金融时间序列分析讲义

李东风

2024 年春季学期

目录

关于本网页 **9**

内容 . . . 9

参考书 . . . 9

下载 . . . 10

I R 软件与金融数据 **11**

1 金融数据分析中的 R 软件介绍 **13**

1.1 本课程的软件需求 . . . 13

1.2 基本 R 使用 . . . 14

1.3 生成时间序列数据 . . . 17

1.4 ts 类型 . . . 19

1.5 zoo 类型 . . . 23

1.6 xts 类型 . . . 38

1.7 tsibble 类型 . . . 44

1.8 quantmod 包的功能 . . . 44

1.9 用 BatchGetSymbols 包下载金融数据 . . . 57

1.10 tseries 包的功能 . . . 59

1.11 R 软件的其它时间序列类型和功能 . . . 59

2 金融数据及其特征 **61**

2.1 资产收益率 . . . 61

2.2 债券收益和价格 . . . 74

2.3 隐含波动率 . . . 80

2.4 收益率分布特性的探索性分析 . . . 82

2.5 收益率的分布特性 . . . 92

2.6 金融数据的图形 . . . 98

2.7 金融数据常用分布 . . . 114

II 一元线性时间序列模型 **121**

3 线性时间序列模型 **123**

3.1 介绍 . . . 123

3.2 平稳性 . . . 131
3.3 相关系数和自相关函数 . . . 132
3.4 线性时间序列 . . . 143
3.5 附录：补充知识 . . . 144

4 自回归模型 147
4.1 自回归模型的概念 . . . 147
4.2 滞后算子 . . . 148
4.3 AR(1) 模型的性质 . . . 148
4.4 AR(1) 模型的自相关函数 . . . 149
4.5 AR(2) 模型的性质 . . . 151
4.6 AR(p) 模型的性质 . . . 162
4.7 偏自相关函数 . . . 162
4.8 信息准则 . . . 167
4.9 AR 模型参数估计方法 . . . 168
4.10 AR 模型检验 . . . 172
4.11 AR 模型拟合优度指标 . . . 175
4.12 用估计的 AR 模型进行预测 . . . 176

5 移动平均模型 181
5.1 移动平均模型的概念 . . . 181
5.2 移动平均模型的性质 . . . 182
5.3 移动平均模型定阶 . . . 183
5.4 移动平均模型的估计 . . . 184
5.5 移动平均模型的预测 . . . 187
5.6 AR 和 MA 的小结 . . . 189

6 ARMA 模型 191
6.1 ARMA 模型的概念 . . . 191
6.2 ARMA 模型的性质 . . . 192
6.3 一般 ARMA 模型 . . . 193
6.4 ARMA 模型辨识 . . . 193
6.5 ARMA 模型预测 . . . 200
6.6 ARMA 模型的三种表示 . . . 201
6.7 附录 . . . 202

7 单位根过程 203
7.1 随机游动 . . . 203
7.2 带漂移的随机游动 . . . 204
7.3 固定趋势模型 . . . 207
7.4 ARIMA 模型 . . . 208
7.5 单位根检验 . . . 209

8 指数平滑 225
8.1 简单指数平滑 . . . 225

8.2 时间序列成分分解 231

9 季节模型 241
9.1 季节差分 243
9.2 乘性季节模型 247
9.3 季节哑变量 253

10 带时间序列误差的回归模型 259
10.1 方法示例 259
10.2 arima 函数说明 273

11 长记忆模型 277
11.1 长记忆模型介绍 277
11.2 长记忆模型性质 278
11.3 长记忆模型建模实例 279

12 模型比较和平均 283
12.1 样本内比较 283
12.2 样本外比较 285
12.3 模型平均 291
12.4 附录：backtest 函数 291

13 线性时间序列案例学习—汽油价格 311
13.1 数据读入与探索性分析 311
13.2 AR(5) 模型 322
13.3 ARMA(1,3) 模型 327
13.4 固定线性趋势模型 331
13.5 引入石油价格解释变量的模型 333
13.6 使用滞后石油价格解释变量的模型 338
13.7 样本外预测 346

14 线性时间序列案例学习—全球温度异常值 351
14.1 数据读入与探索性分析 351
14.2 单位根非平稳模型 354
14.3 线性固定趋势模型 368
14.4 二次固定趋势模型 376
14.5 模型比较 383
14.6 长期预测 389
14.7 讨论 392

15 线性时间序列案例学习—美国月失业率 397
15.1 单变量时间序列模型 397
15.2 一个替代模型 406
15.3 模型比较 410
15.4 使用首次申请失业救济金人数 413
15.5 模型再比较 428

III 资产波动率模型 435

16 资产波动率模型特征 437
16.1 波动率的特征 . . . 437
16.2 波动率模型的结构 . . . 438
16.3 波动率模型的建立 . . . 446
16.4 ARCH 效应的检验 . . . 446

17 ARCH 模型 455
17.1 ARCH 模型公式 . . . 455
17.2 ARCH 模型的性质 . . . 456
17.3 ARCH 模型的优缺点 . . . 459
17.4 ARCH 模型的建模步骤 . . . 459
17.5 ARCH 模型建模实例 . . . 465
17.6 补充推导 . . . 485

18 GARCH 模型 487
18.1 模型方程 . . . 487
18.2 与 ARMA 模型比较 . . . 487
18.3 GARCH 模型的性质 . . . 488
18.4 预测 . . . 489
18.5 模型估计 . . . 489
18.6 两步估计法 . . . 510
18.7 IGARCH 模型 . . . 512
18.8 GARCH-M 模型 . . . 514

19 改进的 GARCH 模型 525
19.1 EGARCH 模型 . . . 525
19.2 GJR-GARCH 模型 . . . 539
19.3 APARCH 模型 . . . 550

20 随机波动率模型 559
20.1 随机波动率模型 . . . 559
20.2 长记忆随机波动率模型 . . . 559

21 其它的波动率计算方法 563
21.1 利用日频计算月频波动率 . . . 563
21.2 使用 OHLC 数据 . . . 571
21.3 附录：用日数据估计月波动率的 R 函数 . . . 582

22 波动率模型的应用 585
22.1 GARCH 波动率期限结构 . . . 585
22.2 期权定价和对冲 . . . 600
22.3 随时间变化的协方差和贝塔值 . . . 602
22.4 最小方差投资组合 . . . 615
22.5 预测 . . . 623

IV 多元时间序列模型 645

23 多元时间序列基本概念 647
23.1 弱平稳与互相关矩阵 . . . 647
23.2 样本互相关阵 . . . 649
23.3 多元混成检验 . . . 666
23.4 附录：用到的源程序代码和数据 . . . 669

24 向量自回归模型 673
24.1 VAR(1) 模型 . . . 673
24.2 模型结构和格兰杰因果性 . . . 673
24.3 VAR 的简化形式和结构形式 . . . 675
24.4 VAR(1) 模型的平稳性条件和矩 . . . 678
24.5 VAR(1) 模型的边缘模型 . . . 680
24.6 VAR(p) 模型 . . . 681
24.7 估计和定阶 . . . 682
24.8 模型检验 . . . 692
24.9 模型简化 . . . 694
24.10基于 VAR 模型的格兰杰因果性检验 . . . 705
24.11预测 . . . 707
24.12脉冲响应函数 . . . 709
24.13`GrangerTest()` 函数 . . . 714

25 协整分析与向量误差修正模型 719
25.1 虚假回归问题 . . . 719
25.2 协整分析概念 . . . 730
25.3 Engle 和 Granger 两阶段法 . . . 730
25.4 VARMA 模型 . . . 737
25.5 误差修正模型与协整 . . . 739
25.6 附录 A：线性模型估计相合性 . . . 749
25.7 附录 B：用到的数据 . . . 749

26 格兰格因果性 753
26.1 介绍 . . . 753
26.2 格兰格因果性的定义 . . . 754
26.3 两变量 VAR 情形 . . . 758
26.4 两变量 VMA 情形 . . . 759
26.5 单变量模型表示 . . . 759
26.6 因果性检验 . . . 760
26.7 多元情形下的因果性检验 . . . 774

V 状态空间模型 777

27 局部水平模型 779

27.1 模型 . . . 779
27.2 局部水平模型与 ARIMA 模型的关系 . . . 780
27.3 滤波、平滑和预报 . . . 782
27.4 卡尔曼滤波 . . . 784
27.5 一步预报误差 . . . 788
27.6 状态的一步预报误差 . . . 789
27.7 状态平滑 . . . 790
27.8 扰动的平滑 . . . 794
27.9 缺失值处理和预测 . . . 796
27.10初值分布参数选取 . . . 796
27.11模型参数估计 . . . 797

28 状态空间模型 **799**
28.1 状态空间模型的公式 . . . 799
28.2 有关 R 扩展包 . . . 801
28.3 一元结构时间序列模型 . . . 807
28.4 ARMA 和 ARIMA 模型 . . . 835
28.5 回归模型 . . . 842

A 测试 **927**

关于本网页

本书为北京大学数学科学学院金融数学系金融数学应用硕士《金融时间序列分析》授课备课资料。课程采用 Ruey S. Tsay 的《金融数据分析导论：基于 R 语言》（An Introduction to Analysis of Financial Data with R）作为主要教材之一。

如果读者使用本书的网页版本，用如下数学公式测试浏览器中数学公式显示是否正常：

$$\text{定积分} = \int_a^b f(x)\,dx$$

如果显示不正常，可以在公式上右键单击，选择 “Math Settings–Math Renderer”，依次使用改成 “Common HTML”，“SVG” 等是否可以变成正常显示。PDF 版本不存在这样的问题。

内容

- R 软件的时间序列相关的功能介绍, Rmd 格式介绍;
- 收益率，债券，波动率，金融数据示例，收益率分布性质，金融数据可视化，统计分布复习;
- 线性时间序列：款平稳，自相关系数函数，ACF 的白噪声检验，AR, 偏相关系数，定阶与参数估计，预测，MA，ARMA, ARIMA，单位根过程，单位根检验，指数平滑方法，季节模型，回归模型的序列相关误差项，协整，长记忆模型，模型比较与模型平均线性时间序列的案例研究;
- 资产波动率，ARCH 效应，ARCH 模型，GARCH 模型，IGARCH 模型，GARCH-M 模型，EGARCH 模型, TGARCH 模型, APARCH 模型，非对称 GARCH 模型，随机波动率模型波动率模型案例研究;
- 多元时间序列的基础知识和 VAR 模型，协整和协整检验，格兰杰因果性;
- 状态空间模型介绍。

参考书

- (谢衷洁, 1990)
- (Tsay, 2013) 主要教材
- (Tsay, 2010)
- (Tsay, 2014) 多元部分教材
- (Shumway & Stoffer, 2019) 图书馆有电子版，入门级别教材，实例和 R 程序丰富
- (Shumway & Stoffer, 2017) 图书馆有电子版，涵盖内容丰富，有 R 实例

- (Durbin & Koopman, 2012) 状态空间模型的详细教材
- (吴喜之 & 刘苗, 2018)
- (何书元, 2003)
- (Brockwell & Davis, 1991) 时间序列分析理论基础的经典教材
- (Lutkepohl & Kratzig, 2004)
- (Cryer & Chan, 2008)
- (Christoffersen, 2003)
- (Gebhard Kirchgässner, 2013), 有格兰格因果、协整等内容
- (Pfaff, 2008)
- (Durbin & Koopman, 2012)

下载

数据下载：ftsdata.zip

本备课笔记的 PDF 版本可以通过点击页面最上方的 PDF 图标下载。

Part I

R 软件与金融数据

Chapter 1

金融数据分析中的 R 软件介绍

1.1 本课程的软件需求

课程采用 Ruey S. Tsay 的《金融数据分析导论：基于 R 语言》（An Introduction to Analysis of Financial Data with R）作为主要教材之一。课程讲述金融时间序列分析的各种模型，以及如何用 R 软件进行建模计算。

作为基础，学生需要重点掌握 R 软件如下功能：

- 基本数据类型，日期和日期时间类型，数据输入输出
- Rmd 格式文件运用
- 时间序列类型 zoo, xts, timeSeries 类型
- quantmod 包

关于 R 软件基本使用，详见李东风的 R 软件教程，第 1、2 章，第 9 章。

关于 Rmd 格式文件, 详见李东风的 R 软件教程，第 20–22 章。

本章需要安装的 R 扩展包：quantmod, lubridate, tidyverse.

注意：本章的例子需要用 `library()` 命令将 `tidyverse`, `lubridate`, `quantmod` 包调入才能运行。方法为

```
library(tidyverse) # Wickham 的数据整理的整套工具
library(lubridate) # 日期和日期时间数据处理
library(quantmod)  # 金融数据的整理与作图功能
```

R 软件中的时间序列类型有基本 R 提供的 ts 类型，zoo 包提供的 zoo 和 zooreg 类型，xts 包提供的 xts 类型，tseries 包提供的 irts 类型，Rmetrics 包提供的 timeSeries 类型，等等。

其它与金融时间序列有关的扩展包还有 fBasics, fGarch, fPortfolio, 等等。

参考:

- CRAN 时间序列任务视点
- CRAN 金融任务视点
- CRAN 计量经济任务视点

1.2 基本 R 使用

R 是一个数据分析、统计建模和作图的软件，其中包含一门计算机语言称为 R 语言，此语言与通常的 C、C++、Java 等编程语言相比，支持更多的数据类型，如向量、矩阵，并提供了多种统计和数学计算方法。

为了在金融时间序列计算中使用 R 语言，这里简单介绍一些最基本的使用方法。

R 软件是一个开源软件，可以免费地从其网站http://www.r-project.org提供的镜像网站下载安装。另外，RStudio 是一个 R 软件的集成开发环境，在该软件中可以更方便地使用 R 软件，虽然 RStudio 是商业软件，但非商业用户可以免费地使用。

R 可以在命令行运行，每次运行一条命令，结果显示在命令下方。也可以将多条 R 命令写成一个源程序文件，然后运行整个文件。在 RStudio 软件中也可以选中若干行程序，运行选中的部分。

1.2.1 四则运算

R 中用 `+ - * / ^` 分别表示加、减、乘、除、乘方。数值可以写成如 `123`, `-123`, `123.45`, `1.23E-5` 这样的形式，其中 `1.23E-5` 表示 1.23×10^{-5}。

1.2.2 字符串

R 中可以用双撇号" 或者单撇号'将文字内容包裹起来，称为字符串，如`"two.sided"`, `" 收益率"`。

1.2.3 向量

R 中最基本的单位是向量，单个数值是长度为 1 的向量。简单的向量可以用 `c(...)` 定义，如

```
x <- c(1.1, 1.02, 1.4, 0.9)
x
## [1] 1.10 1.02 1.40 0.90
```

长度相同的两个向量之间可以做四则运算，结果是对应的元素进行四则运算。单个元素可以和一个向量进行四则运算，结果是该元素与向量的每一个元素进行四则运算。如

```
x + 100
## [1] 101.10 101.02 101.40 100.90
```

1.2.4 矩阵

R 支持矩阵类型，矩阵是有 n 行、p 列的数值排列成的存储结果，在第 i 行、第 j 列有一个数值元素。矩阵 `A` 的第 (i, j) 元素表示为 `A[i,j]`，矩阵的第 j 列作为一个向量，可以表示为 `A[,j,drop=TRUE]`。`nrow(A)` 求 `A` 的行数，`ncol(A)` 求 `A` 的列数。

两个相同形状的矩阵可以进行加、减、乘、除四则运算，结果是对应的元素进行四则运算。单个元素可以与矩阵进行四则运算，结果是该元素与矩阵的每个元素进行四则运算。

两个向量 `x1` 和 `x2`，可以用 `cbind(x1, x2)` 合并为两列的矩阵。

1.2.5 数据框

R 数据框是和矩阵类似的存储结构，但是允许不同列的数据类型不同，比如，一列是日期，一列是评级（字符串），一列是收益率（数值）。

数据框的每一列都有列名，用 `names(d)` 或者 `colnames(d)` 访问。将数据框 `d` 的某一列取出作为一个向量，用 `d[[" 列名"]]` 的格式。

1.2.6 扩展包

R 的许多功能由不同的扩展包（package）提供。为了调用某个扩展包的功能，需要先用 `library()` 命令载入该扩展包，如

```
library(xts)
```

某些扩展包还允许用`包名:: 函数名 ()` 的格式直接调用其中的函数。

1.2.7 日期和日期时间

R 日期可以保存为 Date 类型，一般用整数保存，数值为从 1970-1-1 经过的天数。

R 中用一种叫做 POSIXct 和 POSIXlt 的特殊数据类型保存日期和时间，可以仅包含日期部分，也可以同时有日期和时间。技术上，POSIXct 把日期时间保存为从 1970 年 1 月 1 日零时到该日期时间的时间间隔秒数。日期时间会涉及到所在时区、夏时制等问题，比较复杂。

lubridate 是一个提供了许多日期和日期时间功能的扩展包。对于`"2018-07-02"` 这样的字符型的日期，可以用 `lubridate::ymd("2018-07-02")` 这样的方法转换成 Date 类型，该函数也可以输入字符型向量，向量的每个元素是一个日期字符串，结果为 Date 类型元素组成的向量。

如果 `year, mon, day` 分别是数值类型的年、月、日，可以用 `lubridate::make_date(year, mon, day)` 转换成 Date 类型。

`lubridate::ymd_hms()` 可以将`"2018-07-02 13:15:59"` 这样的日期和时间字符串转换成 POSIXct 类型，`lubridate::make_datetime()` 可以将单独的整数表示的年、月、日、时、分、秒转换成 POSIXct 类型。

`lubridate` 包的 `year(), month(), mday()` 可以从日期或日期时间中取出年、月、日，`hour()、minute()、second()` 可以从日期时间中取出时、分、秒。

更详细的用法见R 软件教程。

1.2.8 读入时间序列数据

常见的数据形式是用文本格式保存，数据之间用空行、空格或者逗号分隔。

1.2.8.1 单个向量读入

最简单的情况是仅有一个序列，数据仅包含序列的数值而不包含时间标签，这时数据可以保存为用空行和空格分隔的文本格式数据，例如文件 `bp500-2691.txt` 是标准普尔 500 指数月超额收益率从 1926 年 1 月到 1991 年 12 月的数据，内容为（节选）：

```
0.0225 -0.044 -0.0591 0.0227 0.0077 0.0432 0.0455 0.0171 0.0229 -0.0313 0.0223 0.0166
-0.0208 0.0477 0.0065 0.0172 0.0522 -0.0094 0.065 0.0445 0.0432 -0.0531 0.0678 0.019
```

```
-0.0051 -0.0176 0.1083 0.0324 0.0127 -0.0405 0.0125 0.0741 0.024 0.0145 0.1199 0.0029
0.0571 -0.0058 -0.0023 0.0161 -0.0428 0.1124 0.0456 0.098 -0.0489 -0.1993 -0.1337 0.0253
```

可以用如下程序读入成 R 向量：

```
x <- scan("bp500-2691.txt")
```

1.2.8.2 带有时间标签的序列

有些序列带有时间标签，同一行的数据之间用空格或者制表符分隔。文件 `d-ibm-0110.txt` 为 IBM 股票从 2001-1-2 到 2010-12-31 的日简单收益率，部分数据如下：

```
date            return
20010102      -0.002206
20010103       0.115696
20010104      -0.015192
20010105       0.008719
20010108      -0.004654
20010109      -0.010688
20010110       0.009453
```

可以用如下程序读入成一个 R 数据框：

```
library(tidyverse)
library(lubridate)
d <- read_table(
  "d-ibm-0110.txt", col_types=cols(
    .default=col_double(),
    date=col_date(format="%Y%m%d")))
```

有些序列的年月日是分开输入的，比如，文件 `m-unrate.txt` 为美国从 1948 年 1 月到 2010 年 9 月的经季节调整的月失业率百分数，部分数据如：

```
Year mon dd  rate
1948 01 01   3.4
1948 02 01   3.8
1948 03 01   4.0
1948 04 01   3.9
1948 05 01   3.5
```

可以用如下程序读入为 R 数据框，并生成日期列：

```
d <- read_table(
  "m-unrate.txt", col_types=cols(
    .default=col_double())) %>%
  mutate(date = make_date(Year, mon, dd)) %>%
  select(-Year, -mon, -dd) %>%
  select(date, everything())
```

同一个文件中有多个序列时，也可以用如上方法读入为 R 数据框。如 m-ibmsp-2611.txt 包含了 IBM 和标普 500 从 1926-01 到 2011-09 的月度简单收益率，部分数据如下：

```
date            ibm           sp
19260130      -0.010381      0.022472
19260227      -0.024476     -0.043956
19260331      -0.115591     -0.059113
19260430       0.089783      0.022688
```

读入为 tibble 数据框：

```
d <- read_table(
  "m-ibmsp-2611.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
```

1.2.8.3 读入 csv 格式

CSV 文件也是一种文本格式的数据文件，相对而言更规范一些，所以经常用于在不同的数据管理和数据分析软件之间传递数据。数据如：

```
"date","ibm","sp"
"19260130",-0.010381,0.022472
"19260227",-0.024476,-0.043956
"19260331",-0.115591,-0.059113
"19260430",0.089783,0.022688
"19260528",0.036932,0.007679
"19260630",0.068493,0.043184
"19260731",0,0.045455
```

可以用如下程序读入为 R 数据框：

```
library(tidyverse)
d <- read_csv(
  "m-ibmsp-2611.csv",
  col_types=cols(
    date=col_date(format="%Y%m%d"),
    .default=col_double()
  )
)
```

1.3 生成时间序列数据

R 软件中有些时间序列分析函数需要特定的时间序列类型数据作为输入。常用的有 ts 类型、xts 类型等。

1.3.1 生成 ts 类型

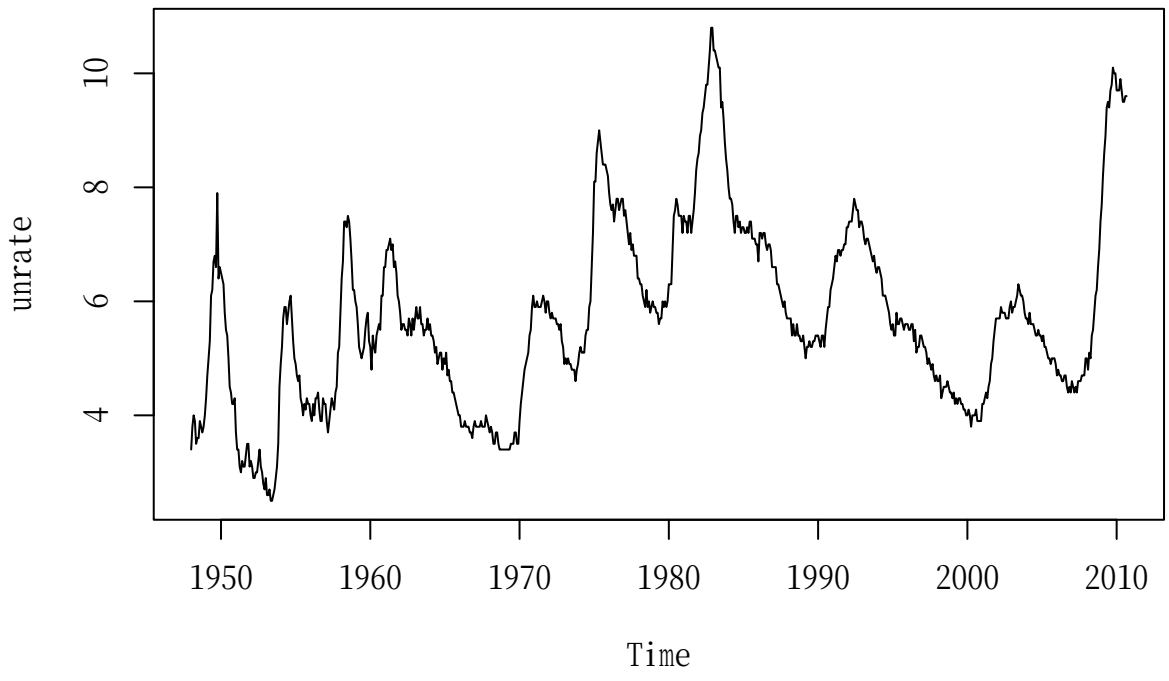
ts 类型用于保存一元或者多元的等间隔时间序列，如月度、季度、年度数据。生成方法如

```
ts(x, start=c(2001, 1), frequency=12)
```

其中 x 是向量或者矩阵，取矩阵值时矩阵的每一列是一个序列。frequency 对月度数据是 12，对季度数据是 4，对年度数据可以缺省（值为 1）。start=c(2001,1) 表示序列开始时间是 2001 年 1 月。如果是年度数据，用如 ts(x, start=2001) 即可。

例如，文件 m-unrate.txt 为美国从 1948 年 1 月到 2010 年 9 月的经季节调整的月失业率百分数，转换成 ts 类型并作时间序列曲线图的程序如下：

```
d <- read.table(
  "m-unrate.txt", header=TRUE,
  colClasses=rep("numeric", 4))
unrate <- ts(d[["rate"]], start=c(1948,1), frequency=12)
plot(unrate)
```



1.3.2 生成 xts 类型

xts 也是一种时间序列数据类型，既可以保存等间隔时间序列数据，也可以保存不等间隔的时间序列数据，并且 xts 类型的数据访问功能更为方便。生成方法如

```
xts(x, date)
```

其中 x 是向量、矩阵或数据框，date 是日期或者日期时间。x 取矩阵或者数据框时每列是一个时间序列。

例如，文件 d-ibm-0110.txt 为 IBM 股票从 2001-1-2 到 2010-12-31 的日简单收益率，读入数据并转换为 xts 类

型并作图的程序如下：

```
library(xts)
d <- read_table(
  "d-ibm-0110.txt", col_types=cols(
    .default=col_double(),
    date=col_date(format="%Y%m%d")))
ibmrtn <- xts(d[["return"]], d[["date"]])
plot(ibmrtn,
  main="IBM Log Return Daily",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  format.labels="%Y",
  col="red")
```

有了 xts 类型的变量 x 后，可以用 coredata(x) 返回 x 的不包含时间的纯数据；用 index(x) 返回 x 的时间标签。

xts 类型可以比较容易地取出子集，如：

- x["2003"] 取出 2003 年的子集；
- x["2003-12"] 取出 2003 年 12 月的子集；
- ‘x[“2003-07/2004-06”] 取出 2003 年 7 月到 2004 年 6 月的子集，等等。

下面几节更详细地叙述各种时间序列对象的用法。

1.4 ts 类型

ts 是基本 R 软件的 stats 包中支持的规则时间序列类型，具有 start 和 frequency 两个属性，对于年度数据，frequency=1，对于月度数据，frequency=12。但是日数据最好不时间标签为具体日历时间的 ts 类型，因为金融数据的日数据在周末和节假日一般无数据，而 ts 类型要求时间是每一天都相连的，不能周五直接跳到周一。

用函数 ts 把一个向量转换为时间序列。如

```
yd <- ts(
  c(4, 8, 7, 7, 3, 1, 8, 9, 8, 6, 3, 5,
    5, 8, 2, 5, 9, 2, 5, 2, 3, 2, 2, 4),
  frequency=1, start=2001); yd
```

```
## Time Series:
## Start = 2001
## End = 2024
## Frequency = 1
##  [1] 4 8 7 7 3 1 8 9 8 6 3 5 5 8 2 5 9 2 5 2 3 2 2 4
```

```
ym <- ts(
  c(9, 6, 3, 5, 4, 8, 2, 5, 8, 4, 6, 3,
    2, 2, 6, 4, 1, 4, 2, 1, 8, 9, 4, 6),
```

```
  frequency=12, start=c(2001,1)); ym
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2001   9   6   3   5   4   8   2   5   8   4   6   3
## 2002   2   2   6   4   1   4   2   1   8   9   4   6
```

其中 freqency 指定采样频率，如月数据是 12。start 指定开始点，如上面的 yd 是年数据，开始于 2001 年；ym 是月数据开始于 2001 年 1 月。

对于多元时间序列，可以用 ts 函数把一个矩阵转换为 ts 类型的多元时间序列对象。用 ts.intersect 函数和 ts.union 函数可以把两个或多个时间序列合并为多元时间序列，时间段取交集或者取并集。

为了使用序列数据进行计算、绘图等，可以用 as.vector 把一元时间序列的数据转换成普通向量。如：

```
as.vector(ym)
```

```
##  [1] 9 6 3 5 4 8 2 5 8 4 6 3 2 2 6 4 1 4 2 1 8 9 4 6
```

对于多元时间序列 x，可以用 x[,1] 这样的格式取出其中分量时间序列，可以用 as.vector(x[,1]) 将其中的分量转换为普通向量，可以借助于 xts 类型用 coredata(as.xts(x)) 将多元时间序列的数据转换为普通矩阵。

在 R 中已安装的时间列示例数据有美国泛美航空公司 1949-1960 的国际航班订票数的月度数据（单位：千人），12 年 144 个月。用

```
data(AirPassengers)
```

调入名字空间。类型和属性如下：

```
attributes(AirPassengers)
```

```
## $tsp
## [1] 1949.000 1960.917   12.000
##
## $class
## [1] "ts"
```

这里时间以年为单位，数据类属为 ts，两个数据值之间的时间间隔为 1/12。

AirPassengers 的具体数值：

```
AirPassengers
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
```

```
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

用 start() 求时间序列的开始点，end() 求时间序列的结束点，frequency() 求采样频率。如

```
start(AirPassengers)
```

```
## [1] 1949    1
```

```
end(AirPassengers)
```

```
## [1] 1960   12
```

```
frequency(AirPassengers)
```

```
## [1] 12
```

aggregate() 函数可以把月度数据加总成年数据。如

```
AP.year <- aggregate(AirPassengers); AP.year
```

```
## Time Series:
## Start = 1949
## End = 1960
## Frequency = 1
##  [1] 1520 1676 2042 2364 2700 2867 3408 3939 4421 4572 5140 5714
```

如果加参数 FUN=mean 可以取均值。

time() 函数对 ts 类型数据返回序列中的每个时间点的时间，结果是一个和原来时间序列形状相同的时间序列。cycle() 函数对月度数据返回序列每个时间点所在的月份, 结果是和原序列时间点相同的一个时间序列。如

```
cy.AP <- cycle(AirPassengers); cy.AP
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949   1   2   3   4   5   6   7   8   9  10  11  12
## 1950   1   2   3   4   5   6   7   8   9  10  11  12
## 1951   1   2   3   4   5   6   7   8   9  10  11  12
## 1952   1   2   3   4   5   6   7   8   9  10  11  12
## 1953   1   2   3   4   5   6   7   8   9  10  11  12
## 1954   1   2   3   4   5   6   7   8   9  10  11  12
## 1955   1   2   3   4   5   6   7   8   9  10  11  12
## 1956   1   2   3   4   5   6   7   8   9  10  11  12
## 1957   1   2   3   4   5   6   7   8   9  10  11  12
## 1958   1   2   3   4   5   6   7   8   9  10  11  12
## 1959   1   2   3   4   5   6   7   8   9  10  11  12
## 1960   1   2   3   4   5   6   7   8   9  10  11  12
```

window() 函数取出时间序列的一段，如果指定 frequency=TRUE 还可以仅取出某个月（季度）。如

```
AP.Jan <- window(AirPassengers, start=c(1949,1),
                 frequency=TRUE); AP.Jan
```

```
## Time Series:
## Start = 1949
## End = 1960
## Frequency = 1
##  [1] 112 115 145 171 196 204 242 284 315 340 360 417
```

stats::lag() 可以计算滞后序列，对 ts 类型输入，lag() 的作用是序列数值不变，但是时间标签增加一个单位或者用 k= 指定的间隔。所以，为了得到通常理解的滞后 1 序列，应该使用 stats::lag(x, k=-1)，如

```
x1 <- ts(1:5, start=2001); x1
```

```
## Time Series:
## Start = 2001
## End = 2005
## Frequency = 1
## [1] 1 2 3 4 5
```

```
x2 <- stats::lag(x1, k=-1); x2
```

```
## Time Series:
## Start = 2002
## End = 2006
## Frequency = 1
## [1] 1 2 3 4 5
```

```
cbind(x1, x2)
```

```
## Time Series:
## Start = 2001
## End = 2006
## Frequency = 1
##      x1 x2
## 2001  1 NA
## 2002  2  1
## 2003  3  2
## 2004  4  3
## 2005  5  4
## 2006 NA  5
```

diff(x) 计算一阶差分，diff(x, lag, differences) 计算滞后为 lag 的阶数位 differences 的差分。

ts.union 可以形成多元时间序列。

filter 函数可以计算递推的或卷积的滤波。

arima 可以拟合 ARIMA 模型。

acf 和 pacf 函数可以作自相关和偏自相关图。

1.5 zoo 类型

R 的 zoo 扩展包提供了比基本 R 中 ts 时间序列类型更灵活的时间序列类型，其时间标签（time stamp）可以使用 R 中任何的日期和时间类型，序列不需要是等时间间隔的，支持多元时间序列。如果序列符合 ts 类型的要求，则与 ts 类型兼容并可以互相转换。zoo 也尽可能提供与 ts 类相同或相似功能的函数。zoo 扩展包提供的时间序列数据类型叫做 zoo 类型。

1.5.1 生成 zoo 类型的时间序列

生成 zoo 类型的时间序列，只要提供两部分输入：一个向量或者矩阵 x 作为观测值，一个下标序列 order.by 作为排序变量以及时间标签。zoo 的设计特点是允许使用任何可排序的数据类型作为时间标签。语法为

```
zoo(x, order.by)
```

当 x 是矩阵时，每行对应同一时间点，不同列是不同的时间序列，但同一行的各个数值对应相同的时间点。

例如

```
set.seed(1)
z.1 <- zoo(sample(3:6, size=12, replace=TRUE),
           make_date(2018, 1, 1) + ddays(0:11)); z.1
```

```
## 2018-01-01 2018-01-02 2018-01-03 2018-01-04 2018-01-05 2018-01-06 2018-01-07
##          3          6          5          3          4          3          5
## 2018-01-08 2018-01-09 2018-01-10 2018-01-11 2018-01-12
##          5          4          4          5          5
```

```
set.seed(2)
z.2 <- zoo(cbind(x=sample(5:10, size=12, replace=TRUE),
                 y=sample(8:13, size=12, replace=TRUE)),
           make_date(2018, 1, 1) + ddays(0:11)); z.2
```

```
##             x  y
## 2018-01-01  9 10
## 2018-01-02 10 13
## 2018-01-03 10  9
## 2018-01-04  5 10
## 2018-01-05  9  8
## 2018-01-06  5 13
## 2018-01-07  8  8
## 2018-01-08  9 11
## 2018-01-09  5 10
## 2018-01-10  6 13
## 2018-01-11  7  8
## 2018-01-12  5 13
```

时间不必是连续的，如

```
set.seed(1)
z.3 <- zoo(101:112,
           make_date(2018, 1, 1) +
             ddays(sample(0:30, 12, replace=FALSE))); z.3
```

```
## 2018-01-01 2018-01-02 2018-01-04 2018-01-07 2018-01-10 2018-01-11 2018-01-14
##        104        105        102        103        112        107        108
## 2018-01-18 2018-01-19 2018-01-23 2018-01-25 2018-01-28
##        109        110        106        101        111
```

注意输入的随机生成的日期次序是乱序的，但生成 zoo 时间序列后一定会按照正常时间序列排好次序。

ts、irts（在 tseries 包中定义）、its 等时间序列类型可以用 `as.zoo()` 转换为 zoo 类型，如果 zoo 类型的时间下标符合转换要求，也可以将 zoo 时间序列用 `as.xxx()` 类函数转换为其它时间序列类型。

文本文件中、字符串中、数据框中保存的时间序列数据可以用 `read.zoo()` 转换为 zoo 时间序列。

1.5.2 zoo 类型的概括情况

设 x 为 zoo 类型的时间序列，`print(x)` 显示 x，对一元序列横向显示，对多元序列纵向显示。

`str(x)` 显示 x 的类型和结构信息，如

```
str(z.2)
```

```
## 'zoo' series from 2018-01-01 to 2018-01-12
##   Data: int [1:12, 1:2] 9 10 10 5 9 5 8 9 5 6 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:2] "x" "y"
##   Index:  Date[1:12], format: "2018-01-01" "2018-01-02" "2018-01-03" "2018-01-04" "2018-01-05" ...
```

`head(x)` 和 `tail(x)` 可以取出序列开头的若干项与末尾的若干项，如:

```
head(z.2)
```

```
##             x  y
## 2018-01-01  9 10
## 2018-01-02 10 13
## 2018-01-03 10  9
## 2018-01-04  5 10
## 2018-01-05  9  8
## 2018-01-06  5 13
```

`summary(x)` 对每个序列以及时间下标作简单概括统计，如

```
summary(z.2)
```

```
##       Index                  x                y
##  Min.   :2018-01-01   Min.   : 5.000   Min.   : 8.00
```

```
##  1st Qu.:2018-01-03   1st Qu.: 5.000   1st Qu.: 8.75
##  Median :2018-01-06   Median : 7.500   Median :10.00
##  Mean   :2018-01-06   Mean   : 7.333   Mean   :10.50
##  3rd Qu.:2018-01-09   3rd Qu.: 9.000   3rd Qu.:13.00
##  Max.   :2018-01-12   Max.   :10.000   Max.   :13.00
```

1.5.3 zoo 类型的子集

对 zoo 类型的时间序列，不管它保存的是一元序列还是多元序列，都可以向对向量那样用一维的下标取子集，如

```
z.2[1]
```

```
##            x  y
## 2018-01-01 9 10
```

```
z.2[10:12]
```

```
##            x  y
## 2018-01-10 6 13
## 2018-01-11 7  8
## 2018-01-12 5 13
```

对多元序列，也可以象矩阵那样取子集，如

```
z.2[1:3, 1:2]
```

```
##             x  y
## 2018-01-01  9 10
## 2018-01-02 10 13
## 2018-01-03 10  9
```

```
z.2[1:3, 2]
```

```
## 2018-01-01 2018-01-02 2018-01-03
##         10         13          9
```

序列也可以直接用时间作为下标，比如

```
z.2[ymd(c("2018-01-01", "2018-01-12"))]
```

```
##            x  y
## 2018-01-01 9 10
## 2018-01-12 5 13
```

如果时间下标本身也是数值，而且不等于序号，可以在用作下标时包裹在 `I()` 函数中以免被认作序号下标。

1.5.4 zooreg 类型

为了与规则时间序列的 ts 类型相对应，zoo 扩展包提供了 zoo 的派生类型 zooreg，代表规则间隔的时间序列，它具有与 ts 相同的时间间隔信息，待允许内部某些时间点不存在。

用 `zoo()` 函数加 `frequency` 选项或者 `zooreg()` 函数生成 zooreg 类型的时间序列：

```
zoo(x, order.by, freqency)
zooreg(x, start, end, frequency, deltat, ts.eps, order.by)
```

例如：

```
zr.1 <- zooreg(c(1, 3, 5, 7, 9, 11, 13, 17),
               start=2001,frequency = 4)
zr.1
```

```
## 2001 Q1 2001 Q2 2001 Q3 2001 Q4 2002 Q1 2002 Q2 2002 Q3 2002 Q4
##       1       3       5       7       9      11      13      17
```

zooreg 类型比 ts 优点是即使删去中间若干点，其类型仍保持为 zooreg 类型：

```
zr.1b <- zr.1[-c(2,4,6,8)]
zr.1b
```

```
## 2001 Q1 2001 Q3 2002 Q1 2002 Q3
##       1       5       9      13
```

```
class(zr.1b)
```

```
## [1] "zooreg" "zoo"
```

```
frequency(zr.1b)
```

```
## [1] 4
```

这样比 ts 必须以 NA 保存缺失数值的时间点要容易处理。

zooreg 也可以以日、时、分、秒等作为时间间隔，如

```
zr.2 <- zooreg(c(1, 3, 5, 7, 9, 11, 13, 17),
               start=ymd("2018-01-01"), frequency = 1)
zr.2
```

```
## 2018-01-01 2018-01-02 2018-01-03 2018-01-04 2018-01-05 2018-01-06 2018-01-07
##          1          3          5          7          9         11         13
## 2018-01-08
##         17
```

可以用 `is.regular(x)` 判断一个 zoo 类型数据是否规则，用 `is.regular(x, strict=TRUE)` 判断一个 zoo 类型数据是否满足和 ts 一样条件的规则序列。

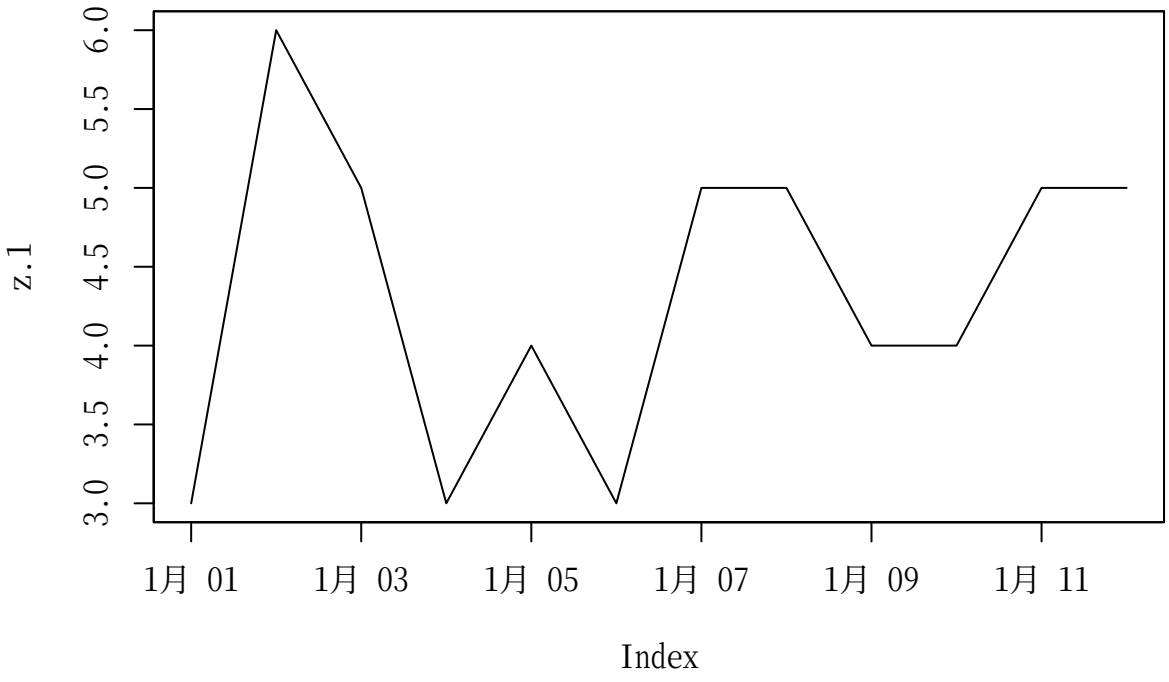
用 `as.ts(x)` 将一个规则的 zooreg 或者 zoo 类型数据转换为 ts 类型的时间序列。如果 `x` 内部有缺失的时间点，转换为 ts 类型时将补上这些时间点，观测值填以 `NA`。当 zoo 类型的下标是数值型、月度、季度时可以与 ts 无损地互相转换。用 `as.zoo(x)` 将 ts 类型的时间序列转换为 zoo 类型。

如果 `x` 可以转换为 ts 类型, 则对 ts 可以应用的 `acf()`, `arima()`, `stl()` 等函数也可以对 `x` 应用。

1.5.5 对 zoo 类型作图

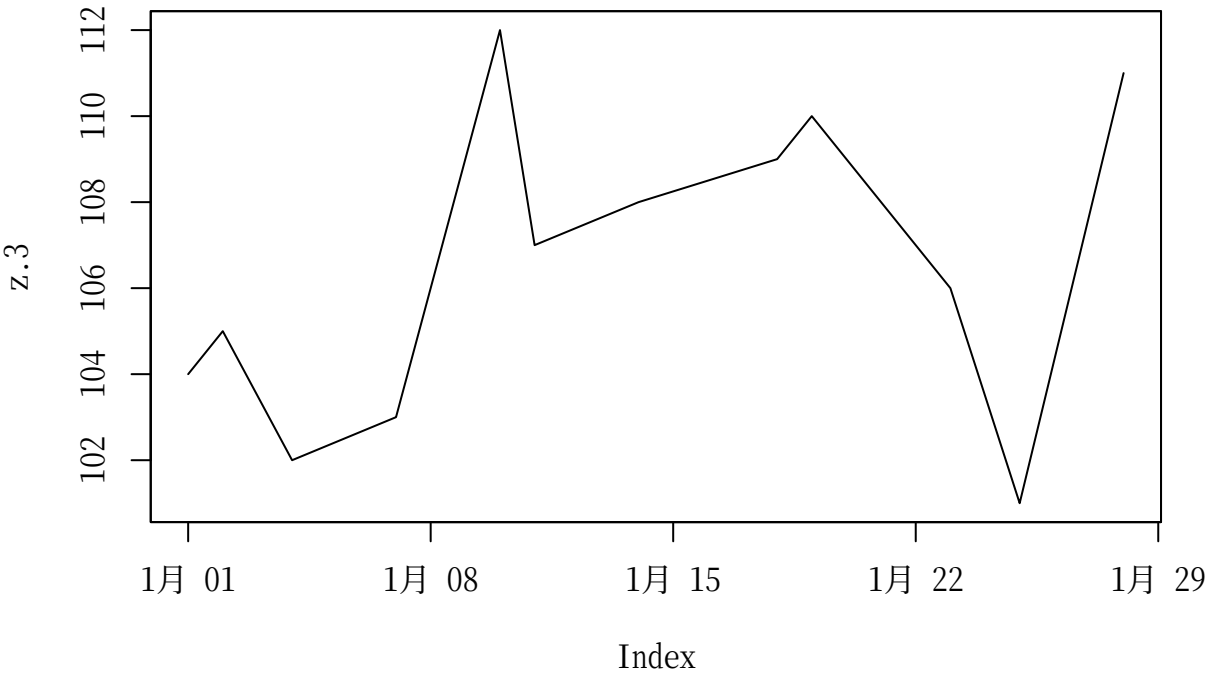
用 `plot()` 函数对 zoo 类型数据做图。一元时如

```
plot(z.1)
```



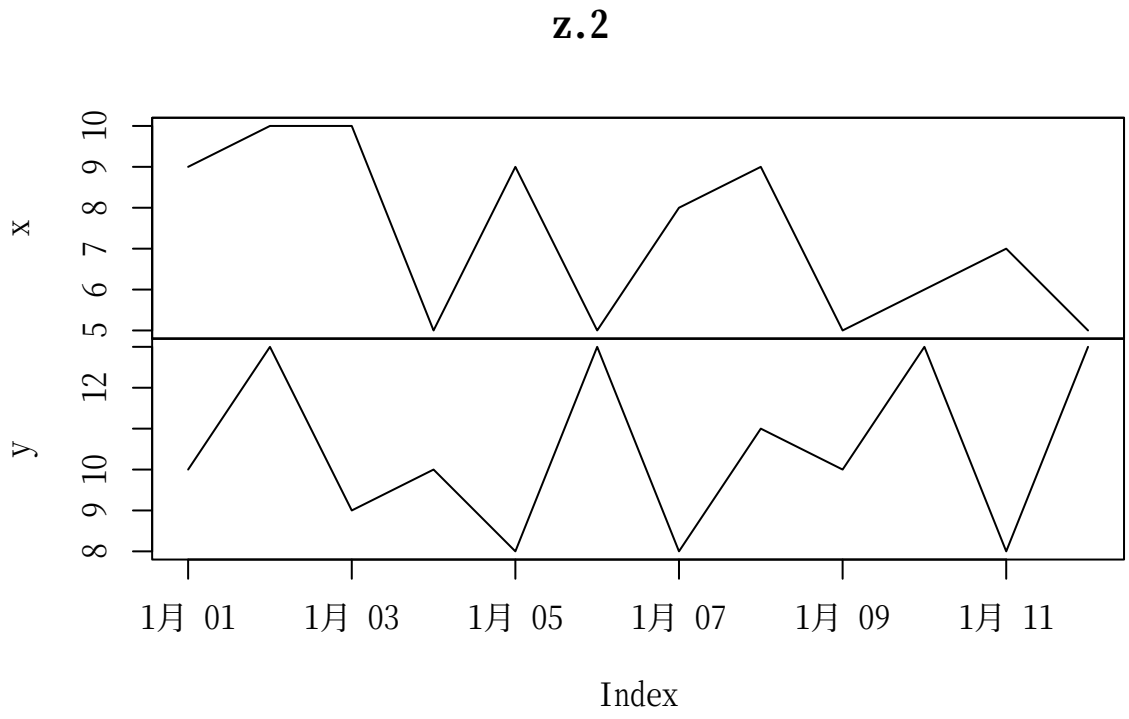
序列不规则时：

```
plot(z.3)
```



对多元序列，plot() 缺省每个序列单独画在一个窗格，如

```
plot(z.2)
```



也可以用 `plot.type="single"` 要求画在同一坐标系中：

```
plot(z.2, plot.type="single", col=c("red", "blue"))
```

plot() 中仍可以用 col 指定颜色，用 lty 指定线型，用 lwd 指定线粗细，用 pch 指定散点类型，用 cex 指定散点大小。当时间跨度为若干年时，经常用 format.labels="%Y" 选项使得时间轴刻度标签仅包含年份。对多元序列，可以为每条曲线指定不同的图形属性。指定图形属性时，可以用一个列表指定，列表元素名对应于序列中某属性的名字，如

```
plot(z.2, plot.type="single",
     col=list(x="red", y="blue"))
```

并且可以仅指定部分属性的图形参数，其它用默认值。

autoplot() 可以基于 ggplot2 包对时间序列绘图。

1.5.6 序列合并

两个保存了不同时间段的时间序列 x, y 可以用 c(x,y) 或者 rbind(x,y) 合并，如

```
c(z.1[1:4], z.1[9:12])
```

```
## 2018-01-01 2018-01-02 2018-01-03 2018-01-04 2018-01-09 2018-01-10 2018-01-11
##          3          6          5          3          4          4          5
## 2018-01-12
##          5
```

```
c(z.2[1:4], z.2[9:12])
```

```
##             x  y
```

```
## 2018-01-01  9 10
## 2018-01-02 10 13
## 2018-01-03 10  9
## 2018-01-04  5 10
## 2018-01-09  5 10
## 2018-01-10  6 13
## 2018-01-11  7  8
## 2018-01-12  5 13
```

两个不同属性的时间序列，可以合并为一个多元时间序列，时间取并集，不存在的值取缺失值。两个序列不需要等长。如

```
x <- cbind(z.2[3:7], z.3[1:4]); x
```

```
##             x  y z.3[1:4]
## 2018-01-01 NA NA      104
## 2018-01-02 NA NA      105
## 2018-01-03 10  9       NA
## 2018-01-04  5 10      102
## 2018-01-05  9  8       NA
## 2018-01-06  5 13       NA
## 2018-01-07  8  8      103
```

用 `names(x)` 给多元时间序列的各个分量赋值，如

```
names(x) <- c("x", "y", "z3")
x
```

```
##             x  y  z3
## 2018-01-01 NA NA 104
## 2018-01-02 NA NA 105
## 2018-01-03 10  9  NA
## 2018-01-04  5 10 102
## 2018-01-05  9  8  NA
## 2018-01-06  5 13  NA
## 2018-01-07  8  8 103
```

为了在合并时仅保留共同的时间点，可以用如

```
merge(z.2[3:7], z.3[1:4], all=FALSE)
```

```
##            x  y z.3[1:4]
## 2018-01-04 5 10      102
## 2018-01-07 8  8      103
```

`merge()` 也可以同时横向合并多个序列，如 `merge(x, y, z)`。除了第一个自变量，允许被合并的其它成分不是时间序列，这时沿用第一个自变量的时间信息。如

```
merge(z.1, 1:12)
```

```
##            z.1 1:12
## 2018-01-01   3    1
## 2018-01-02   6    2
## 2018-01-03   5    3
## 2018-01-04   3    4
## 2018-01-05   4    5
## 2018-01-06   3    6
## 2018-01-07   5    7
## 2018-01-08   5    8
## 2018-01-09   4    9
## 2018-01-10   4   10
## 2018-01-11   5   11
## 2018-01-12   5   12
```

1.5.7 降频与升频

类似于 ts 时间序列，对 zoo 时间序列，也可以用 `aggregate()` 对数据进行降频，比如，从日数据转换成月数据。例：

```
z.ap <- as.zooreg(AirPassengers)
z.apy <- aggregate(z.ap, year, sum)
z.apy
```

```
## 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960
## 1520 1676 2042 2364 2700 2867 3408 3939 4421 4572 5140 5714
```

`aggregate()` 的一般用法是

```
aggregate(x, by, FUN)
```

其中 `by` 可以是与序列等长的一个分组值向量，也可以是一个函数，此函数接受 `index(x)` 为自变量生成分组值，如上例。`FUN` 是对每组作的操作。

在每组中取最后一个值的例子：

```
z.apyc <- aggregate(
  z.ap, year, tail, 1)
z.apyc
```

```
## 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960
##  118  140  166  194  201  229  278  306  336  337  405  432
```

有时需要对序列提高时间频率，比如，将年度数据转换成月度数据。办法是与一个高频的仅用时间下标而无实际数据的序列横向合并。例如，基本 R 软件中有个 `Nile` 对象是尼罗河 1871-1970 的年流量（单位：亿立方米），将其升频到季度数据：

```
z.Nile <- as.zoo(Nile)
head(z.Nile)
```

```
## 1871 1872 1873 1874 1875 1876
## 1120 1160  963 1210 1160 1160
```

```
z.NileM <- merge(
  z.Nile,
  zoo(, seq(start(z.Nile), end(z.Nile), by=1/4))
)
head(z.NileM, 6)
```

```
## 1871(1) 1871(2) 1871(3) 1871(4) 1872(1) 1872(2)
##    1120      NA      NA      NA    1160      NA
```

在升频时年度值变成了第一季度的值，其它三个季度变成了缺失值。这是因为原始的年度数据相当于是 1 月 1 日的数据，可以和第一季度的代表日期对准。

用 na.locf() 可以填补缺失值，填前面最近的一个非缺失值：

```
head(na.locf(z.NileM))
```

```
## 1871(1) 1871(2) 1871(3) 1871(4) 1872(1) 1872(2)
##    1120    1120    1120    1120    1160    1160
```

用 na.approx() 可以对缺失值填充为线性插值，如

```
head(na.approx(z.NileM))
```

```
## 1871(1) 1871(2) 1871(3) 1871(4) 1872(1) 1872(2)
## 1120.00 1130.00 1140.00 1150.00 1160.00 1110.75
```

1.5.8 zoo 时间序列的数学计算

两个序列之间可以进行加减乘除四则运算和逻辑比较运算，结果是对应时间点的相应运算。缺失值被删除。如

```
z.2[3:7] + z.3[1:4]
```

```
##              x   y
## 2018-01-04 107 112
## 2018-01-07 111 111
```

cumsum(), cumprod(), cummin(), cummax() 等函数仍可作用到 zoo 时间序列的每个序列上。

如果序列是价格，用如下方法计算简单收益率与对数收益率：

```
simple.return <- function(x) diff(x)/lag(x, k=-1)[-1]
simple.return(z.1)
```

```
## 2018-01-02 2018-01-03 2018-01-04 2018-01-05 2018-01-06 2018-01-07 2018-01-08
##  0.5000000 -0.2000000 -0.6666667  0.2500000 -0.3333333  0.4000000  0.0000000
## 2018-01-09 2018-01-10 2018-01-11
## -0.2500000  0.0000000  0.2000000
```

```
log.return <- function(x) diff(log(x))
log.return(z.1)
```

```
## 2018-01-02 2018-01-03 2018-01-04 2018-01-05 2018-01-06 2018-01-07 2018-01-08
##  0.6931472 -0.1823216 -0.5108256  0.2876821 -0.2876821  0.5108256  0.0000000
## 2018-01-09 2018-01-10 2018-01-11 2018-01-12
## -0.2231436  0.0000000  0.2231436  0.0000000
```

这样的简单版本删去了第一个时间点，因为这个时间点的盈利缺失。为了包括此点，做法如

```
simple.return <- function(x){
  d <- merge(lag(x, k=-1), x, all=TRUE)
  (d[,2] - d[,1])/d[,1]
}
simple.return(z.1)
```

```
## 2018-01-01 2018-01-02 2018-01-03 2018-01-04 2018-01-05 2018-01-06 2018-01-07
##          0          0          0          0          0          0          0
## 2018-01-08 2018-01-09 2018-01-10 2018-01-11 2018-01-12       <NA>
##          0          0          0          0         NA         NA
```

以上两个 simple.return() 函数定义都依赖于 lag() 函数或者日期对齐，对 zoo 和 xts 类型可用，但是对其它时间序列类型有可能出错。simple.return() 的如下定义不需要输入的时间序列特性：

```
simple.return <- function(x){
  x <- as.vector(x)
  c(NA, diff(x) / x[1:(length(x)-1)])
}
simple.return(z.1)
```

```
##  [1]         NA  1.0000000 -0.1666667 -0.4000000  0.3333333 -0.2500000
##  [7]  0.6666667  0.0000000 -0.2000000  0.0000000  0.2500000  0.0000000
```

1.5.9 zoo 属性访问

zoo 时间序列类型的实际数据可以用 coredata() 读取或修改。读取时对一元序列结果是数值型向量，对多元序列结果是矩阵。

如

```
coredata(z.2)
```

```
##       x  y
##  [1,]  9 10
##  [2,] 10 13
##  [3,] 10  9
##  [4,]  5 10
##  [5,]  9  8
##  [6,]  5 13
```

```
##  [7,]  8  8
##  [8,]  9 11
##  [9,]  5 10
## [10,]  6 13
## [11,]  7  8
## [12,]  5 13
```

```
z.2b <- z.2
coredata(z.2b) <- 100 + coredata(z.2b)
z.2b
```

```
##              x   y
## 2018-01-01 109 110
## 2018-01-02 110 113
## 2018-01-03 110 109
## 2018-01-04 105 110
## 2018-01-05 109 108
## 2018-01-06 105 113
## 2018-01-07 108 108
## 2018-01-08 109 111
## 2018-01-09 105 110
## 2018-01-10 106 113
## 2018-01-11 107 108
## 2018-01-12 105 113
```

用 as.data.frame() 将 zoo 对象转换成数据框，时间下标变成数据框的行名，如

```
as.data.frame(z.2)
```

```
##             x  y
## 2018-01-01  9 10
## 2018-01-02 10 13
## 2018-01-03 10  9
## 2018-01-04  5 10
## 2018-01-05  9  8
## 2018-01-06  5 13
## 2018-01-07  8  8
## 2018-01-08  9 11
## 2018-01-09  5 10
## 2018-01-10  6 13
## 2018-01-11  7  8
## 2018-01-12  5 13
```

zoo 时间序列的时间下标可以用 index() 函数读取或修改，如

```
index(z.1)
```

```
##  [1] "2018-01-01" "2018-01-02" "2018-01-03" "2018-01-04" "2018-01-05"
```

```
##  [6] "2018-01-06" "2018-01-07" "2018-01-08" "2018-01-09" "2018-01-10"
## [11] "2018-01-11" "2018-01-12"
```

```
z.1b <- z.1
index(z.1b) <- seq(from=ymd("2001-07-01"), by=1, length.out=12)
z.1b
```

```
## 2001-07-01 2001-07-02 2001-07-03 2001-07-04 2001-07-05 2001-07-06 2001-07-07
##          3          6          5          3          4          3          5
## 2001-07-08 2001-07-09 2001-07-10 2001-07-11 2001-07-12
##          5          4          4          5          5
```

时间下标的开始和结尾时间用 start() 和 end() 访问：

```
start(z.1)
```

```
## [1] "2018-01-01"
```

```
end(z.1)
```

```
## [1] "2018-01-12"
```

用

```
window(x, index, start, end)
```

可以读取序列中的一段，或进行修改。如

```
window(z.1, start=ymd("2018-01-09"))
```

```
## 2018-01-09 2018-01-10 2018-01-11 2018-01-12
##          4          4          5          5
```

```
window(z.1, index=index(z.1)[1:3])
```

```
## 2018-01-01 2018-01-02 2018-01-03
##          3          6          5
```

zoo 序列也支持 lag() 函数和 diff() 函数。为了生成 z.1 序列的滞后一期序列，做法如

```
merge(z.1, lag(z.1, k=-1))
```

```
##            z.1 lag(z.1, k = -1)
## 2018-01-01   3                3
## 2018-01-02   6                6
## 2018-01-03   5                5
## 2018-01-04   3                3
## 2018-01-05   4                4
## 2018-01-06   3                3
## 2018-01-07   5                5
## 2018-01-08   5                5
## 2018-01-09   4                4
## 2018-01-10   4                4
```

```
## 2018-01-11   5                5
## 2018-01-12   5               NA
## <NA>        NA               NA
```

1.5.10 zoo 序列的缺失值处理

缺失值对实际统计工作造成很大的麻烦，金融时间序列中也包含许多缺失值。对 zoo 时间序列，可以用 `na.omit()`，`na.contiguous()`, `na.approx()`，`na.spline()`，`na.locf()` 等函数处理。`na.omit()` 可以删去含有缺失值的时间点。`na.contiguous()` 则从整个序列中抽取处连续无缺失的最长的一段。

`na.locf()` 将缺失值用时间更早的一个最近非缺失值插补，函数名是 last observation carried forward 的缩写。`na.approx()` 作线性插值，`na.spline()` 作样条插值。这三个函数都会删去序列最开始处的缺失观测。插值函数的横坐标默认使用日期时间的数值，也可以利用选项改为默认两个观测之间等间隔。

1.5.11 zoo 序列的滚动计算

对时间序列经常计算像七日均线这样的滑动平均。对于一般的向量，基本 R 的函数 `filter()` 可以进行各种加权滑动平均和自回归迭代计算。zoo 包对 zoo 时间序列和 ts 时间序列提供了 `rollapply()` 函数，可以计算包括滑动平均在内的多种滚动计算。格式为

```
rollapply(data, width, FUN)
```

其中 `data` 是输入序列，`width` 指定滚动窗口的时间点数，`FUN` 指定对滚动窗口的每一个位置的子集计算的函数。结果的时间点对准窗口的中点，可以选项 `align="right"` 指定结果的时间点对准窗口的最后一个时间点。

例如，每 5 个时间点计算标准差，每次向前滚动一个时间点，标准差的时间点对准 5 个点的中间一个的时间：

```
z.1
```

```
## 2018-01-01 2018-01-02 2018-01-03 2018-01-04 2018-01-05 2018-01-06 2018-01-07
##          3          6          5          3          4          3          5
## 2018-01-08 2018-01-09 2018-01-10 2018-01-11 2018-01-12
##          5          4          4          5          5
```

```
rollapply(z.1, 5, sd)
```

```
## 2018-01-03 2018-01-04 2018-01-05 2018-01-06 2018-01-07 2018-01-08 2018-01-09
##  1.3038405  1.3038405  1.0000000  1.0000000  0.8366600  0.8366600  0.5477226
## 2018-01-10
##  0.5477226
```

加 `fill=NA` 选项使得没有结果的时间点也出现在结果中：

```
rollapply(z.1, 5, sd, fill=NA)
```

```
## 2018-01-01 2018-01-02 2018-01-03 2018-01-04 2018-01-05 2018-01-06 2018-01-07
##         NA         NA  1.3038405  1.3038405  1.0000000  1.0000000  0.8366600
## 2018-01-08 2018-01-09 2018-01-10 2018-01-11 2018-01-12
##  0.8366600  0.5477226  0.5477226         NA         NA
```

对一些常用的滚动计算 zoo 包含提供了单独的函数，如 `rollmean()`, `rollmedian()`, `rollmax()`, `rollsum()`，这几个是结果时间点对应滚动窗口中心的。而 `rollmeanr()`, `rollmedianr()`, `rollmaxr()`, `rollsumr()` 则是结果时间点对准滚动窗口末尾时间点的。

1.6 xts 类型

1.6.1 xts 类型的设计目标

xts 包的设计目标是使得以 xts 为输入的函数可以兼容其它时间序列类型的输入，并可以很方便地增加属性。

xts 包提供了 xts 时间序列类型，这本质上是 zoo 包的 zoo 类型，所以针对 zoo 类型的做法也适用于 xts 类型。

xts 类型与 zoo 类型的差别如下：

- xts 类型必须使用正式的时间类型作为时间下标，zoo 类型的时间下标更普遍。xts 目前仅允许使用 Date, POSIXct, chron, yearmon, yearqtr, timeDate 作为时间下标。
- xts 包含一些独有的内部属性，包括.CLASS, .ROWNAMES 等。这是为了能够无损地与其他时间序列类型互相转换而保存的其它时间序列类型的部分额外信息。
- xts 具有用户添加属性。这使得 xts 类型是用户可定制的数据类型，用户可以随意添加或者删除自定义属性，不会影响到数据的显示和计算。与其他时间序列类型互相转换所需的部分额外信息也保存为用户添加属性。

1.6.2 xts 类型数据的构建

可以用 `xts()` 生成新的 xts 时间序列类型的数据对象，用法类似用 `zoo::zoo()`。如

```
library(xts, quietly = TRUE)
xts.1 <- xts(
  c(5, 5, 4, 6, 4, 3, 3, 3, 4, 5, 5, 4),
  make_date(2018, 1, 1) + ddays(0:11)); xts.1
```

```
##            [,1]
## 2018-01-01    5
## 2018-01-02    5
## 2018-01-03    4
## 2018-01-04    6
## 2018-01-05    4
## 2018-01-06    3
## 2018-01-07    3
## 2018-01-08    3
## 2018-01-09    4
## 2018-01-10    5
## 2018-01-11    5
## 2018-01-12    4
```

```
str(xts.1)
```

```
## An 'xts' object on 2018-01-01/2018-01-12 containing:
```

```
##   Data: num [1:12, 1] 5 5 4 6 4 3 3 3 4 5 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

其它时间序列类型可以用 as.xts() 转换成 xts 类型，如

```
as.xts(z.1)
```

```
##            [,1]
## 2018-01-01    3
## 2018-01-02    6
## 2018-01-03    5
## 2018-01-04    3
## 2018-01-05    4
## 2018-01-06    3
## 2018-01-07    5
## 2018-01-08    5
## 2018-01-09    4
## 2018-01-10    4
## 2018-01-11    5
## 2018-01-12    5
```

1.6.3 xts 类型数据的子集

首先，xts 类型支持 zoo 类型的用法。

若 x 是 xts 类型，则 x[i] 可以取出行子集，x[i,j] 可以取出行、列子集，其中 i 可以是整数下标，还可以是包含日期时间的字符串向量，日期时间类型的向量等。日期时间字符串的格式为 CCYY-MM-DD HH:MM:SS，而且可以省略后面的一部分，其含义是取出前面部分能匹配的所有时间点，这种设计使得取出某年、某月的数据变得十分方便。如

```
xts.ap <- as.xts(AirPassengers)
xts.ap["1949"]
```

```
##           [,1]
## 1月 1949   112
## 2月 1949   118
## 3月 1949   132
## 4月 1949   129
## 5月 1949   121
## 6月 1949   135
## 7月 1949   148
## 8月 1949   148
## 9月 1949   136
## 10月 1949  119
## 11月 1949  104
## 12月 1949  118
```

又比如，设 x 是某个股票的每分钟的数据，则 x["2018-01-18 08"] 取出 2018-1-18 8:00-9:00 之间的所有数据。

也可以用"from/to" 的格式指定一个日期时间范围，而且也不需要开始点和结束点恰好有数据，from 和 to 的时间精度也不需要与数据的实际采样频率相同。省略 from 则从头开始，省略 to 则直到末尾。如

```
xts.1["2018-01-10/2018-01-14"]
```

```
##            [,1]
## 2018-01-10    5
## 2018-01-11    5
## 2018-01-12    4
```

first(x, n) 和 last(x, n) 类似于 head(x, n) 和 last(x, n)，但是对 xts 对象 x，n 除了可以取正整数值以外，还允许用字符串指定时间长度，允许的单位包括 secs, seconds, mins, minutes, hours, days, weeks, months, quarters, years。比如，取出开头的三个月：

```
first(xts.ap, "3 months")
```

```
##          [,1]
## 1月 1949  112
## 2月 1949  118
## 3月 1949  132
```

字符串中取负值时表示扣除，如扣除开始的三个月：

```
start(first(xts.ap, "-3 months"))
```

```
## [1] "4月 1949"
```

取出的单位可以是比实际采样频率更大的时间单位，比如，分钟数据可以用 last(x, "1 day") 取出最后一天的的所有数据。但是，取出的单位不能比实际采样频率更小，比如，日数据不能用小时单位取出。

如果 x 是从其它类型转换得到的，即使取了 x 的子集或者进行了一些其他的操作，也可以尝试用 reclass() 函数恢复原来的类型。

1.6.4 支持的时间标签类型

xts 的时间下标也可以是一个日期或者日期时间类型的向量。支持的类型包括 Date, POSIXct, chron, yearmon, yearqtr, timeDate。可以沿用 zoo 的 index() 函数来读取或者修改 xts 类型的时间下标。

比如，对季度数据，如果原来的时间标签是 Date 或 POSIXct，可以用如下的方法修改时间标签的数据类型：

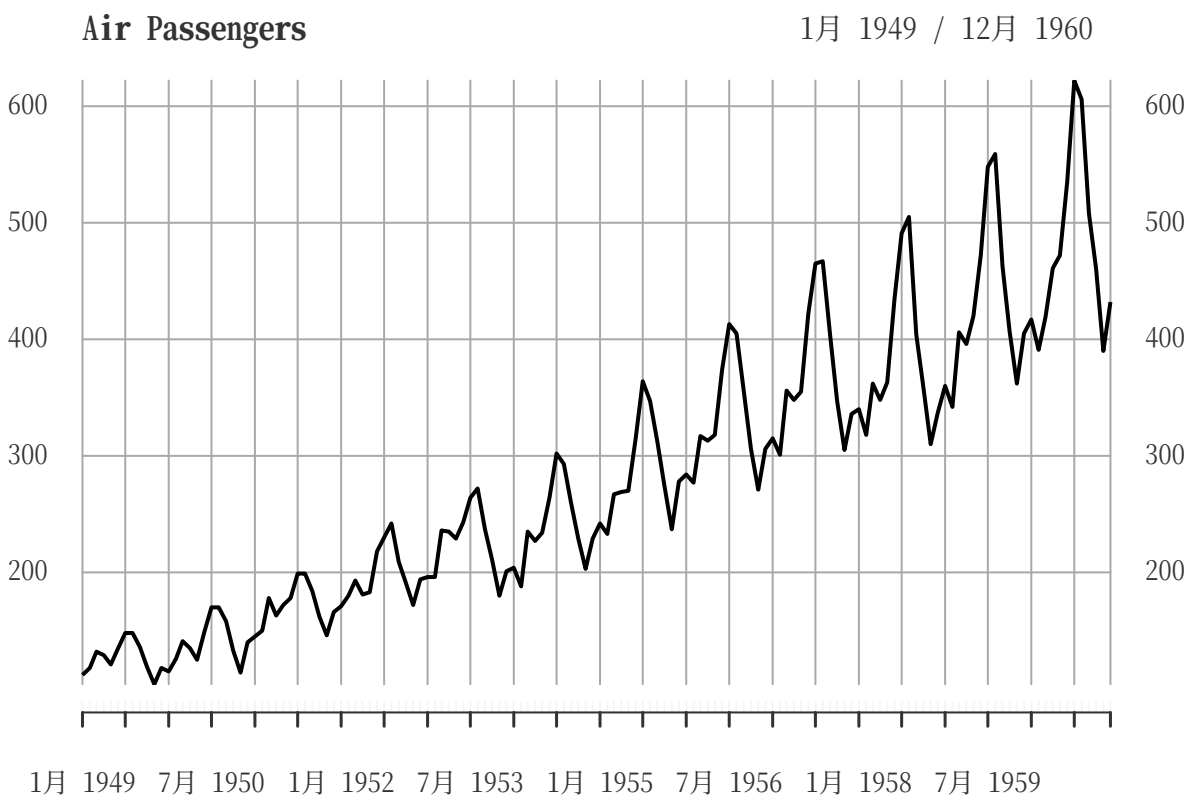
```
index(x) <- as.yearqtr(index(x))
```

as.yearmon() 可以转换为月度格式。

1.6.5 xts 类型数据的图形

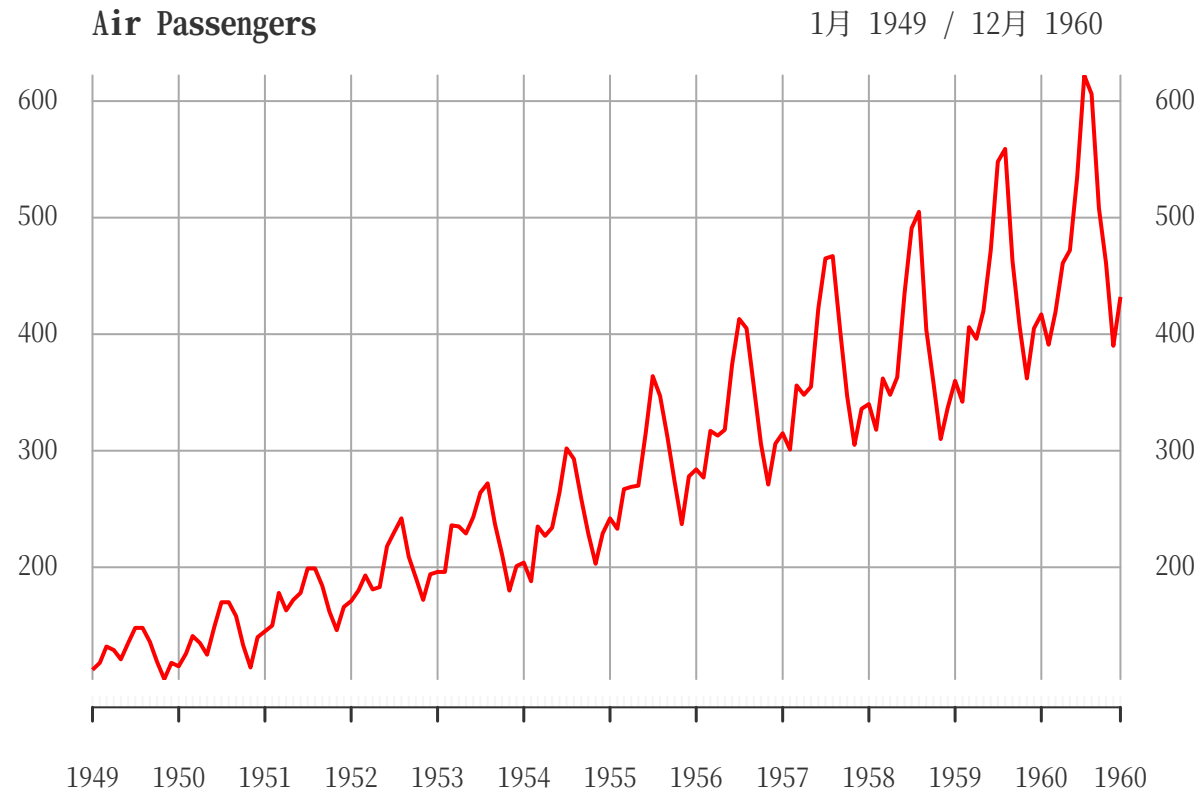
若 x 是 xts 时间序列，plot(x) 会自动选择适当的 x 轴刻度作图，这个函数是仿照 quantmod::chartSeries() 制作的。如

```
plot(xts.ap, main="Air Passengers")
```



