAS does not generate a default value. △

Featured in: Example 4 on page 647

DBPWD="*database password*";

specifies a password that allows access to a database. You can replace the equal sign with a blank.

Featured in: Example 4 on page 647

MEMOSIZE=*field-length*;

specifies the field length for importing Microsoft Access Memo fields.

Range: 1 - 32,767

Default: 1024

Tip: To prevent Memo fields from being imported, you can specify MEMOSIZE=0.

PWD="*password*";

specifies the user password used by the DBMS to validate a specific userid. If the password does not contain lowercase characters, special characters, or national characters, you can omit the quotation marks. You can replace the equal sign with a blank.

Note: The DBMS client software may default to the userid and password that were used to log in to the operating environment; SAS does not generate a default value. △

Featured in: Example 4 on page 647

UID="*userid*";
: identifies the user to the DBMS. If the userid does not contain lowercase characters, special characters, or national characters, you can omit the quotation marks. You can replace the equal sign with a blank.

Note: The DBMS client software may default to the userid and password that were used to log in to the operating environment; SAS does not generate a default value. △

Featured in: Example 4 on page 647

WGDB="*workgroup-database-name*";
: specifies the workgroup (security) database name that contains the USERID and PWD data for the DBMS. If the workgroup database name does not contain lowercase characters, special characters, or national characters, you can omit the quotation marks. You can replace the equal sign with a blank.

Note: A default workgroup database may be used by the DBMS; SAS does not generate a default value. △

Featured in: Example 4 on page 647

Security Levels for Microsoft Access Tables

Microsoft Access tables have the following levels of security, for which specific combinations of security statements must be used:

None
: Do not specify DBPWD=, PWD=, UID=, or WGDB=.

Password
: Specify only DBPWD=.

User-level
: Specify only PWD=, UID=, and WGDB=.

Full
: Specify DBPWD=, PWD=, UID=, and WGDB=.

Each statement has a default value; however, you may find it necessary to provide a value for each statement explicitly.

Examples: IMPORT Procedure

Example 1: Importing a Delimited External File

Procedure features:

PROC IMPORT statement arguments:

DATAFILE=
OUT=
DBMS=
REPLACE

Data source statements:

DELIMITER=
GETNAMES=

Other features:

PRINT procedure

This example imports the following delimited external file and creates a temporary SAS data set named WORK.MYDATA:

```
Region&State&Month&Expenses&Revenue
Southern&GA&JAN2001&2000&8000
Southern&GA&FEB2001&1200&6000
Southern&FL&FEB2001&8500&11000
Northern&NY&FEB2001&3000&4000
Northern&NY&MAR2001&6000&5000
Southern&FL&MAR2001&9800&13500
Northern&MA&MAR2001&1500&1000
```

Program

Specify the input file.

```
proc import datafile="C:\My Documents\myfiles\delimiter.txt"
```

Identify the output SAS data set.

```
   out=mydata
```

Specify that the input file is a delimited external file.

```
   dbms=dlm
```

Overwrite the data set if it exists.

```
   replace;
```

Specify the delimiter. The DELIMITER= option specifies that an & (ampersand) delimits data fields in the input file. The delimiter separates the columns of data in the input file.

```
   delimiter='&';
```

Generate the variable names from the first row of data in the input file.

```
   getnames=yes;
run;
```

Print the WORK.MYDATA data set. PROC PRINT produces a simple listing.

```
options nodate ps=60 ls=80;

proc print data=mydata;
run;
```

SAS Log

The SAS log displays information about the successful import. For this example, PROC IMPORT generates a SAS DATA step, as shown in the partial log that follows.

```
/**********************************************************************
79    *   PRODUCT:   SAS
80    *   VERSION:   9.00
81    *   CREATOR:   External File Interface
82    *   DATE:      24JAN02
83    *   DESC:      Generated SAS Datastep Code
84    *   TEMPLATE SOURCE:  (None Specified.)
85    ***********************************************************************/
86       data MYDATA                                      ;
87      %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
88       infile 'C:\My Documents\myfiles\delimiter.txt' delimiter = '&' MISSOVER
88 !  DSD lrecl=32767 firstobs=2 ;
89          informat Region $8. ;
90          informat State $2. ;
91          informat Month $7. ;
92          informat Expenses best32. ;
93          informat Revenue best32. ;
94          format Region $8. ;
95          format State $2. ;
96          format Month $7. ;
97          format Expenses best12. ;
98          format Revenue best12. ;
99       input
100                  Region $
101                  State $
102                  Month $
103                  Expenses
104                  Revenue
105      ;
106      if _ERROR_ then call symput('_EFIERR_',1);  /* set ERROR detection
106! macro variable */
107      run;

NOTE: Numeric values have been converted to character
      values at the places given by: (Line):(Column).
      106:44
NOTE: The infile 'C:\My Documents\myfiles\delimiter.txt' is:
      File Name=C:\My Documents\myfiles\delimiter.txt,
      RECFM=V,LRECL=32767

NOTE: 7 records were read from the infile 'C:\My
      Documents\myfiles\delimiter.txt'.
      The minimum record length was 29.
      The maximum record length was 31.
NOTE: The data set WORK.MYDATA has 7 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.05 seconds


7 rows created in MYDATA                                    from C:\My
Documents\myfiles\delimiter.txt.



NOTE: .MYDATA was successfully created.
```

Output

This output lists the output data set, MYDATA, created by PROC IMPORT from the delimited external file.

```
                              The SAS System

      Obs     Region      State     Month          Expenses         Revenue

       1      Southern     GA       JAN2001            2000            8000
       2      Southern     GA       FEB2001            1200            6000
       3      Southern     FL       FEB2001            8500           11000
       4      Northern     NY       FEB2001            3000            4000
       5      Northern     NY       MAR2001            6000            5000
       6      Southern     FL       MAR2001            9800           13500
       7      Northern     MA       MAR2001            1500            1000
```

Example 2: Importing a Specific Spreadsheet from an Excel Workbook

Procedure features:

PROC IMPORT statement arguments:

DATAFILE=
OUT=

Data source statements:

SHEET=
GETNAMES=

Other features:

PRINT procedure option:

OBS=

This example imports a specific spreadsheet from an Excel workbook, which contains multiple spreadsheets, and creates a new, permanent SAS data set named SASUSER.ACCOUNTS.

Program

Specify the input file. The filename contains the extension .XLS, which PROC IMPORT recognizes as identifying an Excel 2000 spreadsheet.

```
proc import datafile="c:\myfiles\Accounts.xls"
```

Identify the output SAS data set.

```
     out=sasuser.accounts;
```

Import only the sheet PRICES that is contained in the file ACCOUNTS.XLS.

```
   sheet='Prices$';
```

Do not generate the variable names from the input file. PROC IMPORT will use default variable names.

```
   getnames=no;
run;
```

Print the SASUSER.ACCOUNTS data set. PROC PRINT produces a simple listing. The OBS= data set option limits the output to the first 10 observations.

```
proc print data=sasuser.accounts(obs=10);
run;
```

Output

The following output displays the first 10 observations of the output data set, SASUSER.ACCOUNTS:

```
                              The SAS System                                1

OBS    F1                                  F2                                F3

  1    Dharamsala Tea                      10 boxes x 20 bags             18.00
  2    Tibetan Barley Beer                 24 - 12 oz bottles             19.00
  3    Licorice Syrup                      12 - 550 ml bottles            10.00
  4    Chef Anton's Cajun Seasoning        48 - 6 oz jars                 22.00
  5    Chef Anton's Gumbo Mix              36 boxes                       21.35
  6    Grandma's Boysenberry Spread        12 - 8 oz jars                 25.00
  7    Uncle Bob's Organic Dried Pears     12 - 1 lb pkgs.                30.00
  8    Northwoods Cranberry Sauce          12 - 12 oz jars                40.00
  9    Mishi Kobe Beef                     18 - 500 g pkgss.              97.00
 10    Fish Roe                            12 - 200 ml jars               31.00
```

Example 3: Importing a Subset of Records from an Excel Spreadsheet

Procedure features:

PROC IMPORT statement arguments:

DATAFILE=
OUT=

This example imports a subset of an Excel spreadsheet and creates a temporary SAS data set. The WHERE= SAS data set option is specified in order to import only a subset of records from the Excel spreadsheet.

Program

Specify the input file.

```
proc import datafile='c:\Myfiles\Class.xls'
```

Identify the output SAS data set, and request that only a subset of the records be imported.

```
   out=work.femaleclass (where=(sex='F'));
run;
```

Print the new SAS data set. PROC PRINT produces a simple listing.

```
proc print data=work.femaleclass;
run;
```

Output

The following output displays the output SAS data set, WORK.FEMALECLASS:

```
                         The SAS System                              1

            Obs    Name       Sex    Age    Height    Weight

             1     Alice       F      13     56.5       84.0
             2     Barbara     F      13     65.3       98.0
             3     Carol       F      14     62.8      102.5
             4     Jane        F      12     59.8       84.5
             5     Janet       F      15     62.5      112.5
             6     Joyce       F      11     51.3       50.5
             7     Judy        F      14     64.3       90.0
             8     Louise      F      12     56.3       77.0
             9     Mary        F      15     66.5      112.0
```

Example 4: Importing a Microsoft Access Table

Procedure features:

PROC IMPORT statement arguments:

TABLE=
OUT=
DBMS=

Data source Statements:

DATABASE=
PWD=
UID=
WGDB=

This example imports a Microsoft Access 97 table and creates a permanent SAS data set named SASUSER.CUST. The Access table has user-level security, so it is necessary to specify values for the PWD=, UID=, and WGDB= statements.

Program

Specify the input DBMS table name.

```
proc import table="customers"
```

Identify the output SAS data set.

```
out=sasuser.cust
```

Specify that the input file is a Microsoft Access 97 table.

```
dbms=access97;
```

Identify the user ID to the DBMS.

```
uid="userid";
```

Specify the DBMS password to access the table.

```
pwd="mypassword";
```

Specify the path and filename of the database that contains the table.

```
database="c:\myfiles\east.mdb";
```

Specify the workgroup (security) database name that contains the user ID and password data for the Microsoft Access table.

```
wgdb="c:\winnt\system32\security.mdb";
```

Print the SASUSER.CUST data set. PROC PRINT produces a simple listing. The OBS= data set option limits the output to the first five observations.

```
proc print data=sasuser.cust(obs=5);
run;
```

Output

The following output displays the first five observations of the output data set, SASUSER.CUST.

```
                                The SAS System                              1

Obs    Name                                  Street                          Zipcode

  1    David Taylor                          124 Oxbow Street                72511
  2    Theo Barnes                           2412 McAllen Avenue             72513
  3    Lydia Stirog                          12550 Overton Place             72516
  4    Anton Niroles                         486 Gypsum Street               72511
  5    Cheryl Gaspar                         36 E. Broadway                  72515
```

CHAPTER

26

The MEANS Procedure

Overview: MEANS Procedure

The MEANS procedure provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations. For example, PROC MEANS

- calculates descriptive statistics based on moments
- estimates quantiles, which includes the median
- calculates confidence limits for the mean
- identifies extreme values
- performs a *t* test.

By default, PROC MEANS displays output. You can also use the OUTPUT statement to store the statistics in a SAS data set.

PROC MEANS and PROC SUMMARY are very similar; see Chapter 42, "The SUMMARY Procedure," on page 1257 for an explanation of the differences.

Output 26.1 on page 650 shows the default output that PROC MEANS displays. The data set that PROC MEANS analyzes contains the integers 1 through 10. The output reports the number of observations, the mean, the standard deviation, the minimum value, and the maximum value. The statements that produce the output follow:

```
proc means data=OnetoTen;
run;
```

Output 26.1 The Default Descriptive Statistics

```
                              The SAS System                              1

                           The MEANS Procedure

                      Analysis Variable : Integer

  N            Mean         Std Dev         Minimum         Maximum
 ------------------------------------------------------------------
 10       5.5000000       3.0276504       1.0000000      10.0000000
 ------------------------------------------------------------------
```

Output 26.2 on page 650 shows the results of a more extensive analysis of two variables, MoneyRaised and HoursVolunteered. The analysis data set contains information about the amount of money raised and the number of hours volunteered by high-school students for a local charity. PROC MEANS uses six combinations of two categorical variables to compute the number of observations, the mean, and the range. The first variable, School, has two values and the other variable, Year, has three values. For an explanation of the program that produces the output, see Example 11 on page 706.

Output 26.2 Specified Statistics for Class Levels and Identification of Maximum Values

```
                  Summary of Volunteer Work by School and Year                1

                              The MEANS Procedure

                         N
School           Year  Obs  Variable            N          Mean         Range
-----------------------------------------------------------------------------
Kennedy          1992   15  MoneyRaised        15    29.0800000    39.7500000
                            HoursVolunteered   15    22.1333333    30.0000000

                 1993   20  MoneyRaised        20    28.5660000    23.5600000
                            HoursVolunteered   20    19.2000000    20.0000000

                 1994   18  MoneyRaised        18    31.5794444    65.4400000
                            HoursVolunteered   18    24.2777778    15.0000000

Monroe           1992   16  MoneyRaised        16    28.5450000    48.2700000
                            HoursVolunteered   16    18.8125000    38.0000000

                 1993   12  MoneyRaised        12    28.0500000    52.4600000
                            HoursVolunteered   12    15.8333333    21.0000000

                 1994   28  MoneyRaised        28    29.4100000    73.5300000
                            HoursVolunteered   28    19.1428571    26.0000000
-----------------------------------------------------------------------------
```

```
             Best Results: Most Money Raised and Most Hours Worked           2

                                           Most     Most      Money     Hours
  Obs  School   Year  _TYPE_  _FREQ_  Cash     Time     Raised  Volunteered

    1              .      0       109   Willard  Tonya     78.65       40
    2            1992     1        31   Tonya    Tonya     55.16       40
    3            1993     1        32   Cameron  Amy       65.44       31
    4            1994     1        46   Willard  L.T.      78.65       33
    5  Kennedy     .      2        53   Luther   Jay       72.22       35
    6  Monroe      .      2        56   Willard  Tonya     78.65       40
    7  Kennedy  1992      3        15   Thelma   Jay       52.63       35
    8  Kennedy  1993      3        20   Bill     Amy       42.23       31
    9  Kennedy  1994      3        18   Luther   Che-Min   72.22       33
   10  Monroe   1992      3        16   Tonya    Tonya     55.16       40
   11  Monroe   1993      3        12   Cameron  Myrtle    65.44       26
   12  Monroe   1994      3        28   Willard  L.T.      78.65       33
```

In addition to the report, the program also creates an output data set (located on page 2 of the output) that identifies the students who raised the most money and who volunteered the most time over all the combinations of School and Year and within the combinations of School and Year:

- The first observation in the data set shows the students with the maximum values overall for MoneyRaised and HoursVolunteered.
- Observations 2 through 4 show the students with the maximum values for each year, regardless of school.
- Observations 5 and 6 show the students with the maximum values for each school, regardless of year.
- Observations 7 through 12 show the students with the maximum values for each school-year combination.

Syntax: MEANS Procedure

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

PROC MEANS <*option(s)*> <*statistic-keyword(s)*>;

BY <DESCENDING> *variable-1* <… <DESCENDING> *variable-n*><NOTSORTED>;

CLASS *variable(s)* </ *option(s)*>;

FREQ *variable*;

ID *variable(s)*;

OUTPUT <OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
<*id-group-specification(s)*> <*maximum-id-specification(s)*>
<*minimum-id-specification(s)*> </ *option(s)*> ;

TYPES *request(s)*;

VAR *variable(s)* < / WEIGHT=*weight-variable*>;

WAYS *list*;

WEIGHT *variable*;

To do this	Use this statement
Calculate separate statistics for each BY group	BY
Identify variables whose values define subgroups for the analysis	CLASS
Identify a variable whose values represent the frequency of each observation	FREQ
Include additional identification variables in the output data set	ID
Create an output data set that contains specified statistics and identification variables	OUTPUT
Identify specific combinations of class variables to use to subdivide the data	TYPES
Identify the analysis variables and their order in the results	VAR
Specify the number of ways to make unique combinations of class variables	WAYS
Identify a variable whose values weight each observation in the statistical calculations	WEIGHT

PROC MEANS Statement

See also: Chapter 42, "The SUMMARY Procedure," on page 1257

PROC MEANS <*option(s)*> <*statistic-keyword(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Disable floating point exception recovery	NOTRAP
Specify the amount of memory to use for data summarization with class variables	SUMSIZE=
Override the SAS system option THREADS | NOTHREADS	THREADS | NOTHREADS
Control the classification levels	
Specify a secondary data set that contains the combinations of class variables to analyze	CLASSDATA=
Create all possible combinations of class variable values	COMPLETETYPES
Exclude from the analysis all combinations of class variable values that are not in the CLASSDATA= data set	EXCLUSIVE
Use missing values as valid values to create combinations of class variables	MISSING
Control the statistical analysis	
Specify the confidence level for the confidence limits	ALPHA=
Exclude observations with nonpositive weights from the analysis	EXCLNPWGTS
Specify the sample size to use for the P2 quantile estimation method	QMARKERS=
Specify the quantile estimation method	QMETHOD=
Specify the mathematical definition used to compute quantiles	QNTLDEF=
Select the statistics	*statistic-keyword*
Specify the variance divisor	VARDEF=
Control the output	
Specify the field width for the statistics	FW=
Specify the number of decimal places for the statistics	MAXDEC=
Suppress reporting the total number of observations for each unique combination of the class variables	NONOBS
Suppress all displayed output	NOPRINT
Order the values of the class variables according to the specified order	ORDER=
Display the output	PRINT

To do this	Use this option
Display the analysis for all requested combinations of class variables	PRINTALLTYPES
Display the values of the ID variables	PRINTIDVARS
Control the output data set	
Specify that the _TYPE_ variable contain character values.	CHARTYPE
Order the output data set by descending _TYPE_ value	DESCENDTYPES
Select ID variables based on minimum values	IDMIN
Limit the output statistics to the observations with the highest _TYPE_ value	NWAY

Options

ALPHA=*value*

specifies the confidence level to compute the confidence limits for the mean. The percentage for the confidence limits is (1–*value*)×100. For example, ALPHA=.05 results in a 95% confidence limit.

Default: .05

Range: between 0 and 1

Interaction: To compute confidence limits specify the *statistic-keyword* CLM, LCLM, or UCLM.

See also: “Confidence Limits” on page 679

Featured in: Example 7 on page 699

CHARTYPE

specifies that the _TYPE_ variable in the output data set is a character representation of the binary value of _TYPE_. The length of the variable equals the number of class variables.

Main discussion: “Output Data Set” on page 682

Interaction: When you specify more than 32 class variables, _TYPE_ automatically becomes a character variable.

Featured in: Example 10 on page 704

CLASSDATA=*SAS-data-set*

specifies a data set that contains the combinations of values of the class variables that must be present in the output. Any combinations of values of the class variables that occur in the CLASSDATA= data set but not in the input data set appear in the output and have a frequency of zero.

Restriction: The CLASSDATA= data set must contain all class variables. Their data type and format must match the corresponding class variables in the input data set.

Interaction: If you use the EXCLUSIVE option, then PROC MEANS excludes any observation in the input data set whose combination of class variables is not in the CLASSDATA= data set.

Tip: Use the CLASSDATA= data set to filter or to supplement the input data set.

Featured in: Example 4 on page 689

COMPLETETYPES
creates all possible combinations of class variables even if the combination does not occur in the input data set.

Interaction: The PRELOADFMT option in the CLASS statement ensures that PROC MEANS writes all user-defined format ranges or values for the combinations of class variables to the output, even when a frequency is zero.

Tip: Using COMPLETETYPES does not increase the memory requirements.

Featured in: Example 6 on page 696

DATA=*SAS-data-set*
identifies the input SAS data set.

Main discussion: "Input Data Sets" on page 19

DESCENDTYPES
orders observations in the output data set by descending _TYPE_ value.

Alias: DESCENDING | DESCEND

Interaction: Descending has no effect if you specify NWAY.

Tip: Use DESCENDTYPES to make the overall total (_TYPE_=0) the last observation in each BY group.

See also: "Output Data Set" on page 682

Featured in: Example 9 on page 702

EXCLNPWGTS
excludes observations with nonpositive weight values (zero or negative) from the analysis. By default, PROC MEANS treats observations with negative weights like those with zero weights and counts them in the total number of observations.

Alias: EXCLNPWGT

See also: WEIGHT= on page 673 and "WEIGHT Statement" on page 674

EXCLUSIVE
excludes from the analysis all combinations of the class variables that are not found in the CLASSDATA= data set.

Requirement: If a CLASSDATA= data set is not specified, then this option is ignored.

Featured in: Example 4 on page 689

FW=*field-width*
specifies the field width to display the statistics in printed or displayed output. FW= has no effect on statistics that are saved in an output data set.

Default: 12

Tip: If PROC MEANS truncates column labels in the output, then increase the field width.

Featured in: Example 1 on page 683, Example 4 on page 689, and Example 5 on page 693

IDMIN
specifies that the output data set contain the minimum value of the ID variables.

Interaction: Specify PRINTIDVARS to display the value of the ID variables in the output.

See also: "ID Statement" on page 665

MAXDEC=*number*
specifies the maximum number of decimal places to display the statistics in the printed or displayed output. MAXDEC= has no effect on statistics that are saved in an output data set.

Default: BEST. width for columnar format, typically about 7.

Range: 0-8

Featured in: Example 2 on page 685 and Example 4 on page 689

MISSING
considers missing values as valid values to create the combinations of class variables. Special missing values that represent numeric values (the letters A through Z and the underscore (_) character) are each considered as a separate value.

Default: If you omit MISSING, then PROC MEANS excludes the observations with a missing class variable value from the analysis.

See also: *SAS Language Reference: Concepts* for a discussion of missing values that have special meaning.

Featured in: Example 6 on page 696

NONOBS
suppresses the column that displays the total number of observations for each unique combination of the values of the class variables. This column corresponds to the _FREQ_ variable in the output data set.

See also: "The N Obs Statistic" on page 681

Featured in: Example 5 on page 693 and Example 6 on page 696

NOPRINT
See PRINT | NOPRINT.

NOTHREADS
See THREADS | NOTHREADS.

NOTRAP
disables floating point exception (FPE) recovery during data processing. By default, PROC MEANS traps these errors and sets the statistic to missing.

In operating environments where the overhead of FPE recovery is significant, NOTRAP can improve performance. Note that normal SAS FPE handling is still in effect so that PROC MEANS terminates in the case of math exceptions.

NWAY
specifies that the output data set contain only statistics for the observations with the highest _TYPE_ and _WAY_ values. When you specify class variables, this corresponds to the combination of all class variables.

Interaction: If you specify a TYPES statement or a WAYS statement, then PROC MEANS ignores this option.

See also: "Output Data Set" on page 682

Featured in: Example 10 on page 704

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED
specifies the sort order to create the unique combinations for the values of the class variables in the output, where

DATA
orders values according to their order in the input data set.

Interaction: If you use PRELOADFMT in the CLASS statement, then the order for the values of each class variable matches the order that PROC FORMAT uses to store the values of the associated user-defined format. If you use the CLASSDATA= option, then PROC MEANS uses the order of the unique values of each class variable in the CLASSDATA= data set to order the output levels. If you use both options, then PROC MEANS first uses the user-defined formats to order the output. If you omit EXCLUSIVE, then PROC MEANS appends after the user-defined format and the CLASSDATA= values the unique values of the class variables in the input data set based on the order in which they are encountered.

Tip: By default, PROC FORMAT stores a format definition in sorted order. Use the NOTSORTED option to store the values or ranges of a user defined format in the order that you define them.

FORMATTED
orders values by their ascending formatted values. This order depends on your operating environment.

Alias: FMT | EXTERNAL

FREQ
orders values by descending frequency count so that levels with the most observations are listed first.

Interaction: For multiway combinations of the class variables, PROC MEANS determines the order of a class variable combination from the individual class variable frequencies.

Interaction: Use the ASCENDING option in the CLASS statement to order values by ascending frequency count.

UNFORMATTED
orders values by their unformatted values, which yields the same order as PROC SORT. This order depends on your operating environment.

Alias: UNFMT | INTERNAL

Default: UNFORMATTED

See also: "Ordering the Class Values" on page 676

PRINT | NOPRINT
specifies whether PROC MEANS displays the statistical analysis. NOPRINT suppresses all the output.

Default: PRINT

Tip: Use NOPRINT when you want to create only an OUT= output data set.

Featured in: For an example of NOPRINT, see Example 8 on page 700 and Example 12 on page 709

PRINTALLTYPES
displays all requested combinations of class variables (all _TYPE_ values) in the printed or displayed output. Normally, PROC MEANS shows only the NWAY type.

Alias: PRINTALL

Interaction: If you use the NWAY option, the TYPES statement, or the WAYS statement, then PROC MEANS ignores this option.

Featured in: Example 4 on page 689

PRINTIDVARS
displays the values of the ID variables in printed or displayed output.

Alias: PRINTIDS

Interaction: Specify IDMIN to display the minimum value of the ID variables.

See also: "ID Statement" on page 665

QMARKERS=*number*
specifies the default number of markers to use for the P^2 quantile estimation method. The number of markers controls the size of fixed memory space.

Default: The default value depends on which quantiles you request. For the median (P50), *number* is 7. For the quartiles (P25 and P50), *number* is 25. For the quantiles P1, P5, P10, P90, P95, or P99, *number* is 105. If you request several quantiles, then PROC MEANS uses the largest value of *number*.

Range: an odd integer greater than 3

Tip: Increase the number of markers above the defaults settings to improve the accuracy of the estimate; reduce the number of markers to conserve memory and computing time.

Main Discussion: "Quantiles" on page 680

QMETHOD=OS|P2|HIST

specifies the method PROC MEANS uses to process the input data when it computes quantiles. If the number of observations is less than or equal to the QMARKERS= value and QNTLDEF=5, then both methods produce the same results.

OS

uses order statistics. This is the same method that PROC UNIVARIATE uses.

Note: This technique can be very memory-intensive. △

P2|HIST

uses the P^2 method to approximate the quantile.

Default: OS

Restriction: When QMETHOD=P2, PROC MEANS will not compute weighted quantiles.

Tip: When QMETHOD=P2, reliable estimations of some quantiles (P1,P5,P95,P99) may not be possible for some data sets.

Main Discussion: "Quantiles" on page 680

QNTLDEF=1|2|3|4|5

specifies the mathematical definition that PROC MEANS uses to calculate quantiles when QMETHOD=OS. To use QMETHOD=P2, you must use QNTLDEF=5.

Default: 5

Alias: PCTLDEF=

Main discussion: "Calculating Percentiles" on page 1528

statistic-keyword(s)

specifies which statistics to compute and the order to display them in the output. The available keywords in the PROC statement are

Descriptive statistic keywords

CLM	RANGE
CSS	SKEWNESS|SKEW
CV	STDDEV|STD
KURTOSIS|KURT	STDERR
LCLM	SUM
MAX	SUMWGT
MEAN	UCLM
MIN	USS
N	VAR
NMISS	

Quantile statistic keywords

MEDIAN | P50	Q3 | P75
P1	P90
P5	P95
P10	P99
Q1 | P25	QRANGE
Hypothesis testing keywords	
PROBT	T

Default: N, MEAN, STD, MIN, and MAX

Requirement: To compute standard error, confidence limits for the mean, and the Student's *t*-test, you must use the default value of the VARDEF= option, which is DF. To compute skewness or kurtosis, you must use VARDEF=N or VARDEF=DF.

Tip: Use CLM or both LCLM and UCLM to compute a two-sided confidence limit for the mean. Use only LCLM or UCLM, to compute a one-sided confidence limit.

Main discussion: The definitions of the keywords and the formulas for the associated statistics are listed in "Keywords and Formulas" on page 1578.

Featured in: Example 1 on page 683 and Example 3 on page 687

SUMSIZE=*value*

specifies the amount of memory that is available for data summarization when you use class variables. *value* may be one of the following:

n | *n*K | *n*M | *n*G

specifies the amount of memory available in bytes, kilobytes, megabytes, or gigabytes, respectively. If *n* is 0, then PROC MEANS use the value of the SAS system option SUMSIZE=.

MAXIMUM | MAX

specifies the maximum amount of memory that is available.

Default: The value of the SUMSIZE= system option.

Tip: For best results, do not make SUMSIZE= larger than the amount of physical memory that is available for the PROC step. If additional space is needed, then PROC MEANS uses utility files.

See also: The SAS system option SUMSIZE= in *SAS Language Reference: Dictionary*.

Main discussion: "Computational Resources" on page 677

THREADS | NOTHREADS

enables or disables parallel processing of the input data set. This option overrides the SAS system option THREADS | NOTHREADS. See *SAS Language Reference: Concepts* for more information about parallel processing.

(SAS 9 Early Adopter Feature)

Default: value of SAS system option THREADS | NOTHREADS.

Interaction: PROC MEANS honors the SAS system option THREADS except when a BY statement is specified or the value of the SAS system option CPUCOUNT is less than 2. You can use THREADS in the PROC MEANS statement to force PROC MEANS to use parallel processing in these situations.

VARDEF=*divisor*

specifies the divisor to use in the calculation of the variance and standard deviation. Table 26.1 on page 660 shows the possible values for *divisor* and associated divisors.

Table 26.1 Possible Values for VARDEF=

Value	Divisor	Formula for Divisor
DF	degrees of freedom	$n - 1$
N	number of observations	n
WDF	sum of weights minus one	$(\Sigma_i w_i) - 1$
WEIGHT | WGT	sum of weights	$\Sigma_i w_i$

The procedure computes the variance as $CSS/divisor$, where CSS is the corrected sums of squares and equals $\sum (x_i - \overline{x})^2$. When you weight the analysis variables, CSS equals $\sum w_i (x_i - \overline{x}_w)^2$, where $\overline{x}_w$ is the weighted mean.

Default: DF

Requirement: To compute the standard error of the mean, confidence limits for the mean, or the Student's *t*-test, use the default value of VARDEF=.

Tip: When you use the WEIGHT statement and VARDEF=DF, the variance is an estimate of σ^2, where the variance of the ith observation is $var(x_i) = \sigma^2/w_i$ and w_i is the weight for the ith observation. This yields an estimate of the variance of an observation with unit weight.

Tip: When you use the WEIGHT statement and VARDEF=WGT, the computed variance is asymptotically (for large n) an estimate of $\sigma^2/\overline{w}$, where $\overline{w}$ is the average weight. This yields an asymptotic estimate of the variance of an observation with average weight.

See also: "Weighted Statistics Example" on page 60

Main discussion: "Keywords and Formulas" on page 1578

BY Statement

Produces separate statistics for each BY group.

Main discussion: "BY" on page 54

See also: "Comparison of the BY and CLASS Statements" on page 664

Featured in: Example 3 on page 687

BY <DESCENDING> *variable-1* <...<DESCENDING> *variable-n*> <NOTSORTED>;

Required Arguments

variable

specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you omit the NOTSORTED option in the BY statement, then the observations in the data set either must be sorted by all the variables that you specify or must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The observations are sorted in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, then the procedure treats each contiguous set as a separate BY group.

Using the BY Statement with the SAS System Option NOBYLINE

If you use the BY statement with the SAS system option NOBYLINE, which suppresses the BY line that normally appears in output that is produced with BY-group processing, then PROC MEANS always starts a new page for each BY group. This behavior ensures that if you create customized BY lines by putting BY-group information in the title and suppressing the default BY lines with NOBYLINE, then the information in the titles matches the report on the pages. (See "Creating Titles That Contain BY-Group Information" on page 19 and "Suppressing the Default BY Line" on page 19.)

CLASS Statement

Specifies the variables whose values define the subgroup combinations for the analysis.

Tip: You can use multiple CLASS statements.

Tip: Some CLASS statement options are also available in the PROC MEANS statement. They affect all CLASS variables. Options that you specify in a CLASS statement apply only to the variables in that CLASS statement.

See also: For information about how the CLASS statement groups formatted values, see "Formatted Values" on page 25.

Featured in: Example 2 on page 685, Example 4 on page 689, Example 5 on page 693, Example 6 on page 696, and Example 10 on page 704

CLASS *variable(s)* </ *options*>;

Required Arguments

variable(s)
specifies one or more variables that the procedure uses to group the data. Variables in a CLASS statement are referred to as *class variables*. Class variables are numeric

or character. Class variables can have continuous values, but they typically have a few discrete values that define levels of the variable. You do not have to sort the data by class variables.

Interaction: Use the TYPES statement or the WAYS statement to control which class variables that PROC MEANS uses to group the data.

Tip: To reduce the number of class variable levels, use a FORMAT statement to combine variable values. When a format combines several internal values into one formatted value, PROC MEANS outputs the lowest internal value.

See also: "Using Class Variables" on page 675

Options

ASCENDING

specifies to sort the class variable levels in ascending order.

Alias: ASCEND

Interaction: PROC MEANS issues a warning message if you specify both ASCENDING and DESCENDING and ignores both options.

Featured in: Example 10 on page 704

DESCENDING

specifies to sort the class variable levels in descending order.

Alias: DESCEND

Interaction: PROC MEANS issues a warning message if you specify both ASCENDING and DESCENDING and ignores both options.

EXCLUSIVE

excludes from the analysis all combinations of the class variables that are not found in the preloaded range of user-defined formats.

Requirement: You must specify PRELOADFMT to preload the class variable formats.

Featured in: Example 6 on page 696

GROUPINTERNAL

specifies not to apply formats to the class variables when PROC MEANS groups the values to create combinations of class variables.

Interaction: If you specify the PRELOADFMT option, then PROC MEANS ignores the GROUPINTERNAL option and uses the formatted values.

Interaction: If you specify the ORDER=FORMATTED option, then PROC MEANS ignores the GROUPINTERNAL option and uses the formatted values.

Tip: This option saves computer resources when the numeric class variables contain discrete values.

See also: "Computer Resources" on page 665

MISSING

considers missing values as valid values for the class variable levels. Special missing values that represent numeric values (the letters A through Z and the underscore (_) character) are each considered as a separate value.

Default: If you omit MISSING, then PROC MEANS excludes the observations with a missing class variable value from the analysis.

See also: *SAS Language Reference: Concepts* for a discussion of missing values with special meanings.

Featured in: Example 10 on page 704

MLF

enables PROC MEANS to use the primary and secondary format labels for a given range or overlapping ranges to create subgroup combinations when a multilabel format is assigned to a class variable.

Requirement: You must use PROC FORMAT and the MULTILABEL option in the VALUE statement to create a multilabel format.

Interaction: If you use the OUTPUT statement with MLF, then the class variable contains a character string that corresponds to the formatted value. Because the formatted value becomes the internal value, the length of this variable is the number of characters in the longest format label.

Interaction: Using MLF with ORDER=FREQ may not produce the order that you expect for the formatted values.

Tip: If you omit MLF, then PROC MEANS uses the primary format labels, which corresponds to using the first external format value, to determine the subgroup combinations.