可以用 `major.ticks` 和 `minor.ticks` 为 x 轴指定适当的粗刻度（有数值标注的）和细刻度，用 `grid.ticks.on` 指定 x 轴的格子线的频率，用 `format.labels="%Y"` 使得 x 轴刻度仅包含年份。如

```
plot(xts.ap, main="Air Passengers",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years", format.labels="%Y",
     col="red")
```



对 xts 对象的 plot() 函数还可以用 subset 选项指定一个子集范围，用 multi.panel=TRUE 要求多元时间序列的每个分量在单独的坐标系中作图，等等。

1.6.6 xts 的一些方便函数

1.6.6.1 求序列的时间区间

periodicity(x) 求 xts 对象 x 的时间区间。如

```
periodicity(xts.1)
```

```
## Daily periodicity from 2018-01-01 to 2018-01-12
```

1.6.6.2 求某种采样频率的分界点

endpoints(x, on) 给出按某种频率的分界点,频率包括"us"(微秒), "microseconds", "ms"(毫秒), "milliseconds", "secs", "seconds", "mins", "minutes", "hours", "days", "weeks", "months", "years"。如

```
endpoints(xts.ap, on="years")
```

```
##  [1]   0  12  24  36  48  60  72  84  96 108 120 132 144
```

结果是分组用的坐标，对输出 $t_0, t_1, \ldots, t_n$，$(t_{i-1}, t_i], i = 1, 2, \ldots, n$ 是结果分出的区间。

1.6.6.3 降频函数

对于 OHLC 形式的金融时间序列，即开盘价 (open)、最高价 (high)、最低价 (low)、收盘价 (close) 四个成分的时间序列，并且成分变量名也采用 `Open`, `High`, `Low`, `Close`，可以用 `to.period(x, period)` 将其降频成 `period` 指定的采样频率。比如，对分钟数据 `x`，可以用 `to.period(x, period="days")` 降频成日数据。降频后时间点一般采用每个周期内最后一个时间点，可以用 `indexAt` 选项指定其他的时间点选择。

如果 `x` 是分钟数据,`to.minutes3(x)`, `to.minutes5(x)`, `to.minutes10(x)`, `to.minutes15(x)`, `to.minutes30(x)`, `to.hourly(x)` 将 `x` 降频为 3 分钟到 60 分钟的数据。

`to.daily(x)` 将 `x` 降频为每天的数据，并在时间下标中删去时间部分。`to.weekly(x)` 将 `x` 降频为每周的数据，并在时间下标中删去时间部分，日期采用每周最后一天（周六）的日期。`to.monthly(x)` 将 `x` 降频到月度数据并将时间下标改为 yearmon 类型，`toquarterly(x)` 将 `x` 降频到季度数据并将时间下标改为 yearqtr 类型。`toyearly(x)` 将 `x` 降频到年度数据，日期采用每年有数据的最后一天的日期。

如果数据中原来有成交量，这些降频函数也会计算合并的成交量。

如果 `x` 是一元时间序列，`to.period(x, period)` 可以将其降频并得到 OHLC 四个序列，比如，从收盘价日数据可以降频得到周数据的开盘、最高、最低、收盘，但是并不是严格意义上的开盘、最高、最低。

1.6.6.4 周期操作函数

用 `period.apply(x, INDEX, FUN)` 可以用 `INDEX` 对时间序列 `x` 分组,对每组用 `FUN` 函数计算一个函数值。`INDEX` 常取为 `endpoints(x, on="...")`，给出某种分组周期。比如，计算每年航空旅客人数的月平均值：

```
period.apply(xts.ap,
             INDEX=endpoints(xts.ap, on="years"),
             FUN=mean)
```

```
##               [,1]
## 12月 1949 126.6667
## 12月 1950 139.6667
## 12月 1951 170.1667
## 12月 1952 197.0000
## 12月 1953 225.0000
## 12月 1954 238.9167
## 12月 1955 284.0000
## 12月 1956 328.2500
## 12月 1957 368.4167
## 12月 1958 381.0000
## 12月 1959 428.3333
## 12月 1960 476.1667
```

常用操作如求和等有专门的函数，如 `period.sum()`, `period.min()`, `period.max()`, `period.prod()`。专用函数运行效率更高。

常用的周期也有专门的函数，如 `apply.daily()`, `apply.weekly()`, `apply.monthly()`, `apply.quarterly()`, `apply.yearly()`。这些函数实际是调用 `period.apply()`。

1.7 tsibble 类型

tsibble 包提供了 tsibble 类型，这是遵从 tidyverse 理念的一种时间序列类型，基于 tibble 类型。tsibble 将时间标签作为 tibble 数据框的一列保存，称为 index，还可以有一个或多个 key 列，表示时间的 index 与 key 列共同唯一区分不同观测（行）。其他列可以是时间序列数据，还允许为列表类型。

tsibble 包提供了时间序列数据管理功能，并可以充分利用 tidyverse 的各种工具进行方便地处理计算。

1.7.1 从文本文件读入

以 m-ibmsp-2611.txt 为例，先读入为 tibble 数据框，再用 `as_tsibble()` 函数转换，用 `index=` 指定那一列是日期：

```
library(tsibble)
d <- read_table2(
  "m-ibmsp-2611.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
tsb <- as_tsibble(d, index="date")
head(tsb)
```

有时同一列中保存了不同类别的时间序列，比如，将上面的 ibm 和 sp 保存在同一列中，就需要用 key 指定区分不同类别的变量，如：

```
d1 <- d %>%
  pivot_longer(
    cols = c("ibm", "sp"),
    names_to="series",
    values_to="value")
tsb2 <- as_tsibble(
  d1, index="date", key="series")
head(tsb2)
```

1.7.2 管道和分组处理

tsibble 支持 tidyverse 的管道运算，支持 `group_by`，这时一般按 key 变量分组。还支持 `index_by`，可以按时间分组。

1.8 quantmod 包的功能

quantmod 包的目的是为量化投资者提供方便的原型开发测试工具，而不是提供新的统计方法。quantmod 包提供了金融时间序列数据的一些方便功能，比如从公开数据源载入数据，作股票行情图、时间序列图。

1.8.1 quantmod 包的数据下载功能

quantmod 包提供了一个 `getSymbols()` 函数，可以从多个公开数据源下载金融和经济数据并转换为 R 格式（主要是 xts 格式），数据源包括（有些在国内可能无法访问）：

- 雅虎金融（OHLC 数据）
- 圣路易斯联邦储备银行的联邦储备经济数据库（Federal Reserve Bank of St. Louis FRED®）(有 11000 个经济时间序列)
- Google Finance (OHLC 数据，国内连接困难)
- Oanda, The Currency Site (FX and Metals，国内连接困难)
- 本地的 MySQL 数据库
- R 二进制格式文件 (.RData 和.rda 文件)
- 逗号分隔文件 (.csv 文件)
- 待开发的数据源如 RODBC, Rbloomberg 等

例如，从雅虎金融下载微软公司的日数据，股票代码 MSFT：

```
MSFT <- getSymbols("MSFT", src="yahoo", auto.assign = FALSE)
```

```
head(MSFT)
```

```
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
## 2007-01-03     29.91     30.25    29.40      29.86    76935100      22.96563
## 2007-01-04     29.70     29.97    29.44      29.81    45774500      22.92717
## 2007-01-05     29.63     29.75    29.45      29.64    44607200      22.79642
## 2007-01-08     29.65     30.10    29.53      29.93    50220200      23.01947
## 2007-01-09     30.00     30.18    29.73      29.96    44636600      23.04254
## 2007-01-10     29.80     29.89    29.43      29.66    55017400      22.81180
```

```
str(MSFT)
```

```
## An 'xts' object on 2007-01-03/2018-01-23 containing:
##   Data: num [1:2784, 1:6] 29.9 29.7 29.6 29.6 30 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "MSFT.Open" "MSFT.High" "MSFT.Low" "MSFT.Close" ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
## List of 2
##  $ src    : chr "yahoo"
##  $ updated: POSIXct[1:1], format: "2018-01-24 23:02:15"
```

从雅虎金融获取美国 10 年债日数据：

```
TNX <- getSymbols("^TNX", auto.assign = FALSE) # CBOE 10 年国债日数据
```

```
head(TNX)
```

```
##            TNX.Open TNX.High TNX.Low TNX.Close TNX.Volume TNX.Adjusted
## 2007-01-03    4.658    4.692   4.636     4.664          0        4.664
## 2007-01-04    4.656    4.662   4.602     4.618          0        4.618
## 2007-01-05    4.587    4.700   4.583     4.646          0        4.646
## 2007-01-08    4.668    4.678   4.654     4.660          0        4.660
```

```
## 2007-01-09    4.660    4.670    4.644     4.656          0        4.656
## 2007-01-10    4.666    4.700    4.660     4.682          0        4.682
```

```
str(TNX)
```

```
## An 'xts' object on 2007-01-03/2018-01-24 containing:
##   Data: num [1:2785, 1:6] 4.66 4.66 4.59 4.67 4.66 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "TNX.Open" "TNX.High" "TNX.Low" "TNX.Close" ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
## List of 2
##  $ src    : chr "yahoo"
##  $ updated: POSIXct[1:1], format: "2018-01-25 12:37:47"
```

从 FRED 下载日元对美元汇率日数据：

```
DEXJPUS <- getSymbols("DEXJPUS", src="FRED", auto.assign=FALSE)
```

```
tail(DEXJPUS)
```

```
##            DEXJPUS
## 2018-01-12  111.23
## 2018-01-15      NA
## 2018-01-16  110.74
## 2018-01-17  110.81
## 2018-01-18  110.88
## 2018-01-19  110.56
```

从联邦经济数据库下载美国失业率月度数据：

```
UNRATE <- getSymbols("UNRATE", src="FRED", auto.assign=FALSE)
```

从 Oanda 下载铂金数据：

```
XPTUSD <- getSymbols("XPT/USD", sourse="Oanda", auto.assign=FALSE)
```

可以用选项 `from` 和 `to` 指定仅下载一段时间的数据。

雅虎金融还可以下载国内股市的数据，如下载深圳股票交易所上市的格力电器：

```
gl <- getSymbols("000651.SZ", src="yahoo", auto.assign = FALSE)
```

这个数据中 2011-05-06 和 2011-09-16 的数据缺失，另外有些日期的交易量为 0，其股价是沿用前一天收盘价。

下载兴业银行：

```
xy <- getSymbols("601166.SS", src="yahoo", auto.assign = FALSE)
```

下载国内股票数据要注意，结果有些不够可靠，比如下载上证综指（“000001.SS”）时，2020-02-26 下载数据中交易量为 0，实际为 469000（单位：千手）。下载的个股的交易量单位是一股，指数交易量单位是千手，报价是人民币。

另外，证券数据的日数据都是本地的日期，中国的 2 号等于美国的 1 号，在需要利用全球多个市场建立多变量模型时需要将日期统一到同一个时区。

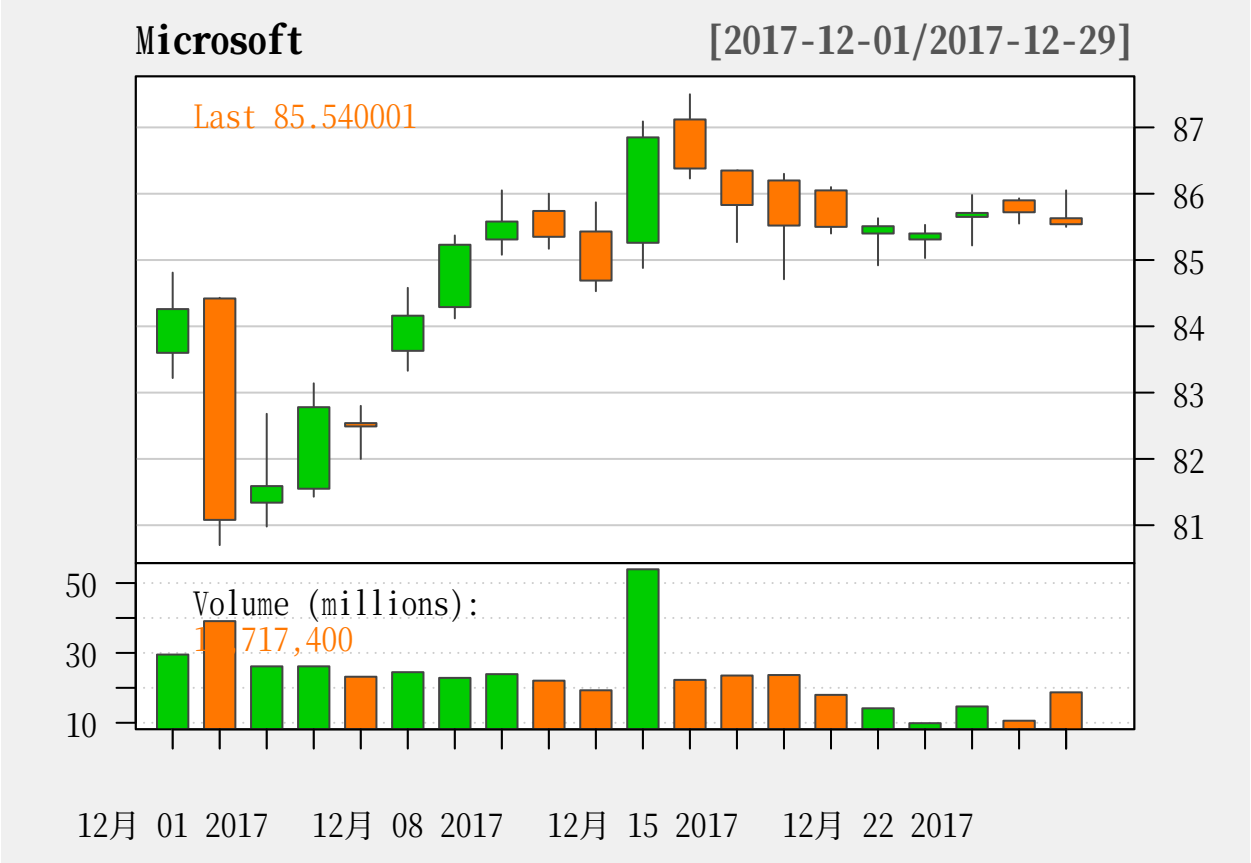
tseries 包的 `get.hist.quote()` 也提供了从网上数据源下载金融时间序列数据的功能。

1.8.2 quantmod 包的图形功能

`chartSeries()` 可以做 K 线图和曲线图等。

例：微软股票的 2017-12 的日数据 K 线图和成交量

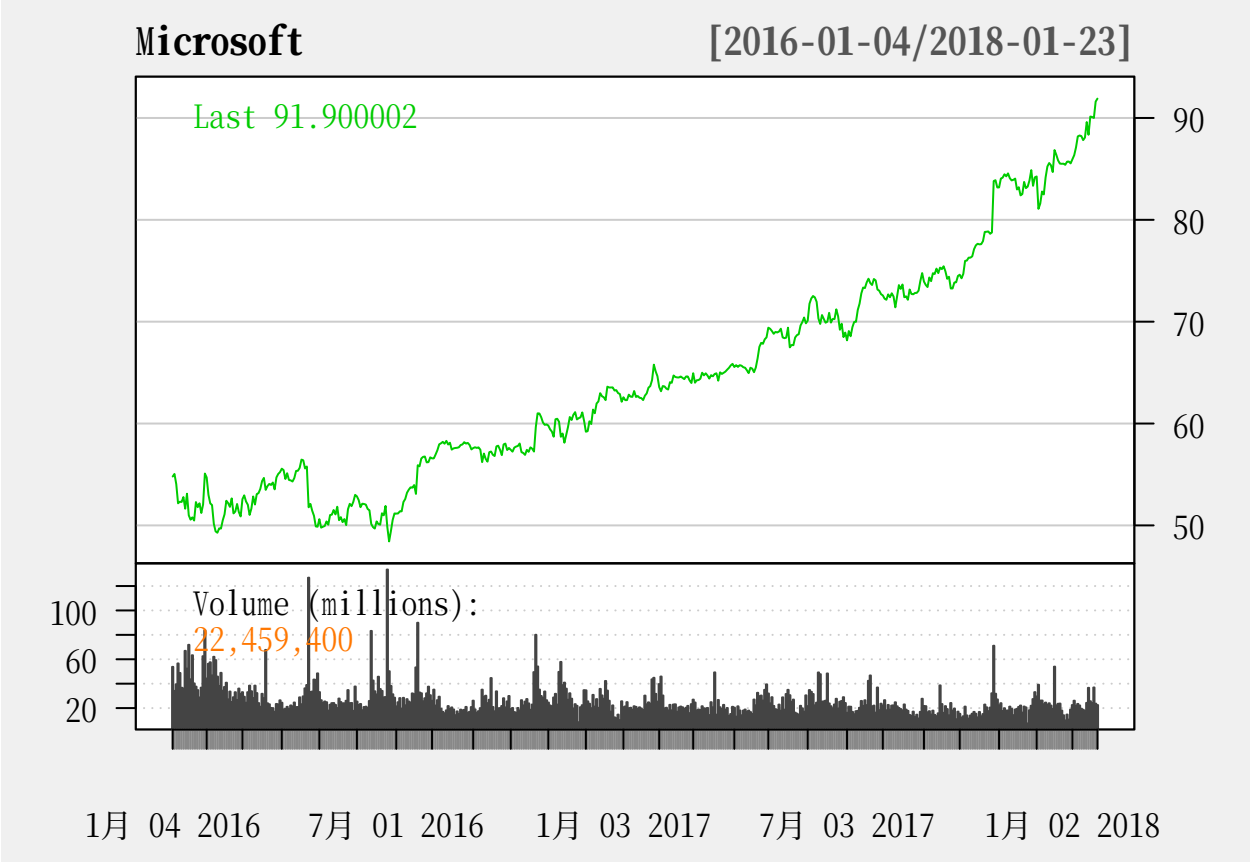
```
chartSeries(MSFT, theme="white", name="Microsoft",
            subset="2017-12-01/2017-12-31")
```



在 K 线图中，条形两端纵坐标为开盘价和收盘价，低开高收时为阳线，高开低收时为阴线；用单色显示时，阳线显示成空心条形，阴线显示成实心条形；用红色和绿色显示时，国内的习惯是红色阳线，绿色阴线，而西方的习惯是绿色阳线，红色阴线。如果最低、最高价不等于开盘价和收盘价，则向下画短线或者向上画短线。这样，每个 K 线单元都表示了 OHLC 四个值。用 `subset=` 选项指定一个序列子集。用 `name=` 指定出现在图形左上角的数据名称标注。

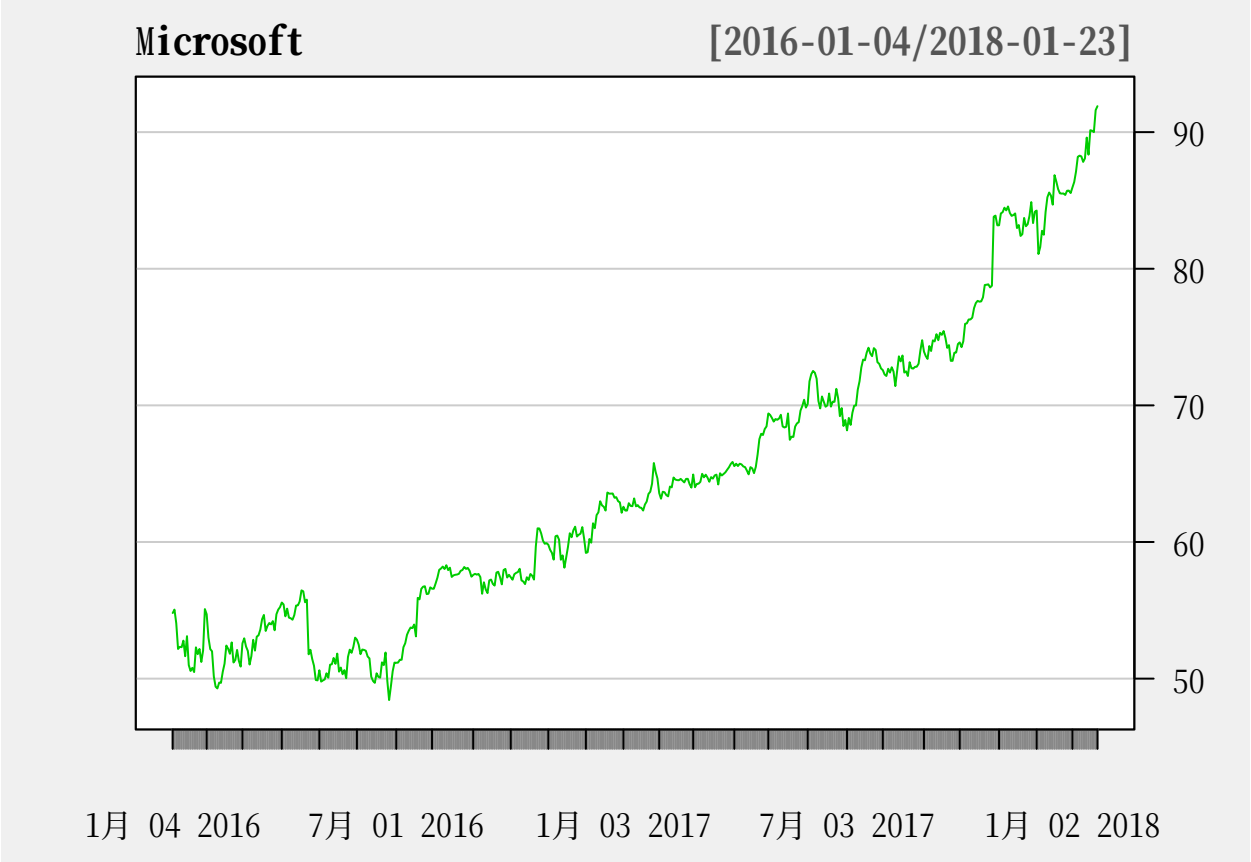
用 `type="line"` 指定画曲线图，对 OHLC 用收盘价数据，如微软股票最近三年的日数据的曲线图：

```
chartSeries(
  MSFT, type="line",
  subset="last 3 years",
  theme="white", name="Microsoft"
  )
```



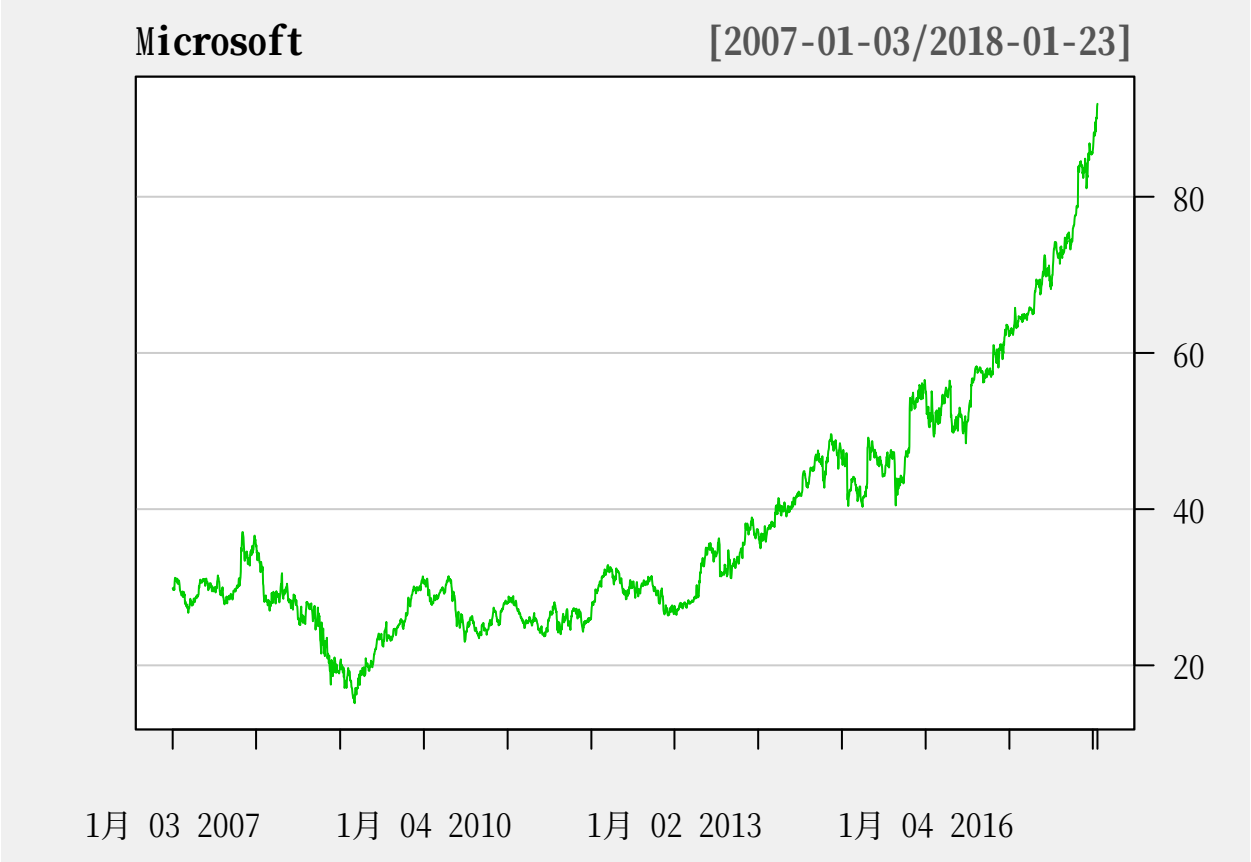
用 TA=NULL 选项去掉成交量图，如

```
chartSeries(
  MSFT, type="line", TA=NULL,
  subset="last 3 years",
  theme="white", name="Microsoft"
  )
```



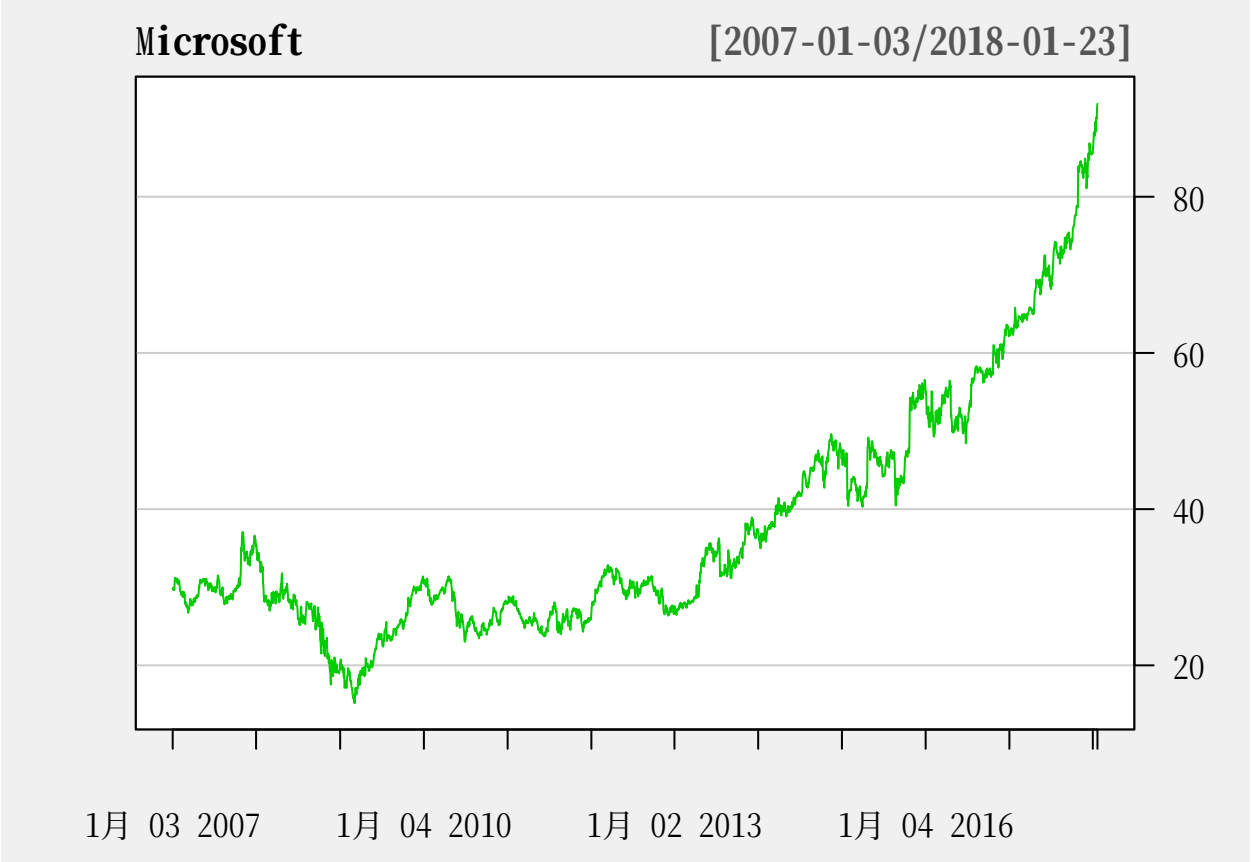
用 major.ticks 指定横轴的粗刻度的频率，用 minor.ticks 指定细刻度的有无，如

```
chartSeries(
  MSFT, type="line", TA=NULL,
  major.ticks="years", minor.ticks=FALSE,
  theme="white", name="Microsoft"
  )
```

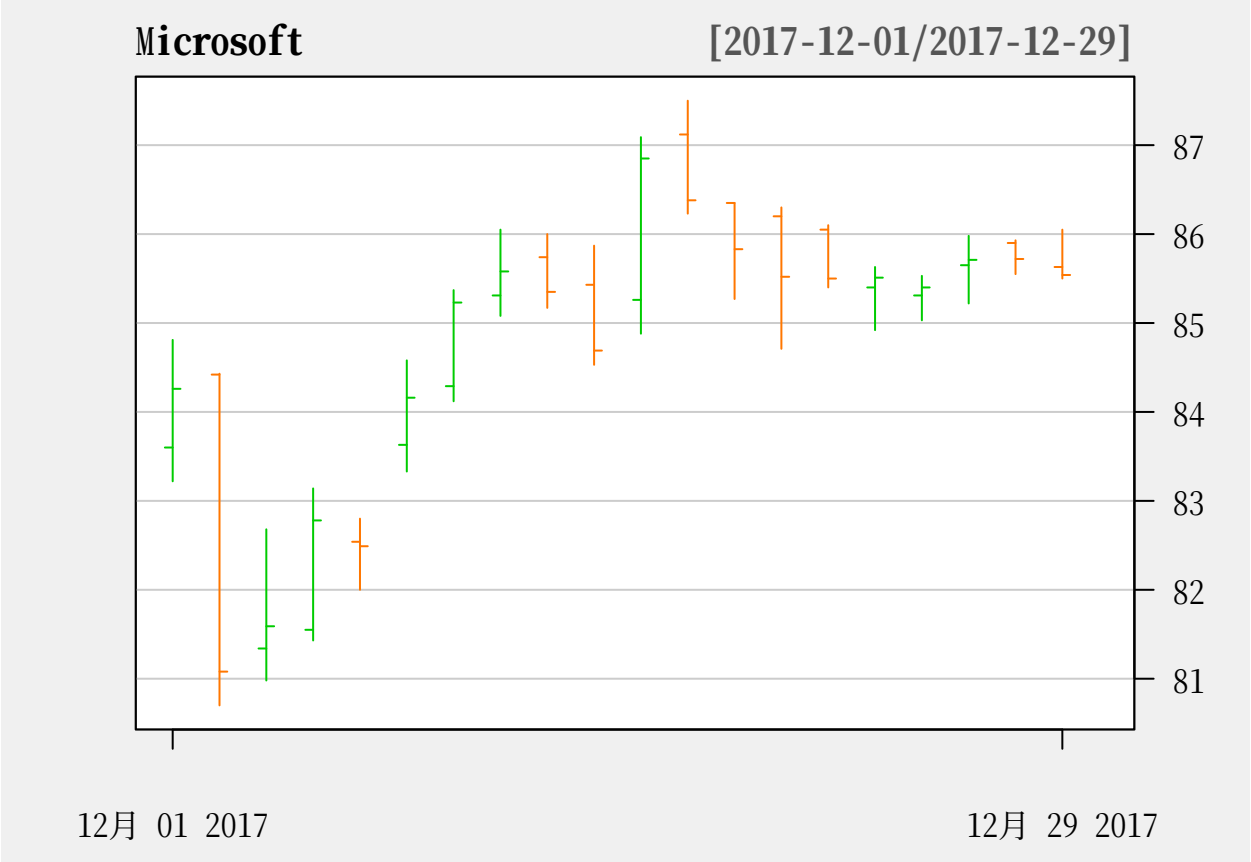


用 reChart() 可以修改前一幅图，如

```
chartSeries(
  MSFT, type="line", TA=NULL,
  major.ticks="years", minor.ticks=FALSE,
  theme="white", name="Microsoft"
  )
```




```
reChart(type="bars", subset="2017-12")
```



`type="bars"` 是另外一种 K 线图做法。K 线图可以用 `type=` 指定`"candles"`, `"matches"`, `"bars"` 做出，图形略有不同。

例：美国 10 年国债价格日数据 (CBOE Interest Rate 10 Year T No) 的收盘价曲线图：

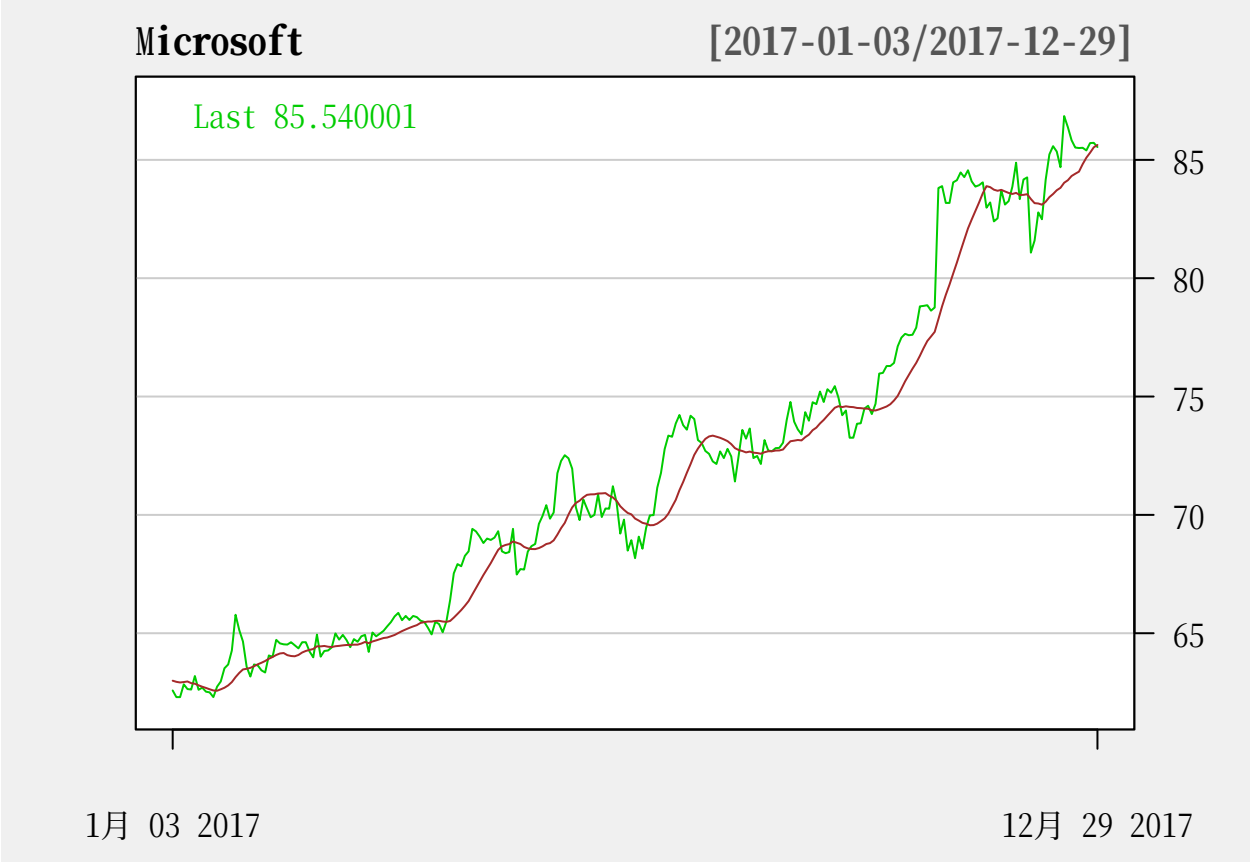
```
chartSeries(TNX, theme="white", TA=NULL, type="line")
```

美国失业率月度数据的曲线图：

```
chartSeries(TNX, theme="white", TA=NULL, type="line",
            major.ticks="years", minor.ticks=FALSE)
```

`quantmod::chartSeries()` 可以用 `TA=` 选项字符串指定叠加许多种技术分析指标曲线，见 `quantmod::addTA()` 的帮助。比如，`TA="addSMA(14)"` 可以添加 14 日均线：

```
chartSeries(
  MSFT, type="line", TA="addSMA(14)",
  subset="2017",
  major.ticks="years", minor.ticks=FALSE,
  theme="white", name="Microsoft"
  )
```



`quantmod::addTA(y)` 可以添加任何一个 xts 时间序列曲线到已有的 `chartSeries()` 图形下方的并列窗格中，用 `on=1` 选项叠加到原图中。

1.8.3 quantmod 包的计算功能

最常见的金融数据都是有时间、开盘价、最高、最低、收盘价、成交量等，这样的数据称为 OHLC 数据。有的没有成交量，有的仅有时间和价格。有的数据中包含 Adjusted 或者 Adjusted Closing Price，这是调整的收盘价，是针对拆分股、分红等对收盘价进行相应的调整。

quantmod 中提供了多个管理 OHLC 数据的方便函数。设 `x` 是一个 OHLC 数据：

- `Op(x)`, `Hi(x)`, `Lo(x)`, `Cl(x)`, `Vo(x)`, `Ad(x)` 分别提取出开盘价、最高、最低、收盘、成交量、调整收盘价。
- `is.OHLC(x)` 判断是否 OHLC 数据，`has.OHLC(x)` 判断是否包含 OHLC 数据，`has.xxx(x)` 判断是否有各个成分（`xxx` 分别取 `Op`, `Hi`, `Lo`, `Cl`, `Vo`, `Ad`）。
- `seriesHi(x)` 求每个分量的最大值, `seriesLo(x)` 求每个分量的最小值。
- 对每个时间点：
  - `OpCl(x)` 计算开盘价到收盘价的变化率；
  - `HiCl(x)` 计算最高价到收盘价的变化率；
  - `LoCl(x)` 计算最低价到收盘价的变化率；
  - `LoHi(x)` 计算最低价到最高价的变化率；
  - `OpHi(x)` 计算开盘价到最高价的变化率；
  - `OpLo(x)` 计算开盘价到最低价的变化率。

- `OpOp(x)` 计算前后两个时间点的开盘价的变化率, `ClCl(x)` 计算前后两个时间点的收盘价的变化率。也是简单收益率。

例：

```
last(MSFT, "5 days")
```

```
##            MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
## 2018-01-17     89.08     90.28    88.75      90.14    25621200         90.14
## 2018-01-18     89.80     90.67    89.66      90.10    24159700         90.10
## 2018-01-19     90.14     90.61    89.66      90.00    36875000         90.00
## 2018-01-22     90.00     91.62    89.74      91.61    23601600         91.61
## 2018-01-23     91.90     92.30    91.54      91.90    22459400         91.90
```

```
OpCl(last(MSFT, "5 days"))
```

```
##            OpCl.last(MSFT, "5 days")
## 2018-01-17               0.011899382
## 2018-01-18               0.003340701
## 2018-01-19              -0.001553128
## 2018-01-22               0.017888900
## 2018-01-23               0.000000000
```

```
ClCl(last(MSFT, "5 days"))
```

```
##            ClCl.last(MSFT, "5 days")
## 2018-01-17                        NA
## 2018-01-18             -0.0004437653
## 2018-01-19             -0.0011098557
## 2018-01-22              0.0178889000
## 2018-01-23              0.0031656042
```

`Lag(x)` 求序列滞后一期的结果，如

```
Lag(Cl(last(MSFT, "5 days")))
```

```
##            Lag.1
## 2018-01-17    NA
## 2018-01-18 90.14
## 2018-01-19 90.10
## 2018-01-22 90.00
## 2018-01-23 91.61
```

`Next(x)` 求序列超前一期的结果，如

```
Next(Cl(last(MSFT, "5 days")))
```

```
##             Next
## 2018-01-17 90.10
## 2018-01-18 90.00
```

```
## 2018-01-19 91.61
## 2018-01-22 91.90
## 2018-01-23    NA
```

Delt(x) 计算某个序列的简单收益率，如

```
Delt(Cl(last(MSFT, "5 days")))
```

```
##            Delt.1.arithmetic
## 2018-01-17                NA
## 2018-01-18     -0.0004437653
## 2018-01-19     -0.0011098557
## 2018-01-22      0.0178889000
## 2018-01-23      0.0031656042
```

Delta(x, k) 可以指定不同的时期 k 计算多期简单收益率。

periodReturn(x, period) 可以降频计算给定周期的收益率。默认计算简单收益率，用 type="log" 选项要求计算对数收益率。特别地，dailyReturn(x)，weeklyReturn(x)，monthlyReturn(x)，quartlyReturn(x)，annualReturn(x) 或者 yearlyReturn(x) 计算某个周期的收益率。allReturns(x) 计算日、周、月、季、年收益率。

1.9 用 BatchGetSymbols 包下载金融数据

quantmod 的 getSymbols 函数每次下载一个资产的价格序列，但是数据集中的变量名都带有资产名字，而且有时会有缺失值或交易量为 0 的日期。BatchGetSymbols 函数可以指定若干个资产的一个时间段同时从雅虎财经下载，并保存为数据框形式，不同资产的价格数据存放在相同的列中，用一个 ticker 变量区分资产，可以将交易量为零的价格设为缺失值或前一时期的值。可以缓存已经下载过的值，仅下载没有本地副本的值。

例如，下载最近 60 天：

```
library(BatchGetSymbols)
first.date <- Sys.Date() - 60
last.date <- Sys.Date()
freq.data <- "daily"
# set tickers
tickers <- c(
  "S&P500" = c,
  "Facebook" = "FB",
  " 上证综指" = "000001.SS",
  " 兴业银行" = "601166.SS",
  " 中兴通讯" = "000063.SZ"
  )
l.out <- BatchGetSymbols(
  tickers = tickers,
  first.date = first.date,
  last.date = last.date,
```

```
  freq.data = freq.data,
  cache.folder = file.path(
    tempdir(), 'BGS_Cache') ) # cache in tempdir()
## Running BatchGetSymbols for:
##    tickers =.Primitive("c"), FB, 000001.SS, 601166.SS, 000063.SZ
##    Downloading data for benchmark ticker
## ^GSPC | yahoo (1|1) | Found cache file
## .Primitive("c") | yahoo (1|5) | Not Cached - Error in download..
## FB | yahoo (2|5) | Found cache file - Got 100% of valid prices | OK!
## 000001.SS | yahoo (3|5) | Found cache file - Got 86% of valid prices | Good stuff!
## 601166.SS | yahoo (4|5) | Found cache file - Got 86% of valid prices | Got it!
## 000063.SZ | yahoo (5|5) | Not Cached | Saving cache - Got 86% of valid prices | You got it!
```

```
load("BatchGetSymbolsEx.Rdata")
```

结果是包含 df.control 和 df.tickers 的列表，df.control 是下载基本信息：

```
l.out$df.control
```

```
## # A tibble: 4 x 6
##   ticker    src   download.status total.obs perc.benchmark.da~ threshold.decisi~
##   <chr>     <chr> <chr>                <int>              <dbl> <chr>
## 1 FB        yahoo OK                      42              1     KEEP
## 2 000001.SS yahoo OK                      38              0.857 KEEP
## 3 601166.SS yahoo OK                      38              0.857 KEEP
## 4 000063.SZ yahoo OK                      38              0.857 KEEP
```

数据内容的变量：

```
names(l.out$df.tickers)
```

```
##  [1] "price.open"          "price.high"          "price.low"
##  [4] "price.close"         "volume"              "price.adjusted"
##  [7] "ref.date"            "ticker"              "ret.adjusted.prices"
## [10] "ret.closing.prices"
```

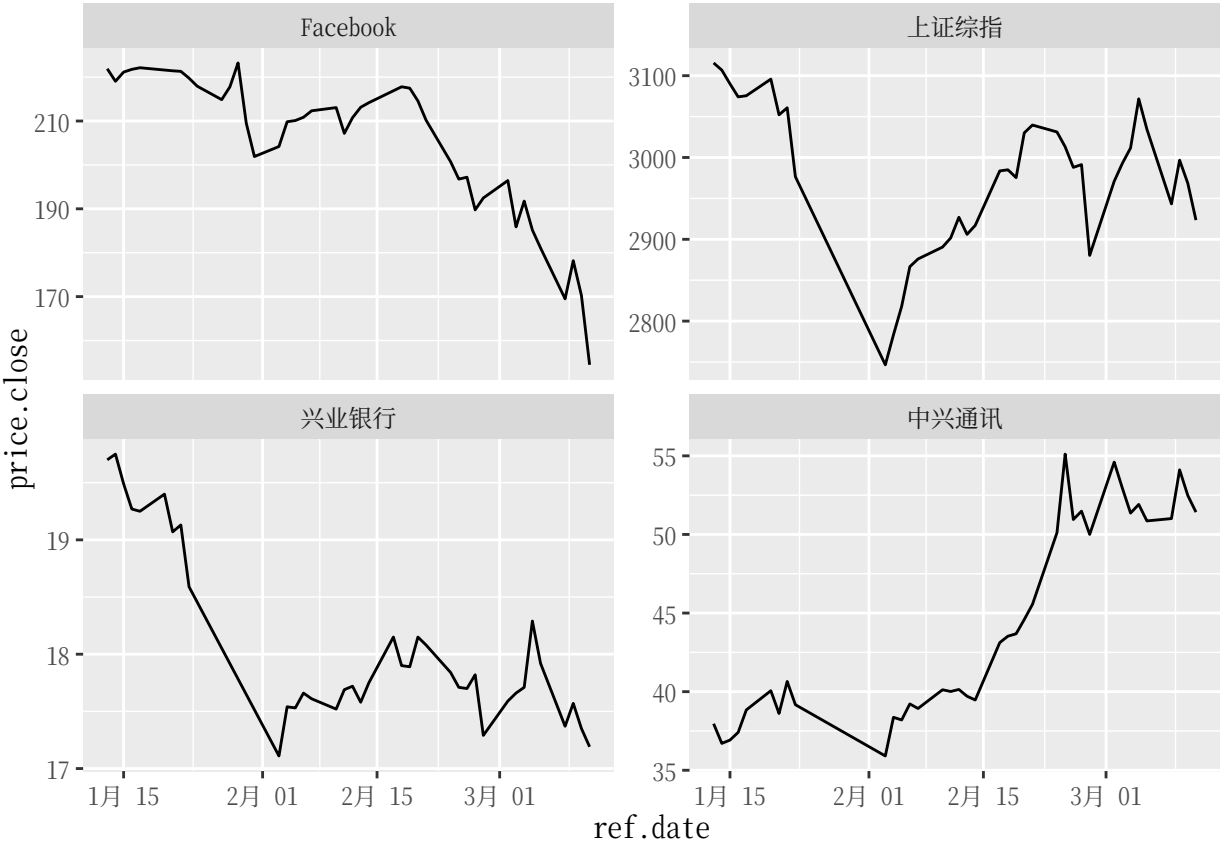
为了将代码转换成名称，写一个将元素值与元素名调换的函数：

```
nameMapInv <- function(x){
  y <- names(x)
  names(y) <- x
  y
}
```

绘图：

```
library(ggplot2, quietly = TRUE)
mapn <- nameMapInv(tickers)
```

```
ggplot(data=l.out$df.tickers, mapping=aes(
  x = ref.date, y = price.close)) +
  geom_line() +
  facet_wrap(~ mapn[ticker], scales = 'free_y')
```



1.10 tseries 包的功能

`get.hist.quote()` 可以下载多个金融数据的日数据。

1.11 R 软件的其它时间序列类型和功能

tseries 包中的 `get.hist.quote()` 函数可以从网上获取各主要股票的日交易数据。

fracdiff 库的 `fracdiff` 可以拟合分数 ARIMA 模型。

tseries 包中的 `garch` 可以拟合 GARCH 和 ARCH 模型。经常用 GARCH(1,1) 拟合收益数据。rugarch 包有更强大的 GARCH 功能，比如使用 t 分布残差。

fSeries 包中的 `garchFit` 可以拟合较复杂的包含 GARCH 的模型。

car 包中的 `durbin.watson` 函数可以对 lm 对象进行 Durbin-Watson 自相关检验。ts 包中的 `Box.test` 可以进行 Box-Pierce 或 Ljung-Box 白噪声检验。tseries 包中的 `adf.test` 可以进行 Dickey-Fuller 单位根检验。

1.11.1 声音数据分析

seewave 包可以分析、合成声音，进行时间、振幅、频率分析，估计声音之间的数量差别，生成新的声音用于回放。读入的声音文件可以是普通文本数值或 WAV 格式，WAV 格式是借助于 sound 包读入的。

读入一个 WAV 文件，用如

```
mus <- loadSample("song.wav")
```

这样读入一个声音对象。用 rate(mus) 求采样频率。用 cutw() 函数来切分出其中的片段，如

```
sec1 <- cutw(mus, from=0.500, to=0.5500, Sample=TRUE)
```

结果为一个矩阵。

为显示一个短片段的波形，用如

```
oscillo(sec1, f=f, title='0.05 秒的声音时间序列图')
```

其中 f 为采样频率。

用 spectro 函数显示动态谱峰位置，如

```
spectro(mus, f=f, flim=c(0,2.0), main=" 动态谱峰图")
```

用 saveSample 把声音片段保存为 WAV 格式，如

```
saveSample(mus, filename="somemus.wav")
```

Chapter 2

金融数据及其特征

课程采用蔡瑞胸（Ruey S. Tsay）的《金融数据分析导论：基于 R 语言》(Tsay, 2013)（An Introduction to Analysis of Financial Data with R）作为主要教材之一。这是第一章金融数据及其特征的授课笔记。

参考：

- (Tsay, 2013)
- (Tsay, 2010)
- (Tsay, 2014)
- (吴喜之 & 刘苗, 2018)
- (何书元, 2003)
- (Lutkepohl & Kratzig, 2004)
- (Cryer & Chan, 2008)
- (Christoffersen, 2003)

2.1 资产收益率

设 P_t 为某资产在 t 时刻的价格或净值，t 按照某种时间单位（如年）计数。

2.1.1 简单收益率

这里假定计算收益率的时间单位保持不变。常用年为单位。

单期简单毛收益率：

$$1 + R_t = \frac{P_t}{P_{t-1}}$$

单期简单净收益率，简单收益率：

$$R_t = \frac{P_t}{P_{t-1}} - 1 = \frac{P_t - P_{t-1}}{P_{t-1}}$$

k 期简单毛收益率：

$$1 + R_t[k] = \frac{P_t}{P_{t-k}} = \prod_{j=0}^{k-1}(1 + R_{t-j}).$$

k 期净收益率：

$$R_t[k] = \frac{P_t}{P_{t-k}} - 1 = \frac{P_t - P_{t-k}}{P_{t-k}}$$

例:

在实际中，收益率（利率）一般按年为单位，但是计息区间长度可以不等于一年。

设在银行借款金额为 P，银行标称年利率为 R(称为名义年利率)，但以半年计息一次的复利方式计算利息，一年后偿还本金和利息。如何计算本息？

解答:

年利率是名义利率，因为每半年计息一次，所以半年期的利率是 $R/2$，即时间单位为半年时，简单收益率是 $R/2$，这是金融行业计算利率的常见做法，即利率按年标注，但是不一定每一年计息一次，如果每年计息 n 次，则 $1/n$ 年的简单收益率为 R/n。

对于本例，半年后本息合计为 $P(1+\frac{R}{2})$，随后利息合并进入本金，后半年的本息合计为 $P(1+\frac{R}{2})^2$，所以一年后需要偿还的本息为 $P(1+\frac{R}{2})^2$。

这里，称 R 为**名义利率** (nominal interest rate)。因为每年计息两次且为复利，所以定义实际利率 (或译为有效利率，effective interest rate)R' 为 R' 使得

$$P(1+R') = \text{在一年后未偿还的本息和}$$

R' 是以一年为时间单位的简单收益率。在本例中实际年利率为

$$R' = (1+\frac{R}{2})^2 - 1 = 1 + R + \frac{1}{4}R^2 - 1 > R.$$

如果名义利率为 R，分 n 期计息，则实际利率 R' 为

$$R' = (1+\frac{R}{n})^n - 1.$$

名义利率和实际利率的关系的深入讨论参见2.1.2。

例 (九出十三归) 这是高利贷的一个典型范例。古代的当铺以物品为抵押放出贷款，但是号称放贷 10 元，仅给 9 元贷款，在放款时就已经收走了 1 元利息；贷款期 3 个月，每月利息 1 元，到期需要换 10 元本金和 3 元利息（实际是 9 元本金和 4 元利息），所以 3 个月的利率（简单收益率）是

$$\frac{13-9}{9} = \frac{4}{9} = 44.4\%,$$

折合年利率为

$$(1+4/9)^4 - 1 = 335\%.$$

注意我国现行民法典以大于 36% 为高利贷，超过 36% 的利息部分在法律上是无效的。

例子数据：苹果公司 2011-12-02 到 2011-12-09 的每日股票收盘价：

```
d.apple <- tibble(
  date=c(ymd("2011-12-02"),
         ymd("2011-12-05") + days(0:4)),
```

```
  price=c(389.70, 393.01, 390.95, 389.09, 390.66, 393.62)
)
knitr::kable(d.apple)
```

date	price
2011-12-02	389.70
2011-12-05	393.01
2011-12-06	390.95
2011-12-07	389.09
2011-12-08	390.66
2011-12-09	393.62

计算单期简单收益率的函数：

```
simple.return <- function(x){
  x <- as.vector(x)
  c(NA, diff(x) / x[1:(length(x)-1)])
}
```

计算简单收益率 (时间单位是交易日，无交易的日期忽略)，精确到百分数的小数点后 2 位：

```
d.apple <- d.apple %>%
  mutate(SR1=round(100*simple.return(price), 2))
knitr::kable(d.apple)
```

date	price	SR1
2011-12-02	389.70	NA
2011-12-05	393.01	0.85
2011-12-06	390.95	-0.52
2011-12-07	389.09	-0.48
2011-12-08	390.66	0.40
2011-12-09	393.62	0.76

从 2011-12-02 到 2011-12-09 是周五到周五，这一周时间的多期简单收益率为：

```
(d.apple[6,"price"] - d.apple[1,"price"]) /
  d.apple[1,"price"]
```

```
##        price
## 1 0.01005902
```

即 1.01%。

设某资产持有 k 年，按复利计算的（平均）年化收益率为

$$\left\{\prod_{j=0}^{k-1}(1+R_{t-j})\right\}^{1/k}-1=\exp\left\{\frac{1}{k}\sum_{j=0}^{k-1}\ln(1+R_{t-j})\right\}-1$$

当各 R_{t-j} 都很小时，可近似 $\ln(1+x)\approx x$, $e^x\approx 1+x$，k 期的平均年化复利近似为

$$\frac{1}{k}\sum_{j=0}^{k-1}R_{t-j}$$

但是当单期利率较大时此公式误差较大。

2.1.2 连续复利收益率

设某资产的的初始值为 C, 名义上的年利率为 r，但是在一年内分成 m 次付息，理论上每次付息 $C\frac{r}{m}$，最终的资产净值应为 $C+C\frac{r}{m}\times m=C(1+r)$；但是，因为提前付息，所以提前支付的利息也进入账户增值，从第二次付息开始，支付的利息就超过了 $C\frac{r}{m}$，使得一年后的净值要高于 $C(1+r)$。一年后的净值为

$$C\left(1+\frac{r}{m}\right)^m$$

当 $m\to\infty$ 时，由极限 $\lim_{x\to+\infty}(1+\frac{1}{x})^x=e$，可知

$$\lim_{m\to\infty}C\left(1+\frac{r}{m}\right)^m=\lim_{m\to\infty}C\left[\left(1+\frac{r}{m}\right)^{\frac{m}{r}}\right]^r=Ce^r.$$

这时 r 称为**连续复利**，它也对应某个时间单位（一般是年），$R=e^r-1$ 是连续复利 r 对应的实际利率, r 与 R 的关系为

$$R=e^r-1,\quad r=\ln(1+R)$$

例如，取年连续复利 $r=0.10$, $m=1,2,3,12,52,365,\infty$，初始资产 1 元，多次付息并计入账户，一年后净值分别为

```
m <- c(1, 2, 3, 12, 52, 365)
mlab <- c(paste(m), "Inf")
y <- c((1 + 0.10/m)^m, exp(0.10))
knitr::kable(tibble(`付息次数`=mlab, `年末净值`=y))
```

付息次数	年末净值
1	1.100000
2	1.102500
3	1.103370
12	1.104713
52	1.105065
365	1.105156
Inf	1.105171

所以按连续复利计算，连续复利的年利率 10%，等价于实际年利率为 10.52%。

按年连续复利的利率为 r，初始资产价格为 P_{t-n}，则 n 年后的资产净值为

$$P_t=P_{t-n}e^{nr}$$

其中 n 可以是整数也可以不是整数。

当 $n=1$ 时，

$$r_t=\ln\frac{P_t}{P_{t-1}}=\ln P_t-\ln P_{t-1}=\ln(1+R_t)$$

其中 R_t 是一年期的简单收益率，r_t 称为连续复合收益率或对数收益率，这就是上面的连续复利，但时间单位可以是任何时间单位。

当收益率较小时，有

$$r_t = \log(1 + R_t) \approx R_t, \quad R_t = e^{r_t} - 1 \approx r_t,$$

两者很接近。

对多期连续复合收益率，

$$r_t[k] = \ln P_t - \ln P_{t-k} = \ln(1 + R_t[k]) = \ln \prod_{j=0}^{k-1}(1 + R_{t-j}) = \sum_{j=0}^{k-1} r_{t-j}$$

所以连续复利比简单复利的公式简单，用加法代替了乘法。

例 (加倍法则):

如果将资金投入一个以每年计息一次，复利利率为 R 的账户中，多少年后资金变成原来的两倍？

解答:

设初始资金为 P，即求 n 使得

$$P(1 + R)^n \geq 2P$$

其中 R 较小时

$$(1 + R)^n = e^{n \ln(1+R)} \approx e^{nR}$$

令

$$e^{nR} \geq 2$$

得

$$n \approx \frac{\ln 2}{R} = \frac{0.693}{R} \approx \frac{0.7}{R}$$

当 $R = 0.07$ 时需要 10 年翻倍。

给定一个价格序列，计算对数收益率序列的函数如下：

```
log.return <- function(x){
  c(NA, diff(log(x)))
}
```

苹果公司股票的日对数收益率：

```
d.apple <- d.apple %>%
  mutate(LR1=round(100*log.return(price), 2))
knitr::kable(d.apple)
```

date	price	SR1	LR1
2011-12-02	389.70	NA	NA
2011-12-05	393.01	0.85	0.85
2011-12-06	390.95	-0.52	-0.53
2011-12-07	389.09	-0.48	-0.48
2011-12-08	390.66	0.40	0.40
2011-12-09	393.62	0.76	0.75

2.1.3 现值分析

假设可以以每期计息一次的方式，以每期 R 的名义利率借款和贷款。这时，第 i 期的期末支付 v 元的当前价值是多少？

应为 P 使得

$$P(1+R)^i = v$$

即 $P = v(1+R)^{-i}$。

计算现值可以用于比较不同的现金流，也可用于计算实际利率。

设在今后的 n 年，每年年末分别收到 $x_1, x_2, \ldots, x_n$ 元。名义年利率为 R。这个现金流的现值为

$$x_1(1+R)^{-1} + x_2(1+R)^{-2} + \cdots + x_n(1+R)^{-n} = \sum_{i=1}^{n} x_i(1+R)^{-i}.$$

2.1.3.1 贷款还款分析

设一个人抵押贷款金额为 L，需要在今后 n 个月的的每月月末偿还等额 A。贷款的月名义利率是 R，每月计息一次（复利）。

1. 已知 L, n, R，则 A 的值是多少？
2. 在第 j 月的月末已经完成支付后，还剩下多少贷款的本金？这一问题对于提前还款十分重要。
3. 在第 j 月的支付中，多少是利息的支付，多少是本金的支付？

解答 1：为了求得 A，只要按利率 R 计算各次支付的现值等于 L，即

$$L = \sum_{i=1}^{n} A(1+R)^{-i}$$

记 $\beta = \frac{1}{1+R}$，则

$$L = A\sum_{i=1}^{n} \beta^i = A\beta\frac{1-\beta^n}{1-\beta}$$

于是

$$A = L\frac{R}{1-(1+R)^{-n}}$$

例如，设 $L = 100$ 万元，$n = 120$ 个月 (10 年)，$R = 0.006$，则：

```
loan_pm <- function(L = 100, n = 120, R = 0.006){
  L * R / (1 - (1 + R)^(-n))
}
loan_pm()
```

```
## [1] 1.171419
```

```
loan_pm() * 120 - 100
```

```
## [1] 40.57025
```

```
((loan_pm() * 120) / 100)^(1/10) - 1
```

```
## [1] 0.03464018
```

每月需要还款 1.17 万元，总共还利息 40 万元。因为中间也还了一部分本金，所以将分期偿还的这些本息看成是在 10 年末尾一起偿还的，年利率只有 3.5%；但是，因为提前偿还了部分本息，所以持有贷款的期间实际借款没有 100 万元那么多，年利率就不止 3.5%。实际年利率是 7.2% 左右。

解答 2：设在第 i 月的月末偿还 A 后还剩的本金为 B_i，$i=0,1,\dots,n$，则 $B_0=L$, $B_n=0$。考虑 $\{B_j\}$ 之间的关系，如果在第 j 月的月末偿还后还有 B_j 债务，则在第 $j+1$ 的月末还有 $(1+R)B_j$ 债务，偿还 A 后还有 B_{j+1} 债务，即

$$B_{j+1}=(1+R)B_j-A,\ j=0,1,\dots,n-1$$

记 $\alpha=1+R$，递推有

$$\begin{aligned}B_1=&\alpha L-A\\B_2=&\alpha B_1-A=\alpha^2L-(1+\alpha)A\\B_3=&\alpha B_2-A=\alpha^3L-(1+\alpha+\alpha^2)A\end{aligned}$$

一般地，对 $j=1,2,\dots,n$ 有

$$B_j=\alpha^jL-A(1+\alpha+\cdots+\alpha^{j-1})=L\alpha^j\frac{\alpha^{n-j}-1}{\alpha^n-1}$$

解答 3：设 I_j 和 P_j 分别表示在第 j 月的月末支付的利息和本金的扣除额，$I_j+P_j=A$。第 $j-1$ 个月的月末偿还后还剩 B_{j-1} 债务，在第 j 个月需要偿还的利息为 $I_j=RB_{j-1}$，于是

$$I_j=L(\alpha-1)\alpha^{j-1}\frac{\alpha^{n-j+1}-1}{\alpha^n-1}.$$

而

$$P_j=A-I_j=L(\alpha-1)\frac{\alpha^{j-1}}{\alpha^n-1}.$$

可以看出本金的偿还额越到后期越多，而还款前期主要支付的是利息。

例如，设 $L=100$ 万元，$n=120$ 个月 (10 年)，$R=0.006$，则：

```
loan_demo <- function(L = 100, n = 120, R = 0.006){
  A <- L * R / (1 - (1 + R)^(-n))
  alpha <- 1 + R
  jvec <- 1:n
  Bvec <- L * alpha^jvec * (alpha^(n - jvec) - 1) / (alpha^n - 1)
  Ivec <- L * (alpha - 1) * (alpha^(n - jvec + 1) - 1) / (alpha^n - 1)
  Pvec <- A - Ivec
  data.frame(
    " 月份" = jvec,
    " 还款" = A,
    " 还本" = Pvec,
    " 利息" = Ivec,
    " 剩余" = Bvec
  )
}
tmp.d <- loan_demo()
knitr::kable(tmp.d[seq(12, 120, by=12),], digits=2, row.names=FALSE)
```

月份	还款	还本	利息	剩余
12	1.17	0.65	0.53	92.91
24	1.17	0.72	0.45	85.30
36	1.17	0.79	0.38	77.11
48	1.17	0.86	0.31	68.32
60	1.17	0.92	0.25	58.88
72	1.17	0.98	0.19	48.73
84	1.17	1.03	0.14	37.83
96	1.17	1.08	0.09	26.11
108	1.17	1.13	0.05	13.52
120	1.17	1.17	0.00	0.00

从计算例子看出，这种等额还款，前期大部分还款是交利息，对贷款者不太有利。

2.1.3.2 回报率

考虑一项投资，初始支出为 a，随后每期分别得到回报 $b_1, b_2, \ldots, b_n$，$b_i \geq 0$, $b_n > 0$。定义该投资每期的**回报率**R 为下述利率的值：R 使得在该利率下现金流序列的复利现值等于零，即

$$-a + \sum_{i=1}^{n} b_i (1+R)^{-i} = 0.$$

定义函数

$$P(R) = -a + \sum_{i=1}^{n} b_i (1+R)^{-i}, \ R \in [-1, \infty)$$

易见 $P(R)$ 为严格单调减的连续函数，$\lim_{R \to -1} = +\infty$, $\lim_{R \to \infty} = -a$，所以存在唯一的 R_* 使得 $P(R_*) = 0$，R_* 就是回报率。

由于 $P(0) = \sum b_i - a$，所以 $\sum b_i > a$ 时 $R_* > 0$，否则 $R_* \leq 0$。

当投资的每期回报率为 R_* 时，通常称这个投资有每期 $100R_*\%$ 的回报率。

求解 $P(R) = 0$ 的方程可以用二分法等数值方法。

如果现金流 $b_1, b_2, \ldots, b_n$ 表示贷出 a 元后每期的偿还额，则 R_* 是贷款人的每期回报率，也是借款人的实际每期利率。

如果现金流不满足 $b_i \geq 0$ 的条件，则函数 $P(R)$ 不再是单调函数，$P(R) = 0$ 不一定有唯一解，即使有唯一解也不能确定解左边和右边的回报性质。

例 2.1. 投资 100 元，分 5 年的回报分别为 $20, 22, 25, 30, 40$ 元。求回报率。

解答：

R 函数 `uniroot()` 可以用来求解一元连续函数的单个根。

```
x <- c(20, 22, 25, 30, 40)
fleft <- function(R) -100 + sum(x / (1+R)^(1:5))
uniroot(fleft, c(-0.99, 10.0), extendInt="downX")
```

```
## $root
## [1] 0.1016542
##
## $f.root
## [1] -0.001301806
##
## $iter
## [1] 12
##
## $init.it
## [1] NA
##
## $estim.prec
## [1] 6.103516e-05
```

程序中 `uniroot()` 的第二个自变量是根可能存在的区间，选项 "`extendInt="downX"`" 说明在初始区间两个端点函数值同号时，允许按照减函数向外扩大搜索区间。结果为 $R = 10.16\%$。

例 2.2. 设某民间借贷 10000 元，借期 1 年，采用 "三分利"，指月利率 3%，需要每月付息，不计复利，到期偿还本金与最后一个月的利息。大约相当于年利率 36%，用现值分析方法计算实际年利率。

解答：

这相当于初始投资 10000 万元，然后每个月获得 300 元回报，共 11 个月，最后一个月获得 10300 回报。设实际年利率（回报率）为 R，记 $a = 10000$, $b_i = 300$, $i = 1, 2, \ldots, 11$, $b_{12} = 10300$, 要解方程

$$-a + \sum_{k=1}^{12} b_i (1+R)^{-k} = 0.$$

用 R 程序计算：

```
x <- c(rep(300, 11), 10300)
fleft <- function(R) -10000 + sum(x / (1+R)^(1:12))
uniroot(fleft, c(-0.99, 10.0), extendInt="downX")
```

```
## $root
## [1] 0.02999609
##
## $f.root
## [1] 0.3894237
##
## $iter
## [1] 11
##
## $init.it
## [1] NA
```

```
##
## $estim.prec
## [1] 6.103516e-05
```

求得的实际年利率（回报率）为 38.94%。

2.1.4 资产组合收益率

设有 N 项资产，在 $t-1$ 时刻组合净值为

$$A_{p,t-1} = \sum_{j=1}^{N} A_{i,t-1} = A_{p,t-1} \sum_{j=1}^{N} w_i$$

其中 $w_i = A_{i,t-1}/A_{p,t-1}$ 是第 i 项资产的权重。于是

$$\begin{aligned} A_{p,t} =& \sum_{j=1}^{N} A_{i,t-1}(1+R_{i,t}) = A_{p,t-1} \sum_{j=1}^{n} w_i(1+R_{i,t}) \\ =& A_{p,t-1} \left(1 + \sum_{j=1}^{n} w_i R_{i,t}\right) \end{aligned}$$

所以资产组合的简单收益率为

$$R_{p,t} = \sum_{j=1}^{n} w_i R_{i,t}$$

注意其中 w_i 是第 $t-1$ 时刻的权重。如果继续计算 $R_{p,t+1}$，权重应该使用 t 时刻的权重。当然，如果资产比例变化不大，使用不变的 $\{w_i\}$ 近似也是可以的。

对于对数收益率没有如此简单的公式。当收益很小时近似有

$$r_{p,t} \approx \sum_{j=1}^{n} w_i r_{i,t}$$

2.1.5 红利支付与收益率

对价格 P_{t-1} 的某资产，如果在 $t-1$ 到 t 之间每单位还支付 D_t 红利，则到 t 时刻时，收益为 $P_t - P_{t-1} + D_t$，所以这时收益率应计算为

$$R_t = \frac{P_t - P_{t-1} + D_t}{P_{t-1}}, \quad r_t = \ln(P_t + D_t) - \ln P_{t-1}$$

2.1.6 超额收益率

$$Z_t = R_t - R_{0t}, \quad z_t = r_t - r_{0t}$$

其中 R_{0t} 和 r_{0t} 是某项参考资产的收益率，如美国短期国债收益率。

超额收益率被认为是如下的套利投资组合的盈利：对该资产持有多头头寸，对参照资产持有空头头寸，且初始净投资额为零。

多头金融头寸是指持有某资产。空头头寸是指从持有某资产的投资者手里借入某资产，然后卖出这些不属于自己的资产，在随后的规定日期卖空者有义务通过买入相同数额的该资产偿还借出者，不能偿还现金，当规定日期时该资产价格下跌时空头持有者可以获利。如果空头持有期间目标资产支付现金红利，空头持有者有义务支付红利给借出者。

2.1.7 关系小结

简单收益率 R_t 与连续复合收益率（对数收益率）r_t 的关系为

$$r_t = \ln(1 + R_t), \quad R_t = e^{r_t} - 1$$

简单的 k 期收益率为

$$R_t[k] = (1 + R_t)(1 + R_{t-1}) \dots (1 + R_{t-k+1}) - 1$$

k 期对数收益率为

$$r_t[k] = r_t + r_{t-1} + \cdots + r_{t-k+1}$$

如果连续复合年利率固定为 r，初始资产为 C，n 年后资产为 A，则

$$A = Ce^{rn}, \quad C = Ae^{-rn}$$

例：若某项资产月对数收益率为 4.46%，则简单收益率为

```
exp(0.0446) - 1
```

```
## [1] 0.04560953
```

即 4.56%。如果某项资产在一个季度的月对数收益率为 4.46%, -7.34%, 10.77%，则该季度的对数收益率为

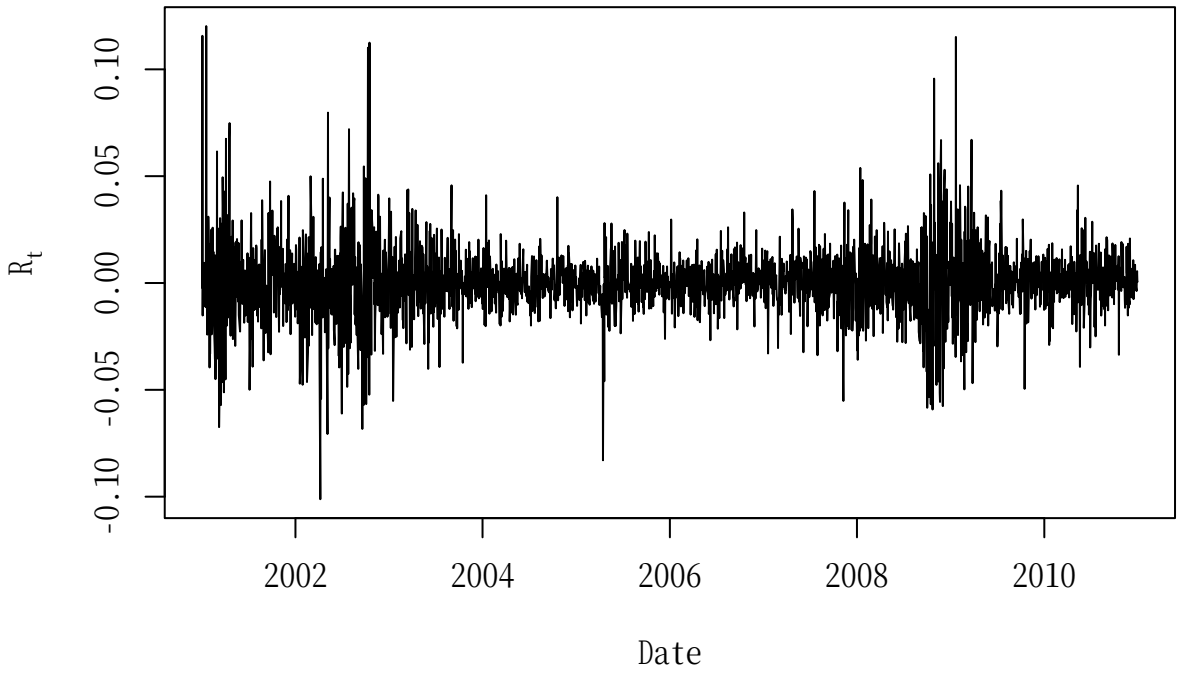
```
0.0446 + (-0.0734) + 0.1077
```

```
## [1] 0.0789
```

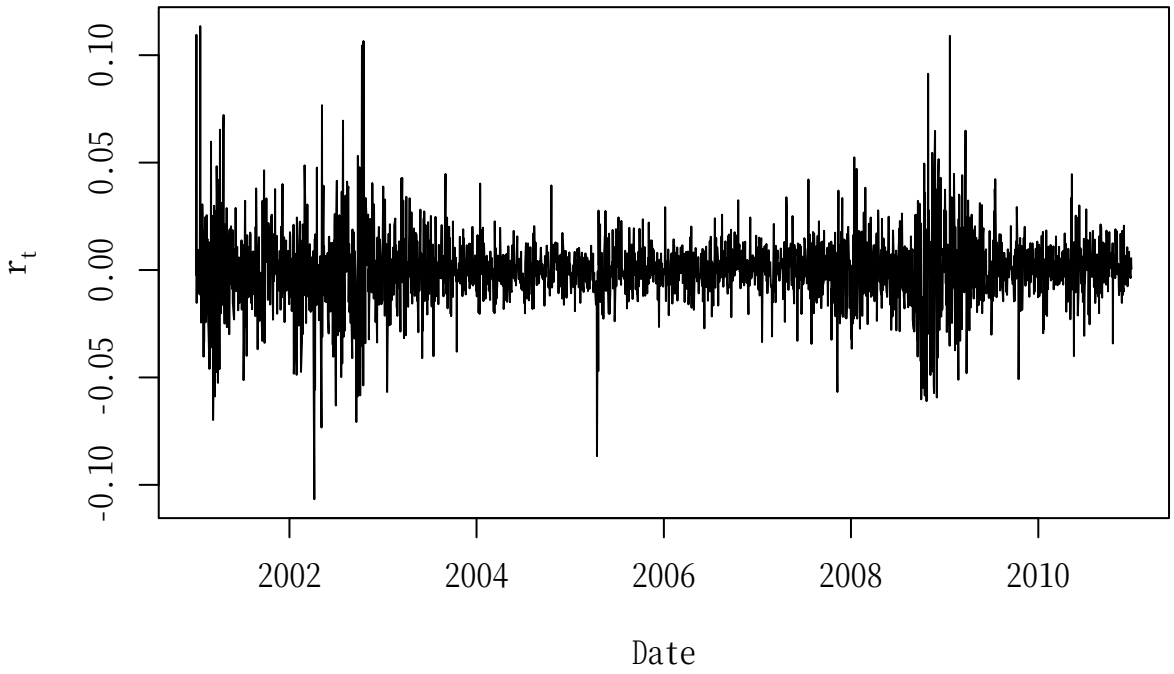
即 7.89%。

IBM 股票从 2001-1-2 到 2010-12-31 的日简单收益率和对数收益率的图形：

```
d.ibm <- read_table(
  "d-ibm-0110.txt",
  col_types=cols(date=col_date(format="%Y%m%d"),
    return=col_double()))
with(d.ibm,  plot(
  date, return, type="l",
  xlab="Date", ylab=expression(R[t])))
```




```
with(d.ibm, plot(
  date, log(1 + return), type="l",
  xlab="Date", ylab=expression(r[t])))
```



共有 2515 个观测。简单收益率与对数收益率的相关系数：

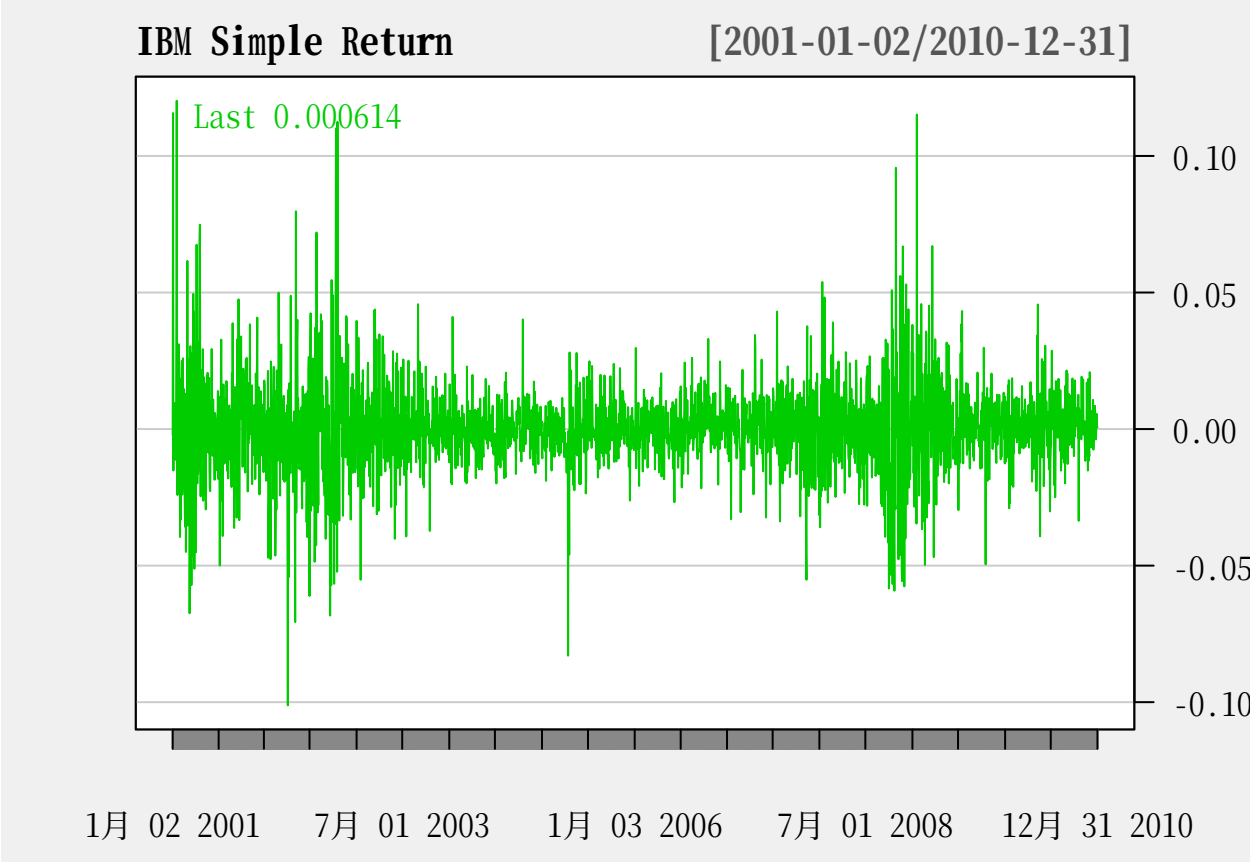
```
with(d.ibm, cor(return, log(1+return)))
```

```
## [1] 0.9997069
```

为 0.9997。当收益率绝对值较小时简单收益率与对数收益率近似相等。

上面将 IBM 股票数据读成了 tibble 数据框。也可以转换成 xts 类型时间序列，如

```
z.ibm <- xts(as.matrix(d.ibm[,-1]), d.ibm[["date"]])
quantmod::chartSeries(
  z.ibm, type="line", theme="white",
  xlab="Date", name="IBM Simple Return")
```



2.2 债券收益和价格

2.2.1 债券类型

投资者以市场价格买入债券，在到期日收回票面价格的现金。买入价格低于票面价格。有些债券还在持有期间定期派发利息，利息按照票面利率 (coupon payment) 和面额计算，比如，面值为 100 元，票面利率为 6%，如果每半年派发一次利息，则每次派息 $100 \times 0.06/2 = 3$ 元。有些债券不在中间派息，这样的债券称为零息债券。

2.2.2 当期收益率

当期收益率仅计算每年的表面收益，不考虑资金的时间成本。

$$\text{当期收益率} = \frac{\text{每年派息额}}{\text{买入价格}} \times 100\%$$

比如，票面价格 100 元，买入价格 90 元，票面利率为每年 5%，则持有一年的表面的年收益率为

$$(100 \times 0.05)/90 = 5.56\%$$

这没有考虑到期时票面价格与买入价格差值造成的收益。

2.2.3 到期收益率

对于零息债券，持有期间没有任何利息收入。如果购入价格为 P，面值为 F，持有 k 年到期，则收益率为

$$\left(\frac{F}{P}\right)^{1/k} - 1$$

这称为到期收益率 (Yield To Maturity, YTM)。

如果持有期间有派息，则到期收益率的计算很复杂，方法是求 y，令投资者在购入时的价格 P 等于持有期间、包括到期时的所有现金收入按照利率 y 贴水到购入时刻的现值。设面额为 F, 售价为 P，共派息 k 次，到期收益率为待定的 y，y 对应的时间区间为两次派息之间的时间区间，各次派息额为 $C_1, C_2, \ldots, C_k$，则方程为

$$P = \frac{C_1}{1+y} + \frac{C_2}{(1+y)^2} + \cdots + \frac{C_k}{(1+y)^k} + \frac{F}{(1+y)^k}$$

当每年派一次息时 y 就是年化的到期收益率。

如果票面利率（名义年利率）为 α，持有 n 年，每年派息 m 次，$k = nm$，每次派息 $F\alpha/m$, 设按 $1/m$ 年计算的到期收益率为 y，则方程为

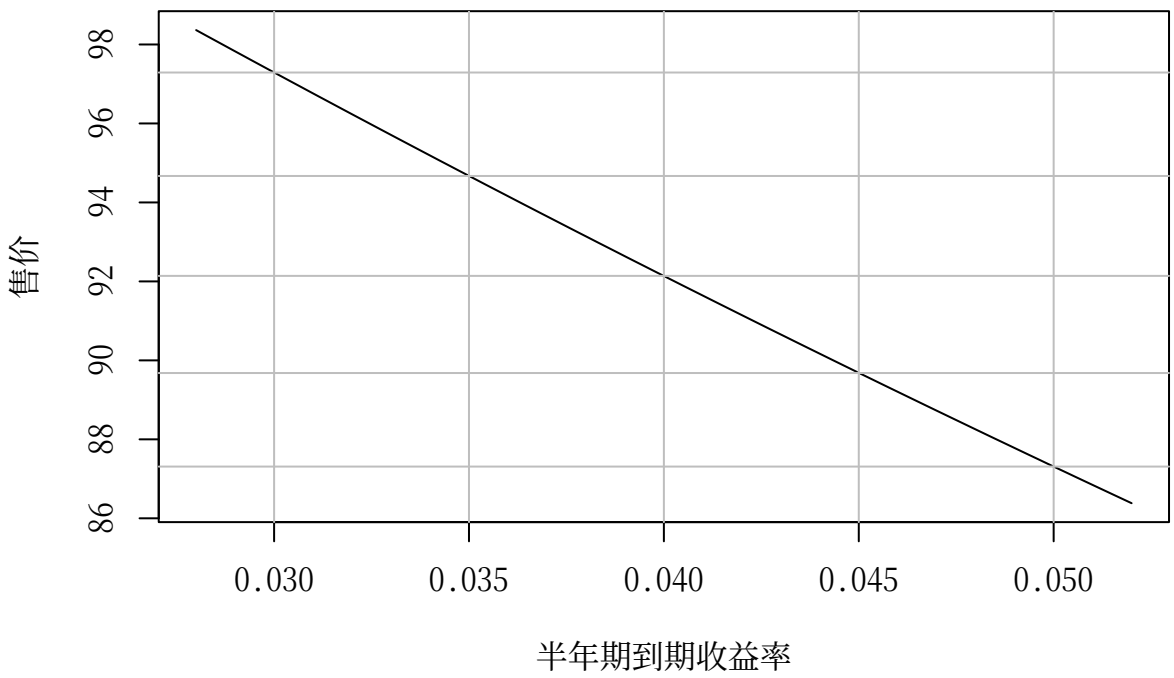
$$\begin{aligned} P =& F\frac{\alpha}{m}\left[\frac{1}{1+y} + \frac{1}{(1+y)^2} + \cdots + \frac{1}{(1+y)^k}\right] + \frac{F}{(1+y)^k} \\ =& \frac{F\alpha}{my}\left[1 - \frac{1}{(1+y)^k}\right] + \frac{F}{(1+y)^k} \end{aligned}$$

设某债券面值为 100 元，售价为 P, 持有时间为 $n = 3$ 年，每年发息 $m = 2$ 次，票面利率为 $\alpha = 0.05$，则一共有 $k = nm = 6$ 次发息，每次发息 $100 \times 0.05/m = 2.5$ 元。设到期收益率按半年期利率计算为 y，则售价与到期收益率 y（半年期）之间的关系为

$$\begin{aligned} P =& \frac{F\alpha}{my}\left[1 - \frac{1}{(1+y)^k}\right] + \frac{F}{(1+y)^k} \\ =& \frac{2.5}{y}\left[1 - (1+y)^{-6}\right] + 100(1+y)^{-6} \end{aligned}$$

作 P 对 y 的曲线图如下：

```
f <- function(y) 2.5/y*(1 - (1+y)^-6) + 100*(1+y)^-6
curve(f(x), 0.028, 0.052, xlab=" 半年期到期收益率", ylab=" 售价")
abline(v=c(0.030, 0.035, 0.040, 0.045, 0.050), col="gray")
pr <- c(97.29, 94.67, 92.14, 89.68, 87.31)
abline(h=pr, col="gray")
```



图中画出了半年期到期利率为 3%, 3.5%, 4%, 4.5%, 5% 时的售价。如果半年期到期利率为 3%，售价为 97.29 元。半年期到期利率 3% 转换为年化到期利率, 结果为

$$(1+0.03)^2-1=0.0609$$

教材 P.6 上的到期利率是按半年期利率计算的名义值，实际年化利率应为 $(1+y)^m-1$。

债券的售价 P 是到期收益率 y 的严格单调减函数 $P=g(y)$，在已知 F, n, m, α，P 的条件下用二分法或牛顿法求解 $g(y)=P$ 可以求出 y 的精确值。比如，用牛顿法求解到期收益率。设要解的方程为 $f(x)=0$，$(f(x)=g(x)-P)$ 牛顿法从某个 x_0 出发，用如下公式迭代：

$$x_{t+1}=x_t-f(x_t)/f'(x_t)$$

其中 $f'(x_t)$ 可以用数值微分方法估计为

$$f'(x_t)\approx\frac{f(x_t+\delta)-f(x_t-\delta)}{2\delta},$$

其中 $\delta=10^{-6}$。初值 x_0 取为

$$x_0=\frac{\alpha}{m}+\left(\frac{F}{P}\right)^{1/k}-1$$

```
ytm.newton <- function(
  P=97.29, F=100, alpha=0.05, n=3, m=2){
  k <- n*m
  f <- function(y){
    F*alpha/(m*y)*(1 - (1+y)^(-k)) + F*(1+y)^(-k) - P
```

```
  }

  x0 <- alpha/m + (F/P)^(1/k) - 1
  eps <- 1E-6 # 方程左边绝对值小于此值时迭代结束
  delta <- 1E-6 # 数值微分的步长
  max.iter <- 100
  iter <- 0
  repeat{
    iter <- iter + 1
    dfx <- (f(x0 + delta) - f(x0 - delta))/(2*delta)
    x0 <- x0 - f(x0)/dfx
    cat(iter, x0, f(x0), "\n")
    if(iter >= max.iter || abs(f(x0))<eps) break
  }

  x0 # 结果是 1/m 年的到期收益率
}
ytm.newton(P=97.29, F=100, alpha=0.05, n=3, m=2)
```

```
## 1 0.03000207 0.0003023458
## 2 0.03000264 5.709069e-10
```

```
## [1] 0.03000264
```

仅迭代了两次就求得了精度有四位有效数字的结果。注意，牛顿法求根需要比较准确的初值，如果初值取得太远，牛顿法可能不收敛。

2.2.4 美国政府债券

2.2.4.1 短期国债

Treasury Bills, T-Bills.

持有时间在 1 年及一年以下，零息债券，购入时以低于面值购入。常用期限：

- 28 天（4 周，一个月）
- 91 天（13 周，三个月）
- 182 天（26 周，半年）
- 364 天（52 周，一年）

其年化利率采用如下的简便公式计算：

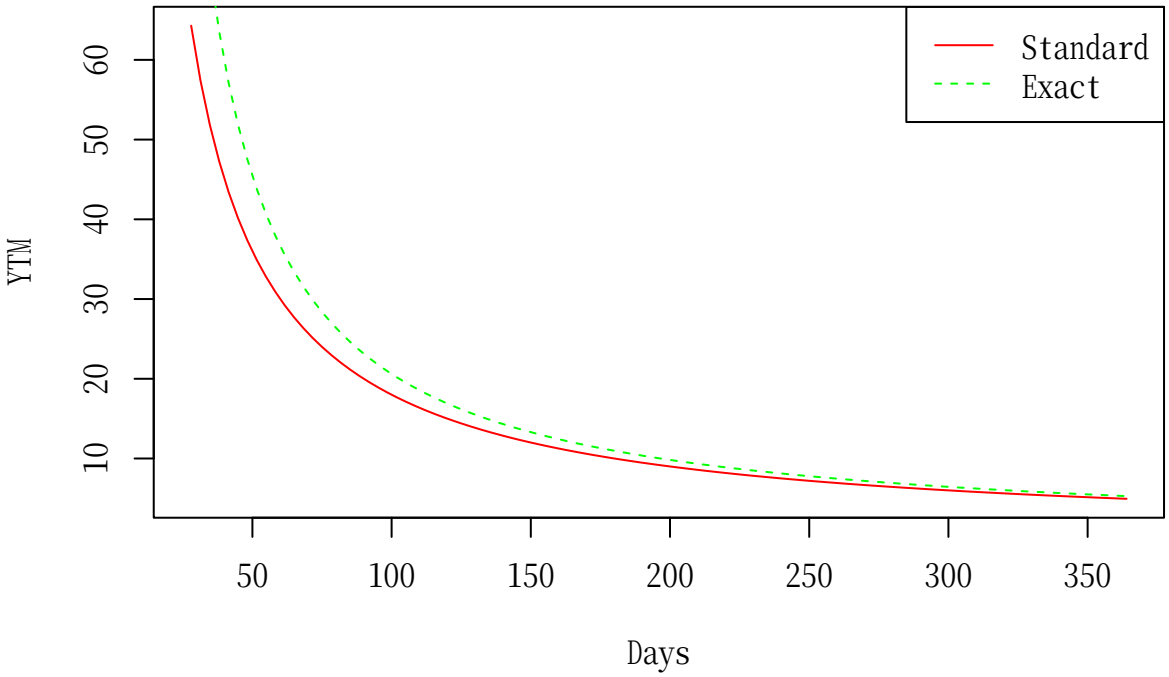
$$\text{折现收益率 }(\%) = \frac{F-P}{F} \times \frac{360}{\text{到期持有天数}} \times 100(\%)$$

更合理的到期收益率公式应为

$$(\%) = \left[\left(\frac{F}{P}\right)^{\frac{365.25}{\text{到期持有天数}}} - 1\right] \times 100(\%)$$

设 $F = 100$, $P = 95$, 到期持有天数在 28 到 364 之间变化时两种不同算法的变化曲线如下：

```
F <- 100
P <- 95
curve((F-P)/F*360/x*100, 28, 364, col="red",
      xlab="Days", ylab="YTM")
curve(((F/P)^(365.25/x)-1)*100, 28, 364,
      add=TRUE, lty=2, col="green")
legend("topright", lty=c(1,2), col=c("red", "green"),
       legend=c("Standard", "Exact"))
```



两条曲线有较大差距，这实际上是因为短期债券的售价太低导致利率过高引起的，在正常利率范围内两者应该差距不大。

设年化利率规定在 5%，面值 $F = 100$，按标准公式计算的售价为

$$P = F\left(1 - R\frac{\text{天数}}{360}\right) = 100\left(1 - 0.05\frac{\text{天数}}{360}\right)$$

按标准公式计算的折现率为 5%，计算相应的精确到期收益率：

```
days <- c(28, 91, 182, 364)
prices <- 100*(1 - 0.05*days/360)
rates <- ((100/prices)^(365.25/days)-1)*100
rbind(days, standard=5, exact=round(rates,2))
```

```
##             [,1]  [,2]  [,3]  [,4]
```

```
## days      28.00 91.00 182.00 364.00
## standard   5.00  5.00   5.00   5.00
## exact      5.21  5.24   5.27   5.34
```

精确利率要高。在标准公式中计算利率用了 $(F-P)/F$，理论上应该用 $(F-P)/P$，所以标准公式低估了利率。因为短期国债并不能很容易地自动转存，所以低估是有一定合理性的。

2.2.4.2 中期国债

Treasury Notes, T Notes.

在 1 年到 10 年内到期，每 6 个月支付一次利息，面值 1000 元。在二级市场上以面值的特殊百分数报价，百分数的小数部分 1/32 为单位，比如报价 95:08，即报价 $1000 \times 95\frac{8}{32} \div 100 = 952.5$。10 年期美国国债是报价频率最高、安全性最高的国债。

2.2.4.3 长期国债

Treasury Bonds, T-Bonds.

30 年期的国债，每 6 个月付息一次。

R 扩展包 quantmod 包含了金融数据建模的功能，比如可以从雅虎、谷歌财经等公开数据源下载多个经济和金融事件序列数据。

```
library(quantmod)
TNX <- getSymbols("^TNX", auto.assign = FALSE) # CBOE 10 年期国债数据
```

```
head(TNX)
```

```
##            TNX.Open TNX.High TNX.Low TNX.Close TNX.Volume TNX.Adjusted
## 2007-01-03    4.658    4.692   4.636     4.664          0        4.664
## 2007-01-04    4.656    4.662   4.602     4.618          0        4.618
## 2007-01-05    4.587    4.700   4.583     4.646          0        4.646
## 2007-01-08    4.668    4.678   4.654     4.660          0        4.660
## 2007-01-09    4.660    4.670   4.644     4.656          0        4.656
## 2007-01-10    4.666    4.700   4.660     4.682          0        4.682
```

```
chartSeries(TNX, theme="white", TA=NULL, type="line") # 仅收盘价，不显示交易量
```

2.3 隐含波动率

- 股票期权 (stock option)
- 看涨期权 (call option)
- 看跌期权 (put option)
- 执行价格（strike price）：在未来规定的期间有权利以执行价格买入或者卖出一定数量的股票。
- 到期日（time to maturity）
- 欧式期权：只有在到期日才能行权
- 美式期权：在到期日及之前都可以行权
- CBOE, 美国芝加哥期权交易所
- 行权时根据给持有人的现金流的正、负、零分为价内期权（in-the-money），价外期权（out-of-the-money），平价期权（at-the-money）。当然，只有价内期权会实际行权。

期权价格影响因素：

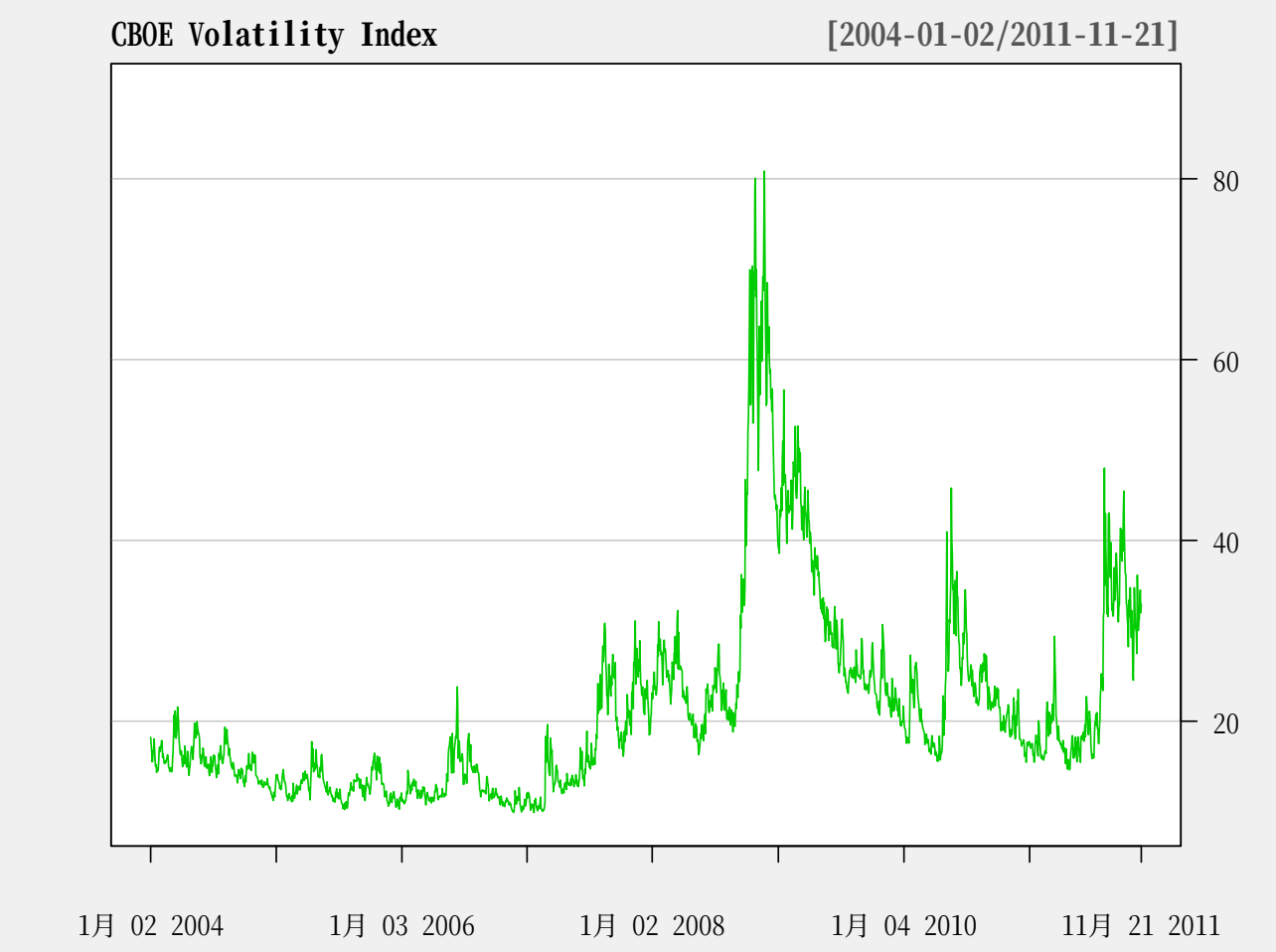
- 执行价格
- 无风险利率
- 当前股价
- 股票的波动率

Black-Scholes 模型：在假设股票价格服从几何布朗运动的条件下，给出了期权价格的解析解，其中包含不可观测的波动率。波动率（volatility）是股票价格的条件标准差。可以从股价、期权价格以及 BS 模型求解出波动率，这样得到的波动率称为隐含波动率（implied volatility）。

VIX: CBOE 的波动率指数，1993 年提出，2003 年修订。

蔡教授提供的文本文件 `d-vix0411.txt` 中，前面几列用空格分隔，后面几列用制表符分隔，这使得程序读入这样的文件很困难，同学一定不要制造这样的文件。为了可以读取，将所有行用 `readr::read_file()` 读入，用字符串函数 `gsub()` 将所有制表符替换成空格，然后再从修改后的字符串读取：

```
tmp.s <- read_file("d-vix0411.txt")
tmp.s <- gsub("\t", " ", tmp.s, fixed=TRUE)
d.vix <- read_table(tmp.s,
  col_types=cols(.default = col_double()))
rm(tmp.s)
vix <- xts(d.vix[4:7],
  make_date(d.vix[["year"]], d.vix[["mon"]], d.vix[["day"]]))
chartSeries(
  vix, type="line", TA=NULL,
  major.ticks="years", minor.ticks=FALSE,
  theme="white", name="CBOE Volatility Index"
  )
```



2.4 收益率分布特性的探索性分析

2.4.1 苹果公司股票日数据

苹果公司日数据下载：

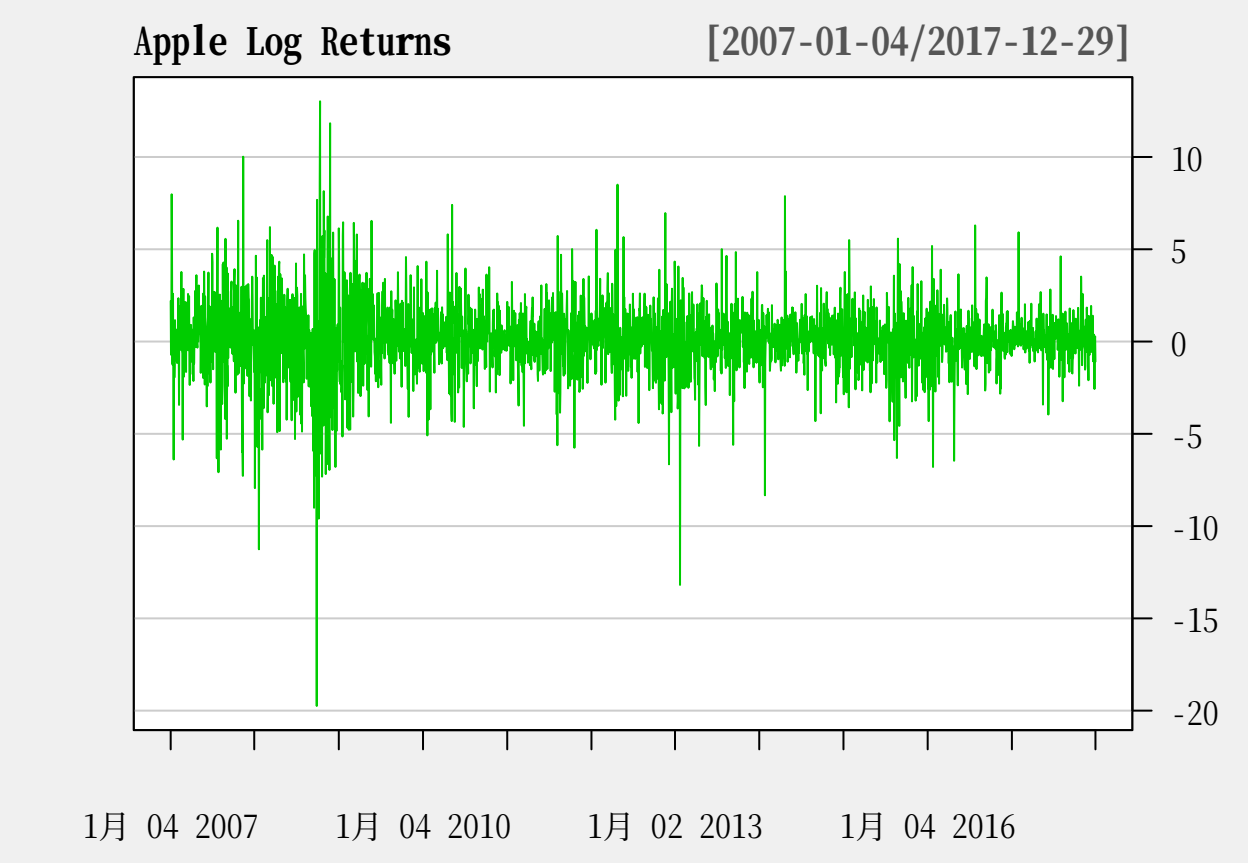
```
library(quantmod)
AAPL <- getSymbols("AAPL", src="yahoo", auto.assign=FALSE)
logret.AAPL <- diff(log(AAPL$AAPL.Adjusted))*100 # 对数收益率
```

计算对数收益率：

```
logret.AAPL <- diff(log(AAPL$AAPL.Adjusted))*100 # 对数收益率
```

对数收益率时间序列 2007-2017 的曲线图：

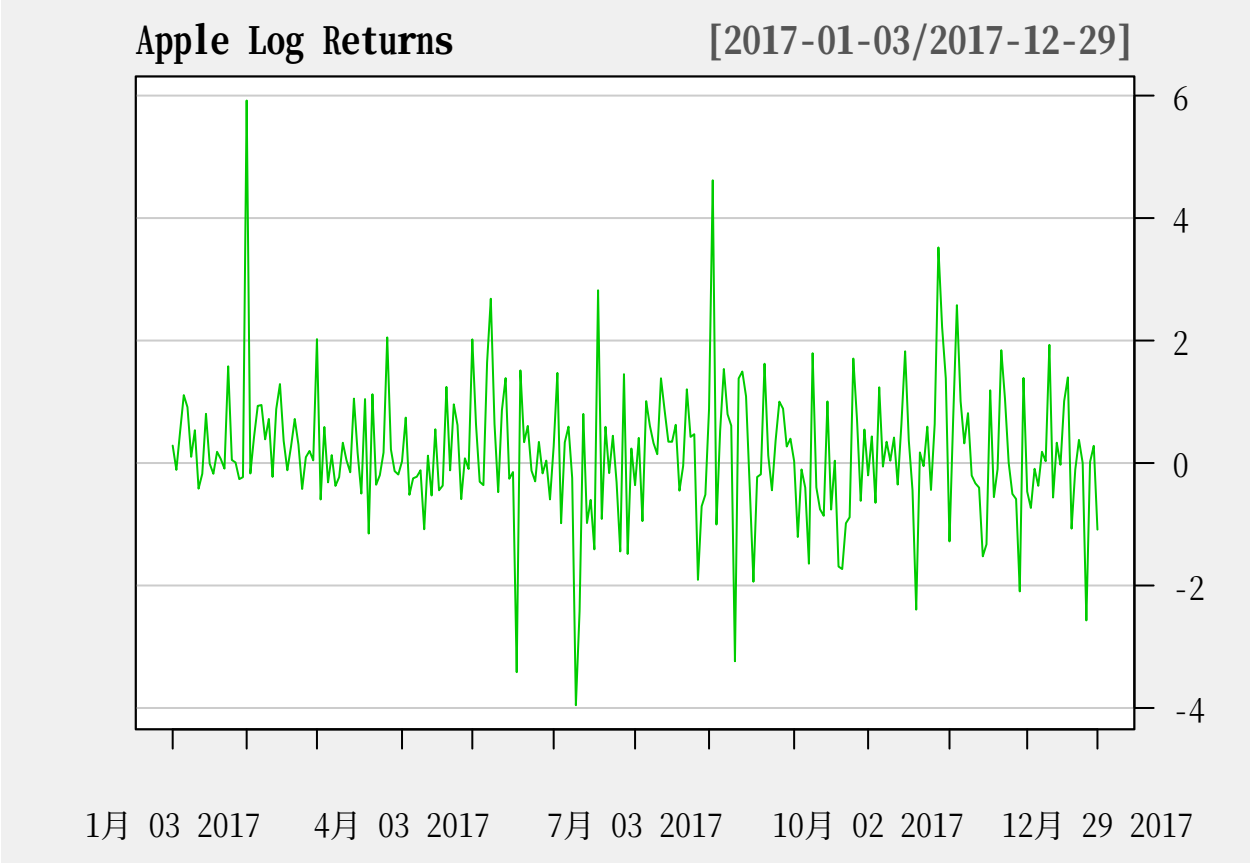
```
chartSeries(
  logret.AAPL, type="l", TA=NULL,
  subset="2007/2017",
  name="Apple Log Returns",
  theme="white", major.ticks="years", minor.ticks=FALSE)
```



仅 2017 年的对数收益率曲线图：

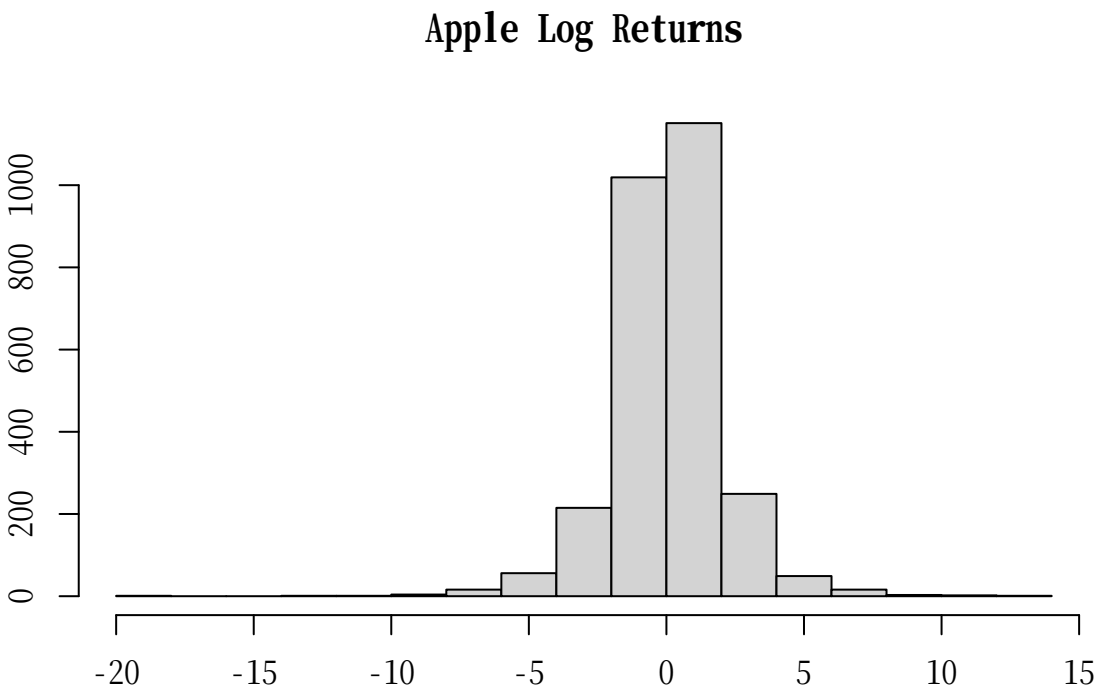
```
chartSeries(
  logret.AAPL, type="l", TA=NULL,
```

```
  subset="2017/2017",
  name="Apple Log Returns",
  theme="white", major.ticks="months", minor.ticks=FALSE)
```



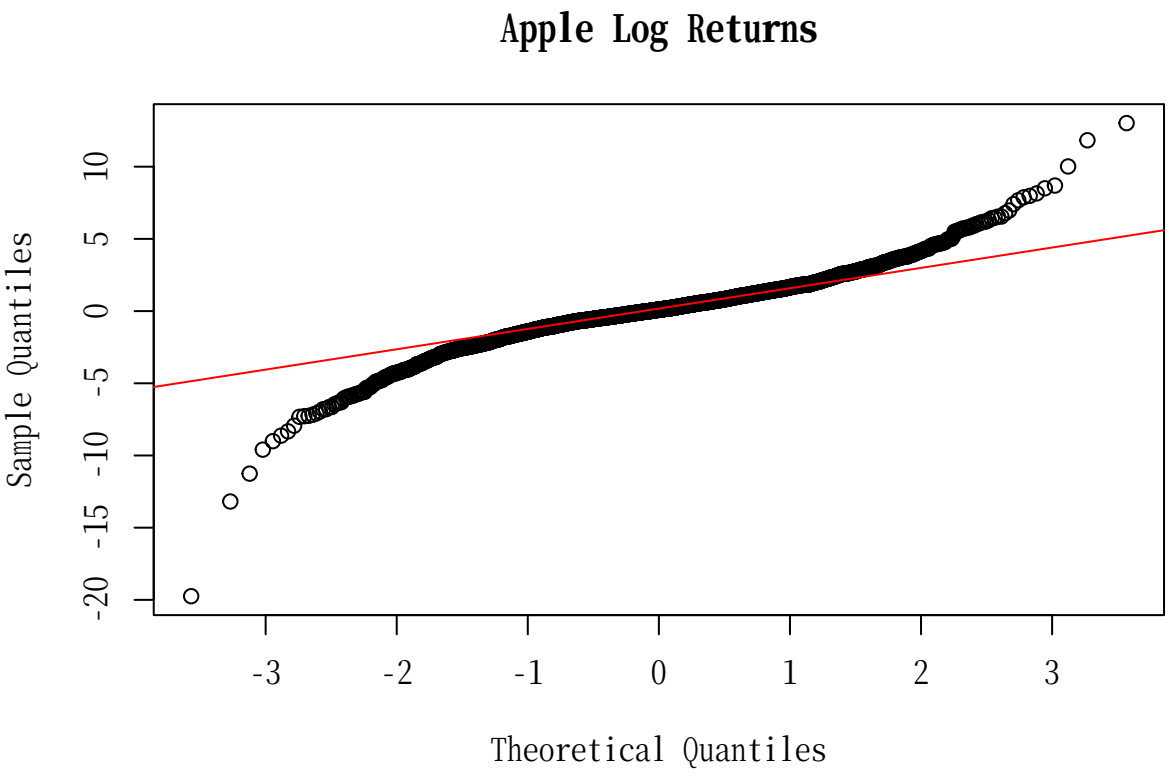
对数收益率的直方图：

```
x <- coredata(logret.AAPL)
hist(x, main="Apple Log Returns", xlab="", ylab="")
```



对数收益率的正态 QQ 图：

```
x <- coredata(logret.AAPL)
qqnorm(x, main="Apple Log Returns")
qqline(x, col="red")
```



从时间序列曲线图、直方图、正态 QQ 图都可以看出股票对数收益率有厚尾现象。

2.4.2 美国十年期国债日数据

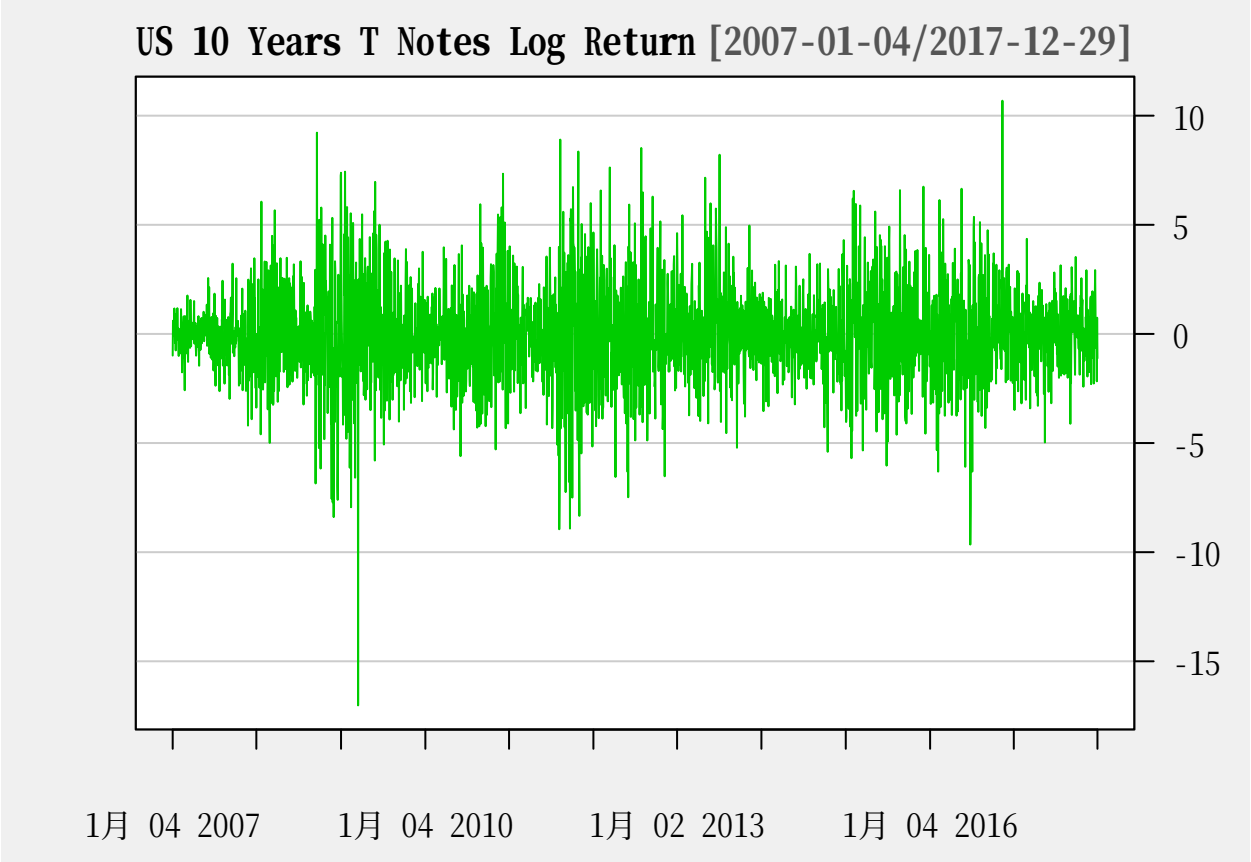
美国十年期国债日对数收益率下载和计算：

```
TNX <- getSymbols("^TNX", src="yahoo", auto.assign=FALSE)
```

```
logret.TNX <- diff(log(TNX$TNX.Adjusted))*100 # 对数收益率
```

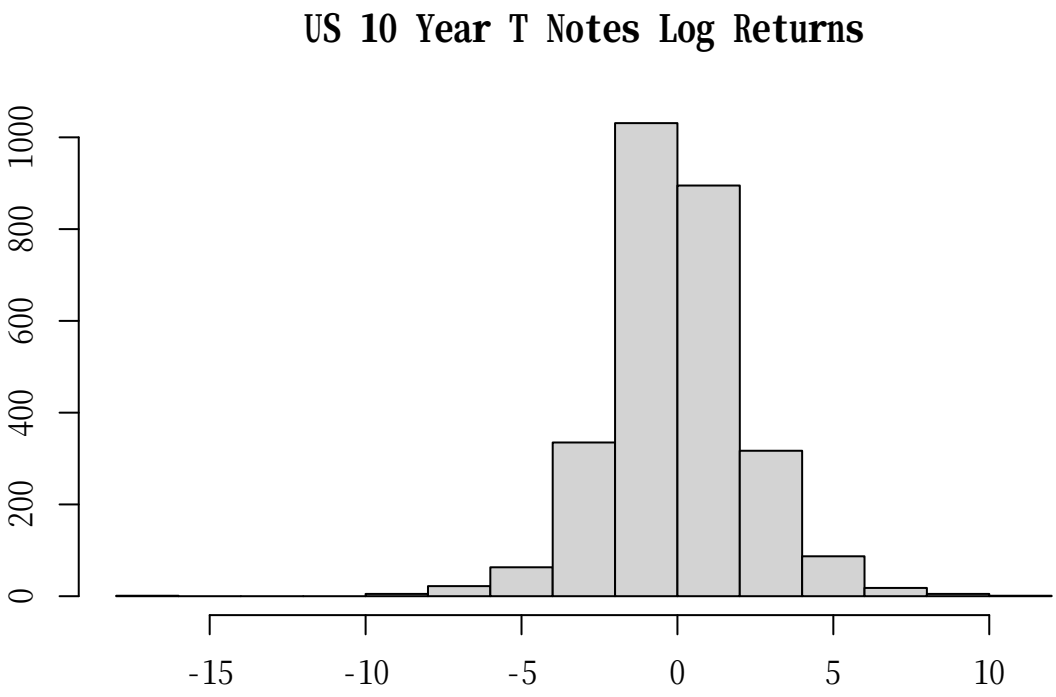
对数收益率时间序列 2007-2017 的曲线图：

```
chartSeries(
  logret.TNX, type="l", TA=NULL,
  subset="2007/2017",
  name="US 10 Years T Notes Log Return",
  theme="white", major.ticks="years", minor.ticks=FALSE)
```



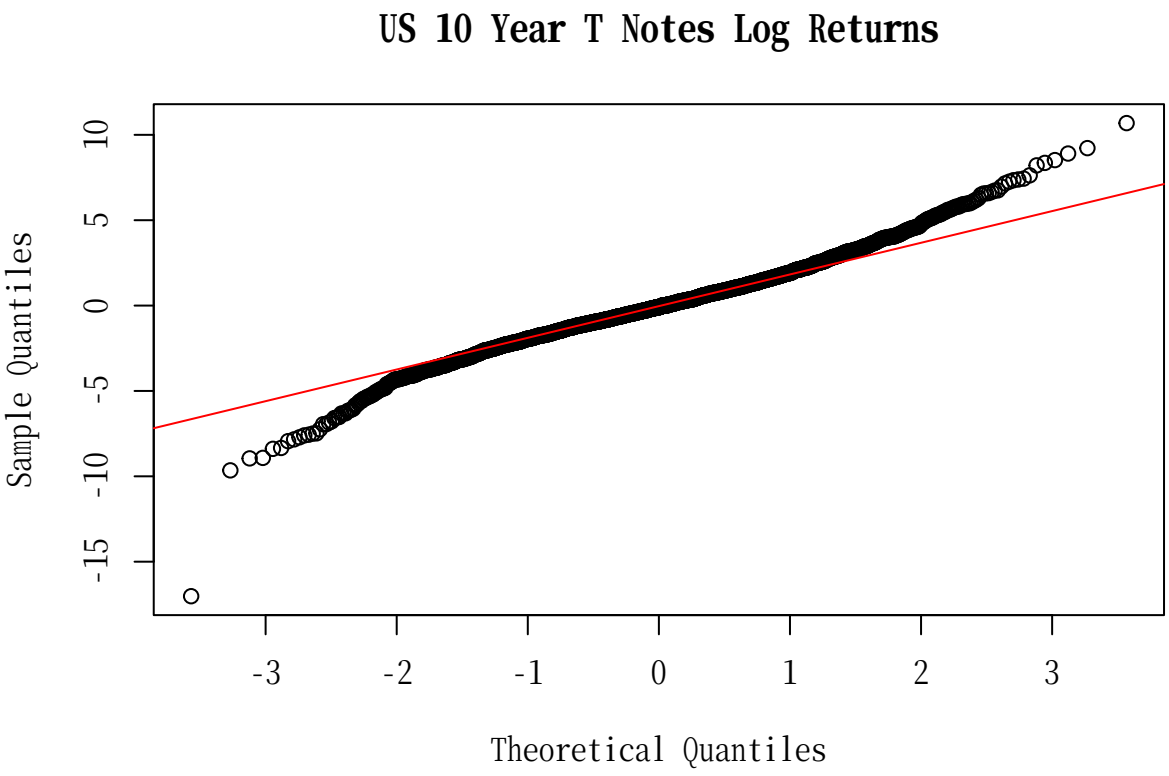
对数收益率的直方图：

```
x <- coredata(logret.TNX)
hist(x, main="US 10 Year T Notes Log Returns", xlab="", ylab="")
```



对数收益率的正态 QQ 图：

```
x <- coredata(logret.TNX)
qqnorm(x, main="US 10 Year T Notes Log Returns")
qqline(x, col="red")
```



从时间序列曲线图、直方图、正态 QQ 图都可以看出股票对数收益率有厚尾现象。比苹果股票收益率的厚尾性弱一些。

2.4.3 欧元对美元汇率日数据

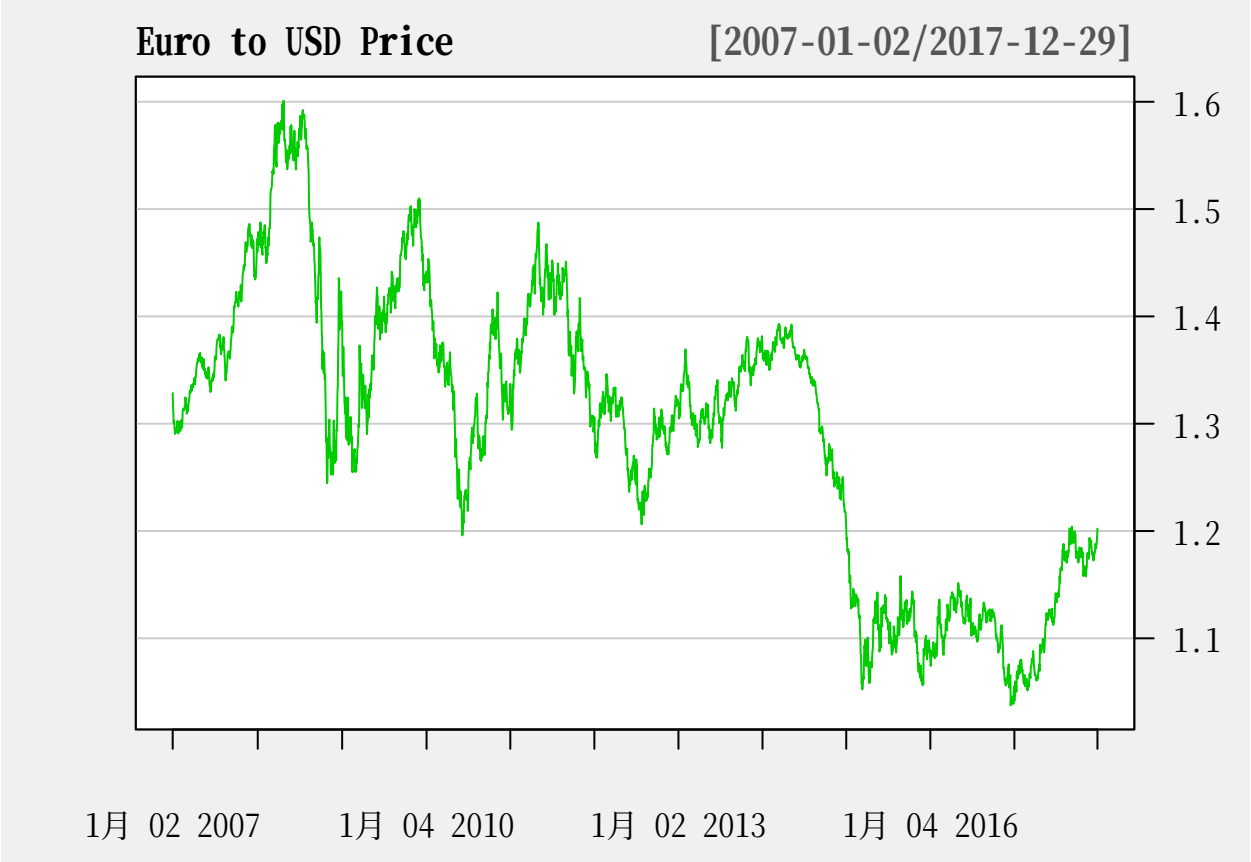
下载欧元对美元汇率的日数据，一元时间序列：

```
DEXUSEU <- getSymbols("DEXUSEU", src="FRED",
  auto.assign=FALSE)
```

```
logret.DEXUSEU <- diff(log(DEXUSEU))*100 # 对数收益率
```

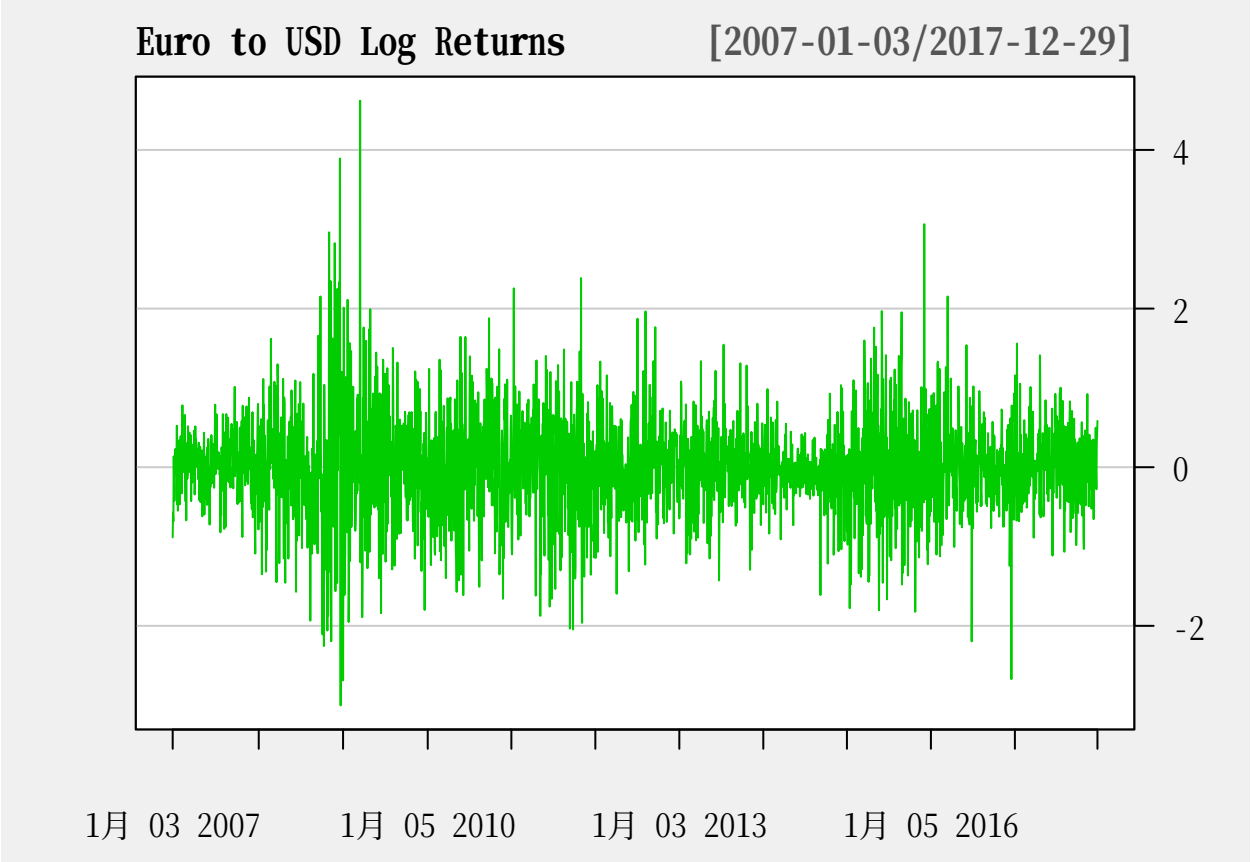
汇率价格的曲线图：

```
chartSeries(
  DEXUSEU, type="l", TA=NULL,
  subset="2007/2017",
  name="Euro to USD Price",
  theme="white", major.ticks="years",
  minor.ticks=FALSE)
```



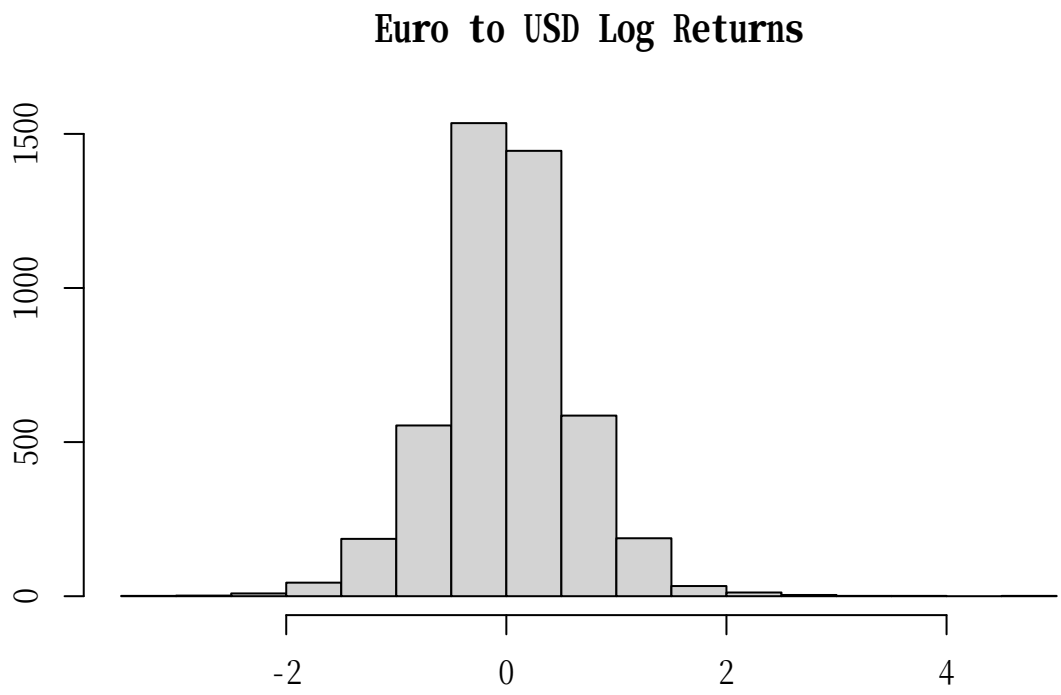
对数收益率时间序列 2007-2017 的曲线图：

```
chartSeries(
  logret.DEXUSEU, type="l", TA=NULL,
  subset="2007/2017",
  name="Euro to USD Log Returns",
  theme="white", major.ticks="years", minor.ticks=FALSE)
```



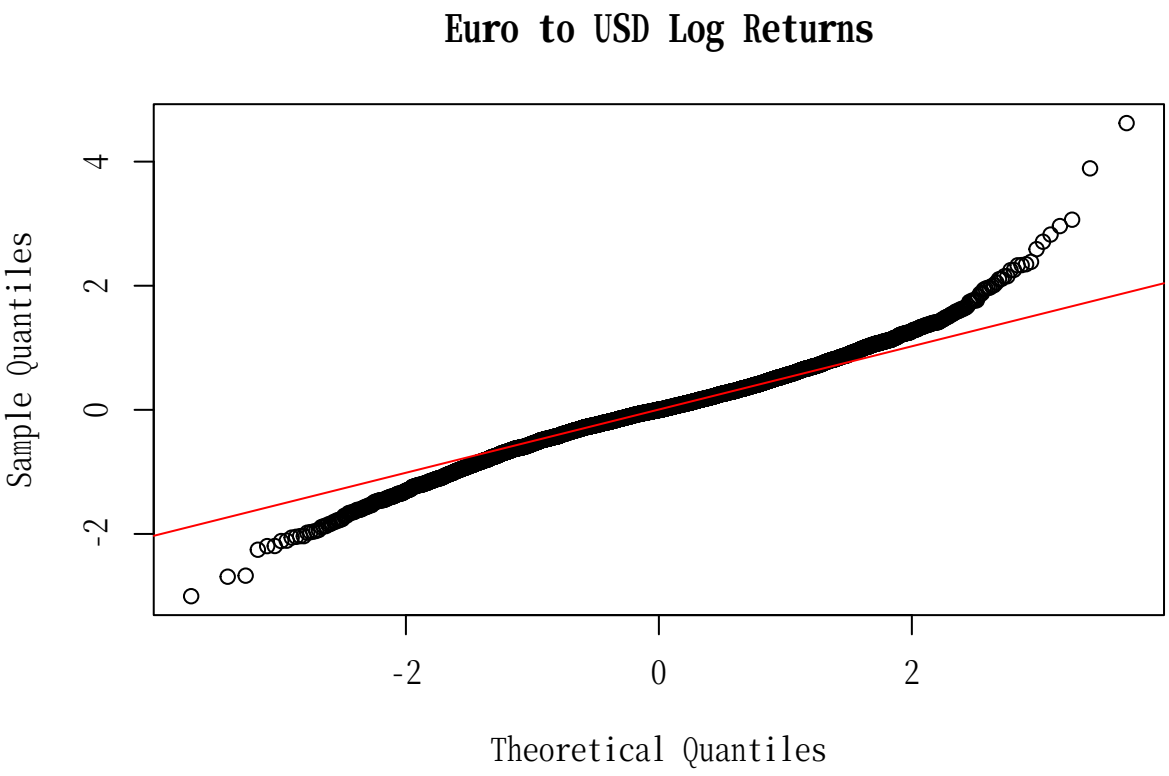
对数收益率的直方图：

```
x <- coredata(logret.DEXUSEU)
hist(x, main="Euro to USD Log Returns", xlab="", ylab="")
```



对数收益率的正态 QQ 图：

```
x <- coredata(logret.DEXUSEU)
qqnorm(x, main="Euro to USD Log Returns")
qqline(x, col="red")
```



也呈现出厚尾性，且右尾比左尾长。

2.5 收益率的分布特性

设 i 资产在时刻 t 的对数收益率为 r_{it}, 简单收益率为 R_{it}, 超额收益率为 z_{it}，$i = 1, \dots, N$, $t = 1, 2, \dots, T$。

2.5.1 统计分布复习

$\mathbb{R}^k$ 表示元素是有 k 个实数值分量的向量组成欧式空间。$x \in \mathbb{R}^k$ 表示其中一个点（向量）。

2.5.1.1 联合分布和边缘分布

随机向量的概念。

$$F_{X,Y}(x, y|\theta) = P_\theta(X \le x, Y \le y)$$

表示随机变量 X 和 Y 的参数为 θ 的**联合分布函数**，$x \in \mathbb{R}^p$, $y \in \mathbb{R}^q$。

若 (X, Y) 有**联合密度函数** $f_{X,Y}(x, y)$, 则

$$F_{X,Y}(x, y) = \int_{-\infty}^{x} \int_{-\infty}^{y} f_{X,Y}(x, y)\, dx\, dy$$

已知联合分布，X 的**边缘分布**为

$$F_X(x|\theta) = F_{X,Y}(x, \infty, \dots, \infty|\theta)$$

一元随机变量 X 的**分布函数** (累积分布函数，cumulative distribution function, CDF)

$$F_X(x|\theta) = P_\theta(X \le x)$$

其中 θ 是分布参数。分布函数单调不减右连续，$\lim_{x\to-\infty} F(x) = 0$, $\lim_{x\to\infty} F(x) = 1$。

2.5.1.2 分位数

如果 X 的一元分布函数 $F(x)$ 严格单调递增且连续，则 $\forall p \in (0,1)$，存在唯一的 x_p 使得 $F(x_p) = p$，x_p 称为随机变量 X 或者分布 $F(x)$ 的 p **分位数**，也记作 $F^{-1}(p)$。一般地，令

$$x_p = \inf\{x|F_X(x) \ge p\},\ \forall p \in (0,1)$$

这样定义的 x_p 存在唯一。也记作 $F^{-1}(p)$。

p 分位数的另一定义是 x_* 满足

$$P(X \le x_*) \ge p,\ P(X \ge x_*) \ge 1-p$$

这样定义的分位数必存在但不一定唯一。

2.5.1.3 条件分布

在 $Y \le y$ 条件下 X 的条件分布为

$$F(x|Y \le y, \theta) = \frac{P(X \le x, Y \le y|\theta)}{P(Y \le y|\theta)} = \frac{F(x,y|\theta)}{F_Y(y|\theta)}$$

有联合密度 $f(x,y|\theta)$ 时，$Y = y$ 条件下 X 有条件密度

$$f(x|y,\theta) = \frac{f(x,y|\theta)}{f_Y(y|\theta)},\quad f(x,y|\theta) = f_Y(y)f(x|y,\theta)$$

$Y = y$ 条件下可以计算条件概率

$$P(X \le x|Y = y, \theta) = \int_{-\infty}^{x} f(x|y,\theta)\,dx$$

条件期望

$$E_\theta(X|Y = y) = \int x f(x|y,\theta)\,dx = g(y),\quad E_\theta(X|Y) = g(Y)$$

条件方差为

$$\mathrm{Var}(X|Y,\theta) = E_\theta\left([X - E(X|Y)]^2|Y\right)$$

2.5.1.4 矩

期望：

$$EX = \int x f(x)\,dx$$

经常记为 μ。

方差

$$\mathrm{Var}(X) = E(X - EX)^2$$

经常记为 σ^2。σ 称为标准差 (standard deviation)。对于资产收益率，方差和标准差是度量其不确定性的指标，可用于资产风险管理和度量。

随机变量 X 的 k 阶**原点矩**

$$m_k' = E(X^k)$$

期望是一阶原点矩。

k 阶**中心矩**

$$m_k = E[(X - EX)^k]$$

方差是二阶中心矩。

随机变量分布中一元正态分布完全由其期望和方差决定，其它分布可能需要更多矩。

偏度 (Skewness): 令 $Y = \frac{X-EX}{\sqrt{\text{Var}(X)}}$，称为 X 的标准化，Y 的三阶矩称为 X 的偏度，可以用来度量分布的对称性，负偏度称为左偏，反映左尾偏长的情况；正偏度称为右偏，反映右尾偏长的情况。

峰度 (Kurtosis): 将 X 标准化为 Y，Y 的四阶矩减去 3 称为 X 的超额峰度或峰度，正态分布的超额分布等于 0。峰度大的分布有**重尾**或厚尾现象，其分布密度在 $\pm\infty$ 处趋于零速度较慢，其样本具有较多的异常值（离群值，outliers）。

2.5.1.5 矩的估计和检验

样本均值

$$\bar{x} = \hat{\mu} = \frac{1}{T}\sum_{t=1}^{T} x_t$$

样本方差

$$S^2 = \hat{\sigma}^2 = \frac{1}{T-1}\sum_{t=1}^{T}(x_t - \bar{x})^2$$

样本偏度

$$\hat{\text{Skew}} = \frac{1}{T-1}\sum_{t=1}^{T}\left(\frac{x_t - \bar{x}}{S}\right)^3$$

样本超额峰度

$$\hat{\text{Kurt}} = \frac{1}{T-1}\sum_{t=1}^{T}\left(\frac{x_t - \bar{x}}{S}\right)^4 - 3$$

R 软件中扩展包 fBasics 中的 `basicStats()` 函数可以计算各种基本的统计量。收益率数据在大多数情况下表现得与独立同分布观测很接近，所以尽管我们不能确保其独立同分布，在实际统计分析时仍使用在独立同分布假设下的各种方法。

为检验 $H_0 : \mu = 0$，可以用统计量

$$Z = \frac{\bar{x}}{S/\sqrt{T}}$$

在 H_0 成立且 T 充分大时 Z 近似服从标准正态分布，利用这样的分布计算检验的 p 值。在 R 软件中

```
t.test(x)
```

可以进行 $H_0 : \mu = 0$ 的检验。

若观测为独立的正态分布样本，则 T 充分大时样本偏度渐近服从 $\text{N}(0, 6/T)$ 分布，样本超额峰度渐近服从 $\text{N}(0, 24/T)$ 分布，据此可以计算检验 H_0：偏度 $= 0$ 和 H_0：超额峰度 $= 0$ 的 p 值。

Jarque 和 Bera(1987) 提出了 Jarque-Bera 检验，零假设是总体服从正态分布，设观测为独立同分布样本，统计量为

$$\mathrm{JB} = \frac{\hat{\mathrm{Skew}}^2}{6/T} + \frac{[\hat{\mathrm{Kurt}} - 3]^2}{24/T}$$

在零假设下 JB 统计量渐近服从 $\chi^2(2)$ 分布。计算右侧 p 值。在 R 软件中

```
fBasics::normalTest(x, method="jb")
```

可以执行正态性 JB 检验。

`tseries::jarque.bera.test()` 也可以执行 Jarque-Bera 检验。

JB 检验的原始文献：CARLOS M. JARQUE and ANIL K. BERA(1980), Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals, Economics Letters 6, pp. 255-259.