See also: The MULTILABEL option in the VALUE statement of the FORMAT procedure on page 451.

Featured in: Example 5 on page 693

Note: When the formatted values overlap, one internal class variable value maps to more than one class variable subgroup combination. Therefore, the sum of the N statistics for all subgroups is greater than the number of observations in the data set (the overall N statistic). △

ORDER=DATA | FORMATTED | FREQ | UNFORMATTED

specifies the order to group the levels of the class variables in the output, where

DATA

orders values according to their order in the input data set.

Interaction: If you use PRELOADFMT, then the order of the values of each class variable matches the order that PROC FORMAT uses to store the values of the associated user-defined format. If you use the CLASSDATA= option in the PROC statement, then PROC MEANS uses the order of the unique values of each class variable in the CLASSDATA= data set to order the output levels. If you use both options, then PROC MEANS first uses the user-defined formats to order the output. If you omit EXCLUSIVE in the PROC statement, then PROC MEANS appends after the user-defined format and the CLASSDATA= values the unique values of the class variables in the input data set based on the order in which they are encountered.

Tip: By default, PROC FORMAT stores a format definition in sorted order. Use the NOTSORTED option to store the values or ranges of a user defined format in the order that you define them.

Featured in: Example 10 on page 704

FORMATTED

orders values by their ascending formatted values. This order depends on your operating environment. If no format has been assigned to a class variable, then the default format, BEST12., is used.

Alias: FMT | EXTERNAL

Featured in: Example 5 on page 693

FREQ

orders values by descending frequency count so that levels with the most observations are listed first.

Interaction: For multiway combinations of the class variables, PROC MEANS determines the order of a level from the individual class variable frequencies.

Interaction: Use the ASCENDING option to order values by ascending frequency count.

Featured in: Example 5 on page 693

UNFORMATTED
orders values by their unformatted values, which yields the same order as PROC SORT. This order depends on your operating environment. This sort sequence is particularly useful for displaying dates chronologically.

Alias: UNFMT | INTERNAL

Default: UNFORMATTED

Tip: By default, all orders except FREQ are ascending. For descending orders, use the DESCENDING option.

See also: "Ordering the Class Values" on page 676

PRELOADFMT
specifies that all formats are preloaded for the class variables.

Requirement: PRELOADFMT has no effect unless you specify either COMPLETETYPES, EXCLUSIVE, or ORDER=DATA and you assign formats to the class variables.

Interaction: To limit PROC MEANS output to the combinations of formatted class variable values present in the input data set, use the EXCLUSIVE option in the CLASS statement.

Interaction: To include all ranges and values of the user-defined formats in the output, even when the frequency is zero, use COMPLETETYPES in the PROC statement.

Featured in: Example 6 on page 696

Comparison of the BY and CLASS Statements

Using the BY statement is similar to using the CLASS statement and the NWAY option in that PROC MEANS summarizes each BY group as an independent subset of the input data. Therefore, no overall summarization of the input data is available. However, unlike the CLASS statement, the BY statement requires that you previously sort BY variables.

When you use the NWAY option, PROC MEANS might encounter insufficient memory for the summarization of all the class variables. You can move some class variables to the BY statement. For maximum benefit, move class variables to the BY statement that are already sorted or that have the greatest number of unique values.

You can use the CLASS and BY statements together to analyze the data by the levels of class variables within BY groups. See Example 3 on page 687.

How PROC MEANS Handles Missing Values for Class Variables

By default, if an observation contains a missing value for any class variable, then PROC MEANS excludes that observation from the analysis. If you specify the MISSING option in the PROC statement, then the procedure considers missing values as valid levels for the combination of class variables.

Specifying the MISSING option in the CLASS statement allows you to control the acceptance of missing values for individual class variables.

Computer Resources

The total of unique class values that PROC MEANS allows depends on the amount of computer memory that is available. See "Computational Resources" on page 677 for more information.

The GROUPINTERNAL option can improve computer performance because the grouping process is based on the internal values of the class variables. If a numeric class variable is not assigned a format and you do not specify GROUPINTERNAL, then PROC MEANS uses the default format, BEST12., to format numeric values as character strings. Then PROC MEANS groups these numeric variables by their character values, which takes additional time and computer memory.

FREQ Statement

Specifies a numeric variable that contains the frequency of each observation.

Main discussion: "FREQ" on page 56

FREQ *variable*;

Required Arguments

variable

specifies a numeric variable whose value represents the frequency of the observation. If you use the FREQ statement, then the procedure assumes that each observation represents *n* observations, where *n* is the value of *variable*. If *n* is not an integer, then SAS truncates it. If *n* is less than 1 or is missing, then the procedure does not use that observation to calculate statistics.

The sum of the frequency variable represents the total number of observations.

Note: The FREQ variable does not affect how PROC MEANS identifies multiple extremes when you use the IDGROUP syntax in the OUTPUT statement. △

ID Statement

Includes additional variables in the output data set.

See Also: Discussion of *id-group-specification* in "OUTPUT Statement" on page 666.

ID *variable(s)*;

Required Arguments

variable(s)

identifies one or more variables from the input data set whose maximum values for groups of observations PROC MEANS includes in the output data set.

Interaction: Use IDMIN in the PROC statement to include the minimum value of the ID variables in the output data set.

Tip: Use the PRINTIDVARS option in the PROC statement to include the value of the ID variable in the displayed output.

Selecting the Values of the ID Variables

When you specify only one variable in the ID statement, the value of the ID variable for a given observation is the maximum (minimum) value found in the corresponding group of observations in the input data set. When you specify multiple variables in the ID statement, PROC MEANS selects the maximum value by processing the variables in the ID statement in the order that you list them. PROC MEANS determines which observation to use from all the ID variables by comparing the values of the first ID variable. If more than one observation contains the same maximum (minimum) ID value, then PROC MEANS uses the second and subsequent ID variable values as “tiebreakers.” In any case, all ID values are taken from the same observation for any given BY group or classification level within a type.

See “Sorting Orders for Character Variables” on page 1101 for information on how PROC MEANS compares character values to determine the maximum value.

OUTPUT Statement

Writes statistics to a new SAS data set.

Tip: You can use multiple OUTPUT statements to create several OUT= data sets.

Featured in: Example 8 on page 700, Example 9 on page 702, Example 10 on page 704, Example 11 on page 706, and Example 12 on page 709

OUTPUT <OUT=*SAS-data-set*> <*output-statistic-specification(s)*>
<*id-group-specification(s)*> <*maximum-id-specification(s)*>
<*minimum-id-specification(s)*> </ *option(s)*>;

Options

OUT=*SAS-data-set*
names the new output data set. If *SAS-data-set* does not exist, then PROC MEANS creates it. If you omit OUT=, then the data set is named DATA*n*, where *n* is the smallest integer that makes the name unique.

Default: DATA*n*

Tip: You can use data set options with the OUT= option. See “Data Set Options” on page 17 for a list.

output-statistic-specification(s)
specifies the statistics to store in the OUT= data set and names one or more variables that contain the statistics. The form of the *output-statistic-specification* is

statistic-keyword<(*variable-list*)>=<*name(s)*>

where

statistic-keyword
specifies which statistic to store in the output data set. The available statistic keywords are

Descriptive statistics keyword	
CSS	RANGE
CV	SKEWNESS | SKEW
KURTOSIS | KURT	STDDEV | STD
LCLM	STDERR
MAX	SUM
MEAN	SUMWGT
MIN	UCLM
N	USS
NMISS	VAR
Quantile statistics keyword	
MEDIAN | P50	Q3 | P75
P1	P90
P5	P95
P10	P99
Q1 | P25	QRANGE
Hypothesis testing keyword	
PROBT	T

By default the statistics in the output data set automatically inherit the analysis variable's format, informat, and label. However, statistics computed for N, NMISS, SUMWGT, USS, CSS, VAR, CV, T, PROBT, SKEWNESS, and KURTOSIS will not inherit the analysis variable's format because this format may be invalid for these statistics (for example, dollar or datetime formats).

Restriction: If you omit *variable* and *name(s)*, then PROC MEANS allows the *statistic-keyword* only once in a single OUTPUT statement, unless you also use the AUTONAME option.

Featured in: Example 8 on page 700, Example 9 on page 702, Example 11 on page 706, and Example 12 on page 709

variable-list
specifies the names of one or more numeric analysis variables whose statistics you want to store in the output data set.

Default: all numeric analysis variables

name(s)
specifies one or more names for the variables in output data set that will contain the analysis variable statistics. The first name contains the statistic for the first analysis variable; the second name contains the statistic for the second analysis variable; and so on.

Default: the analysis variable name. If you specify AUTONAME, then the default is the combination of the analysis variable name and the *statistic-keyword*.

Interaction: If you specify *variable-list*, then PROC MEANS uses the order in which you specify the analysis variables to store the statistics in the output data set variables.

Featured in: Example 8 on page 700

Default: If you use the CLASS statement and an OUTPUT statement without an *output-statistic-specification*, then the output data set contains five observations for each combination of class variables: the value of N, MIN, MAX, MEAN, and STD. If you use the WEIGHT statement or the WEIGHT option in the VAR statement, then the output data set also contains an observation with the sum of weights (SUMWGT) for each combination of class variables.

Tip: Use the AUTONAME option to have PROC MEANS generate unique names for multiple variables and statistics.

id-group-specification

combines the features and extends the ID statement, the IDMIN option in the PROC statement, and the MAXID and MINID options in the OUTPUT statement to create an OUT= data set that identifies multiple extreme values. The form of the *id-group-specification* is

IDGROUP (<MIN|MAX (*variable-list-1*) <…MIN|MAX (*variable-list-n*)>>
<<MISSING> <OBS> <LAST>> OUT <[*n*]>
(*id-variable-list*)=<*name(s)*>)

MIN|MAX(*variable-list*)

specifies the selection criteria to determine the extreme values of one or more input data set variables specified in *variable-list*. Use MIN to determine the minimum extreme value and MAX to determine the maximum extreme value.

When you specify multiple selection variables, the ordering of observations for the selection of *n* extremes is done the same way that PROC SORT sorts data with multiple BY variables. PROC MEANS concatenates the variable values into a single key. The MAX(*variable-list*) selection criterion is similar to using PROC SORT and the DESCENDING option in the BY statement.

Default: If you do not specify MIN or MAX, then PROC MEANS uses the observation number as the selection criterion to output observations.

Restriction: If you specify criteria that are contradictory, then PROC MEANS uses only the first selection criterion.

Interaction: When multiple observations contain the same extreme values in all the MIN or MAX variables, PROC MEANS uses the observation number to resolve which observation to write to the output. By default, PROC MEANS uses the first observation to resolve any ties. However, if you specify the LAST option, then PROC MEANS uses the last observation to resolve any ties.

LAST

specifies that the OUT= data set contains values from the last observation (or the last *n* observations, if *n* is specified). If you do not specify LAST, then the OUT= data set contains values from the first observation (or the first *n* observations, if *n* is specified). The OUT= data set might contain several observations because in addition to the value of the last (first) observation, the OUT= data set contains values from the last (first) observation of each subgroup level that is defined by combinations of class variable values.

Interaction: When you specify MIN or MAX and when multiple observations contain the same extreme values, PROC MEANS uses the observation number to resolve which observation to save to the OUT= data set. If you specify LAST, then PROC MEANS uses the later observations to resolve any ties. If you do not specify LAST, then PROC MEANS uses the earlier observations to resolve any ties.

MISSING
: specifies that missing values be used in selection criteria.

  Alias: MISS

OBS
: includes an _OBS_ variable in the OUT= data set that contains the number of the observation in the input data set where the extreme value was found.

  Interaction: If you use WHERE processing, then the value of _OBS_ might not correspond to the location of the observation in the input data set.

  Interaction: If you use [*n*] to write multiple extreme values to the output, then PROC MEANS creates *n* _OBS_ variables and uses the suffix *n* to create the variable names, where *n* is a sequential integer from 1 to *n*.

[*n*]
: specifies the number of extreme values for each variable in *id-variable-list* to include in the OUT= data set. PROC MEANS creates *n* new variables and uses the suffix _*n* to create the variable names, where *n* is a sequential integer from 1 to *n*.

  By default, PROC MEANS determines one extreme value for each level of each requested type. If *n* is greater than one, then *n* extremes are output for each level of each type. When *n* is greater than one and you request extreme value selection, the time complexity is $\mathrm{O}\left(T * N \log_2 n\right)$, where T is the number of types requested and N is the number of observations in the input data set. By comparison, to group the entire data set, the time complexity is $\mathrm{O}\left(N \log_2 N\right)$.

  Default: 1

  Range: an integer between 1 and 100

  Example: To output two minimum extreme values for each variable, use

```
idgroup(min(x) out[2](x y z)=MinX MinY MinZ);
```

  The OUT= data set contains the variables MinX_1, MinX_2, MinY_1, MinY_2, MinZ_1, and MinZ_2.

(*id-variable-list*)
: identifies one or more input data set variables whose values PROC MEANS includes in the OUT= data set. PROC MEANS determines which observations to output by the selection criteria that you specify (MIN, MAX, and LAST).

name(s)
: specifies one or more names for variables in the OUT= data set.

  Default: If you omit *name*, then PROC MEANS uses the names of variables in the *id-variable-list*.

  Tip: Use the AUTONAME option to automatically resolve naming conflicts.

Alias: IDGRP

Requirement: You must specify the MIN|MAX selection criteria first and OUT(*id-variable-list*)= after the suboptions MISSING, OBS, and LAST.

Tip: You can use *id-group-specification* to mimic the behavior of the ID statement and a *maximum-id-specification* or *mimimum-id-specification* in the OUTPUT statement.

Tip: When you want the output data set to contain extreme values along with other id variables, it is more efficient to include them in the *id-variable-list* than to request separate statistics. For example, the statement

```
output idgrp(max(x) out(x a b)= );
```

is more efficient than the statement

```
output idgrp(max(x) out(a b)= ) max(x)=;
```

Featured in: Example 8 on page 700 and Example 12 on page 709

CAUTION:
The IDGROUP syntax allows you to create output variables with the same name. When this happens, only the first variable appears in the output data set. Use the AUTONAME option to automatically resolve these naming conflicts. △

Note: If you specify fewer new variable names than the combination of analysis variables and identification variables, then the remaining output variables use the corresponding names of the ID variables as soon as PROC MEANS exhausts the list of new variable names. △

maximum-id-specification(s)
specifies that one or more identification variables be associated with the maximum values of the analysis variables. The form of the *maximum-id-specification* is

MAXID <(*variable-1* <(*id-variable-list-1*)> <…*variable-n*
<(*id-variable-list-n*)>>)> = *name(s)*

variable
identifies the numeric analysis variable whose maximum values PROC MEANS determines. PROC MEANS may determine several maximum values for a variable because, in addition to the overall maximum value, subgroup levels, which are defined by combinations of class variables values, also have maximum values.

Tip: If you use an ID statement and omit *variable*, then PROC MEANS uses all analysis variables.

id-variable-list
identifies one or more variables whose values identify the observations with the maximum values of the analysis variable.

Default: the ID statement variables

name(s)
specifies the names for new variables that contain the values of the identification variable associated with the maximum value of each analysis variable.

Tip: If you use an ID statement, and omit *variable* and *id-variable*, then PROC MEANS associates all ID statement variables with each analysis variable. Thus, for each analysis variable, the number of variables that are created in the output data set equals the number of variables that you specify in the ID statement.

Tip: Use the AUTONAME option to automatically resolve naming conflicts.

Limitation: If multiple observations contain the maximum value within a class level, then PROC MEANS saves the value of the ID variable for only the first of those observations in the output data set.

Featured in: Example 11 on page 706

CAUTION:
The MAXID syntax allows you to create output variables with the same name. When this happens, only the first variable appears in the output data set. Use the AUTONAME option to automatically resolve these naming conflicts. △

Note: If you specify fewer new variable names than the combination of analysis variables and identification variables, then the remaining output variables use the corresponding names of the ID variables as soon as PROC MEANS exhausts the list of new variable names. △

minid-specification
See the description of maximum-id-specification on page 670. This option behaves in exactly the same way, except that PROC MEANS determines the minimum values instead of the maximum values. The form of the *minid-specification* is

MINID<(*variable-1* <(*id-variable-list-1*)> <…*variable-n* <(*id-variable-list-n*)>>)> = *name(s)*

AUTOLABEL
specifies that PROC MEANS appends the statistic name to the end of the variable label. If an analysis variable has no label, then PROC MEANS creates a label by appending the statistic name to the analysis variable name.

Featured in: Example 12 on page 709

AUTONAME
specifies that PROC MEANS creates a unique variable name for an output statistic when you do not explicitly assign the variable name in the OUTPUT statement. This is accomplished by appending the *statistic-keyword* to the end of the input variable name from which the statistic was derived. For example, the statement

```
output min(x)=/autoname;
```

produces the x_Min variable in the output data set.

AUTONAME activates the SAS internal mechanism to automatically resolve conflicts in the variable names in the output data set. Duplicate variables will not generate errors. As a result, the statement

```
output min(x)= min(x)=/autoname;
```

produces two variables, x_Min and x_Min2, in the output data set.

Featured in: Example 12 on page 709

KEEPLEN
specifies that statistics in the output data set inherit the length of the analysis variable that PROC MEANS uses to derive them.

CAUTION:
You permanently lose numeric precision when the length of the analysis variable causes PROC MEANS to truncate or round the value of the statistic. However, the precision of the statistic will match that of the input. △

LEVELS
includes a variable named _LEVEL_ in the output data set. This variable contains a value from 1 to *n* that indicates a unique combination of the values of class variables (the values of _TYPE_ variable).

Main discussion: "Output Data Set" on page 682

Featured in: Example 8 on page 700

NOINHERIT
specifies that the variables in the output data set that contain statistics do not inherit the attributes (label and format) of the analysis variables which are used to derive them.

Tip: By default, the output data set includes an output variable for each analysis variable and for five observations that contain N, MIN, MAX, MEAN, and STDDEV. Unless you specify NOINHERIT, this variable inherits the format of the analysis variable, which may be invalid for the N statistic (for example, datetime formats).

WAYS
includes a variable named _WAY_ in the output data set. This variable contains a value from 1 to the maximum number of class variables that indicates how many class variables PROC MEANS combines to create the TYPE value.

Main discussion: "Output Data Set" on page 682

See also: "WAYS Statement" on page 674

Featured in: Example 8 on page 700

TYPES Statement

Identifies which of the possible combinations of class variables to generate.

Main discussion: "Output Data Set" on page 682

Requirement: CLASS statement

Featured in: Example 2 on page 685, Example 5 on page 693, and Example 12 on page 709

TYPES *request(s)*;

Required Arguments

request(s)
specifies which of the 2^k combinations of class variables PROC MEANS uses to create the types, where k is the number of class variables. A request is composed of one class variable name, several class variable names separated by asterisks, or ().

To request class variable combinations quickly, use a grouping syntax by placing parentheses around several variables and joining other variables or variable combinations. For example, the following statements illustrate grouping syntax:

Request	Equivalent to
`types A*(B C);`	`types A*B A*C;`
`types (A  B)*(C D);`	`types A*C A*D B*C B*D;`
`types (A  B C)*D;`	`types A*D B*D C*D;`

Interaction The CLASSDATA= option places constraints on the NWAY type. PROC MEANS generates all other types as if derived from the resulting NWAY type.

Tip: Use ( )to request the overall total (_TYPE_=0).

Tip: If you do not need all types in the output data set, then use the TYPES statement to specify particular subtypes rather than applying a WHERE clause to the data set. Doing so saves time and computer memory.

VAR Statement

Identifies the analysis variables and their order in the output.

Default: If you omit the VAR statement, then PROC MEANS analyzes all numeric variables that are not listed in the other statements. When all variables are character variables, PROC MEANS produces a simple count of observations.

Tip: You can use multiple VAR statements.

See also: Chapter 42, “The SUMMARY Procedure,” on page 1257

Featured in: Example 1 on page 683

VAR *variable(s)* </ WEIGHT=*weight-variable*>;

Required Arguments

variable(s)

identifies the analysis variables and specifies their order in the results.

Option

WEIGHT=*weight-variable*

specifies a numeric variable whose values weight the values of the variables that are specified in the VAR statement. The variable does not have to be an integer. If the value of the weight variable is

Weight value...	PROC MEANS...
0	counts the observation in the total number of observations
less than 0	converts the value to zero and counts the observation in the total number of observations
missing	excludes the observation

To exclude observations that contain negative and zero weights from the analysis, use EXCLNPWGT. Note that most SAS/STAT procedures, such as PROC GLM, exclude negative and zero weights by default.

The weight variable does not change how the procedure determines the range, extreme values, or number of missing values.

Restriction: To compute weighted quantiles, use QMETHOD=OS in the PROC statement.

Restriction: Skewness and kurtosis are not available with the WEIGHT option.

Tip: When you use the WEIGHT option, consider which value of the VARDEF= option is appropriate. See the discussion of VARDEF= on page 659.

Tip: Use the WEIGHT option in multiple VAR statements to specify different weights for the analysis variables.

Note: Prior to Version 7 of SAS, the procedure did not exclude the observations with missing weights from the count of observations. △

WAYS Statement

Specifies the number of ways to make unique combinations of class variables.

Tip: Use the TYPES statement to specify additional combinations of class variables.

Featured in: Example 6 on page 696

WAYS *list*;

Required Arguments

list

specifies one or more integers that define the number of class variables to combine to form all the unique combinations of class variables. For example, you can specify 2 for all possible pairs and 3 for all possible triples. The *list* can be specified in the following ways:

m

m1 m2 … mn

m1,m2,…,mn

m TO *n* <BY *increment*>

m1,m2, TO *m3* <BY *increment*>, *m4*

Range: 0 to maximum number of class variables

Example: To create the two-way types for the classification variables A, B, and C, use

```
class A B C ;
ways 2;
```

This WAYS statement is equivalent to specifying a*b, a*c, and b*c in the TYPES statement.

See also: WAYS option on page 672

WEIGHT Statement

Specifies weights for observations in the statistical calculations.

See also: For information on how to calculate weighted statistics and for an example that uses the WEIGHT statement, see "WEIGHT" on page 59

WEIGHT *variable*;

Required Arguments

variable

specifies a numeric variable whose values weight the values of the analysis variables. The values of the variable do not have to be integers. If the value of the weight variable is

Weight value…	PROC MEANS…
0	counts the observation in the total number of observations
less than 0	converts the value to zero and counts the observation in the total number of observations
missing	excludes the observation

To exclude observations that contain negative and zero weights from the analysis, use EXCLNPWGT. Note that most SAS/STAT procedures, such as PROC GLM, exclude negative and zero weights by default.

Restriction: To compute weighted quantiles, use QMETHOD=OS in the PROC statement.

Restriction: Skewness and kurtosis are not available with the WEIGHT statement.

Interaction: If you use the WEIGHT= option in a VAR statement to specify a weight variable, then PROC MEANS uses this variable instead to weight those VAR statement variables.

Tip: When you use the WEIGHT statement, consider which value of the VARDEF= option is appropriate. See the discussion of VARDEF= on page 659 and the calculation of weighted statistics in "Keywords and Formulas" on page 1578 for more information.

Note: Prior to Version 7 of SAS, the procedure did not exclude the observations with missing weights from the count of observations. △

Concepts: MEANS Procedure

Using Class Variables

The TYPES statement controls which of the available class variables PROC MEANS uses to subgroup the data. The unique combinations of these active class variable values that occur together in any single observation of the input data set determine the data subgroups. Each subgroup that PROC MEANS generates for a given type is called a *level* of that type. Note that for all types, the inactive class variables can still affect the total observation count of the rejection of observations with missing values.

When you use a WAYS statement, PROC MEANS generates types that correspond to every possible unique combination of *n* class variables chosen from the complete set of class variables. For example

```
proc means;
 class a b c d e;
```

```
 ways 2 3;
 run;
```

is equivalent to

```
proc means;
 class a b c d e;
 types a*b a*c a*d a*e b*c b*d b*e c*d c*e d*e
       a*b*c a*b*d a*b*e a*c*d a*c*e a*d*e
       b*c*d b*c*e c*d*e;
 run;
```

If you omit the TYPES statement and the WAYS statement, then PROC MEANS uses all class variables to subgroup the data (the NWAY type) for displayed output and computes all types (2^k) for the output data set.

Ordering the Class Values

PROC MEANS determines the order of each class variable in any type by examining the order of that class variable in the corresponding one-way type. You see the effect of this behavior in the options ORDER=DATA or ORDER=FREQ. When PROC MEANS subdivides the input data set into subsets, the classification process does not apply the options ORDER=DATA or ORDER=FREQ independently for each subgroup. Instead, one frequency and data order is established for all output based on an nonsubdivided view of the entire data set. For example, consider the following statements:

```
data pets;
 input Pet $ Gender $;
 datalines;
dog  m
dog  f
dog  f
dog  f
cat  m
cat  m
cat  f
;

proc means data=pets order=freq;
   class pet gender;
run;
```

The statements produce this output.

```
                          The SAS System                              1

                       The MEANS Procedure

                                              N
               Pet          Gender          Obs
               ----------------------------------
               dog          f                 3

                            m                 1

               cat          f                 1

                            m                 2
               ----------------------------------
```

In the example, PROC MEANS does not list male cats before female cats. Instead, it determines the order of gender for all types over the entire data set. PROC MEANS found more observations for female pets (f=4, m=3).

Computational Resources

PROC MEANS employs the same memory allocation scheme across all operating environments. When class variables are involved, PROC MEANS must keep a copy of each unique value of each class variable in memory. You can estimate the memory requirements to group the class variable by calculating

$$Nc_1\,(Lc_1 + K) + Nc_2\,(Lc_2 + K) + \ldots + Nc_n\,(Lc_n + K)$$

where

Nc_i is the number of unique values for the class variable

Lc_i is the combined unformatted and formatted length of c_i

K is some constant on the order of 32 bytes (64 for 64-bit architectures).

When you use the GROUPINTERNAL option in the CLASS statement, Lc_i is simply the unformatted length of c_i.

Each unique combination of class variables, $c_{1_i}\ c_{2_j}$, for a given type forms a level in that type (see "TYPES Statement" on page 672). You can estimate the maximum potential space requirements for all levels of a given type, when all combinations actually exist in the data (a complete type), by calculating

$$W * Nc_1 * Nc_2 * \ldots * Nc_n$$

where

W is a constant based on the number of variables analyzed and the number of statistics calculated (unless you request QMETHOD=OS to compute the quantiles).

$Nc_1 \ldots Nc_n$ are the number of unique levels for the active class variables of the given type.

Clearly, the memory requirements of the levels overwhelm those of the class variables. For this reason, PROC MEANS may open one or more utility files and write the levels of one or more types to disk. These types are either the primary types that PROC MEANS built during the input data scan or the derived types.

If PROC MEANS must write partially complete primary types to disk while it processes input data, then one or more merge passes may be required to combine type levels in memory with those on disk. In addition, if you use an order other than DATA for any class variable, then PROC MEANS groups the completed types on disk. For this reason, the peak disk space requirements can be more than twice the memory requirements for a given type.

When PROC MEANS uses a temporary work file, you will receive the following note in the SAS log:

```
Processing on disk occurred during summarization.
Peak disk usage was approximately nnn Mbytes.
Adjusting SUMSIZE may improve performance.
```

In most cases processing ends normally.

When you specify class variables in a CLASS statement, the amount of data-dependent memory that PROC MEANS uses before it writes to a utility file is controlled by the SAS system option and PROC option SUMSIZE=. Like the system option SORTSIZE=, SUMSIZE= sets the memory threshold where disk-based operations begin. For best results, set SUMSIZE= to less than the amount of real memory that is likely to be available for the task. For efficiency reasons, PROC MEANS may internally round up the value of SUMSIZE=. SUMSIZE= has no effect unless you specify class variables.

As an alternative, you can set the SAS system option REALMEMSIZE= in the same way that you would set SUMSIZE=. The value of REALMEMSIZE= indicates the amount of real (as opposed to virtual) memory that SAS can expect to allocate. PROC MEANS determines how much data-dependent memory to use before writing to utility files by calculating the lesser of these two values:

- the value of REALMEMSIZE=
- 0.8*(M-U), where M is the value of MEMSIZE= and U is the amount of memory that is already in use.

Operating Environment Information: The REALMEMSIZE= SAS system option is not available in all operating environments. For details, see the SAS Companion for your operating environment. △

If PROC MEANS reports that there is insufficient memory, then increase SUMSIZE= (or REALMEMSIZE=). A SUMSIZE= (or REALMEMSIZE=) value that is greater than MEMSIZE= will have no effect. Therefore, you might also need to increase MEMSIZE=. If PROC MEANS reports insufficient disk space, then increase the WORK space allocation. See the SAS documentation for your operating environment for more information on how to adjust your computation resource parameters.

Another way to enhance performance is by carefully applying the TYPES or WAYS statement, limiting the computations to only those combinations of class variables that you are interested in. In particular, significant resource savings can be achieved by not requesting the combination of all class variables.

Statistical Computations: MEANS Procedure

PROC MEANS uses single-pass algorithms to compute the moment statistics (such as mean, variance, skewness, and kurtosis). See “Keywords and Formulas” on page 1578 for the statistical formulas.

The computational details for confidence limits, hypothesis test statistics, and quantile statistics follow.

Confidence Limits

With the keywords CLM, LCLM, and UCLM, you can compute confidence limits for the mean. A *confidence limit* is a range, constructed around the value of a sample statistic, that contains the corresponding true population value with given probability (ALPHA=) in repeated sampling.

A two-sided $100\,(1-\alpha)\%$ confidence interval for the mean has upper and lower limits

$$\overline{x} \pm t_{(1-\alpha/2;n-1)} \frac{s}{\sqrt{n}}$$

where s is $\sqrt{\frac{1}{n-1}\sum\left(x_i-\overline{x}\right)^2}$ and $t_{(1-\alpha/2;n-1)}$ is the $(1-\alpha/2)$ critical value of the Student's *t* statistics with $n-1$ degrees of freedom.

A one-sided $100\,(1-\alpha)\%$ confidence interval is computed as

$$\overline{x} + t_{(1-\alpha;n-1)} \frac{s}{\sqrt{n}} \qquad \text{(upper)}$$
$$\overline{x} - t_{(1-\alpha;n-1)} \frac{s}{\sqrt{n}} \qquad \text{(lower)}$$

A two-sided $100\,(1-\alpha)\%$ confidence interval for the standard deviation has lower and upper limits

$$s\sqrt{\frac{n-1}{\chi^2_{(1-\alpha/2;n-1)}}},\ s\sqrt{\frac{n-1}{\chi^2_{(\alpha/2;n-1)}}}$$

where $\chi^2_{(1-\alpha/2;n-1)}$ and $\chi^2_{(\alpha/2;n-1)}$ are the $(1-\alpha/2)$ and $\alpha/2$ critical values of the chi-square statistic with $n-1$ degrees of freedom. A one-sided $100\,(1-\alpha)\%$ confidence interval is computed by replacing $\alpha/2$ with α.

A $100\,(1-\alpha)\%$ confidence interval for the variance has upper and lower limits that are equal to the squares of the corresponding upper and lower limits for the standard deviation.

When you use the WEIGHT statement or WEIGHT= in a VAR statement and the default value of VARDEF=, which is DF, the $100\,(1-\alpha)\%$ confidence interval for the weighted mean has upper and lower limits

$$\overline{y}_w \pm t_{(1-\alpha/2)} \frac{s_w}{\sqrt{\sum_{i=1}^{n} w_i}}$$

where $\overline{y}_w$ is the weighted mean, s_w is the weighted standard deviation, w_i is the weight for ith observation, and $t_{(1-\alpha/2)}$ is the $(1-\alpha/2)$ critical value for the Student's *t* distribution with $n-1$ degrees of freedom.

Student's *t* Test

PROC MEANS calculates the t statistic as

$$t = \frac{\overline{x} - \mu_0}{s/\sqrt{n}}$$

where $\overline{x}$ is the sample mean, n is the number of nonmissing values for a variable, and s is the sample standard deviation. Under the null hypothesis, the population mean equals μ_0. When the data values are approximately normally distributed, the probability under the null hypothesis of a t statistic as extreme as, or more extreme than, the observed value (the p-value) is obtained from the t distribution with $n - 1$ degrees of freedom. For large n, the t statistic is asymptotically equivalent to a z test.

When you use the WEIGHT statement or WEIGHT= in a VAR statement and the default value of VARDEF=, which is DF, the Student's t statistic is calculated as

$$t_w = \frac{\overline{y}_w - \mu_0}{s_w/\sqrt{\sum_{i=1}^{n} w_i}}$$

where $\overline{y}_w$ is the weighted mean, s_w is the weighted standard deviation, and w_i is the weight for ith observation. The t_w statistic is treated as having a Student's t distribution with $n - 1$ degrees of freedom. If you specify the EXCLNPWGT option in the PROC statement, then n is the number of nonmissing observations when the value of the WEIGHT variable is positive. By default, n is the number of nonmissing observations for the WEIGHT variable.

Quantiles

The options QMETHOD=, QNTLDEF=, and QMARKERS= determine how PROC MEANS calculates quantiles. QNTLDEF= deals with the mathematical definition of a quantile. See "Calculating Percentiles" on page 1528. QMETHOD= deals with the mechanics of how PROC MEANS handles the input data. The two methods are

OS
: reads all data into memory and sorts it by unique value.

P2
: accumulates all data into a fixed sample size that is used to approximate the quantile.

If data set A has 100 unique values for a numeric variable X and data set B has 1000 unique values for numeric variable X, then QMETHOD=OS for data set B will take 10 times as much memory as it does for data set A. If QMETHOD=P2, then both data sets A and B will require the same memory space to generate quantiles.

The QMETHOD=P2 technique is based on the piecewise-parabolic (P^2) algorithm invented by Jain and Chlamtac (1985). P^2 is a one-pass algorithm to determine quantiles for a large data set. It requires a fixed amount of memory for each variable for each level within the type. However, using simulation studies, reliable estimations of some quantiles (P1, P5, P95, P99) may not be possible for some data sets such as those with heavily tailed or skewed distributions.

If the number of observations is less than the QMARKERS= value, then QMETHOD=P2 produces the same results as QMETHOD=OS when QNTLDEF=5. To compute weighted quantiles, you must use QMETHOD=OS.

Results: MEANS Procedure

Missing Values

PROC MEANS excludes missing values for the analysis variables before calculating statistics. Each analysis variable is treated individually; a missing value for an observation in one variable does not affect the calculations for other variables. The statements handle missing values as follows:

- If a class variable has a missing value for an observation, then PROC MEANS excludes that observation from the analysis unless you use the MISSING option in the PROC statement or CLASS statement.
- If a BY or ID variable value is missing, then PROC MEANS treats it like any other BY or ID variable value. The missing values form a separate BY group.
- If a FREQ variable value is missing or nonpositive, then PROC MEANS excludes the observation from the analysis.
- If a WEIGHT variable value is missing, then PROC MEANS excludes the observation from the analysis.