2.5.2 收益率分布研究例子

作为示例，对苹果公司股票日收盘价的收益率数据进行简单分析。

取 2011-2017 数据子集，计算收盘价的对数收益率：

```
x <- coredata(diff(
  log(AAPL["2011/2017"]$AAPL.Adjusted)))[-1]
```

基本统计量：

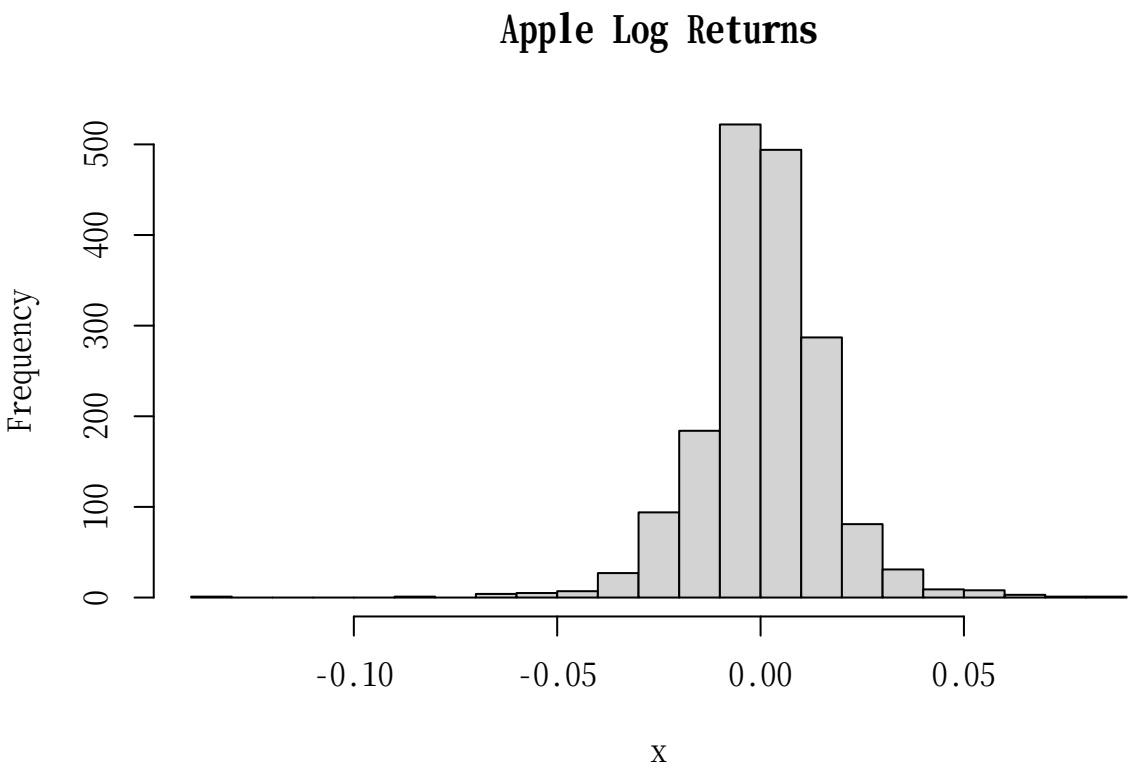
```
fBasics::basicStats(x)
```

```
##                        x
## nobs        1760.000000
## NAs            0.000000
## Minimum       -0.131885
## Maximum        0.085022
## 1. Quartile   -0.006914
## 3. Quartile    0.009411
## Mean           0.000789
## Median         0.000574
## Sum            1.388711
## SE Mean        0.000377
## LCL Mean       0.000050
## UCL Mean       0.001528
## Variance       0.000250
## Stdev          0.015812
## Skewness      -0.323171
## Kurtosis       5.486581
```

这里的 Kurtosis 是超额峰度。有左偏，重尾。

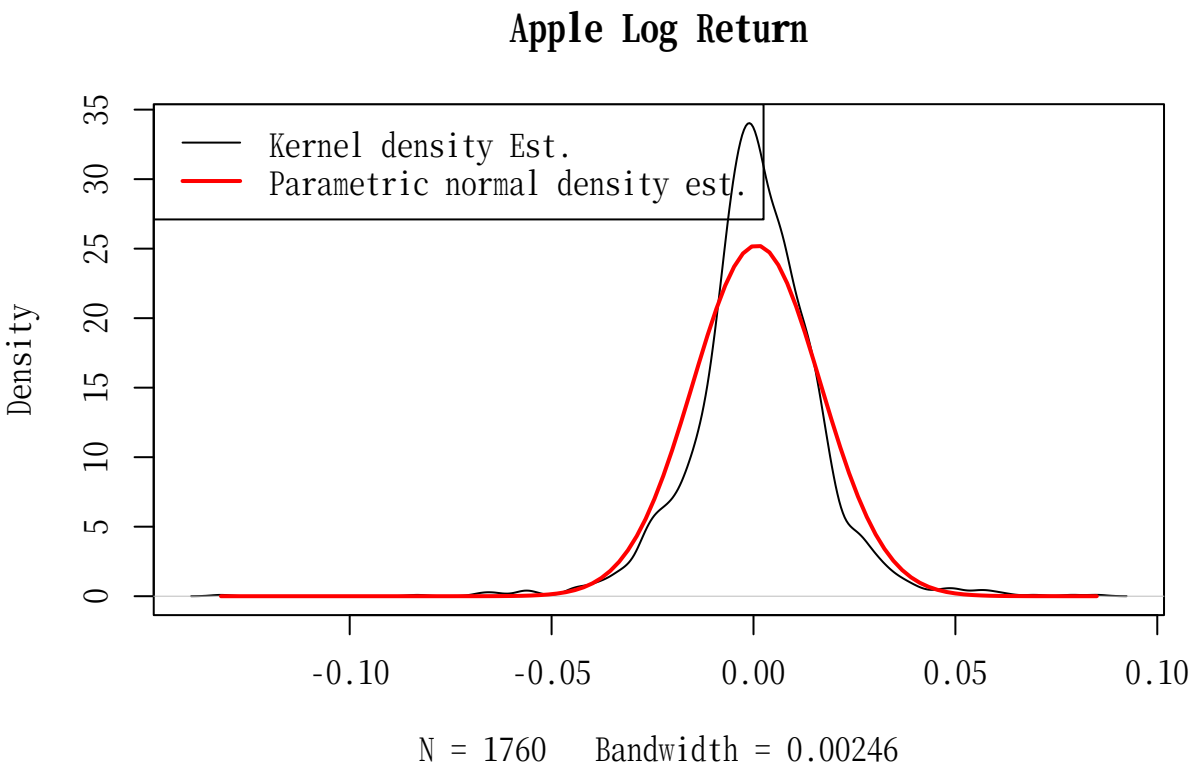
直方图：

```
hist(x, nclass=30, main="Apple Log Returns")
```



作核密度估计图并叠加正态密度估计：

```
tmp.1 <- density(x, na.rm=TRUE)
tmp.x <- seq(min(x, na.rm=TRUE), max(x, na.rm=TRUE),
             length.out=100)
tmp.y <- dnorm(tmp.x, mean(x, na.rm=TRUE),
               sd(x, na.rm=TRUE))
tmp.ra <- range(c(tmp.1$y, tmp.y), na.rm=TRUE)
plot(tmp.1, main="Apple Log Return",
     ylim=tmp.ra)
lines(tmp.x, tmp.y, lwd=2, col="red")
legend("topleft", lwd=c(1,2),
       col=c("black", "red"),
       legend=c("Kernel density Est.",
                "Parametric normal density est."))
```



比正态密度明显重尾，且左尾比右尾长。

均值为零的检验：

```
t.test(x)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 2.0935, df = 1759, p-value = 0.03644
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.0000498348 0.0015282461
## sample estimates:
##    mean of x
## 0.0007890405
```

检验 p 值为 0.036，在 0.05 水平下显著，可认为均值不为零，显著地大于零。

单独计算偏度，及其标准化：

```
tmp <- fBasics::basicStats(x)["Skewness", 1]; tmp
```

```
## [1] -0.323171
```

```
tmp/sqrt(6/length(x))
```

```
## [1] -5.534942
```

标准化值的绝对值超过 1.96，在 0.05 水平下拒绝偏度等于零的假设。

单独计算超额峰度，及其标准化：

```
tmp <- fBasics::basicStats(x)["Kurtosis", 1]; tmp
```

```
## [1] 5.486581
```

```
tmp/sqrt(24/length(x))
```

```
## [1] 46.98427
```

标准化值的绝对值超过 1.96，在 0.05 水平下拒绝峰度等于零的假设。

正态性 JB 检验:

```
fBasics::normalTest(x, method="jb")
```

```
##
## Title:
##  Jarque - Bera Normalality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 2245.9834
##   P VALUE:
##     Asymptotic p Value: < 2.2e-16
##
## Description:
##  Wed Mar 23 16:18:43 2022 by user: Lenovo
```

p 值为小于万分之一的值，在 0.05 水平下显著。

2.6 金融数据的图形

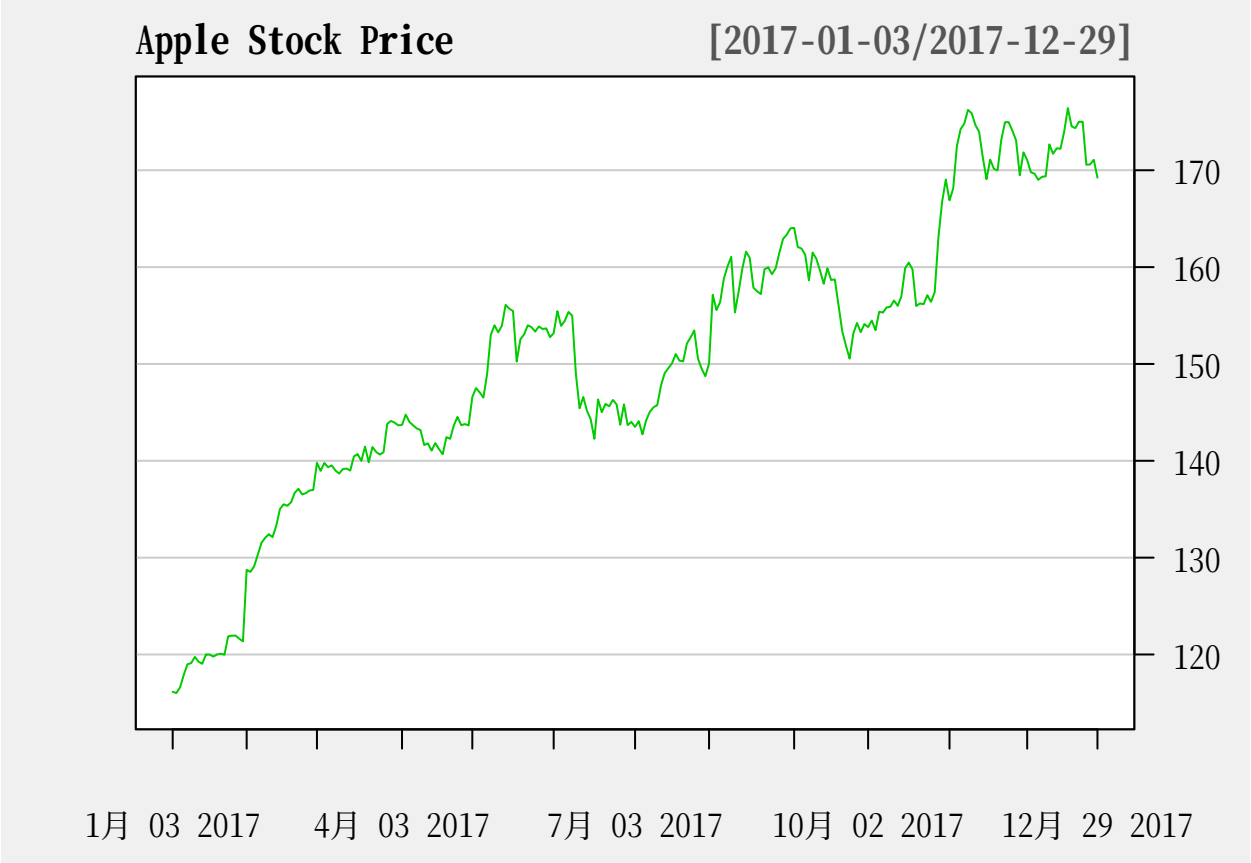
- 时间序列曲线图
- K 线图
- 叠加均线等
- 直方图，核密度估计，参数密度估计
- QQ 图
- ACF 和 PACF 等

2.6.1 时间序列曲线图实例

用 `quantmod::chartSeries()` 作图，优点是支持曲线图、K 线图、交易量同步显示、增加均线等分析。缺点是可定制性受限。

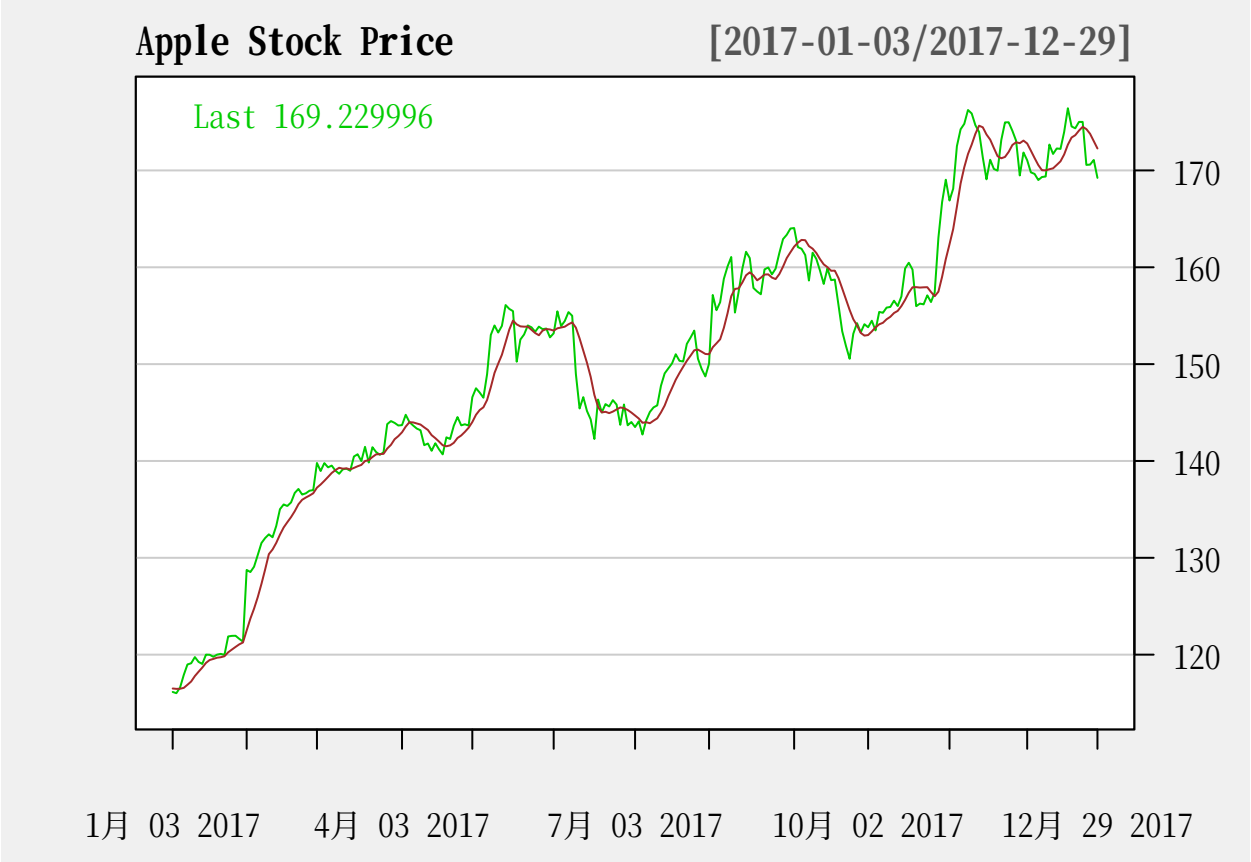
苹果公司 2017 年日收盘价的时间序列图：

```
chartSeries(
  AAPL, type="line",
  subset="2017", TA=NULL,
  theme="white", name="Apple Stock Price",
  major.ticks="months", minor.ticks=FALSE)
```



带有 7 日均线的曲线图：

```
chartSeries(
  AAPL, type="line",
  subset="2017", TA="addSMA(7)",
  theme="white", name="Apple Stock Price",
  major.ticks="months", minor.ticks=FALSE)
```



用 `TA="..."` 选项可以添加技术曲线。详见 `quantmod::TA()` 的文档。

用 ggplot2 包的作图功能则可以进行较多的控制，比如对时间坐标轴的精细控制。如：

```
library(ggplot2)
p <- ggplot(data = tibble(
  time=index(AAPL),
  price=coredata(AAPL)[,"AAPL.Close",drop=TRUE]),
  mapping = aes(x=time, y=price))
p + geom_line()
```

修改时间轴，每 5 年标数值，每年画竖线：

```
p + geom_line() +
  scale_x_date(
  name = "year",
  date_breaks = "5 years",
  date_labels = "%Y",
  date_minor_breaks = "1 year",
  expand = c(0, 0))
```

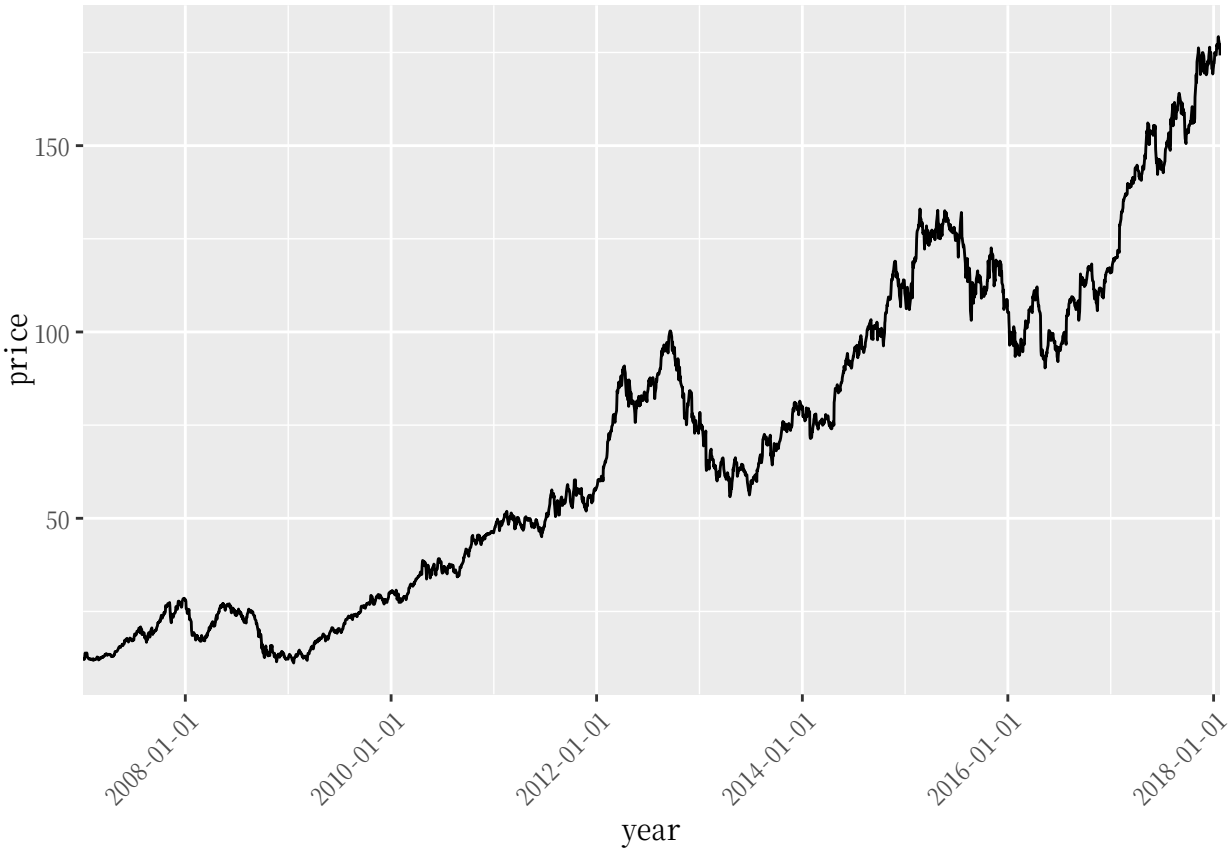
选项 `date_labels` 是日期刻度的一种格式规定，比如”%Y” 是四位数的年份，“%Y-%m” 是年月格式，等等。

也可以人为指定年份：

```
p + geom_line() +
  scale_x_date(
  name = "year",
  breaks = lubridate::make_date(c(2008, 2010, 2012, 2014, 2016)),
  date_labels = "%Y",
  date_minor_breaks = "1 year",
  expand = c(0, 0))
```

可以要求日期旋转 45 度显示：

```
p + geom_line() +
  scale_x_date(
    name = "year",
    date_labels = "%Y-%m-%d",
    expand = c(0, 0)) +
  theme(
    axis.text.x = element_text(
      angle = 45, vjust = 1, hjust = 1)  )
```



2.6.2 K 线图实例

用 `quantmod::chartSeries()` 作图。

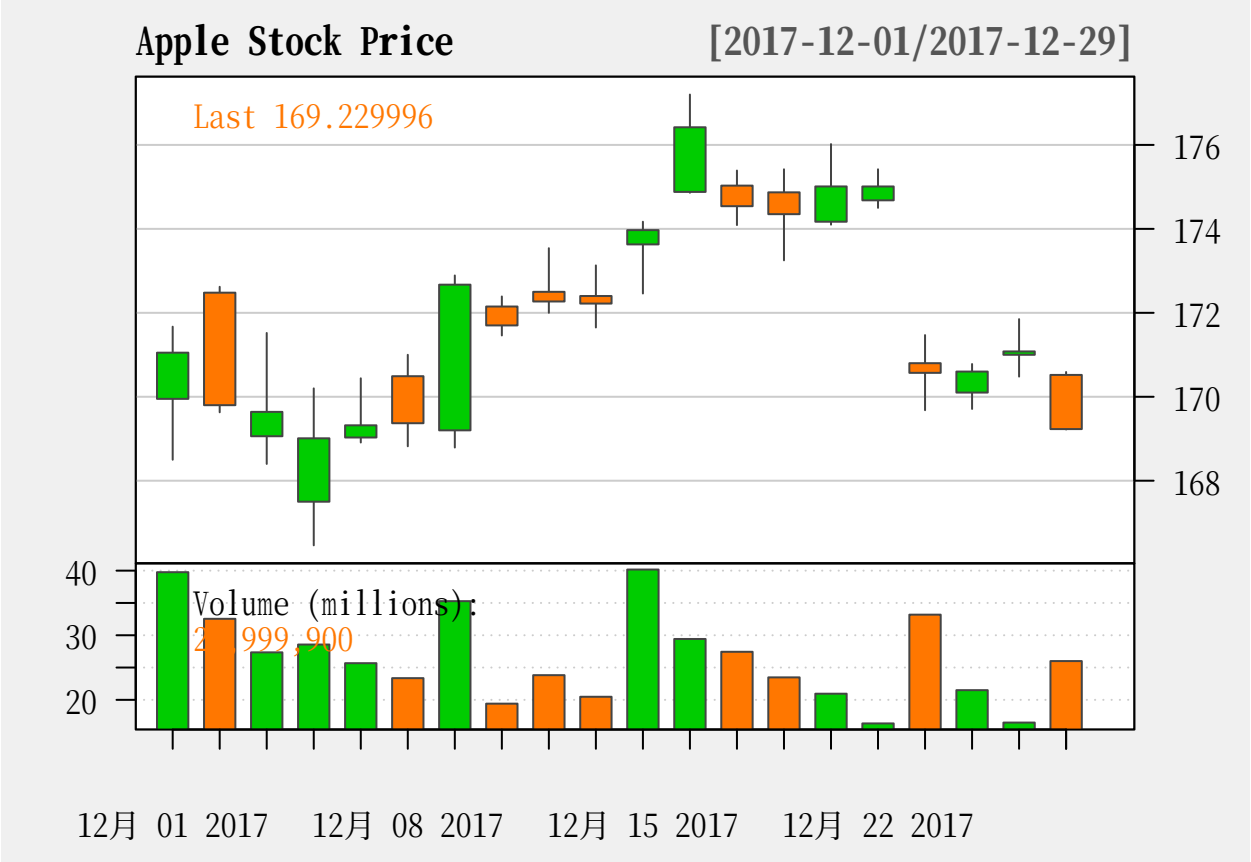
苹果公司 2017 年日收盘价的 K 线图，带有成交量：

```
chartSeries(
  AAPL,
  subset="2017",
  theme="white", name="Apple Stock Price",
  major.ticks="months", minor.ticks=FALSE)
```

当时间点比较多时，K 线图画成了“火柴”形状，每一个 K 线单元用窄线画出。

仅画最后一个月：

```
chartSeries(
  AAPL,
  subset="2017-12",
  theme="white", name="Apple Stock Price",
  major.ticks="auto", minor.ticks=FALSE)
```



可见时间点比较少时，每个 K 线单元画成条形。

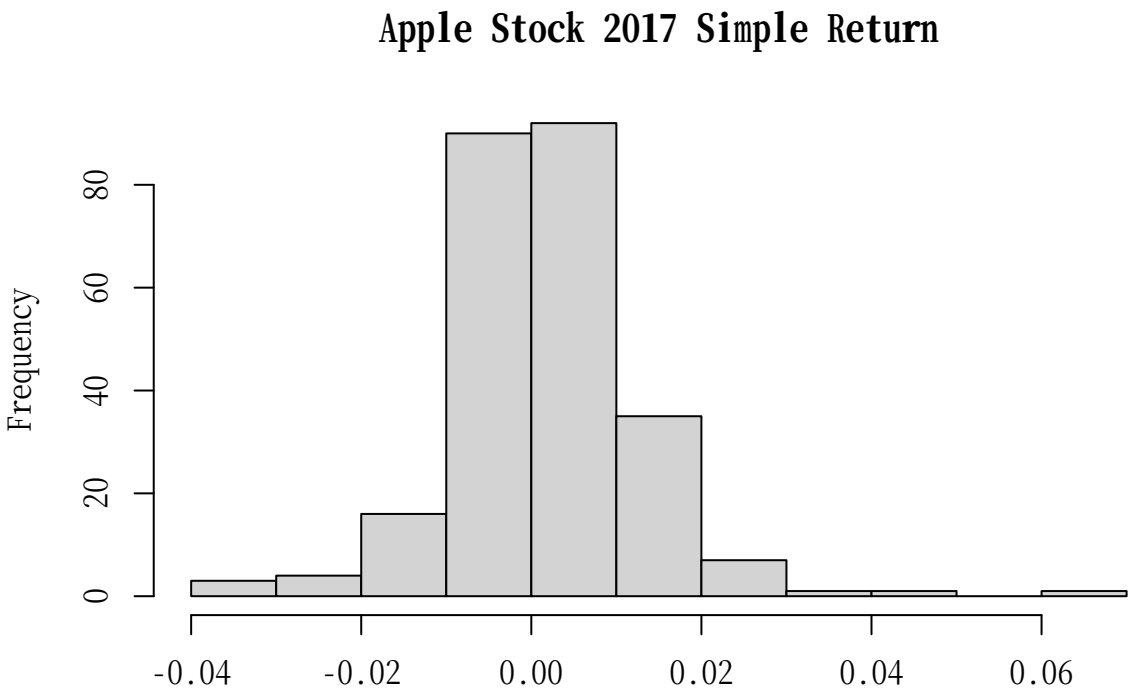
在 K 线图中，条形两端纵坐标为开盘价和收盘价，低开高收时为阳线，高开低收时为阴线；用单色显示时，阳线显示成空心条形，阴线显示成实心条形；用红色和绿色显示时，国内的习惯是红色阳线，绿色阴线，而西方的习惯是绿色阳线，红色阴线。如果最低、最高价不等于开盘价和收盘价，则向下画短线或者向上画短线。这样，每个 K 线单元都表示了 OHLC 四个值。

子集还可以指定成`"from/to"` 的格式。

2.6.3 密度估计

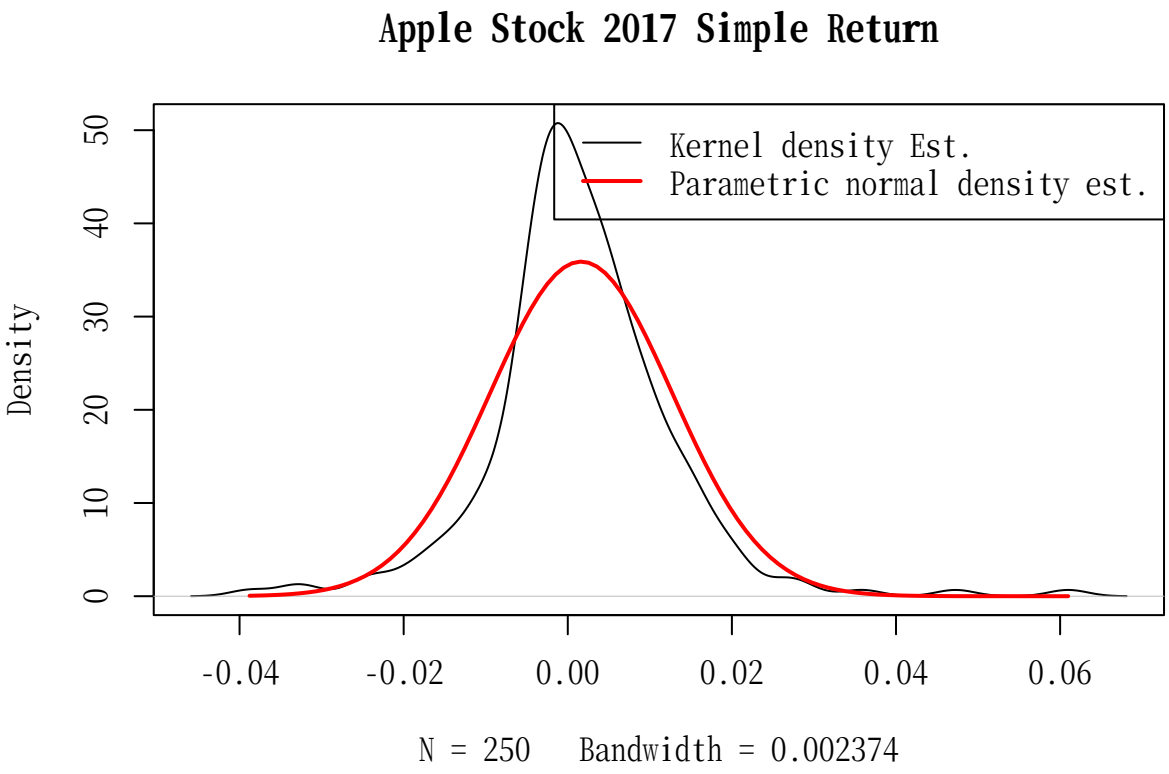
苹果公司 2017 年简单收益率的直方图：

```
x <- simple.return(AAPL["2017","AAPL.Adjusted"])
hist(x, main="Apple Stock 2017 Simple Return", xlab="")
```



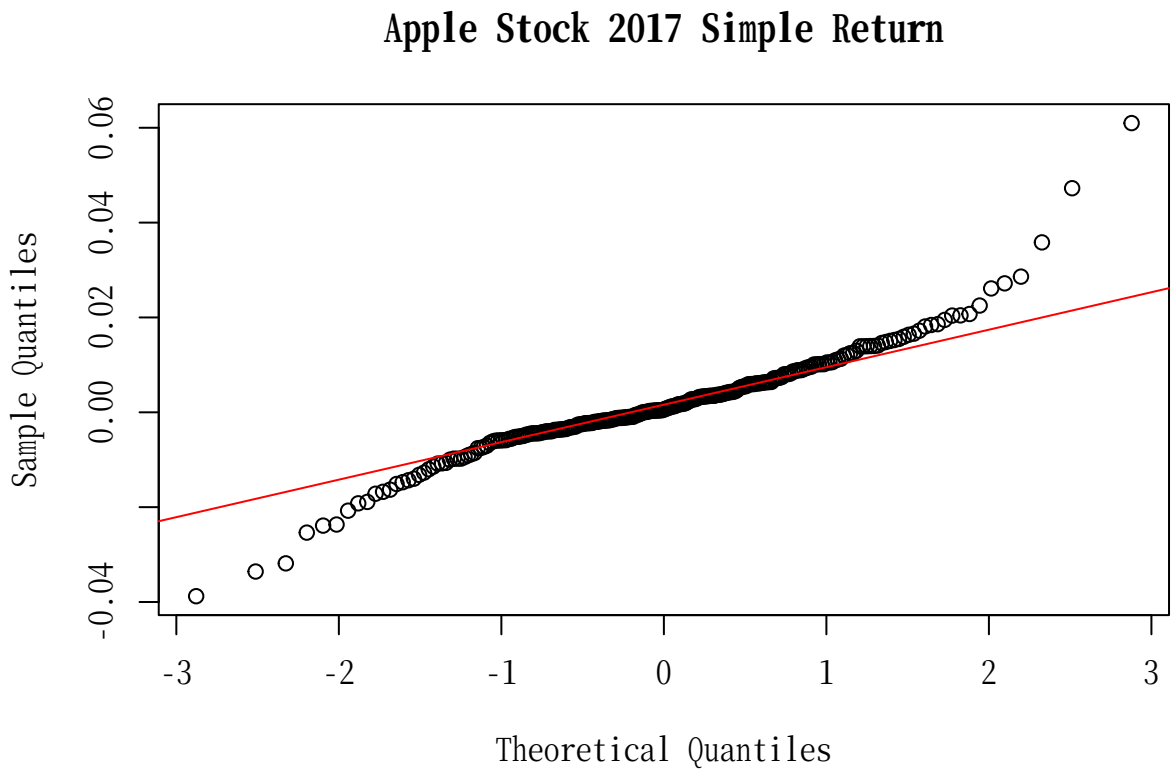
作核密度估计图并叠加正态密度估计 (重复前面例子):

```
tmp.1 <- density(x, na.rm=TRUE)
tmp.x <- seq(min(x, na.rm=TRUE), max(x, na.rm=TRUE),
             length.out=100)
tmp.y <- dnorm(tmp.x, mean(x, na.rm=TRUE),
               sd(x, na.rm=TRUE))
tmp.ra <- range(c(tmp.1$y, tmp.y), na.rm=TRUE)
plot(tmp.1, main="Apple Stock 2017 Simple Return",
     ylim=tmp.ra)
lines(tmp.x, tmp.y, lwd=2, col="red")
legend("topright", lwd=c(1,2),
       col=c("black", "red"),
       legend=c("Kernel density Est.",
                "Parametric normal density est."))
```




```
qqnorm(x, main="Apple Stock 2017 Simple Return")
qqline(x, col="red")
```

2.6.4 QQ 图



呈现左右均重尾的分布。非正态。

2.6.5 散点图与回归直线

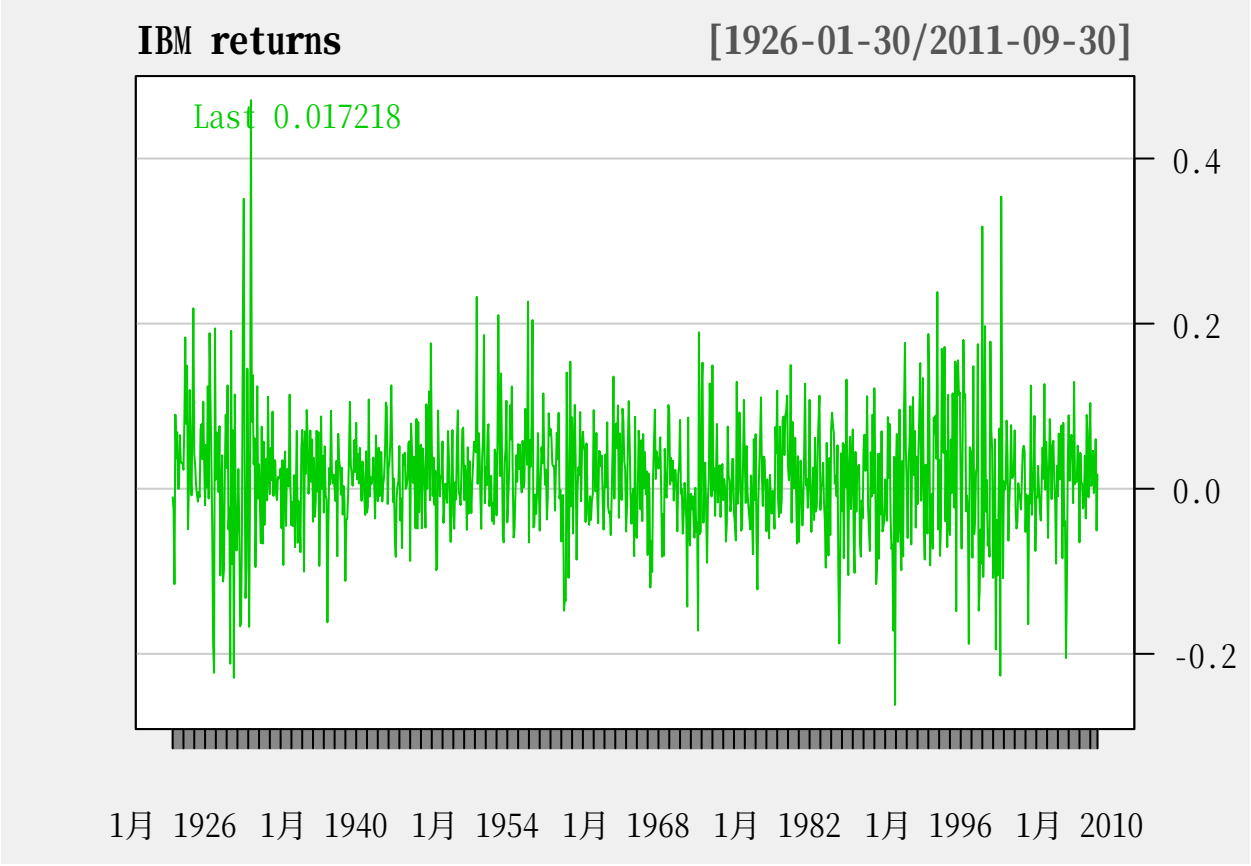
读入 IBM 和 S&P 的月度收益率数据，从 1926-01 到 2011-09, 并转换为 xts 格式，时间下标改为 yearmon 格式：

```
d <- read_table(
  "m-ibmsp-2611.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
ibmsp <- xts(as.matrix(d[,2:3]), d$date)
tclass(ibmsp) <- "yearmon"
head(ibmsp)
```

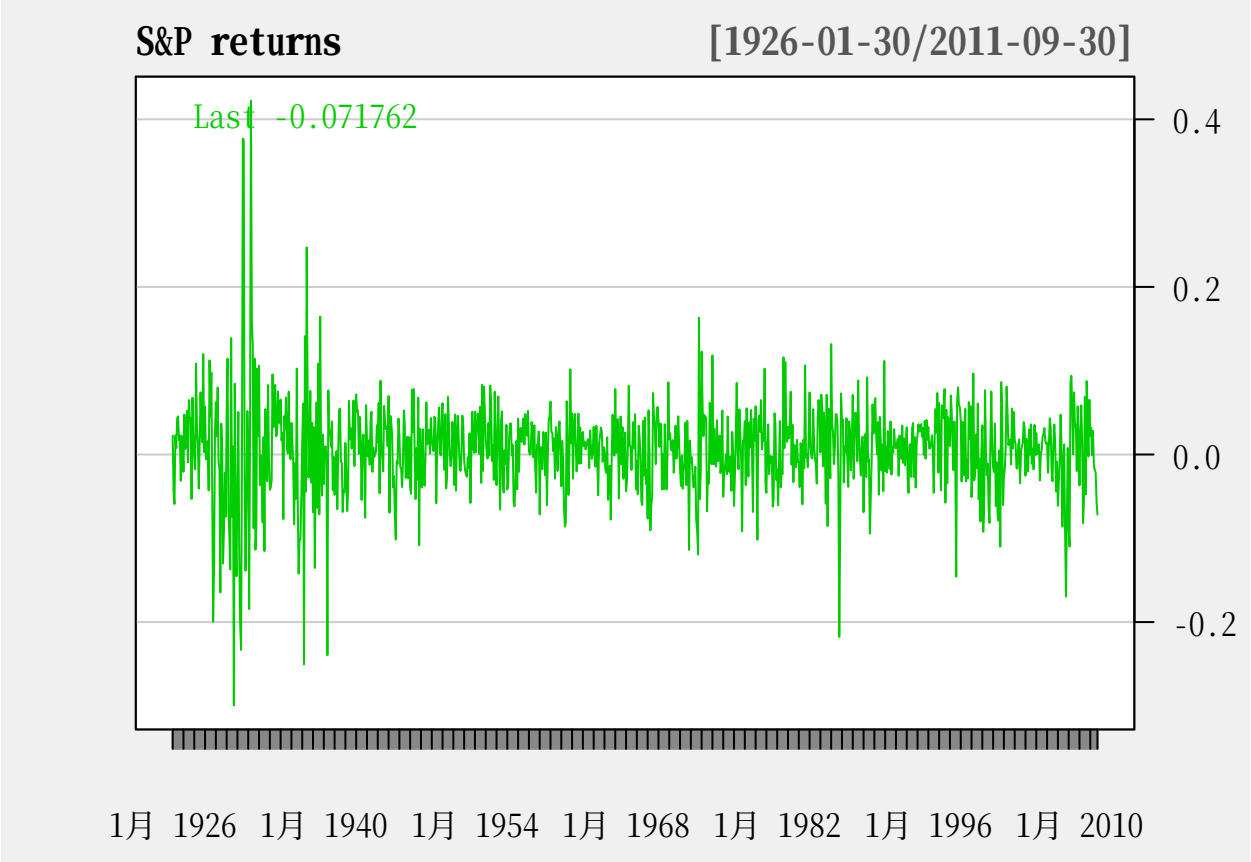
```
##                ibm        sp
## 1月 1926 -0.010381  0.022472
## 2月 1926 -0.024476 -0.043956
## 3月 1926 -0.115591 -0.059113
## 4月 1926  0.089783  0.022688
## 5月 1926  0.036932  0.007679
## 6月 1926  0.068493  0.043184
```

序列图：

```
chartSeries(
  ibmsp[,"ibm"], type="lines",
  theme="white", name="IBM returns",
  major.ticks="years", minor.ticsk=FALSE)
```

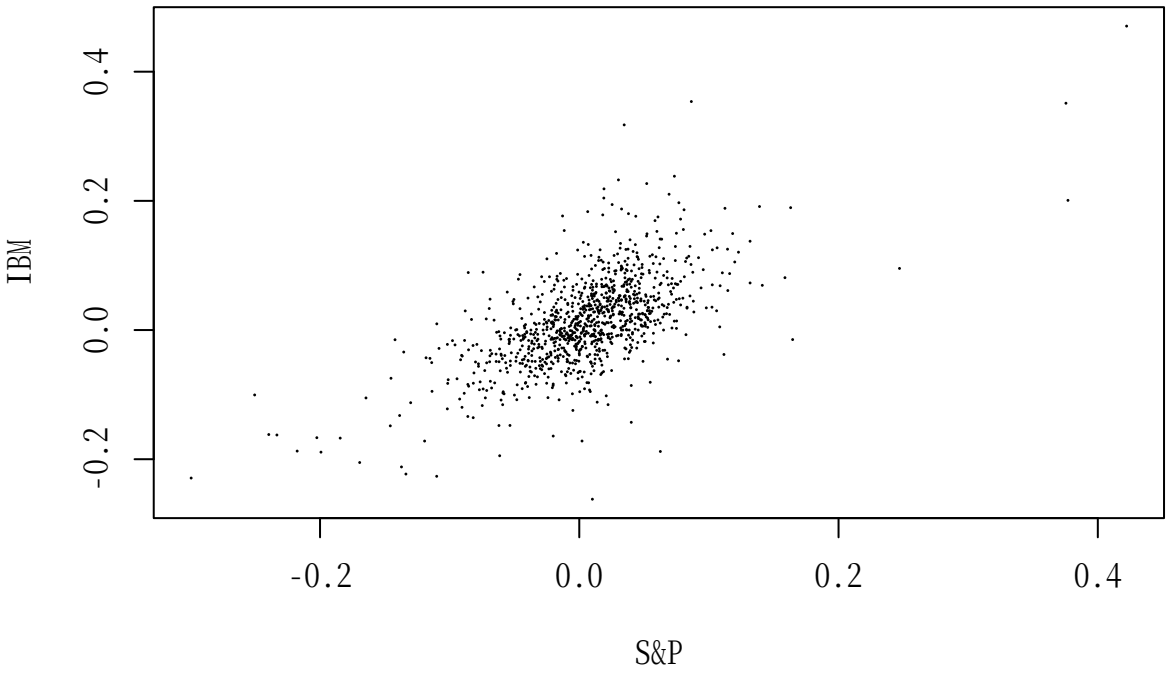



```
chartSeries(
  ibmsp[,"sp"], type="lines",
  theme="white", name="S&P returns",
  major.ticks="years", minor.ticsk=FALSE)
```



IBM 收益率对标准普尔指数收益率的散点图：

```
d <- as.matrix(coredata(ibmsp))
plot(d[,"sp"], d[,"ibm"], pch=16, cex=0.2,
     xlab="S&P", ylab="IBM")
```



计算相关系数：

```
d <- as.matrix(coredata(ibmsp))
cor.test(d[,"sp"], d[,"ibm"])
```

```
##
##  Pearson's product-moment correlation
##
## data:  d[, "sp"] and d[, "ibm"]
## t = 26.664, df = 1027, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6020164 0.6743519
## sample estimates:
##       cor
## 0.6395979
```

相关系数为 0.64，显著不等于零。

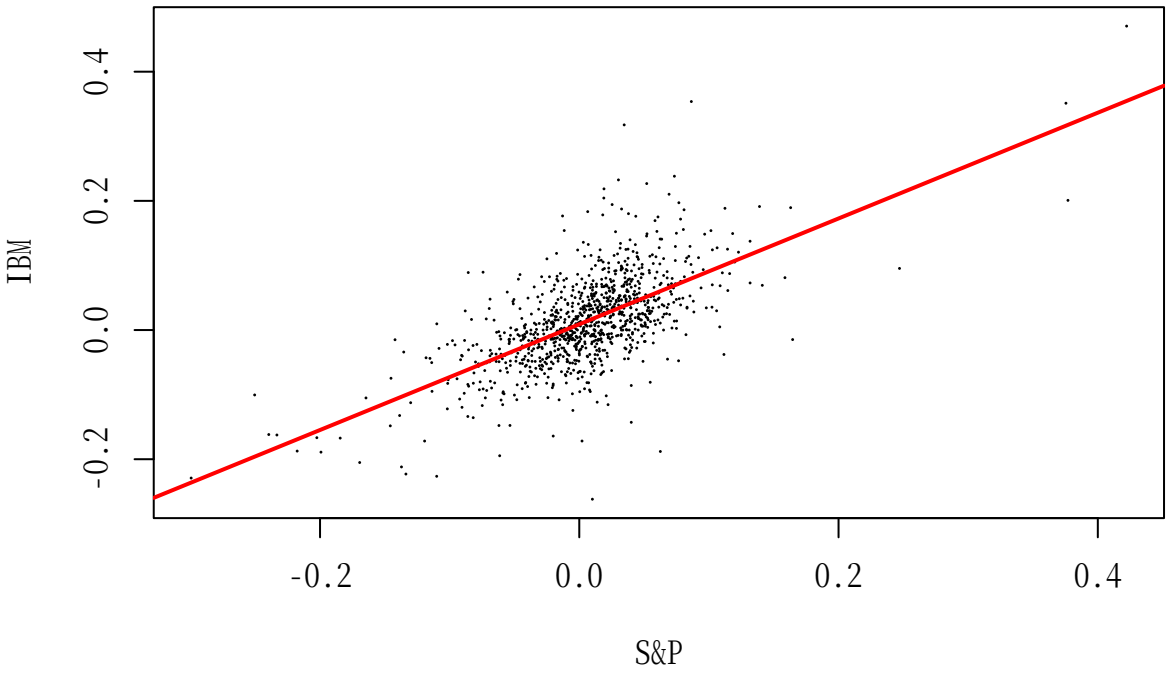
作线性回归并在散点图上叠加回归直线：

```
d <- as.data.frame(coredata(ibmsp))
plot(d[["sp"]], d[["ibm"]], pch=16, cex=0.2,
     xlab="S&P", ylab="IBM")
```

```
lm1 <- lm(ibm ~ sp, data=d)
print(summary(lm1))
```

```
##
## Call:
## lm(formula = ibm ~ sp, data = d)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.279155 -0.032137 -0.002261  0.030647  0.280313
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.008974   0.001706   5.259 1.76e-07 ***
## sp          0.818776   0.030707  26.664  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05443 on 1027 degrees of freedom
## Multiple R-squared:  0.4091, Adjusted R-squared:  0.4085
## F-statistic:   711 on 1 and 1027 DF,  p-value: < 2.2e-16
```

```
abline(lm1, col="red", lwd=2)
```



回归方程为

$$\text{IBM} = 0.008474 + 0.818776\text{S\&P}$$

一般地，

$$r_t = \alpha + \beta m_t + \varepsilon_t$$

其中 r_t 是某资产的收益率，m_t 是市场的收益率，α 表示相对于市场收益的超额收益率。这样的模型称为市场模型 (market model)。

2.7 金融数据常用分布

2.7.1 正态分布

早期研究中假设简单收益率 $\{R_t, t = 1, 2, \ldots, T\}$ 独立同分布，服从正态分布。但是存在不一致的问题：

- 简单收益率一定大于等于 -100%，正态分布没有这样的限制；
- 多期毛收益率是单期毛收益率的乘积，不再服从正态分布；
- 收益率大多是厚尾分布，而正态分布超额峰度等于零。

2.7.2 对数正态分布

另一种假设是，对数收益率 r_t 独立同分布，同正态 $N(\mu, \sigma^2)$ 分布。这时，简单收益率 $R_t = \exp(r_t) - 1$ 服从对数正态分布，

$$
\begin{aligned}
ER_t &= \exp\left(\mu + \frac{1}{2}\sigma^2\right) - 1, \\
\mathrm{Var}(R_t) &= \exp(2\mu + \sigma^2)\left(e^{\sigma^2} - 1\right)
\end{aligned} \tag{2.1}
$$

反之，如果 R_t 服从对数正态分布，$m_1 = ER_t$, $m_2 = \mathrm{Var}(R_t)$，则 $r_t = \ln(1 + R_t)$ 服从正态分布，且

$$
\begin{aligned}
Er_t &= \ln\left[\frac{m_1 + 1}{\sqrt{1 + \frac{m_2}{(1+m_1)^2}}}\right] \\
\mathrm{Var}(r_t) &= \ln\left[1 + \frac{m_2}{(1 + m_1)^2}\right]
\end{aligned} \tag{2.2}
$$

如果 r_t 是正态分布，多期的对数收益率 $r_t[k]$ 也是正态分布，于是多期简单收益率 $R_t[k]$ 也是对数正态分布。$R_t = e^{r_t} - 1 \geq -1$ 总成立。这样的假定一致性比较好。但是，有些股票的简单收益率的分布表现与对数正态分布不符，对数收益率分布表现与正态分布不符。对数收益率一般也有厚尾性。

2.7.3 稳态分布

稳态分布 (stable distribution，或译为稳定分布) 是正态分布的推广。正态分布的一个重要特性是独立同分布的正态随机变量之和仍服从正态分布，这符合连续复合收益率（对数收益率）r_t 的分布要求。稳态分布则是满足这样的性质的分布，独立同稳态分布的随机变量之和仍服从稳态分布。非正态的稳态分布可以实现厚尾性。正态分布是稳态分布的特例。关于稳态分布可参考 (武东 & 汤银才, 2007)。

除了正态分布以外，其它的稳态分布都没有二阶矩，但仍可以有位置参数、刻度参数，和两个形状参数 α、β。$\alpha \in (0, 2]$ 是稳定分布的特征参数，与分布尾部厚薄有关，正态分布对应于 $\alpha = 2$；β 刻画分布左偏和右偏的程度。独立同稳态分布的随机变量和仍服从稳态分布，且 α、β 参数不变。

柯西 (Cauchy) 分布就是稳态分布，标准柯西分布密度为

$$
p(x) = \frac{1}{\pi(1 + x^2)},\ x \in (-\infty, \infty)
$$

分布密度对称，数学期望和方差都不存在。

2.7.4 混合正态分布

设 δ 为正值随机变量，比如服从 Gamma 分布，设对数收益率在给定 δ 条件下服从条件正态分布 $\mathrm{N}(\mu, \delta^{-1})$，称 r_t 的边缘分布为尺度混合的正态分布。

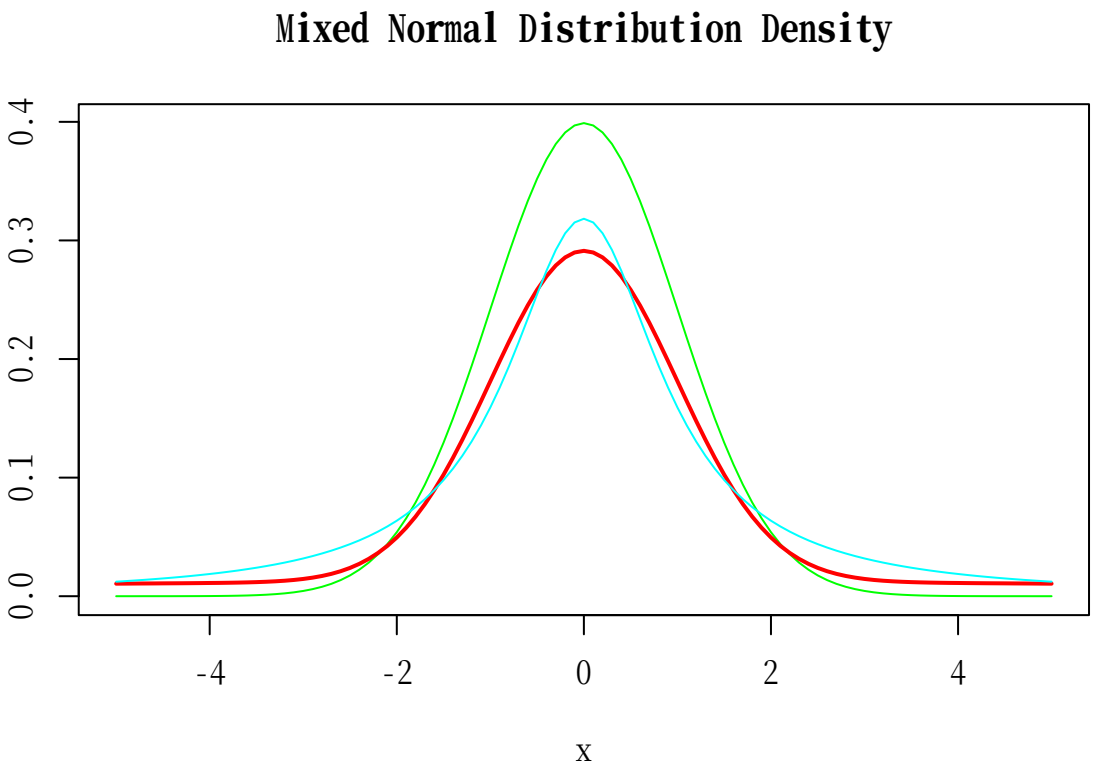
若 $f_1(x)$ 表示 $\mathrm{N}(\mu, \sigma_1^2)$ 的密度，$f_2(x)$ 表示 $\mathrm{N}(\mu, \sigma_2^2)$ 的密度，$0 < \alpha < 1$，若对数收益率 r_t 密度为

$$
f(x) = \alpha f_1(x) + (1 - \alpha) f_2(x)
$$

则称 r_t 服从混合正态分布。

例如，$\mu = 0$, $\sigma_1 = 1$, $\sigma_2 = 10$, $\alpha = 0.7$:

```
curve(dnorm(x, 0, 1), -5, 5,
      ylab="", lwd=1, col="green",
      main="Mixed Normal Distribution Density")
curve(0.7*dnorm(x, 0, 1) + 0.3*dnorm(x, 0, 10), -5, 5, add=TRUE,
      lwd=2, col="red")
curve(dcauchy(x), -5, 5, add=TRUE, col="cyan")
```



其中红色粗线为混合正态密度，绿色细线是标准正态密度。从密度函数图形可以看出明显的重尾。浅蓝色为标准柯西分布，重尾更为明显。

2.7.5 多元收益率的分布

金融中不关心一般的多元分布，主要关心在 $r_{t-1}, \ldots, r_1$ 条件下 r_t 的条件分布，尤其是条件期望和条件方差。

设随机向量 $X = (X_1, \ldots X_p)^T$，期望和协方差矩阵为

$$EX = \mu = [EX_1, \ldots, EX_p]^T, \quad \mathrm{Var}(X) = \Sigma = E\left[(X-\mu)(X-\mu)^T\right]$$

$\mathrm{Var}(X)$ 是 $p \times p$ 方阵，主对角线位置是各个分量的方差，(i,j) 位置是 X_i 与 X_j 的协方差 $\mathrm{Cov}(X_i, X_j)$。

设 x_t, $t = 1, 2, \ldots, T$ 为多元观测，则适当条件下（如独立同分布样本）可以估计多元分布的期望和方差：

$$\hat{\mu} = \frac{1}{T}\sum_{t=1}^{T} x_t, \quad \hat{\Sigma} = \frac{1}{T-1}\sum_{t=1}^{T}(x_t - \hat{\mu})(x_t - \hat{\mu})^T$$

这实际上就是一元的均值估计，一元的方差估计，和二元的协方差估计。

例如，从 1926-01 到 2011-09 的 IBM 和 S&P 的月度收益率构成多元数据，估计其均值：

```
x <- as.data.frame(coredata(ibmsp))
colMeans(x)
```

```
##         ibm          sp
## 0.013818251 0.005916719
```

计算协方差阵和相关系数阵：

```
cov(x)
```

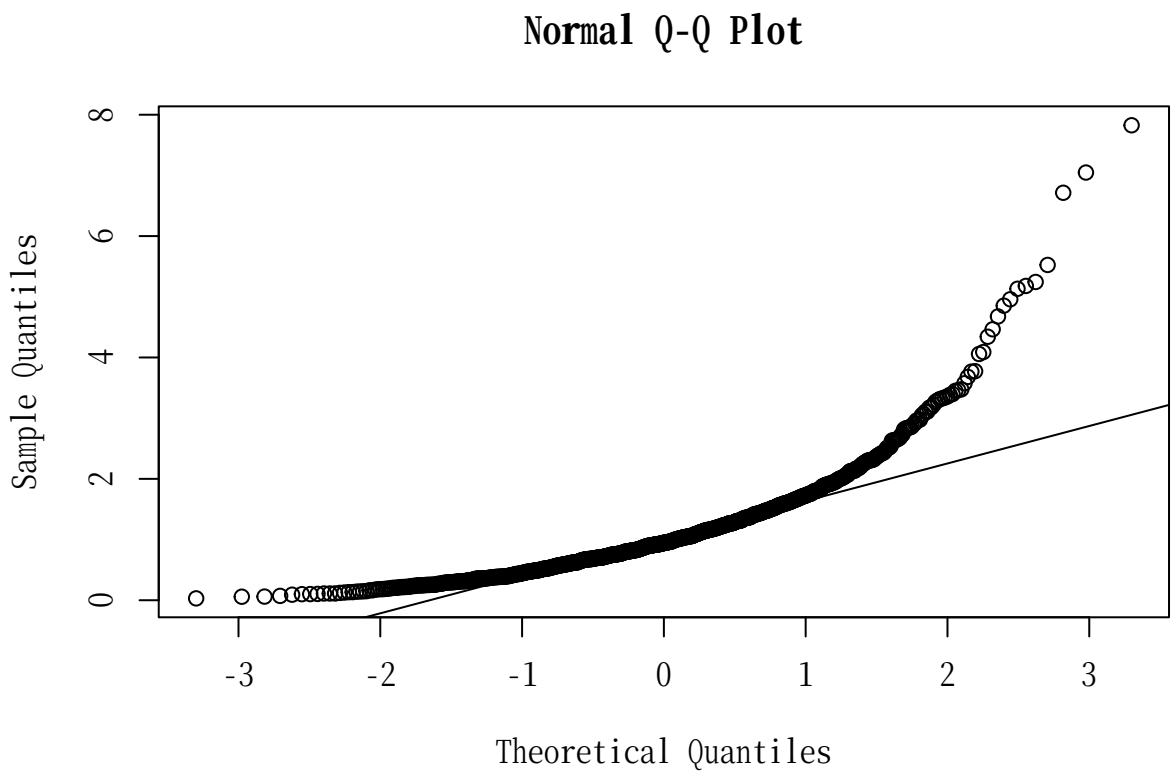
```
##             ibm          sp
## ibm 0.005008351 0.002502326
## sp  0.002502326 0.003056181
```

```
cor(x)
```

```
##           ibm        sp
## ibm 1.0000000 0.6395979
## sp  0.6395979 1.0000000
```

psych 包的 mardia() 函数进行推广的正态性偏度、峰度检验，并作广义 QQ 图：

```
psych::mardia(x)
```




```
## Call: psych::mardia(x = x)
```

```
##
## Mardia tests of multivariate skew and kurtosis
## Use describe(x) the to get univariate tests
## n.obs = 1029   num.vars =  2
## b1p =  0.27   skew =  46.8  with probability  <=  3.3e-09
##  small sample skew =  47.03  with probability <=  3e-09
## b2p =  21   kurtosis =  52.12  with probability <=  0
```

mvnormtest 包的 `msharpiro.test()` 可以执行多元的 Shapiro-Wilk 正态检验。零假设是服从多元正态分布。

```
mvnormtest::mshapiro.test(t(as.matrix(x)))
```

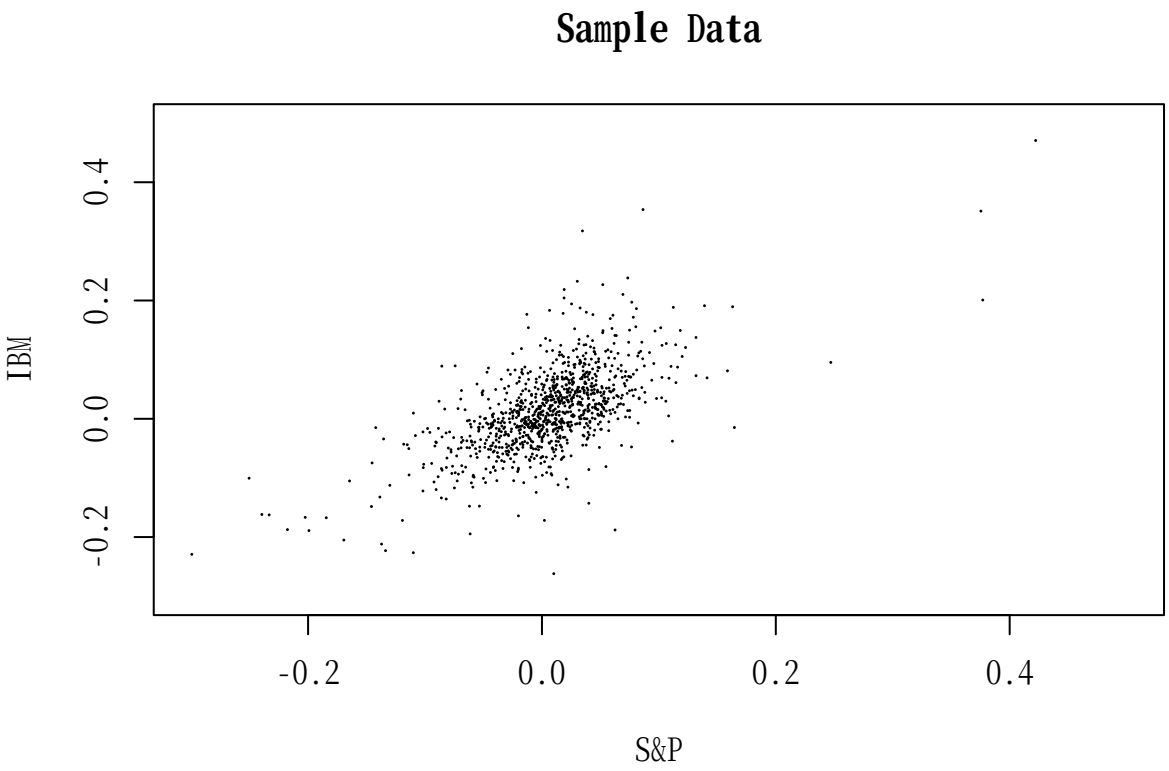
```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.92228, p-value < 2.2e-16
```

这些检验都显著地拒绝了正态性的零假设。

还可以模拟生成期望和协方差阵相同的多元正态随机样本数据，观察真实样本与模拟正态样本散点图的差别。mnormt 包的 `rmnorm()` 函数生成独立的多元正态随机样本。

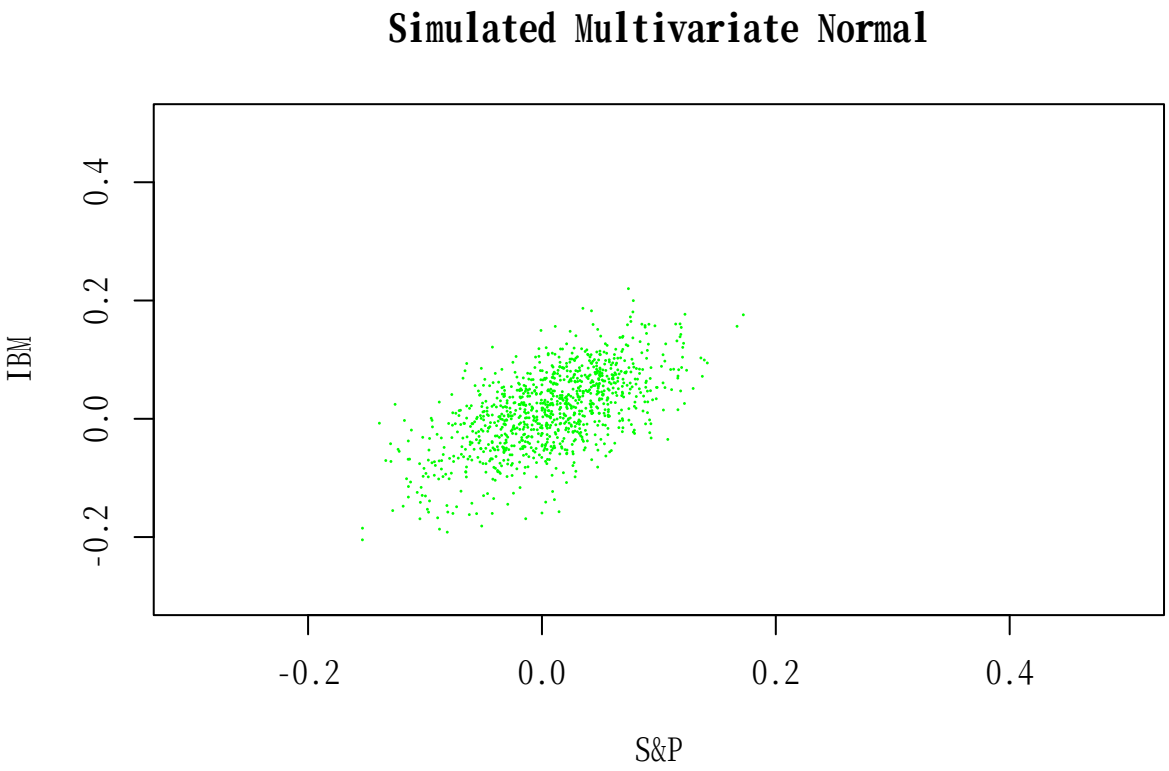
原始样本的散点图：

```
plot(x[["sp"]], x[["ibm"]], pch=16, cex=0.2,
     xlim=c(-0.3, 0.5), ylim=c(-0.3, 0.5),
     main="Sample Data",
     xlab="S&P", ylab="IBM")
```



生成一、二阶矩相同的正态分布样本，作散点图：

```
y <- mnormt::rmnorm(nrow(x), mean=colMeans(x), varcov=var(x))
plot(y[,2], y[,1], pch=16, cex=0.2, col="green",
     xlim=c(-0.3, 0.5), ylim=c(-0.3, 0.5),
     main="Simulated Multivariate Normal",
     xlab="S&P", ylab="IBM")
```



模拟数据分布集中得多。注意两个图形比较时，应采用相同的坐标范围。

Part II

一元线性时间序列模型

Chapter 3

线性时间序列模型

3.1 介绍

课程采用蔡瑞胸（Ruey S. Tsay）的《金融数据分析导论：基于 R 语言》(Tsay, 2013)（An Introduction to Analysis of Financial Data with R）作为主要教材之一。“线性时间序列模型” 这一部分是教材的第二章和第三章的授课笔记，本章讲授时间序列的线性模型，包括：

- 一些基本概念
- AR, MA, ARMA 模型
- 单位根过程
- 指数平滑
- 季节模型
- 回归模型中误差项有序列相关的处理
- 长记忆的分数阶差分模型
- 模型比较
- 实例分析

3.1.1 例子：苹果公司 2007 年到 2017 年股票日收盘价

```
chartSeries(
  AAPL, type="line", TA=NULL,
  subset="2003/2017",
  major.ticks="years", minor.ticks=FALSE,
  theme="white", name="Apple"
  )
```

股价序列呈现缓慢的、非单调的上升趋势，局部又有短暂的波动。

3.1.2 例子：可口可乐公司盈利季度数据

可口可乐公司每季度发布的每股盈利数据。读入：

图 3.1: 苹果公司股票日收盘价

```
da <- read_table(
  "q-ko-earns8309.txt",
  col_types=cols(
    .default = col_double(),
    pends = col_date("%Y%m%d"),
    anntime = col_date("%Y%m%d")
    ) )
ko.Rqtr <- xts(da[["value"]], da[["pends"]])
```

时间序列图：

```
chartSeries(
  ko.Rqtr, type="line", TA=NULL,
  major.ticks="years", minor.ticks=FALSE,
  theme="white", name="Coca Kola Quarterly Return"
  )
```

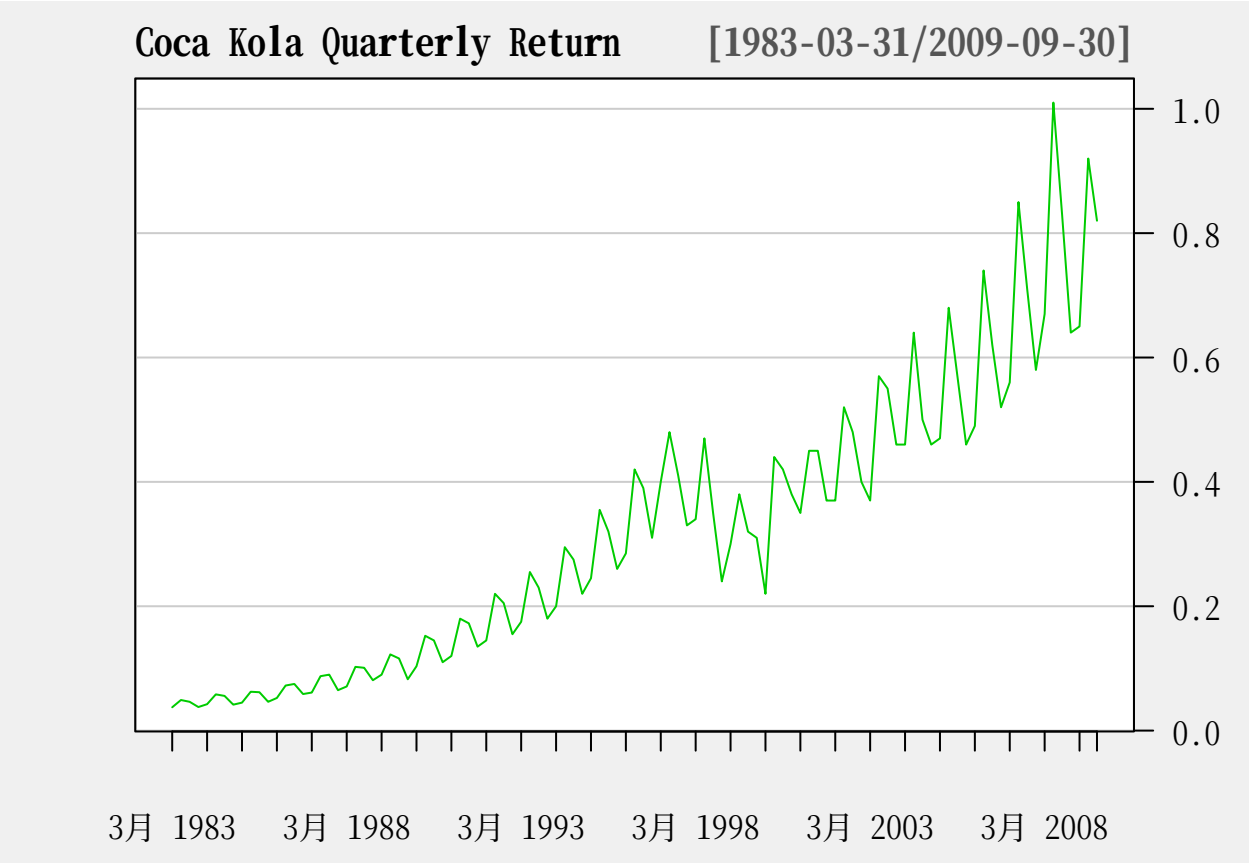
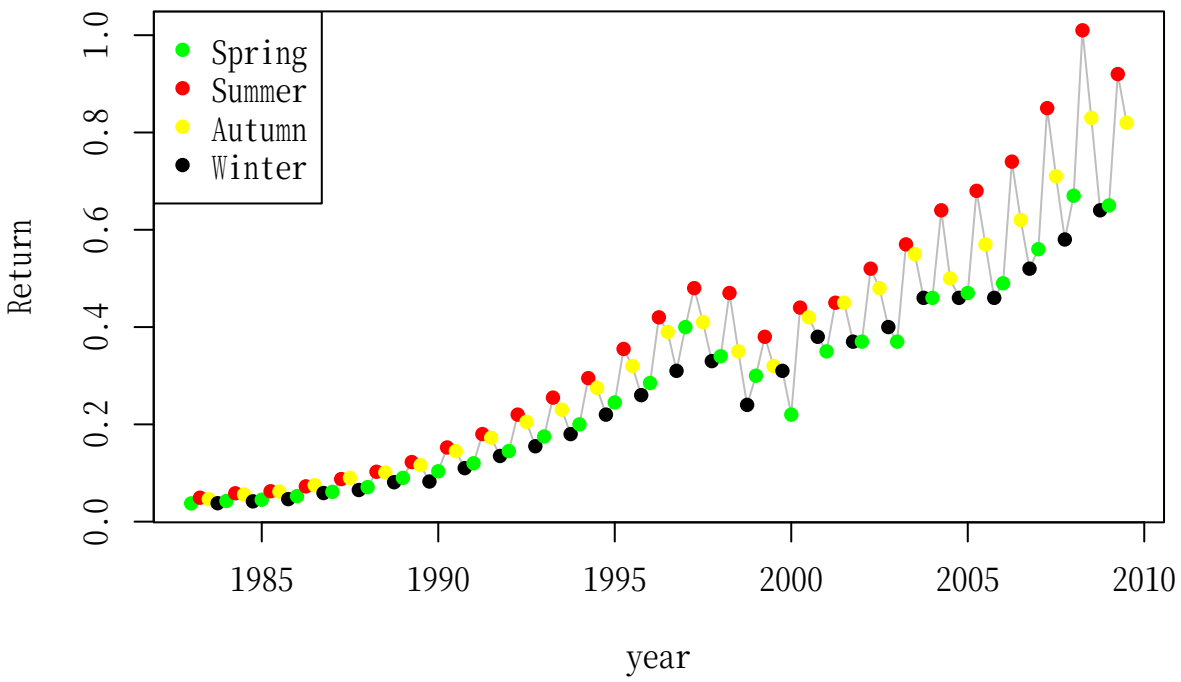