PROC MEANS tabulates the number of the missing values. Before the number of missing values are tabulated, PROC MEANS excludes observations with frequencies that are nonpositive when you use the FREQ statement and observations with weights that are missing or nonpositive (when you use the EXCLNPWGT option) when you use the WEIGHT statement. To report this information in the procedure output use the NMISS statistical keyword in the PROC statement.

Column Width for the Output

You control the column width for the displayed statistics with the FW= option in the PROC statement. Unless you assign a format to a numeric class or an ID variable, PROC MEANS uses the value of the FW= option. When you assign a format to a numeric class or an ID variable, PROC MEANS determines the column width directly from the format. If you use the PRELOADFMT option in the CLASS statement, then PROC MEANS determines the column width for a class variable from the assigned format.

The N Obs Statistic

By default when you use a CLASS statement, PROC MEANS displays an additional statistic called N Obs. This statistic reports the total number of observations or the sum of the observations of the FREQ variable that PROC MEANS processes for each class level. PROC MEANS might omit observations from this total because of missing values in one or more class variables or because of the effect of the EXCLUSIVE option when you use it with the PRELOADFMT option or the CLASSDATA= option. Because of this and the exclusion of observations when the WEIGHT variable contains missing values, there is not always a direct relationship between N Obs, N, and NMISS.

In the output data set, the value of N Obs is stored in the _FREQ_ variable. Use the NONOBS option in the PROC statement to suppress this information in the displayed output.

Output Data Set

PROC MEANS can create one or more output data sets. The procedure does not print the output data set. Use PROC PRINT, PROC REPORT, or another SAS reporting tool to display the output data set.

Note: By default the statistics in the output data set automatically inherit the analysis variable's format and label. However, statistics computed for N, NMISS, SUMWGT, USS, CSS, VAR, CV, T, PROBT, SKEWNESS, and KURTOSIS do not inherit the analysis variable's format because this format may be invalid for these statistics. Use the NOINHERIT option in the OUTPUT statement to prevent the other statistics from inheriting the format and label attributes. △

The output data set can contain these variables:

- the variables specified in the BY statement.
- the variables specified in the ID statement.
- the variables specified in the CLASS statement.
- the variable _TYPE_ that contains information about the class variables. By default _TYPE_ is a numeric variable. If you specify CHARTYPE in the PROC statement, then _TYPE_ is a character variable. When you use more than 32 class variables, _TYPE_ is automatically a character variable.
- the variable _FREQ_ that contains the number of observations that a given output level represents.
- the variables requested in the OUTPUT statement that contain the output statistics and extreme values.
- the variable _STAT_ that contains the names of the default statistics if you omit statistic keywords.
- the variable _LEVEL_ if you specify the LEVEL option.
- the variable _WAY_ if you specify the WAYS option.

The value of _TYPE_ indicates which combination of the class variables PROC MEANS uses to compute the statistics. The character value of _TYPE_ is a series of zeros and ones, where each value of one indicates an active class variable in the type. For example, with three class variables, PROC MEANS represents type 1 as 001, type 5 as 101, and so on.

Usually, the output data set contains one observation per level per type. However, if you omit statistical keywords in the OUTPUT statement, then the output data set contains five observations per level (six if you specify a WEIGHT variable). Therefore, the total number of observations in the output data set is equal to the sum of the levels for all the types you request multiplied by 1, 5, or 6, whichever is applicable.

If you omit the CLASS statement (_TYPE_= 0), then there is always exactly one level of output per BY group. If you use a CLASS statement, then the number of levels for each type that you request has an upper bound equal to the number of observations in the input data set. By default, PROC MEANS generates all possible types. In this case the total number of levels for each BY group has an upper bound equal to

$$m \cdot \left(2^k - 1\right) \cdot n + 1$$

where k is the number of class variables and n is the number of observations for the given BY group in the input data set and m is 1, 5, or 6.

PROC MEANS determines the actual number of levels for a given type from the number of unique combinations of each active class variable. A single level is composed of all input observations whose formatted class values match.

Figure 26.1 on page 683 shows the values of _TYPE_ and the number of observations in the data set when you specify one, two, and three class variables.

Figure 26.1 The Effect of Class Variables on the OUTPUT Data Set

three CLASS variables / two CLASS variables / one CLASS variable

C	B	A	_WAY_	_TYPE_	Subgroup defined by	Number of observations of this _TYPE_ and _WAY_ in the data set	Total number of observations in the data set
0	0	0	0	0	Total	1	
0	0	1	1	1	A	a	1+a
0	1	0	1	2	B	b	
0	1	1	2	3	A*B	a*b	1+a+b+a*b
1	0	0	1	4	C	c	
1	0	1	2	5	A*C	a*c	
1	1	0	2	6	B*C	b*c	1+a+b+a*b+c
1	1	1	3	7	A*B*C	a*b*c	+a*c+b*c+a*b*c
Character binary equivalent of _TYPE_ (CHARTYPE option)					A ,B ,C=CLASS variables	a, b, c,=number of levels of A, B, C, respectively	

Examples: MEANS Procedure

Example 1: Computing Specific Descriptive Statistics

Procedure features:

PROC MEANS statement options:

- statistic keywords
- FW=

VAR statement

This example

- □ specifies the analysis variables
- □ computes the statistics for the specified keywords and displays them in order
- □ specifies the field width of the statistics.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the CAKE data set. CAKE contains data from a cake-baking contest: each participant's last name, age, score for presentation, score for taste, cake flavor, and number of cake layers. The number of cake layers is missing for two observations. The cake flavor is missing for another observation.

```
data cake;
   input LastName $ 1-12 Age 13-14 PresentScore 16-17
         TasteScore 19-20 Flavor $ 23-32 Layers 34 ;
   datalines;
Orlando     27 93 80  Vanilla    1
Ramey       32 84 72  Rum        2
Goldston    46 68 75  Vanilla    1
Roe         38 79 73  Vanilla    2
Larsen      23 77 84  Chocolate  .
Davis       51 86 91  Spice      3
Strickland  19 82 79  Chocolate  1
Nguyen      57 77 84  Vanilla    .
Hildenbrand 33 81 83  Chocolate  1
Byron       62 72 87  Vanilla    2
Sanders     26 56 79  Chocolate  1
Jaeger      43 66 74             1
Davis       28 69 75  Chocolate  2
Conrad      69 85 94  Vanilla    1
Walters     55 67 72  Chocolate  2
Rossburger  28 78 81  Spice      2
Matthew     42 81 92  Chocolate  2
Becker      36 62 83  Spice      2
Anderson    27 87 85  Chocolate  1
Merritt     62 73 84  Chocolate  1
;
```

Specify the analyses and the analysis options. The statistic keywords specify the statistics and their order in the output. FW= uses a field width of eight to display the statistics.

```
proc means data=cake n mean max min range std fw=8;
```

Specify the analysis variables. The VAR statement specifies that PROC MEANS calculate statistics on the PresentScore and TasteScore variables.

```
   var PresentScore TasteScore;
```

Specify the title.

```
   title 'Summary of Presentation and Taste Scores';
run;
```

Output

PROC MEANS lists PresentScore first because this is the first variable that is specified in the VAR statement. A field width of eight truncates the statistics to four decimal places.

```
                   Summary of Presentation and Taste Scores                  1

                            The MEANS Procedure

Variable         N         Mean      Maximum      Minimum        Range      Std Dev
------------------------------------------------------------------------------------
PresentScore    20      76.1500      93.0000      56.0000      37.0000       9.3768
TasteScore      20      81.3500      94.0000      72.0000      22.0000       6.6116
------------------------------------------------------------------------------------
```

Example 2: Computing Descriptive Statistics with Class Variables

Procedure features:

PROC MEANS statement option:

MAXDEC=

CLASS statement

TYPES statement

This example

- analyzes the data for the two-way combination of class variables and across all observations
- limits the number of decimal places for the displayed statistics.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the GRADE data set. GRADE contains each student's last name, gender, status of either undergraduate (1) or graduate (2), expected year of graduation, class section (A or B), final exam score, and final grade for the course.

```
data grade;
   input Name $ 1-8 Gender $ 11 Status $13 Year $ 15-16
         Section $ 18 Score 20-21 FinalGrade 23-24;
   datalines;
Abbott    F 2 97 A 90 87
Branford  M 1 98 A 92 97
Crandell  M 2 98 B 81 71
Dennison  M 1 97 A 85 72
Edgar     F 1 98 B 89 80
Faust     M 1 97 B 78 73
Greeley   F 2 97 A 82 91
Hart      F 1 98 B 84 80
Isley     M 2 97 A 88 86
Jasper    M 1 97 B 91 93
;
```

Generate the default statistics and specify the analysis options. Because no statistics are specified in the PROC MEANS statement, all default statistics (N, MEAN, STD, MIN, MAX) are generated. MAXDEC= limits the displayed statistics to three decimal places.

```
proc means data=grade maxdec=3;
```

Specify the analysis variable. The VAR statement specifies that PROC MEANS calculate statistics on the Score variable.

```
   var Score;
```

Specify subgroups for the analysis. The CLASS statement separates the analysis into subgroups. Each combination of unique values for Status and Year represents a subgroup.

```
   class Status Year;
```

Specify which subgroups to analyze. The TYPES statement requests that the analysis be performed on all the observations in the GRADE data set as well as the two-way combination of Status and Year, which results in four subgroups (because Status and Year each have two unique values).

```
   types () status*year;
```

Specify the title.

```
   title 'Final Exam Grades for Student Status and Year of Graduation';
run;
```

Output

PROC MEANS displays the default statistics for all the observations (_TYPE_=0) and the four class levels of the Status and Year combination (Status=1, Year=97; Status=1, Year=98; Status=2, Year=97; Status=2, Year=98).

```
           Final Exam Grades for Student Status and Year of Graduation         1

                                The MEANS Procedure

                             Analysis Variable : Score

      N
    Obs     N            Mean         Std Dev         Minimum         Maximum
    ------------------------------------------------------------------------
     10    10          86.000           4.714          78.000          92.000
    ------------------------------------------------------------------------


                             Analysis Variable : Score

                  N
  Status  Year  Obs   N          Mean       Std Dev       Minimum       Maximum
  ----------------------------------------------------------------------------
  1       97      3   3        84.667         6.506        78.000        91.000

          98      3   3        88.333         4.041        84.000        92.000

  2       97      3   3        86.667         4.163        82.000        90.000

          98      1   1        81.000          .           81.000        81.000
  ----------------------------------------------------------------------------
```

Example 3: Using the BY Statement with Class Variables

Procedure features:
PROC MEANS statement option:
statistic keywords
BY statement
CLASS statement

Other features:
SORT procedure

Data set: GRADE on page 686

This example

- separates the analysis for the combination of class variables within BY values
- shows the sort order requirement for the BY statement
- calculates the minimum, maximum, and median.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Sort the GRADE data set. PROC SORT sorts the observations by the variable Section. Sorting is required in order to use Section as a BY variable in the PROC MEANS step.

```
proc sort data=Grade out=GradeBySection;
   by section;
run;
```

Specify the analyses. The statistic keywords specify the statistics and their order in the output.

```
proc means data=GradeBySection min max median;
```

Divide the data set into BY groups. The BY statement produces a separate analysis for each value of Section.

```
   by Section;
```

Specify the analysis variable. The VAR statement specifies that PROC MEANS calculate statistics on the Score variable.

```
   var Score;
```

Specify subgroups for the analysis. The CLASS statement separates the analysis by the values of Status and Year. Because there is no TYPES statement in this program, analyses are performed for each subgroup, within each BY group.

```
   class Status Year;
```

Specify the titles.

```
   title1 'Final Exam Scores for Student Status and Year of Graduation';
   title2 ' Within Each Section';
```

```
run;
```

Output

```
        Final Exam Scores for Student Status and Year of Graduation         1
                             Within Each Section

---------------------------------- Section=A -----------------------------------

                            The MEANS Procedure

                        Analysis Variable : Score

                       N
     Status    Year   Obs         Minimum         Maximum          Median
     -------------------------------------------------------------------------
     1         97       1      85.0000000      85.0000000      85.0000000

               98       1      92.0000000      92.0000000      92.0000000

     2         97       3      82.0000000      90.0000000      88.0000000
     -------------------------------------------------------------------------


---------------------------------- Section=B -----------------------------------

                        Analysis Variable : Score

                       N
     Status    Year   Obs         Minimum         Maximum          Median
     -------------------------------------------------------------------------
     1         97       2      78.0000000      91.0000000      84.5000000

               98       2      84.0000000      89.0000000      86.5000000

     2         98       1      81.0000000      81.0000000      81.0000000
     -------------------------------------------------------------------------
```

Example 4: Using a CLASSDATA= Data Set with Class Variables

Procedure features:

PROC MEANS statement options:

- CLASSDATA=
- EXCLUSIVE
- FW=
- MAXDEC=
- PRINTALLTYPES

CLASS statement

Data set: CAKE on page 684

This example

□ specifies the field width and decimal places of the displayed statistics

- uses only the values in CLASSDATA= data set as the levels of the combinations of class variables
- calculates the range, median, minimum, and maximum
- displays all combinations of the class variables in the analysis.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the CAKETYPE data set. CAKETYPE contains the cake flavors and number of layers that must occur in the PROC MEANS output.

```
data caketype;
   input Flavor $ 1-10  Layers 12;
   datalines;
Vanilla    1
Vanilla    2
Vanilla    3
Chocolate  1
Chocolate  2
Chocolate  3
;
```

Specify the analyses and the analysis options. The FW= option uses a field width of seven and the MAXDEC= option uses zero decimal places to display the statistics. CLASSDATA= and EXCLUSIVE restrict the class levels to the values that are in the CAKETYPE data set. PRINTALLTYPES displays all combinations of class variables in the output.

```
proc means data=cake range median min max fw=7 maxdec=0
           classdata=caketype exclusive printalltypes;
```

Specify the analysis variable. The VAR statement specifies that PROC MEANS calculate statistics on the TasteScore variable.

```
   var TasteScore;
```

Specify subgroups for analysis. The CLASS statement separates the analysis by the values of Flavor and Layers. Note that these variables, and only these variables, must appear in the CAKETYPE data set.

```
   class flavor layers;
```

Specify the title.

```
   title 'Taste Score For Number of Layers and Cake Flavor';
run;
```

Output

PROC MEANS calculates statistics for the 13 chocolate and vanilla cakes. Because the CLASSDATA= data set contains 3 as the value of Layers, PROC MEANS uses 3 as a class value even though the frequency is zero.

```
                 Taste Score For Number of Layers and Cake Flavor               1

                               The MEANS Procedure

                         Analysis Variable : TasteScore

                   N
                 Obs      Range     Median    Minimum    Maximum
                 --------------------------------------------------
                  13         22         80         72         94
                 --------------------------------------------------


                          Analysis Variable : TasteScore

                            N
            Layers    Obs      Range     Median    Minimum    Maximum
           ----------------------------------------------------------
                 1      8         19         82         75         94

                 2      5         20         75         72         92

                 3      0          .          .          .          .
           ----------------------------------------------------------


                         Analysis Variable : TasteScore

                             N
         Flavor         Obs      Range     Median    Minimum    Maximum
         -------------------------------------------------------------
         Chocolate        8         20         81         72         92

         Vanilla          5         21         80         73         94
         -------------------------------------------------------------


                         Analysis Variable : TasteScore

                                   N
    Flavor         Layers    Obs      Range     Median    Minimum    Maximum
    -------------------------------------------------------------------------
    Chocolate           1      5          6         83         79         85

                        2      3         20         75         72         92

                        3      0          .          .          .          .

    Vanilla             1      3         19         80         75         94

                        2      2         14         80         73         87

                        3      0          .          .          .          .
    -------------------------------------------------------------------------
```

Example 5: Using Multilabel Value Formats with Class Variables

Procedure features:

PROC MEANS statement options:

statistic keywords
FW=
NONOBS

CLASS statement options:

MLF
ORDER=

TYPES statement

Other features

FORMAT procedure

FORMAT statement

Data set: CAKE on page 684

This example

- □ computes the statistics for the specified keywords and displays them in order
- □ specifies the field width of the statistics
- □ suppresses the column with the total number of observations
- □ analyzes the data for the one-way combination of cake flavor and the two-way combination of cake flavor and participant's age
- □ assigns user-defined formats to the class variables
- □ uses multilabel formats as the levels of class variables
- □ orders the levels of the cake flavors by the descending frequency count and orders the levels of age by the ascending formatted values.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=64;
```

Create the $FLVRFMT. and AGEFMT. formats. PROC FORMAT creates user-defined formats to categorize the cake flavors and ages of the participants. MULTILABEL creates a multilabel format for Age. A multilabel format is one in which multiple labels can be assigned to the same value, in this case because of overlapping ranges. Each value is represented in the output for each range in which it occurs.

```
proc format;
   value $flvrfmt
                'Chocolate'='Chocolate'
                'Vanilla'='Vanilla'
```

```
                  'Rum','Spice'='Other Flavor';
   value agefmt (multilabel)
                  15 - 29='below 30 years'
                  30 - 50='between 30 and 50'
                  51 - high='over 50 years'
                  15 - 19='15 to 19'
                  20 - 25='20 to 25'
                  25 - 39='25 to 39'
                  40 - 55='40 to 55'
                  56 - high='56 and above';
run;
```

Specify the analyses and the analysis options. FW= uses a field width of six to display the statistics. The statistic keywords specify the statistics and their order in the output. NONOBS suppresses the N Obs column.

```
proc means data=cake fw=6 n min max median nonobs;
```

Specify subgroups for the analysis. The CLASS statements separate the analysis by values of Flavor and Age. ORDER=FREQ orders the levels of Flavor by descending frequency count. ORDER=FMT orders the levels of Age by ascending formatted values. MLF specifies that multilabel value formats be used for Age.

```
   class flavor/order=freq;
   class  age /mlf order=fmt;
```

Specify which subgroups to analyze. The TYPES statement requests the analysis for the one-way combination of Flavor and the two-way combination of Flavor and Age.

```
    types flavor flavor*age;
```

Specify the analysis variable. The VAR statement specifies that PROC MEANS calculate statistics on the TasteScore variable.

```
   var TasteScore;
```

Format the output. The FORMAT statement assigns user-defined formats to the Age and Flavor variables for this analysis.

```
   format age agefmt. flavor $flvrfmt.;
```

Specify the title.

```
   title 'Taste Score for Cake Flavors and Participant''s Age';
run;
```

Output

The one-way combination of class variables appears before the two-way combination. A field width of six truncates the statistics to four decimal places. For the two-way combination of Age and Flavor, the total number of observations is greater than the one-way combination of Flavor. This situation arises because of the multilabel format for age, which maps one internal value to more than one formatted value.

The order of the levels of Flavor is based on the frequency count for each level. The order of the levels of Age is based on the order of the user-defined formats.

```
           Taste Score for Cake Flavors and Participant's Age            1

                          The MEANS Procedure

                    Analysis Variable : TasteScore

                 Flavor           N       Min       Max    Median
                 ------------------------------------------------
                 Chocolate        9     72.00     92.00     83.00

                 Vanilla          6     73.00     94.00     82.00

                 Other Flavor     4     72.00     91.00     82.00
                 ------------------------------------------------


                    Analysis Variable : TasteScore

      Flavor          Age                    N       Min       Max    Median
      ----------------------------------------------------------------------
      Chocolate       15 to 19               1     79.00     79.00     79.00

                      20 to 25               1     84.00     84.00     84.00

                      25 to 39               4     75.00     85.00     81.00

                      40 to 55               2     72.00     92.00     82.00

                      56 and above           1     84.00     84.00     84.00

                      below 30 years         5     75.00     85.00     79.00

                      between 30 and 50      2     83.00     92.00     87.50

                      over 50 years          2     72.00     84.00     78.00

      Vanilla         25 to 39               2     73.00     80.00     76.50

                      40 to 55               1     75.00     75.00     75.00

                      56 and above           3     84.00     94.00     87.00

                      below 30 years         1     80.00     80.00     80.00

                      between 30 and 50      2     73.00     75.00     74.00

                      over 50 years          3     84.00     94.00     87.00

      Other Flavor    25 to 39               3     72.00     83.00     81.00

                      40 to 55               1     91.00     91.00     91.00

                      below 30 years         1     81.00     81.00     81.00

                      between 30 and 50      2     72.00     83.00     77.50

                      over 50 years          1     91.00     91.00     91.00
      ----------------------------------------------------------------------
```

Example 6: Using Preloaded Formats with Class Variables

Procedure features:

PROC MEANS statement options:

- COMPLETETYPES
- FW=
- MISSING
- NONOBS

CLASS statement options:

- EXCLUSIVE
- ORDER=
- PRELOADFMT

WAYS statement

Other features

FORMAT procedure

FORMAT statement

Data set: CAKE on page 684

This example

- specifies the field width of the statistics
- suppresses the column with the total number of observations
- includes all possible combinations of class variables values in the analysis even if the frequency is zero
- considers missing values as valid class levels
- analyzes the one-way and two-way combinations of class variables
- assigns user-defined formats to the class variables
- uses only the preloaded range of user-defined formats as the levels of class variables
- orders the results by the value of the formatted data.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=64;
```

Create the LAYERFMT. and $FLVRFMT. formats. PROC FORMAT creates user-defined formats to categorize the number of cake layers and the cake flavors. NOTSORTED keeps $FLVRFMT unsorted to preserve the original order of the format values.

```
proc format;
   value layerfmt 1='single layer'
                  2-3='multi-layer'
```

```
                    .='unknown';
   value $flvrfmt (notsorted)
                    'Vanilla'='Vanilla'
                    'Orange','Lemon'='Citrus'
                    'Spice'='Spice'
                    'Rum','Mint','Almond'='Other Flavor';
run;
```

Generate the default statistics and specify the analysis options. FW= uses a field width of seven to display the statistics. COMPLETETYPES includes class levels with a frequency of zero. MISSING considers missing values valid values for all class variables. NONOBS suppresses the N Obs column. Because no specific analyses are requested, all default analyses are performed.

```
proc means data=cake fw=7 completetypes missing nonobs;
```

Specify subgroups for the analysis. The CLASS statement separates the analysis by values of Flavor and Layers. PRELOADFMT and EXCLUSIVE restrict the levels to the preloaded values of the user-defined formats. ORDER=DATA orders the levels of Flavor and Layer by formatted data values.

```
   class flavor layers/preloadfmt exclusive order=data;
```

Specify which subgroups to analyze. The WAYS statement requests one-way and two-way combinations of class variables.

```
   ways 1 2;
```

Specify the analysis variable. The VAR statement specifies that PROC MEANS calculate statistics on the TasteScore variable.

```
   var TasteScore;
```

Format the output. The FORMAT statement assigns user-defined formats to the Flavor and Layers variables for this analysis.

```
   format layers layerfmt. flavor $flvrfmt.;
```

Specify the title.

```
   title 'Taste Score For Number of Layers and Cake Flavors';
run;
```

Output

The one-way combination of class variables appears before the two-way combination. PROC MEANS reports only the level values that are listed in the preloaded range of user-defined formats even when the frequency of observations is zero (in this case, citrus). PROC MEANS rejects entire observations based on the exclusion of any single class value in a given observation. Therefore, when the number of layers is unknown, statistics are calculated for only one observation. The other observation is excluded because the flavor chocolate was not included in the preloaded user-defined format for Flavor.

The order of the levels is based on the order of the user-defined formats. PROC FORMAT automatically sorted the Layers format and did not sort the Flavor format.

```
               Taste Score For Number of Layers and Cake Flavors              1

                              The MEANS Procedure

                        Analysis Variable : TasteScore

          Layers           N       Mean    Std Dev    Minimum    Maximum
          --------------------------------------------------------------
          unknown          1     84.000          .     84.000     84.000

          single layer     3     83.000      9.849     75.000     94.000

          multi-layer      6     81.167      7.548     72.000     91.000
          --------------------------------------------------------------


                        Analysis Variable : TasteScore

          Flavor           N       Mean    Std Dev    Minimum    Maximum
          --------------------------------------------------------------
          Vanilla          6     82.167      7.834     73.000     94.000

          Citrus           0          .          .          .          .

          Spice            3     85.000      5.292     81.000     91.000

          Other Flavor     1     72.000          .     72.000     72.000
          --------------------------------------------------------------


                        Analysis Variable : TasteScore

  Flavor          Layers           N       Mean    Std Dev    Minimum    Maximum
  ------------------------------------------------------------------------------
  Vanilla         unknown          1     84.000          .     84.000     84.000

                  single layer     3     83.000      9.849     75.000     94.000

                  multi-layer      2     80.000      9.899     73.000     87.000

  Citrus          unknown          0          .          .          .          .

                  single layer     0          .          .          .          .

                  multi-layer      0          .          .          .          .

  Spice           unknown          0          .          .          .          .

                  single layer     0          .          .          .          .

                  multi-layer      3     85.000      5.292     81.000     91.000

  Other Flavor    unknown          0          .          .          .          .

                  single layer     0          .          .          .          .

                  multi-layer      1     72.000          .     72.000     72.000
  ------------------------------------------------------------------------------
```

Example 7: Computing a Confidence Limit for the Mean

Procedure features:

PROC MEANS statement options:

ALPHA=
FW=
MAXDEC=

CLASS statement

This example

- specifies the field width and number of decimal places of the statistics
- computes a two-sided 90 percent confidence limit for the mean values of MoneyRaised and HoursVolunteered for the three years of data.

If this data is representative of a larger population of volunteers, then the confidence limits provide ranges of likely values for the true population means.

Program

Create the CHARITY data set. CHARITY contains information about high-school students' volunteer work for a charity. The variables give the name of the high school, the year of the fund-raiser, the first name of each student, the amount of money each student raised, and the number of hours each student volunteered. A DATA step on page 1617 creates this data set.

```
data charity;
   input School $ 1-7 Year 9-12 Name $ 14-20 MoneyRaised 22-26
         HoursVolunteered 28-29;
   datalines;
Monroe  1992 Allison 31.65 19
Monroe  1992 Barry   23.76 16
Monroe  1992 Candace 21.11  5

     . . . more data lines . . .

Kennedy 1994 Sid     27.45 25
Kennedy 1994 Will    28.88 21
Kennedy 1994 Morty   34.44 25
;
```

Specify the analyses and the analysis options. FW= uses a field width of eight and MAXDEC= uses two decimal places to display the statistics. ALPHA=0.1 specifies a 90% confidence limit, and the CLM keyword requests two-sided confidence limits. MEAN and STD request the mean and the standard deviation, respectively.

```
proc means data=charity fw=8 maxdec=2 alpha=0.1 clm mean std;
```

Specify subgroups for the analysis. The CLASS statement separates the analysis by values of Year.

```
   class Year;
```

Specify the analysis variables. The VAR statement specifies that PROC MEANS calculate statistics on the MoneyRaised and HoursVolunteered variables.

```
   var MoneyRaised HoursVolunteered;
```

Specify the titles.

```
   title 'Confidence Limits for Fund Raising Statistics';
   title2 '1992-94';
run;
```

Output

PROC MEANS displays the lower and upper confidence limits for both variables for each year.

```
                 Confidence Limits for Fund Raising Statistics                1
                                    1992-94

                              The MEANS Procedure

           N                        Lower 90%    Upper 90%
    Year  Obs  Variable           CL for Mean  CL for Mean       Mean    Std Dev
  -----------------------------------------------------------------------------
    1992   31  MoneyRaised              25.21        32.40      28.80      11.79
               HoursVolunteered         17.67        23.17      20.42       9.01

    1993   32  MoneyRaised              25.17        31.58      28.37      10.69
               HoursVolunteered         15.86        20.02      17.94       6.94

    1994   46  MoneyRaised              26.73        33.78      30.26      14.23
               HoursVolunteered         19.68        22.63      21.15       5.96
  -----------------------------------------------------------------------------
```

Example 8: Computing Output Statistics

Procedure features:

PROC MEANS statement option:
- NOPRINT

CLASS statement

OUTPUT statement options
- statistic keywords
- IDGROUP
- LEVELS
- WAYS

Other features:
PRINT procedure

Data set: GRADE on page 686

This example

- suppresses the display of PROC MEANS output
- stores the average final grade in a new variable
- stores the name of the student with the best final exam scores in a new variable
- stores the number of class variables are that are combined in the _WAY_ variable
- stores the value of the class level in the _LEVEL_ variable
- displays the output data set.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Specify the analysis options. NOPRINT suppresses the display of all PROC MEANS output.

```
proc means data=Grade noprint;
```

Specify subgroups for the analysis. The CLASS statement separates the analysis by values of Status and Year.

```
   class Status Year;
```

Specify the analysis variable. The VAR statement specifies that PROC MEANS calculate statistics on the FinalGrade variable.

```
   var FinalGrade;
```

Specify the output data set options. The OUTPUT statement creates the SUMSTAT data set and writes the mean value for the final grade to the new variable AverageGrade. IDGROUP writes the name of the student with the top exam score to the variable BestScore and the observation number that contained the top score. WAYS and LEVELS write information on how the class variables are combined.

```
   output out=sumstat mean=AverageGrade
          idgroup (max(score) obs out (name)=BestScore)
          / ways levels;
run;
```

Print the output data set WORK.SUMSTAT. The NOOBS option suppresses the observation numbers.

```
proc print data=sumstat noobs;
   title1 'Average Undergraduate and Graduate Course Grades';
   title2 'For Two Years';
run;
```

Output

The first observation contains the average course grade and the name of the student with the highest exam score over the two-year period. The next four observations contain values for each class variable value. The remaining four observations contain values for the Year and Status combination. The variables _WAY_, _TYPE_, and _LEVEL_ show how PROC MEANS created the class variable combinations. The variable _OBS_ contains the observation number in the GRADE data set that contained the highest exam score.

Average Undergraduate and Graduate Course Grades 1
For Two Years

Status	Year	_WAY_	_TYPE_	_LEVEL_	_FREQ_	Average Grade	Best Score	_OBS_
		0	0	1	10	83.0000	Branford	2
	97	1	1	1	6	83.6667	Jasper	10
	98	1	1	2	4	82.0000	Branford	2
1		1	2	1	6	82.5000	Branford	2
2		1	2	2	4	83.7500	Abbott	1
1	97	2	3	1	3	79.3333	Jasper	10
1	98	2	3	2	3	85.6667	Branford	2
2	97	2	3	3	3	88.0000	Abbott	1
2	98	2	3	4	1	71.0000	Crandell	3

Example 9: Computing Different Output Statistics for Several Variables

Procedure features:

PROC MEANS statement options:

- DESCEND
- NOPRINT

CLASS statement

OUTPUT statement options:

- statistic keywords

Other features:

PRINT procedure

WHERE= data set option

Data set: GRADE on page 686

This example

- suppresses the display of PROC MEANS output
- stores the statistics for the class level and combinations of class variables that are specified by WHERE= in the output data set
- orders observations in the output data set by descending _TYPE_ value
- stores the mean exam scores and mean final grades without assigning new variables names
- stores the median final grade in a new variable
- displays the output data set.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Specify the analysis options. NOPRINT suppresses the display of all PROC MEANS output. DESCEND orders the observations in the OUT= data set by descending _TYPE_ value.

```
proc means data=Grade noprint descend;
```

Specify subgroups for the analysis. The CLASS statement separates the analysis by values of Status and Year.

```
   class Status Year;
```

Specify the analysis variables. The VAR statement specifies that PROC MEANS calculate statistics on the Score and FinalGrade variables.

```
   var Score FinalGrade;
```

Specify the output data set options. The OUTPUT statement writes the mean for Score and FinalGrade to variables of the same name. The median final grade is written to the variable MedianGrade. The WHERE= data set option restricts the observations in SUMDATA. One observation contains overall statistics (_TYPE_=0). The remainder must have a status of 1.

```
   output out=Sumdata (where=(status='1' or _type_=0))
          mean= median(finalgrade)=MedianGrade;
run;
```

Print the output data set WORK.SUMDATA.

```
proc print data=Sumdata;
   title 'Exam and Course Grades for Undergraduates Only';
   title2 'and for All Students';
run;
```

Output

The first three observations contain statistics for the class variable levels with a status of 1. The last observation contains the statistics for all the observations (no subgroup). Score contains the mean test score and FinalGrade contains the mean final grade.

```
                  Exam and Course Grades for Undergraduates Only                1
                               and for All Students

                                                                Final     Median
   Obs    Status    Year    _TYPE_    _FREQ_     Score          Grade      Grade

    1       1        97        3         3      84.6667       79.3333      73
    2       1        98        3         3      88.3333       85.6667      80
    3       1                  2         6      86.5000       82.5000      80
    4                          0        10      86.0000       83.0000      83
```

Example 10: Computing Output Statistics with Missing Class Variable Values

Procedure features:

PROC MEANS statement options:

- CHARTYPE
- NOPRINT
- NWAY

CLASS statement options:

- ASCENDING
- MISSING
- ORDER=

OUTPUT statement

Other features:

PRINT procedure

Data set: CAKE on page 684

This example

- □ suppresses the display of PROC MEANS output
- □ considers missing values as valid level values for only one class variable
- □ orders observations in the output data set by the ascending frequency for a single class variable
- □ stores observations for only the highest _TYPE_ value

- stores _TYPE_ as binary character values
- stores the maximum taste score in a new variable
- displays the output data set.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Specify the analysis options. NWAY prints observations with the highest _TYPE_ value. NOPRINT suppresses the display of all PROC MEANS output.

```
proc means data=cake nway noprint;
```

Specify subgroups for the analysis. The CLASS statements separate the analysis by Flavor and Layers. ORDER=FREQ and ASCENDING order the levels of Flavor by ascending frequency. MISSING uses missing values of Layers as a valid class level value.

```
   class flavor /order=freq ascending;
   class layers /missing;
```

Specify the analysis variable. The VAR statement specifies that PROC MEANS calculate statistics on the TasteScore variable.

```
   var TasteScore;
```

Specify the output data set options. The OUTPUT statement creates the CAKESTAT data set and outputs the maximum value for the taste score to the new variable HighScore.

```
   output out=cakestat max=HighScore;
run;
```

Print the output data set WORK.CAKESTAT.

```
proc print data=cakestat;
   title 'Maximum Taste Score for Flavor and Cake Layers';
run;
```

Output

The CAKESTAT output data set contains only observations for the combination of both class variables, Flavor and Layers. Therefore, the value of _TYPE_ is 3 for all observations. The observations are ordered by ascending frequency of Flavor. The missing value in Layers is a valid value for this class variable. PROC MEANS excludes the observation with the missing flavor because it is an invalid value for Flavor.

```
                   Maximum Taste Score for Flavor and Cake Layers              1

                                                                 High
             Obs     Flavor       Layers     _TYPE_     _FREQ_    Score

              1      Rum             2          3          1        72
              2      Spice           2          3          2        83
              3      Spice           3          3          1        91
              4      Vanilla         .          3          1        84
              5      Vanilla         1          3          3        94
              6      Vanilla         2          3          2        87
              7      Chocolate       .          3          1        84
              8      Chocolate       1          3          5        85
              9      Chocolate       2          3          3        92
```

Example 11: Identifying an Extreme Value with the Output Statistics

Procedure features:

CLASS statement

OUTPUT statement options:

- statistic keyword
- MAXID

Other features:

PRINT procedure

Data set: CHARITY on page 699

This example

- □ identifies the observations with maximum values for two variables
- □ creates new variables for the maximum values
- □ displays the output data set.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Specify the analyses. The statistic keywords specify the statistics and their order in the output. CHARTYPE writes the _TYPE_ values as binary characters in the output data set

```
proc means data=Charity n mean range chartype;
```

Specify subgroups for the analysis. The CLASS statement separates the analysis by School and Year.

```
   class School Year;
```

Specify the analysis variables. The VAR statement specifies that PROC MEANS calculate statistics on the MoneyRaised and HoursVolunteered variables.

```
   var MoneyRaised HoursVolunteered;
```

Specify the output data set options. The OUTPUT statement writes the new variables, MostCash and MostTime, which contain the names of the students who collected the most money and volunteered the most time, respectively, to the PRIZE data set.

```
   output out=Prize maxid(MoneyRaised(name)
          HoursVolunteered(name))= MostCash MostTime
          max= ;
```

Specify the title.

```
   title 'Summary of Volunteer Work by School and Year';
run;
```

Print the WORK.PRIZE output data set.

```
proc print data=Prize;
   title 'Best Results: Most Money Raised and Most Hours Worked';
run;
```

Output

The first page of output shows the output from PROC MEANS with the statistics for six class levels: one for Monroe High for the years 1992, 1993, and 1994; and one for Kennedy High for the same three years.

```
                 Summary of Volunteer Work by School and Year                1

                             The MEANS Procedure

                          N
School           Year   Obs  Variable            N          Mean         Range
------------------------------------------------------------------------------
Kennedy          1992    15  MoneyRaised        15    29.0800000    39.7500000
                             HoursVolunteered   15    22.1333333    30.0000000

                 1993    20  MoneyRaised        20    28.5660000    23.5600000
                             HoursVolunteered   20    19.2000000    20.0000000

                 1994    18  MoneyRaised        18    31.5794444    65.4400000
                             HoursVolunteered   18    24.2777778    15.0000000

Monroe           1992    16  MoneyRaised        16    28.5450000    48.2700000
                             HoursVolunteered   16    18.8125000    38.0000000

                 1993    12  MoneyRaised        12    28.0500000    52.4600000
                             HoursVolunteered   12    15.8333333    21.0000000

                 1994    28  MoneyRaised        28    29.4100000    73.5300000
                             HoursVolunteered   28    19.1428571    26.0000000
------------------------------------------------------------------------------
```

The output from PROC PRINT shows the maximum MoneyRaised and HoursVolunteered values and the names of the students who are responsible for them. The first observation contains the overall results, the next three contain the results by year, the next two contain the results by school, and the final six contain the results by School and Year.

```
            Best Results: Most Money Raised and Most Hours Worked           2

                                            Most     Most       Money     Hours
  Obs  School    Year  _TYPE_  _FREQ_  Cash     Time      Raised  Volunteered

    1               .     00       109    Willard  Tonya      78.65       40
    2            1992     01        31    Tonya    Tonya      55.16       40
    3            1993     01        32    Cameron  Amy        65.44       31
    4            1994     01        46    Willard  L.T.       78.65       33
    5  Kennedy      .     10        53    Luther   Jay        72.22       35
    6  Monroe       .     10        56    Willard  Tonya      78.65       40
    7  Kennedy   1992     11        15    Thelma   Jay        52.63       35
    8  Kennedy   1993     11        20    Bill     Amy        42.23       31
    9  Kennedy   1994     11        18    Luther   Che-Min    72.22       33
   10  Monroe    1992     11        16    Tonya    Tonya      55.16       40
   11  Monroe    1993     11        12    Cameron  Myrtle     65.44       26
   12  Monroe    1994     11        28    Willard  L.T.       78.65       33
```

Example 12: Identifying the Top Three Extreme Values with the Output Statistics

Procedure features:

PROC MEANS statement option:

NOPRINT

CLASS statement

OUTPUT statement options:

statistic keywords
AUTOLABEL
AUTONAME
IDGROUP

TYPES statement

Other features:

FORMAT procedure

FORMAT statement

PRINT procedure

RENAME = data set option

Data set: CHARITY on page 699

This example

- □ suppresses the display of PROC MEANS output
- □ analyzes the data for the one-way combination of the class variables and across all observations
- □ stores the total and average amount of money raised in new variables
- □ stores in new variables the top three amounts of money raised, the names of the three students who raised the money, the years when it occurred, and the schools the students attended
- □ automatically resolves conflicts in the variable names when names are assigned to the new variables in the output data set
- □ appends the statistic name to the label of the variables in the output data set that contain statistics that were computed for the analysis variable.
- □ assigns a format to the analysis variable so that the statistics that are computed from this variable inherit the attribute in the output data set
- □ renames the _FREQ_ variable in the output data set
- □ displays the output data set and its contents.