图 3.2: 可口可乐季度盈利

序列仍体现出缓慢的、非单调的上升趋势，又有明显的每年的周期变化（称为季节性），还有短期的波动。

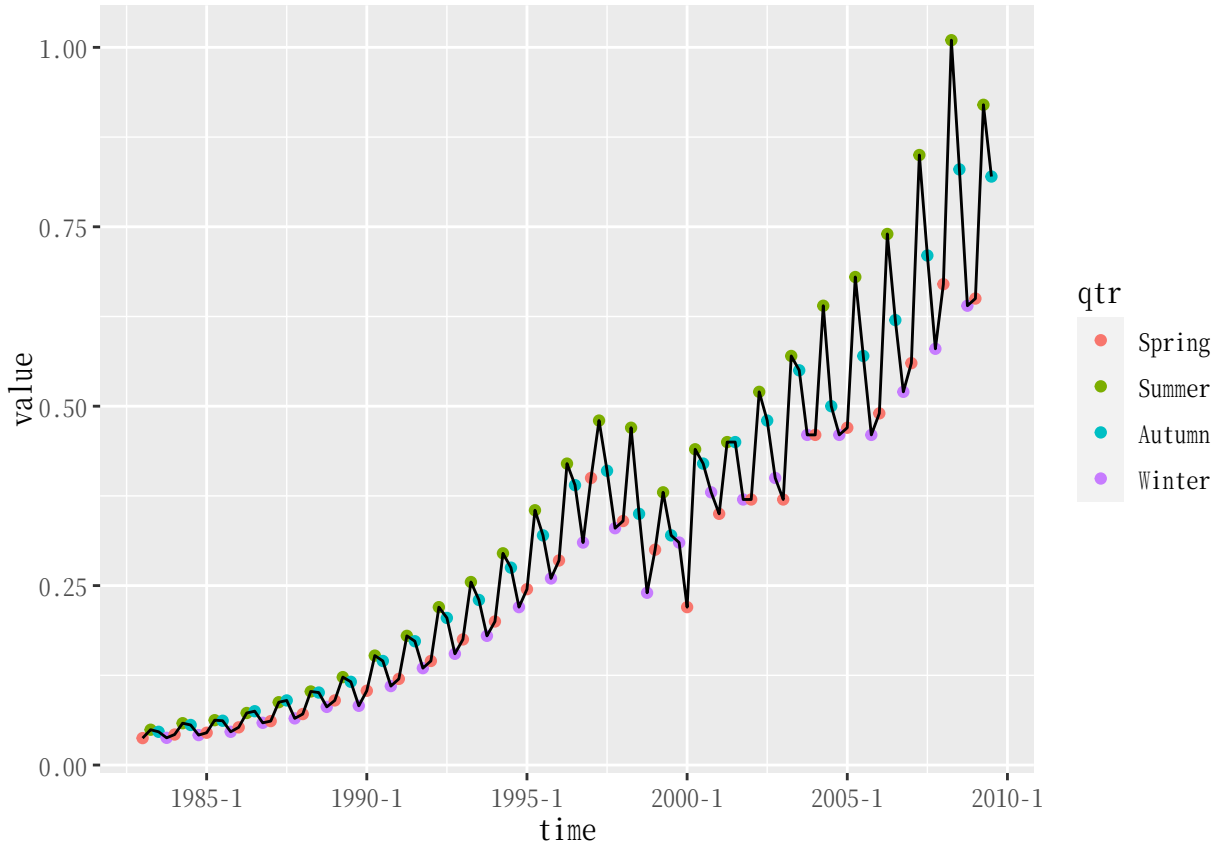
下面用基本 R 的 `plot()` 作图并用不同颜色标出不同季节。

```
tmp.x <- year(index(ko.Rqtr)) + (quarter(index(ko.Rqtr))-1)/4
tmp.y <- c(coredata(ko.Rqtr))
plot(tmp.x, tmp.y, type="l", col="gray",
     xlab="year", ylab="Return")
cpal <- c("green", "red", "yellow", "black")
points(tmp.x, tmp.y, pch=16,
       col=cpal[quarter(index(ko.Rqtr))])
legend("topleft", pch=16, col=cpal,
       legend=c("Spring", "Summer", "Autumn", "Winter"))
```



用 ggplot2 包绘图：

```
da_b <- da %>%
  mutate(time = as.yearqtr(pends),
    qtr = factor(quarter(pends),
       levels = 1:4,
       labels=c("Spring", "Summer", "Autumn", "Winter")))
ggplot(data = da_b, mapping = aes(
  x = time, y = value)) +
  geom_point(mapping=aes(color=qtr)) +
  geom_line()
```



现在可以看出，每年一般冬季和春季最低，夏季最高，秋季介于夏季和冬季之间。

3.1.3 例子：标普 500 指数月对数收益率

```
d <- read_table(
  "m-ibmsp-2611.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
sp.rmon <- xts(log(1 + d[["sp"]]), d[["date"]])
```

```
chartSeries(
  sp.rmon, type="line", TA=NULL,
  major.ticks="auto", minor.ticks=FALSE,
  theme="white", name="S&P 500 Monthly Returns"
  )
```

收益率在 0 上下波动，除了个别时候基本在某个波动范围之内。

3.1.4 例子：美国国债 3 月期和 6 月期周利率

```
d <- read_table2(
  "w-tb3ms.txt",
  col_types=cols(.default=col_double()))
```

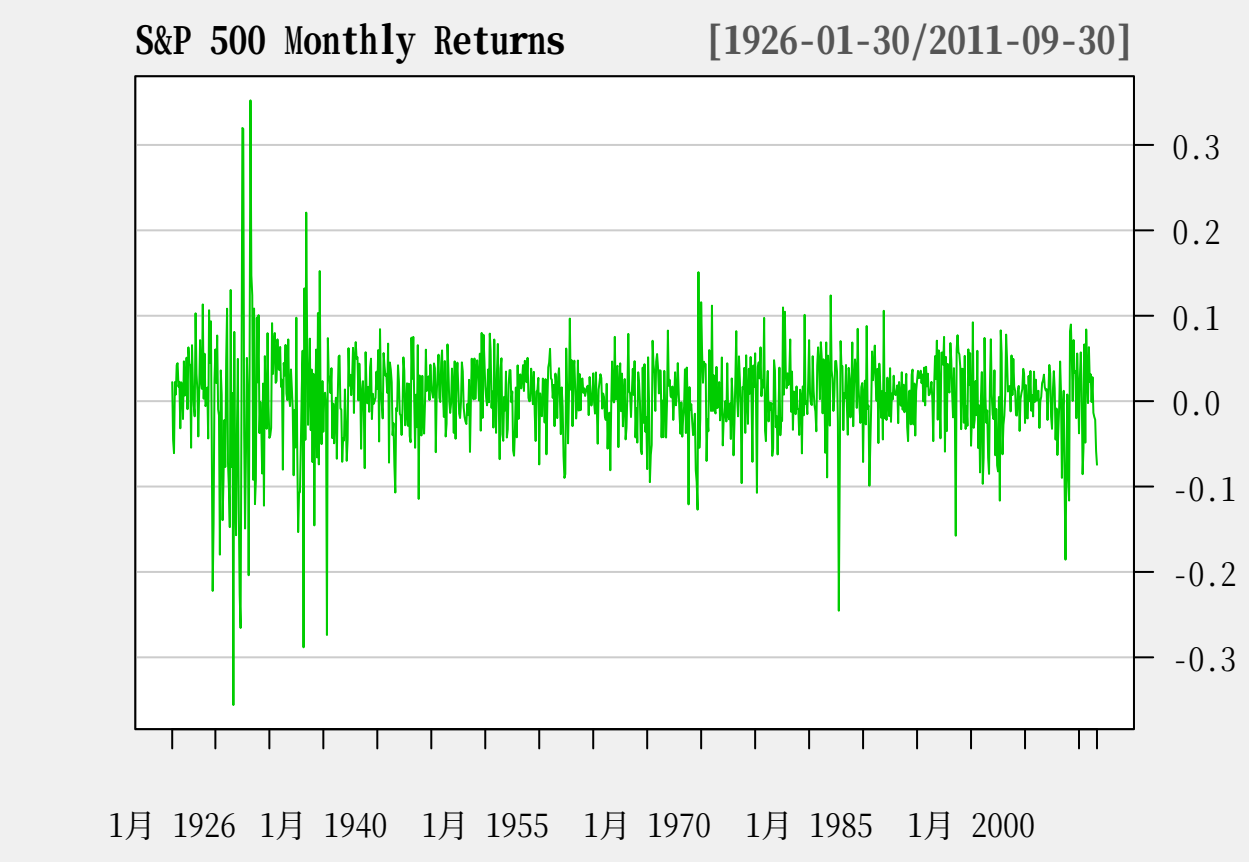


图 3.3: 标普 500 月收益率

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## Please use `read_table()` instead.
```

```
x1 <- xts(d[["rate"]], make_date(d[["year"]], d[["mon"]], d[["day"]]))
d <- read_table2(
  "w-tb6ms.txt",
  col_types=cols(.default=col_double()))
```

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## Please use `read_table()` instead.
```

```
x2 <- xts(d[["rate"]], make_date(d[["year"]], d[["mon"]], d[["day"]]))
tb36ms <- merge(x1, x2)
names(tb36ms) <- c("tb3ms", "tb6ms")
rm(d, x1, x2)
```

用 xts 包的 plot() 函数作图：

```
plot(tb36ms, type="l", grid.ticks.on="years")
```

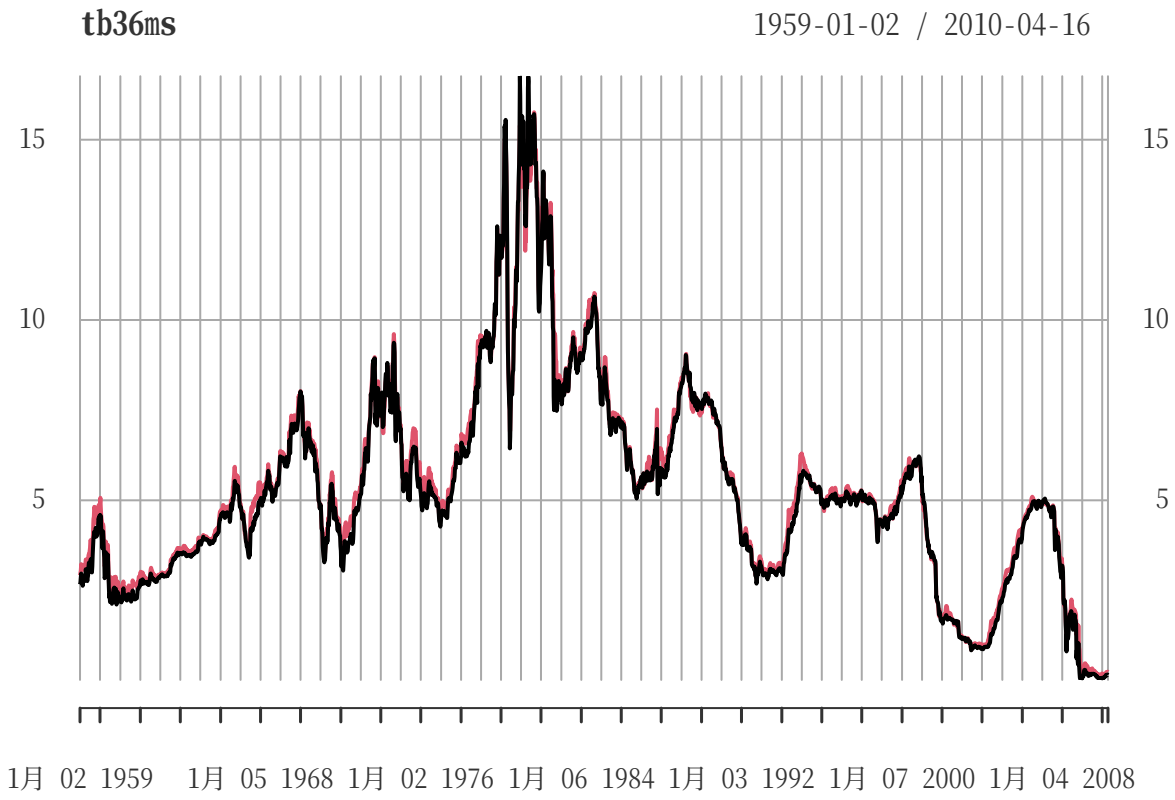
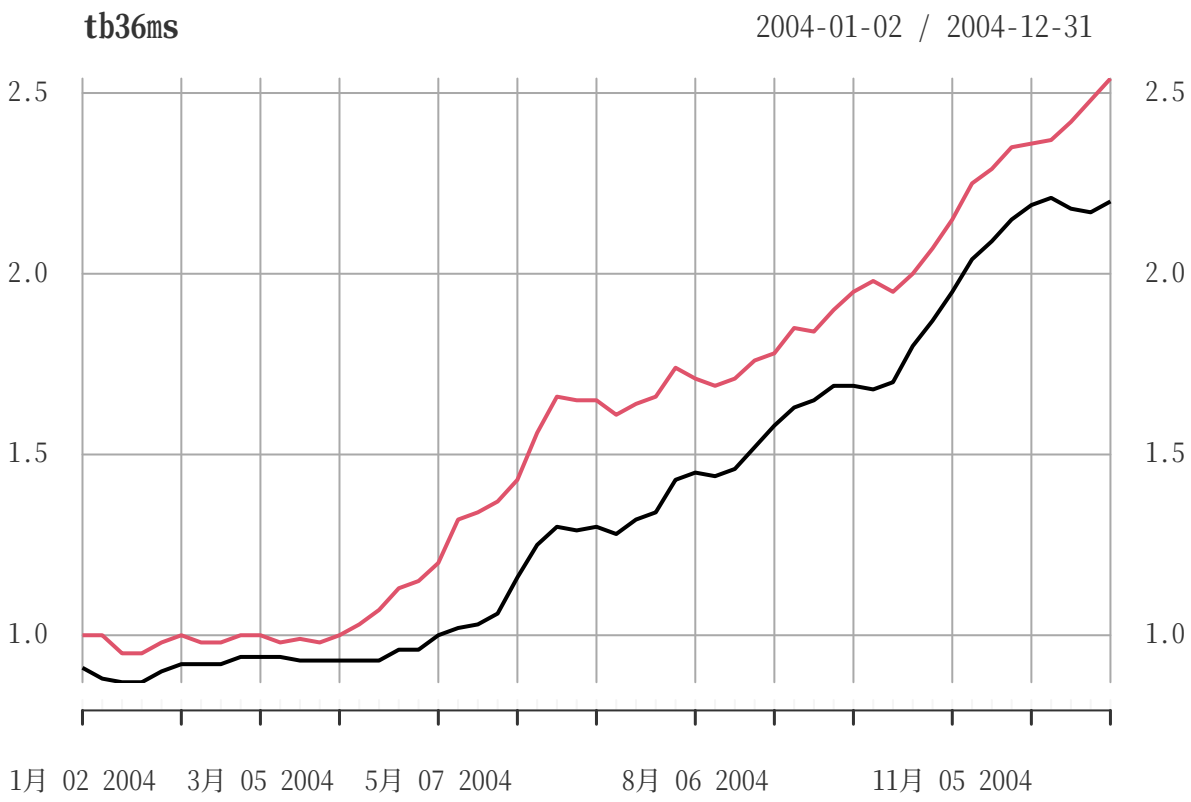


图 3.4: 美国 3 月期和 6 月期国债周利率

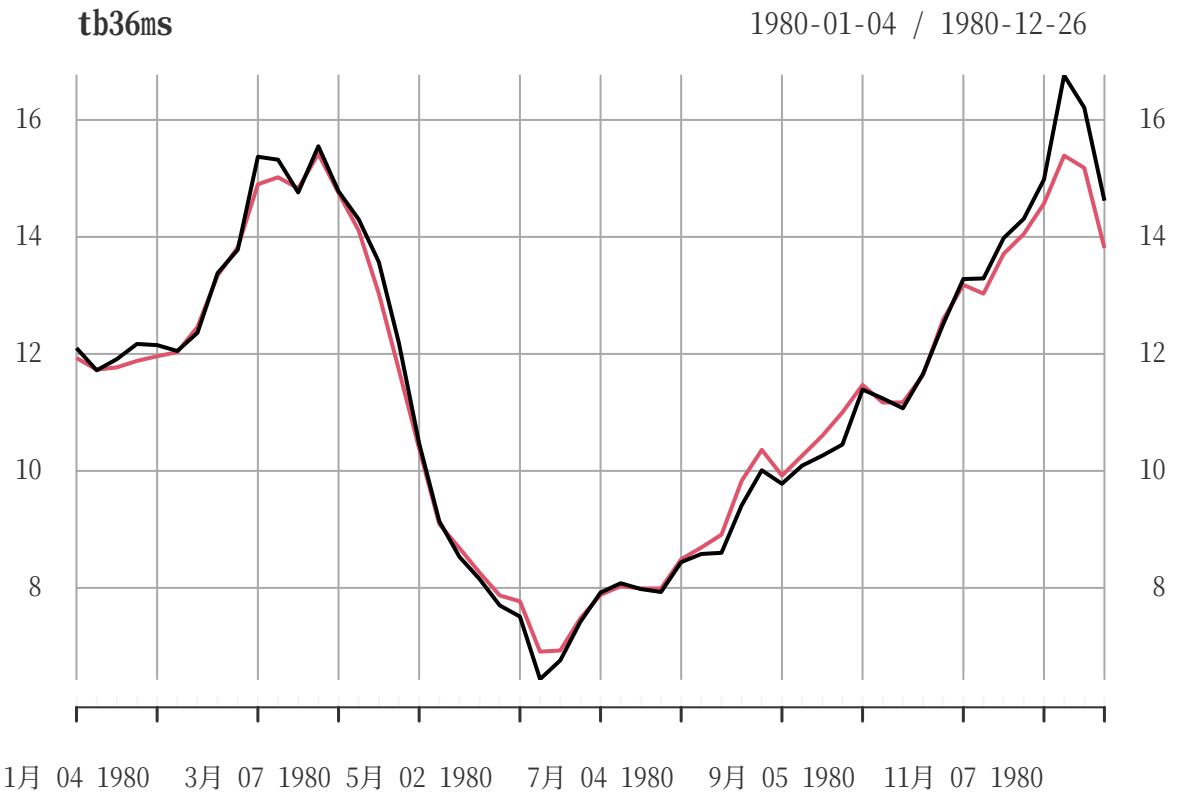
聚焦到 2004 年的数据：

```
plot(tb36ms, type="l", subset="2004")
```



红色是 6 月期国债利率，黑色是 3 月期国债。一般 6 月期高，但是有些时期 3 月期超过了 6 月期，如 1980 年：

```
plot(tb36ms, type="l", subset="1980")
```



3.2 平稳性

如图3.3那样的收益率数据基本呈现出在一个水平线（一般是 0）上下波动，且波动范围基本不变。这样的表现是时间序列“弱平稳序列”的表现。由弱平稳性，可以对未来的标普 500 收益率预测如下：均值在 0 左右，上下幅度在 ± 0.2 之间。

弱平稳需要一阶矩和二阶矩有限。记 $Ex_t = \mu$ 不变，$\gamma_0 = \mathrm{Var}(x_t) = E(x_t - \mu)^2$ 不变。某些分布是没有有限的二阶矩的，比如柯西分布，这样的分布就不适用传统的线性时间序列理论。

稍后给出弱平稳的理论定义。

如图3.2这样的价格序列则呈现出水平的上下起伏，如果分成几段平均的话，各段的平均值差距较大。这体现出非平稳的特性。

时间序列：设有随机变量序列 $\{x_t, t = \dots, -2, -1, 0, 1, 2, \dots\}$，称其为一个时间序列。其中 x_t 是一个随机变量，也可以写成大写的 X_t。时间序列 $\{X_t\}$ 严格来说是一个二元的函数 $X(t, \omega)$, $t \in \mathbb{Z}$($\mathbb{Z}$ 表示所有整数组成的集合)，$\omega \in \Omega$，Ω 表示在一定的条件下所有可能的试验结果的集合。经济和金融中的时间序列我们只能观察到其中某一个 $\omega_0 \in \Omega$ 对应的结果，称为一条“轨道”。而针对随机变量的许多理论性质都是在 $\omega \in \Omega$ 上讨论的，比如 $EX_t = \int X_t(\omega) P(d\omega)$ 是 $X_t(\omega)$ 对 $\omega \in \Omega$ 的平均。

为了能够用一条轨道的观测样本得到所有 $\omega \in \Omega$ 的性质，需要时间序列满足“遍历性”。

时间序列的样本：设 $\{x_t, t = 1, 2, \dots, T\}$ 是时间序列中的一段。仍将 x_t 看成随机变量，也可以写成大写的 X_t。如果有了具体数值，那么样本就是一条轨道中的一段。

自协方差函数：时间序列 $\{X_t\}$ 中两个随机变量的协方差 $\mathrm{Cov}(X_s, X_t)$ 叫做自协方差。如果 $\mathrm{Cov}(X_s, X_t) = \gamma_{|t-s|}$ 仅依赖于 $t-s$，则称

$$\gamma_k = \mathrm{Cov}(X_{t-k}, X_t), k = 0, 1, 2, ...$$

为时间序列 $\{X_t\}$ 的自协方差函数。因为 $\mathrm{Cov}(X_s, X_t) = \mathrm{Cov}(X_t, X_s)$，所以 $\gamma_{-k} = \gamma_k$。易见 $\gamma_0 = \mathrm{Var}(X_t)$。

由 Cauchy-Schwartz 不等式，

$$|\gamma_k| = |E[(X_{t-k} - \mu)(X_t - \mu)]| \le \left(E(X_{t-k} - \mu)^2 \, E(X_t - \mu)^2\right)^{1/2} = \gamma_0$$

弱平稳序列 (宽平稳序列，weakly stationary time series): 如果时间序列 $\{X_t\}$ 存在有限的二阶矩且满足：

(1) $EX_t = \mu$ 与 t 无关；

(2) $\mathrm{Var}(X_t) = \gamma_0$ 与 t 无关;

(3) $\gamma_k = \mathrm{Cov}(X_{t-k}, X_t)$, $k = 1, 2, ...$ 与 t 无关，

则称 $\{X_t\}$ 为弱平稳序列。

适当条件下可以用时间序列的样本估计自协方差函数，这是用一条轨道的信息推断所有实验结果 Ω，估计公式为

$$\hat{\gamma}_k = \frac{1}{T} \sum_{t=k+1}^{T} (x_{t-k} - \bar{x})(x_t - \bar{x}), k = 0, 1, ..., T-1$$

称 $\hat{\gamma}_k$ 为样本自协方差。注意这里用了 $1/T$ 而不是 $1/(T-k)$，用 $1/(T-k)$ 在获得无偏性的同时会造成一些理论上的困难。

3.3 相关系数和自相关函数

3.3.1 相关系数

图3.5是 IBM 股票月度简单收益率对标普 500 收益率的散点图。从图中看出，两者有明显的正向相关关系。

两个随机变量 X 和 Y 的相关系数定义为

$$\rho(X, Y) = \rho_{xy} = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}} = \frac{E[(X - \mu_x)(Y - \mu_y)]}{\sqrt{E(X - \mu_x)^2 E(Y - \mu_y)^2}}$$

如果有 (X, Y) 的独立同分布样本 (x_t, y_t)，$t = 1, 2, ..., T$，可估计相关系数为

$$\hat{\rho}_{xy} = \frac{\sum_{t=1}^{T}(x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum_{t=1}^{T}(x_t - \bar{x})^2 \sum_{t=1}^{T}(y_t - \bar{y})^2}}$$

对于不独立的样本，比如时间序列样本，也可以计算相关系数，其估计合理性需要一些模型假设。

对于联合分布非正态的情况，有时相关系数不能很好地反映 X 和 Y 的正向或者负向的相关。斯皮尔曼（Spearman）相关系数是计算 X 的样本的秩（名次）与 Y 的样本的秩之间的相关系数，也称为 Spearman rank correlation。

另一种常用的非参数相关系数是肯德尔 tau(Kendall's τ) 系数，反映了一致数对和非一致数对之间的差别。对随机向量 (X, Y)，设 (X_1, Y_1), (X_2, Y_2) 相互独立且联合分布与 (X, Y) 联合分布相同，定义 X 和 Y 的肯德尔 tau 系数为

$$\tau = P\left[(X_1 - X_2)(Y_1 - Y_2) > 0\right] - P\left[(X_1 - X_2)(Y_1 - Y_2) < 0\right]$$

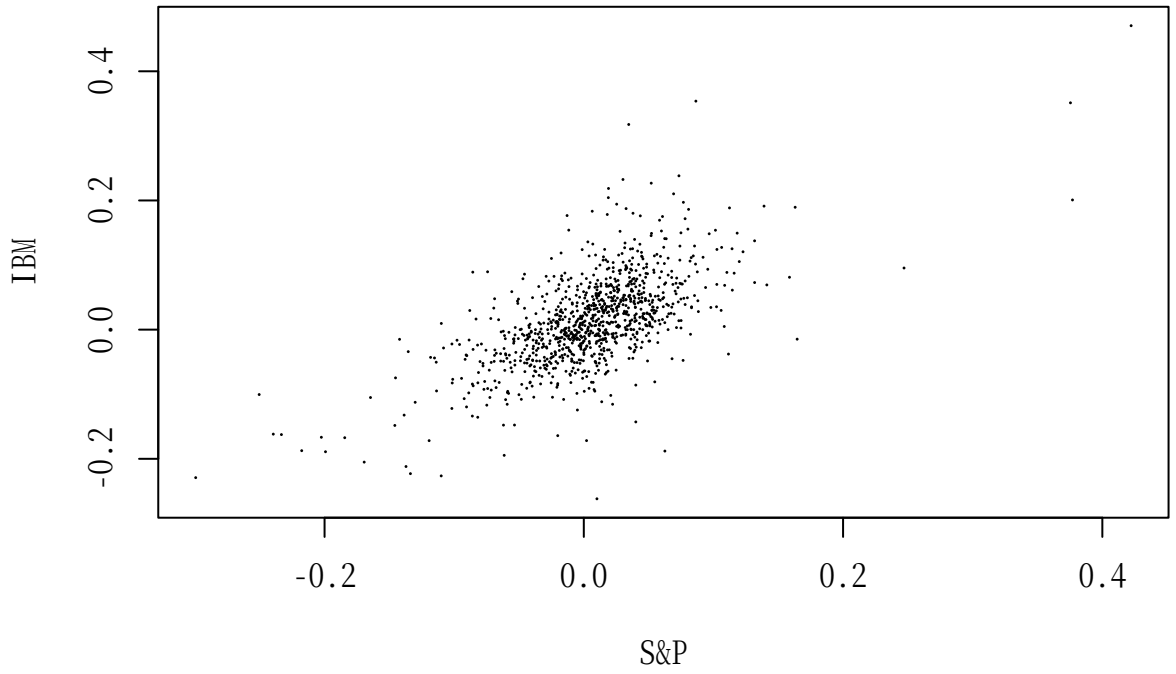


图 3.5: IBM 对标普 500 月度简单收益率

即两个观测的分量次序一致的概率减去分量次序相反的概率。一致的概率越大，说明两个的正向相关性越强。

对 IBM 收益率与标普收益率数据计算这三种相关系数：

```
cor(d[,"sp"], d[,"ibm"])
```

```
## [1] 0.6395979
```

```
cor(d[,"sp"], d[,"ibm"], method="spearman")
```

```
## [1] 0.6065789
```

```
cor(d[,"sp"], d[,"ibm"], method="kendall")
```

```
## [1] 0.4328066
```

3.3.2 自相关函数与白噪声

设 $\{X_t\}$ 为弱平稳序列，$\{\gamma_k\}$ 为自协方差函数。则

$$\rho(X_{t-k}, X_t) = \frac{\mathrm{Cov}(X_{t-k}, X_t)}{\sqrt{\mathrm{Var}(X_{t-k})\mathrm{Var}(X_t)}} = \frac{\gamma_k}{\sqrt{\gamma_0\gamma_0}} = \frac{\gamma_k}{\gamma_0},\ k = 0, 1, \dots,\ \forall t$$

记 $\rho_k = \gamma_k/\gamma_0$，这是 X_{t-k} 与 X_t 的相关系数且与 t 无关，称 $\{\rho_k, k = 0, 1, \dots\}$ 为时间序列 $\{X_t\}$ 的自相关函数（Autocorrelation function, ACF）。$\rho_0 = 1$。

如果弱平稳序列 $\{X_t\}$ 满足 $\rho_k = 0,\ k = 1, 2, \dots$，称 $\{X_t\}$ 为**白噪声序列**。如果随机变量序列 $\{X_t\}$ 独立且期望和方差不随时间而变，则 $\{X_t\}$ 是白噪声序列，称为独立白噪声。如果独立白噪声还是同分布的，称为独立同分布白噪声。

适当条件下 ρ_k 可以从时间序列样本估计为

$$\hat{\rho}_k = \frac{\hat{\gamma}_k}{\hat{\gamma}_0},\ k = 0, 1, \dots$$

$\hat{\rho}_0 = 1$。称 $\hat{\rho}_k, k = 1, 2, \dots$ 为**样本自相关函数**。

如果时间序列严平稳遍历，则 $\hat{\rho}_k$ 是 ρ_k 的强相合估计。

若 $\{X_t\}$ 为有二阶矩的独立同分布随机变量列，则 $\hat{\rho}_k (k > 0)$ 渐近服从 $\mathrm{N}(0, \frac{1}{T})$。

如果 $\{\varepsilon_t\}$ 是零均值独立同分布白噪声，q 为非负整数，$\{\psi_j, j = 0, 1, \dots, q\}$ 是数列，$\psi_0 = 1$，

$$X_t = \mu + \sum_{j=0}^{q} \psi_j \varepsilon_{t-j},\ t \in \mathbb{Z},$$

则从 $\{X_t, t = 1, \dots, T\}$ 计算的 ACF 满足：当 $k > q$ 时，$\sqrt{T}\hat{\rho}_k$ 渐近服从 $\mathrm{N}(0, 1 + 2\sum_{j=0}^{q} \rho_j^2)$, 这称为 Bartlett 公式。参见 (何书元, 2003) P.131 §4.2 的例 2.1。原始文献：MAURICE STEVENSON BARTLETT, On the Theoretical Specification and Sampling Properties of Auto-Correlated Time Series, Journal of the Royal Statistical Society (Supplement) 8 (1946), pp. 24-41.

在基本 R 软件中，`acf(x)` 可以估计时间序列 `x` 的自相关函数并对其前面若干项画图。

例 3.1. 例：CRSP 的第 10 分位组合的月对数收益率，1967-1 到 2009-12。第 10 分位组合是 NYSE、AMEX、NASDAQ 市值最小的 10% 股票组成的投资组合，每年都重新调整。

- CRSP 是 Center for Research in Security Prices, 位于 Chicago Booth。

- NYSE(The New York Stock Exchange, 纽约证券交易所),
- AMEX(American Stock Exchange, 美国证券交易所，在纽约华尔街附近),
- NASDAQ(National Association of Securities Dealers Automated Quotations，纳斯达克，位于纽约)。

```
d <- read_table2(
  "m-dec12910.txt", col_types=cols(
    .default=col_double(),
    date=col_date(format="%Y%m%d")))
```

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## i Please use `read_table()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
dec <- xts(as.matrix(d[,-1]), d$date)
tclass(dec) <- "yearmon"
d10 <- ts(coredata(dec)[,"dec10"], start=c(1967,1), frequency=12)
plot(d10, main="CRSP Lower 10% Mothly Returns")
```

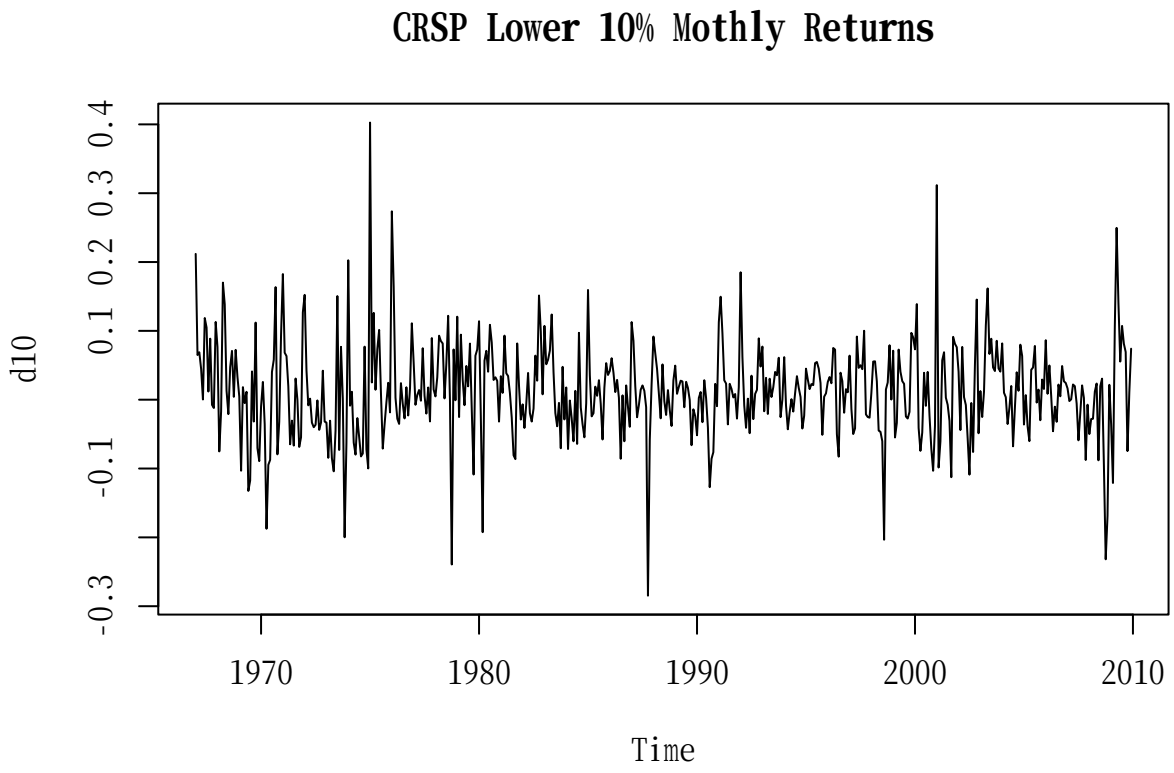


图 3.6: CRSP 第 10 分位组合月对数收益率

用 `acf()` 作时间序列的自相关函数图：

```
acf(d10)
```

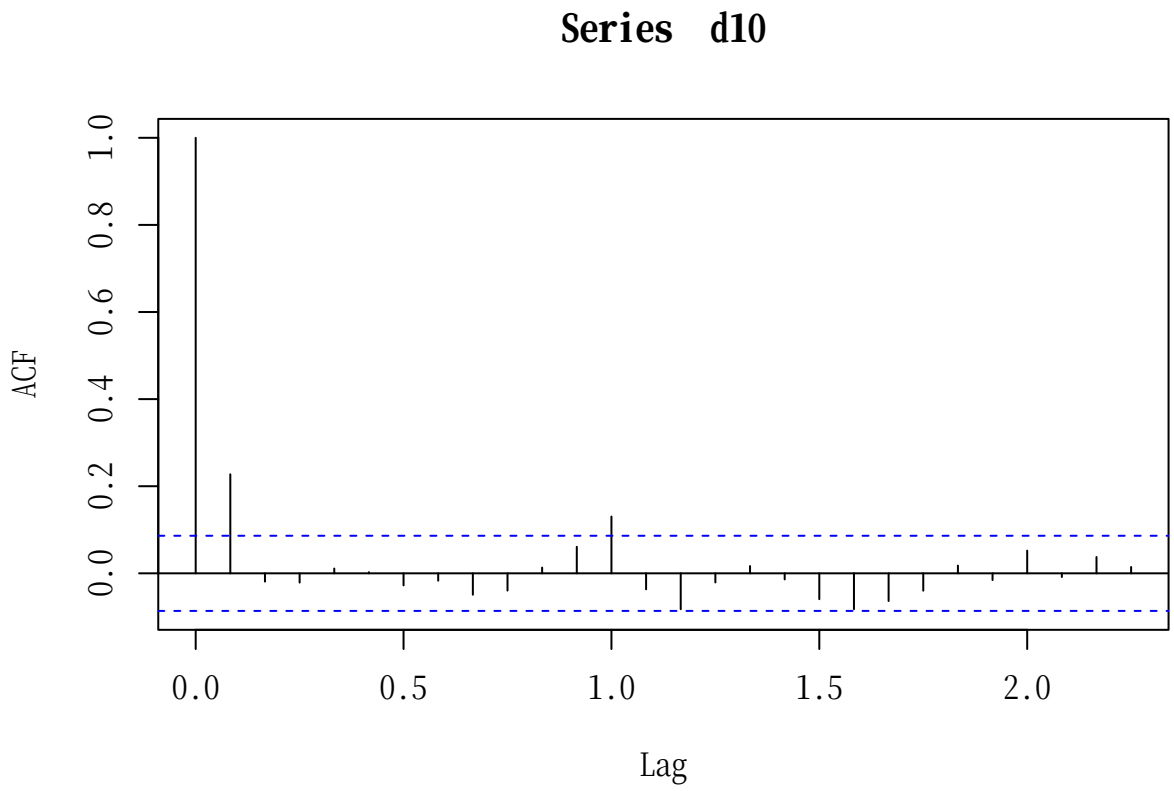


图 3.7: CRSP 第 10 分位组合月对数收益率的 ACF

`acf()` 的返回值是一个列表，其中 `lag` 相当于 k，`acf` 相当于 $\hat{\rho}_k$。用 `plot=FALSE` 取消默认的图形输出。

ACF 图中横轴上下两条水平下是在独立同分布白噪声假设下的加减两倍标准差，即 $\pm\frac{2}{\sqrt{T}}$。如果独立同分布白噪声假设成立，每个 $\hat{\rho}_k$ 有 95% 以上的概率落入这两条线之间。

ACF 图 $k=0$ 处总对应 $\hat{\rho}_0=1$。

上图的 $\hat{\rho}_1$ 和 $\hat{\rho}_{12}$ 都超出了界限（因为是月度数目，横轴的单位是 1/12 为一个时间点）。从此图可以认为此投资组合的收益率不是白噪声。

标准库 stats 的 `acf()` 作自相关函数图总是有 $\rho_0=1$，这在其它系数很小时会使得其它系数显得很难分辨，另外，当时间序列是月度或季度数据时，横坐标是以年为单位。所以，forecast 包提供了一个类似功能的 `Acf()` 函数，如：

```
forecast::Acf(d10)
```

TSA 包的 `acf()` 函数也不做滞后 0 的点，如：

```
TSA::acf(d10)
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
```

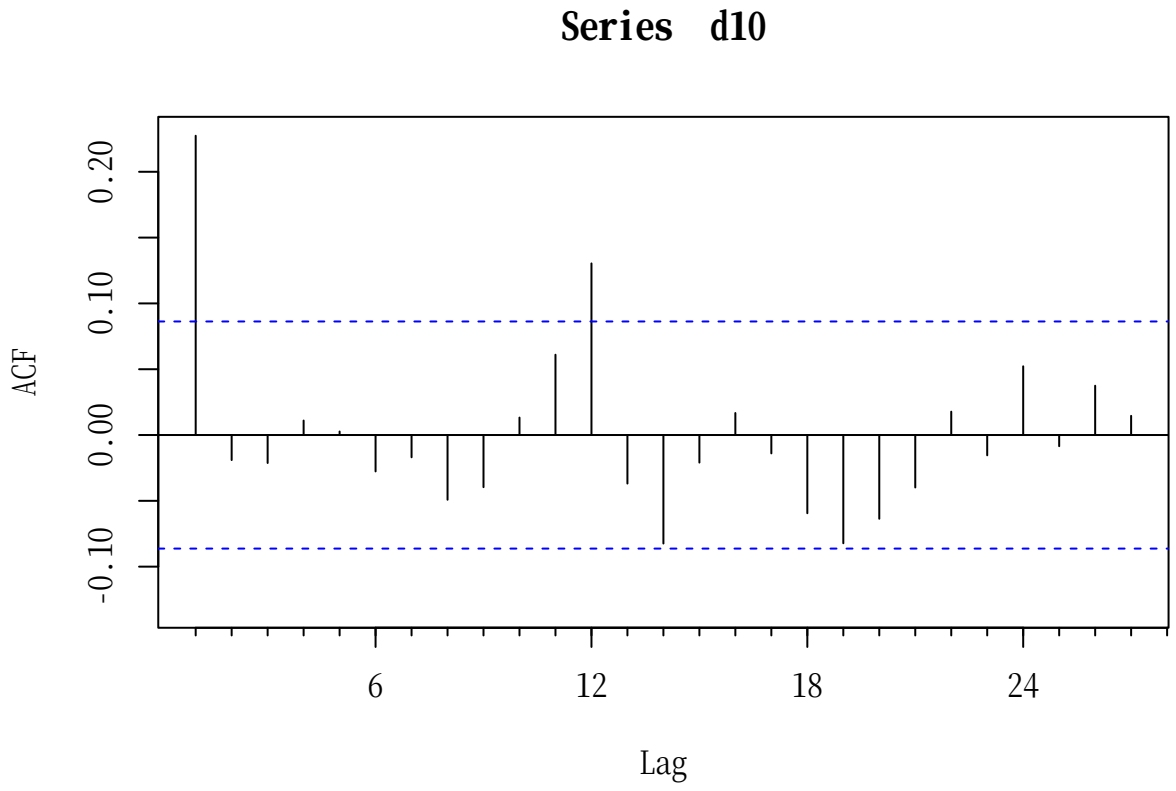


图 3.8: CRSP 第 10 分位组合月对数收益率的 ACF

```
##   plot.Arima   forecast
```

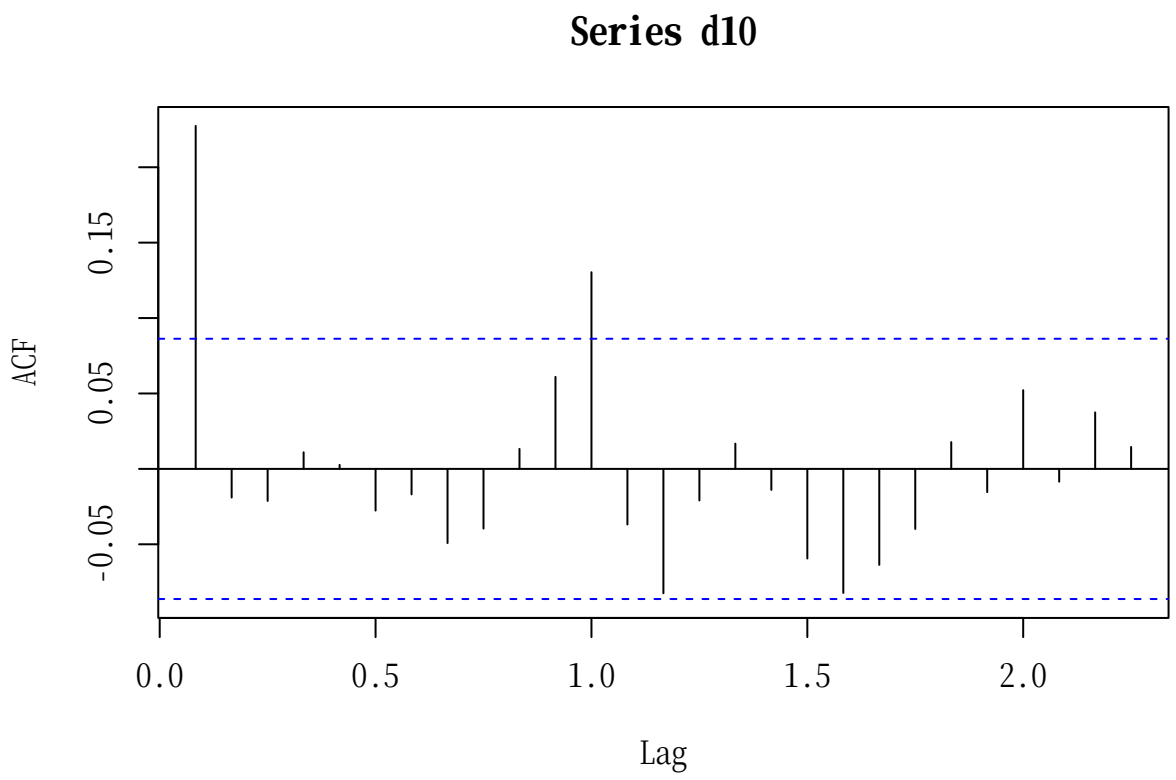


图 3.9: CRSP 第 10 分位组合月对数收益率的 ACF

3.3.3 用单个自相关系数作白噪声检验

如果 $\{X_t\}$ 是独立同分布白噪声，则 $\hat{\rho}_k(k \geq 1)$ 近似 $\mathrm{N}(0, 1/T)$。若 H_0 是序列为白噪声，取统计量

$$t = \sqrt{T}\hat{\rho}_k$$

如果 $|t| > \mathrm{qnorm}(1-\alpha/2)$，则拒绝白噪声零假设。实际中常取 $\alpha = 0.05$, $\mathrm{qnorm}(1-\alpha/2) \approx 2$，当 $\hat{\rho}_1$ 超出 $\pm 2/\sqrt{T}$ 则拒绝 H_0，有多个 $\hat{\rho}_k$ 超出 $\pm 2/\sqrt{T}$ 也可拒绝 H_0，有一个 t 统计量值很大（比如超出 ± 3）也可拒绝 H_0。

在判断 $\{X_t\}$ 是否 $X_t = \mu + \sum_{j=0}^{q} \psi_j \varepsilon_{t-j}$ 这样的模型时，根据 Bartlet 公式，可取

$$t = \frac{\hat{\rho}_k}{\sqrt{\frac{1}{T}\left(1 + 2\sum_{j=1}^{k-1} \hat{\rho}_j^2\right)}},\ k > q$$

当 t 超出 $\mathrm{qnorm}(1-\alpha/2)$ 时拒绝这样的模型。

例 3.2. 这是例3.1 的继续。有研究者认为小市值股票倾向于在每年的一月份有正的收益率。

为此，用 $H_0:\rho_{12}=0$ 对 $H_a:\rho_{12}\neq 0$ 的检验来验证。如果一月份有取正值的倾向，则相隔 12 个月的值会有正相关。

计算 $\hat{\rho}_{12}$ 的值：

```
tmp1 <- acf(d10, plot=FALSE)
r12 <- tmp1$acf[abs(tmp1$lag-12/12)<1E-10]
r12
```

```
## [1] 0.130411
```

计算 t 统计量的值，检验 p 值：

```
t12 <- sqrt(tmp1$n.used)*r12; t12
```

```
## [1] 2.962369
```

```
pv <- 2*(1 - pnorm(abs(t12))); pv
```

```
## [1] 0.003052812
```

p 值小于 0.05, 拒绝了 $H_0:\rho_{12}=0$，这个检验的结果支持一月份效应的存在性。

3.3.4 Ljung-Box 白噪声检验

为了检验时间序列样本是否来自白噪声序列，可以检验 $\rho_k=0,k=1,2,...$ 的零假设。前面检验单个 ρ_k 的做法如果针对多个进行检验就有多重检验的第一类错误增大的问题。

Box 和 Pierce(Box & Pierce, 1970) 提出了混成统计量 (Portmanteau statistic)

$$Q_*(m)=T\sum_{j=1}^{m}\hat{\rho}_j^2$$

用来检验

$$H_0:\rho_1=\cdots=\rho_m=0\longleftrightarrow H_a:\text{不全为零}$$

在 $\{X_t\}$ 是独立白噪声序列条件下，$Q_*(m)$ 近似服从 $\chi^2(m)$ 分布。给定检验水平 α，当 $Q_*(m)>\text{qchisq}(1-\alpha,m)$ 时拒绝 H_0，否定白噪声假设。如果检验的序列是线性时间序列估计的残差序列，则卡方自由度应改为 m 减去估计的系数个数。

Ljung 和 Box(Ljung & Box, 1978) 对此检验方法进行了改进。统计量改为

$$Q(m)=T(T+2)\sum_{j=1}^{m}\frac{\hat{\rho}_j^2}{T-j}$$

在独立同分布白噪声假设下仍近似服从 $\chi^2(m)$ 分布。当 $Q(m)>\text{qchisq}(1-\alpha,m)$ 时拒绝 H_0，否定白噪声假设。这个检验称为 Ljung-Box 白噪声检验。如果检验的序列是线性时间序列估计的残差序列，则卡方自由度应改为 m 减去估计的系数个数。比如，对 ARMA(p,q) 模型建模的残差作白噪声检验，卡方自由度应改为 $m-(p+q)$。

在 R 软件中，`Box.test(x, type="Ljung-Box")` 执行 Ljung-Box 白噪声检验。`Box.test(x, type="Box-Pierce")` 执行 Box-Pierce 混成检验。用 `fitdf=` 指定要减去的自由度个数。

R 的 portes 包提供了多种一元和多元白噪声检验功能。

例 3.3. 检验 IBM 股票月收益率是否白噪声。

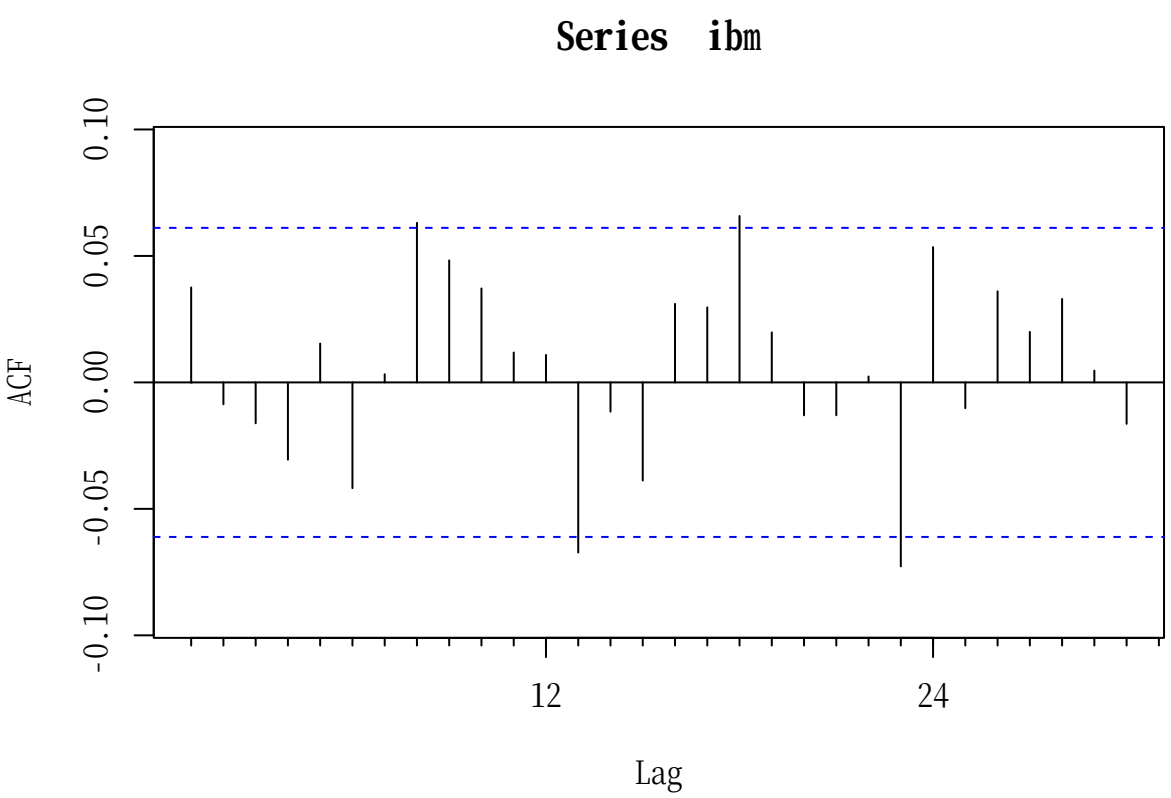
考虑 IBM 股票从 1926-01 到 2011-09 的月度收益率数据，简单收益率和对数收益率分别考虑。

读入数据：

```
d <- read_table(
  "m-ibmsp-2611.txt", col_types=cols(
    .default=col_double(),
    date=col_date(format="%Y%m%d")))
ibm <- ts(d[["ibm"]], start=c(1926,1), frequency=12)
```

读入的是简单收益率的月度数据。作 ACF 图：

```
forecast::Acf(ibm)
```



从 ACF 来看月度简单收益率是白噪声。

作 Ljung-Box 白噪声检验，分别取 $m = 12$ 和 $m = 24$：

```
Box.test(ibm, lag=12, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
```

```
## data:  ibm
## X-squared = 13.098, df = 12, p-value = 0.362
```

```
Box.test(ibm, lag=24, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ibm
## X-squared = 35.384, df = 24, p-value = 0.0629
```

在 0.05 水平下均不拒绝零假设，支持 IBM 月度简单收益率是白噪声的零假设。

从简单收益率计算对数收益率，并进行 LB 白噪声检验：

```
Box.test(log(1 + ibm), lag=12, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  log(1 + ibm)
## X-squared = 12.814, df = 12, p-value = 0.3827
```

```
Box.test(log(1 + ibm), lag=24, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  log(1 + ibm)
## X-squared = 34.506, df = 24, p-value = 0.07607
```

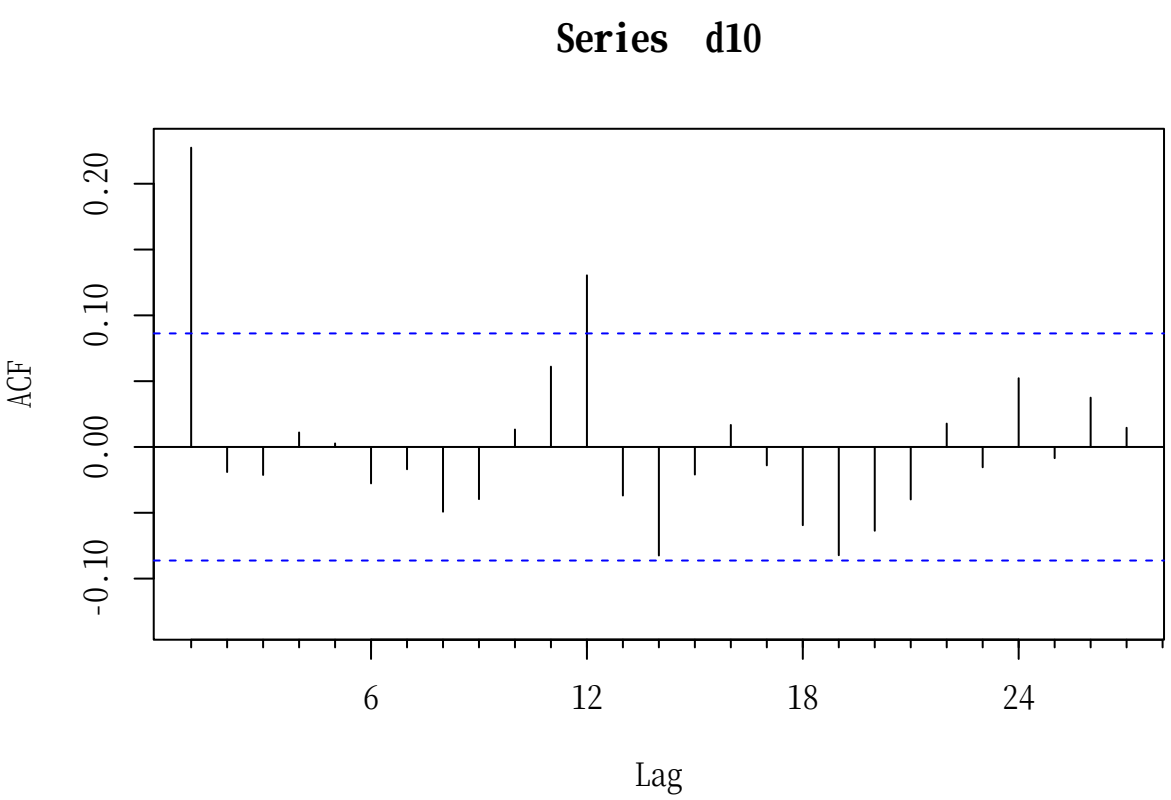
在 0.05 水平下不拒绝零假设。

Box-Pierce 检验和 Ljung-Box 检验受到 m 取值的影响，建议采用 $m \approx \ln T$，且序列为季度、月度这样的周期序列时，m 应取为周期的整数倍。

例 3.4. 这是例3.1和例3.2的继续。对 CRSP 最低 10 分位的资产组合的月简单收益率作白噪声检验。

此组合的收益率序列的 ACF:

```
forecast::Acf(d10)
```



针对 $m = 12$ 和 $m = 24$ 作 Ljung-Box 白噪声检验：

```
Box.test(d10, type="Ljung-Box", lag=12)
```

```
##
##  Box-Ljung test
##
## data:  d10
## X-squared = 41.06, df = 12, p-value = 4.789e-05
```

```
Box.test(d10, type="Ljung-Box", lag=24)
```

```
##
##  Box-Ljung test
##
## data:  d10
## X-squared = 56.246, df = 24, p-value = 0.0002122
```

在 0.05 水平下均拒绝零假设，认为 CRSP 最低 10 分位的投资组合的月度简单收益率不是白噪声。

有效市场假设认为收益率是不可预测的，也就不会有非零的自相关。但是，股价的决定方式和指数收益率的计算方式等可能会导致在观测到的收益率序列中有自相关性。高频金融数据中很常见自相关性。

常见的白噪声检验还有 TREVOR S. BREUSCH (1978) 和 LESLIE G. GODFREY (1978) 提出的拉格朗日乘子法检验（LM 检验）。零假设为白噪声，对立假设为 AR、MA 或者 ARMA。参见：

- TREVOR S. BREUSCH(1978), Testing for Autocorrelation in Dynamic Linear Models, Australian Economic Papers 17, pp. 334 – 355
- LESLIE G. GODFREY(1978), Testing Against General Autoregressive and Moving Average Error Models When Regressors Include Lagged Dependent Variables, Econometrica 46 , S. 1293 – 1302

3.4 线性时间序列

设 $\{X_t\}$ 是独立同分布的二阶矩有限的随机变量，称 $\{X_t\}$ 为**独立同分布白噪声** (white noise)。最常用的白噪声一般假设均值为零。如果 $\{X_t\}$ 独立同 $\mathrm{N}(0,\sigma^2)$ 分布，称 $\{X_t\}$ 为高斯 (Gaussian) 白噪声或正态白噪声。

白噪声序列的自相关函数为零（$\rho_0 = 1$ 除外）。

实际应用中如果样本自相关函数近似为零（ACF 图中都位于控制线之内或基本不超出控制线），则可认为该序列是白噪声的样本。

R 的 `Box.test()` 函数提供了 Box-Pierce 检验和 Ljung-Box 检验功能。R 的 portes 包提供了多种一元和多元白噪声检验功能。

如：IBM 月度收益率可以认为是白噪声（见例3.3）；CRSP 最低 10 分位投资组合月度收益率不是白噪声（见例3.4）。

设 $\{\varepsilon_t\}$ 是零均值独立同分布白噪声，$\mathrm{Var}(\varepsilon_t) = \sigma^2$, 数列 $\{\psi_j\}$ 满足 $\sum_j \psi_j^2 < \infty$，$\psi_0 = 1$, 令

$$X_t = \mu + \sum_{j=0}^{\infty} \psi_j \varepsilon_{t-j}, t \in \mathbb{Z} \tag{3.1}$$

则称 $\{X_t\}$ 是（因果）线性时间序列，ε_t 代表了在时刻 t 增加的变动信息，称为**新息** (innovation) 或者扰动 (shock)。因果线性时间序列满足新息 $\varepsilon_{t+j}(j \geq 1)$ 与历史的 X_t, X_{t-1} 独立。而且，$\{X_{t-1}, X_{t-2}, \dots\}$ 与 $\{\varepsilon_{t-1}, \varepsilon_{t-2}, \dots\}$ 可以互相线性地表示。

这个定义可以放宽到 $\{\varepsilon_t\}$ 是宽平稳的不相关列（宽白噪声）的情形。这时，称(3.1)为 $\{X_t\}$ 的 **Wold 分解**，称 $\{X_t\}$ 为**纯非决定性序列**。参见:

- 何书元 (2003), 应用时间序列分析. 北京大学出版社. 第 5 章。
- HERMAN WOLD(1938), A Study in the Analysis of Stationary Time Series, Almquist and Wicksell, Stockholm .(博士论文)

许多弱平稳时间序列是线性时间序列，后面讲到的 AR 模型、MA 模型、ARMA 模型都属于线性时间序列。另外的许多弱平稳时间序列可以被线性时间序列近似。非平稳的时间序列不是线性时间序列。

$\{\psi_j\}$ 称为模型的 ψ 权重。易见

$$EX_t = \mu, \quad \mathrm{Var}(X_t) = \sigma^2 \sum_{j=0}^{\infty} \psi_j^2,$$

自协方差函数为

$$\begin{aligned}\gamma_k =&\mathrm{Cov}(X_t, X_{t-k}) = E\left[\left(\sum_{i=0}^{\infty}\psi_i\varepsilon_{t-i}\right)\left(\sum_{i=0}^{\infty}\psi_i\varepsilon_{t-i-l}\right)\right]\\ =&E\left(\sum_{i,j=0}^{\infty}\psi_i\psi_j\varepsilon_{t-i}\varepsilon_{t-j-k}\right) = \sigma^2\sum_{i,j=0}^{\infty}\psi_i\psi_j\delta_{i-j-k}\\ =&\sigma^2\sum_{j=0}^{\infty}\psi_j\psi_{j+k}\end{aligned}$$

其中 δ_k 当 $k=0$ 时为 1，当 $k\neq 0$ 时为 0。

自相关函数为

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{\sum_{j=0}^{\infty}\psi_j\psi_{j+k}}{1+\sum_{j=1}^{\infty}\psi_j^2},\ k\geq 0$$

线性时间序列模型满足 $\psi_j\to 0,\ j\to\infty$，所以历史上的扰动的影响会逐渐消失。另外 $\rho_k\to 0,\ k\to\infty$，所以相距较远的观测之间的相关性很小。

不是所有的弱平稳时间序列都有这样的性质。非平稳序列更是不需要满足这些性质。

3.5 附录：补充知识

3.5.1 严平稳

对时间序列 $\{X_t, t\in\mathbb{Z}\}$，如果对任意的 $t\in\mathbb{Z}$ 和正整数 n, k，$(X_t,\ldots,X_{t+n-1})$ 总是与 $(X_{t+k},\ldots,X_{t+n-1+k})$ 同分布，则称 $\{X_t\}$ 为**严平稳**时间序列。

如果严平稳时间序列 $\{X_t\}$ 有二阶矩，则它也是宽平稳的。

如果宽平稳时间序列 $\{X_t\}$ 的任意有限维分布都服从广义正态分布，则 $\{X_t\}$ 也是严平稳的。

如果 $\{\varepsilon_t\}$ 为独立同分布零均值白噪声，方差为 σ^2，$\{\psi_j\}$ 绝对可和或者平方可和，则线性时间序列

$$X_t = \mu+\sum_{j=0}^{\infty}\psi_j\varepsilon_{t-j}, t\in\mathbb{Z}$$

同时宽平稳和严平稳的时间序列。

3.5.2 严平稳遍历性

时间序列建模，是从理论上无穷长度的随机变量序列 $\{X_t(\omega), t=\ldots,-1,0,1,\ldots,\omega\in\Omega\}$ 中获得一段观测值，称为一条**轨道**，如 $\{X_1(\omega_0), X_2(\omega_0),\ldots,X_T(\omega_0)\}$，其中 ω_0 是概率空间 Ω 中的一个结果点。虽然 T 可以越来越大，但是我们还是仅仅观测到结果空间中的一个点，希望据此推断所有 $\omega\in\Omega$ 的性质。这需要序列 $\{X_t\}$ 具有**遍历性**。

许多序列没有遍历性。例如，$X_t\equiv\xi$，其中 ξ 为 U(0,1) 随机变量，则每条轨道是一个常数，但不同轨道的常数值不同，从一条轨道无法推断所有结果。有如，设随机变量 U 服从 $\mathrm{U}(0,2\pi)$，$A>0$, $f\in(0,2\pi)$ 为常数，则时间序列

$$X_t = A\cos(ft+U),\ t\in\mathbb{Z}$$

是宽平稳时间序列，但其每条轨道是一条正弦曲线上的离散点，不同轨道的相位不同，从一条轨道无法推断所有轨道的分布情况。

如果从时间序列的一条轨道就可以推断出它的所有有限维分布，就称其为严平稳遍历的。这里不给出遍历性的严格定义，仅给出一些严平稳遍历的充分条件。可以证明，宽平稳的正态时间序列是严平稳遍历的，上述由零均值独立同分布白噪声产生的线性序列是严平稳遍历的。

严平稳遍历的性质：

定理: 如果 $\{X_t\}$ 是严平稳遍历序列，则

(1) 强大数律: 如果 $E|X_1| < \infty$ 则

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} X_t = EX_1, \text{ a.s.}$$

(2) 对任意函数 $\phi(x_1, x_2, \dots, x_m)$，$Y_t = \phi(X_t, X_{t-1}, \dots, X_{t+m-1})$ 也是严平稳遍历序列。

Chapter 4

自回归模型

4.1 自回归模型的概念

如果 $\rho_1 \neq 0$，则 X_t 与 X_{t-1} 相关，可以用 X_{t-1} 预测 X_t。最简单的预测为线性组合，如下模型：

$$X_t = \phi_0 + \phi_1 X_{t-1} + \varepsilon_t \tag{4.1}$$

称为一阶**自回归**模型 (Autoregression model)，记作 AR(1) 模型。其中 $\{\varepsilon_t\}$ 是零均值独立同分布白噪声序列，方差为 σ^2，并设 ε_t 与 $X_{t-1}, X_{t-2}, \ldots$ 独立。系数 $|\phi_1| < 1$。更一般的定义中仅要求 $\{\varepsilon_t\}$ 是零均值白噪声，不要求独立同分布。

这个模型与一元线性回归模型 $Y_i = \phi_0 + \phi_1 x_i + \varepsilon_i$ 在某些方面类似，比如，ε_t 都是起到误差或者扰动的作用。但是，在自回归模型中，自变量 X_{t-1} 在时刻 $t-1$ 时作为因变量，所以自回归模型中因变量和自变量不是两个变量而是同一个变量的不同时刻。回归模型的理论结果不能直接应用到自回归模型当中。

AR(1) 模型也是马尔可夫 (Markov) 过程：X_t 在 $X_{t-1}, X_{t-2}, \ldots$ 条件下的条件分布，只与 X_{t-1} 有关。已知 X_{t-1} 后，用 $X_{t-1}, X_{t-2}, \ldots$ 去预测 X_t，与仅用 X_{t-1} 去预测的效果相同。这种性质称为**马氏性**。

条件期望和条件方差：

$$E(X_t|X_{t-1}) = \phi_0 + \phi_1 X_{t-1}, \quad \mathrm{Var}(X_t|X_{t-1}) = \sigma^2$$

即在 $X_{t-1} = x_{t-1}$ 已知后，X_t 条件的条件分布是期望为 $\phi_0 + \phi_1 x_{t-1}$, 方差为 σ^2 的分布。可以证明

$$\mathrm{Var}(X_t) = \frac{\sigma^2}{1-\phi_1^2}$$

因为 $|\phi_1| < 1$，所以 X_t 在 $X_{t-1} = x_{t-1}$ 已知条件下的条件方差小于其无条件方差，也就是说用 X_{t-1} 的信息去预测 X_t，可以使得 X_t 的波动减小，能够达到预测的效果。

AR(1) 模型的推广是 AR(p) 模型：

$$X_t = \phi_0 + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t \tag{4.2}$$

其中 $\{\varepsilon_t\}$ 是零均值独立同分布白噪声序列，方差为 σ^2，且 ε_t 与 $X_{t-1}, X_{t-2}, \ldots$ 独立。系数 $\phi_1, \ldots, \phi_p$ 需要满足如下条件：方程

$$1 - \phi_1 z - \cdots - \phi_p z^p = 0$$

的所有复根 z_* 都满足 $|z_*|>1$，上述方程的左边的多项式称为 AR(p) 模型的**特征多项式**，特征多项式的所有复根称为**特征根**，对系数的条件称为 “特征根都在单位圆外”。在有的教材中将特征多项式的根的倒数定义为特征根，在 (Tsay, 2013) 中就是以倒数为特征根。对 AR(1) 就是要求 $|\phi_1|<1$。

更一般的定义中仅要求 $\{\varepsilon_t\}$ 是零均值白噪声，不要求独立同分布。

4.2 滞后算子

设 $\{\xi_t, t\in\mathbb{Z}\}$ 为弱平稳时间序列或者常数列，定义如下的滞后算子 B:

$$B\xi_t=\xi_{t-1},\quad B^j\xi_t=\xi_{t-j},\ j\in\mathbb{Z}$$

考虑复变函数 $P(z)=\sum_{j=0}^{\infty}a_jz^j$，其中 $\{a_j\}$ 为实数列，设 $\sum_{j=0}^{\infty}|a_j|<\infty$（称为系数**绝对可和**），有限阶多项式为 $P(z)$ 的特例。定义

$$P(B)\xi_t=\sum_{j=0}^{\infty}a_j\xi_{j-t}.$$

$P(B)$ 也称为一个（常系数时齐）线性滤波器，$P(B)\xi_t$ 是 $\{\xi_t\}$ 序列的一个滑动平均变换。

设 $Q(z)=\sum_{j=0}^{\infty}b_jz^j$，$\{b_j\}$ 为实数列，绝对可和，则 $C(z)=P(z)Q(z)=\sum_{j=0}^{\infty}c_jz^j$，其中

$$c_j=\sum_{i=0}^{j}a_ib_{j-i},\ j=0,1,...$$

且 $\{c_j\}$ 绝对可和，称数列 $\{c_j\}$ 是数列 $\{a_j\}$ 和数列 $\{b_j\}$ 的**离散卷积**。对任意弱平稳列或者常数列 $\{\xi_t\}$ 均有

$$P(B)Q(B)\xi_t=Q(B)P(B)\xi_t=C(B)\xi_t,\ t\in\mathbb{Z}$$

若 $\xi_t\equiv\xi$ 与 t 无关，则 $B^j\xi=\xi$。比如，$B^j1=1$，$P(B)1=P(1)$。

令 $D(z)=\frac{P(z)}{Q(z)}$，若 $Q(z)\neq0$ 对任意满足 $|z|\le1$ 的复数 z 均成立，则 $D(z)=\sum_{j=0}^{\infty}d_jz^j$, $\{d_j\}$ 绝对可和，且

$$P(B)\xi_t=Q(B)D(B)\xi_t=D(B)Q(B)\xi_t,\ t\in\mathbb{Z}$$

4.3 AR(1) 模型的性质

研究 AR 模型的性质，可以在对实际数据选择适当模型时，知道那些情况下使用 AR 模型，使用何种 AR 模型是合适的。

对理论证明感兴趣的同学请参见 (何书元, 2003) 的论述。

AR(1) 模型(4.1)要求 $|\phi_1|<1$，这是(4.1)中的 $\{X_t\}$ 有弱平稳解的充分必要条件。

充分性：当 $|\phi_1|<1$ 且 $\phi_0=0$ 时，因为

$$\frac{1}{1-\phi_1z}=\sum_{j=0}^{\infty}\phi_1^jz^j,\quad \frac{1}{1-\phi_1B}=\sum_{j=0}^{\infty}\phi_1^jB^j$$

取

$$X_t^*=\sum_{j=0}^{\infty}\phi_1^j\varepsilon_{t-j}$$

右侧的随机变量级数均方收敛且 a.s. 收敛。将上式代入 $X_t = \phi_1 X_{t-1} + \varepsilon_t$，直接验证或者利用滞后算子性质可以证明 X_t^* 满足方程。所以 $\phi_0 = 0, |\phi_1| < 1$ 时 AR(1) 模型有 X_t^* 这样的因果线性时间序列形式的弱平稳解。对于一般的 ϕ_0，因为平稳要求 $EX_t = EX_{t-1} = \mu$ 所以要求 $\mu = \phi_0 + \phi_1\mu$ 即 $\mu = \frac{\phi_0}{1-\phi_1}$，令

$$X_t^* = \mu + \sum_{j=0}^{\infty} \phi_1^j \varepsilon_{t-j}$$

可以验证其满足 AR(1) 模型，是因果线性时间序列形式的弱平稳解。

必要性证明：如果模型有弱平稳解 X_t，因为要求 ε_t 与 $X_{t-1}, X_{t-2}, \ldots$ 独立，在模型两边求方差得

$$\begin{aligned}
&\mathrm{Var}(X_t) = \mathrm{Var}(\phi_0 + \phi_1 X_{t-1} + \varepsilon_t) = \phi_1^2 \mathrm{Var}(X_{t-1}) + \sigma^2, \\
&\sigma_X^2 = \phi_1^2 \sigma_X^2 + \sigma^2, \\
&\sigma_X^2 = \frac{\sigma^2}{1-\phi_1^2}
\end{aligned}$$

因为 $\sigma_X^2 > 0$ 所以 $1 - \phi_1^2 > 0, |\phi_1| < 1$。

上述证明过程也证明了

$$\mu = EX_t = \frac{\phi_0}{1-\phi_1}, \quad \sigma_X = \mathrm{Var}(X_t) = \frac{\sigma^2}{1-\phi_1^2}.$$

注：可以证明，$\phi_1 = 1$ 时，一定没有弱平稳列满足方程 $X_t = \phi_0 + \phi_1 X_{t-1} + \varepsilon_t$。当 $|\phi_1| > 1$ 时，取

$$X_t = \mu - \sum_{j=1}^{\infty} \phi_1^{-j} \varepsilon_{t+j},\ t \in \mathbb{Z}$$

则 $\{X_t\}$ 是满足 $X_t = \phi_0 + \phi_1 X_{t-1} + \varepsilon_t$ 的弱平稳时间序列，但 ε_t 不再与 $X_{t-1}, X_{t-2}, \ldots$ 相互独立。

由 $\mu = \phi_0/(1-\phi_1)$ 得 $\phi_0 = (1-\phi_1)\mu$，AR(1) 模型可以写成

$$X_t = (1-\phi_1)\mu + \phi_1 X_{t-1} + \varepsilon_t$$

即 t 时刻的值，是历史均值与 $t-1$ 时刻值的加权平均，加上一个与历史值相互独立的扰动。

当 $\phi_1 = 0$ 时，AR(1) 模型就退化成白噪声，所以 AR(1) 模型通常都认为 $\phi_1 \neq 0$。$|\phi_1|$ 越接近于 1，说明前后的相关性越强。

4.4 AR(1) 模型的自相关函数

AR(1) 模型的平稳解满足 ϵ_t 与 $X_{t-1}, X_{t-2}, \ldots$ 独立。将模型写成

$$X_t - \mu = \phi_1(X_{t-1} - \mu) + \varepsilon_t \tag{4.3}$$

在(4.3)两边乘以 ε_t 后取期望，有

$$E[\varepsilon_t(X_t - \mu)] = \phi_1 E[\varepsilon_t(X_{t-1} - \mu)] + \sigma^2 = \sigma^2$$

在(4.3)两边乘以 $X_t - \mu$ 后取期望，有

$$\gamma_0 = \phi_1 \gamma_1 + \sigma^2$$

在(4.3)两边乘以 $X_{t-j}-\mu$ 后取期望，有

$$\gamma_j = \phi_1\gamma_{j-1},\ j=1,2,...$$

即有

$$\gamma_0 = \sigma_X^2 = \frac{\sigma^2}{1-\phi_1^2} = \phi_1\gamma_1 + \sigma^2 \tag{4.4}$$

$$\gamma_j = \phi_1\gamma_{j-1},\ j=1,2,... \tag{4.5}$$

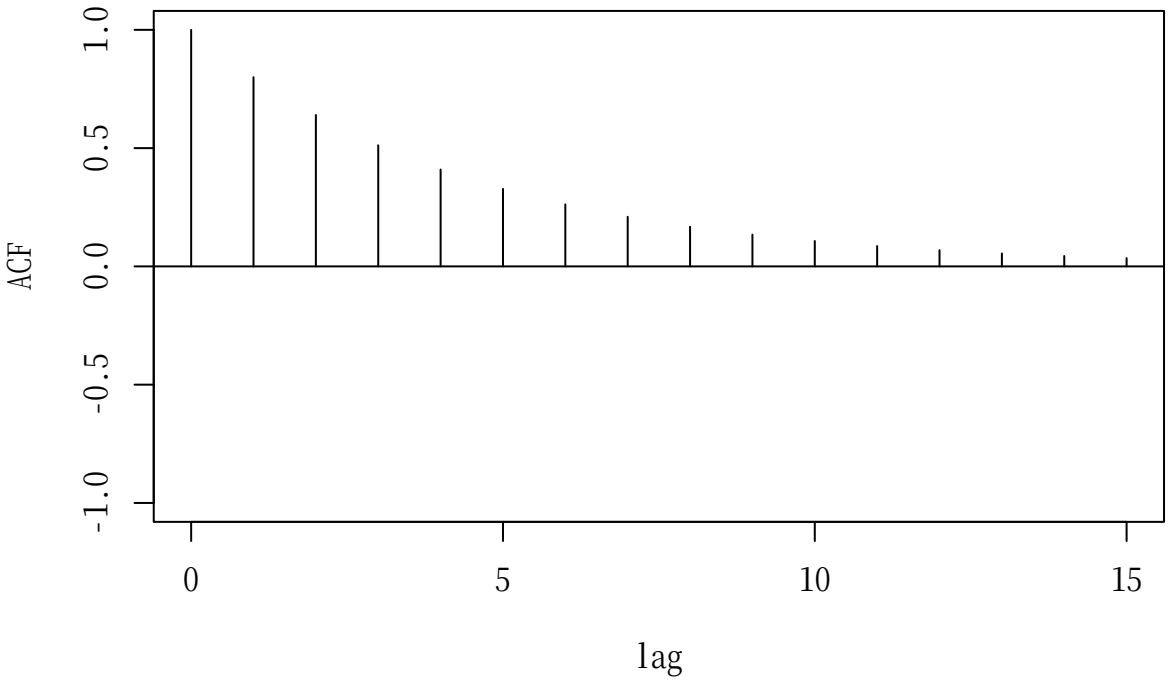
于是 ACF 为

$$\rho_j = \frac{\gamma_j}{\gamma_0} = \phi_1^j,\ j=1,2,...$$

当 $0<\phi_1<1$ 时，ACF 为正的单调下降序列，以负指数速度（几何级数）下降。当 $-1<\phi_1<0$ 时，ACF 为负正交替的序列，绝对值以负指数速度下降。

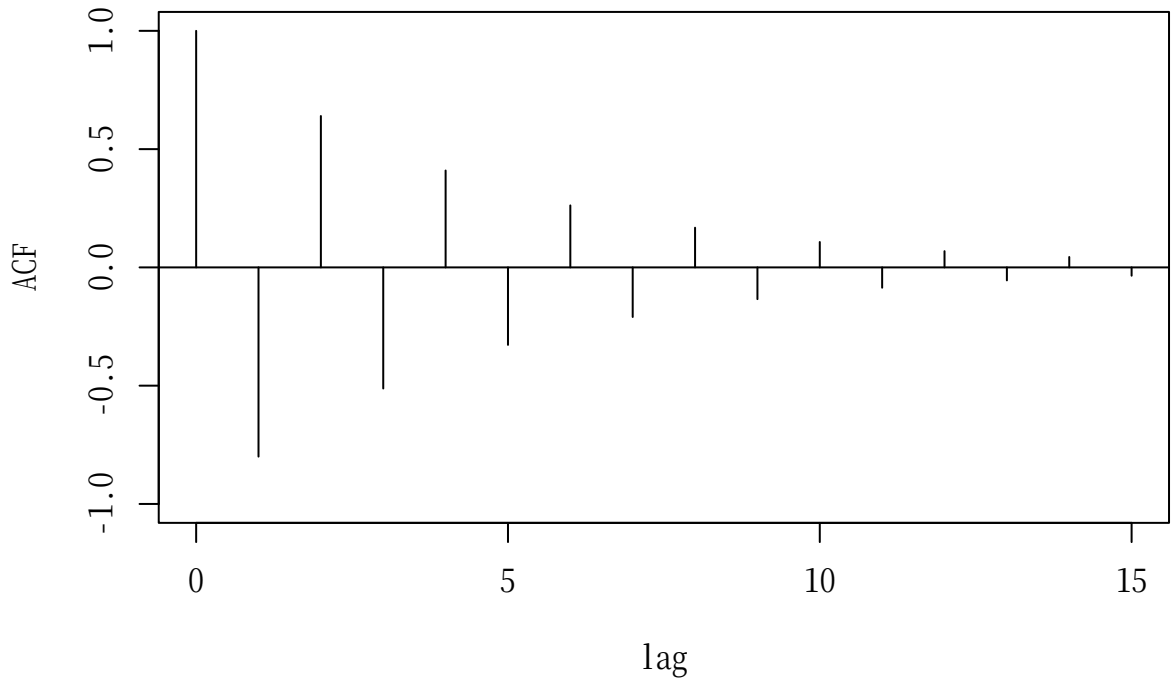
例：$\phi_1=0.8$ 的 ACF 图形：

```
tmp.x <- 0:15
tmp.y <- 0.8^tmp.x
plot(tmp.x, tmp.y, type="h", xlab="lag", ylab="ACF",
     ylim=c(-1, 1))
abline(h=0)
```



例：$\phi_1=-0.8$ 的 ACF 图形：

```
tmp.x <- 0:15
tmp.y <- (-0.8)^tmp.x
plot(tmp.x, tmp.y, type="h", xlab="lag", ylab="ACF",
     ylim=c(-1, 1))
abline(h=0)
```



4.5 AR(2) 模型的性质

AR(2) 模型为

$$X_t = \phi_0 + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \varepsilon_t \tag{4.6}$$

其中 $\{\varepsilon_t\}$ 为独立同分布的零均值白噪声, ε_t 与 $X_{t-1}, X_{t-2}, \ldots$ 独立。系数满足特征方程的根都在单位圆外要求。特征方程为

$$1 - \phi_1 z - \phi_2 z^2 = 0 \tag{4.7}$$

这个模型有因果线性时间序列形式的弱平稳解。对(4.6)两边取期望得 $\mu = EX_t$ 的方程

$$\mu = \phi_0 + \phi_1 \mu + \phi_2 \mu$$

所以

$$\mu = \frac{\phi_0}{1 - \phi_1 - \phi_2}$$

将(4.6)中的 X_t 减去 μ，方程变成

$$X_t - \mu = \phi_1(X_{t-1} - \mu) + \phi_2(X_{t-2} - \mu) + \varepsilon_t \tag{4.8}$$

在(4.8)两边乘以 $X_{t-j} - \mu$，并利用 ε_t 与 $X_{t-1}, X_{t-2}, ...$ 独立的性质，可得

$$\gamma_j = \phi_1\gamma_{j-1} + \phi_2\gamma_{j-2},\ j = 1, 2, ...$$

两边除以 γ_0 得

$$\rho_j = \phi_1\rho_{j-1} + \phi_2\rho_{j-2},\ j = 1, 2, ... \tag{4.9}$$

这种形式的递推称为齐次线性差分方程。(4.9)可以用滞后算子写成

$$(1 - \phi_1 B - \phi_2 B^2)\rho_j = 0,\ j = 1, 2, ... \tag{4.10}$$

可以看出，递推中用到的 $1 - \phi_1 B - \phi_2 B^2$ 就是将特征多项式中的 z 替换成了滞后算子 B，这样递推得到的 ρ_j 的性质自然就是由特征多项式决定的。

(4.9)的递推需要初值 ρ_0 和 ρ_1。$\rho_0 = 1$，而

$$\rho_1 = \phi_1\rho_0 + \phi_2\rho_1, \quad \rho_1 = \frac{\phi_1}{1 - \phi_2}$$

这样就可以给定 ϕ_1, ϕ_2 后递推计算出 $\rho_j, j = 1, 2, 3, ...$。特征根都在单位圆外的条件可以保证 $\lim_{j\to\infty}\rho_j = 0$ （参见(何书元, 2003)§2.1），且衰减速度为负指数速度。$\lim_{j\to\infty}\rho_j = 0$ 也是弱平稳列的必要条件。

特征多项式 $1 - \phi_1 z - \phi_2 z^2 = 0$ 的根的判别式为 $\Delta = \phi_1^2 + 4\phi_2$，当 $\phi_2 \geq -4\phi_1^2$ 时为实根，当 $\phi_2 < -4\phi_1^2$ 时为复根。两个特征根都在单位圆外的充分必要条件是

$$|\phi_1| < 2,\ |\phi_2| < 1,\ \phi_2 < 1 \pm \phi_1$$

图4.1的三角形阴影部分代表了 (ϕ_1, ϕ_2) 使得模型有平稳解的区域，不包含三角形边界；蓝色区域以及蓝色和红色边界线为两个实特征根的区域，红色区域为两个共轭复根的区域。

有两个实根时，两个根为

$$z_1, z_2 = \frac{\phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2}}{-2\phi_2}$$

这两个根的绝对值都大于 1。付过两个根都是正的，则 ρ_j 为正，呈现出负指数速度单调衰减。如果两个根正负不同，则绝对值仍以负指数速度衰减但是不单调，当负根绝对值更小时会正负交替衰减。

有共轭复根时，两个根为

$$z_1, z_2 = \frac{\phi_1}{-2\phi_2} \pm i\frac{\sqrt{|\phi_1^2 + 4\phi_2|}}{-2\phi_2}$$

复根的模为

$$|z_i| = \frac{1}{\sqrt{-\phi_2}}$$

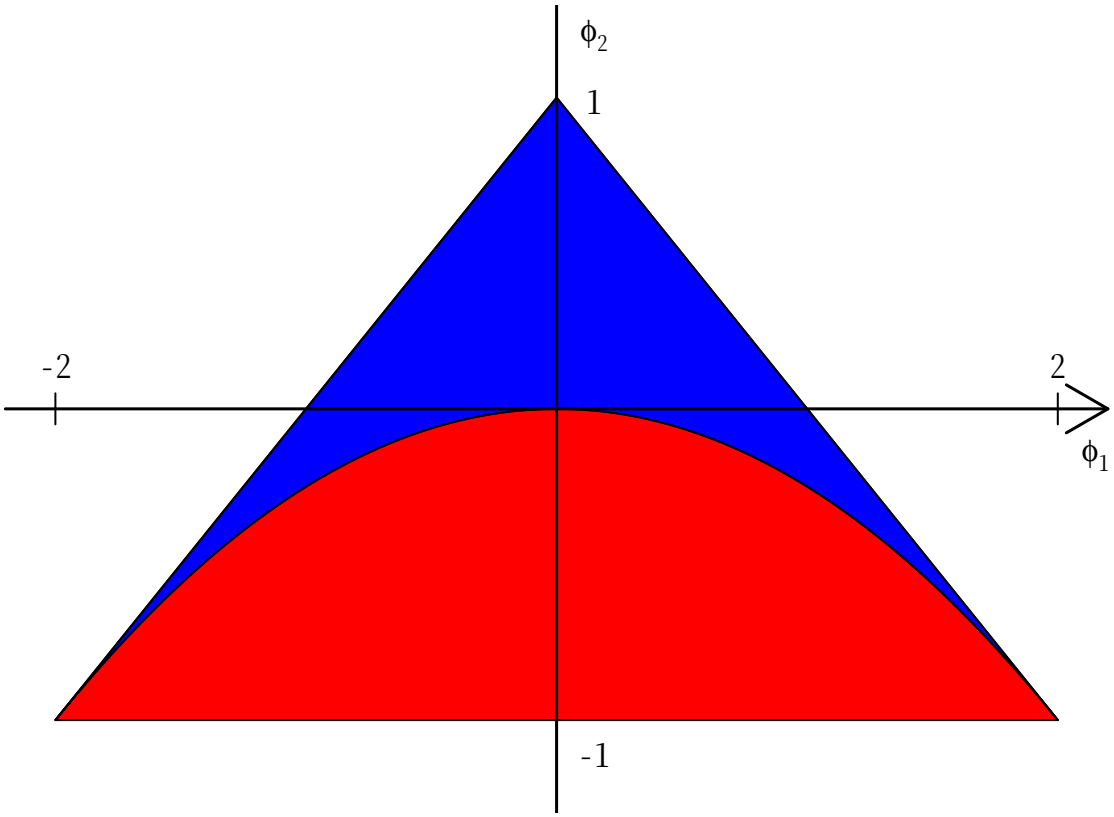


图 4.1: AR(2) 的平稳区域

辐角为

$$\omega = \cos^{-1} \frac{\phi_1}{2\sqrt{-\phi_2}}$$

反过来有

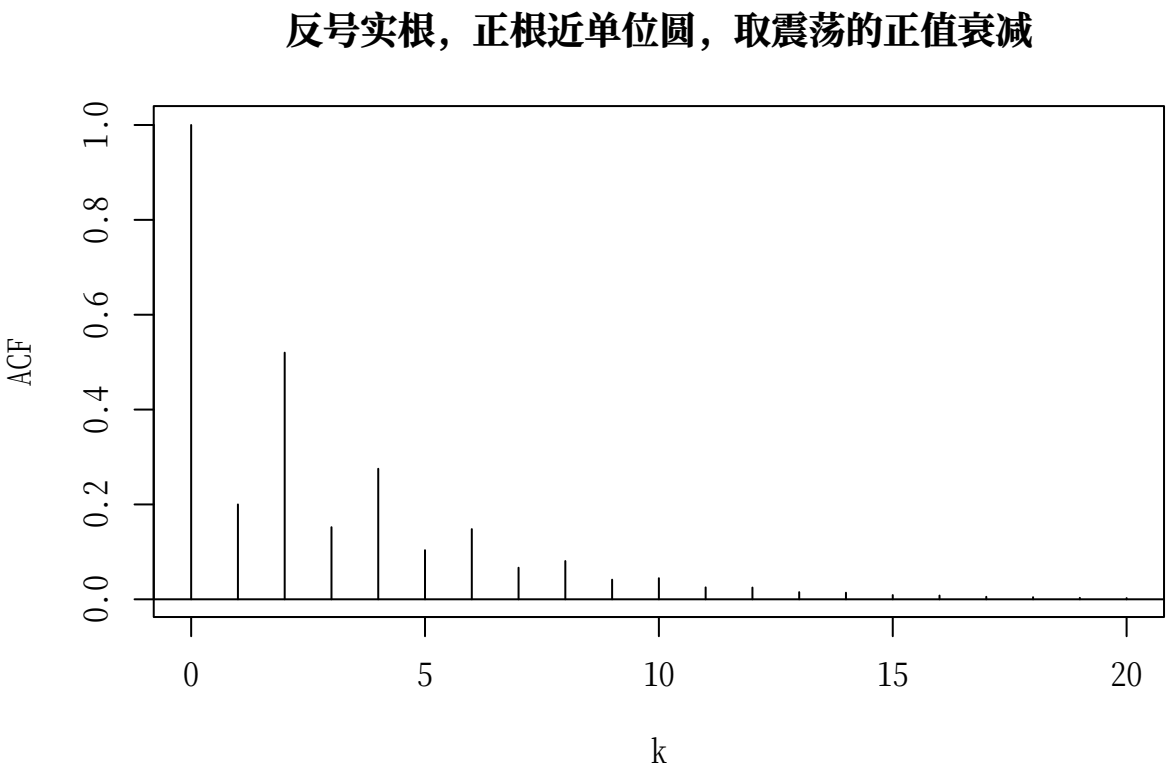
$$\phi_2 = -\frac{1}{|z_1|^2},\ \phi_2 = \frac{2}{|z_1|}\cos\omega$$

ω 这个辐角对应的周期为

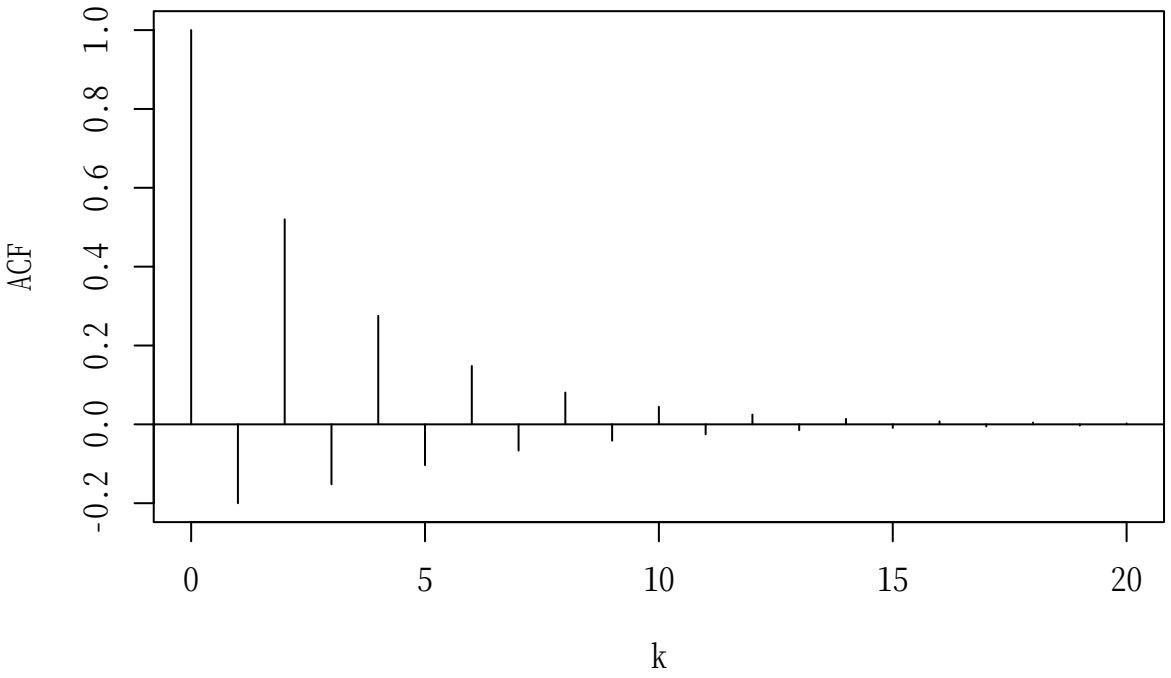
$$P = \frac{2\pi}{\omega} = \frac{2\pi}{\cos^{-1}\frac{\phi_1}{2\sqrt{-\phi_2}}}$$

这样的 AR(2) 模型会体现出随机地平均以周期 P 的波动；ρ_j 序列呈现出以周期 P 震荡的幅度负指数衰减的变化。

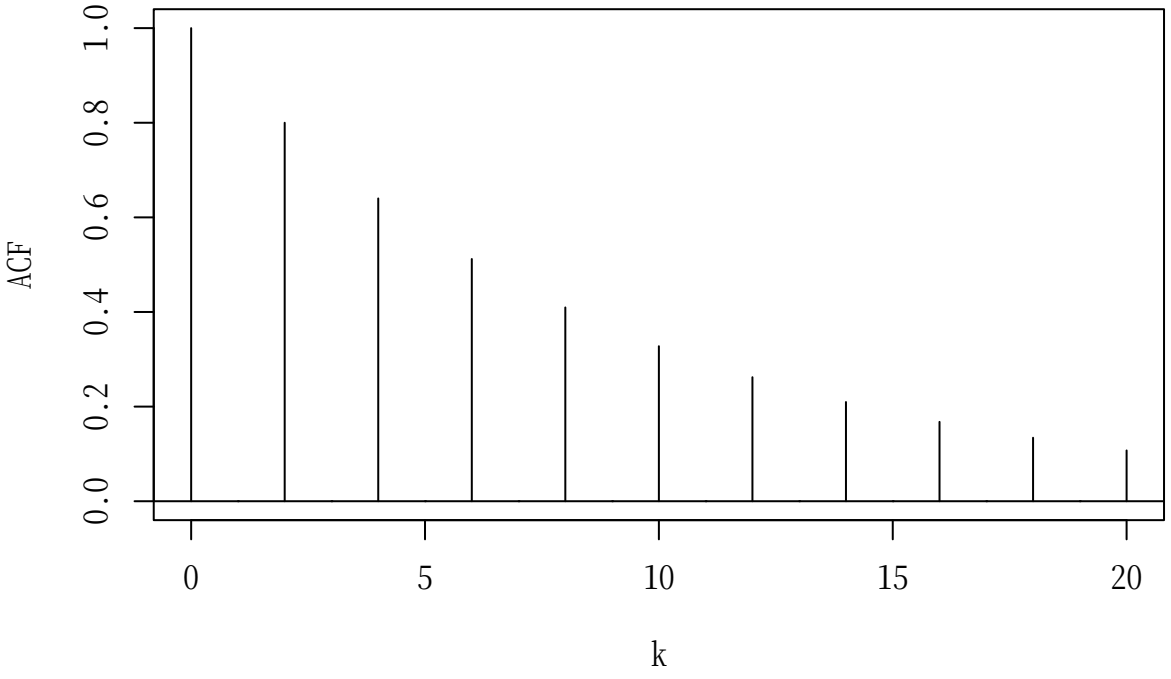
下面对不同的 ϕ_1, ϕ_2 取值画出 ACF 的典型图形。



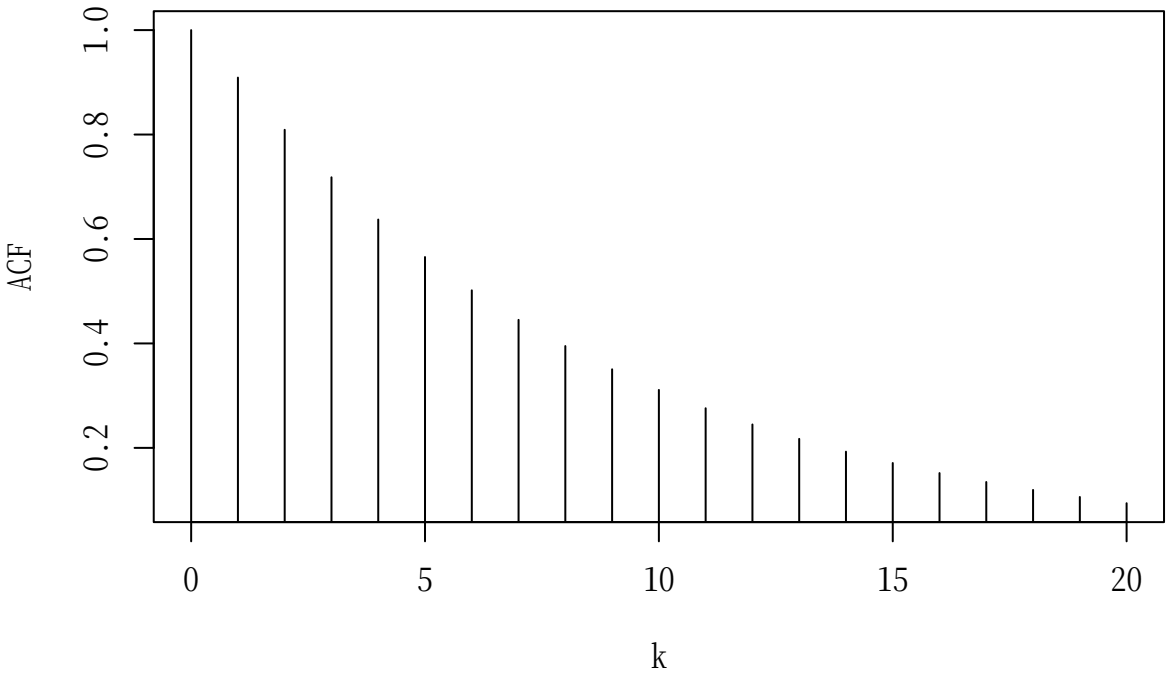
反号实根，负根近单位圆，正负交替衰减



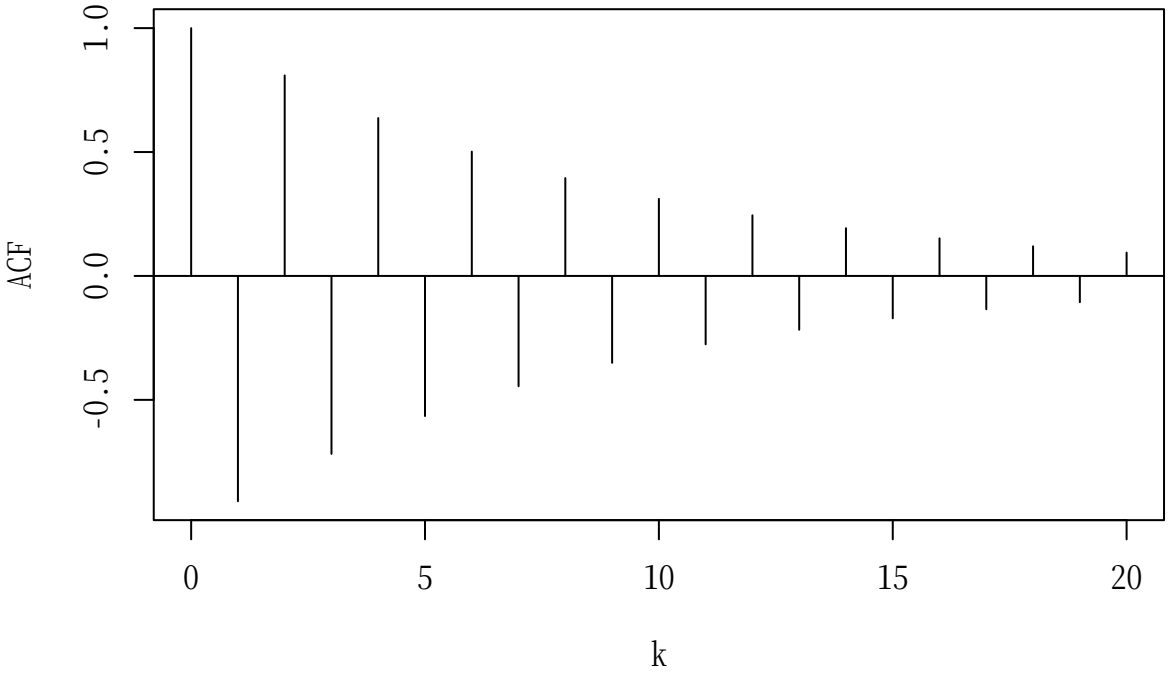
根为相反数，非负震荡衰减



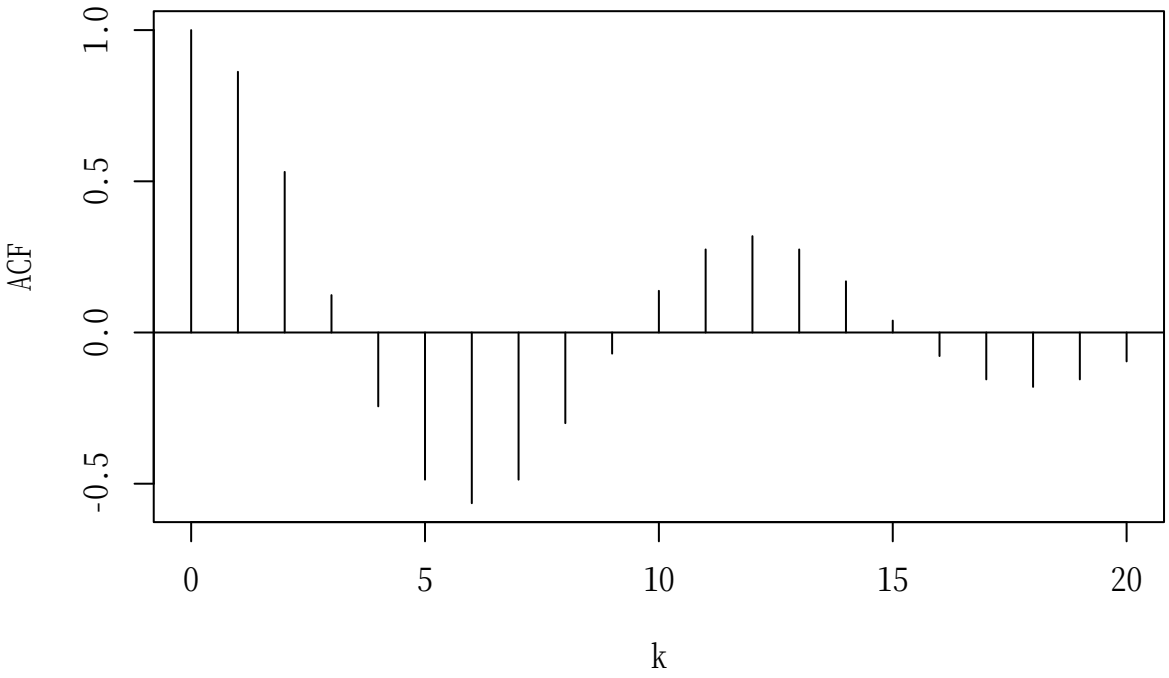
两正根，单调正值衰减



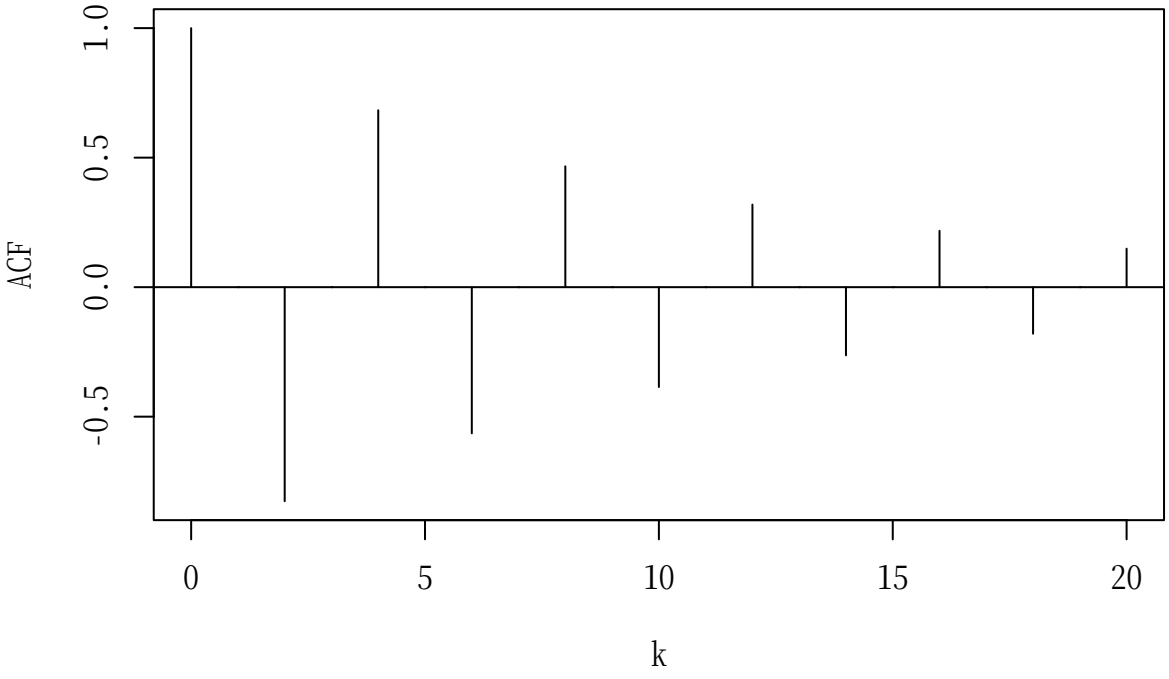
两负根，正负交替衰减

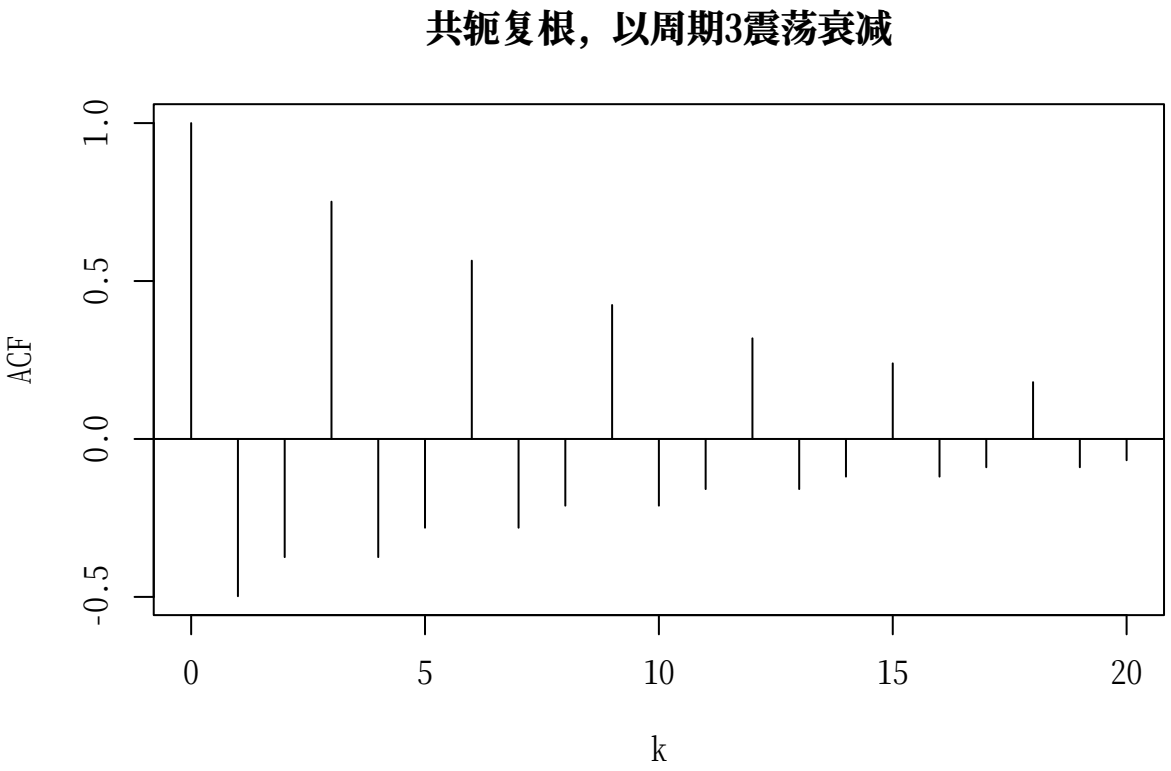


共轭复根，以周期12震荡衰减



共轭复根，以周期4震荡衰减





上图中的模型 AR(2) 分别为：

$$
\begin{aligned}
X_t =& 0.1X_{t-1} + 0.5X_{t-2} + \varepsilon_t \\
X_t =& -0.1X_{t-1} + 0.5X_{t-2} + \varepsilon_t \\
X_t =& 0.8X_{t-1} + \varepsilon_t \\
X_t =& X_{t-1} - 0.1X_{t-2} + \varepsilon_t \\
X_t =& -X_{t-1} - 0.1X_{t-2} + \varepsilon_t \\
X_t =& \left(\frac{2}{1.1}\cos\frac{\pi}{6}\right)X_{t-1} - \frac{1}{1.1^2}X_{t-2} + \varepsilon_t \\
X_t =& \left(\frac{2}{1.1}\cos\frac{\pi}{2}\right)X_{t-1} - \frac{1}{1.1^2}X_{t-2} + \varepsilon_t \\
X_t =& \left(\frac{2}{1.1}\cos\frac{2\pi}{3}\right)X_{t-1} - \frac{1}{1.1^2}X_{t-2} + \varepsilon_t
\end{aligned}
$$

例 4.1. 考虑美国的国民生产总值 (GNP) 经过季节调整后的季度增长率。时间是 1947 年第二季度到 2010 年第一季度，总计 252 个观测值。

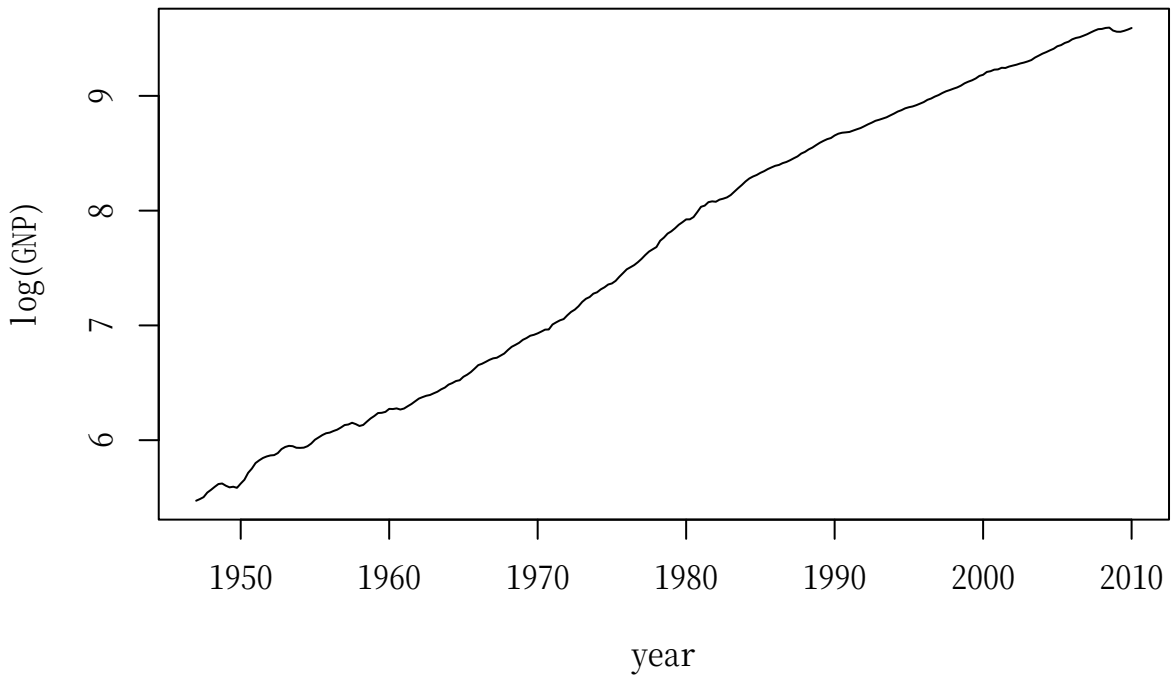
读入数据并计算对数增长率：

```
da <- read_table("q-gnp4710.txt", col_types=cols(.default = col_double()))
gnp <- ts(da[["VALUE"]], start=c(1947, 1), frequency=4)
```

这个序列从 1947 年第一季度到 2010 年第一季度共 253 个观测。

GNP 的对数值的图形：

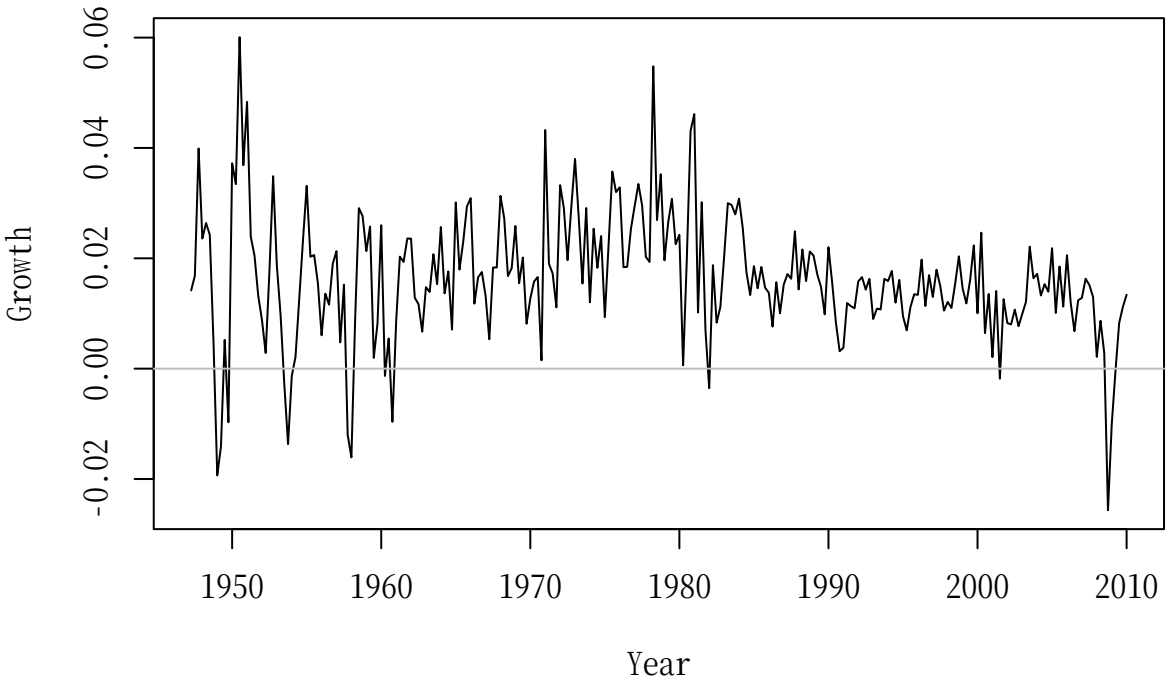
```
plot(log(gnp), xlab="year", ylab="log(GNP)")
```



以上的 GNP 是经过季节调整的，所以看不出季节性。

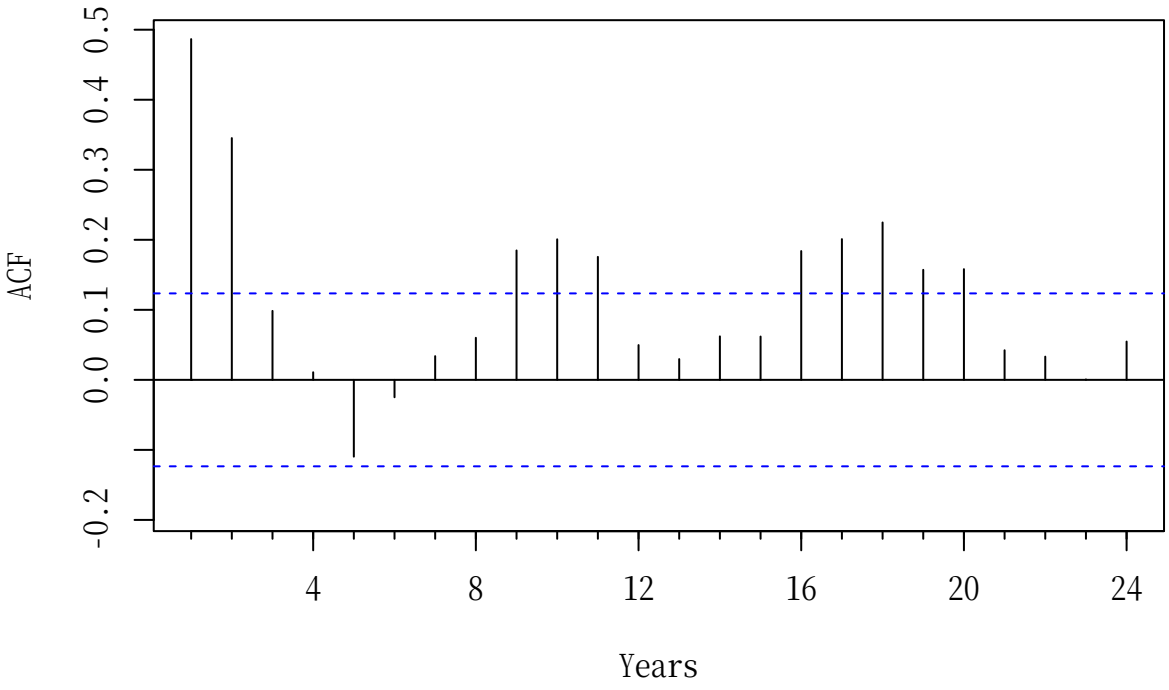
计算 GNP 的对数值的差分作为对数增长率，并作图：

```
rate <- diff(log(gnp))
plot(rate, xlab="Year", ylab="Growth")
abline(h=0, col="gray")
```



增长率的 ACF 估计：

```
forecast::Acf(rate, xlab="Years", main="")
```



这个图形呈现出一定周期的震荡衰减，可尝试 AR 建模。

拟合的模型为

$$(1-0.438B-0.206B^2+0.156B^3)(X_t-0.016)=\varepsilon_t,\ \sigma^2=\mathrm{Var}(\epsilon_t)=9.55\times10^{-5}$$

R 函数 `polyroot(x)` 可以求出多项式的所有复根，`x` 是多项式按升幂排列的系数。

```
roots <- polyroot(c(1, -0.438, -0.206, 0.156)); roots
```

```
## [1]  1.614790+0.866168i -1.909068+0.000000i  1.614790-0.866168i
```

```
abs(roots)
```

```
## [1] 1.832429 1.909068 1.832429
```

有一个实根 -1.9091 和一对共轭复根 $1.6148\pm0.8662i$, 复根的模为 1.8324，复根对应的周期为

```
2*pi/Arg(roots[1])
```

```
## [1] 12.7619
```

12.8 个季度，约为 3 年。这是美国 GNP 的随机周期，也是 ACF 震荡的周期。

时间序列的周期研究是很常用的工具。

4.6 AR(p) 模型的性质

对于一般的 AR(p) 模型，其 ACF 的性质以及序列的随机周期，也由其特征根决定。ACF 可以是单调衰减、震荡衰减、正负交替衰减、呈周期震荡衰减。在有复特征根根或者有接近 -1 的特征根时时间序列呈现出一定的随机周期变化。

由平稳性得

$$\mu = \frac{\phi_0}{1-\phi_1-\cdots-\phi_p}$$

自相关函数 (ACF) 满足如下的递推（差分方程）

$$(1-\phi_1 B-\cdots-\phi_p B^p)\rho_j = 0,\ j=1,2,...$$

AR(p) 模型的平稳解是线性时间序列。

4.7 偏自相关函数

实际数据用 AR 模型建模时，阶数 p 是未知的，确定 p 的问题称为**定阶**。一般常用偏自相关函数和 AIC 准则。

设 $X_1,...,X_n,Y$ 为随机变量，

$$L(Y|X_1,...,X_n) = \underset{\hat{Y}=b_0+b_1X_1+\cdots+b_nX_n}{\operatorname{argmin}} E(Y-\hat{Y})^2$$

称为用 $X_1,...,X_n$ 对 Y 的最优线性预测。$Y-L(Y|X_1,...,X_n)$ 与 $Z-L(Z|X_1,...,X_n)$ 的相关系数称为 Y 和 Z 在扣除 $X_1,...,X_n$ 影响后的**偏相关系数**。

对平稳线性时间序列，对 $n=1,2,...$，有

$$L(X_t|X_{t-1},...,X_{t-n}) = \phi_{n0}+\phi_{n1}X_{t-1}+\cdots+\phi_{nn}X_{t-n}$$

其中 $\phi_{nj}, j=0,1,...,n$ 与 t 无关。称 ϕ_{nn} 为时间序列 $\{X_t\}$ 的**偏自相关系数**，$\{\phi_{nn}\}$ 序列称为时间序列 $\{X_t\}$ 的**偏自相关函数** (PACF)。

ϕ_{nn} 实际是 X_t 与 X_{t-n} 在扣除 $X_{t-2},...,X_{t-n+1}$ 的影响后的偏相关系数。ϕ_{11} 就是 ρ_1。

ϕ_{nn} 用样本进行估计，得到的估计值 $\hat{\phi}_{nn}, n=1,2,...$ 称为**样本偏自相关函数**。在 R 软件中用 `pacf(x)` 估计并作图。

如果 $\{X_t\}$ 服从如下 AR(p) 模型：

$$X_t = \phi_0+\phi_1X_{t-1}+\cdots+\phi_pX_{t-p}+\varepsilon_t,\ \phi_p\neq 0$$

这意味着用 $X_{t-1},X_{t-2},...$ 的线性组合预测 X_t 时，只需要用到 $X_{t-1},...,X_{t-p}$，增加 $X_{t-p-1},X_{t-p-2},...$ 不能改进预测。这意味着 $\phi_{kk}=0,\ k>p$。这种性质叫做 AR 模型的偏自相关函数截尾性。

AR(p) 序列的样本偏自相关函数 $\hat{\phi}_{kk}$ 满足如下性质：

- $T\to\infty$ 时 $\hat{\phi}_{pp}\to\phi_p\neq 0$。
- 对 $k>p$, $\hat{\phi}_{kk}\to 0 (T\to\infty)$。
- 对 $k>p$，$\hat{\phi}_{kk}$ 渐近方差为 $\frac{1}{T}$。

这样，可以用类似对 ACF 的白噪声检验那样给 PACF 图画出 $\pm\frac{2}{\sqrt{T}}$ 的上下界限，以此判断 PACF 在哪里截尾。

例 4.2. 考虑 CRSP 价值加权指数月度收益率，时间从 1926-1 到 2008-12。

```
d <- read_table(
  "m-ibm3dx2608.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
ibmind <- xts(as.matrix(d[,-1]), d$date)
tclass(ibmind) <- "yearmon"
vw <- ts(coredata(ibmind)[,"vwrtn"], start=c(1926,1), frequency=12)
head(ibmind)
```

```
##             ibmrtn     vwrtn     ewrtn     sprtn
## 1月 1926 -0.010381  0.000724  0.023174  0.022472
## 2月 1926 -0.024476 -0.033374 -0.053510 -0.043956
## 3月 1926 -0.115591 -0.064341 -0.096824 -0.059113
## 4月 1926  0.089783  0.038358  0.032946  0.022688
## 5月 1926  0.036932  0.012172  0.001035  0.007679
## 6月 1926  0.068493  0.056888  0.050487  0.043184
```

```
plot(vw, main="CRSP Value Weighted Index Monthly Return")
abline(h=0, col="gray")
```

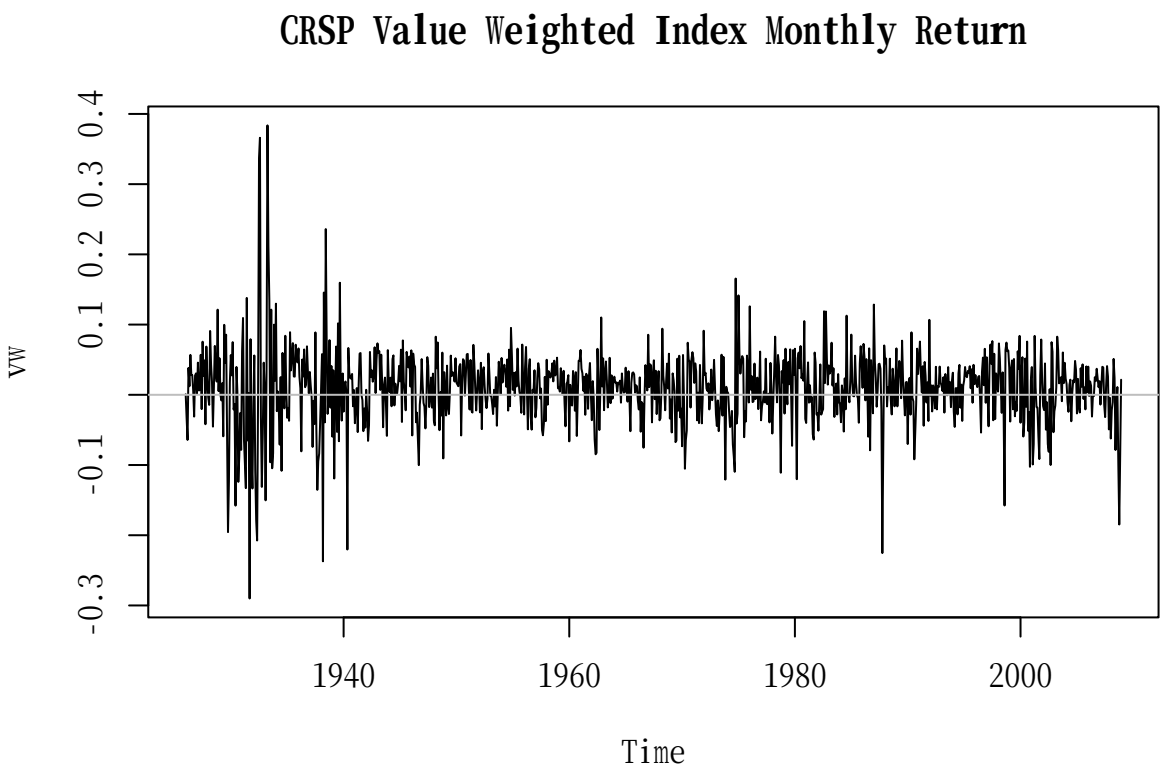


图 4.2: CRSP 价值加权指数月收益率

```
forecast::Acf(vw, main="")
```

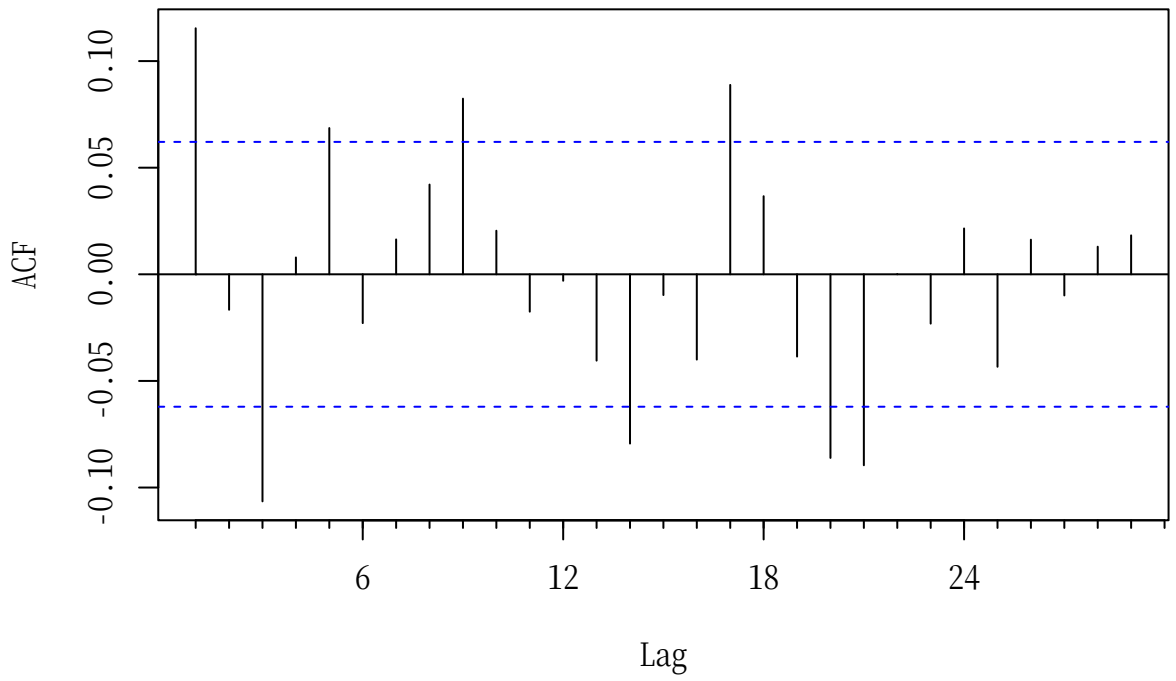


图 4.3: CRSP 价值加权指数月收益率的 ACF

ACF 到 $k = 21$ 还没有截尾。PACF 图:

```
pacf(vw, main="")
```

两倍的渐近标准差为:

```
2/sqrt(length(vw))
```

```
## [1] 0.06337243
```

PACF 值的列表:

```
tmp <- pacf(vw, main="", plot=FALSE, lag.max=12)
cbind(k=round(tmp$lag*12), pacf=round(tmp$acf, 4))
```

```
##       k    pacf
## [1,]  1  0.1154
## [2,]  2 -0.0304
## [3,]  3 -0.1025
## [4,]  4  0.0326
## [5,]  5  0.0618
```

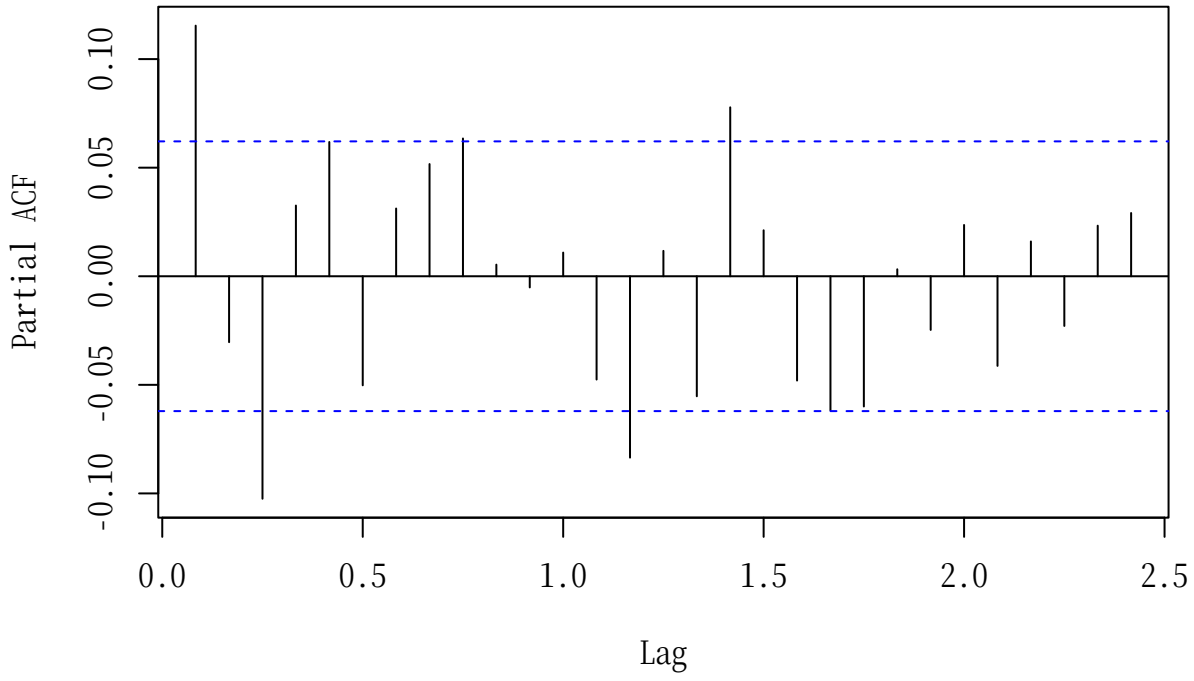


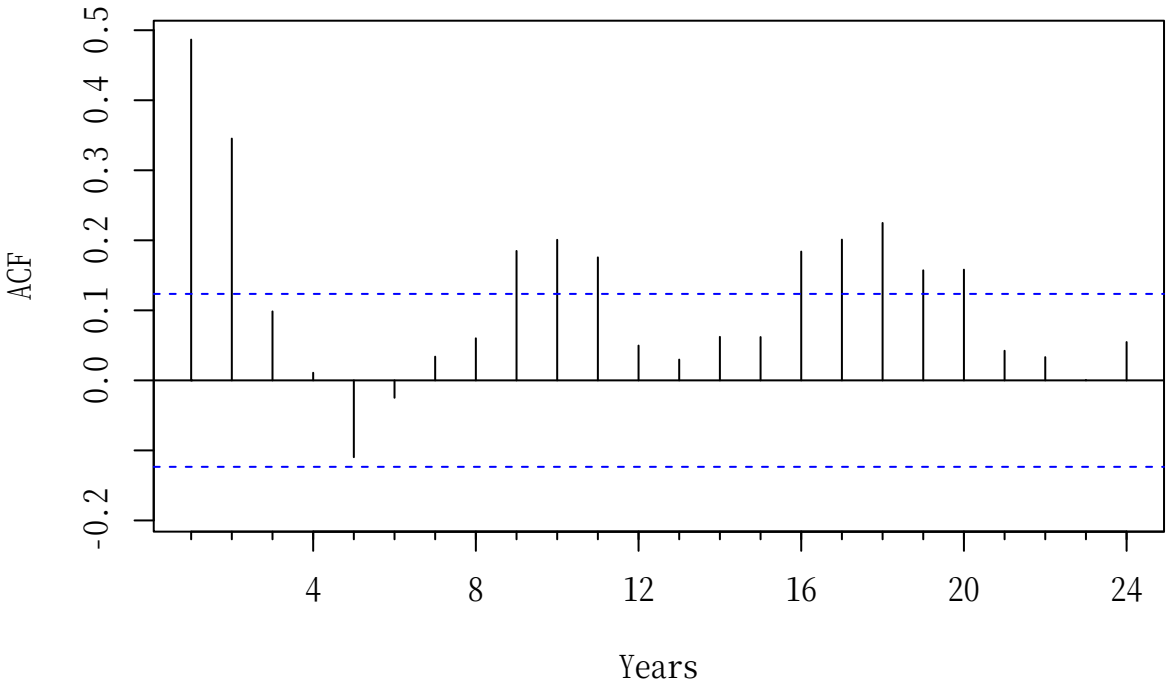
图 4.4: CRSP 价值加权指数月度收益率的 PACF

```
##  [6,]  6 -0.0502
##  [7,]  7  0.0312
##  [8,]  8  0.0517
##  [9,]  9  0.0635
## [10,] 10  0.0053
## [11,] 11 -0.0052
## [12,] 12  0.0109
```

如果仅看前 12 个 PACF，取 $p = 3$ 应该可以，但是实际上 PACF 在 $k = 17$ 仍未截尾。

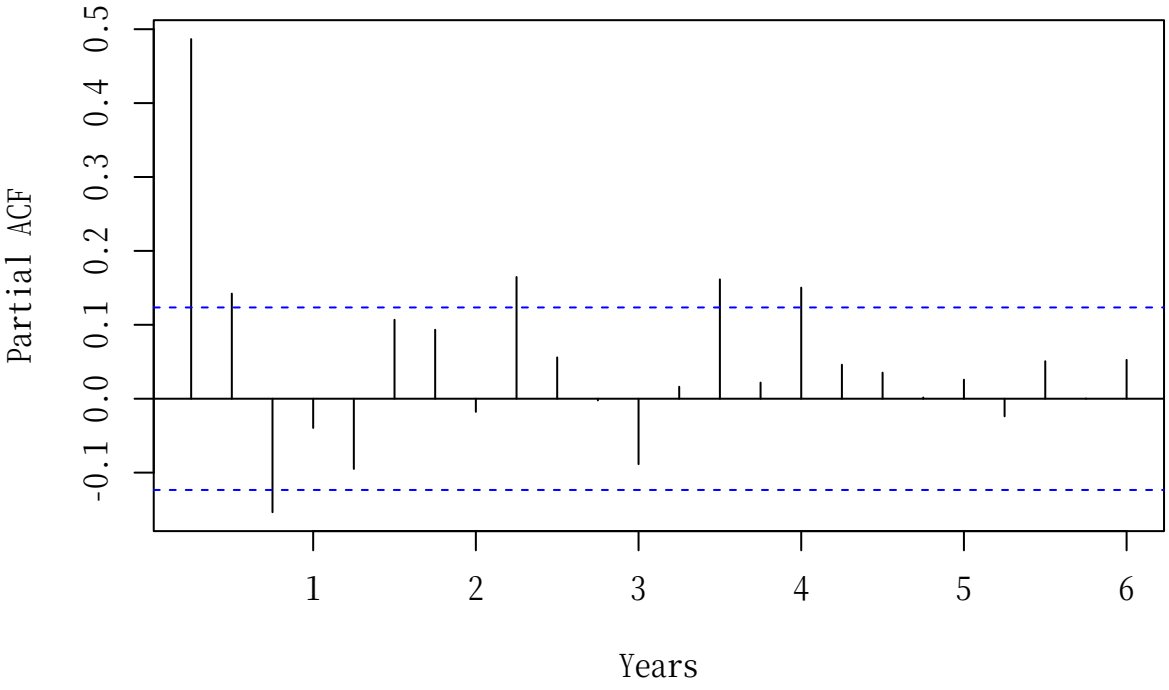
例 4.3. 考虑例4.1中的美国经过季节调整后的 GNP 季度对数增长率数据。

```
forecast::Acf(rate, xlab="Years", main="")
```



ACF 明显不截尾。

```
pacf(rate, xlab="Years", main="")
```



PACF 虽然在 $k = 3, 9, 14, 16$ 等位置超出界限，但是超出不多，可考虑用 AR(3) 建模。

4.8 信息准则

信息准则是统计建模中常用的模型比较工具，其基本思想是模型拟合数据的拟合优度与模型简单化的折衷。

AIC 准则（Akaike's Information Criterion）：

$$\text{AIC} = -\frac{2}{T}\ln(\text{似然函数值}) + \frac{2}{T}(\text{参数个数})$$

其中似然函数值是在参数最大似然估计处的似然函数值。当模型为高斯 AR(p)，即 $\{\epsilon_t\}$ 是独立同 $\text{N}(0, \sigma^2)$ 序列时的 AR(p) 模型时，AIC 公式为

$$\text{AIC}(k) = \ln \tilde{\sigma}_k^2 + \frac{2k}{T}$$

其中 k 是模型的阶，$\tilde{\sigma}_k^2$ 是阶为 k 的条件下 ε_t 的方差的最大似然估计。$\ln \tilde{\sigma}_k^2$ 代表了模型对数据的拟合优劣，此值越大拟合越差；$\frac{2k}{T}$ 是对模型复杂程度的惩罚，此值越大，模型越复杂，稳定性越差，对未来的情况的适应性也越差。在某个范围内取 k 使得 AIC(k) 最小，就达成了拟合优度与模型简单程度的折衷。

另一个常用的信息准则是 BIC 准则 (Bayesian Information Criterion)，高斯 AR 模型为：

$$\text{BIC}(k) = \ln \tilde{\sigma}_k^2 + \frac{k \ln T}{T}$$

BIC 倾向于取比 AIC 更低阶的模型。

可以取 $k = 0, 1, \ldots, P_0$ 计算 AIC 或 BIC，取最小值点的 k。P_0 可取为 $10 \log_{10} T$。

参考：

- HIROTUGU AKAIKE, Fitting Autoregressive Models for Prediction, Annals of the Institute of Statistical Mathematics AC-19 (1974), pp. 364 – 385
- GIDEON SCHWARZ, Estimating the Dimensions of a Model, Annals of Statistics 6(1978), pp. 461 – 464

例 4.4. 考虑例4.1中的美国经过季节调整后的 GNP 季度对数增长率数据的 AR 建模定阶。

基本 R 软件 stats 包中的 `ar()` 函数可以对时间序列样本进行 AR 建模，默认采用 AIC 准则定阶。用选项 `aic=FALSE, order.max=p` 可以指定 p 阶模型。

```
gnprate <- diff(log(gnp))
resm <- ar(gnprate, method="mle"); resm
```

```
##
## Call:
## ar(x = gnprate, method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.4318   0.1985  -0.1180   0.0189  -0.1607   0.0900   0.0615  -0.0814
##       9
##  0.1940
##
## Order selected 9  sigma^2 estimated as  8.918e-05
```

产生了一个 9 阶的模型。作其 AIC 的图形：

```
plot(as.numeric(names(resm$aic)), resm$aic, type="h",
     xlab="k", ylab="AIC")
```

从 AIC 图形可以看出，如果取较低的阶，3 阶也是可以的。

虽然 `ar()` 函数没有提供 BIC 的值，但是因为 $\mathrm{BIC}(k) - \mathrm{AIC}(k) = \frac{k(\ln T - 2)}{T}$, 可以自己计算：

```
tmp.T <- length(gnprate)
tmp.bic <- resm$aic + as.numeric(names(resm$aic)) * (log(tmp.T) - 2)/tmp.T
plot(as.numeric(names(resm$aic)), tmp.bic, type="h",
     xlab="k", ylab="BIC")
```

BIC 也建议 $p = 9$。

4.9 AR 模型参数估计方法

AR 模型有多种估计方法，比如，用普通线性回归的最小二乘法估计，假设正态分布用最大似然估计，Yule-Walker 递推计算，Burg 递推计算，等等。

在 `stats::ar()` 函数中，`method="ols"` 为最小二乘估计，设 ϕ_i 的估计为 $\hat{\phi}_i$，则拟合值为

$$\hat{x}_t = \hat{\phi}_0 + \hat{\phi}_1 x_{t-1} + \cdots + \hat{\phi}_p x_{t-p},\ t = p+1, \ldots, T$$

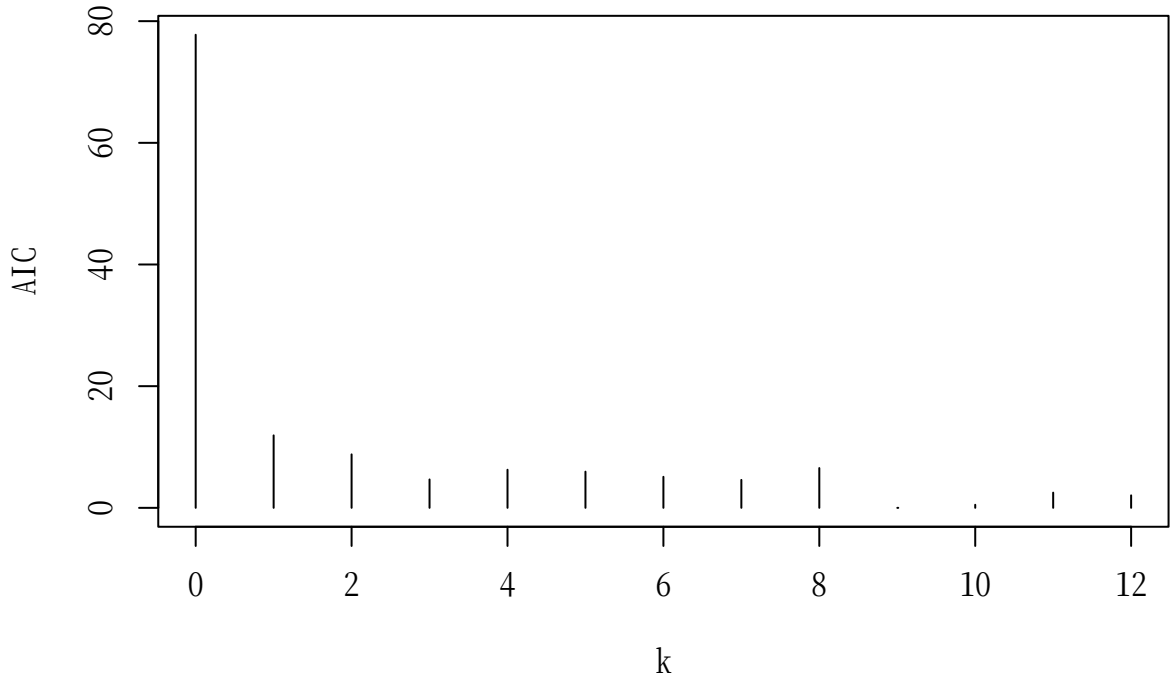


图 4.5: GNP 对数增长率用 AR 建模的 AIC

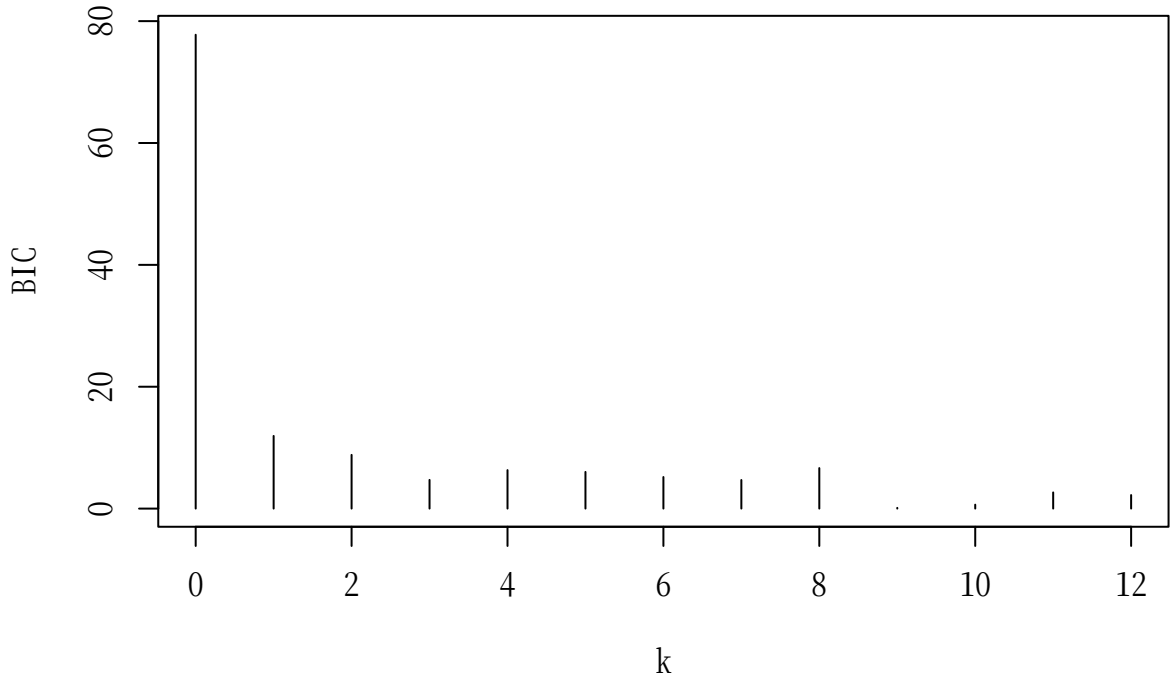


图 4.6: GNP 对数增长率用 AR 建模的 BIC

残差为

$$e_t = x_t - \hat{x}_t,\ t = p+1, \ldots, T$$

相应的新息方差 $\sigma^2 = \mathrm{Var}(\varepsilon_t)$ 的估计为

$$\hat{\sigma}^2 = \frac{1}{T-2p-1}\sum_{t=p+1}^{T} e_t^2$$

如果使用高斯条件最大似然估计（认为 $x_1, \ldots, x_p$ 固定），则 $\hat{\phi}_i$ 估计不变，但是新息方差得估计变成了

$$\tilde{\sigma}^2 = \frac{1}{T-p}\sum_{t=p+1}^{T} e_t^2 = \frac{T-2p-1}{T-p}\sum_{t=p+1}^{T} e_t^2$$

例 4.5. 对例4.5的 CRSP 价值加权指数月度收益率用 AR(3) 建模。

先用 AIC 定阶：