Program

Set the SAS system options. The NODATE option suppresses the display of the date and time in the output. PAGENO= specifies the starting page number. LINESIZE= specifies the output line length, and PAGESIZE= specifies the number of lines on an output page.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the YRFMT. and $SCHFMT. formats. PROC FORMAT creates user-defined formats that assign the value of **All** to the missing levels of the class variables.

```
proc format;
   value yrFmt . = " All";
   value $schFmt ' ' = "All    ";
run;
```

Generate the default statistics and specify the analysis options. NOPRINT suppresses the display of all PROC MEANS output.

```
proc means data=Charity noprint;
```

Specify subgroups for the analysis. The CLASS statement separates the analysis by values of School and Year.

```
   class School Year;
```

Specify which subgroups to analyze. The TYPES statement requests the analysis across all the observations and for each one-way combination of School and Year.

```
   types () school year;
```

Specify the analysis variable. The VAR statement specifies that PROC MEANS calculate statistics on the MoneyRaised variable.

```
   var MoneyRaised;
```

Specify the output data set options. The OUTPUT statement creates the TOP3LIST data set. RENAME= renames the _FREQ_ variable that contains frequency count for each class level. SUM= and MEAN= specify that the sum and mean of the analysis variable (MoneyRaised) are written to the output data set. IDGROUP writes 12 variables that contain the top three amounts of money raised and the three corresponding students, schools, and years. AUTOLABEL appends the analysis variable name to the label for the output variables that contain the sum and mean. AUTONAME resolves naming conflicts for these variables.

```
   output out=top3list(rename=(_freq_=NumberStudents))sum= mean=
          idgroup( max(moneyraised) out[3] (moneyraised name
            school year)=)/autolabel autoname;
```

Format the output. The LABEL statement assigns a label to the analysis variable MoneyRaised. The FORMAT statement assigns user-defined formats to the Year and School variables and a SAS dollar format to the MoneyRaised variable.

```
   label MoneyRaised='Amount Raised';
   format year yrfmt. school $schfmt.
```

```
          moneyraised dollar8.2;
run;
```

Print the output data set WORK.TOP3LIST.

```
proc print data=top3list;
   title1 'School Fund Raising Report';
   title2 'Top Three Students';
run;
```

Display information about the TOP3LIST data set. PROC DATASETS displays the contents of the TOP3LIST data set. NOLIST suppresses the directory listing for the WORK data library.

```
proc datasets library=work nolist;
   contents data=top3list;
   title1 'Contents of the PROC MEANS Output Data Set';
run;
```

Output

The output from PROC PRINT shows the top three values of MoneyRaised, the names of the students who raised these amounts, the schools the students attended, and the years when the money was raised. The first observation contains the overall results, the next three contain the results by year, and the final two contain the results by school. The missing class levels for School and Year are replaced with the value **ALL**.

The labels for the variables that contain statistics that were computed from MoneyRaised include the statistic name at the end of the label.

```
                      School Fund Raising Report                        1
                               Top Three Students

                                             Money    Money
                                   Number   Raised_  Raised_  Money     Money     Money
Obs School   Year _TYPE_ Students      Sum     Mean Raised_1 Raised_2 Raised_3

 1  All       All    0       109  $3192.75   $29.29   $78.65   $72.22   $65.44
 2  All      1992    1        31   $892.92   $28.80   $55.16   $53.76   $52.63
 3  All      1993    1        32   $907.92   $28.37   $65.44   $47.33   $42.23
 4  All      1994    1        46  $1391.91   $30.26   $78.65   $72.22   $56.87
 5  Kennedy   All    2        53  $1575.95   $29.73   $72.22   $52.63   $43.89
 6  Monroe    All    2        56  $1616.80   $28.87   $78.65   $65.44   $56.87



Obs Name_1  Name_2  Name_3  School_1 School_2 School_3 Year_1 Year_2 Year_3

 1  Willard Luther  Cameron Monroe   Kennedy  Monroe    1994   1994   1993
 2  Tonya   Edward  Thelma  Monroe   Monroe   Kennedy   1992   1992   1992
 3  Cameron Myrtle  Bill    Monroe   Monroe   Kennedy   1993   1993   1993
 4  Willard Luther  L.T.    Monroe   Kennedy  Monroe    1994   1994   1994
 5  Luther  Thelma  Jenny   Kennedy  Kennedy  Kennedy   1994   1992   1992
 6  Willard Cameron L.T.    Monroe   Monroe   Monroe    1994   1993   1994
```

```
                  Contents of the PROC MEANS Output Data Set                  2

                            The DATASETS Procedure

Data Set Name        WORK.TOP3LIST                     Observations          6
Member Type          DATA                              Variables             18
Engine               V9                                Indexes               0
Created              18:59 Thursday, March 14, 2002    Observation Length    144
Last Modified        18:59 Thursday, March 14, 2002    Deleted Observations  0
Protection                                             Compressed            NO
Data Set Type                                          Sorted                NO
Label
Data Representation  WINDOWS
Encoding             wlatin1  Western (Windows)


                      Engine/Host Dependent Information

Data Set Page Size          12288
Number of Data Set Pages    1
First Data Page             1
Max Obs per Page            85
Obs in First Data Page      6
Number of Data Set Repairs  0
File Name                   filename
Release Created             9.0000B0
Host Created                WIN_PRO


                  Alphabetic List of Variables and Attributes

     #    Variable            Type    Len    Format       Label

     7    MoneyRaised_1       Num       8    DOLLAR8.2    Amount Raised
     8    MoneyRaised_2       Num       8    DOLLAR8.2    Amount Raised
     9    MoneyRaised_3       Num       8    DOLLAR8.2    Amount Raised
     6    MoneyRaised_Mean    Num       8    DOLLAR8.2    Amount Raised_Mean
     5    MoneyRaised_Sum     Num       8    DOLLAR8.2    Amount Raised_Sum
    10    Name_1              Char      7
    11    Name_2              Char      7
    12    Name_3              Char      7
     4    NumberStudents      Num       8
     1    School              Char      7    $SCHFMT.
    13    School_1            Char      7    $SCHFMT.
    14    School_2            Char      7    $SCHFMT.
    15    School_3            Char      7    $SCHFMT.
     2    Year                Num       8    YRFMT.
    16    Year_1              Num       8    YRFMT.
    17    Year_2              Num       8    YRFMT.
    18    Year_3              Num       8    YRFMT.
     3    _TYPE_              Num       8
```

See the TEMPLATE procedure in *SAS Output Delivery System User's Guide* for an example of how to create a custom table definition for this output data set.

References

Jain R. and Chlamtac I., (1985) "The P^2 Algorithm for Dynamic Calculation of Quantiles and Histograms Without Sorting Observations," *Communications of the Association of Computing Machinery*, 28:10.

CHAPTER

27

The OPTIONS Procedure

Overview: OPTIONS Procedure

The OPTIONS procedure lists the current settings of SAS system options. The results are displayed in the SAS log.

SAS system options control how the SAS System formats output, handles files, processes data sets, interacts with the operating environment, and does other tasks that are not specific to a single SAS program or data set. You can change the settings of SAS system options

- □ in the SAS command
- □ in a configuration or autoexec file
- □ in the SAS OPTIONS statement
- □ through the SAS System Options window
- □ by using SCL functions
- □ in a STARTSAS window or a STARTSAS statement
- □ in other ways, depending on your operating environment. See the companion for your operating environment for details.

For information about SAS system options, see the section on SAS system options in *SAS Language Reference: Dictionary*.

The log that results from running PROC OPTIONS shows both the portable and host systems options, their settings, and short descriptions. Output 27.1 on page 713 shows a partial log that displays the settings of portable options through those that begin with the letter “C.”

```
proc options;
run;
```

Output 27.1 Log Showing a Partial Listing of SAS System Options

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.04 seconds
      cpu time            0.00 seconds

6
7    proc options;
8    run;
    SAS (r) Proprietary Software Release 9.XX  TS0A1

Portable Options:

 APPLETLOC=\\dntsrc\sas\m900\avdobj\jar
                  Location of Java applets
 ARMAGENT=        ARM Agent to use to collect ARM records
 ARMFORMAT=DEFAULT SAS collector to use to log ARM records
 ARMLOC=          Identify location where ARM records are to be written
 ARMSUBSYS=ARM_NONE
                  Enable/Disable ARMing of SAS subsystems
 ASYNCHIO         Enable asynchronous input/output
 NOAUTOSIGNON     SAS/CONNECT remote submit will not automatically attempt
                  to SIGNON
 NOBATCH          Do not use the batch set of default values for SAS system
                  options
 BINDING=DEFAULT  Controls the binding edge for duplexed output
 BOTTOMMARGIN=0.000
                  Bottom margin for printed output
 BUFNO=1          Number of buffers for each SAS data set
 BUFSIZE=0        Size of buffer for page of SAS data set
 BYERR            Set the error flag if a null data set is input to the SORT
                  procedure
 BYLINE           Print the by-line at the beginning of each by-group
 BYSORTED         Require SAS data set observations to be sorted for BY
                  processing
 NOCAPS           Do not translate source input to uppercase
 NOCARDIMAGE      Do not process SAS source and data lines as 80-byte records
 CATCACHE=0       Number of SAS catalogs to keep in cache memory
 CBUFNO=0         Number of buffers to use for each SAS catalog
 CENTER           Center SAS procedure output
 NOCHARCODE       Do not use character combinations as substitute for
                  special characters not on the keyboard
 CLEANUP          Attempt recovery from out-of-resources condition
 NOCMDMAC         Do not support command-style macros
 CMPOPT           Enable SAS compiler performance optimizations
 NOCOLLATE        Do not collate multiple copies of printed output
```

Output 27.2 on page 714 shows a log that PROC OPTIONS produces for a single SAS system option.

```
options pagesize=60;
proc options option=pagesize;
run;
```

Output 27.2 The Setting of a Single SAS System Option

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.03 seconds
      cpu time            0.00 seconds

25   options pagesize=60;
26   proc options option=pagesize;
27   run;
    SAS (r) Proprietary Software Release 9.XXX  TS0A1

 PAGESIZE=60       Number of lines printed per page of output
NOTE: PROCEDURE OPTIONS used
```

Syntax: OPTIONS Procedure

PROC OPTIONS <*option(s)*>;

PROC OPTIONS Statement

PROC OPTIONS <*option(s)*>;

To do this	Use this option
Choose the format of the listing	
Specify the long form	LONG
Specify the short form	SHORT
Display the option's description, type and group	DEFINE
Display the option's value and scope	VALUE
Restrict the number of options displayed	
Display options belonging to a group	GROUP=
Display host options only	HOST
Display portable options only	NOHOST
Display a single option	OPTION=

Options

DEFINE
displays the short description of the option, the option group, option type, and how to set and display the option value.

Interaction: This option has no effect when SHORT is specified.

GROUP=*group-name*
displays the options in the group specified by *group-name*. For more information on options groups, see *SAS Language Reference: Dictionary*.

HOST | NOHOST
displays only host options (HOST) or displays only portable options (NOHOST).

Alias: PORTABLE is an alias for NOHOST.

LONG | SHORT
specifies the format for displaying the settings of the SAS system options. LONG lists each option on a separate line with a description; SHORT produces a compressed listing without the descriptions.

Default: LONG

Featured in: Example 1 on page 717

NOHOST
See HOST | NOHOST.

OPTION=*option-name*
displays a short description and the value (if any) of the option specified by *option-name*. DEFINE and VALUE provide additional information about the option.

option-name
specifies the option to use as input to the procedure.

Requirement: If a SAS system option uses an equals sign, such as PAGESIZE=, do not include the equals sign when specifying the option to OPTION=.

Featured in: Example 2 on page 718

SHORT
See LONG | SHORT.

VALUE
displays the option value and scope, as well as how the value was set.

Interaction: This option has no effect when SHORT is specified.

Results: OPTIONS Procedure

SAS Log

SAS writes the options list to the SAS log. SAS system options of the form *option* | NO*option* are listed as either *option* or NO*option*, depending on the current setting, but they are always sorted by the positive form. For example, NOCAPS would be listed under the Cs.

Operating Environment Information: PROC OPTIONS produces additional information that is specific to the environment under which you are running the SAS System. Refer to the SAS documentation for your operating environment for more information about this and for descriptions of host-specific options. △

Examples: OPTIONS Procedure

Example 1: Producing the Short Form of the Options Listing

Procedure features:
PROC OPTIONS statement option:
SHORT

This example shows how to generate the short form of the listing of SAS system option settings. Compare this short form with the long form shown in “Overview: OPTIONS Procedure” on page 713.

Program

SHORT lists the SAS system options and their settings without any descriptions.

```
proc options short;
run;
```

Log (partial)

Output 27.3

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.09 seconds
      cpu time            0.00 seconds

16   proc options short;
17   run;
    SAS (r) Proprietary Software Release 9.XX  TS0A1

Portable Options:

 APPLETLOC=\\dntsrc\sas\m900\avdobj\jar ARMAGENT= ARMFORMAT=DEFAULT ARMLOC=
ARMSUBSYS=ARM_NONE ASYNCHIO NOAUTOSIGNON NOBATCH BINDING=DEFAULT
BOTTOMMARGIN=0.000 BUFNO=1 BUFSIZE=0 BYERR BYLINE BYSORTED NOCAPS NOCARDIMAGE
CATCACHE=0 CBUFNO=0 CENTER NOCHARCODE CLEANUP NOCMDMAC CMPOPT NOCOLLATE
COLORPRINTING COMPRESS=NO CONNECTPERSIST=YES CONNECTREMOTE= CONNECTSTATUS
CONNECTWAIT CONSOLELOG= COPIES=1 CPUCOUNT=1 CPUID DATASTMTCHK=COREKEYWORDS
DATE DATESTYLE=MDY DBSLICEPARM=(THREADED_APPS, 2) DBSRVTP=NONE NODETAILS
DEVICE= DFLANG=ENGLISH DKRICOND=ERROR DKROCOND=WARN DLDMGACTION=REPAIR NODMR
NODMS NODMSEXP NODMSSYNCHK DQLOCALE= DSNFERR NODTRESET NODUPLEX NOECHOAUTO
EMAILAUTHPROTOCOL=NONE EMAILHOST=LOCALHOST EMAILID= EMAILPORT=25 EMAILPW=
ENGINE=V9 NOERRORABEND NOERRORBYABEND ERRORCHECK=NORMAL ERRORS=20 NOEXPLORER
FIRSTOBS=1 FMTERR FMTSEARCH=(WORK LIBRARY) FONTSLOC=C:\V9setup\font
FORMCHAR=£$<>\^_{|}~+=|-/\<>* FORMDLIM= FORMS=DEFAULT GISMAPS= GWINDOW
HELPENCMD HELPINDEX=(/help/common.hlp/index.txt /help/common.hlp/keywords.htm
common.hhk) HELPTOC=(/help/common.hlp/contents.txt /help/common.hlp/toc.htm
common.hhc) IBUFSIZE=0 NOIMPLMAC INITCMD= INITSTMT= INVALIDDATA=. LABEL
LEFTMARGIN=0.000 LINESIZE=78
```

Example 2: Displaying the Setting of a Single Option

Procedure features:

PROC OPTIONS statement option:

OPTION=
DEFINE

VALUE

This example shows how to display the setting of a single SAS system option. The log shows the current setting of the SAS system option PAGESIZE=. The DEFINE and VALUE options display additional information.

Program

PAGESIZE=60 sets the number of lines on a page to 60.

```
options pagesize=60;
```

OPTION=PAGESIZE displays the setting of PAGESIZE in the log. DEFINE and VALUE display additional information.

```
proc options option=pagesize define value;
run;
```

Log

Output 27.4

```
NOTE: PROCEDURE PRINTTO used (Total process time):
      real time           0.07 seconds
      cpu time            0.01 seconds

6    proc options option=pagesize define value;
7    run;
    SAS (r) Proprietary Software Release 9.XX  TS0A1

Option Value Information For SAS Option PAGESIZE
    Option Value: 55
    Option Scope: Line Mode Process
    How option value set:  Unknown
Option Definition Information for SAS Option PAGESIZE
    Group= LOG_LISTCONTROL
    Group Description: SAS log and procedure output settings
    Description: Number of lines printed per page of output
    Type: The option value is of type LONG
          Range of Values: The minimum is 15 and the maximum is 32767
          Valid Syntax(any casing): MIN|MAX|n|nK|nM|nG|nT|hex
    When Can Set: Startup or anytime during the SAS Session
    SAS Language: Can "get" the option value using SAS language
    SAS Language: Can "set" the option value using SAS language
    Print or Display: Special keyword is NOT required
    Documentation: See http://sww.sas.com/sas/m900/dhost/doc/optArchive.html
NOTE: PROCEDURE OPTIONS used (Total process time):
      real time           0.63 seconds
      cpu time            0.07 seconds

8    proc printto; run;
```

CHAPTER
28

The OPTLOAD Procedure

Overview: OPTLOAD Procedure

The OPTLOAD procedure reads SAS system option settings that are stored in the SAS registry or a SAS data set and puts them into effect.

You can load SAS system option settings from a SAS data set or registry key by using

- the DMOPTLOAD command from a command line in the SAS windowing environment. For example, DMOPTLOAD key= “core\options”.
- the PROC OPTLOAD statement.

Some SAS options are not t be saved with PROC OPTSAVE and therefore cannot be loaded with OPTLOAD. The following is a list of these options:

- ARMAGENT system option
- ARMFORMAT system option
- ARMLOC system option
- ARMSUBSYS system option
- AWSDEF system option
- FONTALIAS system option
- SORTMSG system option
- STIMER system option
- TPSEC system option
- All SAS system options that can be specified only during startup
- All SAS system options that identify a password.

Syntax: OPTLOAD Procedure

PROC OPTLOAD <*options*>;

PROC OPTLOAD Statement

PROC OPTLOAD <*options*>;

To do this	Use this option
Load SAS system option settings from an existing registry key	KEY=
Load SAS system option settings from an existing data set	DATA=

Options

DATA=*libref.dataset*
specifies the library and data set name from where SAS system option settings are loaded. The SAS variable OPTNAME contains the character value of the SAS system option name, and the SAS variable OPTVALUE contains the character value of the SAS system option setting.

Requirement: The SAS library and data set must exist.

Default: If you omit the DATA= option and the KEY= option, the procedure will use the default SAS library and data set. The default library is where the current user profile resides. Unless you specify a library, the default library is SASUSER. If SASUSER is being used by another active SAS session, then the temporary WORK library is the default location from which the data set is loaded. The default data set name is MYOPTS.

KEY=*"SAS registry key"*
specifies the location in the SAS registry of stored SAS system option settings. The registry is retained in SASUSER. If SASUSER is not available, then the temporary WORK library is used. For example, KEY="OPTIONS".

Requirement: *"SAS registry key"* must be an existing SAS registry key.

Requirement: You must use quotation marks around the *"SAS registry key"* name. Separate the names in a sequence of key names with a backslash (\). For example, KEY=*"CORE\OPTIONS"*.

CHAPTER 29

The OPTSAVE Procedure

Overview: OPTSAVE Procedure

PROC OPTSAVE saves the current SAS system option settings in the SAS registry or in a SAS data set.

SAS system options can be saved across SAS sessions. You can save the settings of the SAS system options in a SAS data set or registry key by using

- the DMOPTSAVE command from a command line in the SAS windowing environment. Use the command like this: DMOPTSAVE <*save-location*>.
- the PROC OPTSAVE statement.

Some SAS options will not be saved with PROC OPTSAVE. The following is a list of these options:

- ARMAGENT system option
- ARMFORMAT system option
- ARMLOC system option
- ARMSUBSYS system option
- AWSDEF system option
- FONTALIAS system option
- SORTMSG system option
- STIMER system option
- TPSEC system option
- All SAS system options that can be specified only during startup
- All SAS system options that identify a password.

Syntax: OPTSAVE Procedure

Tip: The only statement that is used with the OPTSAVE procedure is the PROC statement.

PROC OPTSAVE <*options* >;

PROC OPTSAVE Statement

PROC OPTSAVE <*options* >;

To do this	Use this option
Save SAS system option settings to a registry key	KEY=
Save SAS system option settings to a SAS data set	DATA=

Options

KEY=*"SAS registry key"*

specifies the location in the SAS registry of stored SAS system option settings. The registry is retained in SASUSER. If SASUSER is not available, then the temporary WORK library is used. For example, KEY="OPTIONS".

Restriction: *"SAS registry key"* names cannot span multiple lines.

Requirement: Separate the names in a sequence of key names with a backslash (\). Individual key names can contain any character except a backslash.

Requirement: The length of a key name cannot exceed 255 characters (including the backslashes).

Requirement: You must use quotation marks around the *"SAS registry key"* name.

Tip: To specify a subkey, enter multiple key names starting with the root key.

Caution: If the key already exists, it will be overwritten. If the specified key does not already exist in the current SAS registry, then the key is automatically created when option settings are saved in the SAS registry.

DATA=*libref.dataset*

specifies the names of the library and data set where SAS system option settings are saved. The SAS variable OPTNAME contains the character value of the SAS system option name. The SAS variable OPTVALUE contains the character value of the SAS system option setting.

Caution: If the data set already exists, it will be overwritten.

Default: If you omit the DATA= and the KEY= options, the procedure will use the default SAS library and data set. The default SAS library is where the current user profile resides. Unless you specify a SAS library, the default library is SASUSER. If SASUSER is in use by another active SAS session, then the temporary WORK library is the default location where the data set is saved. The default data set name is MYOPTS.

CHAPTER
30
The PLOT Procedure

Overview: PLOT Procedure

The PLOT procedure plots the values of two variables for each observation in an input SAS data set. The coordinates of each point on the plot correspond to the two variables' values in one or more observations of the input data set.

Output 30.1 on page 726 is a simple plot of the high values of the Dow Jones Industrial Average (DJIA) between 1954 and 1994. PROC PLOT determines the plotting symbol and the scales for the axes. These are the statements that produce the output:

```
options nodate pageno=1 linesize=64
   pagesize=25;

proc plot data=djia;
   plot high*year;
   title 'High Values of the Dow Jones';
   title2 'Industrial Average';
   title3 'from 1954 to 1994';
run;
```

Output 30.1 A Simple Plot

```
                 High Values of the Dow Jones                1
                      Industrial Average
                      from 1954 to 1994

    Plot of High*Year.  Legend: A = 1 obs, B = 2 obs, etc.

4000 +                                              A
     |                                             A
     |                                           AA
High |                                          A
     |                                       A A
     |                                        A
2000 +                                      A
     |                                     A
     |                                   AA
     |                AAAAAAAAAAAAAAAAAAA
     |        AAAAAAAA
     |      AA
   0 +
     ---+---------+---------+---------+---------+---------+--
      1950      1960      1970      1980      1990      2000

                               Year
```

You can also overlay two plots, as shown in Output 30.2 on page 726. One plot shows the high values of the DJIA; the other plot shows the low values. The plot also shows that you can specify plotting symbols and put a box around a plot. The statements that produce Output 30.2 on page 726 are shown in Example 3 on page 752.

Output 30.2 Plotting Two Sets of Values at Once

```
                     Plot of Highs and Lows                    1
              for the Dow Jones Industrial Average

            Plot of High*Year.  Symbol used is '*'.
            Plot of Low*Year.  Symbol used is 'o'.

      ---+---------+---------+---------+---------+---------+---
 4000 +                                              *        +
      |                                             *         |
      |                                            * o        |
      |                                           *oo         |
 High |                                          *            |
      |                                       * *             |
      |                                           o           |
      |                                        *oo            |
 2000 +                                      * o              +
      |                                       o               |
      |                                     *o                |
      |                                   **o                 |
      |                ****** ************oo                  |
      |           *****oooooo*o o oooooooo                    |
      |      *****oooo           o                            |
      |      o                                                |
    0 +                                                       +
      ---+---------+---------+---------+---------+---------+---
       1950      1960      1970      1980      1990      2000

                                Year

NOTE: 7 obs hidden.
```

PROC PLOT can also label points on a plot with the values of a variable, as shown in Output 30.3 on page 727. The data plotted represent population density and crime rates for selected U.S. states. The SAS code that produces Output 30.3 on page 727 is shown in Example 11 on page 769.

Output 30.3 Labeling Points on a Plot

```
                               A Plot of Population Density and Crime Rates                                     1

                       Plot of Density*CrimeRate$State.  Symbol is value of State.

        ---+------------+------------+------------+------------+------------+------------+------------+---
Density |                                                                                                |
    500 +                                                                                                +
        |                                                                                                |
        |                                            Maryland                                            |
        |                                               M                                                |
        |                                                                                                |
        |                                                                                                |
        |                                                                                                |
        |                                                                                                |
        |                                     Delaware                                                   |
        |                                        D                                                       |
        |            Pennsylvania          Ohio                                                          |
        |                 P                 O                                                            |
    250 +                                                                                                +
        |                                           Illinois                                             |
        |                                              I                                     Florida     |
        |                                                                                       F        |
        |                              North Carolina             California                             |
        |                   New                   South               C                                  |
        |   West         Hampshire   Alabama N   Carolina                                                |
        | Virginia           N                T     S       G Georgia                                    |
        |    W           Mississippi       A Tennessee                  Washington Texas                 |
        |                     M  Vermont V    M Missouri    Oklahoma        W        T                   |
        |         South        Arkansas A    M Minnesota       O          Oregon                         |
        |         Dakota               I Idaho                    Nevada    O                            |
      0 +           S N North Dakota                                N                                    +
        ---+------------+------------+------------+------------+------------+------------+------------+---
         2000         3000         4000         5000         6000         7000         8000         9000

                                                    CrimeRate
```

Syntax: PLOT Procedure

Requirement: At least one PLOT statement is required.

Tip: Supports RUN-group processing

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can use the ATTRIB, FORMAT, LABEL, and WHERE statements. See Chapter 3, "Statements with the Same Function in Multiple Procedures," on page 53 for details. You can also use any global statements as well. See "Global Statements" on page 18 for a list.

PROC PLOT <*option(s)*>;

BY <DESCENDING> *variable-1*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

PLOT *plot-request(s)* </ *option(s)*>;

To do this	Use this statement
Produce a separate plot for each BY group	BY
Describe the plots you want	PLOT

PROC PLOT Statement

Reminder: You can use data set options with the DATA= option. See “Data Set Options” on page 17 for a list.

PROC PLOT <*option(s)*>;

To do this	Use this option
Specify the input data set	DATA=
Control the axes	
Include missing character variable values	MISSING
Exclude observations with missing values	NOMISS
Uniformly scale axes across BY groups	UNIFORM
Control the appearance of the plot	
Specify the characters that construct the borders of the plot	FORMCHAR=
Suppress the legend at the top of the plot	NOLEGEND
Specify the aspect ratio of the characters on the output device	VTOH=
Control the size of the plot	
Specify the percentage of the available horizontal space for each plot	HPERCENT=
Specify the percentage of the available vertical space for each plot	VPERCENT=

Options

DATA=*SAS-data-set*
specifies the input SAS data set.

Main discussion: See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures."

FORMCHAR <(*position(s)*)>=’*formatting-character(s)*’

defines the characters to use for constructing the borders of the plot.

position(s)
identifies the position of one or more characters in the SAS formatting-character string. A space or a comma separates the positions.

Default: Omitting (*position(s)*), is the same as specifying all twenty possible SAS formatting characters, in order.

Range: PROC PLOT uses formatting characters 1, 2, 3, 5, 7, 9, and 11. The following table shows the formatting characters that PROC PLOT uses.

Position	Default	Used to draw
1	|	vertical separators
2	-	horizontal separators
3 5 9 1 1	-	corners
7	+	intersection of vertical and horizontal separators

formatting-character(s)
lists the characters to use for the specified positions. PROC PLOT assigns characters in *formatting-character(s)* to *position(s)*, in the order that they are listed. For instance, the following option assigns the asterisk (*) to the third formatting character, the pound sign (#) to the seventh character, and does not alter the remaining characters:

```
formchar(3,7)='*#'
```

Interaction: The SAS system option FORMCHAR= specifies the default formatting characters. The system option defines the entire string of formatting characters. The FORMCHAR= option in a procedure can redefine selected characters.

Tip: You can use any character in *formatting-characters*, including hexadecimal characters. If you use hexadecimal characters, you must put an **x** after the closing quote. For instance the following option assigns the hexadecimal character 2D to the third formatting character, the hexadecimal character 7C to the seventh character, and does not alter the remaining characters:

```
formchar(3,7)='2D7C'x
```

Tip: Specifying all blanks for *formatting-character(s)* produces plots with no borders, for example

```
formchar (1,2,7)=''
```

HPERCENT=*percent(s)*
specifies one or more percentages of the available horizontal space to use for each plot. HPERCENT= enables you to put multiple plots on one page. PROC PLOT tries to fit as many plots as possible on a page. After using each of the *percent(s)*, PROC PLOT cycles back to the beginning of the list. A zero in the list forces PROC PLOT to go to a new page even though it could fit the next plot on the same page.

hpercent=33
prints three plots per page horizontally, each plot is one-third of a page wide.

hpercent=50 25 25
prints three plots per page, the first is twice as wide as the other two.

`hpercent=33 0`
produces plots that are one-third of a page wide, each plot is on a separate page.

`hpercent=300`
produces plots three pages wide.

At the beginning of every BY group and after each RUN statement, PROC PLOT returns to the beginning of the *percent(s)* and starts printing a new page.

Alias: HPCT=

Default: 100

Featured in: Example 4 on page 753

MISSING
includes missing character variable values in the construction of the axes. It has no effect on numeric variables.

Interaction: overrides the NOMISS option for character variables

NOLEGEND
suppresses the legend at the top of each plot. The legend lists the names of the variables being plotted and the plotting symbols used in the plot.

NOMISS
excludes observations for which either variable is missing from the calculation of the axes. Normally, PROC PLOT draws an axis based on all the values of the variable being plotted, including points for which the other variable is missing.

Interaction: The HAXIS= option overrides the effect of NOMISS on the horizontal axis. The VAXIS= option overrides the effect on the vertical axis.

Interaction: NOMISS is overridden by MISSING for character variables.

Featured in: Example 10 on page 767

UNIFORM
uniformly scales axes across BY groups. Uniform scaling allows you to directly compare the plots for different values of the BY variables.

Restriction: You cannot use PROC PLOT with the UNIFORM option with an engine that supports concurrent access if another user is updating the data set at the same time.

VPERCENT=*percent(s)*
specifies one or more percentages of the available vertical space to use for each plot. If you use a percentage greater than 100, PROC PLOT prints sections of the plot on successive pages.

Alias: VPCT=

Default: 100

Featured in: Example 4 on page 753

See also: HPERCENT= on page 730

VTOH=*aspect-ratio*
specifies the aspect ratio (vertical to horizontal) of the characters on the output device. *aspect-ratio* is a positive real number. If you use the VTOH= option, PROC PLOT spaces tick marks so that the distance between horizontal tick marks is nearly equal to the distance between vertical tick marks. For example, if characters are twice as high as wide, specify VTOH=2.

Minimum: 0

Interaction: VTOH= has no effect if you use the HSPACE= and the VSPACE= options in the PLOT statement.

See also: HAXIS= on page 736 for a way to equate axes so that the given distance represents the same data range on both axes.

BY Statement

Produces a separate plot and starts a new page for each BY group.

Main discussion: "BY" on page 54

Featured in: Example 8 on page 762

BY <DESCENDING> *variable-1*
<...<DESCENDING> *variable-n*>
<NOTSORTED>;

Required Arguments

variable
specifies the variable that the procedure uses to form BY groups. You can specify more than one variable. If you do not use the NOTSORTED option in the BY statement, the observations in the data set must either be sorted by all the variables that you specify, or they must be indexed appropriately. Variables in a BY statement are called *BY variables*.

Options

DESCENDING
specifies that the observations are sorted in descending order by the variable that immediately follows the word DESCENDING in the BY statement.

NOTSORTED
specifies that observations are not necessarily sorted in alphabetic or numeric order. The data are grouped in another way, for example, chronological order.

The requirement for ordering or indexing observations according to the values of BY variables is suspended for BY-group processing when you use the NOTSORTED option. In fact, the procedure does not use an index if you specify NOTSORTED. The procedure defines a BY group as a set of contiguous observations that have the same values for all BY variables. If observations with the same values for the BY variables are not contiguous, the procedure treats each contiguous set as a separate BY group.

PLOT Statement

Requests the plots to be produced by PROC PLOT.

Tip: You can use multiple PLOT statements.

PLOT *plot-request(s)* </ *option(s)*>;

To do this	Use this option
Control the axes	
Specify the tick-mark values	HAXIS= and VAXIS=
Expand the axis	HEXPAND and VEXPAND
Specify the number of print positions	HPOS= and VPOS=
Reverse the order of the values	HREVERSE and VREVERSE
Specify the number of print positions between tick marks	HSPACE= and VSPACE=
Assign a value of zero to the first tick mark	HZERO and VZERO
Specify reference lines	
Draw a line perpendicular to the specified values on the axis	HREF= and VREF=
Specify a character to use to draw the reference line	HREFCHAR= and VREFCHAR=
Put a box around the plot	BOX
Overlay plots	OVERLAY
Produce a contour plot	
Draw a contour plot	CONTOUR
Specify the plotting symbol for one contour level	S*contour-level*=
Specify the plotting symbol for multiple contour levels	SLIST=
Label points on a plot	
List the penalty and the placement state of the points	LIST=
Force the labels away from the origin	OUTWARD=
Change default penalties	PENALTIES=
Specify locations for the placement of the labels	PLACEMENT=
Specify a split character for the label	SPLIT=
List all placement states in effect	STATES

Required Arguments

plot-request(s)

specifies the variables (vertical and horizontal) to plot and the plotting symbol to use to mark the points on the plot.

Each form of *plot-request(s)* supports a label variable. A label variable is preceded by a dollar sign ($) and specifies a variable whose values label the points on the plot. For example,

```
plot y*x $ label-variable
```

```
plot y*x='*' $ label-variable
```

See “Labeling Plot Points with Values of a Variable” on page 745 for more information. In addition, see Example 9 on page 764 and all the examples that follow it.

The *plot-request(s)* can be one or more of the following:

*vertical*horizontal* <$ *label-variable*>

specifies the variable to plot on the vertical axis and the variable to plot on the horizontal axis.

For example, the following statement requests a plot of Y by X:

```
plot y*x;
```

Y appears on the vertical axis, X on the horizontal axis.

This form of the plot request uses the default method of choosing a plotting symbol to mark plot points. When a point on the plot represents the values of one observation in the data set, PROC PLOT puts the character A at that point. When a point represents the values of two observations, the character B appears. When a point represents values of three observations, the character C appears, and so on through the alphabet. The character Z is used for the occurrence of 26 or more observations at the same printing position.

*vertical*horizontal='character'* <$ *label-variable*>

specifies the variables to plot on the vertical and horizontal axes and specifies a plotting symbol to mark each point on the plot. A single character is used to represent values from one or more observations.

For example, the following statement requests a plot of Y by X, with each point on the plot represented by a plus sign (+):

```
plot y*x='+';
```

*vertical*horizontal=variable* <$ *label-variable*>

specifies the variables to plot on the vertical and horizontal axes and specifies a variable whose values are to mark each point on the plot. The variable can be either numeric or character. The first (left-most) nonblank character in the formatted value of the variable is used as the plotting symbol (even if more than one value starts with the same letter). When more than one observation maps to the same plotting position, the value from the first observation marks the point. For example, in the following statement GENDER is a character variable with values of **FEMALE** and **MALE** : the values **F** and **M** mark each observation on the plot.

```
plot height*weight=gender;
```

Specifying Variable Lists in Plot Requests

You can use SAS variable lists in plot requests. For example, the following are valid plot requests:

Plot request	What is plotted
(a - - d)	**a*b a*c a*d b*c b*d** **c*d**
(x1 - x4)	**x1*x2** **x1*x3 x1*x4 x2*x3** **x2*x4 x3*x4**
(_numeric_)	All combinations of numeric variables
y*(x1 - x4)	**y*x1** **y*x2 y*x4 y*x4**

If both the vertical and horizontal specifications request more than one variable and a variable appears in both lists, it will not be plotted against itself. For example, the following statement does not plot B*B and C*C:

```
plot (a b c)*(b c d);
```

Specifying Combinations of Variables

The operator in *request* is either an asterisk (*) or a colon (:). An asterisk combines the variables in the lists to produce all possible combinations of *x* and *y* variables. For example, the following plot requests are equivalent:

```
plot (y1-y2) * (x1-x2);

plot y1*x1 y1*x2 y2*x1 y2*x2;
```

A colon combines the variables pairwise. Thus, the first variables of each list combine to request a plot, as do the second, third, and so on. For example, the following plot requests are equivalent:

```
plot (y1-y2) : (x1-x2);

plot y1*x1 y2*x2;
```

Options

BOX
draws a border around the entire plot, rather than just on the left side and bottom.

Featured in: Example 3 on page 752

CONTOUR<=*number-of-levels*>
draws a contour plot using plotting symbols with varying degrees of shading where *number-of-levels* is the number of levels for dividing the range of *variable*. The plot request must be of the form *vertical*horizontal=variable* where *variable* is a numeric variable in the data set. The intensity of shading is determined by the values of this variable.

When you use CONTOUR, PROC PLOT does not plot observations with missing values for *variable*.

Overprinting, if it is allowed by the OVP system option, is used to produce the shading. Otherwise, single characters varying in darkness are used. The CONTOUR option is most effective when the plot is dense.

Default: 10

Range: 1-10

Featured in: Example 7 on page 759

HAXIS=*axis-specification*
specifies the tick-mark values for the horizontal axis.

- *For numeric values*, *axis-specification* is either an explicit list of values, a BY increment, or a combination of both:

  n <*...n*>

  BY *increment*

  n TO *n* BY *increment*

  The values must be in either ascending or descending order. Use a negative value for *increment* to specify descending order. The specified values are spaced evenly along the horizontal axis even if the values are not uniformly distributed. Numeric values can be specified in the following ways:

HAXIS= value	Comments
`10 to 100 by 5`	Values appear in increments of 5, starting at 10 and ending at 100.
`by 5`	Values are incremented by 5. PROC PLOT determines the minimum and maximum values for the tick marks.
`10 100 1000 10000`	Values are not uniformly distributed. This specification produces a logarithmic plot. If PROC PLOT cannot determine the function implied by the axis specification, it uses simple linear interpolation between the points. To determine whether PROC PLOT correctly interpolates a function, you can use the DATA step to generate data that determines the function and see whether it appears linear when plotted. See Example 5 on page 755 for an example.
`1 2 10 to 100 by 5`	A combination of the previous specifications.

- *For character variables*, *axis-specification* is a list of unique values that are enclosed in quotes:

  'value-1' <*...'value-n'*>

  For example,

```
haxis='Paris' 'London' 'Tokyo'
```

  The character strings are case-sensitive. If a character variable has an associated format, *axis-specification* must specify the formatted value. The values can appear in any order.

- *For axis variables that contain date-time values*, *axis-specification* is either an explicit list of values or a starting and an ending value with an increment specified:

  'date-time-value'i <*...'date-time-value'i*>

  'date-time-value'i TO <*...'date-time-value'i*>
  <BY *increment*>

  'date-time-value'i
  any SAS date, time, or datetime value described for the SAS functions INTCK and INTNX. The suffix *i* is one of the following:

D	date
T	time
DT	datetime

  increment
  one of the valid arguments for the INTCK or INTNX functions: For dates, *increment* can be one of the following:

  DAY
  WEEK
  MONTH
  QTR
  YEAR

  For datetimes, *increment* can be one of the following:

  DTDAY
  DTWEEK
  DTMONTH
  DTQTR
  DTYEAR

  For times, *increment* can be one of the following:

  HOUR
  MINUTE
  SECOND

  For example,

```
haxis='01JAN95'd to '01JAN96'd
   by month

haxis='01JAN95'd to '01JAN96'd
   by qtr
```

  Note: You must use a FORMAT statement to print the tick-mark values in an understandable form. △

Interaction: You can use the HAXIS= and VAXIS= options with the VTOH= option to equate axes. If your data are suitable, use HAXIS=BY *n* and VAXIS=BY *n* with the same value for *n* and specify a value for the VTOH= option. The number of columns separating the horizontal tick marks is nearly equal to the number of lines separating the vertical tick marks times the value of the VTOH= option. In some cases, PROC PLOT cannot simultaneously use all three values and changes one or more of the values.

Featured in: Example 2 on page 751, Example 5 on page 755, and Example 6 on page 757

HEXPAND
expands the horizontal axis to minimize the margins at the sides of the plot and to maximize the distance between tick marks, if possible.

HEXPAND causes PROC PLOT to ignore information about the spacing of the data. Plots produced with this option waste less space but may obscure the nature of the relationship between the variables.

HPOS=*axis-length*
specifies the number of print positions on the horizontal axis. The maximum value of *axis-length* that allows a plot to fit on one page is three positions less than the value of the LINESIZE= system option because there must be space for the procedure to print information next to the vertical axis. The exact maximum depends on the number of characters in the vertical variable's values. If *axis-length* is too large to fit on a line, PROC PLOT ignores the option.

HREF=*value-specification*
draws lines on the plot perpendicular to the specified values on the horizontal axis. PROC PLOT includes the values you specify with the HREF= option on the horizontal axis unless you specify otherwise with the HAXIS= option.

For the syntax for *value-specification*, see HAXIS= on page 736.

Featured in: Example 8 on page 762

HREFCHAR='*character*'
specifies the character to use to draw the horizontal reference line.

Default: vertical bar (|)

See also: FORMCHAR= option on page 730 and HREF= on page 738

HREVERSE
reverses the order of the values on the horizontal axis.

HSPACE=*n*
specifies that a tick mark will occur on the horizontal axis at every *n*th print position, where *n* is the value of HSPACE=.

HZERO
assigns a value of zero to the first tick mark on the horizontal axis.

Interaction: PROC PLOT ignores HZERO if the horizontal variable has negative values or if the HAXIS= option specifies a range that does not begin with zero.

LIST<=*penalty-value*>
lists the horizontal and vertical axis values, the penalty, and the placement state of all points plotted with a penalty greater than or equal to *penalty-value*. If no plotted points have a penalty greater than or equal to *penalty-value*, then no list is printed.

Tip: LIST is equivalent to LIST=0.

See also: "Understanding Penalties" on page 746

Featured in: Example 11 on page 769

OUTWARD='*character*'
tries to force the point labels outward, away from the origin of the plot, by protecting positions next to symbols that match *character* that are in the direction of the origin (0,0). The algorithm tries to avoid putting the labels in the protected positions, so they usually move outward.

Tip: This option is useful only when you are labeling points with the values of a variable.

OVERLAY

overlays all plots specified in the PLOT statement on one set of axes. The variable names, or variable labels if they exist, from the first plot are used to label the axes. Unless you use the HAXIS= or the VAXIS= option, PROC PLOT automatically scales the axes in the way that best fits all the variables.

When the SAS system option OVP is in effect and overprinting is allowed, the plots are superimposed; otherwise, when NOOVP is in effect, PROC PLOT uses the plotting symbol from the first plot to represent points appearing in more than one plot. In such a case, the output includes a message telling you how many observations are hidden.

Featured in: Example 3 on page 752

PENALTIES<(*index-list*)>=*penalty-list*

changes the default penalties. The *index-list* provides the positions of the penalties in the list of penalties. The *penalty-list* contains the values you are specifying for the penalties indicated in the *index-list*. The *index-list* and the *penalty-list* can contain one or more integers. In addition, both *index-list* and *penalty-list* accept the form:

```
value TO value
```

See also: "Understanding Penalties" on page 746

Featured in: Example 13 on page 775

PLACEMENT=(*expression(s)*)

controls the placement of labels by specifying possible locations of the labels relative to their coordinates. Each *expression* consists of a list of one or more suboptions (H=, L=, S=, or V=) that are joined by an asterisk or a colon. PROC PLOT uses the asterisk and colon to expand each expression into combinations of values for the four possible suboptions. The asterisk creates every possible combination of values in the expression list. A colon creates only pairwise combinations. The colon takes precedence over the asterisk. With the colon, if one list is shorter than the other, the values in the shorter list are reused as necessary.

Use the following suboptions to control the placement:

H=*integer(s)*

specifies the number of horizontal spaces (columns) to shift the label relative to the starting position. Both positive and negative integers are valid. Positive integers shift the label to the right; negative integers shift it to the left. For example, you can use the H= suboption in the following way:

```
place=(h=0 1 -1 2 -2)
```

You can use the keywords BY ALT in this list. BY ALT produces a series of numbers whose signs alternate between positive and negative and whose absolute values change by one after each pair. For instance, the following PLACE= specifications are equivalent:

```
place=(h=0 -1 to -3 by alt)

place=(h=0 -1 1 -2 2 -3 3)
```

If the series includes zero, the zero appears twice. For example, the following PLACE= options are equivalent:

```
place=(h= 0 to 2 by alt)

place=(h=0 0 1 -1 2 -2)
```

Default: H=0

Range: –500 to 500

L=*integer(s)*

specifies the number of lines onto which the label may be split.

Default: L=1

Range: 1-200

S=*start-position(s)*

specifies where to start printing the label. The value for *start-position* can be one or more of the following

CENTER

the procedure centers the label around the plotting symbol.

RIGHT

the label starts at the plotting symbol location and continues to the right.

LEFT

the label starts to the left of the plotting symbol and ends at the plotting symbol location.

Default: CENTER

V=*integer(s)*

specifies the number of vertical spaces (lines) to shift the label relative to the starting position. V= behaves the same as the H= suboption, described earlier.

A new expression begins when a suboption is not preceded by an operator. Parentheses around each expression are optional. They make it easier to recognize individual expressions in the list. However, the entire expression list must be in parentheses, as shown in the following example. Table 30.1 on page 741 shows how this expression is expanded and describes each placement state.

```
place=((v=1)
        (s=right left : h=2 -2)
        (v=-1)
        (h=0 1 to 2 by alt * v=1 -1)
        (l=1 to 3 * v=1 to 2 by alt *
         h=0 1 to 2 by alt))
```

Each combination of values is a *placement state*. The procedure uses the placement states in the order in which they appear in the placement states list, so specify your most preferred placements first. For each label, the procedure tries all states, then uses the first state that places the label with minimum penalty. Once all labels are initially placed, the procedure cycles through the plot multiple times, systematically refining the placements. The refinement step tries to both minimize the penalties and to use placements nearer to the beginning of the states list. However, PROC PLOT uses a heuristic approach for placements, so the procedure does not always find the best set of placements.

Alias: PLACE=

Defaults: There are two defaults for the PLACE= option. If you are using a blank as your plotting symbol, the default placement state is PLACE=(S=CENTER : V=0 : H=0 : L=1), which centers the label. If you are using anything other than a blank, the default is PLACE=((S=RIGHT LEFT : H=2 –2) (V=1 –1 * H=0 1 -1 2 -2)). The default for labels placed with symbols includes multiple positions around the plotting symbol so the procedure has flexibility when placing labels on a crowded plot.

Tip: Use the STATES option to print a list of placement states.

See also: "Labeling Plot Points with Values of a Variable" on page 745

Featured in: Example 11 on page 769 and Example 12 on page 773

Table 30.1 Expanding an Expression List into Placement States

Expression	Placement state	Meaning
(V=1)	S=CENTER L=1 H=0 V=1	Center the label, relative to the point, on the line above the point. Use one line for the label.
(S=RIGHT LEFT : H=2 –2)	S=RIGHT L=1 H=2 V=0	Begin the label in the second column to the right of the point. Use one line for the label.
	S=LEFT L=1 H=–2 V=0	End the label in the second column to the left of the point. Use one line for the label.
(V=–1)	S=CENTER L=1 H=0 V=– 1	Center the label, relative to the point, on the line below the point. Use one line for the label.
(H=0 1 to 2 BY ALT * V=1 –1)	S=CENTER L=1 H=0 V=1	Center the label, relative to the point, on the line above the point.
	S=CENTER L=1 H=0 V=–1	Center the label, relative to the point, on the line below the point.
	S=CENTER L=1 H=1 V=1	From center, shift the label one column to the right on the line above the point.
	S=CENTER L=1 H=1 V=–1	From center, shift the label one column to the right on the line below the point.
	S=CENTER L=1 H=–1 V=1	From center, shift the label one column to the left on the line above the point.
	S=CENTER L=1 H=– 1 V=–1	From center, shift the label one column to the left on the line below the point.
	S=CENTER L=1 H=2 V=1 S=CENTER L=1 H=2 V=–1	From center, shift the labels two columns to the right, first on the line above the point, then on the line below.
	S=CENTER L=1 H=–2 V=1 S=CENTER L=1 H=–2 V=–1	From center, shift the labels two columns to the left, first on the line above the point, then on the line below.
(L=1 to 3 * V=1 to 2 BY ALT * H=0 1 to 2 BY ALT)	S=CENTER L=1 H=0 V=1	Center the label, relative to the point, on the line above the point. Use one line for the label.
	S=CENTER L=1 H=1 V=1 S=CENTER L=1 H=–1 V=1 S=CENTER L=1 H=2 V=1 S=CENTER L=1 H=–2 V=1	From center, shift the label one or two columns to the right or left on the line above the point. Use one line for the label.

Expression	Placement state	Meaning
	S=CENTER L=1 H=0 V=–1	Center the label, relative to the point, on the line below the point. Use one line for the label.
	S=CENTER L=1 H=1 V=–1 S=CENTER L=1 H=–1 V=–1 S=CENTER L=1 H=2 V=–1 S=CENTER L=1 H=–2 V=–1	From center, shift the label one or two columns to the right and the left on the line below the point.
	. . .	Use the same horizontal shifts on the line two lines above the point and on the line two lines below the point.
	S=CENTER L=1 H=– 2 V=–2	
	S=CENTER L=2 H=0 V=1	Repeat the whole process splitting the label over two lines. Then repeat it splitting the label over three lines.
	. . .	
	S=CENTER L=3 H=– 2 V=–2	

S*contour-level*='*character-list*'
specifies the plotting symbol to use for a single contour level. When PROC PLOT produces contour plots, it automatically chooses the symbols to use for each level of intensity. You can use the S= option to override these symbols and specify your own. You can include up to three characters in *character-list*. If overprinting is not allowed, PROC PLOT uses only the first character.

For example, to specify three levels of shading for the Z variable, use the following statement:

```
plot y*x=z /
   contour=3 s1='A' s2='+' s3='X0A';
```

You can also specify the plotting symbols as hexadecimal constants:

```
plot y*x=z /
   contour=3 s1='7A'x  s2='7F'x s3='A6'x;
```

This feature was designed especially for printers where the hex constants can represent grey-scale fill characters.

Range: 1 to the highest contour level (determined by the CONTOUR option).

See also: SLIST= and CONTOUR

SLIST='*character-list-1*' <...'*character-list-n*'>
specifies plotting symbols for multiple contour levels. Each *character-list* specifies the plotting symbol for one contour level: the first *character-list* for the first level, the second *character-list* for the second level, and so on. For example:

```
plot y*x=z /
   contour=5  slist='.' ':' '!' '=' '+O';
```

Default: If you omit a plotting symbol for each contour level, PROC PLOT uses the default symbols:

```
slist='.' ',' '-' '=' '+' 'O' 'X'
      'W' '*' '#'
```

Restriction: If you use the SLIST= option, it must be listed last in the PLOT statement.

See also: S*contour-level*= and CONTOUR=

SPLIT='*split-character*'

when labeling plot points, specifies where to split the label when the label spans two or more lines. The label is split onto the number of lines specified in the L= suboption to the PLACEMENT= option. If you specify a split character, the procedure always splits the label on each occurrence of that character, even if it cannot find a suitable placement. If you specify L=2 or more but do not specify a split character, the procedure tries to split the label on blanks or punctuation but will split words if necessary.

PROC PLOT shifts split labels as a block, not as individual fragments (a *fragment* is the part of the split label that is contained on one line). For example, to force **`This is a label`** to split after the **`a`** , change it to **`This is a*label`** and specify SPLIT='* '.

See also: "Labeling Plot Points with Values of a Variable" on page 745

STATES

lists all the placement states in effect. STATES prints the placement states in the order that you specify them in the PLACE= option.

VAXIS=*axis-specification*

specifies tick mark values for the vertical axis. VAXIS= follows the same rules as theHAXIS= option on page 736.

Featured in: Example 7 on page 759 and Example 12 on page 773

VEXPAND

expands the vertical axis to minimize the margins above and below the plot and to maximize the space between vertical tick marks, if possible.

See also: HEXPAND on page 738

VPOS=*axis-length*

specifies the number of print positions on the vertical axis. The maximum value for *axis-length* that allows a plot to fit on one page is 8 lines less than the value of the SAS system option PAGESIZE= because you must allow room for the procedure to print information under the horizontal axis. The exact maximum depends on the titles used, whether or not plots are overlayed, and whether or not CONTOUR is specified. If the value of *axis-length* specifies a plot that cannot fit on one page, the plot spans multiple pages.

See also: HPOS= on page 738

VREF=*value-specification*

draws lines on the plot perpendicular to the specified values on the vertical axis. PROC PLOT includes the values you specify with the VREF= option on the vertical axis unless you specify otherwise with the VAXIS= option. For the syntax for *value-specification*, see HAXIS= on page 736.

Featured in: Example 2 on page 751

VREFCHAR='*character*'

specifies the character to use to draw the vertical reference lines.

Default: horizontal bar (-)

See also: FORMCHAR= option on page 730, HREFCHAR= on page 738, and VREF= on page 743

VREVERSE
reverses the order of the values on the vertical axis.

VSPACE=*n*
specifies that a tick mark will occur on the vertical axis at every *n*th print position, where *n* is the value of VSPACE=.

VZERO
assigns a value of zero to the first tick mark on the vertical axis.

Interaction: PROC PLOT ignores the VZERO option if the vertical variable has negative values or if the VAXIS= option specifies a range that does not begin with zero.

Concepts: PLOT Procedure

RUN Groups

PROC PLOT is an interactive procedure. It remains active after a RUN statement is executed. Usually, SAS terminates a procedure after executing a RUN statement. Once you start the procedure, you can continue to submit any valid statements without resubmitting the PROC PLOT statement. Thus, you can easily experiment with changing labels, values of tick marks, and so forth. Any options submitted in the PROC PLOT statement remain in effect until you submit another PROC PLOT statement.

When you submit a RUN statement, PROC PLOT executes all the statements submitted since the last PROC PLOT or RUN statement. Each group of statements is called a *RUN group*. With each RUN group, PROC PLOT begins a new page and begins with the first item in the VPERCENT= and HPERCENT= lists, if any.

To terminate the procedure, submit a QUIT statement, a DATA statement, or a PROC statement. Like the RUN statement, each of these statements completes a RUN group. If you do not want to execute the statements in the RUN group, use the RUN CANCEL statement, which terminates the procedure immediately.

You can use the BY statement interactively. The BY statement remains in effect until you submit another BY statement or terminate the procedure.

See Example 11 on page 769 for an example of using RUN group processing with PROC PLOT.

Generating Data with Program Statements

When you generate data to be plotted, a good rule is to generate fewer observations than the number of positions on the horizontal axis. PROC PLOT then uses the increment of the horizontal variable as the interval between tick marks.

Because PROC PLOT prints one character for each observation, using SAS program statements to generate the data set for PROC PLOT can enhance the effectiveness of continuous plots. For example, suppose that you want to generate data in order to plot the following equation, for x ranging from 0 to 100:

$$y = 2.54 + 3.83x$$

You can submit these statements:

```
options linesize=80;
 data generate;
    do x=0 to 100 by 2;
       y=2.54+3.83*x;
       output;
    end;
 run;
 proc plot data=generate;
    plot y*x;
 run;
```

If the plot is printed with a LINESIZE= value of 80, about 75 positions are available on the horizontal axis for the X values. Thus, 2 is a good increment: 51 observations are generated, which is fewer than the 75 available positions on the horizontal axis.

However, if the plot is printed with a LINESIZE= value of 132, an increment of 2 produces a plot with a space between each plotting symbol. For a smoother line, a better increment is 1, since 101 observations are generated.

Labeling Plot Points with Values of a Variable

Pointer Symbols

When you are using a label variable and do not specify a plotting symbol or if the value of the variable you use as the plotting symbol is null ('00'x), PROC PLOT uses pointer symbols as plotting symbols. Pointer symbols associate a point with its label by pointing in the general direction of the label placement. PROC PLOT uses four different pointer symbols based on the value of the S= and V= suboptions in the PLACEMENT= option. The table below shows the pointer symbols:

S=	V=	Symbol
LEFT	any	<
RIGHT	any	>
CENTER	>0	^
CENTER	<=0	v

If you are using pointer symbols and multiple points coincide, PROC PLOT uses the number of points as the plotting symbol if it is between 2 and 9. If it is more than 9, the procedure uses an asterisk.

Note: Because of character set differences among operating environments, the pointer symbol for S=CENTER and V>0 may differ from the one shown here. △

Understanding Penalties

PROC PLOT assesses the quality of placements with penalties. If all labels are plotted with zero penalty, no labels collide and all labels are near their symbols. When it is not possible to place all labels with zero penalty, PROC PLOT tries to minimize the total penalty. Table 30.2 on page 746 gives a description of the penalty, the default value of the penalty, the index you use to reference the penalty, and the range of values you can specify if you change the penalties. Each penalty is described in more detail in Table 30.3 on page 746.

Table 30.2 Penalties Table

Penalty	Default penalty	Index	Range
not placing a blank	1	1	0-500
bad split, no split character specified	1	2	0-500
bad split with split character	50	3	0-500
free horizontal shift, *fhs*	2	4	0-500
free vertical shift, *fvs*	1	5	0-500
vertical shift weight, *vsw*	2	6	0-500
vertical/horizontal shift denominator, *vhsd*	5	7	1-500
collision state	500	8	0-10,000
(reserved for future use)		9-14	
not placing the first character	11	15	0-500
not placing the second character	10	16	0-500
not placing the third character	8	17	0-500
not placing the fourth character	5	18	0-500
not placing the fifth through 200th character	2	19-214	0-500

Table 30.3 on page 746 contains the index values from Table 30.2 on page 746 with a description of the corresponding penalty.

Table 30.3 Index Values for Penalties

1	a nonblank character in the plot collides with an embedded blank in a label, or there is not a blank or a plot boundary before or after each label fragment.
2	a split occurs on a nonblank or nonpunctuation character when you do not specify a split character.
3	a label is placed with a different number of lines than the L= suboption specifies, when you specify a split character.

4-7	a label is placed far away from the corresponding point. PROC PLOT calculates the penalty according to this (integer arithmetic) formula: $$[\mathrm{MAX}(\lvert H\rvert - fhs, 0) + vsw \times \mathrm{MAX}(\lvert V\rvert - (L+fvs+(V > 0))/2, 0)]/vhsd$$ Notice that penalties 4 through 7 are actually just components of the formula used to determine the penalty. Changing the penalty for a free horizontal or free vertical shift to a large value such as 500 has the effect of removing any penalty for a large horizontal or vertical shift. Example 6 on page 757 illustrates a case in which removing the horizontal shift penalty is useful.
8	a label may collide with its own plotting symbol. If the plotting symbol is blank, a collision state cannot occur. See "Collision States" on page 747 for more information.
15-214	a label character does not appear in the plot. By default, the penalty for not printing the first character is greater than the penalty for not printing the second character, and so on. By default, the penalty for not printing the fifth and subsequent characters is the same.

Note: Labels can share characters without penalty. △

Changing Penalties

You can change the default penalties with the PENALTIES= option in the PLOT statement. Because PROC PLOT considers penalties when it places labels, changing the default penalties can change the placement of the labels. For example, if you have labels that all begin with the same two-letter prefix, you might want to increase the default penalty for not printing the third, fourth, and fifth characters to 11, 10, and 8 and decrease the penalties for not printing the first and second characters to 2. The following PENALTIES= option accomplishes this change:

```
penalties(15 to 20)=2 2 11 10 8 2
```

This example extends the penalty list. The twentieth penalty of 2 is the penalty for not printing the sixth through 200th character. When the last index i is greater than 18, the last penalty is used for the (i – 14)th character and beyond.

You can also extend the penalty list by just specifying the starting index. For example, the following PENALTIES= option is equivalent to the one above:

```
penalties(15)=2 2 11 10 8 2
```

Collision States

Collision states are placement states that may cause a label to collide with its own plotting symbol. PROC PLOT usually avoids using collision states because of the large default penalty of 500 that is associated with them. PROC PLOT does not consider the actual length or splitting of any particular label when determining if a placement state is a collision state. The following are the rules that PROC PLOT uses to determine collision states:

- When S=CENTER, placement states that do not shift the label up or down sufficiently so that all of the label is shifted onto completely different lines from the symbol are collision states.
- When S=RIGHT, placement states that shift the label zero or more positions to the left without first shifting the label up or down onto completely different lines from the symbol are collision states.
- When S=LEFT, placement states that shift the label zero or more positions to the right without first shifting the label up or down onto completely different lines from the symbol are collision states.

Note: A collision state cannot occur if you do not use a plotting symbol. △

Reference Lines

PROC PLOT places labels and computes penalties before placing reference lines on a plot. The procedure does not attempt to avoid rows and columns that contain reference lines.

Hidden Label Characters

In addition to the number of hidden observations and hidden plotting symbols, PROC PLOT prints the number of hidden label characters. Label characters can be hidden by plotting symbols or other label characters.

Overlaying Label Plots

When you overlay a label plot and a nonlabel plot, PROC PLOT tries to avoid collisions between the labels and the characters of the nonlabel plot. When a label character collides with a character in a nonlabel plot, PROC PLOT adds the usual penalty to the penalty sum.

When you overlay two or more label plots, all label plots are treated as a single plot in avoiding collisions and computing hidden character counts. Labels of different plots never overprint, even with the OVP system option in effect.

Computational Resources Used for Label Plots

This section uses the following variables to discuss how much time and memory PROC PLOT uses to construct label plots:

n number of points with labels

len constant length of labels

s number of label pieces, or fragments

p number of placement states specified in the PLACE= option.

Time

For a given plot size, the time required to construct the plot is roughly proportional to $n \times len$. The amount of time required to split the labels is roughly proportional to ns^2. Generally, the more placement states you specify, the more time that PROC PLOT needs to place the labels. However, increasing the number of horizontal and vertical shifts gives PROC PLOT more flexibility to avoid collisions, often resulting in less time used to place labels.

Memory

PROC PLOT uses 24*p* bytes of memory for the internal placement state list. PROC PLOT uses $n\left(84 + 5len + 4s\left(1 + 1.5\left(s + 1\right)\right)\right)$ bytes for the internal list of labels. PROC PLOT buildsall plots in memory; each printing position uses one byte of memory. If you run out of memory, request fewer plots in each PLOT statement and put a RUN statement after each PLOT statement.

Results: PLOT Procedure

Scale of the Axes

Normally, PROC PLOT looks at the minimum difference between each pair of the five lowest ordered values of each variable (the *delta*) and ensures that there is no more than one of these intervals per print position on the final scaled axis, if possible. If there is not enough room to do this, and if PROC PLOT guesses that the data were artificially generated, it puts a fixed number of deltas in each print position. Otherwise, it ignores the value.

Printed Output

Each plot uses one full page unless the plot's size is changed by the VPOS= and HPOS= options in the PLOT statement, the VPERCENT= or HPERCENT= options in the PROC PLOT statement, or the PAGESIZE= and LINESIZE= system options. Titles, legends, and variable labels are printed at the top of each page. Each axis is labeled with the variable's name or, if it exists, the variable's label.

Normally, PROC PLOT begins a new plot on a new page. However, the VPERCENT= and HPERCENT= options enable you to print more than one plot on a page. VPERCENT= and HPERCENT= are described earlier in "PROC PLOT Statement" on page 729.

PROC PLOT always begins a new page after a RUN statement and at the beginning of a BY group.

Missing Values

If values of either of the plotting variables are missing, PROC PLOT does not include the observation in the plot. However, in a plot of Y*X, values of X with corresponding missing values of Y are included in scaling the X axis, unless the NOMISS option is specified in the PROC PLOT statement.

Hidden Observations

By default, PROC PLOT uses different plotting symbols (A, B, C, and so on) to represent observations whose values coincide on a plot. However, if you specify your own plotting symbol or if you use the OVERLAY option, you may not be able to recognize coinciding values.

If you specify a plotting symbol, PROC PLOT uses the same symbol regardless of the number of observations whose values coincide. If you use the OVERLAY option and overprinting is not in effect, PROC PLOT uses the symbol from the first plot request. In both cases, the output includes a message telling you how many observations are hidden.

Examples: PLOT Procedure

Example 1: Specifying a Plotting Symbol

Procedure features:
PLOT statement
plotting symbol in plot request

This example expands on Output 30.1 on page 726 by specifying a different plotting symbol.

Program

```
options nodate number pageno=1 linesize=80 pagesize=35;
```

The data set DJIA contains the high and low closing marks for the Dow Jones Industrial Average from 1954 to 1994. A DATA step on page 1621 creates this data set.