```
resm <- ar(vw, method="mle"); resm
```

```
##
## Call:
## ar(x = vw, method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.1167  -0.0112  -0.1126   0.0217   0.0735  -0.0452   0.0254   0.0462
##       9
##  0.0660
##
## Order selected 9  sigma^2 estimated as  0.002831
```

用 AIC 定阶为 9 阶。AIC 数值：

```
round(resm$aic, 2)
```

```
##     0     1     2     3     4     5     6     7     8     9    10    11    12
## 22.33 10.99 12.07  3.35  4.37  2.46  1.96  3.04  2.24  0.00  1.97  3.94  5.81
```

从 AIC 数值看出，选较低阶时 $p = 3$ 是一个可以接受的选择。使用 `arima()` 函数估计，这个函数的输出比较容易解释。

```
mean.vw <- mean(vw); mean.vw
```

```
## [1] 0.00890439
```

```
resm2 <- arima(vw, order=c(3,0,0))
resm2
```

```
##
## Call:
## arima(x = vw, order = c(3, 0, 0))
##
```

```
## Coefficients:
##          ar1      ar2      ar3  intercept
##       0.1158  -0.0187  -0.1042     0.0089
## s.e.  0.0315   0.0317   0.0317     0.0017
##
## sigma^2 estimated as 0.002875:  log likelihood = 1500.86,  aic = -2991.73
```

结果中的 intercept 实际是均值 μ 的估计，$\hat{\mu} = 0.0089$。$\hat{\sigma} = 0.054$。模型为:

$$(X_t - 0.0089) = 0.1158(X_{t-1} - 0.0089) - 0.0187(X_{t-2} - 0.0089) - 0.1042(X_{t-3} - 0.0089)$$

结果还给出了系数的标准误差。以 $\hat{\theta} \pm 2SE(\hat{\theta})$ 作为简单的近似 95% 置信区间，结果标准误差说明 μ, ρ_1, ρ_3 是显著不等于 0 的，而 ρ_2 不显著。

$\hat{\mu} = 0.008904$，这是 CRSP 价值加权指数的平均月度简单收益率。折合成年收益率，再换算到 1926-2008 年共 83 年的总简单收益率：

```
(1 + mean.vw)^12 - 1
```

```
## [1] 0.1122442
```

```
((1 + mean.vw)^12)^83 - 1
```

```
## [1] 6832.002
```

按平均月度简单增长率 0.8904% 计算，得到年度简单增长率为 11.22%；83 年可增长 6832 倍。

按实际数据 83 年的实际增长计算 83 年的总简单增长率，然后折合到一年的复利：

```
prod(1 + vw) - 1
```

```
## [1] 1591.953
```

```
prod(1 + vw)^(1/83) - 1
```

```
## [1] 0.09290084
```

实际的复利年均增长率为 9.29%，而不是 11.22%; 实际的 83 年的简单增长率是 1592 倍，而不是 6832 倍。

用 $\hat{\mu}$ 计算的结果与用真实数据计算的结果的 83 年增长率有这么大差距，是因为 $\hat{\mu} > 0$，按 $\hat{\mu}$ 计算总是在按几何级数增长；但是按 $\prod(1 + x_t)$ 计算则有时增长有时下降，所以实际的总增长率要低得多。

R 的 forecast 包提供了一个 auto.arima() 函数，可以自动进行模型选择。

tseries 包的 arma() 函数可以用条件极小二乘方法估计 ARMA 模型参数，使用了 optim() 函数来求极值。结果可以用 summary() 显示。tseries::arma() 的结果还支持 print(), plot(), coef(), vcov(), residuals(), fitted() 等信息提取函数。

4.10 AR 模型检验

拟合 AR 模型后，如果模型是能够准确表示数据的规律的，则拟合的残差应该近似为白噪声，应该能通过白噪声检验。因为残差是从模型估计计算得到的，自由度有损失，在 Box.test() 中用 fitdf=p 指定自由度减少个数。

例 4.6. 例如，对例4.5 的 CRSP 价值加权指数月简单收益率的 AR(3) 模型的残差进行白噪声检验：

```
resm2 <- arima(vw, order=c(3,0,0))
Box.test(resm2$residuals, lag=12, type="Ljung", fitdf=3)
```

```
##
##  Box-Ljung test
##
## data:  resm2$residuals
## X-squared = 16.352, df = 9, p-value = 0.05988
```

结果在 0.05 水平下不显著，说明模型拟合是充分的。

从例4.5的估计结果看出 ϕ_2 不限制。在 `arima()` 函数中可以用 `fixed=` 指定某些系数固定为给定值，`NA` 表示不规定。这样做的一个潜在问题是得到的模型可能不符合平稳性条件。

```
resm3 <- arima(vw, order=c(3,0,0), fixed=c(NA, 0, NA, NA)); resm3
```

```
## Warning in arima(vw, order = c(3, 0, 0), fixed = c(NA, 0, NA, NA)): some AR
## parameters were fixed: setting transform.pars = FALSE
```

```
##
## Call:
## arima(x = vw, order = c(3, 0, 0), fixed = c(NA, 0, NA, NA))
##
## Coefficients:
##          ar1  ar2      ar3  intercept
##       0.1136    0  -0.1063     0.0089
## s.e.  0.0313    0   0.0315     0.0017
##
## sigma^2 estimated as 0.002876:  log likelihood = 1500.69,  aic = -2993.38
```

检验稳定性：

```
abs(polyroot(c(1, -coef(resm3)[1:3])))
```

```
## [1] 2.031585 2.279433 2.031585
```

三个根都在单位圆外，符合平稳性条件。

对固定 $\phi_2 = 0$ 后的模型进行残差的白噪声检验：

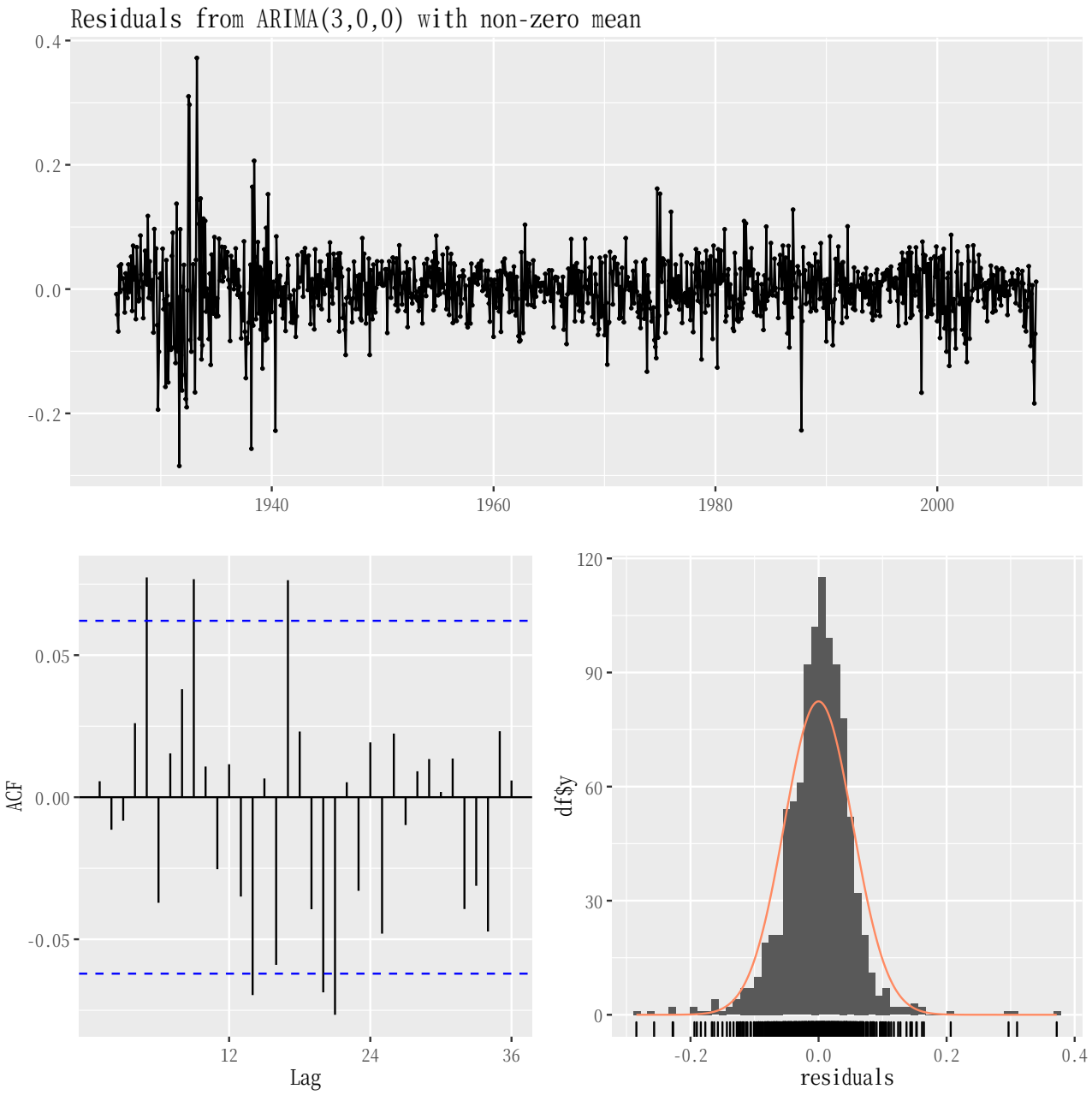
```
Box.test(resm3$residuals, lag=12, type="Ljung", fitdf=2)
```

```
##
##  Box-Ljung test
##
## data:  resm3$residuals
## X-squared = 16.828, df = 10, p-value = 0.07827
```

不否认残差为白噪声。提取残差也可以用 `residuals(resm3)`。

也可以用 forecast::checkresiduals() 函数进行模型诊断：

```
forecast::checkresiduals(resm3)
```



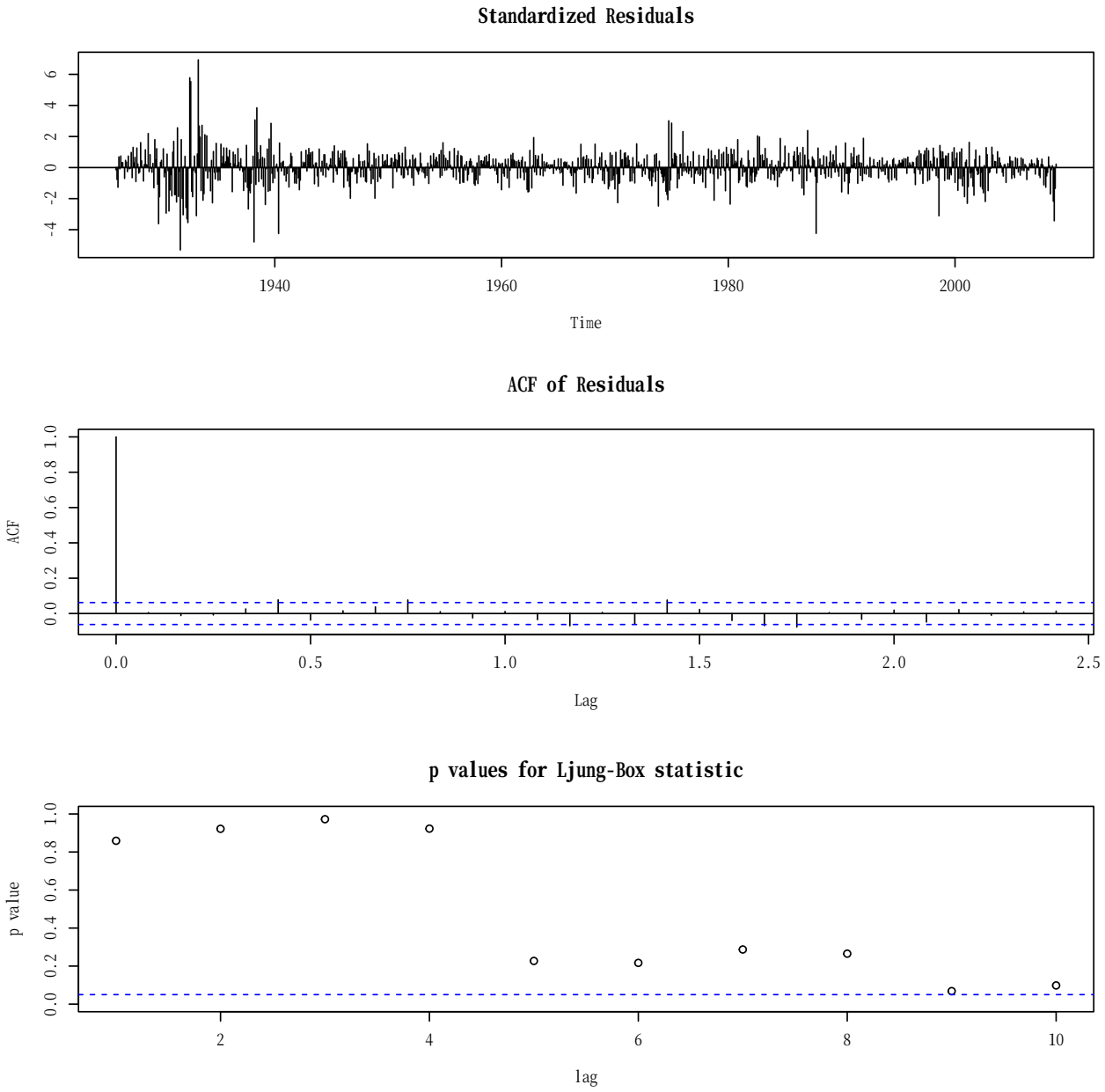

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,0) with non-zero mean
## Q* = 46.896, df = 21, p-value = 0.0009699
##
## Model df: 3.   Total lags used: 24
```

结果图形中包括残差，残差的 ACF 图，残差的直方图和正态密度拟合。此函数自动进行了残差的 Ljung-Box 白噪声

检验，进行了自由度调整，但没有考虑限定 $\phi_2 = 0$ 问题。可以用 `lag` 选项指定 Ljung-Box 检验使用的平方和项数。

stats 包的 `tsdiag()` 函数也可以对模型残差进行诊断，包括标准化残差图、残差的 ACF 图、残差的 Ljung-Box 白噪声检验 p 值，包括使用不同平方和项数的检验 p 值，但是可能没有进行自由度调整。程序如：

```
tsdiag(resm3)
```



4.11 AR 模型拟合优度指标

类似于线性回归模型的拟合优度判断，在线性时间序列建模中，也可以定义如下的拟合优度 R^2 统计量

$$R^2 = 1 - \text{——}$$

比如，拟合了 AR(p) 模型后，可以计算 R^2 为

$$R^2 = 1 - \frac{\sum_{t=p+1}^{T} e_t^2}{\sum_{t=p+1}^{T}(x_t - \bar{x})^2}$$

其中 $\bar{x} = \frac{1}{T-p}\sum_{t=p+1}^{T} x_t$。$0 \le R^2 \le 1$。$R^2$ 越大，说明模型对数据拟合越好，残差越小。这样的指标仅对线性时间序列有意义。如果是非平稳的单位根过程，拟合 AR 模型的 R^2 会趋于 1。

只要数据不变，R^2 随阶数 p 的增加而增大。根据统计学中的精简性原则（称为 Ocam's razor 原则），应尽可能选择简单模型。所以提出调整的 R^2 指标 (adjusted R^2)，定义为

$$R^2_{\text{Adj}} = 1 - \frac{\hat{\sigma}^2}{\hat{\sigma}_x^2}$$

其中 $\hat{\sigma}^2$ 是新息方差 σ^2 的估计，$\hat{\sigma}_x^2$ 是 X_t 的样本方差。仍有 $0 \le R^2_{\text{Adj}} \le 1$，但 R^2_{Adj} 对阶数 p 的增加有惩罚。

4.12 用估计的 AR 模型进行预测

前面定义的 $L(Y|X_1, \dots, X_n)$ 是最佳线性预测。还可以定义 $\hat{Y}$ 为 $X_1, \dots, X_n$ 的任意函数，使得 $E(Y - \hat{Y})^2$ 最小，称为最佳预测（最优预测）。最佳预测等于条件期望 $E(Y|X_1, \dots, X_n)$。

对 AR(p) 模型（新息为独立同分布白噪声），最佳预测与最佳线性预测相同。

在时刻 h，用截止到 h 位置的信息预测 $x_{h+\ell}$，得到的最佳预测记为 $\hat{x}_h(\ell)$。

4.12.1 超前一步预测

对 $\ell = 1$，因为

$$X_{h+1} = \phi_0 + \phi_1 X_h + \cdots + \phi_p X_{h+1-p} + \varepsilon_{h+1}$$

以及 $E(\varepsilon_{h+1}|X_1, \dots, X_h) = 0$，有

$$\hat{x}_h(1) = E(X_{h+1}|X_1, \dots, X_h) = \phi_0 + \phi_1 X_h + \cdots + \phi_p X_{h+1-p}$$

一步预测误差为

$$e_h(1) = x_{h+1} - \hat{x}_h(1) = \varepsilon_{h+1},\ \text{Var}(e_h(1)) = \sigma^2$$

可见模型中的新息方差 σ^2 也是超前一步预测的均方误差。

如果 ε_t 服从正态分布，则 X_{h+1} 的超前一步预测的 95% 预测区间为 $\hat{x}_h(1) \pm 1.96\sigma$。

对于时间序列数据，真实的系数 ϕ_i 是未知的，只能得到估计量 $\hat{\phi}_i$，当 T 充分大时可以在预测公式中用 $\hat{\phi}_i$ 代替 ϕ_i 进行预测。这样得到的预测区间是不够准确的，更好做法是用 MCMC，参见 (Tsay, 2010) 第 12 章。

4.12.2 超前二步预测

要计算 $\hat{x}_h(2)$，注意到

$$X_{h+2} = \phi_0 + \phi_1 X_{h+1} + \cdots + \phi_p X_{h+2-p} + \varepsilon_{h+2}$$

所以

$$\begin{aligned}\hat{x}_h(2) =& E(x_{h+2}|x_1,\dots,x_h) = \phi_0 + \phi_1 E(x_{h+1}|x_1,\dots,x_h) + \phi_2 x_h + \cdots + \phi_p x_{h+2-p}\\ =& \phi_0 + \phi_1 \hat{x}_h(1) + \phi_2 x_h + \cdots + \phi_p x_{h+2-p}\end{aligned}$$

预测误差为

$$e_h(2) = x_{h+2} - \hat{x}_h(2) = \phi_1 e_h(1) + \varepsilon_{h+2}$$

预测均方误差为

$$E[e_h(2)]^2 = \mathrm{Var}(e_h(2)) = \sigma^2(1+\phi_1^2)$$

这里用到了 ε_{h+2} 与 $e_h(1) = x_{h+1} - \hat{x}_h(1)$ 独立。显然超前两步预测均方误差大于等于超前一步均方误差，这对一般情况都是合理的，预测得越远，我们现有知识的作用就越小。

4.12.3 超前多步预测

一般地有

$$\hat{x}_h(\ell) = \phi_0 + \sum_{j=1}^{p} \phi_j \hat{x}_h(\ell - j)$$

其中

$$\hat{x}_h(i) = \begin{cases} x_{h+i}, & \text{当} i \le 0 \\ \hat{x}_h(i), & \text{当} i > 0 \end{cases}$$

为了计算 $\hat{x}_h(\ell)$，只要递推计算 $\hat{x}_h(1),\dots,\hat{x}_h(\ell-1)$，就可以按上面的公式得到 $\hat{x}_h(\ell)$。超前 ℓ 步的预测误差为

$$e_h(\ell) = x_{h+\ell} - \hat{x}_h(\ell)$$

对平稳 AR(p) 模型，当超前步数 $\ell \to +\infty$ 时，$\hat{x}_h(\ell) \to \mu$。这种性质称为均值回转 (mean reversion)。

对 AR(1) 模型的零均值平稳解 $\{x_t\}$，可以看出

$$\hat{x}_h(\ell) = \phi_1^{\ell} x_h$$

这也是与极限 0 之间的差距，而 $|\phi_1|^{\ell}$ 则代表了趋向于极限的速度，当 $|\phi_1|^{\ell} = \frac{1}{2}$ 时趋向于极限 0 就可以认为极限过程已经到了一半，这个 $\ell = \ln(0.5)/\ln|\phi_1|$ 称为均值回转的**半衰期**。半衰期越短，多步预报的作用越差。

例 4.7. 例如，对例4.5 的 CRSP 价值加权指数月简单收益率的 AR(3) 模型的残差进行多步预测。用前 82 年的 984 个观测预测最后一年的 12 个观测。

```
resm4 <- arima(vw[1:984], order=c(3,0,0))
pred4 <- predict(resm4, n.ahead=12, se.fit=TRUE)
cbind(Observed=round(c(vw[985:996]), 4),
      Predict=round(c(pred4$pred), 4),
      SE=round(c(pred4$se), 3))
```

```
##       Observed Predict    SE
##  [1,]  -0.0623  0.0075 0.053
##  [2,]  -0.0220  0.0160 0.054
##  [3,]  -0.0105  0.0117 0.054
##  [4,]   0.0511  0.0098 0.054
```

```
##  [5,]   0.0238  0.0088 0.054
##  [6,]  -0.0786  0.0092 0.054
##  [7,]  -0.0132  0.0094 0.054
##  [8,]   0.0110  0.0096 0.054
##  [9,]  -0.0981  0.0095 0.054
## [10,]  -0.1847  0.0095 0.054
## [11,]  -0.0852  0.0095 0.054
## [12,]   0.0215  0.0095 0.054
```

对预测结果的作图。这个预测是知道真实值的。

```
plot(window(vw, start=c(2007, 1), end=c(2008,12)),
     ylab="", type="b", ylim=c(-0.2, 0.2), lwd=2)
lines(ts(c(pred4$pred), start=c(2008,1), frequency = 12),
      col="red", lwd=1, lty=2, type="b", pch=2)
lines(ts(c(pred4$pred) - 2*c(pred4$se), start=c(2008,1), frequency = 12),
      col="green", lwd=1, lty=3, type="l")
lines(ts(c(pred4$pred) + 2*c(pred4$se), start=c(2008,1), frequency = 12),
      col="green", lwd=1, lty=3, type="l")
```

```
#foreplot(pred=pred4, rt=vw, orig=984, start=(83-2)*12+1, p=0.95)
```

图中黑色线是真实值，红色点是多步预测值，绿色线是 95% 预测界限。可以看出多步预测基本是按均值预测。

用 forecast 包计算超前多步预报：

```
fc <- forecast::forecast(resm4, h=12)
fc
```

```
##     Point Forecast       Lo 80      Hi 80       Lo 95     Hi 95
## 985    0.007455686 -0.06083694 0.07574831 -0.09698887 0.1119002
## 986    0.015952488 -0.05270459 0.08460957 -0.08904946 0.1209544
## 987    0.011701663 -0.05695841 0.08036174 -0.09330486 0.1167082
## 988    0.009806653 -0.05927763 0.07889093 -0.09584863 0.1154619
## 989    0.008770886 -0.06033071 0.07787249 -0.09691089 0.1144527
## 990    0.009164644 -0.05993698 0.07826627 -0.09651718 0.1148465
## 991    0.009432520 -0.05967458 0.07853962 -0.09625768 0.1151227
## 992    0.009565095 -0.05954249 0.07867268 -0.09612583 0.1152560
## 993    0.009530555 -0.05957703 0.07863814 -0.09616037 0.1152215
## 994    0.009495151 -0.05961250 0.07860281 -0.09619589 0.1151862
## 995    0.009477748 -0.05962992 0.07858541 -0.09621331 0.1151688
## 996    0.009480420 -0.05962725 0.07858808 -0.09621063 0.1151715
```

```
plot(fc, xlim=c(900, 1000))
points(985:996, vw[985:996])
```

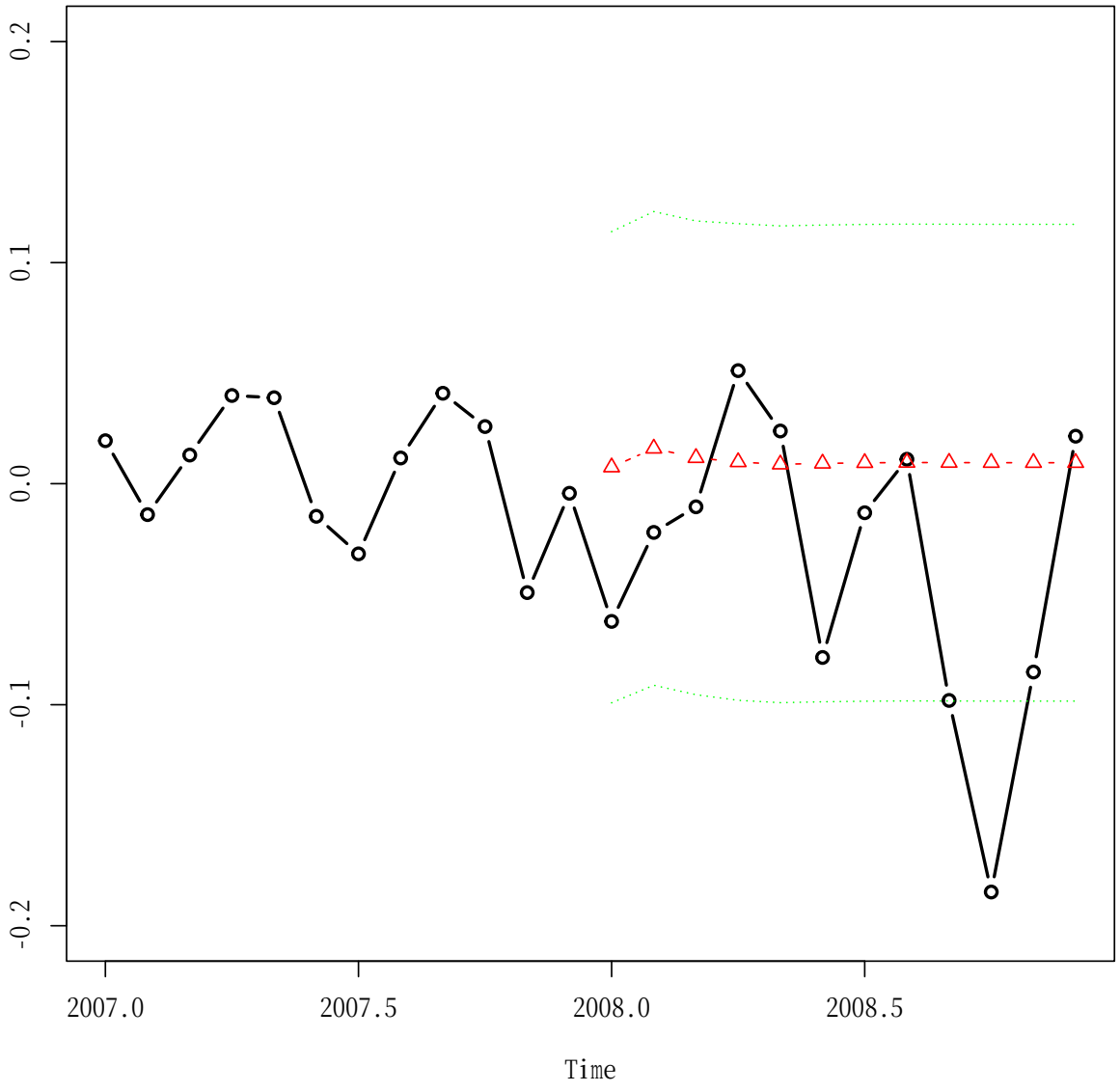
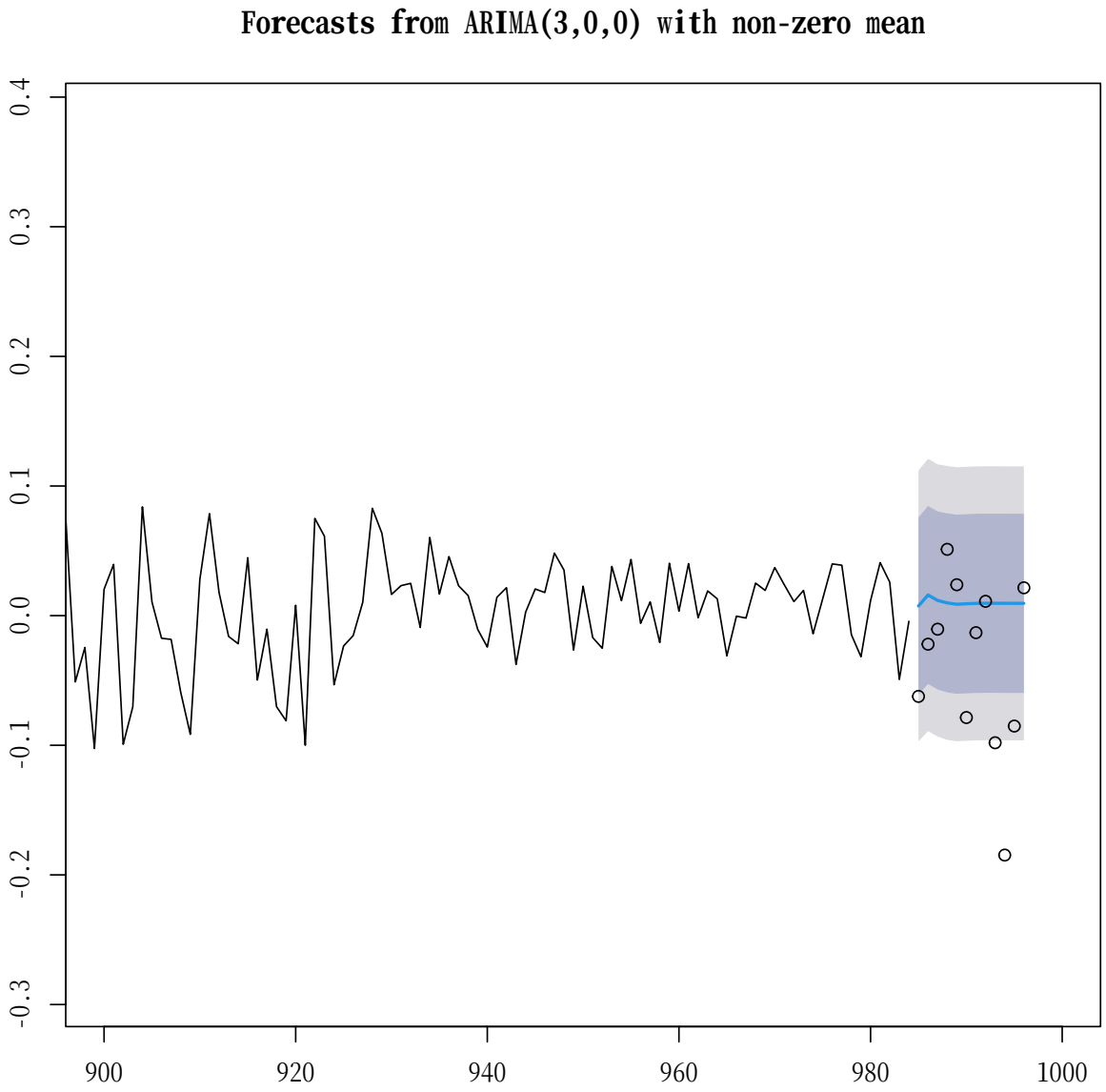


图 4.7: CRSP 价值加权简单月收益率 AR(3) 模型 12 步预测


Forecasts from ARIMA(3,0,0) with non-zero mean
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
900
920
940
960
980
1000

Chapter 5

移动平均模型

5.1 移动平均模型的概念

移动平均模型是具有 q 步外不相关性质的平稳列的模型；对于高阶的 AR 模型，有些可以用低阶的 MA 模型更好地描述。一般的 AR 模型也可以用高阶 MA 模型近似。

理论上，AR 模型也可以是无穷阶的：

$$X_t = \phi_0 + \sum_{j=1}^{\infty} \phi_j X_{t-j} + \varepsilon_t$$

其中 $\{\phi_j\}$ 应绝对可和。一个特例为

$$X_t = \phi_0 - \sum_{j=1}^{\infty} (-\theta_1)^j X_{t-j} + \varepsilon_t$$

其中 $0 < |\theta| < 1$。将模型写成：

$$X_t + \sum_{j=1}^{\infty} (-\theta_1)^j X_{t-j} = \phi_0 + \varepsilon_t \tag{*}$$

以 $t-1$ 代入，并乘以 $-\theta_1$，有

$$\sum_{j=1}^{\infty} (-\theta_1)^j X_{t-j} = -\phi_0\theta_1 - \theta_1\varepsilon_{t-1}$$

代入到 (*) 式中得

$$X_t = \phi_0(1+\theta_1) + \varepsilon_t + \theta_1\varepsilon_{t-1}$$

这样的模型称为 MA(1) 模型。

一般地，若 $\{\varepsilon_t\}$ 是零均值独立同分布白噪声，方差为 σ^2，$|\theta_1| < 1$，令

$$X_t = \theta_0 + \varepsilon_t + \theta_1\varepsilon_{t-1}$$

易见 $\{X_t\}$ 为线性时间序列形式的弱平稳列，称为 MA(1) 序列。

类似地，MA(2) 序列的模型为

$$X_t = \theta_0 + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2}$$

MA(q) 序列的模型为

$$X_t = \theta_0 + \varepsilon_t + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q}$$

此模型也有特征多项式

$$1+\theta_1 z+\cdots+\theta_q z^q$$

特征方程的根称为特征根，特征根都在单位圆外的条件称为 MA 模型的可逆条件。平稳性并不需要特征根的条件。

上面从 AR(∞) 导出 MA(1) 的过程，实际用了滞后算子的一些运算法则：设 $P(z)=\sum_{j=0}^{\infty}\phi_j z^j$ 和 $Q(z)=\sum_{j=0}^{\infty}\theta_j z^j$，$\sum_{j=0}^{\infty}|\phi_j|<\infty$, $\sum_{j=0}^{\infty}|\theta_j|<\infty$, 则 $P(z)Q(z)=R(z)=\sum_{j=0}^{\infty}r_j z^j$，且对弱平稳列 $\{\xi_t\}$ 有 $P(B)Q(B)\xi_t=R(B)\xi_t$。详见 (何书元, 2003)§2.1，以及李东风 “应用时间序列分析课堂演示”。

5.2 移动平均模型的性质

以 MA(1) 和 MA(2) 为例讨论 MA 序列的性质，一般 MA(q) 序列类似讨论即可。

5.2.1 平稳性与自相关函数性质

以 MA(1) 为例。$X_t=\theta_0+\varepsilon_t+\theta_1\varepsilon_{t-1}$，其中 $\{\varepsilon_t\}$ 是零均值独立同分布白噪声，θ_0,θ_1 是任意实数，平稳性不需要特征根的条件。

易见

$$EX_t=\theta_0,\ \forall t,\quad \mathrm{Var}(X_t)=\sigma^2(1+\theta_1^2)$$

而

$$\gamma_1=E[(X_t-\theta_0)(X_{t-1}-\theta_0)]=E[(\varepsilon_t+\theta_1\varepsilon_{t-1})(\varepsilon_{t-1}+\theta_1\varepsilon_{t-2})]=\theta_1 E\varepsilon_{t-1}^2=\sigma^2\theta_1$$

对 $k>1$ 有

$$\gamma_k=E[(\varepsilon_t+\theta_1\varepsilon_{t-1})(\varepsilon_{t-k}+\theta_1\varepsilon_{t-k-1})]=0\ (k>1)$$

因为 $k>1$，所以 $t-k-1<t-k<t-1<t$，求协方差时均不相关。

所以，对于 MA(1) 序列，有

$$\gamma_k=\begin{cases}\sigma^2(1+\theta_1^2), & k=0\\ \sigma^2\theta_1, & k=1,\\ 0, & k>1\end{cases}$$

相应地，MA(1) 的自相关函数为

$$\rho_k=\begin{cases}1, & k=0\\ \frac{\theta_1}{1+\theta_1^2}, & k=1,\\ 0, & k>1\end{cases}$$

这就验证了 MA(1) 序列是弱平稳列。MA(1) 的自相关函数在 $k>1$ 后为零的性质叫做 MA 序列的自相关函数截尾性。

对于 MA(q) 序列

$$X_t=\theta_0+\varepsilon_t+\theta_1\varepsilon_{t-1}+\cdots+\theta_q\varepsilon_{t-q}$$

易见

$$EX_t=\theta_0,\quad \mathrm{Var}(X_t)=\sigma^2(1+\theta_1^2+\cdots+\theta_q^2)$$

其自相关函数 ρ_k 也满足 q 后截尾性，即 $\rho_k=0,\forall k>q$。如果 $\theta_q\neq 0$，则 $\rho_q\neq 0$。这样，MA(q) 序列的两个时间点的观测 X_s 和 X_t 当 $|s-t|>q$ 时不相关，代表了一种特殊的 “有限记忆” 的模型。MA 序列的自相关函数截尾性也是在模型识别和定阶时的重要依据。

5.2.2 可逆性

对 MA(1) 模型，当 $|\theta_1| < 1$ 时，根据本章开始的推导可得

$$\varepsilon_t = -\phi_0 + X_t + \sum_{j=1}^{\infty} (-\theta_1)^j X_{t-j}$$

其中的级数是可以在 a.s. 意义和均方意义下收敛的。这表明新息 ε_t 可以用当前的观测 X_t 以及历史观测 $X_{t-j}, j = 1, 2, \ldots$ 的线性组合表示，而且历史观测 X_{t-j} 所在时刻离 t 时刻越远，其作用越小。这种性质叫做模型的可逆性。MA 模型的平稳性不需要可逆性条件，但是从理论讨论的角度，可逆的线性时间序列更合理：$\{X_t, X_{t-1}, \ldots\}$ 与 $\{\varepsilon_t, \varepsilon_{t-1}, \ldots\}$ 可以互相线性表示，对任意 $t \in \mathbb{Z}$ 成立。

5.3 移动平均模型定阶

MA(q) 序列的理论自相关函数 ρ_k 在 q 后截尾，$\rho_q \neq 0$, $\rho_k = 0$, $k > q$。

在 $\{X_t\}$ 为独立同分布白噪声列的条件下，$k > 0$ 的 $\hat{\rho}_k$ 渐近 $\mathrm{N}(0, \frac{1}{T})$ 分布，所以查看 ACF 图，最后一个显著不等于零的 $\hat{\rho}_k$ 的位置可以暂定为 MA 模型的阶。

实际上，如 $\{X_t\}$ 是 MA(q) 序列，则对 $k > q$，$\sqrt{T}\hat{\rho}_k$ 渐近服从正态分布，渐近均值为零，渐近方差为

$$1 + 2\rho_1^2 + \cdots + 2\rho_q^2$$

也可以用 AIC 定阶：

$$\mathrm{AIC}(k) = \ln \hat{\sigma}_k^2 + \frac{2k}{T}$$

其中 $\hat{\sigma}_k^2$ 是用 MA(k) 建模时新息方差的最大似然估计。

例 5.1. 考虑 CRSP 等权指数月度收益率，时间从 1926-1 到 2008-12。

```
d <- read_table2(
  "m-ibm3dx2608.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
```

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## i Please use `read_table()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
ibmind <- xts(as.matrix(d[,-1]), d$date)
rm(d)
tclass(ibmind) <- "yearmon"
ew <- ts(coredata(ibmind)[,"ewrtn"], start=c(1926,1), frequency=12)
head(ibmind)
```

```
##             ibmrtn     vwrtn     ewrtn     sprtn
## 1月 1926 -0.010381  0.000724  0.023174  0.022472
## 2月 1926 -0.024476 -0.033374 -0.053510 -0.043956
## 3月 1926 -0.115591 -0.064341 -0.096824 -0.059113
## 4月 1926  0.089783  0.038358  0.032946  0.022688
## 5月 1926  0.036932  0.012172  0.001035  0.007679
## 6月 1926  0.068493  0.056888  0.050487  0.043184
```

```
plot(ew, main="CRSP Equal Weighted Index Monthly Return")
abline(h=0, col="gray")
```

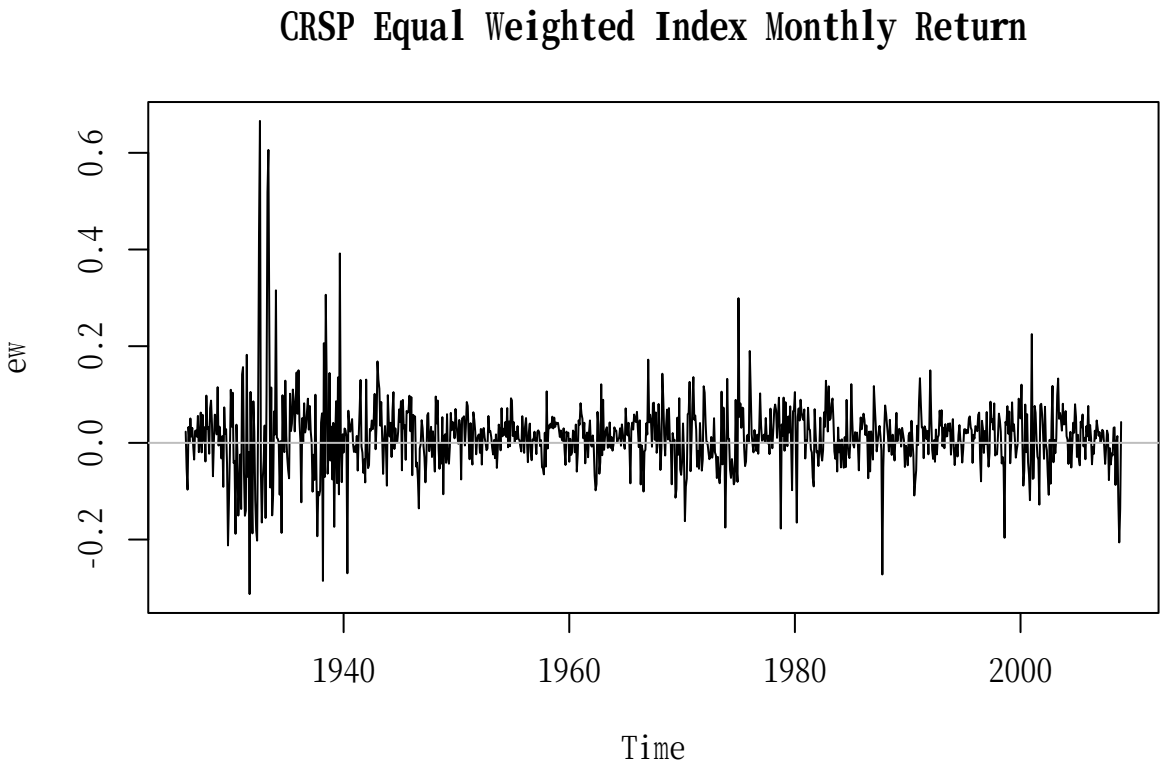


图 5.1: CRSP 等权指数月度收益率

```
forecast::Acf(ew, main="")
```

ACF 在 $k=1$ 很大，在 $k=3$ 和 $k=9$ 也比较明显。可以考虑拟合 MA(3) 或 MA(9)。

5.4 移动平均模型的估计

MA 模型参数的估计方法有：

- 矩估计法，利用 $\{\gamma_k\}$ 与 $\{\theta_k\}$、σ^2 的关系求非线性方程组解；

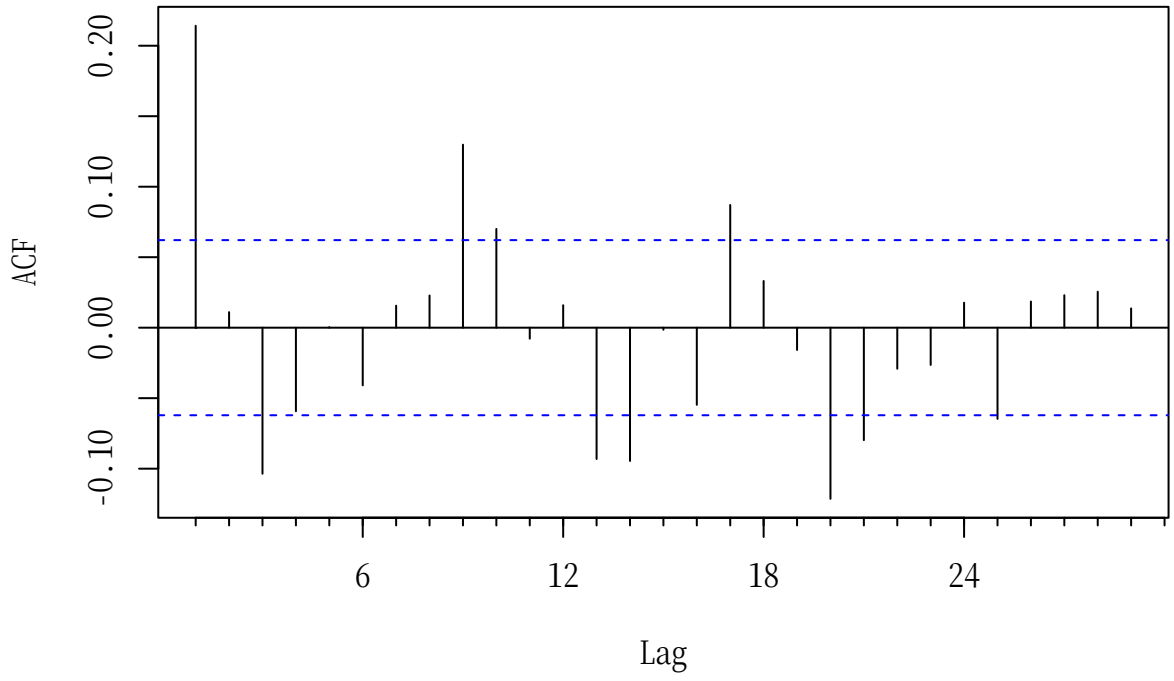


图 5.2: CRSP 等权指数月收益率的 ACF

- 逆相关函数法，将 MA 模型转换为长阶自回归模型，用估计自回归模型的方法估计，能保证可逆性；
- 新息估计法；
- 条件最大似然估计法；
- 精确最大似然估计法。

前面三种方法见 (何书元, 2003)§6.2。

条件最大似然估计法和完全最大似然估计法都假定 $\{\varepsilon_t\}$ 为高斯白噪声，计算似然函数。在条件最大似然估计法中，近似假定 $\varepsilon_t = 0, t \le 0$，这样就可以得到 $\varepsilon_1 = x_1 - \theta_0$, $\varepsilon_2 = x_2 - \theta_0 - \theta_1\varepsilon_1$ 等递推表示，将其代入 $\varepsilon_t, t = 1, 2, \dots, T$ 的独立联合正态分布密度中就得到了条件似然函数，求其关于 σ^2 和 $\theta_0, \theta_1, \dots, \theta_q$ 的最大值点。

在精确最大似然估计中，将 $\varepsilon_t, t = 1-q, 2-q, \dots, -1, 0$ 也作为未知参数，与其它模型参数一起估计。

条件最大似然估计更容易计算，在 T 充分大时两者的结果趋于相同。在样本量较小时精确最大似然估计结果更为精确。见 (Tsay, 2010) 第 8 章。

例 5.2. 考虑例5.1中的 CRSP 等权指数月度收益率用 MA 建模，时间从 1926-1 到 2008-12。

例5.1提示建立 MA(9)。在 R 中用 `arima()` 函数可以建立 AR 模型和 MA 模型。

```
resm1 <- arima(ew, order=c(0,0,9)); resm1
```

```
##
## Call:
## arima(x = ew, order = c(0, 0, 9))
##
## Coefficients:
##          ma1     ma2      ma3      ma4     ma5      ma6     ma7      ma8
##       0.2144  0.0374  -0.1203  -0.0425  0.0232  -0.0302  0.0482  -0.0276
## s.e.  0.0316  0.0321   0.0328   0.0336  0.0319   0.0318  0.0364   0.0354
##          ma9  intercept
##       0.1350     0.0122
## s.e.  0.0323     0.0028
##
## sigma^2 estimated as 0.005043:  log likelihood = 1220.86,  aic = -2419.72
```

对残差作 Ljung-Box 白噪声检验:

```
Box.test(resm1$residuals, type="Ljung", lag=12, fitdf=9)
```

```
##
##  Box-Ljung test
##
## data:  resm1$residuals
## X-squared = 6.0921, df = 3, p-value = 0.1072
```

结果不显著，检验结果支持所建立的模型。

从结果的标准误差构造近似 95% 置信区间，可以看出 θ_k 在 $k = 2, 4, 5, 6, 7, 8$ 处不显著。所以，可以用 `arima()` 函数的 `fixed=` 指定这些参数固定为 0：

```
resm1b <- arima(ew, order=c(0,0,9),
                fixed=c(NA,0,NA,0,0,0,0,0,NA,NA))
resm1b
```

```
##
## Call:
## arima(x = ew, order = c(0, 0, 9), fixed = c(NA, 0, NA, 0, 0, 0, 0, 0, NA, NA))
##
## Coefficients:
##          ma1  ma2      ma3  ma4  ma5  ma6  ma7  ma8     ma9  intercept
##       0.1909    0  -0.1199    0    0    0    0    0  0.1227     0.0122
## s.e.  0.0293    0   0.0338    0    0    0    0    0  0.0312     0.0027
##
## sigma^2 estimated as 0.005097:  log likelihood = 1215.61,  aic = -2421.22
```

没有限定之前，AIC 为 -2419.72；限定后，AIC 为 -2421.22，限定后的 AIC 更优。

为了比较不同的模型，可以逐个尝试不同的模型并比较 AIC 的值。

注意，R 的 `stats::arima()` 函数在估计 ARMA(p,q) 模型时，输出结果的对应模型格式为

$$(1-\phi_1 B-\cdots-\phi_p B^p)(X_t-\mu)=(1+\theta_1 B+\cdots+\theta_q B^q)\varepsilon_t$$

其中 μ 对应于输出中的截距项 (intercept)。系数会输出为 `ar1`, ……, `arp`, `ma1`, ……, `maq`, `intercept` 的次序，`ar`k 对应 ϕ_k，`ma`k 对应 θ_k，`intercept` 对应 μ。

5.5 移动平均模型的预测

因为 MA(q) 序列在间隔超过 q 步以后就独立，所以超前多步预测，只能预测到 q 步，从 $q+1$ 步开始就只能用均值 μ 预测了。

以 MA(1) 为例，

$$X_t=\theta_0+\varepsilon_t+\theta_1\varepsilon_{t-1}$$

超前一步：

$$\hat{x}_h(1)=E(X_{h+1}|x_1,\ldots,x_h)=\theta_0+\theta_1\varepsilon_h$$

这里利用了 $E(\varepsilon_{h+1}|x_1,\ldots,x_h)=0$。$\varepsilon_h$ 是第 h 个新息，可以作为模型的残差计算，或者通过将 MA 模型表达为 AR 模型来计算。较精确的做法是 “递推预报”，参见 (何书元, 2003)§5.3。

超前两步:

$$\hat{x}_h(2)=E(\theta_0+\varepsilon_{h+2}+\theta_1\varepsilon_{h+1}|x_1,\ldots,x_h)=\theta_0$$

从两步开始的超前多步预报就变成 $EX_t=\theta_0$ 了。

类似地，对于 MA(2) 序列，

$$\hat{x}_h(1)=\theta_0+\theta_1\varepsilon_h+\theta_2\varepsilon_{h-1},\ \hat{x}_h(2)=\theta_0+\theta_2\varepsilon_h$$

对 $k = 3, 4, \ldots$ 则有 $\hat{x}_h(k) = \theta_0 = EX_t$。

在 R 软件中，用 `stats::arima()` 函数建模后，对建模结果用 `predict()` 函数计算预测值和对应的近似标准误差。

例 5.3. 考虑例5.2中的 CRSP 等权指数月度收益率用稀疏系数的 MA(9) 建模，但保留最后 10 个月的数据作为验证。原始数据中时间从 1926-1 到 2008-12。

```
resm1c <- arima(ew[1:986], order=c(0,0,9),
                fixed=c(NA,0,NA,0,0,0,0,0,NA,NA))
resm1c
```

```
##
## Call:
## arima(x = ew[1:986], order = c(0, 0, 9), fixed = c(NA, 0, NA, 0, 0, 0, 0, 0,
##     NA, NA))
##
## Coefficients:
##          ma1  ma2      ma3  ma4  ma5  ma6  ma7  ma8     ma9  intercept
##       0.1844    0  -0.1206    0    0    0    0    0  0.1218     0.0128
## s.e.  0.0295    0   0.0338    0    0    0    0    0  0.0312     0.0027
##
## sigma^2 estimated as 0.005066:  log likelihood = 1206.44,  aic = -2402.88
```

```
pred1c <- predict(resm1c, n.ahead=10, se.fit=TRUE)
tmp.tab <- cbind(Observed=round(c(ew[987:996]), 4),
                 Predicted=round(c(pred1c$pred), 4),
                 SE=round(c(pred1c$se), 4))
row.names(tmp.tab) <- sprintf("2008-%02d", 3:12)
tmp.tab
```

```
##         Observed Predicted     SE
## 2008-03  -0.0260    0.0043 0.0712
## 2008-04   0.0312    0.0136 0.0724
## 2008-05   0.0322    0.0150 0.0724
## 2008-06  -0.0871    0.0145 0.0729
## 2008-07  -0.0010    0.0120 0.0729
## 2008-08   0.0141    0.0018 0.0729
## 2008-09  -0.1209    0.0122 0.0729
## 2008-10  -0.2060    0.0055 0.0729
## 2008-11  -0.1366    0.0085 0.0729
## 2008-12   0.0431    0.0128 0.0734
```

因为次贷危机影响，实际收益率不如预测的那么好。可以看出当 $k = 10$ 的时候（模型 $q = 9$）预测等于序列均值。超前多步预测的标准误差逐渐增加到等于序列的样本标准差：

```r
sd(c(ew[1:986]))
```

```
## [1] 0.07368157
```

5.6 AR 和 MA 的小结

- 对 MA(q) 模型，ACF 对定阶有意义，因为其 q 后截尾；
- 对 AR(p) 模型，PACF 对定阶有意义，因为其 p 后截尾；
- MA 模型的序列不管系数如何总是平稳的，实际上还是因果线性时间序列，当特征根都在单位圆外时是可逆的；
- AR 模型只有当特征根都在单位圆外时才有 ϵ_t 与 $X_{t-1}, X_{t-2}, \ldots$ 独立的弱平稳解；
- 对 AR 和 MA 序列，超前多步预测趋于序列的均值，预测均方误差趋于序列的方差。

Chapter 6

ARMA 模型

6.1 ARMA 模型的概念

AR 模型有偏自相关函数截尾性质；MA 模型有相关函数截尾性质。有些因果线性时间序列有与 AR 和 MA 类似的表现，但是不能在低阶实现偏自相关函数截尾或者相关函数截尾。

ARMA 模型结合了 AR 和 MA 模型，在对数据拟合优度相近的情况下往往可以得到更简单的模型，而且不要求偏自相关函数截尾也不要求相关函数截尾。

ARMA(1,1) 模型为

$$X_t = \phi_0 + \phi_1 X_{t-1} + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

或

$$X_t - \phi_1 X_{t-1} = \phi_0 + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

其中 $|\phi_1| < 1, |\theta_1| < 1, -\phi_1 \neq \theta_1$。$\{\varepsilon_t\}$ 是独立同分布零均值白噪声列, ε_t 与 $X_{t-1}, X_{t-2}, ...$ 独立。

一般的 ARMA(p, q) 类似。

AR(p) 可以看成 ARMA$(p, 0)$, MA(q) 可以看成是 ARMA$(0, q)$。

在 ARMA(1,1) 的系数条件中，$|\phi_1| < 1$ 是平稳解条件，$|\theta_1| < 1$ 是可逆性条件，$-\phi_1 \neq \theta_1$ 是为了模型不至于退化：模型也可以写成

$$(1 - \phi_1 B) X_t = \phi_0 + (1 + \theta_1 B) \varepsilon_t$$

如果不加这个条件，两边的滞后算子的多项式就可以消去。

6.2 ARMA 模型的性质

形式地，

$$\begin{aligned}
\frac{1}{1-\phi_1 z} =&\sum_{j=0}^{\infty}\phi_1^j z^j,\\
\frac{1}{1-\phi_1 B} =&\sum_{j=0}^{\infty}\phi_1^j B^j,\\
\frac{1+\theta_1 z}{1-\phi_1 z} =&1+(\phi_1+\theta_1)\sum_{j=1}^{\infty}\phi_1^{j-1} z^j,\\
\frac{1+\theta_1 B}{1-\phi_1 B} =&1+(\phi_1+\theta_1)\sum_{j=1}^{\infty}\phi_1^{j-1} B^j,
\end{aligned}$$

于是

$$\begin{aligned}
X_t =&\frac{1}{1-\phi_1 B}\left\{\phi_0+(1+\theta_1 B)\varepsilon_t\right\}\\
=&\frac{\phi_0}{1-\phi_1}+\frac{1+\theta_1 B}{1-\phi_1 B}\varepsilon_t\\
=&\frac{\phi_0}{1-\phi_1}+\varepsilon_t+(\phi_1+\theta_1)\sum_{j=1}^{\infty}\phi_1^{j-1}\varepsilon_{t-j}
\end{aligned}\tag{6.1}$$

这是因果型线性时间序列，是弱平稳的，满足上述 ARMA(1,1) 模型方程。

$$\begin{aligned}
EX_t =&\frac{\phi_0}{1-\phi_1},\\
\mathrm{Var}(X_t) =&\sigma^2\left(1+(\phi_1+\theta_1)^2\sum_{j=1}^{\infty}(\phi_1^2)^{j-1}\right)=\sigma^2\frac{1+\theta_1^2+2\phi_1\theta_1}{1-\phi_1^2}
\end{aligned}$$

由方差的表达式可知 $|\phi_1|<1$ 是平稳性的必要条件。

求自协方差函数和自相关函数。因为协方差与均值无关，而 ϕ_0 只影响到均值，所以求协方差与自相关函数时可以假定 $\phi_0=0$。对

$$X_t-\phi_1 X_{t-1}=\varepsilon_t+\theta_1\varepsilon_{t-1}\tag{6.2}$$

在(6.2)两边乘以 X_{t-1} 并取期望得

$$\gamma_1-\phi_1\gamma_0=\theta_1 E(\varepsilon_{t-1}X_{t-1})$$

由 (*) 可知 $E(\varepsilon_t X_t)=\sigma^2$, 所以 $E(\varepsilon_{t-1}X_{t-1})=\sigma^2$

$$\begin{aligned}
\gamma_1-\phi_1\gamma_0=&\sigma^2\theta_1,\\
\gamma_1=&\sigma^2\frac{(\phi_1+\theta_1)(1+\theta_1\phi_1)}{1-\phi_1^2}
\end{aligned}$$

在(6.2)两边乘以 $X_{t-k}(k\geq 2)$ 并取期望，得

$$\gamma_k-\phi_1\gamma_{k-1}=0,\quad \gamma_k=\phi_1\gamma_{k-1}=\phi_1^{k-1}\gamma_1$$

所以 ARMA(1,1) 的自相关函数为

$$\rho_k = \begin{cases} \dfrac{(\phi_1+\theta_1)(1+\phi_1\theta_1)}{1+2\phi_1\theta_1+\theta_1^2}, & k=1 \\ \phi_1\rho_{k-1} = \phi_1^{k-1}\rho_1, & k\geq 2 \end{cases}$$

所以 ARMA(1,1) 的 ACF 与 AR(1) 的 ACF 很相似，但是从 $k=2$ 处才开始负指数衰减。与 AR 类似，自相关函数不能有限步截尾。

ARMA(1,1) 的偏自相关函数与 MA(1) 的偏自相关函数类似，但负指数衰减从 $k=2$ 开始，也不能在有限步截尾。

总之，ARMA(1,1) 的平稳性条件与 AR(1) 相同，自相关函数与偏自相关函数均不能有限步截尾 (设 $\phi_1\neq 0, \theta_1\neq 0$)。

6.3 一般 ARMA 模型

一般 ARMA 模型为

$$X_t = \phi_0 + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q}$$

其中 $\{\varepsilon\}$ 为独立同分布零均值白噪声列，ε_t 与 $X_{t-1}, X_{t-2}, \ldots$ 独立。$1-\phi_1 z - \cdots - \phi_p z^p$ 称为特征多项式，特征多项式的根都在单位园外，这是平稳性条件。一般要求 $1+\theta_1 z + \cdots + \theta_q z^q$ 的根也都在单位圆外，这个条件称为可逆性条件。两个多项式没有公共根，否则同一模型可能会有不同的表示。

平稳解的均值为

$$EX_t = \frac{\phi_0}{1-\phi_1-\cdots-\phi_p}$$

6.4 ARMA 模型辨识

可以逐个从低阶模型尝试，$p+q$ 越小越好，找到 AIC 最小的选择，用精确最大似然或者条件最大似然方法估计参数。对残差进行白噪声检验以验证模型是否充分。

R 的 forecast 包提供了一个 `auto.arima()` 函数，可以自动进行模型选择。TSA 包提供了一个 `armasubsets()` 函数用于模型选择。

Tsay 和 Tiao(1984) 提出了一个对 ARMA 定阶的辅助工具 EACF，其结果可以用与 (p,q) 有关的二维表格表示，结果包含由字母 “O” 组成的三角形，锐角顶点在 (p,q) 位置。如

	MA							
AR	0	1	2	3	4	5	6	7
0	X	X	X	X	X	X	X	X
1	X	O	O	O	O	O	O	O
2	$*$	X	O	O	O	O	O	O
3	$*$	$*$	X	O	O	O	O	O
4	$*$	$*$	$*$	X	O	O	O	O
5	$*$	$*$	$*$	$*$	X	O	O	O

例 6.1. 考虑 3M 公司股票从 1946 年 2 月到 2008 年 12 月的月对数收益率，共有 755 个观测。

```
d <- read_table2(
  "m-3m4608.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
```

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## i Please use `read_table()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
mmm <- xts(log(1 + d[["rtn"]]), d$date)
rm(d)
tclass(mmm) <- "yearmon"
ts.3m <- ts(coredata(mmm), start=c(1946,2), frequency=12)
head(ts.3m)
```

```
##           Series 1
## [1,] -0.081125460
## [2,]  0.018421282
## [3,] -0.105360516
## [4,]  0.190518702
## [5,]  0.005114897
## [6,]  0.073743834
```

```
plot(ts.3m, main="3M Monthly Log Return")
abline(h=0, col="gray")
```

ACF 图形：

```
forecast::Acf(ts.3m, main="")
```

ACF 很接近于白噪声。

PACF 图形：

```
pacf(ts.3m, main="")
```

PACF 也比较接近于白噪声但是有比较多的超出界限的值，尽管超出量不大。

用 TSA 包提供的 `eacf()` 函数辨识模型：

```
TSA::eacf(ts.3m, 6, 12)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12
## 0 o o x o o x o o o x o  x  o
## 1 x o x o o x o o o o o  x  o
## 2 x x x o o x o o o o o  o  o
## 3 x x x o o o o o o o o  o  o
```

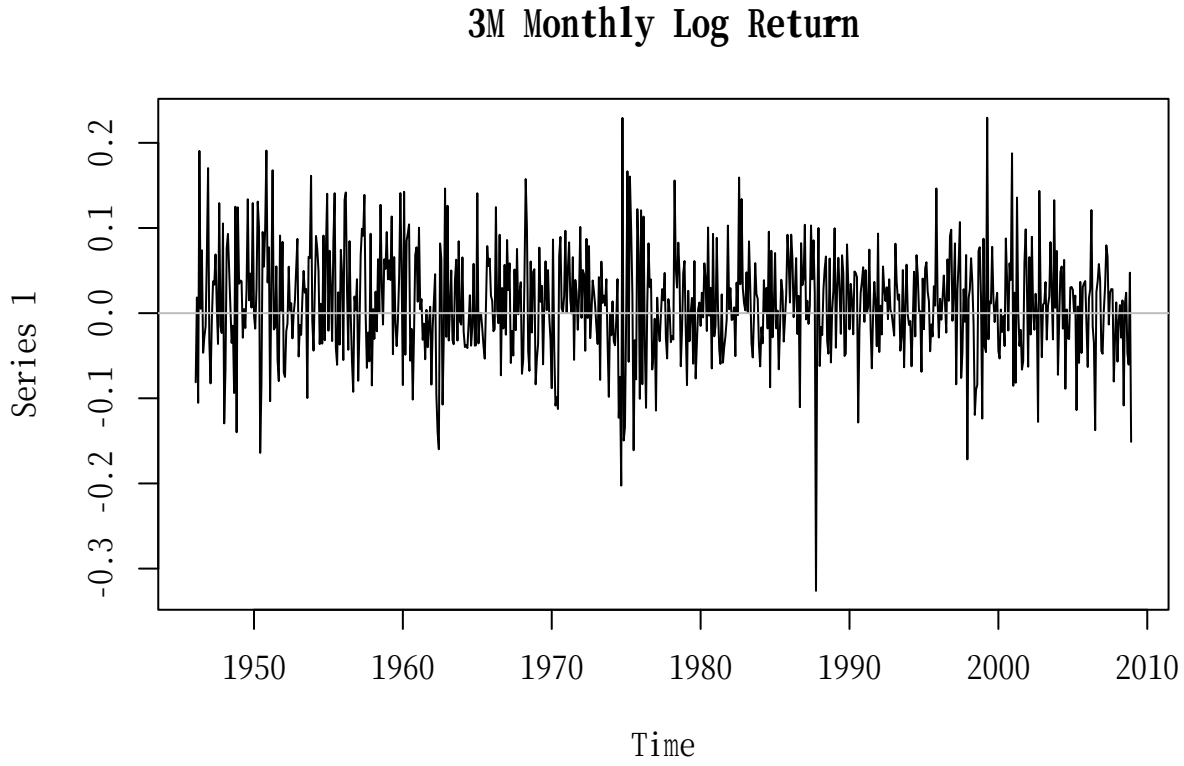


图 6.1: 3M 公司月对数收益率

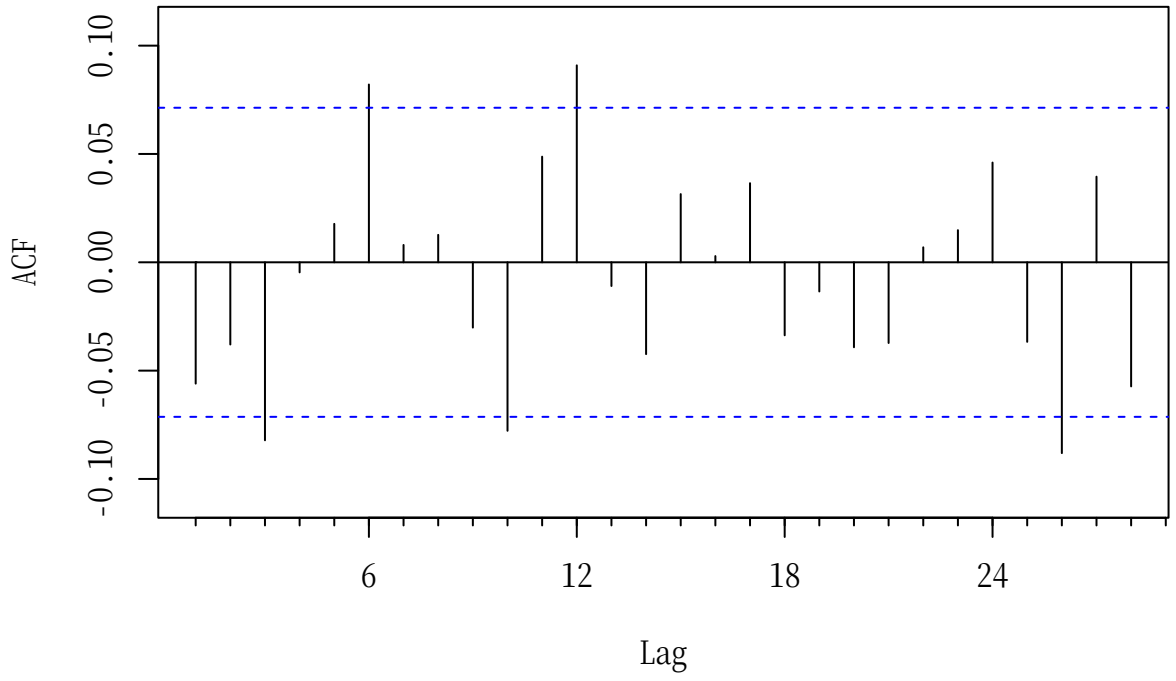


图 6.2: 3M 公司月对数收益率的 ACF

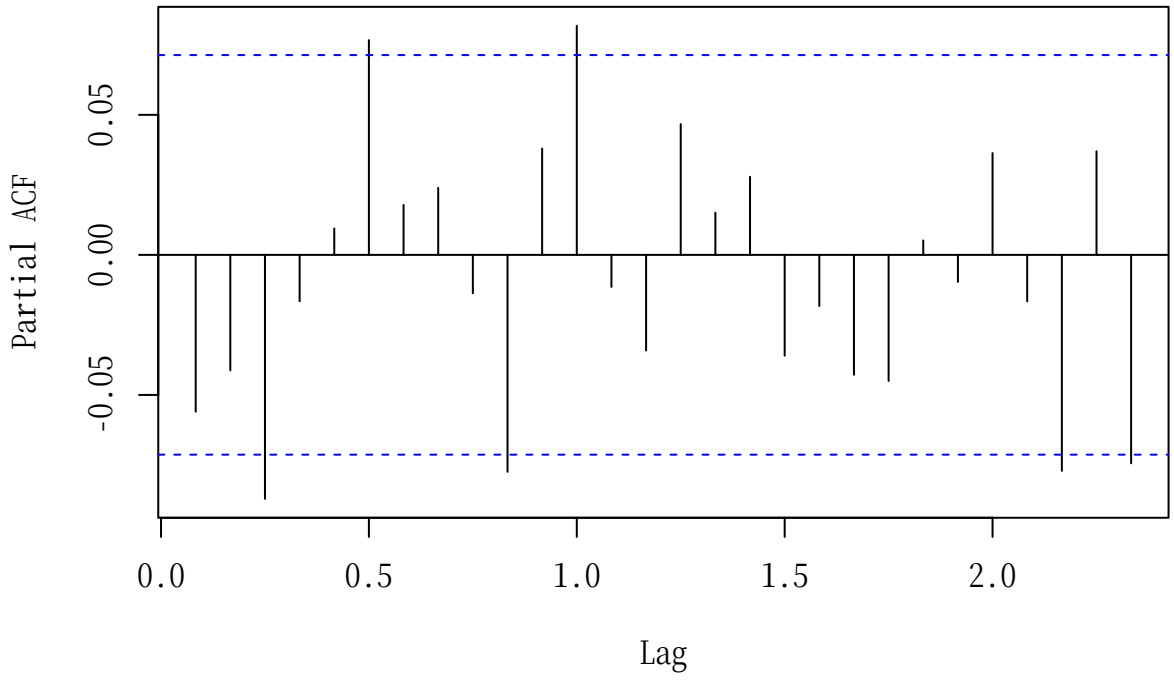


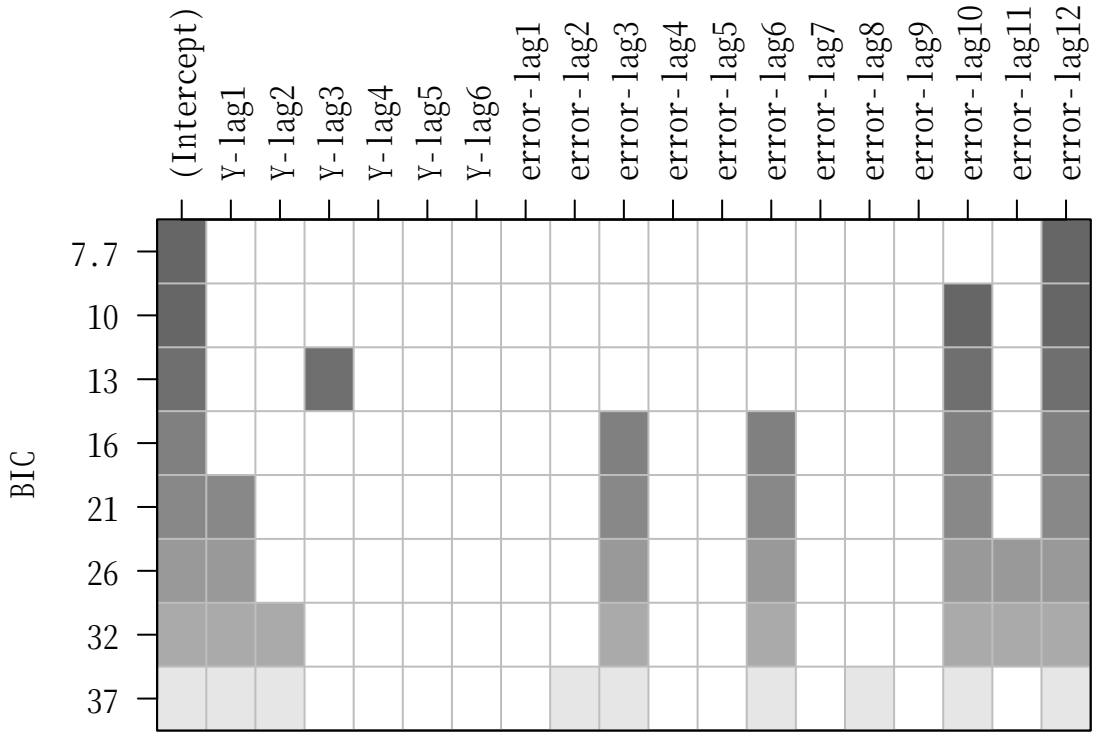
图 6.3: 3M 公司月对数收益率的 PACF

```
## 4 x o x o o o o o o o o  o  o
## 5 x x x o x o o o o o o  o  o
## 6 x x x x x o o o o o o  o  o
```