```
data djia;
      input Year @7 HighDate date7. High @24 LowDate date7. Low;
      format highdate lowdate date7.;
      datalines;
1954  31DEC54  404.39  11JAN54  279.87
1955  30DEC55  488.40  17JAN55  388.20
...more data lines...
1993  29DEC93 3794.33  20JAN93 3241.95
1994  31JAN94 3978.36  04APR94 3593.35
;
```

The plot request plots the values of High on the vertical axis and the values of Year on the horizontal axis. It also specifies an asterisk as the plotting symbol.

```
proc plot data=djia;
   plot high*year='*';
   title 'High Values of the Dow Jones Industrial Average';
   title2 'from 1954 to 1994';
run;
```

Output

PROC PLOT determines the tick marks and the scale of both axes.

```
       High Values of the Dow Jones Industrial Average                1
                      from 1954 to 1994

           Plot of High*Year.  Symbol used is '*'.

High |
     |
4000 +                                              *
     |                                             *
     |
     |                                            *
     |                                           *
3000 +                                          *
     |                                       * *
     |
     |
     |                                        *
2000 +                                      *
     |
     |                                     *
     |
     |                                   **
1000 +                 ***** ***  *** ***
     |             ****     *   **   *
     |        *****
     |      **
     |
   0 +
     |
     ---+---------+---------+---------+---------+---------+--
      1950      1960      1970      1980      1990      2000

                               Year
```

Example 2: Controlling the Horizontal Axis and Adding a Reference Line

Procedure features:
  PLOT statement options:
    HAXIS=
    VREF=

Data set: DJIA on page 750

This example specifies values for the horizontal axis and draws a reference line from the vertical axis.

Program

```
options nodate pageno=1 linesize=80 pagesize=35;
```

The plot request plots the values of High on the vertical axis and the values of Year on the horizontal axis. It also specifies an asterisk as the plotting symbol. HAXIS= specifies that the horizontal axis will show the values 1950 to 1995 in five-year increments. VREF= draws a reference line that extends from the value 3000 on the vertical axis.

```
proc plot data=djia;
   plot high*year='*' / haxis=1950 to 1995 by 5 vref=3000;
   title 'High Values of Dow Jones Industrial Average';
   title2 'from 1954 to 1994';
run;
```

Output

```
                  High Values of Dow Jones Industrial Average                  1
                               from 1954 to 1994

                    Plot of High*Year.  Symbol used is '*'.

High |
     |
4000 +                                                                      *
     |                                                                     *
     |
     |                                                                   *
     |                                                                  *
3000 +----------------------------------------------------------------*---------
     |                                                           *  *
     |
     |
     |                                                             *
2000 +                                                          *
     |
     |                                                        *
     |
     |                                                     **
1000 +                        * ** **   ** *    ** *  * **
     |                  ** **         *     * *     *
     |          ** ** *
     |      * *
     |
   0 +
     |
     -+-------+-------+-------+-------+-------+-------+-------+-------+-------+-
    1950    1955    1960    1965    1970    1975    1980    1985    1990    1995

                                         Year
```

Example 3: Overlaying Two Plots

Procedure features:

PLOT statement options

- BOX
- OVERLAY

Data set: DJIA on page 750

This example overlays two plots and puts a box around the plot.

Program

```
options nodate pageno=1 linesize=64 pagesize=30;
```

The first plot request plots High on the vertical axis, plots Year on the horizontal axis, and specifies an asterisk as a plotting symbol. The second plot request plots Low on the vertical axis, plots Year on the horizontal axis, and specifies an '**o**' as a plotting symbol. OVERLAY superimposes the second plot onto the first. BOX draws a box around the plot. OVERLAY and BOX apply to both plot requests.

```
proc plot data=djia;
   plot high*year='*'
        low*year='o' / overlay box;
   title 'Plot of Highs and Lows';
   title2 'for the Dow Jones Industrial Average';
run;
```

Output

```
                     Plot of Highs and Lows                    1
              for the Dow Jones Industrial Average

            Plot of High*Year.  Symbol used is '*'.
            Plot of Low*Year.   Symbol used is 'o'.

      ---+---------+---------+---------+---------+---------+---
 4000 +                                              *        +
      |                                             *         |
      |                                            * o        |
      |                                           *oo         |
 High |                                          *            |
      |                                       * *             |
      |                                           o           |
      |                                        *oo            |
 2000 +                                      * o              +
      |                                       o               |
      |                                     *o                |
      |                                   **o                 |
      |                ****** ************oo                  |
      |           *****oooooo*o o oooooooo                    |
      |      *****oooo           o                            |
      |      o                                                |
    0 +                                                       +
      ---+---------+---------+---------+---------+---------+---
       1950      1960      1970      1980      1990      2000

                                Year

NOTE: 7 obs hidden.
```

Example 4: Producing Multiple Plots per Page

Procedure features:

PROC PLOT statement options

HPERCENT=
VPERCENT=

Data set: DJIA on page 750

This example puts three plots on one page of output.

Program

```
options nodate pageno=1 linesize=120 pagesize=60;
```

VPERCENT= specifies that 50% of the vertical space on the page of output is used for each plot. HPERCENT= specifies that 50% of the horizontal space is used for each plot.

```
proc plot data=djia vpercent=50 hpercent=50;
```

This plot request plots the values of High on the vertical axis and the values of Year on the horizontal axis. It also specifies an asterisk as the plotting symbol.

```
   plot high*year='*';
```

This plot request plots the values of Low on the vertical axis and the values of Year on the horizontal axis. It also specifies an asterisk as the plotting symbol.

```
   plot low*year='o';
```

The first plot request plots High on the vertical axis, plots Year on the horizontal axis, and specifies an asterisk as a plotting symbol. The second plot request plots Low on the vertical axis, plots Year on the horizontal axis, and specifies an '**o**' as a plotting symbol. OVERLAY superimposes the second plot onto the first. BOX draws a box around the plot. OVERLAY and BOX apply to both plot requests.

```
   plot high*year='*' low*year='o' / overlay box;
title 'Plots of the Dow Jones Industrial Average';
title2 'from 1954 to 1994';
run;
```

Output

```
                                       Plots of the Dow Jones Industrial Average                                       1
                                                   from 1954 to 1994

         Plot of High*Year.  Symbol used is '*'.                        Plot of Low*Year.  Symbol used is 'o'.

4000 +                                            *           4000 +
     |                                           *                 |
     |                                                             |                                            o
     |                                          *                  |                                           o
High |                                         *               Low |                                          o
     |                                       **                    |
     |                                     *                       |
     |                                                             |                                        oo
     |                                      *                      |                                       o
2000 +                                    *                   2000 +
     |                                                             |                                     oo
     |                                   *                         |                                    o
     |                                 **                          |
     |                      **  *   ***                            |                                 ooo
     |              ********  ** ***                               |               o oo  ooo  oo o o
     |        ******                                               |         ooo oo o  oo   oo  o o o
     |    ****                                                     |     oooo   o
     |                                                             |    o
   0 +                                                           0 +
     -+---------+---------+---------+---------+---------+-         -+---------+---------+---------+---------+---------+-
    1950      1960      1970      1980      1990      2000        1950      1960      1970      1980      1990      2000

                              Year                                                          Year



         Plot of High*Year.  Symbol used is '*'.
         Plot of Low*Year.  Symbol used is 'o'.

     -+---------+---------+---------+---------+---------+-
4000 +                                            *      +
     |                                           *       |
     |                                          * o      |
     |                                         *oo       |
High |                                        *          |
     |                                     * *           |
     |                                         o         |
     |                                      *oo          |
2000 +                                    * o            +
     |                                     o             |
     |                                   *o              |
     |                                 **o               |
     |              ****** ************oo                |
     |         *****oooooo*o o oooooooo                  |
     |    *****oooo           o                          |
     |    o                                              |
   0 +                                                   +
     -+---------+---------+---------+---------+---------+-
    1950      1960      1970      1980      1990     2000

                             Year

NOTE: 7 obs hidden.
```

Example 5: Plotting Data on a Logarithmic Scale

Procedure features:

PLOT statement option
HAXIS=

This example uses a DATA step to generate data. The PROC PLOT step shows two plots of the same data — one plot without a horizontal axis specification and one plot with a logarithmic scale specified for the horizontal axis.

Program

```
options nodate pageno=1 linesize=80 pagesize=40;
```

The DATA step generates the values of X and Y. The values of X result from using specified values of Y as an exponent.

```
data equa;
   do Y=1 to 3 by .1;
      X=10**y;
      output;
   end;
run;
```

HPERCENT= makes room for two plots side-by-side by specifying that 50% of the horizontal space is used for each plot.

```
proc plot data=equa hpercent=50;
```

The plot requests plot Y on the vertical axis and X on the horizontal axis. HAXIS= specifies a logarithmic scale for the horizontal axis for the second plot.

```
    plot y*x;
    plot y*x / haxis=10 100 1000;
    title 'Two Plots with Different';
    title2 'Horizontal Axis Specifications';
run;
```

Output

```
                            Two Plots with Different                           1
                         Horizontal Axis Specifications

     Plot of Y*X.  A=1, B=2, etc.              Plot of Y*X.  A=1, B=2, etc.



  Y |                                       Y |
    |                                         |
3.0 +                                A    3.0 +                                A
2.9 +                         A           2.9 +                              A
2.8 +                    A                2.8 +                             A
2.7 +                A                    2.7 +                           A
2.6 +             A                       2.6 +                          A
2.5 +          A                          2.5 +                        A
2.4 +        A                            2.4 +                      A
2.3 +      A                              2.3 +                     A
2.2 +     A                               2.2 +                   A
2.1 +    A                                2.1 +                  A
2.0 +   A                                 2.0 +                A
1.9 +   A                                 1.9 +              A
1.8 +  A                                  1.8 +             A
1.7 +  A                                  1.7 +           A
1.6 + A                                   1.6 +          A
1.5 + A                                   1.5 +        A
1.4 + A                                   1.4 +      A
1.3 + A                                   1.3 +     A
1.2 + A                                   1.2 +   A
1.1 +A                                    1.1 +  A
1.0 +A                                    1.0 +A
    |                                         |
    -+---------------+---------------+        -+---------------+---------------+
     0              500           1000         10             100           1000

                     X                                         X
```

Example 6: Plotting Date Values on an Axis

Procedure features:
PLOT statement option
HAXIS=

This example shows how you can specify date values on an axis.

Program

```
options nodate pageno=1 linesize=120 pagesize=40;
```

EMERGENCY_CALLS contains the number of phone calls to an emergency help line.

```
data emergency_calls;
   input Date : date7. Calls @@;
   label calls='Number of Calls';
   datalines;
1APR94 134   11APR94 384  13FEB94 488
2MAR94 289   21MAR94 201  14MAR94 460
3JUN94 184   13JUN94 152  30APR94 356
4JAN94 179   14JAN94 128  16JUN94 480
5APR94 360   15APR94 350  24JUL94 388
6MAY94 245   15DEC94 150  17NOV94 328
7JUL94 280   16MAY94 240  25AUG94 280
8AUG94 494   17JUL94 499  26SEP94 394
9SEP94 309   18AUG94 248  23NOV94 590
19SEP94 356  24FEB94 201  29JUL94 330
10OCT94 222  25MAR94 183  30AUG94 321
11NOV94 294  26APR94 412   2DEC94 511
27MAY94 294  22DEC94 413  28JUN94 309
;
```

The plot request plots Calls on the vertical axis and Date on the horizontal axis. HAXIS= uses a monthly time for the horizontal axis. The notation **'1JAN94'd** is a date constant. The value **'1JAN95'd** ensures that the axis will have enough room for observations from December.

```
proc plot data=emergency_calls;
   plot calls*date / haxis='1JAN94'd to '1JAN95'd by month;
```

The FORMAT statement assigns the DATE7. format to Date.

```
   format date date7.;
   title 'Calls to City Emergency Services Number';
   title2 'Sample of Days for 1994';
run;
```

Output

PROC PLOT uses the variables' labels on the axes.

```
                                        Calls to City Emergency Services Number                                        1
                                                Sample of Days for 1994

                                Plot of Calls*Date.  Legend: A = 1 obs, B = 2 obs, etc.

      |
      |
  600 +                                                                                                   A
      |
      |
      |
      |                                                                                                     A
N 500 +                                                             A     A
u     |               A                                   A
m     |                        A
b     |
e     |                                    A                                                                      A
r 400 +                                                                                  A
      |                                A                              A
o     |                              A  A    A                                                         A
f     |                                                                A
      |                                                                          A                      A
C 300 +                                              A        A                    A                 A
a     |                    A                                     A             A
l     |
l     |                                       A  A                           A
s     |                                                                                      A
  200 +                  A       A
      |   A                       A                    A
      |                                                   A                                                     A
      |                             A
      |      A
  100 +
      |
      ---+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--
      01JAN94  01FEB94  01MAR94  01APR94  01MAY94  01JUN94  01JUL94  01AUG94  01SEP94  01OCT94  01NOV94  01DEC94 01JAN95

                                                             Date
```

Example 7: Producing a Contour Plot

Procedure features:

PLOT statement option

CONTOUR=

This example shows how to represent the values of three variables with a two-dimensional plot by setting one of the variables as the CONTOUR variable. The variables X and Y appear on the axes, and Z is the contour variable. Program statements are used to generate the observations for the plot, and the following equation describes the contour surface:

$$z = 46.2 + .09x - .0005x^2 + .1y - .0005y^2 + .0004xy$$

Program

```
options nodate pageno=1 linesize=64 pagesize=25;
```

The DATA step creates the CONTOURS data set.

```
data contours;
   format Z 5.1;
   do X=0 to 400 by 5;
      do Y=0 to 350 by 10;
         z=46.2+.09*x-.0005*x**2+.1*y-.0005*y**2+.0004*x*y;
         output;
      end;
   end;
run;
```

PROC PRINT prints the CONTOURS data set. The OBS= data set option limits the printing to only the first 5 observations.

```
proc print data=contours(obs=5) noobs;
   title 'CONTOURS Data Set';
   title2 'First 5 Observations Only';
run;
```

CONTOURS contains observations with values of X ranging from 0 to 400 by 5 and with values of Y ranging from 0 to 350 by 10.

```
                    CONTOURS Data Set                         1
                First 5 Observations Only

                         Z    X     Y

                      46.2    0     0
                      47.2    0    10
                      48.0    0    20
                      48.8    0    30
                      49.4    0    40
```

NOOVP ensures that overprinting is not used in the plot.

```
options nodate pageno=1 linesize=120 pagesize=60 noovp;
```

The plot request plots Y on the vertical axis, plots X on the horizontal axis, and specifies Z as the contour variable. CONTOUR=10 specifies that the plot will divide the values of Z into ten increments, and each increment will have a different plotting symbol.

```
proc plot data=contours;
plot y*x=z / contour=10;
    title 'A Contour Plot';
run;
```

Output

The shadings associated with the values of Z appear at the bottom of the plot. The plotting symbol # shows where high values of Z occur.

```
                                   A Contour Plot                                   1

                                Contour plot of Y*X.

  Y |
    |
350 +  ======++++++OOOOOOOOXXXXXXXXXXXWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWXXXXXXXXXXXOOOOOOOO
340 +  ====++++++OOOOOOOXXXXXXXXXXWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWXXXXXXXXXXXOOOOOOO
330 +  =++++++OOOOOOOXXXXXXXXXWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWXXXXXXXXXOOOOO
320 +  +++++OOOOOOOXXXXXXXXWWWWWWWWWWWWWW********************WWWWWWWWWWWWWWXXXXXXXXXOOOO
310 +  +++OOOOOOXXXXXXXXWWWWWWWWWWWW*****************************WWWWWWWWWWWXXXXXXXXOOOO
300 +  +OOOOOOXXXXXXXXWWWWWWWWWW***********************************WWWWWWWWWWXXXXXXXXOOO
290 +  OOOOOXXXXXXXWWWWWWWWWW****************************************WWWWWWWWWXXXXXXXOOO
280 +  OOOXXXXXXXWWWWWWWWW********************####********************WWWWWWWWWXXXXXXXOO
270 +  OXXXXXXXWWWWWWWWW**************##################***************WWWWWWWWXXXXXXXOO
260 +  XXXXXXWWWWWWWW*************#########################************WWWWWWWWXXXXXXXOO
250 +  XXXXWWWWWWWW************#############################************WWWWWWWWXXXXXXOO
240 +  XXXWWWWWWW***********#################################***********WWWWWWWWXXXXXXOO
230 +  XWWWWWWWW**********####################################**********WWWWWWWXXXXXXXOO
220 +  WWWWWWW**********######################################**********WWWWWWWXXXXXXOOO
210 +  WWWWWW*********########################################**********WWWWWWWXXXXXXOOO
200 +  WWWWW*********#########################################*********WWWWWWWXXXXXXOOOO
190 +  WWW**********##########################################*********WWWWWWWXXXXXXOOOO
180 +  WW*********###########################################*********WWWWWWWXXXXXXOOOOO
170 +  W*********############################################*********WWWWWWWXXXXXXOOOOO
160 +  W*********###########################################*********WWWWWWWXXXXXXOOOOO+
150 +  *********###########################################*********WWWWWWWXXXXXXOOOOO++
140 +  ********###########################################*********WWWWWWWXXXXXXOOOOO+++
130 +  ********##########################################*********WWWWWWWXXXXXXOOOOO++++
120 +  ********########################################**********WWWWWWWXXXXXXOOOOO+++++
110 +  ********#######################################**********WWWWWWWXXXXXXOOOOO+++++=
100 +  ********#####################################**********WWWWWWWXXXXXXOOOOOO+++++==
 90 +  ********###################################**********WWWWWWWWXXXXXXOOOOO+++++====
 80 +  *********################################***********WWWWWWWXXXXXXXOOOOO+++++====-
 70 +  **********############################************WWWWWWWWXXXXXXOOOOOO+++++====--
 60 +  ************######################**************WWWWWWWWXXXXXXXOOOOO+++++=====---
 50 +  ***************###############***************WWWWWWWWWXXXXXXXOOOOOO+++++====----'
 40 +  W******************************************WWWWWWWWWXXXXXXXOOOOOO+++++=====----''
 30 +  WW**************************************WWWWWWWWWWXXXXXXXOOOOOO+++++=====----''''
 20 +  WWWW********************************WWWWWWWWWWWXXXXXXXXOOOOOO++++++====-----''''.
 10 +  WWWWWW**************************WWWWWWWWWWWWWXXXXXXXXOOOOOO++++++=====----''''...
  0 +  WWWWWWWWWW*****************WWWWWWWWWWWWWWWXXXXXXXXOOOOOOO++++++=====----'''''....
    |
    ---+---------+---------+---------+---------+---------+---------+---------+---------+--
       0         50        100       150       200       250       300       350       400

                                         X

Symbol            z     Symbol            z     Symbol            z     Symbol            z     Symbol            z

.....     2.2 -  8.1     -----   14.0 - 19.9     +++++   25.8 - 31.7     XXXXX   37.6 - 43.5     *****   49.4 - 55.4
'''''     8.1 - 14.0     =====   19.9 - 25.8     OOOOO   31.7 - 37.6     WWWWW   43.5 - 49.4     #####   55.4 - 61.3
```

Example 8: Plotting BY Groups

Procedure features:
PLOT statement option
HREF=

Other features:
BY statement

This example shows BY group processing in PROC PLOT.

Program

```
options nodate pageno=1 linesize=80 pagesize=35;
```

EDUCATION contains educational data* about some U.S. states. DropoutRate is the percentage of high school dropouts. Expenditures is the dollar amount the state spends on each pupil. MathScore is the score of 8th graders on a standardized math test. Not all states participated in the math test. A DATA step on page 1622 creates this data set.

```
data education;
   input State $14. +1 Code $ DropoutRate Expenditures MathScore
         Region $;
   label dropout='Dropout Percentage - 1989'
          expend='Expenditure Per Pupil - 1989'
            math='8th Grade Math Exam - 1990';
   datalines;
Alabama        AL 22.3 3197 252 SE
Alaska         AK 35.8 7716 .   W
...more data lines...
New York       NY 35.0 .    261 NE
North Carolina NC 31.2 3874 250 SE
North Dakota   ND 12.1 3952 281 MW
Ohio           OH 24.4 4649 264 MW
;
```

PROC SORT sorts EDUCATION by Region so that Region can be used as the BY variable in PROC PLOT.

```
proc sort data=education;
   by region;
run;
```

The BY statement creates a separate plot for each value of Region.

* Data are from the U.S. Department of Education.

```
proc plot data=education;
   by region;
```

The plot request plots Expenditures on the vertical axis, plots DropoutRate on the horizontal axis, and specifies an asterisk as the plotting symbol. HREF= draws a reference line extending from 28.6 on the horizontal axis. The reference line represents the national average.

```
    plot expenditures*dropoutrate='*' / href=28.6;
    title 'Plot of Dropout Rate and Expenditure Per Pupil';
 run;
```

Output

PROC PLOT produces a plot for each BY group. Only the plots for **Midwest** and **Northeast** are shown.

```
                 Plot of Dropout Rate and Expenditure Per Pupil                1

---------------------------------- Region=MW -----------------------------------

             Plot of Expenditures*DropoutRate.  Symbol used is '*'.

    Expenditures |                                                  |
            5500 +                                                  |
                 |                                                  |
                 |                                                  |
                 |                                                  |
                 |                                                  | *
            5000 +                                                  |
                 |                                *                 |
                 |      *                                           |
                 |                                                  |
                 |                                       *          |
            4500 +                                                  |
                 |            *          *                          |
                 |           **                                *    |
                 |                                                  |
                 |                                                  |
            4000 +       *                                          |
                 |                                                  |
                 |                                                  |
                 |                                                  |
                 |                                                  |
            3500 +                                                  |
                 |                                                  |
                 ---+------------+------------+------------+------------+--
                   10           15           20           25           30

                                  Dropout Percentage - 1989
```

```
                 Plot of Dropout Rate and Expenditure Per Pupil                2

---------------------------------- Region=NE -----------------------------------

             Plot of Expenditures*DropoutRate.  Symbol used is '*'.

    Expenditures |                                     |
            8000 +                                     |
                 |                                     |
                 |                *                    |
                 |                                     |
                 |                                     |
            7000 +                                     |
                 |       *                             |
                 |                                     |
                 |                                     |
                 |                                     |
            6000 +                                    *|
                 |                               *     |
                 |                                     |
                 |                                     *
                 |                                     |
            5000 +                                     |
                 |                      *       *      |
                 |                                     |
                 |                                     |
                 |                                     |
            4000 +                                     |
                 |                                     |
                 ---+------------+------------+------------+------------+--
                   15           20           25           30           35

                                  Dropout Percentage - 1989

NOTE: 1 obs had missing values.
```

Example 9: Adding Labels to a Plot

Procedure features:

PLOT statement

label variable in plot request

Data set: EDUCATION on page 762

This example shows how to modify the plot request to label points on the plot with the values of variables. This example adds labels to the plot shown in Example 8 on page 762.

Program

```
options nodate pageno=1 linesize=80 pagesize=35;
```

PROC SORT sorts EDUCATION by Region so that Region can be used as the BY variable in PROC PLOT.

```
proc sort data=education;
   by region;
run;
```

The BY statement creates a separate plot for each value of Region.

```
proc plot data=education;
   by region;
```

The plot request plots Expenditures on the vertical axis, plots DropoutRate on the horizontal axis, and specifies an asterisk as the plotting symbol. The label variable specification (**`$ state`**) in the plot request labels each point on the plot with the name of the corresponding state. HREF= draws a reference line extending from 28.6 on the horizontal axis. The reference line represents the national average.

```
   plot expenditures*dropoutrate='*' $ state / href=28.6;
   title 'Plot of Dropout Rate and Expenditure Per Pupil';
run;
```

Output

PROC PLOT produces a plot for each BY group. Only the plots for **Midwest** and **Northeast** are shown.

```
                  Plot of Dropout Rate and Expenditure Per Pupil               1

 --------------------------------- Region=MW ----------------------------------

           Plot of Expenditures*DropoutRate$State.  Symbol used is '*'.

    Expenditures |                                                  |
            5500 +                                                  |
                 |                                                  |
                 |                                                  |
                 |                                                  |
                 |                                         Michigan *
            5000 +                                                  |
                 |                                * Illinois        |
                 |      * Minnesota                                 |
                 |                                                  |
                 |                                       * Ohio     |
            4500 +                                                  |
                 |            * Nebraska * Kansas                   |
                 |      Iowa ** Indiana                           * Missouri
                 |                                                  |
                 |                                                  |
            4000 +       * North Dakota                             |
                 |                                                  |
                 |                                                  |
                 |                                                  |
                 |                                                  |
            3500 +                                                  |
                 |                                                  |
                 ---+------------+------------+------------+------------+--
                   10           15           20           25           30

                                 Dropout Percentage - 1989
```

```
                Plot of Dropout Rate and Expenditure Per Pupil              2

---------------------------------- Region=NE -----------------------------------

          Plot of Expenditures*DropoutRate$State.  Symbol used is '*'.

    Expenditures |                                      |
            8000 +                                      |
                 |                                      |
                 |               * New Jersey           |
                 |                                      |
                 |                                      |
            7000 +                                      |
                 |      * Connecticut                   |
                 |                                      |
                 |                                      |
                 |                                      |
            6000 +                                     *|Massachusetts
                 |                                * Maryland
                 |                                      |
                 |                                      * Delaware
                 |                                      |
            5000 +                                      |
                 |                     * Maine * New Hampshire
                 |                                      |
                 |                                      |
                 |                                      |
            4000 +                                      |
                 |                                      |
                 ---+------------+------------+------------+------------+--
                   15           20           25           30           35

                                 Dropout Percentage - 1989

NOTE: 1 obs had missing values.
```

Example 10: Excluding Observations That Have Missing Values

Procedure features:

PROC PLOT statement option

NOMISS

Data set: EDUCATION on page 762

This example shows how missing values affect the calculation of the axes.

Program

```
options nodate pageno=1 linesize=80 pagesize=35;
```

PROC SORT sorts EDUCATION by Region so that Region can be used as the BY variable in PROC PLOT.

```
proc sort data=education;
   by region;
```

```
run;
```

NOMISS excludes observations that have a missing value for either of the axis variables.

```
proc plot data=education nomiss;
```

The BY statement creates a separate plot for each value of Region.

```
    by region;
```

The plot request plots Expenditures on the vertical axis, plots DropoutRate on the horizontal axis, and specifies an asterisk as the plotting symbol. The label variable specification (**`$ state`**) in the plot request labels each point on the plot with the name of the corresponding state. HREF= draws a reference line extending from 28.6 on the horizontal axis. The reference line represents the national average.

```
    plot expenditures*dropoutrate='*' $ state / href=28.6;
    title 'Plot of Dropout Rate and Expenditure Per Pupil';
run;
```

Output

PROC PLOT produces a plot for each BY group. Only the plot for the

`Northeast`

is shown. Because

`New York`

has a missing value for Expenditures, the observation is excluded and PROC PLOT does not use the value 35 for DropoutRate to calculate the horizontal axis. Compare the horizontal axis in this output with the horizontal axis in the plot for **`Northeast`** in Example 9 on page 764.

```
                Plot of Dropout Rate and Expenditure Per Pupil              1

---------------------------------- Region=NE -----------------------------------

          Plot of Expenditures*DropoutRate$State.  Symbol used is '*'.

Expenditures |                                                           |
        8000 +                                                           |
             |                                                           |
             |                    * New Jersey                           |
             |                                                           |
             |                                                           |
        7000 +                                                           |
             |    * Connecticut                                          |
             |                                                           |
             |                                                           |
             |                                                           |
        6000 +                                        Massachusetts *    |
             |                                             * Maryland    |
             |                                                           |
             |                                               Delaware *|
             |                                                           |
        5000 +                                                           |
             |                             * Maine         * New Hampshire
             |                                                           |
             |                                                           |
             |                                                           |
        4000 +                                                           |
             |                                                           |
             --+--------+--------+--------+--------+--------+--------+--------+-
              16       18       20       22       24       26       28       30

                                  Dropout Percentage - 1989

NOTE: 1 obs had missing values.
```

Example 11: Adjusting Labels on a Plot with the PLACEMENT= Option

Procedure features:

PLOT statement options

- label variable in plot request
- LIST=
- PLACEMENT=

Other features:

RUN group processing

This example illustrates the default placement of labels and how to adjust the placement of labels on a crowded plot. The labels are values of variable in the data set.*

This example also shows RUN group processing in PROC PLOT.

Program

```
options nodate pageno=1 linesize=120 pagesize=37;
```

CENSUS contains the variables CrimeRate and Density for selected states. CrimeRate is the number of crimes per 100,000 people. Density is the population density per square mile in the 1980 census. A DATA step on page 1616 creates this data set.

```
data census;
   input Density CrimeRate State $ 14-27 PostalCode $ 29-30;
   datalines;
263.3 4575.3 Ohio           OH
62.1 7017.1  Washington     WA

...more data lines...

111.6 4665.6 Tennessee      TN
120.4 4649.9 North Carolina NC
;
```

The plot request plots Density on the vertical axis, CrimeRate on the horizontal axis, and uses the first letter of the value of State as the plotting symbol. This makes it easier to match the symbol with its label. The label variable specification (**`$ state`**) in the plot request labels each point with the corresponding state name. BOX draws a box around the plot. LIST= lists the labels that have penalties greater than or equal to 1. HAXIS= and VAXIS= specify increments only. PROC PLOT uses the data to determine the range for the axes.

```
proc plot data=census;
   plot density*crimerate=state $ state / box list=1
        haxis=by 1000 vaxis=by 250;
   title 'A Plot of Population Density and Crime Rates';
run;
```

* The data are from the U.S. Bureau of the Census and the 1987 Uniform Crime Reports, FBI.

The labels **`Tennessee`**, **`South Carolina`**, **`Arkansas`**, **`Minnesota`**, and **`South Dakota`** have penalties. The default placement states do not provide enough possibilities for PROC PLOT to avoid penalties given the proximity of the points. Seven label characters are hidden.

```
                                      A Plot of Population Density and Crime Rates                                 1

                              Plot of Density*CrimeRate$State.  Symbol is value of State.

               ---+------------+------------+------------+------------+------------+------------+------------+---
       Density |                                                                                                |
           500 +                                                                                                +
               |                                                                                                |
               |                                                                                                |
               |                                               M Maryland                                       |
               |                                                                                                |
               |                                                                                                |
               |                                                                                                |
               |                                                                                                |
               |                                                                                                |
               |                                        D Delaware                                              |
               |                                                                                                |
               |                 P Pennsylvania    O Ohio                                                       |
           250 +                                                                                                +
               |                                                                                                |
               |                                              I Illinois                                        |
               |                                                                                       F Florida|
               |                                                                                                |
               |                              North Carolina                 C California                       |
               |                                TennNssee       Georgia                                         |
               |                    N New Hampshire  T     S South Garolina                                     |
               |    W West Virginia               A Alabama                                                     |
               |         Mississippi M  Vermont V    M Missouri         Washington W        T Texas             |
               |                          MinneAoArkMnsas             O Oklahoma                                |
               |        North Dakota          I Idaho                              O Oregon                     |
             0 +           S Nouth Dakota                                  N Nevada                             +
               ---+------------+------------+------------+------------+------------+------------+------------+---
                2000         3000         4000         5000         6000         7000         8000         9000

                                                           CrimeRate

NOTE: 7 label characters hidden.
```

```
                                      A Plot of Population Density and Crime Rates                                 2

                               List of Point Locations, Penalties, and Placement States
```

Label	Vertical Axis	Horizontal Axis	Penalty	Starting Position	Lines	Vertical Shift	Horizontal Shift
Tennessee	111.60	4665.6	2	Center	1	1	-1
South Carolina	103.40	5161.9	2	Right	1	0	2
Arkansas	43.90	4245.2	6	Right	1	0	2
Minnesota	51.20	4615.8	7	Left	1	0	-2
South Dakota	9.10	2678.0	11	Right	1	0	2

Because PROC PLOT is interactive, the procedure is still running at this point in the program. It is not necessary to restart the procedure to submit another plot request. LIST=1 produces no output because there are no penalties of 1 or greater.

```
   plot density*crimerate=state $ state / box list=1
        haxis=by 1000 vaxis=by 250
```

PLACEMENT= gives PROC PLOT more placement states to use to place the labels. PLACEMENT= contains three expressions. The first expression specifies the preferred positions for the label. The first expression resolves to placement states centered above the plotting symbol, with the label on one or two lines. The second and third expressions resolve to placement states that enable PROC PLOT to place the label in multiple positions around the plotting symbol.

```
        placement=((v=2 1 : l=2 1)
              ((l=2 2 1 : v=0 1 0) * (s=right left : h=2 -2))
              (s=center right left * l=2 1 * v=0 1 -1 2 *
               h=0 1 to 5 by alt));
title 'A Plot of Population Density and Crime Rates';
run;
```

Output

No collisions occur in the plot.

```
                           A Plot of Population Density and Crime Rates                                  3

                   Plot of Density*CrimeRate$State.  Symbol is value of State.

       ---+------------+------------+------------+------------+------------+------------+------------+---
 Density |                                                                                              |
     500 +                                                                                              +
         |                                                                                              |
         |                                         Maryland                                             |
         |                                            M                                                 |
         |                                                                                              |
         |                                                                                              |
         |                                                                                              |
         |                                                                                              |
         |                                  Delaware                                                    |
         |                                     D                                                        |
         |            Pennsylvania          Ohio                                                        |
         |                P                  O                                                          |
     250 +                                                                                              +
         |                                        Illinois                                              |
         |                                           I                                    Florida       |
         |                                                                                   F          |
         |                            North Carolina            California                              |
         |                   New                   South             C                                  |
         |   West         Hampshire   Alabama N   Carolina                                              |
         | Virginia           N               T     S       G Georgia                                   |
         |    W           Mississippi       A Tennessee                 Washington Texas                |
         |                     M  Vermont V    M Missouri    Oklahoma        W        T                 |
         |         South        Arkansas A    M Minnesota       O          Oregon                       |
         |         Dakota               I Idaho                   Nevada    O                           |
       0 +           S N North Dakota                              N                                    +
       ---+------------+------------+------------+------------+------------+------------+------------+---
         2000         3000         4000         5000         6000         7000         8000         9000

                                                 CrimeRate
```

Example 12: Adjusting Labeling on a Plot with a Macro

Procedure features:

PLOT statement options

label variable in plot request
PLACEMENT=

Data set: CENSUS on page 770

This example illustrates the default placement of labels and uses a macro to adjust the placement of labels. The labels are values of a variable in the data set.

Program

```
options nodate pageno=1 linesize=120 pagesize=37;
```

The %PLACE macro provides an alternative to using the PLACEMENT= option. The higher the value of **n**, the more freedom PROC PLOT has to place labels.

```
%macro place(n);
   %if &n > 13 %then %let n = 13;
       placement=(
   %if &n <= 0 %then (s=center); %else (h=2 -2 : s=right left);
   %if &n = 1 %then (v=1 * h=0 -1 to -2 by alt);
   %else %if &n = 2 %then (v=1 -1 * h=0 -1 to -5 by alt);
   %else %if &n > 2 %then (v=1 to 2 by alt * h=0 -1 to -10 by alt);
   %if &n > 3 %then
       (s=center right left * v=0 1 to %eval(&n - 2) by alt *
       h=0 -1 to %eval(-3 * (&n - 2)) by alt *
       l=1 to %eval(2 + (10 * &n - 35) / 30)); )
   %if &n > 4 %then penalty(7)=%eval((3 * &n) / 2);
%mend;
```

```
proc plot data=census;
```

The plot request plots Density on the vertical axis, CrimeRate on the horizontal axis, and uses the first letter of the value of State as the plotting symbol. The label variable specification (**$ state**) in the plot request labels each point with the corresponding state name. BOX draws a box around the plot. LIST= lists the labels that have penalties greater than or equal to 1. HAXIS= and VAXIS= specify increments only. PROC PLOT uses the data to determine the range for the axes. The PLACE macro determines the placement of the labels.

```
   plot density*crimerate=state $ state / box list=1
           haxis=by 1000 vaxis=by 250 %place(4);
   title 'A Plot of Population Density and Crime Rates';
run;
```

Output

No collisions occur in the plot.

```
                               A Plot of Population Density and Crime Rates                                     1

                       Plot of Density*CrimeRate$State.  Symbol is value of State.

        ---+------------+------------+------------+------------+------------+------------+------------+---
Density |                                                                                                |
    500 +                                                                                                +
        |                                                                                                |
        |                                                                                                |
        |                                               M Maryland                                       |
        |                                                                                                |
        |                                                                                                |
        |                                                                                                |
        |                                                                                                |
        |                                                                                                |
        |                                        D Delaware                                              |
        |                                                                                                |
        |                 P Pennsylvania    O Ohio                                                       |
    250 +                                                                                                +
        |                                                                                                |
        |                                              I Illinois                                        |
        |                                                                                       F Florida|
        |                                                                                                |
        |                              North Carolina                 C California                       |
        |                                    N Tennessee                                                 |
        |                    N New Hampshire  T     S       G Georgia                                    |
        |    W West Virginia       Alabama A  South Carolina                                             |
        |         Mississippi M  Vermont V    M Missouri         Washington W        T Texas             |
        |                      Arkansas A    M Minnesota       O Oklahoma                                |
        |      South Dakota            I Idaho                              O Oregon                     |
      0 +           S N North Dakota                                N Nevada                             +
        ---+------------+------------+------------+------------+------------+------------+------------+---
         2000         3000         4000         5000         6000         7000         8000         9000

                                                    CrimeRate
```

Example 13: Changing a Default Penalty

Procedure features:

PLOT statement option

PENALTIES=

Data set: CENSUS on page 770

This example demonstrates how changing a default penalty affects the placement of labels. The goal is to produce a plot that has labels that do not detract from how the points are scattered.

Program

```
options nodate pageno=1 linesize=120 pagesize=37;
```

The plot request plots Density on the vertical axis, CrimeRate on the horizontal axis, and uses the first letter of the value of State as the plotting symbol. The label variable specification (**`$ state`**) in the plot request labels each point with the corresponding state name.

```
proc plot data=census;
   plot density*crimerate=state $ state /
```

PLACEMENT= specifies that the preferred placement states are 100 columns to the left and the right of the point, on the same line with the point.

```
        placement=(h=100 to 10 by alt * s=left right)
```

PENALTIES(4)= changes the default penalty for a free horizontal shift to 500, which removes all penalties for a horizontal shift. LIST= shows how far PROC PLOT shifted the labels away from their respective points.

```
        penalties(4)=500  list=0
```

HAXIS= creates a horizontal axis long enough to leave space for the labels on the sides of the plot. VAXIS= specifies that the values on the vertical axis be in increments of 100.

```
        haxis=0 to 13000 by 1000 vaxis=by 100;
title 'A Plot of Population Density and Crime Rates';
run;
```

Output

```
                                 A Plot of Population Density and Crime Rates                                     1

                         Plot of Density*CrimeRate$State.  Symbol is value of State.

Density |
    500 +
        |
        |
        |
        |                                            M                                                   Maryland
    400 +
        |
        |
        |
        |
    300 +                                        D                                                       Delaware
        |
        |                         P           O                                                 Pennsylvania Ohio
        |
        |
    200 +                                          I                                                     Illinois
        |Florida                                                          F
        |                                                  C                                               California
        |
        |                                     T                                                North Carolina Tennessee
    100 +Georgia                        N            S    G                                New Hampshire South Carolina
        |                      W                 A M                                     Alabama Missouri West Virginia
        |Washington Texas                M     V  M                  W     T              Vermont Minnesota Mississippi
        |Oklahoma                              A             O                                                 Arkansas
        |Oregon                               I                      O                                            Idaho
      0 +                        S N                         N                         North Dakota South Dakota Nevada
        ---+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+--
           0      1000     2000    3000    4000    5000    6000    7000    8000    9000    10000   11000   12000   13000

                                                          CrimeRate

NOTE: 1 obs hidden.
```

A Plot of Population Density and Crime Rates 2

List of Point Locations, Penalties, and Placement States

Label	Vertical Axis	Horizontal Axis	Penalty	Starting Position	Lines	Vertical Shift	Horizontal Shift
Maryland	428.70	5477.6	0	Right	1	0	55
Delaware	307.60	4938.8	0	Right	1	0	59
Pennsylvania	264.30	3163.2	0	Right	1	0	65
Ohio	263.30	4575.3	0	Right	1	0	66
Illinois	205.30	5416.5	0	Right	1	0	56
Florida	180.00	8503.2	0	Left	1	0	-64
California	151.40	6506.4	0	Right	1	0	45
Tennessee	111.60	4665.6	0	Right	1	0	61
North Carolina	120.40	4649.9	0	Right	1	0	46
New Hampshire	102.40	3371.7	0	Right	1	0	52
South Carolina	103.40	5161.9	0	Right	1	0	52
Georgia	94.10	5792.0	0	Left	1	0	-42
West Virginia	80.80	2190.7	0	Right	1	0	76
Alabama	76.60	4451.4	0	Right	1	0	41
Missouri	71.20	4707.5	0	Right	1	0	47
Mississippi	53.40	3438.6	0	Right	1	0	68
Vermont	55.20	4271.2	0	Right	1	0	44
Minnesota	51.20	4615.8	0	Right	1	0	49
Washington	62.10	7017.1	0	Left	1	0	-49
Texas	54.30	7722.4	0	Left	1	0	-49
Arkansas	43.90	4245.2	0	Right	1	0	65
Oklahoma	44.10	6025.6	0	Left	1	0	-43
Idaho	11.50	4156.3	0	Right	1	0	69
Oregon	27.40	6969.9	0	Left	1	0	-53
South Dakota	9.10	2678.0	0	Right	1	0	67
North Dakota	9.40	2833.0	0	Right	1	0	52
Nevada	7.30	6371.4	0	Right	1	0	50

CHAPTER

31

The PMENU Procedure

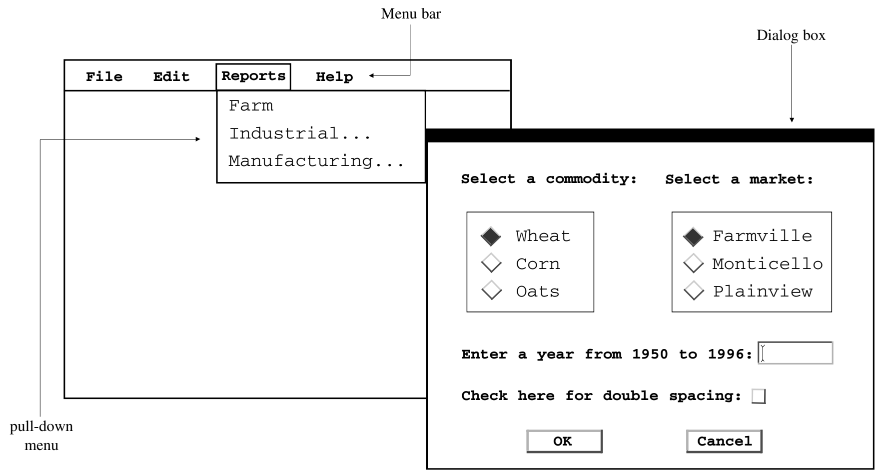
Overview: PMENU Procedure

The PMENU procedure defines menus that can be used in DATA step windows, macro windows, both SAS/AF and SAS/FSP windows, or in any SAS application that enables you to specify customized menus.

Menus can replace the command line as a way to execute commands. To activate menus, issue the PMENU command from any command line. *Menus must be activated in order for them to appear.*

When menus are activated, each active window has a *menu bar*, which lists items that you can select. Depending upon which item you select, SAS either processes a command, displays a menu or a submenu, or requests that you complete information in a *dialog box*. The *dialog box* is simply a box of questions or choices that require answers before an action can be performed. The following figure illustrates features that you can create with PROC PMENU.

Figure 31.1 Menu Bar, Pull-Down Menu, and Dialog Box



Note: A menu bar in some operating environments may appear as a popup menu or may appear at the bottom of the window. △

The PMENU procedure produces no immediately visible output. It simply builds a catalog entry of type PMENU that can be used later in an application.

Syntax: PMENU Procedure

Restriction: You must use at least one MENU statement followed by at least one ITEM statement.

Tip: Supports RUN group processing

Tip: Supports the Output Delivery System. See "Output Delivery System" on page 32 for details.

Reminder: You can also use appropriate global statements with this procedure. See Chapter 2, "Fundamental Concepts for Using Base SAS Procedures," on page 15 for a list.

PROC PMENU <CATALOG=<*libref.*>*catalog*>
<DESC '*entry-description*'>;

MENU *menu-bar*;
ITEM *command* <*option(s)*>;
ITEM *'menu-item'* <*option(s)*>;
DIALOG *dialog-box 'command-string field-number-specification'*;
CHECKBOX <ON> *#line @column 'text-for-selection'* <COLOR=*color*> <SUBSTITUTE=*'text-for-substitution'*>;
RADIOBOX DEFAULT=*button-number*;
RBUTTON <NONE> *#line @column 'text-for-selection'* <COLOR=*color*> <SUBSTITUTE=*'text-for-substitution'*>;
TEXT *#line @column field-description*

<ATTR=*attribute*> <COLOR=*color*>;
MENU *pull-down-menu*;
SELECTION *selection 'command-string'*;
SEPARATOR;
SUBMENU *submenu-name SAS-file*;

To do this	Use this statement
Define choices a user can make in a dialog box	CHECKBOX
Describe a dialog box that is associated with an item in a pull-down menu	DIALOG
Identify an item to be listed in a menu bar or in a pull-down menu	ITEM
Name the catalog entry or define a pull-down menu	MENU
List and define mutually exclusive choices within a dialog box	RADIOBOX and RBUTTON
Define a command that is submitted when an item is selected	SELECTION
Draw a line between items in a pull-down menu	SEPARATOR
Define a common submenu associated with an item	SUBMENU
Specify text and the input fields for a dialog box	TEXT

PROC PMENU Statement

Invokes the PMENU procedure and specifies where to store all PMENU catalog entries created in the PROC PMENU step.

PROC PMENU <CATALOG=<*libref.*>*catalog*>
<DESC *'entry-description'*>;

Options

CATALOG=*<libref.>catalog*
specifies the catalog in which you want to store PMENU entries.

Default: If you omit *libref*, the PMENU entries are stored in a catalog in the SASUSER data library. If you omit CATALOG=, the entries are stored in the SASUSER.PROFILE catalog.

Featured in: Example 1 on page 796

DESC *'entry-description'*
provides a description for the PMENU catalog entries created in the step.

Default: `Menu description`

Note: These descriptions are displayed when you use the CATALOG window in the windowing environment or the CONTENTS statement in the CATALOG procedure. △

CHECKBOX Statement

Defines choices that a user can make within a dialog box.

Restriction: Must be used after a DIALOG statement.

CHECKBOX <ON> #*line* @*column*
  'text-for-selection'
  <COLOR=*color*> <SUBSTITUTE=*'text-for-substitution'*>;

Required Arguments

column
specifies the column in the dialog box where the checkbox and text are placed.

line
specifies the line in the dialog box where the checkbox and text are placed.

text-for-selection
defines the text that describes this check box. This text appears in the window and, if the SUBSTITUTE= option is not used, is also inserted into the command in the preceding DIALOG statement when the user selects the check box.

Options

COLOR=*color*
defines the color of the check box and the text that describes it.

ON
indicates that by default this check box is active. If you use this option, you must specify it immediately after the CHECKBOX keyword.

SUBSTITUTE=*'text-for-substitution'*
specifies the text that is to be inserted into the command in the DIALOG statement.

Check Boxes in a Dialog Box

Each CHECKBOX statement defines a single item that the user can select independent of other selections. That is, if you define five choices with five CHECKBOX statements, the user can select any combination of these choices. When the user selects choices, the *text-for-selection* values that are associated with the selections are inserted into the command string of the previous DIALOG statement at field locations prefixed by an ampersand (&).

DIALOG Statement

Describes a dialog box that is associated with an item on a pull-down menu.

Restriction: Must be followed by at least one TEXT statement.

Featured in: Example 2 on page 798, Example 3 on page 801, and Example 4 on page 807

DIALOG *dialog-box* '*command-string*
field-number-specification';

Required Arguments

command-string
is the command or partial command that is executed when the item is selected. The limit of the *command-string* that results after the substitutions are made is the command-line limit for your operating environment. Typically, the command-line limit is approximately 80 characters.

The limit for '*command-string field-number-specification*' is 200 characters.

Note: If you are using PROC PMENU to submit any command that is valid only in the PROGRAM EDITOR window (such as the INCLUDE command), you must have the windowing environment running, and you must return control to the PROGRAM EDITOR window. △

dialog-box
is the same name specified for the DIALOG= option in a previous ITEM statement.

field-number-specification
can be one or more of the following:

@1...@*n*

%1...%*n*

&1...&*n*

You can embed the field numbers, for example @1, %1, or &1, in the command string and mix different types of field numbers within a command string. The numeric portion of the field number corresponds to the relative position of TEXT, RADIOBOX, and CHECKBOX statements, not to any actual number in these statements.

@1...@*n*
are optional TEXT statement numbers that can add information to the command before it is submitted. Numbers preceded by an at sign (@) correspond to TEXT statements that use the LEN= option to define input fields.

%1...%*n*
are optional RADIOBOX statement numbers that can add information to the command before it is submitted. Numbers preceded by a percent sign (%) correspond to RADIOBOX statements following the DIALOG statement.

Note: Keep in mind that the numbers correspond to RADIOBOX statements, not to RBUTTON statements. △

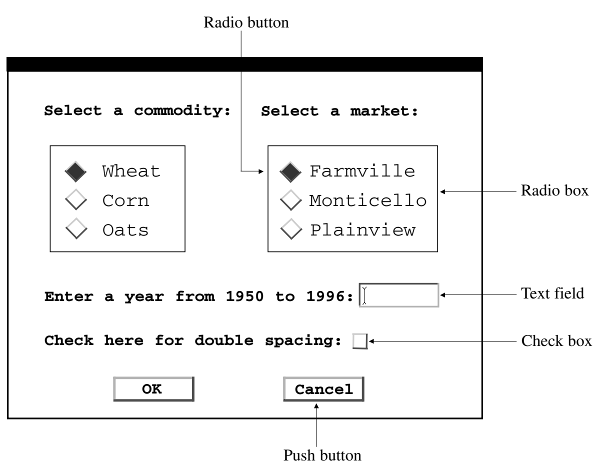
&1...&*n*
: are optional CHECKBOX statement numbers that can add information to the command before it is submitted. Numbers preceded by an ampersand (&) correspond to CHECKBOX statements following the DIALOG statement.

Note: To specify a literal @ (at sign), % (percent sign), or & (ampersand) in the *command-string*, use a double character: @@ (at signs), %% (percent signs), or && (ampersands). △

Details

- □ You cannot control the placement of the dialog box. The dialog box is not scrollable. The size and placement of the dialog box are determined by your windowing environment.
- □ To use the DIALOG statement, specify an ITEM statement with the DIALOG= option in the ITEM statement.
- □ The ITEM statement creates an entry in a menu bar or in a pull-down menu, and the DIALOG= option specifies which DIALOG statement describes the dialog box.
- □ You can use CHECKBOX, RADIOBOX, and RBUTTON statements to define the contents of the dialog box.
- □ Figure 31.2 on page 784 shows a typical dialog box. A dialog box can request information in three ways:
  - □ Fill in a field. Fields that accept text from a user are called *text fields*.
  - □ Choose from a list of mutually exclusive choices. A group of selections of this type is called a *radio box*, and each individual selection is called a *radio button*.
  - □ Indicate whether you want to select other independent choices. For example, you could choose to use various options by selecting any or all of the listed selections. A selection of this type is called a *check box*.

Figure 31.2 A Typical Dialog Box



Dialog boxes have two or more *push buttons*, such as OK and Cancel, automatically built into the box.* A *push button* causes an action to occur.

ITEM Statement

Identifies an item to be listed in a menu bar or in a pull-down menu.

Featured in: Example 1 on page 796

ITEM *command* <*option(s)*><*action-options*>;

ITEM *'menu-item'* <*option(s)*><*action-options*>;

To do this	Use this option
Specify the action for the item	
Associate the item with a dialog box	DIALOG=
Associate the item with a pull-down menu	MENU=
Associate the item with a command	SELECTION=
Associate the item with a common submenu	SUBMENU=
Specify help text for an item	HELP=
Define a key that can be used instead of the pull-down menu	ACCELERATE=
Indicate that the item is not an active choice in the window	GRAY
Provide an ID number for an item	ID=
Define a single character that can select the item	MNEMONIC=
Place a check box or a radio button next to an item	STATE=

Required Arguments

command
a single word that is a valid SAS command for the window in which the menu appears. Commands that are more than one word, such as WHERE CLEAR, must be in single quotes. The *command* appears in uppercase letters on the menu bar.

If you want to control the casing of a SAS command on the menu, enclose the command in single quotes and the case that you used then appears on the menu.

menu-item
a word or text string, enclosed in quotes, that describes the action that occurs when the user selects this item. A menu item should not begin with a percent sign (%).

* The actual names of the push buttons vary in different windowing environments.

Options

ACCELERATE=*name-of-key*
defines a key sequence that can be used instead of selecting an item. When the user presses the key sequence, it has the same effect as selecting the item from the menu bar or pull-down menu.

Restriction: The functionality of this option is limited to only a few characters. For details, see the SAS documentation for your operating environment.

Restriction: This option is not available in all operating environments. If you include this option and it is not available in your operating environment, the option is ignored.

action-option
is one of the following:

DIALOG=*dialog-box*
the name of an associated DIALOG statement, which displays a dialog box when the user selects this item.

Featured in: Example 3 on page 801

MENU=*pull-down-menu*
the name of an associated MENU statement, which displays a pull-down menu when the user selects this item.

Featured in: Example 1 on page 796

SELECTION=*selection*
the name of an associated SELECTION statement, which submits a command when the user selects this item.

Featured in: Example 1 on page 796

SUBMENU=*submenu*
the name of an associated SUBMENU statement, which displays a pmenu entry when the user selects this item.

Featured in: Example 1 on page 796

If no DIALOG=, MENU=, SELECTION=, or SUBMENU= option is specified, the *command* or *menu-item* text string is submitted as a command-line command when the user selects the item.

GRAY
indicates that the item is not an active choice in this window. This option is useful when you want to define standard lists of items for many windows, but not all items are valid in all windows. When this option is set and the user selects the item, no action occurs.

HELP='*help-text*'
specifies text that is displayed when the user displays the menu item. For example, if you use a mouse to pull down a menu, hold the mouse button on the item and the text is displayed.

Restriction: This option is not available in all operating environments. If you include this option and it is not available in your operating environment, the option is ignored.

Tip: The place where the text is displayed is operating environment-specific.

ID=*integer*
a value that is used as an identifier for an item in a pull-down menu. This identifier is used within a SAS/AF application to selectively gray or ungray items in a menu or to set the state of an item as a check box or a radio button.

Minimum: 3001

Restriction: Integers from 0 - 3000 are reserved for operating environment and SAS System use.

Restriction: This option is not available in all operating environments. If you include this option and it is not available in your operating environment, the option is ignored.

Tip: ID= is useful with the WINFO function in SAS Screen Control Language.

Tip: You can use the same ID for more than one item.

See also: STATE= option on page 787

MNEMONIC=*character*

underlines the first occurrence of *character* in the text string that appears on the pull-down menu. The *character* must be in the text string.

The *character* is typically used in combination with another key, such as ALT. When you use the key sequence, it has the same effect as putting your cursor on the item. But it *does not* invoke the action that the item controls.

Restriction: This option is not available in all operating environments. If you include this option and it is not available in your operating environment, the option is ignored.

STATE=CHECK|RADIO

provides the ability to place a check box or a radio button next to an item that has been selected.

Tip: STATE= is used with the ID= option and the WINFO function in SAS Screen Control Language.

Restriction: This option is not available in all operating environments. If you include this option and it is not available in your operating environment, the option is ignored.

Defining Items on the Menu Bar

You must use ITEM statements to name all the items that appear in a menu bar. You also use the ITEM statement to name the items that appear in any pull-down menus. The items that you specify in the ITEM statement can be commands that are issued when the user selects the item, or they can be descriptions of other actions that are performed by associated DIALOG, MENU, SELECTION, or SUBMENU statements.

All ITEM statements for a menu must be placed immediately after the MENU statement and before any DIALOG, SELECTION, SUBMENU, or other MENU statements. In some operating environments, you can insert SEPARATOR statements between ITEM statements to produce lines separating groups of items in a pull-down menu. See “SEPARATOR Statement” on page 791 for more information.

CAUTION:

If you specify a menu bar that is too long for the window, it might be truncated or wrapped to multiple lines. △

MENU Statement

Names the catalog entry that stores the menus or defines a pull-down menu.

Featured in: Example 1 on page 796

MENU *menu-bar*;

MENU *pull-down-menu*;

Required Arguments

One of the following arguments is required:

menu-bar
names the catalog entry that stores the menus.

pull-down-menu
names the pull-down menu that appears when the user selects an item in the menu bar. The value of *pull-down-menu* must match the *pull-down-menu* name that is specified in the MENU= option in a previous ITEM statement.

Defining Pull-Down Menus

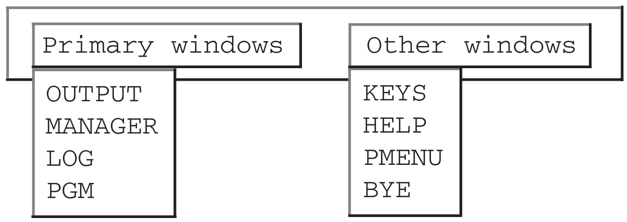
When used to define a pull-down menu, the MENU statement must follow an ITEM statement that specifies the MENU= option. Both the ITEM statement and the MENU statement for the pull-down menu must be in the same RUN group as the MENU statement that defines the menu bar for the PMENU catalog entry.

For both menu bars and pull-down menus, follow the MENU statement with ITEM statements that define each of the items that appear on the menu. Group all ITEM statements for a menu together. For example, the following PROC PMENU step creates one catalog entry, WINDOWS, which produces a menu bar with two items, **`Primary windows`** and **`Other windows`**. When you select one of these items, a pull-down menu is displayed.

```
libname proclib 'SAS-data-library';

proc pmenu cat=proclib.mycat;

      /* create catalog entry */
   menu windows;
   item 'Primary windows' menu=prime;
   item 'Other windows' menu=other;

      /* create first pull-down menu */
   menu prime;
   item output;
   item manager;
   item log;
   item pgm;

      /* create second pull-down menu */
   menu other;
   item keys;
   item help;
   item pmenu;
   item bye;

   /* end of run group */
run;
```

The following figure shows the resulting menu selections.

Figure 31.3 Pull-Down Menu



RADIOBOX Statement

Defines a box that contains mutually exclusive choices within a dialog box.

Restriction: Must be used after a DIALOG statement.

Restriction: Must be followed by one or more RBUTTON statements.

Featured in: Example 3 on page 801

RADIOBOX DEFAULT=*button-number*;

Required Arguments

DEFAULT=*button-number*
indicates which radio button is the default.

Default: 1

Details

The RADIOBOX statement indicates the beginning of a list of selections. Immediately after the RADIOBOX statement, you must list an RBUTTON statement for each of the selections the user can make. When the user makes a choice, the text value that is associated with the selection is inserted into the command string of the previous DIALOG statement at field locations prefixed by a percent sign (%).

RBUTTON Statement

Lists mutually exclusive choices within a dialog box.

Restriction: Must be used after a RADIOBOX statement.

Featured in: Example 3 on page 801

RBUTTON <NONE> #*line* @*column*
'text-for-selection' <COLOR=*color*> <SUBSTITUTE=*'text-for-substitution'*>;

Required Arguments

column
specifies the column in the dialog box where the radio button and text are placed.

line
specifies the line in the dialog box where the radio button and text are placed.

text-for-selection
defines the text that appears in the dialog box and, if the SUBSTITUTE= option is not used, defines the text that is inserted into the command in the preceding DIALOG statement.

CAUTION:
Be careful not to overlap columns and lines when placing text and radio buttons. You receive an error message if you overlap text or buttons. In addition, specify space between other text and a radio button. △

Options

COLOR=*color*
defines the color of the radio button and the text that describes the button.

Restriction: This option is not available in all operating environments. If you include this option and it is not available in your operating environment, the option is ignored.

NONE
defines a button that indicates none of the other choices. Defining this button enables the user to ignore any of the other choices. No characters, including blanks, are inserted into the DIALOG statement.

Restriction: If you use this option, it must occur immediately after the RBUTTON keyword.

SUBSTITUTE=*'text-for-substitution'*
specifies the text that is to be inserted into the command in the DIALOG statement.

Featured in: Example 3 on page 801

SELECTION Statement

Defines a command that is submitted when an item is selected.

Restriction: Must be used after an ITEM statement

Featured in: Example 1 on page 796 and Example 4 on page 807

SELECTION *selection 'command-string'*;

Required Arguments

selection
is the same name specified for the SELECTION= option in a previous ITEM statement.

command-string
is a text string, enclosed in quotes, that is submitted as a command-line command when the user selects this item. There is a limit of 200 characters for *command-string*. However, the command-line limit of approximately 80 characters cannot be exceeded. The command-line limit differs slightly for various operating environments.

Details

You define the name of the item in the ITEM statement and specify the SELECTION= option to associate the item with a subsequent SELECTION statement. The SELECTION statement then defines the actual command that is submitted when the user chooses the item in the menu bar or pull-down menu.

You are likely to use the SELECTION statement to define a command string. You create a simple alias by using the ITEM statement, which invokes a longer command string that is defined in the SELECTION statement. For example, you could include an item in the menu bar that invokes a WINDOW statement to allow data entry. The actual commands that are processed when the user selects this item are the commands to include and submit the application.

Note: If you are using PROC PMENU to issue any command that is valid only in the PROGRAM EDITOR window (such as the INCLUDE command), you must have the windowing environment running, and you must return control to the PROGRAM EDITOR window. △

SEPARATOR Statement

Draws a line between items on a pull-down menu.

Restriction: Must be used after an ITEM statement.

Restriction: Not available in all operating environments.

SEPARATOR;

SUBMENU Statement

Specifies the SAS file that contains a common submenu associated with an item.

Featured in: Example 1 on page 796

SUBMENU *submenu-name SAS-file*;

Required Arguments

submenu-name
specifies a name for the submenu statement. To associate a submenu with a menu item, *submenu-name* must match the submenu name specified in the SUBMENU= action-option in the ITEM statement.

SAS-file
specifies the name of the SAS file that contains the common submenu.

TEXT Statement

Specifies text and the input fields for a dialog box.

Restriction: Can be used only after a DIALOG statement.

Featured in: Example 2 on page 798

TEXT *#line @column field-description*
<ATTR=*attribute*> <COLOR=*color*>;

Required Arguments

column
specifies the starting column for the text or input field.

field-description
defines how the TEXT statement is used. The *field-description* can be one of the following:

LEN=*field-length*
is the length of an input field in which the user can enter information. If the LEN= argument is used, the information entered in the field is inserted into the command string of the previous DIALOG statement at field locations prefixed by an at sign (@).

Featured in: Example 2 on page 798

'*text*'
is the text string that appears inside the dialog box at the location defined by *line* and *column*.

line
specifies the line number for the text or input field.

Options

ATTR=*attribute*

defines the attribute for the text or input field. Valid attribute values are

- BLINK
- HIGHLIGH
- REV_VIDE
- UNDERLIN

Restriction: This option is not available in all operating environments. If you include this option and it is not available in your operating environment, the option is ignored.

Restriction: Your hardware may not support all of these attributes.

COLOR=*color*

defines the color for the text or input field characters. These are the color values that you can use:

BLACK	BROWN
GRAY	MAGENTA
PINK	WHITE
BLUE	CYAN
GREEN	ORANGE
RED	YELLOW

Restriction: This option is not available in all operating environments. If you include this option and it is not available in your operating environment, the option is ignored.

Restriction: Your hardware may not support all of these colors.

Concepts: PMENU Procedure

Procedure Execution

You can define multiple menus by separating their definitions with RUN statements. A group of statements that ends with a RUN statement is called a *RUN group*. You must completely define a PMENU catalog entry before submitting a RUN statement. You do not have to restart the procedure after a RUN statement.

You must include an initial MENU statement that defines the menu bar, and you must include all ITEM statements and any SELECTION, MENU, SUBMENU, and DIALOG statements as well as statements that are associated with the DIALOG statement within the same RUN group. For example, the following statements define two separate PMENU catalog entries. Both are stored in the same catalog, but each PMENU catalog entry is independent of the other. In the example, both PMENU catalog entries create menu bars that simply list windowing environment commands the user can select and execute:

```
libname proclib 'SAS-data-library';
```

```
proc pmenu catalog=proclib.mycat;
   menu menu1;
   item end;
   item bye;
run;

   menu menu2;
   item end;
   item pgm;
   item log;
   item output;
run;
```

When you submit these statements, you receive a message that says that the PMENU entries have been created. To display one of these menu bars, you must associate the PMENU catalog entry with a window and then activate the window with the menus turned on, as described in “Steps for Building and Using PMENU Catalog Entries” on page 794.

Ending the Procedure

Submit a QUIT, DATA, or new PROC statement to execute any statements that have not executed and end the PMENU procedure. Submit a RUN CANCEL statement to cancel any statements that have not executed and end the PMENU procedure.

Steps for Building and Using PMENU Catalog Entries

In most cases, building and using PMENU entries requires the following steps:

1 Use PROC PMENU to define the menu bars, pull-down menus and other features that you want. Store the output of PROC PMENU in a SAS catalog.

2 Define a window using SAS/AF and SAS/FSP Software, or the WINDOW or %WINDOW statement in base SAS software.

3 Associate the PMENU catalog entry created in step 1 with a window by using one of the following:
   - the MENU= option in the WINDOW statement in base SAS software. See “Associating a Menu with a Window” on page 810.
   - the MENU= option in the %WINDOW statement in the macro facility.
   - the **`Command Menu`** field in the GATTR window in PROGRAM entries in SAS/AF Software.
   - the Keys, Pmenu, and Commands window in a FRAME entry in SAS/AF Software. See Example 5 on page 813.
   - the PMENU function in SAS/AF and SAS/FSP Software.
   - the SETPMENU command in SAS/FSP Software. See Example 1 on page 796.

4 Activate the window you have created. Make sure that the menus are turned on.

Templates for Coding PROC PMENU Steps

The following coding templates summarize how to use the statements in the PMENU procedure. Refer to descriptions of the statements for more information:

- Build a simple menu bar. All items on the menu bar are windowing environment commands:

```
proc pmenu;
   menu menu-bar;
   item command;
   ...more-ITEM-statements...
run;
```

□ Create a menu bar with an item that produces a pull-down menu:

```
proc pmenu;
    menu menu-bar;
    item 'menu-item' menu=pull-down-menu;
    ...more-ITEM-statements...
    menu pull-down-menu;
    ...ITEM-statements-for-pull-down-menu...
run;
```

□ Create a menu bar with an item that submits a command other than that which appears on the menu bar:

```
proc pmenu;
   menu menu-bar;
   item 'menu-item' selection=selection;
   ...more-ITEM-statements...
   selection selection 'command-string';
run;
```

□ Create a menu bar with an item that opens a dialog box, which displays information and requests text input:

```
proc pmenu;
   menu menu-bar;
   item 'menu-item' menu=pull-down-menu;
   ...more-ITEM-statements...
   menu pull-down-menu;
      item 'menu-item' dialog=dialog-box;
      dialog dialog-box 'command @1';
         text #line @column 'text';
         text #line @column LEN=field-length;
run;
```

□ Create a menu bar with an item that opens a dialog box, which permits one choice from a list of possible values:

```
proc pmenu;
   menu menu-bar;
   item 'menu-item' menu=pull-down-menu;
   ...more-ITEM-statements...
   menu pull-down-menu;
      item 'menu-item' dialog=dialog-box;
      dialog dialog-box 'command %1';
         text #line @column 'text';
         radiobox default=button-number;
         rbutton #line @column
                 'text-for-selection';
         ...more-RBUTTON-statements...
run;
```

□ Create a menu bar with an item that opens a dialog box, which permits several independent choices:

```
proc pmenu;
   menu menu-bar;
   item 'menu-item' menu=pull-down-menu;
   ...more-ITEM-statements...
   menu pull-down-menu;
      item 'menu-item' dialog=dialog-box;
      dialog dialog-box 'command &1';
         text #line @column 'text';
         checkbox #line @column 'text';
         ...more-CHECKBOX-statements...
run;
```

Examples: PMENU Procedure

The windows in these examples were produced in the UNIX environment and may appear slightly different from the same windows in other operating environments.

You should know the operating environment-specific system options that can affect how menus are displayed and merged with existing SAS menus. For details, see the SAS documentation for your operating environment.

Example 1: Building a Menu Bar for an FSEDIT Application

Procedure features:

PROC PMENU statement option:
- CATALOG=

ITEM statement options:
- MENU=
- SELECTION=
- SUBMENU=

MENU statement

SELECTION statement

SUBMENU statement

This example creates a menu bar that can be used in an FSEDIT application to replace the default menu bar. The selections available on these pull-down menus do not enable end users to delete or duplicate observations.

Program

```
libname proclib 'SAS-data-library';
```

CATALOG= specifies PROCLIB.MENUCAT as the catalog that stores the menus.

```
proc pmenu catalog=proclib.menucat;
```

The MENU statement specifies PROJECT as the name of the catalog entry. The menus are stored in the catalog entry PROCLIB.MENUCAT.PROJECT.PMENU.

```
menu project;
```

The ITEM statements specify the items for the menu bar. The value of the MENU= option is used in a subsequent MENU statement. The **Edit** item uses a common predefined submenu; the menus for the other items are defined in this PROC step.

```
   item 'File' menu=f;
   item 'Edit' submenu=editmnu;
   item 'Scroll' menu=s;
   item 'Help' menu=h;
```

This group of statements defines the selections available under **File** on the menu bar. The first ITEM statement specifies **Goback** as the first selection under **File**. The value of the SELECTION= option corresponds to the subsequent SELECTION statement, which specifies END as the command that is issued for that selection. The second ITEM statement specifies that the SAVE command is issued for that selection.

```
   menu f;
      item 'Goback' selection=g;
      item 'Save';
      selection g 'end';
```

The SUBMENU statement associates a predefined submenu that is located in the SAS file SASHELP.CORE.EDIT with the **Edit** item on the menu bar. The name of this SUBMENU statement is **EDITMNU**, which corresponds with the name in the SUBMENU= action-option in the ITEM statement for the **Edit** item.

```
   submenu editmnu sashelp.core.edit;
```

This group of statements defines the selections available under **Scroll** on the menu bar.

```
   menu s;
      item 'Next Obs' selection=n;
      item 'Prev Obs' selection=p;
      item 'Top';
      item 'Bottom';
      selection n 'forward';
      selection p 'backward';
```

This group of statements defines the selections available under **Help** on the menu bar. The SETHELP command specifies a HELP entry that contains user-written information for this FSEDIT application. The semicolon that appears after the HELP entry name allows the HELP command to be included in the string. The HELP command invokes the HELP entry.

```
   menu h;
      item 'Keys';
```

```
          item 'About this application' selection=hlp;
          selection hlp 'sethelp user.menucat.staffhlp.help;help';
quit;
```

Associating a Menu Bar with an FSEDIT Session

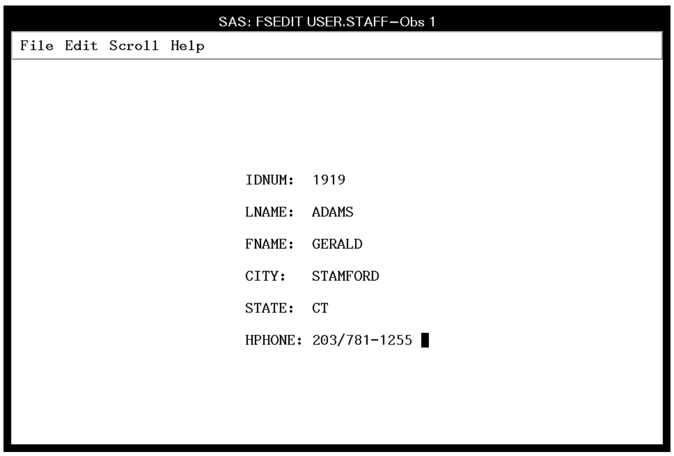
The following SETPMENU command associates the customized menu bar with the FSEDIT window.

```
setpmenu proclib.menucat.project.pmenu;pmenu on
```

You can also specify the menu bar on the command line in the FSEDIT session or with a CALL EXECCMD in Screen Control Language.

See "Associating a Menu Bar with an FSEDIT Session" on page 805 for other methods of associating the customized menu bar with the FSEDIT window.

The FSEDIT window shows the menu bar.



Example 2: Collecting User Input in a Dialog Box

Procedure features:

DIALOG statement

TEXT statement option:

LEN=

This example adds a dialog box to the menus created in Example 1 on page 796. The dialog box enables the user to use a WHERE clause to subset the SAS data set.

Tasks include

- □ collecting user input in a dialog box
- □ creating customized menus for an FSEDIT application.

Program

```
libname proclib 'SAS-data-library';
```

CATALOG= specifies PROCLIB.MENUCAT as the catalog that stores the menus.

```
proc pmenu c=proclib.menucat;
```

The MENU statement specifies PROJECT as the name of the catalog entry. The menus are stored in the catalog entry PROCLIB.MENUCAT.PROJECT.PMENU.

```
   menu project;
```

The ITEM statements specify the items for the menu bar. The value of the MENU= option is used in a subsequent MENU statement.

```
      item 'File' menu=f;
      item 'Edit' menu=e;
      item 'Scroll' menu=s;
      item 'Subset' menu=sub;
      item 'Help' menu=h;
```

This group of statements defines the selections under **File** on the menu bar. The first ITEM statement specifies **Goback** as the first selection under **File**. The value of the SELECTION= option corresponds to the subsequent SELECTION statement, which specifies END as the command that is issued for that selection. The second ITEM statement specifies that the SAVE command is issued for that selection.

```
      menu f;
         item 'Goback' selection=g;
         item 'Save';
         selection g 'end';
```

This group of statements defines the selections available under **Edit** on the menu bar.

```
      menu e;
         item 'Cancel';
         item 'Add';
```

This group of statements defines the selections available under **Scroll** on the menu bar.

```
      menu s;
         item 'Next Obs' selection=n;
         item 'Prev Obs' selection=p;
         item 'Top';
         item 'Bottom';
         selection n 'forward';
         selection p 'backward';
```

This group of statements defines the selections available under **Subset** on the menu bar. The value **d1** in the DIALOG= option is used in the subsequent DIALOG statement.

```
      menu sub;
         item 'Where' dialog=d1;
         item 'Where Clear';
```

This group of statements defines the selections available under **Help** on the menu bar. The SETHELP command specifies a HELP entry that contains user-written information for this FSEDIT application. The semicolon allows for the HELP command to be included in the string. The HELP command invokes the HELP entry.

```
      menu h;
         item 'Keys';
         item 'About this application' selection=hlp;
         selection hlp 'sethelp proclib.menucat.staffhlp.help;help';
```

The DIALOG statement builds a WHERE command. The arguments for the WHERE command are provided by user input into the text entry fields described by the three TEXT statements. The @1 notation is a placeholder for user input in the text field. The TEXT statements specify the text in the dialog box and the length of the input field.

```
      dialog d1 'where @1';
         text #2 @3 'Enter a valid WHERE clause or UNDO';
         text #4 @3 'WHERE ';
         text #4 @10 len=40;
quit;
```

Associating a Menu Bar with an FSEDIT Window

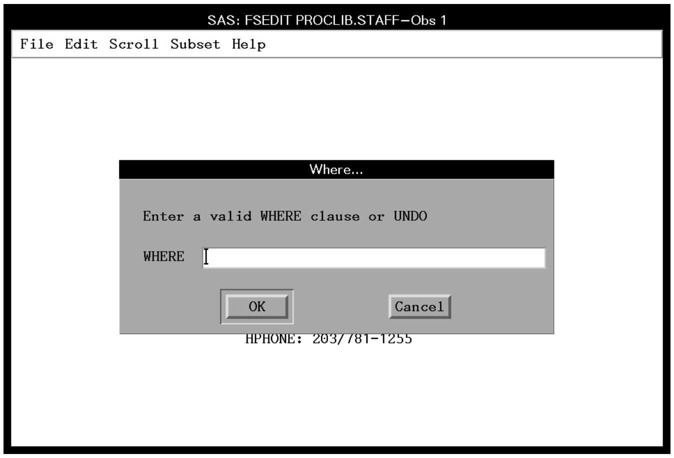
The following SETPMENU command associates the customized menu bar with the FSEDIT window.

```
setpmenu proclib.menucat.project.pmenu;pmenu on
```

You can also specify the menu bar on the command line in the FSEDIT session or with a CALL EXECCMD command in SAS Screen Control Language (SCL). Refer to *SAS Component Language: Reference* for complete documentation on SCL.

See “Associating a Menu Bar with an FSEDIT Session” on page 805 for other methods of associating the customized menu bar with the FSEDIT window.

This dialog box appears when the user chooses **Subset** and then **Where**.



Example 3: Creating a Dialog Box to Search Multiple Variables

Procedure features:

DIALOG statement
- SAS macro invocation

ITEM statement
- DIALOG= option

RADIOBOX statement option:
- DEFAULT=

RBUTTON statement option:
- SUBSTITUTE=

Other features: SAS macro invocation

This example shows how to modify the menu bar in an FSEDIT session to enable a search for one value across multiple variables. The example creates customized menus to use in an FSEDIT session. The menu structure is the same as in the preceding example, except for the WHERE dialog box.

Once selected, the menu item invokes a macro. The user input becomes values for macro parameters. The macro generates a WHERE command that expands to include all the variables needed for the search.

Tasks include

- associating customized menus with an FSEDIT session
- searching multiple variables with a WHERE clause
- extending PROC PMENU functionality with a SAS macro.

Program

```
libname proclib 'SAS-data-library';
```

CATALOG= specifies PROCLIB.MENUCAT as the catalog that stores the PMENU entry.

```
proc pmenu catalog=proclib.menucat;
```

The MENU statement specifies STAFF as the name of the catalog entry. The menus are stored in the catalog entry PROCLIB.MENUCAT.PROJECT.PMENU.

```
   menu project;
```

The ITEM statements specify the items for the menu bar. The value of the MENU= option is used in a subsequent MENU statement.

```
      item 'File' menu=f;
      item 'Edit' menu=e;
      item 'Scroll' menu=s;
      item 'Subset' menu=sub;
      item 'Help' menu=h;
```

This group of statements defines the selections under **File** on the menu bar. The first ITEM statement specifies **Goback** as the first selection under **File**. The value of the SELECTION= option corresponds to the subsequent SELECTION statement, which specifies END as the command that is issued for that selection. The second ITEM statement specifies that the SAVE command is issued for that selection.

```
      menu f;
         item 'Goback' selection=g;
         item 'Save';
         selection g 'end';
```

The ITEM statements define the selections under **Edit** on the menu bar.

```
      menu e;
         item 'Cancel';
         item 'Add';
```

This group of statements defines the selections under **Scroll** on the menu bar. If the quoted string in the ITEM statement is not a valid command, the SELECTION= option corresponds to a subsequent SELECTION statement, which specifies a valid command.

```
      menu s;
         item 'Next Obs' selection=n;
         item 'Prev Obs' selection=p;
         item 'Top';
         item 'Bottom';
         selection n 'forward';
         selection p 'backward';
```

This group of statements defines the selections under **Subset** on the menu bar. The DIALOG= option names a dialog box that is defined in a subsequent DIALOG statement.

```
menu sub;
   item 'Where' dialog=d1;
   item 'Where Clear';
```

This group of statements defines the selections under **Help** on the menu bar. The SETHELP command specifies a HELP entry that contains user-written information for this FSEDIT application. The semicolon that appears after the HELP entry name allows the HELP command to be included in the string. The HELP command invokes the HELP entry.

```
menu h;
   item 'Keys';
   item 'About this application' selection=hlp;
   selection hlp 'sethelp proclib.menucat.staffhlp.help;help';
```

WBUILD is a SAS macro. The double percent sign that precedes WBUILD is necessary to prevent PROC PMENU from expecting a field number to follow. The field numbers %1, %2, and %3 equate to the values specified by the user with the radio boxes. The field number @1 equates to the search value that the user enters. See "How the WBUILD Macro Works" on page 806.

```
dialog d1 '%%wbuild(%1,%2,@1,%3)';
```

The TEXT statement specifies text for the dialog box that appears on line 1 and begins in column 1. The RADIOBOX statement specifies that a radio box will appear in the dialog box. DEFAULT= specifies that the first radio button (**Northeast**) will be selected by default. The RBUTTON statements specify the mutually exclusive choices for the radio buttons: **Northeast**, **Northwest**, **Southeast**, or **Southwest**. SUBSTITUTE= gives the value that is substituted for the %1 in the DIALOG statement above if that radio button is selected.

```
text #1 @1 'Choose a region:';
radiobox default=1;
   rbutton #3 @5 'Northeast' substitute='NE';
   rbutton #4 @5 'Northwest' substitute='NW';
   rbutton #5 @5 'Southeast' substitute='SE';
   rbutton #6 @5 'Southwest' substitute='SW';
```

The TEXT statement specifies text for the dialog box that appears on line 8 (#8) and begins in column 1 (@1). The RADIOBOX statement specifies that a radio box will appear in the dialog box. DEFAULT= specifies that the first radio button (**Pollutant A**) will be selected by default. The RBUTTON statements specify the mutually exclusive choices for the radio buttons: **Pollutant A** or **Pollutant B**. SUBSTITUTE= gives the value that is substituted for the %2 in the preceding DIALOG statement if that radio button is selected.

```
text #8 @1 'Choose a contaminant:';
radiobox default=1;
   rbutton #10 @5  'Pollutant A' substitute='pol_a,2';
```

```
                rbutton #11 @5  'Pollutant B' substitute='pol_b,4';
```

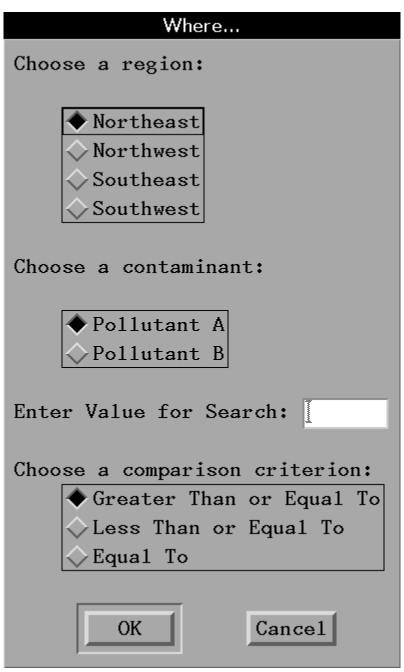
The first TEXT statement specifies text for the dialog box that appears on line 13 and begins in column 1. The second TEXT statement specifies an input field that is 6 bytes long that appears on line 13 and begins in column 25. The value that the user enters in the field is substituted for the @1 in the preceding DIALOG statement.

```
             text #13 @1 'Enter Value for Search:';
             text #13 @25 len=6;
```

The TEXT statement specifies text for the dialog box that appears on line 15 and begins in column 1. The RADIOBOX statement specifies that a radio box will appear in the dialog box. DEFAULT= specifies that the first radio button (**Greater Than or Equal To**) will be selected by default. The RBUTTON statements specify the mutually exclusive choices for the radio buttons. SUBSTITUTE= gives the value that is substituted for the %3 in the preceding DIALOG statement if that radio button is selected.

```
             text #15 @1 'Choose a comparison criterion:';
             radiobox default=1;
                rbutton #16 @5 'Greater Than or Equal To'
                                substitute='GE';
                rbutton #17 @5 'Less Than or Equal To'
                                substitute='LE';
                rbutton #18 @5 'Equal To' substitute='EQ';
quit;
```

This dialog box appears when the user selects **Subset** and then **Where**.



Associating a Menu Bar with an FSEDIT Session

The SAS data set PROCLIB.LAKES has data about several lakes. Two pollutants, pollutant A and pollutant B, were tested at each lake. Tests were conducted for pollutant A twice at each lake, and the results are recorded in the variables POL_A1 and POL_A2. Tests were conducted for pollutant B four times at each lake, and the results are recorded in the variables POL_B1 - POL_B4. Each lake is located in one of four regions. The following output lists the contents of PROCLIB.LAKES:

Output 31.1

```
                               PROCLIB.LAKES                               1

region    lake         pol_a1    pol_a2    pol_b1    pol_b2    pol_b3    pol_b4

  NE      Carr          0.24      0.99      0.95      0.36      0.44      0.67
  NE      Duraleigh     0.34      0.01      0.48      0.58      0.12      0.56
  NE      Charlie       0.40      0.48      0.29      0.56      0.52      0.95
  NE      Farmer        0.60      0.65      0.25      0.20      0.30      0.64
  NW      Canyon        0.63      0.44      0.20      0.98      0.19      0.01
  NW      Morris        0.85      0.95      0.80      0.67      0.32      0.81
  NW      Golf          0.69      0.37      0.08      0.72      0.71      0.32
  NW      Falls         0.01      0.02      0.59      0.58      0.67      0.02
  SE      Pleasant      0.16      0.96      0.71      0.35      0.35      0.48
  SE      Juliette      0.82      0.35      0.09      0.03      0.59      0.90
  SE      Massey        1.01      0.77      0.45      0.32      0.55      0.66
  SE      Delta         0.84      1.05      0.90      0.09      0.64      0.03
  SW      Alumni        0.45      0.32      0.45      0.44      0.55      0.12
  SW      New Dam       0.80      0.70      0.31      0.98      1.00      0.22
  SW      Border        0.51      0.04      0.55      0.35      0.45      0.78
  SW      Red           0.22      0.09      0.02      0.10      0.32      0.01
```

A DATA step on page 1646 creates PROCLIB.LAKES.

The following statements initiate a PROC FSEDIT session for PROCLIB.LAKES:

```
proc fsedit data=proclib.lakes screen=proclib.lakes;
run;
```

To associate the customized menu bar menu with the FSEDIT session, do any one of the following:

- enter a SETPMENU command on the command line. The command for this example is

  ```
  setpmenu proclib.menucat.project.pmenu
  ```

  Turn on the menus by entering PMENU ON on the command line.

- enter the SETPMENU command in a Command window.
- include an SCL program with the FSEDIT session that uses the customized menus and turns on the menus, for example:

  ```
  fseinit:
    call execcmd('setpmenu proclib.menucat.project.pmenu;
                 pmenu on;');
  return;
  init:
  return;
  main:
  return;
  term:
  return;
  ```

How the WBUILD Macro Works

Consider how you would learn whether any of the lakes in the Southwest region tested for a value of .50 or greater for pollutant A. Without the customized menu item, you would issue the following WHERE command in the FSEDIT window:

```
where region="SW" and (pol_a1 ge .50 or pol_a2 ge .50);
```

Using the custom menu item, you would select **Southwest**, **Pollutant A**, enter .50 as the value, and choose **Greater Than or Equal To** as the comparison criterion. Two lakes, **New Dam** and **Border**, meet the criteria.

The WBUILD macro uses the four pieces of information from the dialog box to generate a WHERE command:

- One of the values for region, either **NE**, **NW**, **SE**, or **SW**, becomes the value of the macro parameter REGION.
- Either **pol_a,2** or **pol_b,4** become the values of the PREFIX and NUMVAR macro parameters. The comma is part of the value that is passed to the WBUILD macro and serves to delimit the two parameters, PREFIX and NUMVAR.
- The value that the user enters for the search becomes the value of the macro parameter VALUE.
- The operator that the user chooses becomes the value of the macro parameter OPERATOR.

To see how the macro works, again consider the following example, in which you want to know if any of the lakes in the southwest tested for a value of .50 or greater for pollutant A. The values of the macro parameters would be

REGION	**SW**
PREFIX	**pol_a**
NUMVAR	2
VALUE	.50
OPERATOR	GE

The first %IF statement checks to make sure that the user entered a value. If a value has been entered, the macro begins to generate the WHERE command. First, the macro creates the beginning of the WHERE command:

```
where region="SW" and (
```

Next, the %DO loop executes. For pollutant A, it executes twice because NUMVAR=2. In the macro definition, the period in **&prefix.&i** concatenates **pol_a** with **1** and with **2**. At each iteration of the loop, the macro resolves PREFIX, OPERATOR, and VALUE, and it generates a part of the WHERE command. On the first iteration, it generates

```
pol_a1 GE .50
```

The %IF statement in the loop checks to see if the loop is working on its last iteration. If it is not, the macro makes a compound WHERE command by putting an **OR** between the individual clauses. The next part of the WHERE command becomes

```
OR pol_a2 GE .50
```

The loop ends after two executions for pollutant A, and the macro generates the last of the WHERE command:

)

Results from the macro are placed on the command line. The following code is the definition of the WBUILD macro. The underlined code shows the parts of the WHERE command that are text strings that the macro does not resolve:

```
%macro wbuild(region,prefix,numvar,value,operator);
       /* check to see if value is present */
    %if &value ne %then %do;
       where region="&region" AND (
               /* If the values are character,     */
               /* enclose &value in double quotes. */
         %do i=1 %to &numvar;
             &prefix.&i &operator &value
                /* if not on last variable,  */
                /* generate 'OR'             */
            %if &i ne &numvar %then %do;
                 OR
            %end;
         %end;
          )
    %end;

%mend wbuild;
```

Example 4: Creating Menus for a DATA Step Window Application

Procedure features:
DIALOG statement
SELECTION statement
Other features: FILENAME statement

This example defines an application that enables the user to enter human resources data for various departments and to request reports from the data sets created by the data entry.

The first part of the example describes the PROC PMENU step that creates the menus. The subsequent sections describe how to use the menus in a DATA step window application.

Tasks include

- associating customized menus with a DATA step window
- creating menus for a DATA step window
- submitting SAS code from a menu selection
- creating a pull-down menu selection that calls a dialog box.

Program

The LIBNAME statement defines the SAS data library in which the PMENU entries are stored. The FILENAME statements define the external files in which the programs to create the windows are stored.

```
libname proclib 'SAS-data-library';
filename de     'external-file';
filename prt    'external-file';
```

CATALOG= specifies PROCLIB.MENUS as the catalog that stores menus.

```
proc pmenu catalog=proclib.menus;
```

The MENU statement specifies SELECT as the name of the catalog entry. The menus are stored in the catalog entry PROCLIB.MENUS.SELECT.PMENU.

```
menu select;
```

The ITEM statements specify the three items on the menu bar. The value of the MENU= option is used in a subsequent MENU statement.

```
item 'File' menu=f;
item 'Data_Entry' menu=deptsde;
item 'Print_Report' menu=deptsprt;
```

This group of statements defines the selections under **File**. The value of the SELECTION= option is used in a subsequent SELECTION statement.

```
menu f;
   item 'End this window' selection=endwdw;
   item 'End this SAS session' selection=endsas;
   selection endwdw 'end';
   selection endsas 'bye';
```

This group of statements defines the selections under **Data_Entry** on the menu bar. The ITEM statements specify that **For Dept01** and **For Dept02** appear under **Data_Entry**. The value of the SELECTION= option equates to a subsequent SELECTION statement, which contains the string of commands that are actually submitted. The value of the DIALOG= option equates to a subsequent DIALOG statement, which describes the dialog box that appears when this item is selected.

```
menu deptsde;
   item 'For Dept01' selection=de1;
   item 'For Dept02' selection=de2;
   item 'Other Departments' dialog=deother;
```

The commands in single quotes are submitted when the user selects **For Dept01** or **For Dept02**. The END command ends the current window and returns to the PROGRAM EDITOR window so that further commands can be submitted. The INCLUDE command includes the SAS statements that create the data entry window. The CHANGE command modifies the DATA statement in the included program so that it creates the correct data set. See "Using a Data Entry Program" on page 811. The SUBMIT command submits the DATA step program.

```
selection de1 'end;pgm;include de;change xx 01;submit';
selection de2 'end;pgm;include de;change xx 02;submit';
```

The DIALOG statement defines the dialog box that appears when the user selects **Other Departments**. The DIALOG statement modifies the command string so that the name of the department that is entered by the user is used to change **deptxx** in the SAS program that is included. See "Using a Data Entry Program" on page 811. The first two TEXT statements specify text that appears in the dialog box. The third TEXT statement specifies an input field. The name that is entered in this field is substituted for the @1 in the DIALOG statement.

```
dialog deother 'end;pgm;include de;c deptxx @1;submit';
   text #1 @1 'Enter department name';
   text #2 @3 'in the form DEPT99:';
   text #2 @25 len=7;
```

This group of statements defines the choices under the **Print_Report** item. These ITEM statements specify that **For Dept01** and **For Dept02** appear in the pull-down menu. The value of the SELECTION= option equates to a subsequent SELECTION statement, which contains the string of commands that are actually submitted.

```
menu deptsprt;
   item 'For Dept01' selection=prt1;
   item 'For Dept02' selection=prt2;
   item 'Other Departments' dialog=prother;
```

The commands in single quotes are submitted when the user selects **For Dept01** or **For Dept02**. The END command ends the current window and returns to the PROGRAM EDITOR window so that further commands can be submitted. The INCLUDE command includes the SAS statements that print the report. See "Printing a Program" on page 812. The CHANGE command modifies the PROC PRINT step in the included program so that it prints the correct data set. The SUBMIT command submits the PROC PRINT program.

```
   selection prt1
         'end;pgm;include prt;change xx 01 all;submit';
   selection prt2
         'end;pgm;include prt;change xx 02 all;submit';
```

The DIALOG statement defines the dialog box that appears when the user selects **Other Departments**. The DIALOG statement modifies the command string so that the name of the department that is entered by the user is used to change **deptxx** in the SAS program that is included. See "Printing a Program" on page 812. The first two TEXT statements specify text that appears in the dialog box. The third TEXT statement specifies an input field. The name entered in this field is substituted for the @1 in the DIALOG statement.

```
dialog prother 'end;pgm;include prt;c deptxx @1 all;submit';
   text #1 @1 'Enter department name';
   text #2 @3 'in the form DEPT99:';
   text #2 @25 len=7;
```

The RUN statement ends this RUN group.

```
run;
```

The MENU statement specifies ENTRDATA as the name of the catalog entry that this RUN group is creating. **File** is the only item in the menu bar. The selections available are **End this window** and **End this SAS session.**

```
      menu entrdata;
         item 'File' menu=f;
         menu f;
            item 'End this window' selection=endwdw;
            item 'End this SAS session' selection=endsas;
            selection endwdw 'end';
            selection endsas 'bye';
   run;
quit;
```

Associating a Menu with a Window

The first group of statements defines the primary window for the application. These statements are stored in the file that is referenced by the HRWDW fileref:

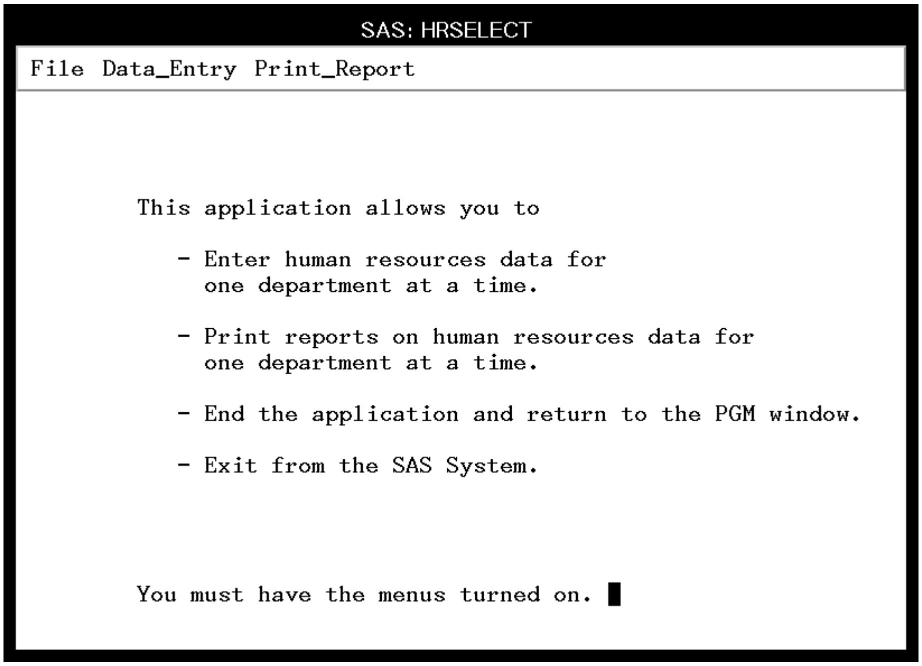
The WINDOW statement creates the HRSELECT window. MENU= associates the PROCLIB.MENUS.SELECT.PMENU entry with this window.

```
data _null_;
   window hrselect menu=proclib.menus.select
   #4  @10 'This application allows you to'
   #6  @13 '- Enter human resources data for'
   #7  @15 'one department at a time.'
   #9  @13 '- Print reports on human resources data for'
   #10 @15 'one department at a time.'
   #12 @13 '- End the application and return to the PGM window.'
   #14 @13 '- Exit from the SAS System.'
   #19 @10 'You must have the menus turned on.';
```

The DISPLAY statement displays the window HRSELECT.

```
   display hrselect;
run;
```

Primary window, HRSELECT.



Using a Data Entry Program

When the user selects **Data_Entry** from the menu bar in the HRSELECT window, a pull-down menu is displayed. When the user selects one of the listed departments or chooses to enter a different department, the following statements are invoked. These statements are stored in the file that is referenced by the DE fileref.

The WINDOW statement creates the HRDATA window. MENU= associates the PROCLIB.MENUS.ENTRDATA.PMENU entry with the window.

```
data proclib.deptxx;
    window hrdata menu=proclib.menus.entrdata
    #5  @10 'Employee Number'
    #8  @10 'Salary'
    #11 @10 'Employee Name'
    #5  @31 empno $4.
    #8  @31 salary 10.
    #11 @31 name $30.
    #19 @10 'Press ENTER to add the observation to the data set.';
```

The DISPLAY statement displays the HRDATA window.

```
    display hrdata;
run;
```

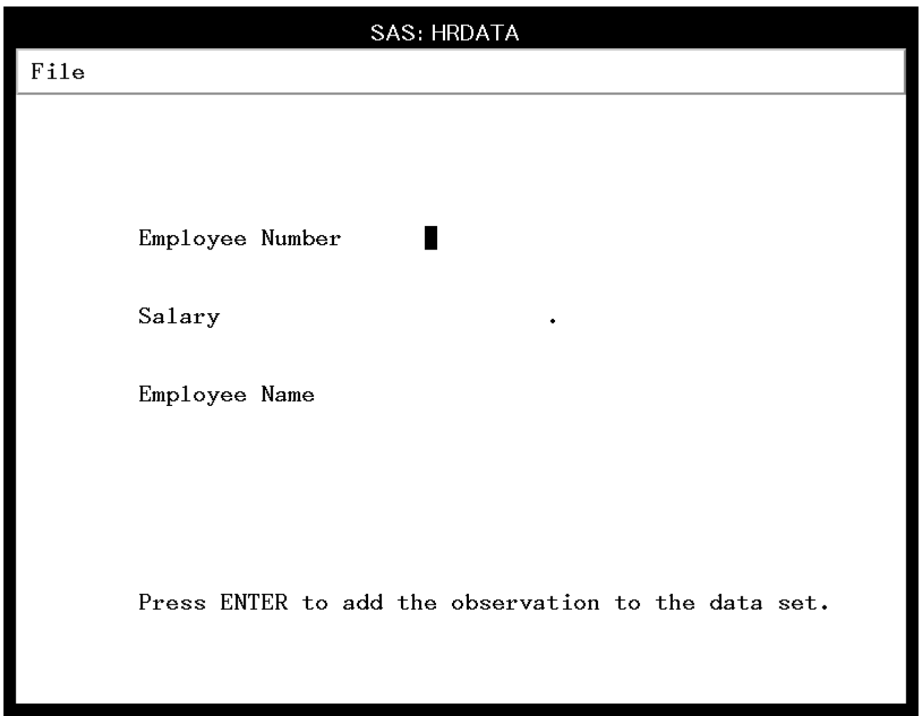
The %INCLUDE statement recalls the statements in the file HRWDW. The statements in HRWDW redisplay the primary window. See theHRSELECT on page 810 window.

```
filename hrwdw 'external-file';
%include hrwdw;
run;
```

The SELECTION and DIALOG statements in the PROC PMENU step modify the DATA statement in this program so that the correct department name is used when the data set is created. That is, if the user selects **Other Departments** and enters **DEPT05**, the DATA statement is changed by the command string on the DIALOG statement to

```
data proclib.dept05;
```

Data entry window, HRDATA.



Printing a Program

When the user selects **Print_Report** from the menu bar, a pull-down menu is displayed. When the user selects one of the listed departments or chooses to enter a different department, the following statements are invoked. These statements are stored in the external file referenced by the PRT fileref.

PROC PRINTTO routes the output to an external file.

```
proc printto file='external-file' new;
run;
```

The **xx**'s are changed to the appropriate department number by the CHANGE command in the SELECTION or DIALOG statement in the PROC PMENU step. PROC PRINT prints that data set.

```
libname proclib 'SAS-data-library';

proc print data=proclib.deptxx;
   title 'Information for deptxx';
run;
```

This PROC PRINTTO steps restores the default output destination. See Chapter 33, "The PRINTTO Procedure," on page 879 for documentation on PROC PRINTTO.

```
proc printto;
run;
```

The %INCLUDE statement recalls the statements in the file HRWDW. The statements in HRWDW redisplay the primary window.

```
filename hrwdw 'external-file';
%include hrwdw;
run;
```

Example 5: Associating Menus with a FRAME Application

Procedure features:
ITEM statement
MENU statement

Other features: SAS/AF software

This example creates menus for a FRAME entry and gives the steps necessary to associate the menus with a FRAME entry from SAS/AF software.

Program

```
libname proclib 'SAS-data-library';
```

CATALOG= specifies PROCLIB.MENUCAT as the catalog that stores the menus.

```
proc pmenu catalog=proclib.menucat;
```

The MENU statement specifies FRAME as the name of the catalog entry. The menus are stored in the catalog entry PROCLIB.MENUS.FRAME.PMENU.

```
   menu frame;
```

The ITEM statements specify the items in the menu bar. The value of MENU= corresponds to a subsequent MENU statement.

```
      item 'File' menu=f;
      item 'Help' menu=h;
```

The MENU statement equates to the MENU= option in a preceding ITEM statement. The ITEM statements specify the selections that are available under **File** in the menu bar.

```
      menu f;
         item 'Cancel';
         item 'End';
```

The MENU statement equates to the MENU= option in a preceding ITEM statement. The ITEM statements specify the selections that are available under **Help** on the menu bar. The value of the SELECTION= option equates to a subsequent SELECTION statement.

```
      menu h;
         item 'About the application' selection=a;
         item 'About the keys'   selection=k;
```

The SETHELP command specifies a HELP entry that contains user-written information for this application. The semicolon that appears after the HELP entry name allows the HELP command to be included in the string. The HELP command invokes the HELP entry.

```
            selection a 'sethelp proclib.menucat.app.help;help';
            selection k 'sethelp proclib.menucat.keys.help;help';
run;
quit;
```

Steps to Associate Menus with a FRAME

1 In the BUILD environment for the FRAME entry, from the menu bar, select

   View ▶ Properties Window

2 In the Properties window, select the **Value** field for the *pmenuEntry* Attribute Name. The Select An Entry window opens.
3 In the Select An Entry window, enter the name of the catalog entry that is specified in the PROC PMENU step that creates the menus.
4 Test the FRAME as follows from the menu bar of the FRAME:

   Build ▶ Test

Notice that the menus are now associated with the FRAME.

Refer to *Getting Started with the FRAME Entry: Developing Object-Oriented Applications* for more information on SAS programming with FRAME entries.