从上面的结果不易选择 ARMA 阶。函数的默认检验系数非零的检验水平是 0.05 水平。如果用 0.01 水平，则选择 $p = q = 0$。

TSA 包还提供了 `armasubsets()` 函数用来选择 ARMA 模型阶，办法是用长阶自回归获得新息 ε_t 的估计，然后用普通最小二乘估计 ARMA 系数，用回归的自变量选择方法进行模型选择。如：

```
resr <- TSA::armasubsets(ts.3m, nar = 6, nma = 12)
plot(resr)
```



选项 `nar` 和 `nma` 分别是 AR 部分和 MA 部分最多考虑的阶数，图形中方格的每一行是一种回归子集选择，默认采用 BIC 比较不同模型，可以用 `scale` 选项选择其它比较准则，结果第一行对应于 BIC 最优的模型，从结果看，是仅有截距项和 θ_{12} 的模型。

用 forecasts 包的 `auto.arima()` 函数定阶：

```
forecast::auto.arima(
  ts.3m, max.p = 6, max.q = 6,
  max.P = 1, max.Q = 1)
```

```
## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
```

```
##   plot.Arima   TSA
```

```
## Series: ts.3m
## ARIMA(3,0,1)(1,0,1)[12] with non-zero mean
##
## Coefficients:
##          ar1      ar2      ar3      ma1    sar1     sma1    mean
##       0.0453  -0.0285  -0.0837  -0.1124  0.5319  -0.4435  0.0103
## s.e.  0.3146   0.0417   0.0387   0.3147  0.2885   0.3049  0.0023
##
## sigma^2 = 0.003998:  log likelihood = 1016.63
## AIC=-2017.25   AICc=-2017.06   BIC=-1980.24
```

auto.arima() 函数选择了一个季节 ARMA 模型。参数 max.p, max.q, max.P, max.Q 分别指定要考虑的模型各个最大阶数。

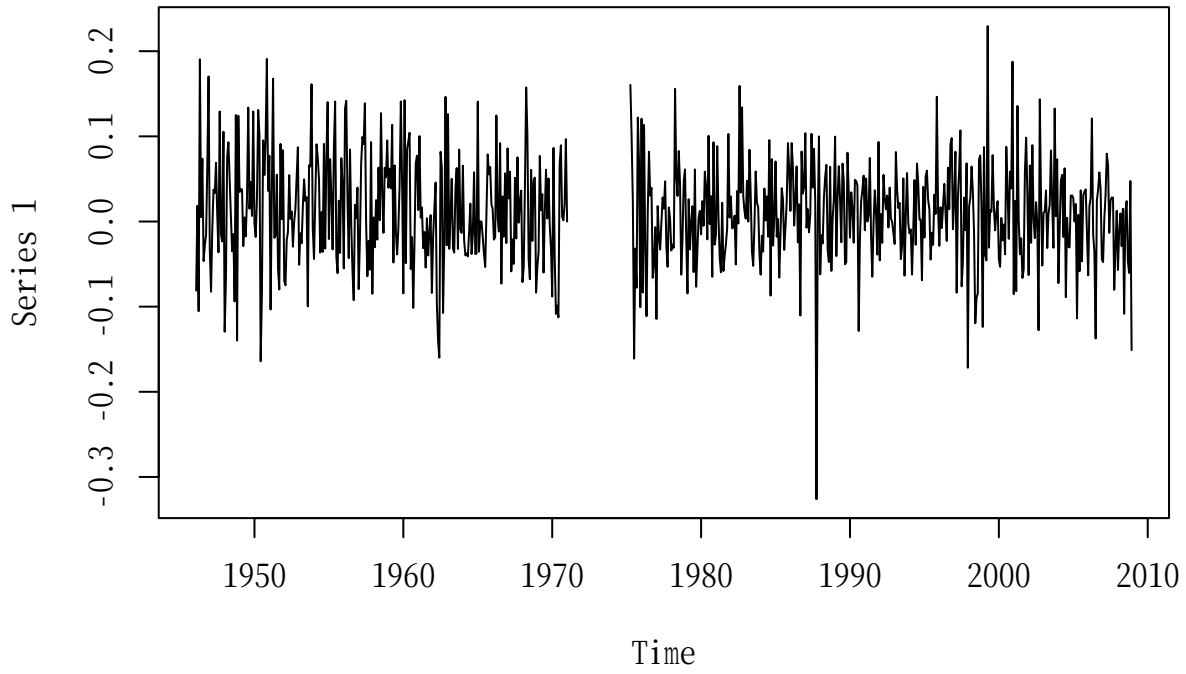
使用 stats::arima() 估计模型：

```
arima(ts.3m, order = c(3, 0, 1),
  seasonal = list(
    order = c(1, 0, 1), period = 12))
```

```
##
## Call:
## arima(x = ts.3m, order = c(3, 0, 1), seasonal = list(order = c(1, 0, 1), period = 12))
##
## Coefficients:
##          ar1      ar2      ar3      ma1    sar1     sma1  intercept
##       0.0453  -0.0285  -0.0837  -0.1124  0.5319  -0.4435     0.0103
## s.e.  0.3146   0.0417   0.0387   0.3147  0.2885   0.3049     0.0023
##
## sigma^2 estimated as 0.003961:  log likelihood = 1016.63,  aic = -2017.25
```

stats::arima() 是使用状态空间模型卡尔曼滤波来进行精确最大似然估计的。状态空间模型能够自然地处理缺失值问题，所以我们测试有缺失时的情况：

```
ts.3m.miss <- ts.3m
ts.3m.miss[301:350] <- NA
plot(ts.3m.miss)
```




```
arima(ts.3m.miss, order = c(3, 0, 1),
  seasonal = list(
    order = c(1, 0, 1), period = 12))
```

```
##
## Call:
## arima(x = ts.3m.miss, order = c(3, 0, 1), seasonal = list(order = c(1, 0, 1),
##     period = 12))
##
## Coefficients:
##          ar1      ar2      ar3      ma1    sar1     sma1  intercept
##       0.1638  -0.0369  -0.0977  -0.2174  0.7152  -0.6368     0.0107
## s.e.  0.2187   0.0400   0.0407   0.2176  0.2258   0.2481     0.0024
##
## sigma^2 estimated as 0.003841:  log likelihood = 960.01,  aic = -1904.02
```

6.5 ARMA 模型预测

ARMA(p,q) 的预测与 AR 和 MA 的预测都有关系。对 AR 部分仍是递推地向前预测，对 MA 部分，需要估计新息 ε_t 的值。

超前一步预测：

$$\hat{x}_h(1) = \phi_0 + \sum_{j=1}^{p} \phi_j x_{h+1-j} + \sum_{j=1}^{q} \theta_j \hat{\varepsilon}_{h+1-j}$$

其中 $\hat{\varepsilon}_i = E(\varepsilon_i | X_1, ..., X_h)$。

对超前 k 步预测有

$$\hat{x}_h(k) = \phi_0 + \sum_{j=1}^{p} \phi_j \hat{x}_{h+k-j} + \sum_{j=1}^{q} \theta_j \hat{\varepsilon}_{h+k-j}$$

其中 $h+k-j \le h$ 时 $\hat{x}_{h+k-j} = x_{h+k-j}$, $h+k-j > h$ 时 $\hat{\varepsilon}_{h+k-j} = 0$。预测误差为 $e_h(k) = x_{h+k} - \hat{x}_{h+k}$。

6.6 ARMA 模型的三种表示

设 ARMA(p,q) 模型的系数满足平稳性条件与可逆性条件。

第一种表示：

$$(1 - \phi_1 B - \cdots - \phi_p B^p) X_t = \phi_0 + (1 + \theta_1 B + \cdots + \theta_q B^q) \varepsilon_t$$

6.6.1 ARMA 模型的 MA 表示

用滞后算子的性质，令 $P(z) = 1 - \sum_{j=1}^{p} \phi_p z^j$, $Q(z) = 1 + \sum_{j=1}^{q} \theta_q z^j$，则

$$\Psi(z) = \frac{Q(z)}{P(z)} = \sum_{j=0}^{\infty} \psi_j z^j,$$

其中 $\{\psi_j\}$ 绝对可和（实际上，$j \to \infty$ 时 c_j 以负指数速度衰减）。所以

$$X_t = \mu + \Psi(B)\varepsilon_t = \mu + \sum_{j=0}^{\infty} \psi_j \varepsilon_{t-j},\ t \in \mathbb{Z}$$

这称为 ARMA 模型的 MA 表示或者 Wold 表示。$\mu = EX_t = \phi_0 / P(1) = \phi_0 / (1 - \phi_1 - \cdots - \phi_p)$。$\psi_0 = 1$。系数 $\{\psi_j, j = 0, 1, ...\}$ 称为 ARMA 模型的脉冲响应函数或者 Wold 系数。ψ_j 是脉冲响应函数的意义是，如果 $\epsilon_t = 1$，它将给 X_{t+j} 施加一个 ψ_j 的增量影响。

ψ_j 随 $j \to +\infty$ 以负指数速度衰减，这样的性质是合理的，即一个时刻发生的事情的影响应该随历史向前推进而迅速降低。

MA 表示对估计多步预测的均方误差有影响。理论上

$$E(X_{h+k} | \mathscr{F}_h) = \mu + \sum_{i=0}^{\infty} \psi_{i+k} \varepsilon_{h-i} \tag{6.3}$$

其中 $\mathscr{F}_h$ 是包含 $\{X_s, s \le t\}$ 的最小 σ 代数，表示到时刻 h 为止的观测信息，$E(\varepsilon_{h+k} | \mathscr{F}_h) = 0$, $k \ge 1$。于是理论的预测误差为

$$e_h(k) = X_{h+k} - E(X_{h+k} | \mathscr{F}_h) = \sum_{j=0}^{k-1} \psi_j \varepsilon_{h+k-j}$$

均方误差为

$$\mathrm{Var}(e_h(k)) = \sigma^2 \sum_{j=0}^{k-1} \psi_j^2$$

因为 $\{\psi_j\}$ 绝对可和，所以(6.3)中的级数当 $k \to \infty$ 时趋于 0，于是长期预测值趋于 μ，这称为均值反转。当 $k \to \infty$ 是多步预测的均方误差趋于 $\sigma^2 \sum_{j=0}^{\infty} \psi_j^2 = \mathrm{Var}(X_t)$，即用均值作为预测的均方误差。

6.6.2 ARMA 模型的 AR 表示

当平稳性与可逆性条件都成立时，令

$$\pi(z) = \frac{P(z)}{Q(z)}$$

则

$$\pi(z) = \sum_{j=0}^{\infty} \pi_j z^j,$$

$\{\pi_j\}$ 绝对可和，当 $j \to \infty$ 时 π_j 以负指数速度收敛到 0。$\pi_0 = 1$。于是 ARMA 模型有如下的 AR 表示

$$\pi(B)X_t = \frac{\phi_0}{Q(1)} + \varepsilon_t$$

或

$$X_t = \tilde{\phi}_0 - \pi_1 X_{t-1} - \pi_2 X_{t-2} - \cdots + \varepsilon_t$$

其中 $\tilde{\phi}_0 = \phi_0/Q(1) = \phi_0/(1 + \theta_1 + \cdots + \theta_q)$。这称为 ARMA 的 AR 表示或者长阶自回归形式。这说明平稳可逆 ARMA 序列可以用自回归模型近似表示。π_j 称为 ARMA 模型的 π 权重。

6.7 附录

文献和历史：

- EVGENIJ EVGENIEVICH SLUTZKY 和 GEORGE UDNY YULE 在 19 世纪初提出了 MA 模型和 AR 模型。
- HERMAN WOLD(1938), A Study in the Analysis of Stationary Time Series, Almquist and Wicksell, Stockholm. 博士论文。完善了线性时间序列模型。
- GEORGE E.P. BOX and GWILYM M. JENKINS(1970), Time Series Analysis: Forecasting and Control, Holden Day, San Francisco et al.; 2nd enlarged edition 1976. 本书给出了应用线性时间序列模型的理论与方法。
- 下面的文献提出，一元时间序列模型短期预测常常优于大量经济变量的联立方程的预测：CLIVE W.J. GRANGER and PAUL NEWBOLD(1975), Economic Forecasting: The Atheist's Viewpoint, in: G.A. RENTON (ed.), Modelling the Economy, Heinemann, London pp. 131 – 148.

Chapter 7

单位根过程

前面的 AR、MA、ARMA 主要应用于简单收益率和对数收益率。对于价格序列，一般其水平是缓慢变化的，包括缓慢的增长趋势与一定的周期波动。这样的序列不满足弱平稳的条件，是非平稳时间序列。

典型的非平稳时间序列模型是单位根 (unit root) 非平稳时间序列。

7.1 随机游动

考虑 $\{p_t\}$ 的模型

$$p_t = p_{t-1} + \varepsilon_t,\ t = 1, 2, ... \tag{7.1}$$

其中 $\{\varepsilon_t\}$ 是零均值独立同分布白噪声列。称 $\{p_t\}$ 是一个**随机游动** (random walk)。

如果 p_t 是股票的对数价格，p_0 是初始上市 (initial public offering) 的对数价格 (即 IPO 对数价格)。若 ε_t 的分布关于 0 对称，则给定 p_{t-1} 的条件下预测 p_t，上升与下降的概率均为 $\frac{1}{2}$，无法预测升降。

(7.1)表面上看起来像是 AR(1) 模型，但是 $\phi_1 = 1$ 不满足 AR(1) 的平稳性条件，而且

$$p_t = p_0 + \sum_{j=0}^{t-1} \varepsilon_{t-j} \tag{7.2}$$

于是

$$E(p_t|p_0) = p_0, \quad \mathrm{Var}(p_t|p_0) = \sigma^2 t$$

所以 $\{p_t\}$ 非平稳。从(7.2)，每个 ε_{t-j} 对 p_t 的影响的权重都等于 1，而对于 AR(1) 模型，ε_{t-j} 对 p_t 的影响的权重等于 $|\phi_1|^j$，随距离 j 的增大按负指数速度衰减。称(7.2)中的 ε_{t-j} 对 p_t 是永久影响的。

随机游动的水平不可预测，多步预测没有均值反转性质。易见

$$\hat{p}_h(1) = E(p_h + \varepsilon_{h+1}|\mathscr{F}_h) = p_h$$

其中 $\mathscr{F}_t$ 代表截止到时刻 t 位置的所有观测信息，定义是包含 $p_t, p_{t-1}, ...$ 的 σ 代数。

又

$$\hat{p}_h(2) = E(p_h + \varepsilon_{h+1} + \varepsilon_{h+2}|\mathscr{F}_h) = p_h$$

可见 $\hat{p}_h(k) = p_h$, $k = 1, 2, ...$。所以随机游动只能用最后一个观测的水平不经衰减地预测，没有均值反转。而 AR 的预测则是均值与最近一个或几个观测的加权平均，超前多步预测有均值反转。

$\hat{p}_h(k)$ 的预测均方误差为

$$E(e_h(k))^2 = E(p_{h+k} - \hat{p}_h(k))^2 = E(\sum_{j=1}^{k} \varepsilon_{h+j})^2 = k\sigma^2 \to \infty \ (k \to \infty)$$

这也说明了此模型不可预测。

从(7.2)也有 $\mathrm{Var}(p_t|p_0) \to \infty (t \to \infty)$。这意味着对数价格可以取到接近正负无穷值，对于个股有合理性，但是对于综合股指取负无穷则不太合理。

单位根过程的 ACF 估计是不相合的，对单位根过程的样本作 ACF 图，其衰减速度很慢很慢。

设 $p_0 = 0$，单位根过程 $\{p_t\}$ 有如下特点：

- p_t 期望值等于 0；
- p_t 方差等于 $\sigma^2 t$，随 t 线性增长，趋于无穷；
- 历史的扰动（新息）的影响不衰减；
- 预测只能用最后一个观测值作为预测，预测均方误差趋于无穷。
- 样本 ACF 表现为基本不衰减，近似等于 1。

7.2 带漂移的随机游动

上面的随机游动模型的金融意义一般 p_t 是对数价格，则 ε_t 是零均值的对数收益率。实际的对数收益率常常是非零的，正数居多。所以，模型可以推广为

$$p_t = \mu + p_{t-1} + \varepsilon_t, \ t = 1, 2, ...$$

其中 $\{\varepsilon_t\}$ 仍为零均值独立同分布白噪声列。常数 μ 并不代表均值，而是对数价格 p_t 的增长速度，称为模型的**漂移**(drift)。设初始价格为 p_0，则

$$\begin{aligned} p_1 =& p_0 + \mu + \varepsilon_1 \\ p_2 =& p_0 + 2\mu + \varepsilon_1 + \varepsilon_2 \\ & \cdots\cdots \\ p_t =& p_0 + t\mu + \varepsilon_1 + \cdots + \varepsilon_t \end{aligned}$$

于是

$$E(p_t|p_0) = p_0 + \mu t, \quad \mathrm{Var}(p_t|p_0) = \sigma^2 t$$

所以带漂移的随机游动与不带漂移的随机游动相比，其条件方差不变，但是条件均值多了一个随 t 线性增长（若 $\mu > 0$）的 μt 项。

这时，实际的序列的图形将沿着 $y = p_0 + \mu t$ 这条直线附近变化。如果 $\mu = 0$，则图形也有缓慢的水平变化但是没有这样的固定趋势。

带漂移的随机游动 p_t，可以分解为两部分：

$$p_t = (p_0 + \mu t) + p_t^*$$

其中 $p_t^* = \sum_{j=1}^{t} \varepsilon_t$ 是从 0 出发的不带漂移的随机游动，$p_0 + \mu t$ 是一个非随机的线性趋势。

例 7.1. 考虑例6.1中 3M 公司股票从 1946 年 2 月到 2008 年 12 月的月对数收益率，共有 755 个观测。从中恢复对数价格。

```
d <- read_table(
  "m-3m4608.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
mmm <- xts(log(1 + d[["rtn"]]), d$date)
rm(d)
tclass(mmm) <- "yearmon"
ts.3m <- ts(coredata(mmm), start=c(1946,2), frequency=12)
ts.3mlogp <- ts(cumsum(c(c(ts.3m))), start=c(1946,2), frequency=12)
c(mean=mean(ts.3m), sd=sd(ts.3m))
```

```
##       mean         sd
## 0.01029941 0.06371910
```

对数收益率的白噪声检验:

```
Box.test(ts.3m, type="Ljung", lag=12)
```

```
##
##  Box-Ljung test
##
## data:  ts.3m
## X-squared = 27.688, df = 12, p-value = 0.006143
```

白噪声检验不能通过。暂时忽略这个问题。

上面恢复的对数价格假设开始日期前一月，即 1946 年 1 月的对数价格为 0。设对数收益率为均值非零的白噪声列，则 p_t 模型为

$$p_t = p_{t-1} + 0.01030 + \varepsilon_t,$$

其中 $\sigma_\varepsilon = 0.06372$。分解成两部分即

$$p_t = 0.01030t + p_t^*,$$

$\{p_t^*\}$ 是随机游动，扰动（新息）的标准差为 0.06372。

$\{p_t\}$ 的 ACF 图形如下：

```
forecast::Acf(ts.3mlogp, main="", lag=36)
```

下面做出 $p_t, p_t^*, 0.0130t$ 这三个序列的图形：

```
plot(c(time(ts.3mlogp)), c(ts.3mlogp), ylim=c(-1, 8.2), type="l",
     xlab="Year", ylab="ln(Price)")
tmp.x <- seq(length(ts.3mlogp))
```

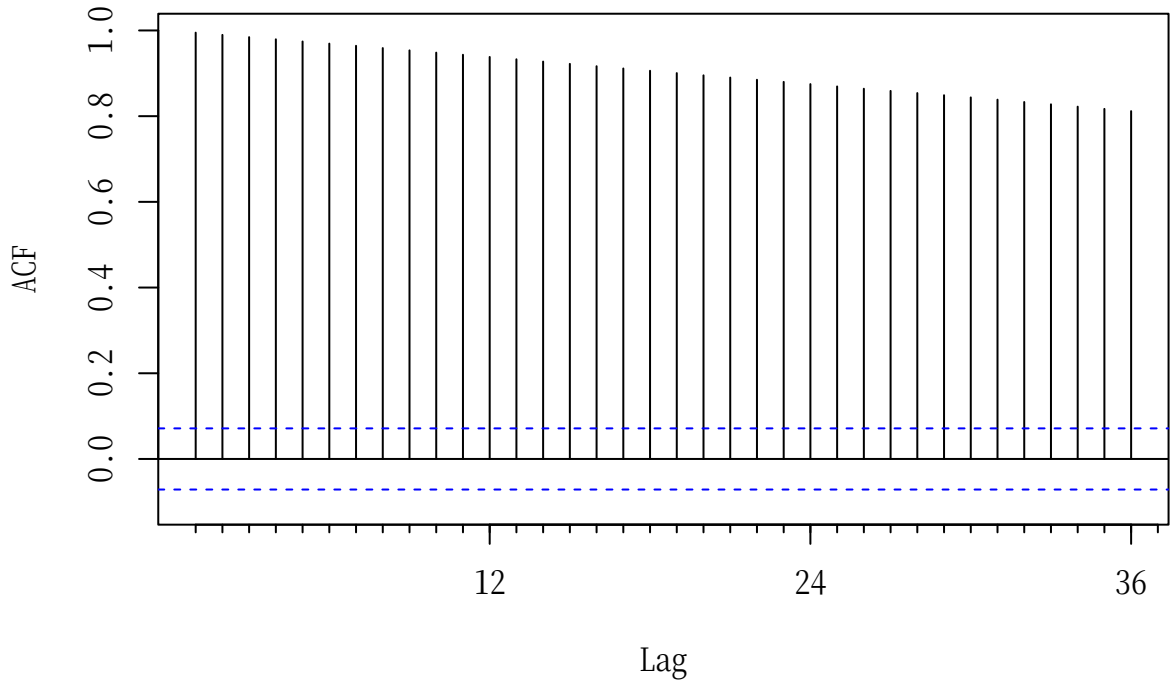


图 7.1: 3M 股票对数价格的 ACF

```
tmp.y <- 0.01030*tmp.x
tmp.y2 <- c(ts.3mlogp) - tmp.y
tmp.y <- ts(tmp.y, start=c(1946,2), frequency=12)
lines(c(time(tmp.y)), c(tmp.y), col="red", lwd=2, lty=3)
lines(c(time(tmp.y)), c(tmp.y2), col="green")
legend("topleft", lty=c(1,3,1), col=c("black", "red", "green"),
       legend=c("log(P)", "Linear Trend", "Random Walk"))
abline(h=0, col="gray", lty=3)
```

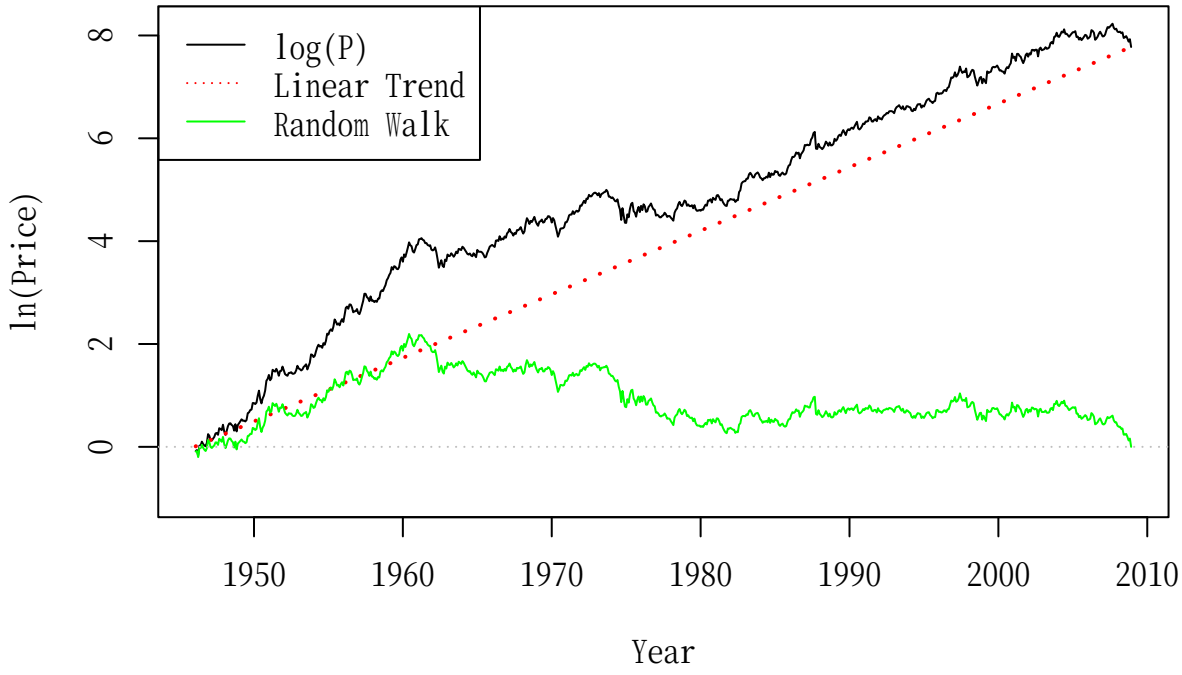


图 7.2: 3M 股票对数价格带漂移随机游动

```
rm(tmp.x, tmp.y, tmp.y2)
```

7.3 固定趋势模型

设 $\{X_t\}$ 为弱平稳时间序列令

$$Y_t = a + bt + X_t$$

则 $EY_t = (a + \mu_x) + bt$，$\mathrm{Var}(Y_t) = \mathrm{Var}(X_t) = \sigma_x^2$，均值非常数所以 $\{Y_t\}$ 非平稳。但是，减去一个固定趋势 $a + bt$ 后 $\{Y_t\}$ 就变成了平稳列，这样的 $\{Y_t\}$ 与随机游动或者带漂移的随机游动有着本质的区别。

随机游动 $p_t = p_{t+1} + \varepsilon_t$ 与固定趋势加扰动 $Y_t = a + bt + X_t$ 都能呈现出缓慢的趋势变化。区别在于：

- 随机游动的方差是线性增长的，固定趋势的观测值方差不变；
- 随机游动的扰动的影响是永久的，固定趋势的扰动的影响仅在一个时刻（如果扰动 X_t 是白噪声）或者很短时间（如果是扰动 X_t 是线性时间序列）；
- 随机游动的趋势没有固定方向，固定趋势的变化形状是固定的；
- 固定趋势模型 Y_t 减去一个固定的回归函数 $Y = a + bt$ 就可以变成平稳列，随机游动减去任意的非随机函数都不能变平稳，可以用差分运算变成平稳。

在 AR 和 ARMA 模型中，常数项 ϕ_0 与平稳均值有关。但是在带漂移的随机游动模型中，常数项 μ 是每一步的平均增量，是固定线性趋势的斜率。所以时间序列模型中的常数项可能会依模型的不同而具有迥然不同的含义。

7.4 ARIMA 模型

将带漂移的随机游动模型中的白噪声替换成一个 ARMA 平稳列，其主要的性质仍能保留。即

$$Y_t = Y_{t-1} + \mu + X_t$$

其中 $\{X_t\}$ 是零均值平稳可逆 ARMA(p,q) 平稳列。这时有

$$Y_t = Y_0 + \mu t + \sum_{j=1}^{t} X_j$$

于是

$$E(Y_t|Y_0) = Y_0 + \mu t, \quad \mathrm{Var}(Y_t|Y_0) = \text{随}t\text{增大而趋于}\infty$$

当 $\mu = 0$ 时，$\{Y_t\}$ 的均值固定。Y_t 的条件方差为 t 的线性函数。X_{t-j} 对 Y_t 的影响不衰减，是永久有影响的。这些表现与带漂移的随机游动基本相同。

称 $\{Y_t\}$ 服从 ARIMA$(p,1,q)$ 模型，是非平稳的。$\{Y_t\}$ 不能通过减去任何的非随机趋势变成平稳。但是，差分运算

$$\Delta Y_t = (1-B)Y_t = Y_t - Y_{t-1} = \mu + X_t$$

将 ARIMA$(p,1,q)$ 序列 $\{Y_t\}$ 转化成平稳可逆的 ARMA(p,q) 序列。

设零均值平稳可逆 ARMA(p,q) 序列 $\{X_t\}$ 的模型为

$$P(B)X_t = Q(B)\varepsilon_t$$

其中 $\{\varepsilon_t\}$ 为零均值独立同分布白噪声列，$P(z) = 1-\phi_1 z-\cdots-\phi_p z^p$, $Q(z) = 1+\theta_1 z+\cdots+\theta_q z^q$。因为 $(1-B)Y_t = \mu + X_t$，所以

$$P(B)(1-B)Y_t = P(B)(\mu + X_t) = P(1)\mu + P(B)X_t = P(1)\mu + Q(B)\varepsilon_t$$

记 $\tilde{P}(z) = P(z)(1-z)$, $\phi_0 = P(1)\mu$，则 Y_t 的模型可以写成

$$\tilde{P}(B)Y_t = \phi_0 + Q(B)\varepsilon_t$$

这个模型形式上与一个 ARMA$(p+1,q)$ 模型相同，但是其 $p+1$ 个特征根中有一个等于 1，其余特征根才在单位圆外。

对 ARIMA 模型建模，只要计算 Y_t 的差分，然后对差分建立 ARMA 模型即可。

有些序列需要二阶差分才能平稳，二阶差分即

$$\begin{aligned}\Delta^2 Y_t =& (1-B)^2 Y_t = (1-B)(Y_t - Y_{t-1}) = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) \\ =& Y_t - 2Y_{t-1} + Y_{t-2}\end{aligned}$$

如果 $\xi_t = a + bt$，则

$$(1 - B)\xi_t = b$$

即差分还可以消除线性趋势。

如果 $\xi_t = a + bt + ct^2$，则

$$(1 - B)^2\xi_t = 2c$$

即二阶差分可以消除二次多项式趋势。

在 R 中用 `diff(x)` 计算一阶差分，结果从原来的第二个时间点给出；用 `diff(x, lag=2)` 计算二阶差分，结果从原来的第三个时间点给出。

如果 Y_t 本身已经是弱平稳列，则不应对 Y_t 进行差分。如果 Y_t 是非随机的线性趋势加平稳列，虽然差分能将其变成平稳列，但是也不应该使用差分来做而是应该用回归来做，用差分来做会在 ARMA 模型的 MA 部分引入不必要的单位根。

7.5 单位根检验

单位根非平稳列是金融中最常用的非平稳模型，单位根非平稳列不能使用平稳列的模型来建模。所以，要建模的序列应该进行 “单位根检验”。

对不带漂移的单位根过程，考虑如下的基础模型：

$$p_t = \phi_1 p_{t-1} + \varepsilon_t \tag{7.3}$$

其中 $\{\varepsilon_t\}$ 是零均值独立同分布白噪声列。$|\phi_1| \leq 1$。考虑如下零假设与对立假设：

$$H_0 : \phi_1 = 1 \longleftrightarrow H_a : \phi_1 < 1$$

这样的检验问题称为单位根检验问题。基础模型也可以是带有常数项的：

$$p_t = \phi_0 + \phi_1 p_{t-1} + \varepsilon_t \tag{7.4}$$

对模型(7.3)作最小二乘估计

$$\hat{\phi}_1 = \frac{\sum_{t=1}^{T} p_{t-1}p_t}{\sum_{t=1}^{T} p_t^2}, \quad \hat{\sigma}^2 = \frac{1}{T-1}\sum_{t=1}^{T}(p_t - \hat{\phi}_1 p_{t-1})^2$$

其中 $p_0 = 0$, T 为样本量。取检验统计量

$$\mathrm{DF} = \frac{\hat{\phi}_1 - 1}{\mathrm{SE}(\hat{\phi}_1)} = \frac{\sum_{t=1}^{T} p_{t-1}e_t}{\hat{\sigma}\sqrt{\sum_{t=1}^{T} p_{t-1}^2}}$$

当 T 充分大时在 H_0 下有渐近分布，当 DF 统计量足够小的时候拒绝 H_0。p 值一般通过随机模拟计算。这个检验称为 Dicky-Fuller 检验。

在使用(7.4)作为基础模型时，如果实际上 $\phi_0 = 0$，则 DF 统计量也有非标准的渐近分布，可以用随机模拟方法计算 p 值。如果实际上 $\phi_0 \neq 0$，则 DF 统计量渐近正态分布，但是需要很大的样本量。

许多经济和金融序列并不能仅用随机游动来描述，可能需要用 ARIMA。因为 ARMA 模型可以看成长阶自回归，所以检验是否 ARIMA 模型，可以用 $q=0$ 的 ARIMA 作为基础模型。对序列 $\{x_t\}$ 为了检验其是否有单位根，考虑如下的基础模型：

$$X_t = c_t + \beta X_{t-1} + \sum_{j=1}^{p-1} \phi_j \Delta X_{t-j} + e_t \tag{7.5}$$

当 $\beta=1$ 时，就是 ΔX_t 的 AR$(p-1)$ 模型；当 $\beta<1$ 时，是 X_t 的 AR(p) 模型。c_t 是非随机的趋势部分，可以取 0，或常数，或 $a+bt$ 这样的非随机线性趋势。检验假设

$$H_0 : \beta = 1 \longleftrightarrow H_a : \beta < 1$$

如果拒绝 H_0，就说明没有单位根。使用统计量

$$\text{ADF} = \frac{\hat{\beta}-1}{\text{SE}(\hat{\beta})}$$

当 ADF 统计量足够小的时候拒绝 H_0。

基础模型(7.5)也可以改写成

$$\Delta X_t = c_t + \beta_c X_{t-1} + \sum_{j=1}^{k} \phi_j \Delta X_{t-j} + e_t, \tag{7.6}$$

其中 $\beta_c = \beta - 1$，$k = p-1$，检验

$$H_0 : \beta_c = 0 \longleftrightarrow H_a : \beta_c < 0$$

这个检验称为 ADF 检验 (Augmented Dicky-Fuller Test)。

fUnitRoots 包的 `adfTest()` 函数可以执行单位根 ADF 检验。其中的 `lags` 参数指定(7.6)中的差分后 AR 阶数 k。`type` 参数对应 ((7.6)) 中的 c_t，选项为：

- `nc`: $c_t = 0$;
- `c`: $c_t = c$;
- `ct`: $c_t = a + bt$。

urca 包的 `ur.df()` 函数，tseries 包的 `adf.test()` 函数也可以执行单位根 ADF 检验。

注意，单位根 DF 检验和 ADF 检验都是在拒绝 H_0（显著）时否认有单位根 (结论是平稳)，不显著时承认有单位根 (结论是非平稳)。

例 7.2. 考虑美国的国民生产总值 (GNP) 经过季节调整后的对数值。时间是 1947 年第一季度到 2010 年第一季度，总计 253 个观测值。

读入数据并计算季节调整后的 GNP 的对数值，以及对数值的一阶差分：

```
da <- read_table("q-gnp4710.txt", col_types=cols(
  .default = col_double()))
gnp <- ts(log(da[["VALUE"]]),
```

```
          start=c(1947, 1), frequency=4)
dgnp <- diff(gnp)
rm(da)
```

GNP 的对数值的图形：

```
plot(gnp, xlab="year", ylab="log(GNP)")
```

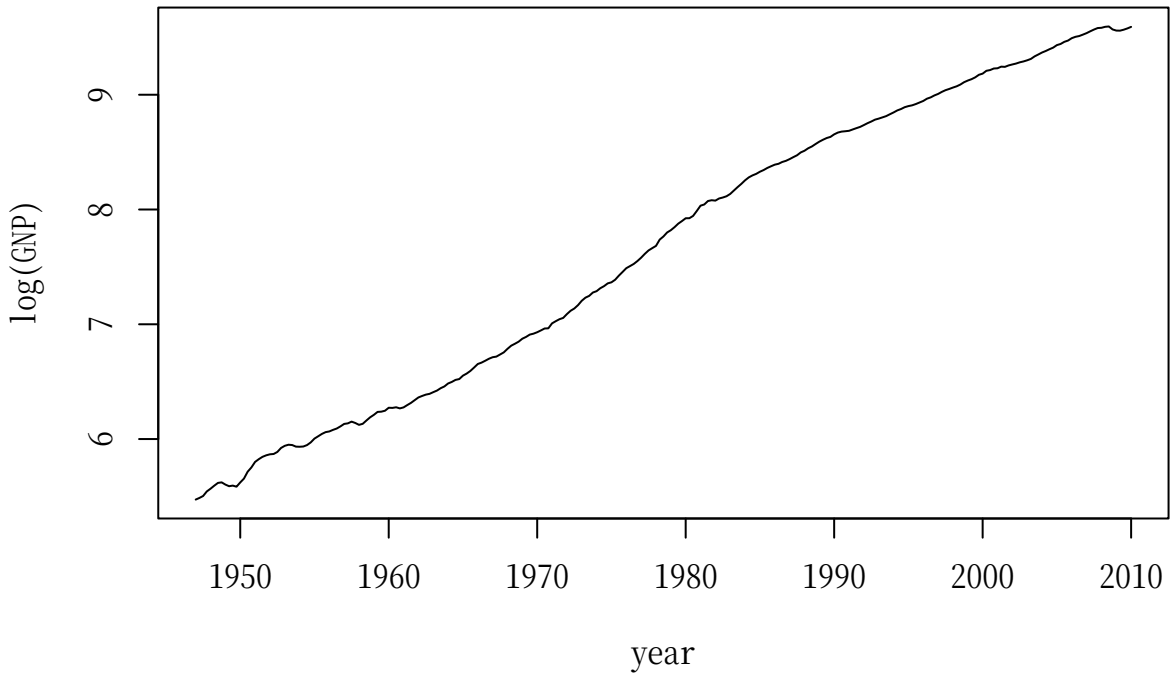


图 7.3: 美国经季节调整的 GNP 对数值

对数 GNP 明显地不平稳。其 ACF 图形如下：

```
forecast::Acf(gnp, main="")
```

对数 GNP 差分的时间序列图：

```
plot(dgnp, xlab="year", ylab="log(GNP)")
abline(h=0, col="gray", lty=3)
```

对数 GNP 差分的 ACF:

```
forecast::Acf(dgnp, main="")
```

对数 GNP 差分的 PACF:

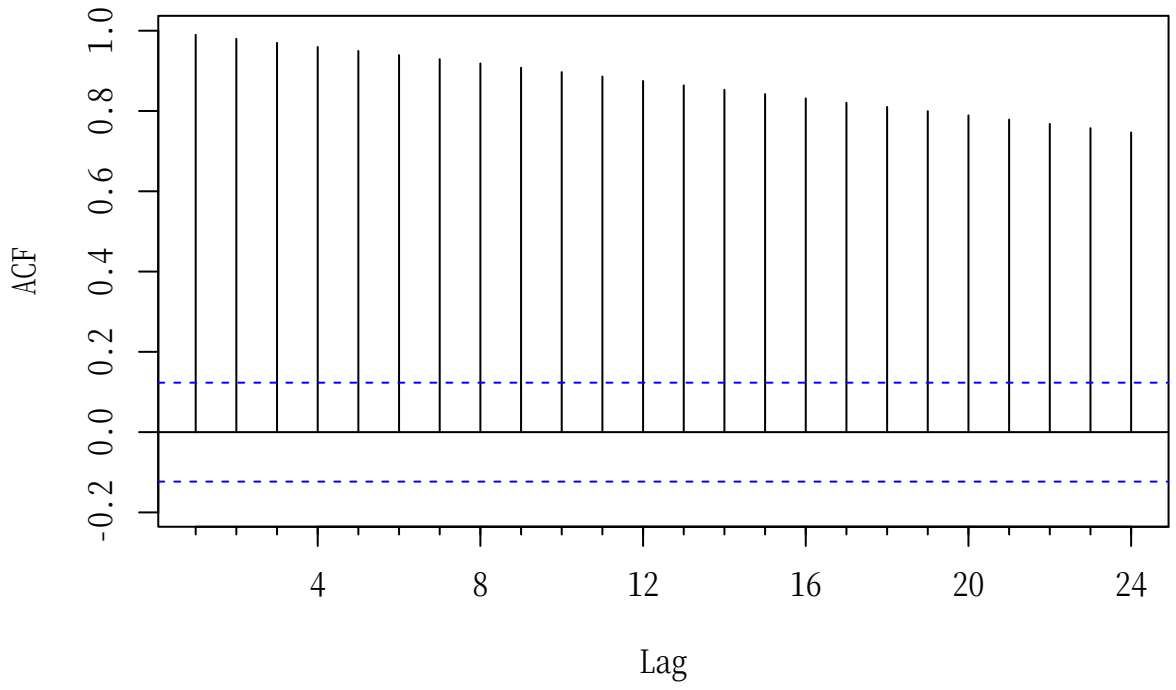


图 7.4: 美国经季节调整的 GNP 对数值的 ACF

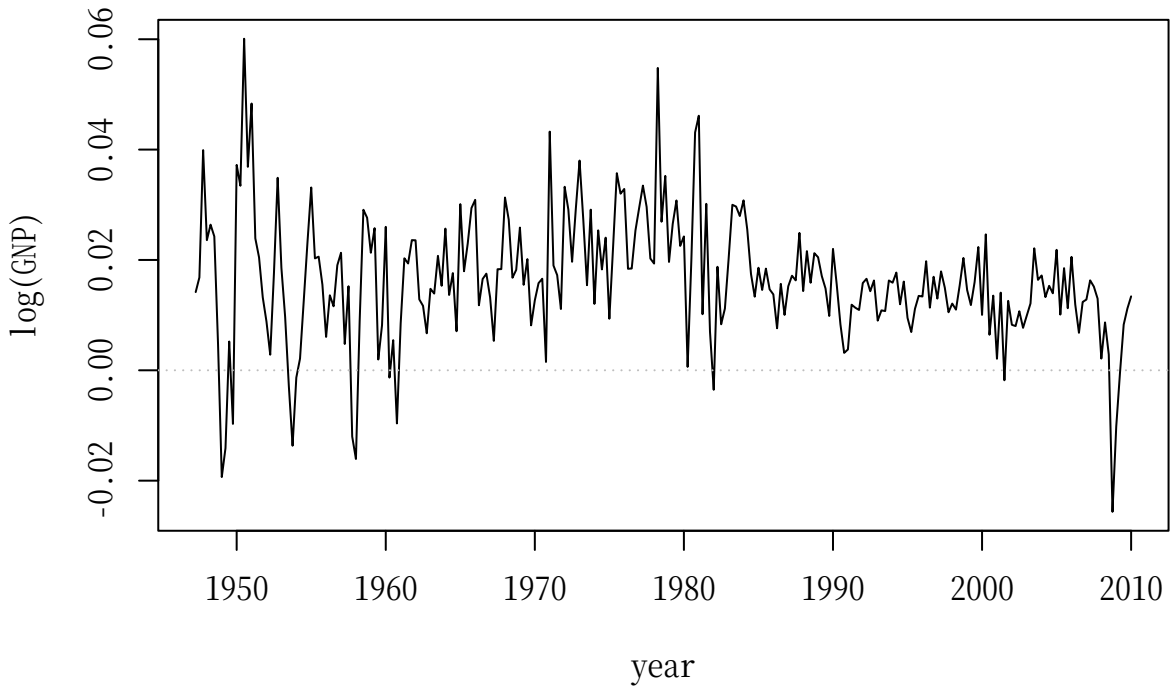


图 7.5: 美国经季节调整的对数 GNP 差分

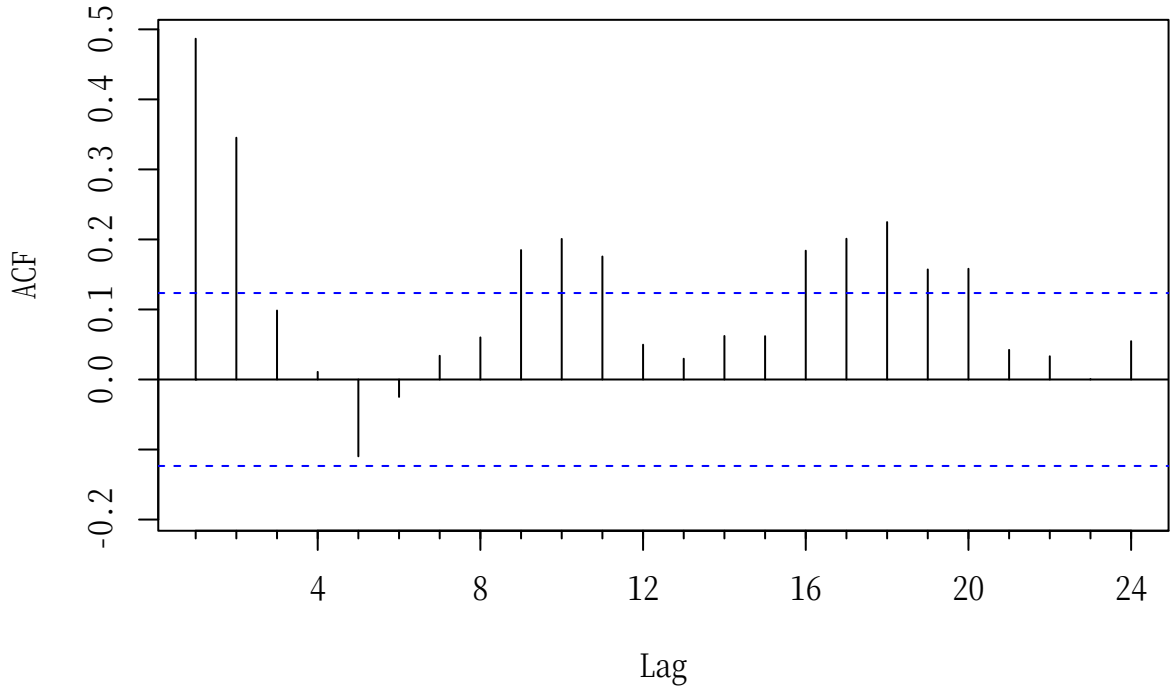


图 7.6: 美国经季节调整的对数 GNP 差分的 ACF

```
forecast::Pacf(dgnp, main="")
```

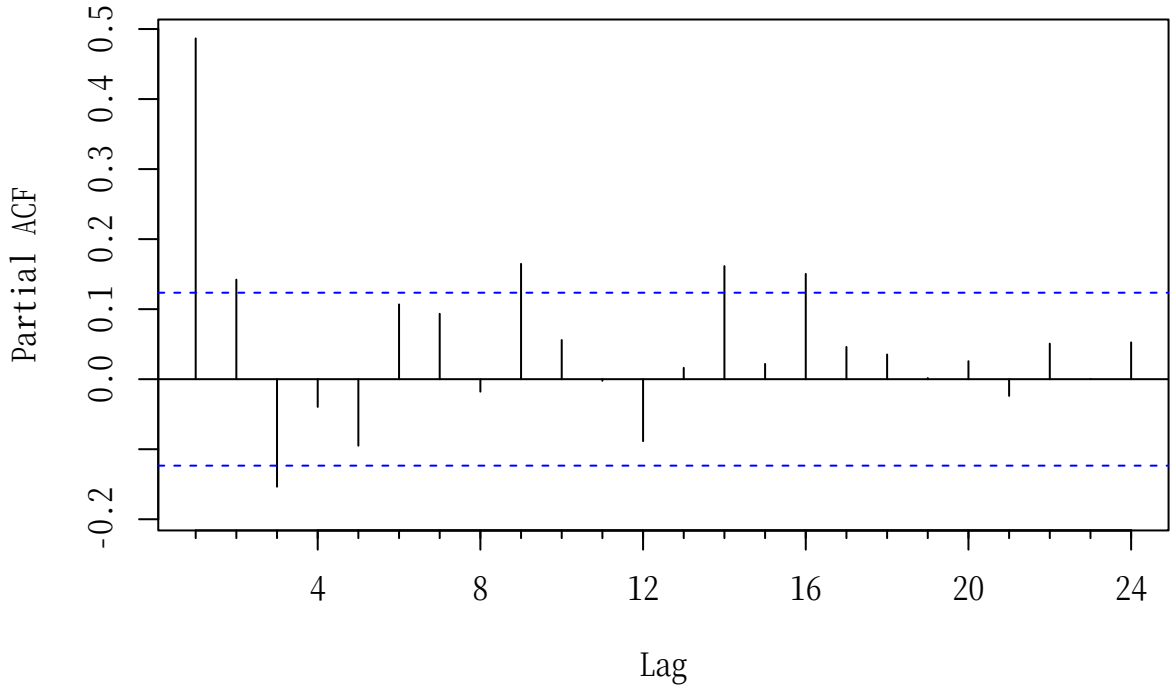


图 7.7: 美国经季节调整的对数 GNP 差分的 PACF

如果用 AR(p) 为对数 GNP 差分序列建模，PACF 图提示用 $p = 16$。用 AIC 定阶：

```
ar(dgnp, method="mle")
```

```
##
## Call:
## ar(x = dgnp, method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.4318   0.1985  -0.1180   0.0189  -0.1607   0.0900   0.0615  -0.0814
##       9
##  0.1940
##
## Order selected 9  sigma^2 estimated as  8.918e-05
```

差分序列用 AIC 取 AR 为 9 阶。

ADF 的基础模型需要一个 AR 阶数，这是差分序列的 AR 阶数，取 $p = 9$。用 `fUnitRoots::adfTest()` 对 GNP 的对数值进行 ADF 单位根检验：

```
fUnitRoots::adfTest(gnp, lags=9, type="c")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 9
##   STATISTIC:
##     Dickey-Fuller: -1.8467
##   P VALUE:
##     0.3691
##
## Description:
##  Wed Mar 23 16:19:04 2022 by user: Lenovo
```

结果 p 值较大，说明不能拒绝零假设，即对数 GNP 序列有单位根。

GNP 对数序列的图形也像是有非随机线性增长趋势的情况。为此，仍使用 ADF 检验，但是允许有非随机常数项和线性项：

```
fUnitRoots::adfTest(gnp, lags=9, type="ct")
```

```
## Warning in fUnitRoots::adfTest(gnp, lags = 9, type = "ct"): p-value greater than
## printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 9
##   STATISTIC:
##     Dickey-Fuller: -0.0094
##   P VALUE:
##     0.99
##
## Description:
##  Wed Mar 23 16:19:04 2022 by user: Lenovo
```

结果仍是承认零假设，认为有单位根存在。

尝试人为地拟合非随机线性增长趋势，检验残差是否有单位根：

```
tmp.t <- c(time(gnp))
tmp.y <- residuals( lm(c(gnp) ~ tmp.t) )
```

```
fUnitRoots::adfTest(tmp.y, type="nc")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -0.0763
##   P VALUE:
##     0.592
##
## Description:
##  Wed Mar 23 16:19:04 2022 by user: Lenovo
```

结果说明用回归去掉非随机的线性增长趋势后仍有单位根存在。

使用 urca 包的 `ur.df()` 执行单位根 ADF 检验，基础模型带有常数漂移项：

```
library(urca)
ur.df(gnp, type="drift", lags=9) |>
  summary()
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.033589 -0.004595  0.000503  0.003911  0.035557
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0135157  0.0045219   2.989  0.00310 **
## z.lag.1     -0.0009040  0.0004895  -1.847  0.06606 .
## z.diff.lag1  0.3903745  0.0641690   6.084 4.81e-09 ***
```

```
## z.diff.lag2  0.2057039  0.0681595   3.018  0.00283 **
## z.diff.lag3 -0.1111342  0.0693754  -1.602  0.11053
## z.diff.lag4  0.0263875  0.0692327   0.381  0.70345
## z.diff.lag5 -0.1399063  0.0684739  -2.043  0.04216 *
## z.diff.lag6  0.0857675  0.0704276   1.218  0.22453
## z.diff.lag7  0.0520310  0.0696968   0.747  0.45610
## z.diff.lag8 -0.0879122  0.0685969  -1.282  0.20127
## z.diff.lag9  0.1816295  0.0639332   2.841  0.00490 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.009396 on 232 degrees of freedom
## Multiple R-squared:  0.3142, Adjusted R-squared:  0.2847
## F-statistic: 10.63 on 10 and 232 DF,  p-value: 8.56e-15
##
##
## Value of test-statistic is: -1.8467 6.7291
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.44 -2.87 -2.57
## phi1  6.47  4.61  3.79
```

可以将统计量的第一个 (-1.8467) 与 tau2 统计量的 5pct 临界值 (-2.87) 比较，没有小于临界值，所以可以接受零假设，认为有单位根。这个检验是左侧否定域。

`urca::ur.df()` 的 `type` 参数也对应于 ((7.6)) 中的 c_t，选项为：

- `"none"`: $c_t = 0$;
- `"drift"`: $c_t = c$;
- `"trend"`: $c_t = a + bt$。

使用 tseries 包的 `adf.test()` 执行单位根 ADF 检验，对应于 ((7.6)) 中的 $c_t = a + bt$。参数 k 为 ((7.6)) 中的 k。

```
tseries::adf.test(gnp, k=9)
```

```
## Warning in tseries::adf.test(gnp, k = 9): p-value greater than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  gnp
## Dickey-Fuller = -0.0093764, Lag order = 9, p-value = 0.99
## alternative hypothesis: stationary
```

forecast 包提供了一个 `ndiffs()` 函数，可以用检验的方法估计需要多少次差分变成平稳，如：

```
forecast::ndiffs(gnp, test="adf")
```

```
## [1] 1
```

即需要进行一阶差分，然后平稳。

例 7.3. 考虑标普 500 指数从 1950-01-03 到 2008-04-11 的 OHLC 数据日数据。共 14662 个观测。

读入数据，从中计算日收盘价对数值序列及其一阶差分：

```
da <- read_table("d-sp55008.txt", col_types=cols(.default = col_double()))
xts.sp5d <- xts(da[,-(1:3)], make_date(da$year, da$mon, da$day))
str(xts.sp5d)
```

```
## An 'xts' object on 1950-01-03/2008-04-11 containing:
##   Data: num [1:14662, 1:6] 16.7 16.9 16.9 17 17.1 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:6] "open" "high" "low" "close" ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```
sp5d <- log(xts.sp5d[,"close"])
delta.sp5d <- diff(coredata(sp5d)[,1])
```

作日对数收盘价的时间序列图：

```
plot(sp5d, type="l", main="S&P 500 Daily Log Close",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="auto")
```

明显非平稳。一阶差分的时间序列图：

```
plot(diff(sp5d), type="l", main="S&P 500 Daily Log Return",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="auto")
```

一阶差分的 PACF:

```
pacf(delta.sp5d, main="")
```

用 AIC 对日对数收益率定阶：

```
ar(delta.sp5d, method="mle")
```

```
##
## Call:
## ar(x = delta.sp5d, method = "mle")
```

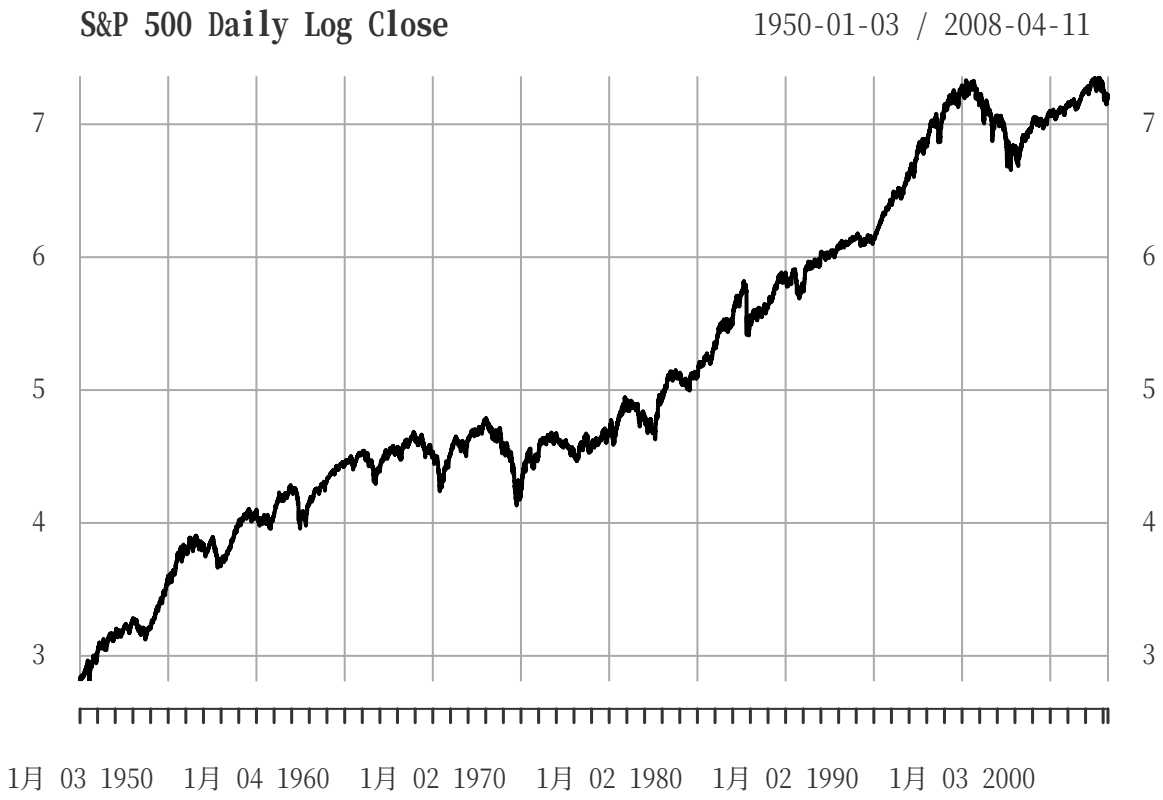


图 7.8: S&P 500 日对数收盘价

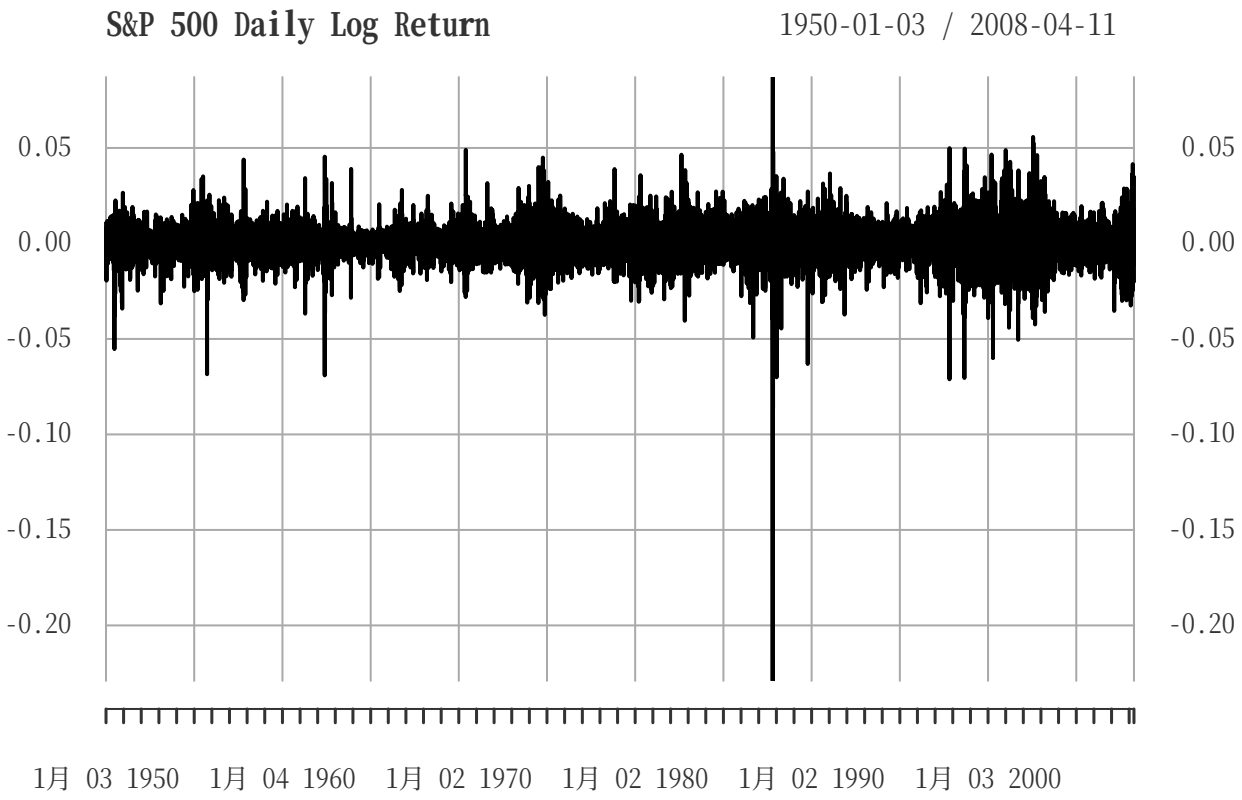


图 7.9: S&P 500 日对数收益率

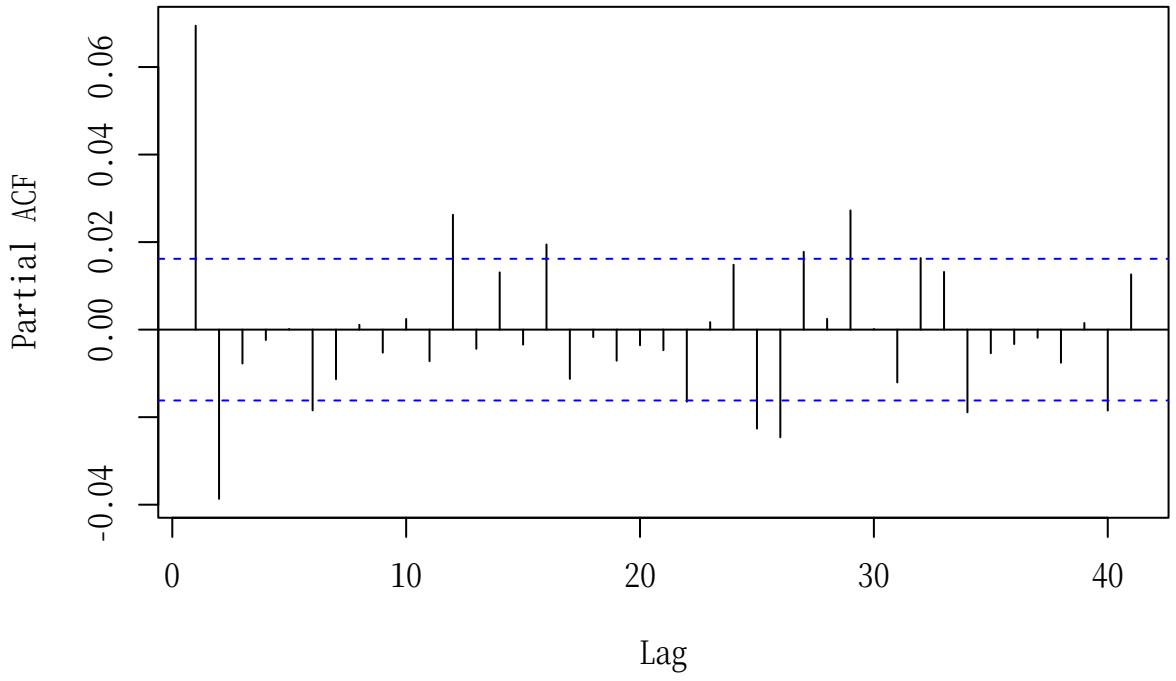


图 7.10: S&P 500 日对数收益率的 PACF

```
##
## Coefficients:
##       1        2
##  0.0721  -0.0387
##
## Order selected 2  sigma^2 estimated as  8.068e-05
```

AIC 定阶 $p = 2$。按 $p = 2$ 对标普 500 日对数收盘价作 ADF 白噪声检验：

```
fUnitRoots::adfTest(c(coredata(sp5d)), lags=2, type="ct")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 2
##   STATISTIC:
##     Dickey-Fuller: -2.0179
##   P VALUE:
##     0.5708
##
## Description:
##  Wed Mar 23 16:19:05 2022 by user: Lenovo
```

检验不显著，说明存在单位根。如果对日对数收益率（即对数收盘价序列的差分）进行 ADF 检验，则显著，说明一阶差分后使得序列变得平稳：

```
fUnitRoots::adfTest(delta.sp5d, lags=2, type="ct")
```

```
## Warning in fUnitRoots::adfTest(delta.sp5d, lags = 2, type = "ct"): p-value
## smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 2
##   STATISTIC:
##     Dickey-Fuller: -70.5501
##   P VALUE:
##     0.01
##
## Description:
```

```
##  Wed Mar 23 16:19:05 2022 by user: Lenovo
```

Chapter 8

指数平滑

8.1 简单指数平滑

指数平滑最早是来自一种简单的预测方法：用历史数据的线性组合预测下一时间点的值，线性组合系数随距离变远而按负指数（几何级数）衰减：

$$\hat{x}_h(1) \approx wx_h + w^2x_{h-1} + \cdots = \sum_{j=1}^{\infty} w^j x_{h+1-j}$$

其中 $0 < w < 1$，w 越小，距离远的历史观测对预测的贡献越小。

因为是加权平均，所以所有加权的和应该等于零，注意到

$$\sum_{j=1}^{\infty} w^j = \frac{w}{1-w}$$

所以第 j 个权重应为

$$\frac{w^j}{\frac{w}{1-w}} = (1-w)w^{j-1},\ j = 1, 2, ...$$

于是

$$\hat{x}_h(1) = (1-w)(x_h + wx_{h-1} + w^2x_{h-2} + ...) = (1-w)\sum_{j=0}^{\infty} w^j x_{h-j}$$

这种预测方法叫做**指数平滑方法** (exponential smoothing method)，在早期应用中权重 w 是凭经验选取的。

经研究，ARIMA(0,1,1) 的 $\hat{x}_h(1)$ 的公式恰好具有如上形式。对如下的 ARIMA(0,1,1) 模型：

$$(1-B)X_t = (1-\theta B)\varepsilon_t$$

两边除以 $1-\theta B$，由于

$$\frac{1}{1-\theta z} = \sum_{j=0}^{\infty} \theta^j z^j$$

所以模型可以化为

$$(1-B)\sum_{j=0}^{\infty}\theta^j X_{t-j}=\varepsilon_t$$

$$\sum_{j=0}^{\infty}\theta^j X_{t-j}-\sum_{j=0}^{\infty}\theta^j X_{t-j-1}=\varepsilon_t$$

$$X_t+\sum_{j=1}^{\infty}\theta^j X_{t-j}-\sum_{j=1}^{\infty}\theta^{j-1}X_{t-j}=\varepsilon_t$$

$$X_t-(1-\theta)\sum_{j=1}^{\infty}\theta^{j-1}X_{t-j}=\varepsilon_t$$

$$X_t-(1-\theta)\sum_{j=0}^{\infty}\theta^{j}X_{t-1-j}=\varepsilon_t$$

以 $t=h+1$ 代入得

$$X_{h+1}=(1-\theta)\sum_{j=0}^{\infty}\theta^j X_{h-j}+\varepsilon_{h+1}$$

于是

$$\hat{x}_h(1)=E(X_{h+1}|\mathscr{F}_h)=(1-\theta)\sum_{j=0}^{\infty}\theta^j X_{h-j}$$

这就是 $w=\theta$ 的指数平滑预测公式。

因为指数平滑预测与 ARIMA(0,1,1) 预测的等价性，可以用 ARIMA 建模方法估计权重 w。注意 `arima()` 函数估计的 MA 系数是 $1+\theta_1 z+\cdots+\theta_q z^q$ 形式的系数。也可以用 ARIMA 模型的充分性检验来判断指数平滑预测是否有意义。

例 8.1. 考虑 CBOE 的波动率指数（VIX）2004-01-02 到 2011-11-21 的日收盘价的对数序列，用指数平滑方法作一步预测。共 1988 个观测。

读入数据，从中计算日收盘价对数值序列：

```
da <- read_table("d-vix0411.txt", col_types=cols(.default = col_double()))
xts.vix <- xts(da[,-(1:3)], make_date(da$year, da$mon, da$day))
str(xts.vix)
```

```
## An 'xts' object on 2004-01-02/2011-11-21 containing:
##   Data: num [1:1988, 1:4] 18 18.4 17.7 16.7 15.4 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "Open" "High" "Low" "Close"
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```
vix <- log(xts.vix[,"Close"])
delta.vix <- diff(vix)[-1]
```

作日对数收盘价的时间序列图：

```
plot(vix, type="l", main="VIX Daily Log Close",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="auto")
```

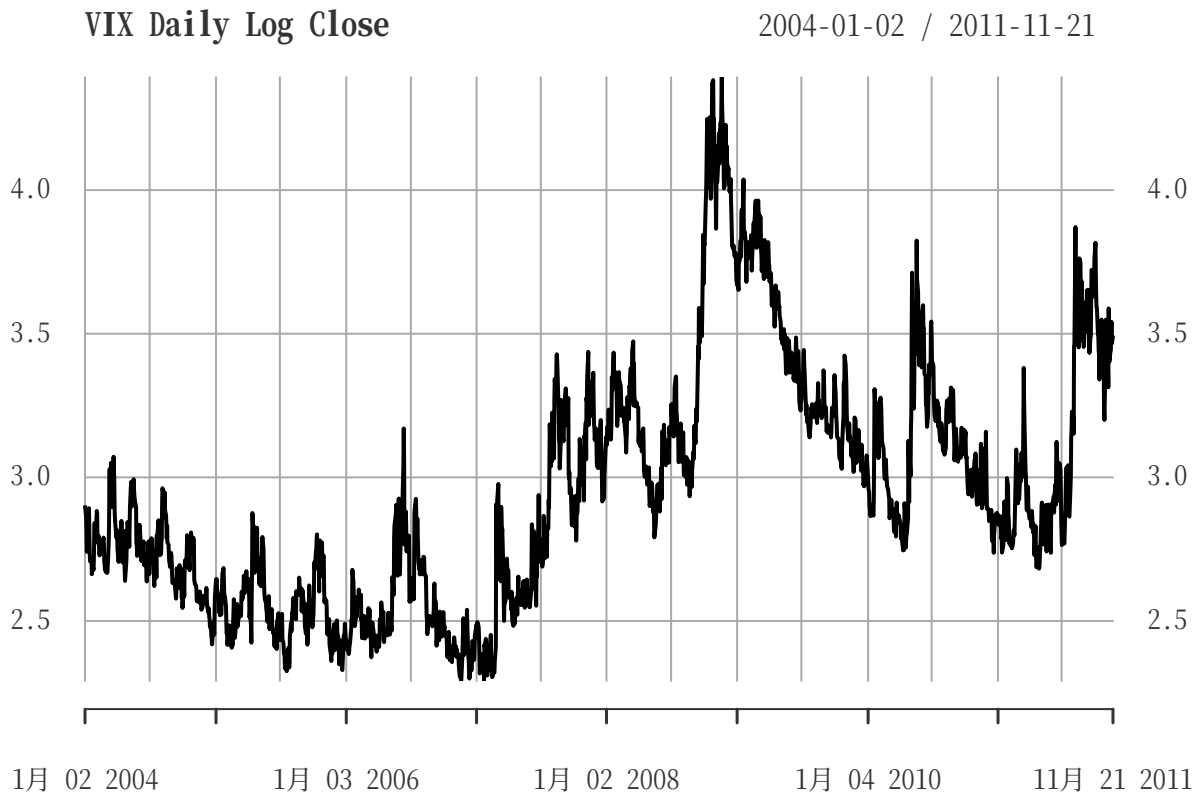


图 8.1: VIX 日对数收盘价

明显不平稳且没有固定趋势。

作日对数收益率的时间序列图：

```
plot(delta.vix, type="l", main="VIX Daily Log Return",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="auto")
```

日对数收益率的 ACF:

```
forecast::Acf(c(coredata(delta.vix)), main="")
```

虽然在滞后 10 的位置 ACF 也超界，但是在低阶就只有滞后 1 明显超界，所以对数收盘价用 ARIMA(0,1,1) 还是合理的。ARIMA 建模：

```
resm <- arima(c(coredata(vix)), order=c(0,1,1)); resm
```

```
##
## Call:
```

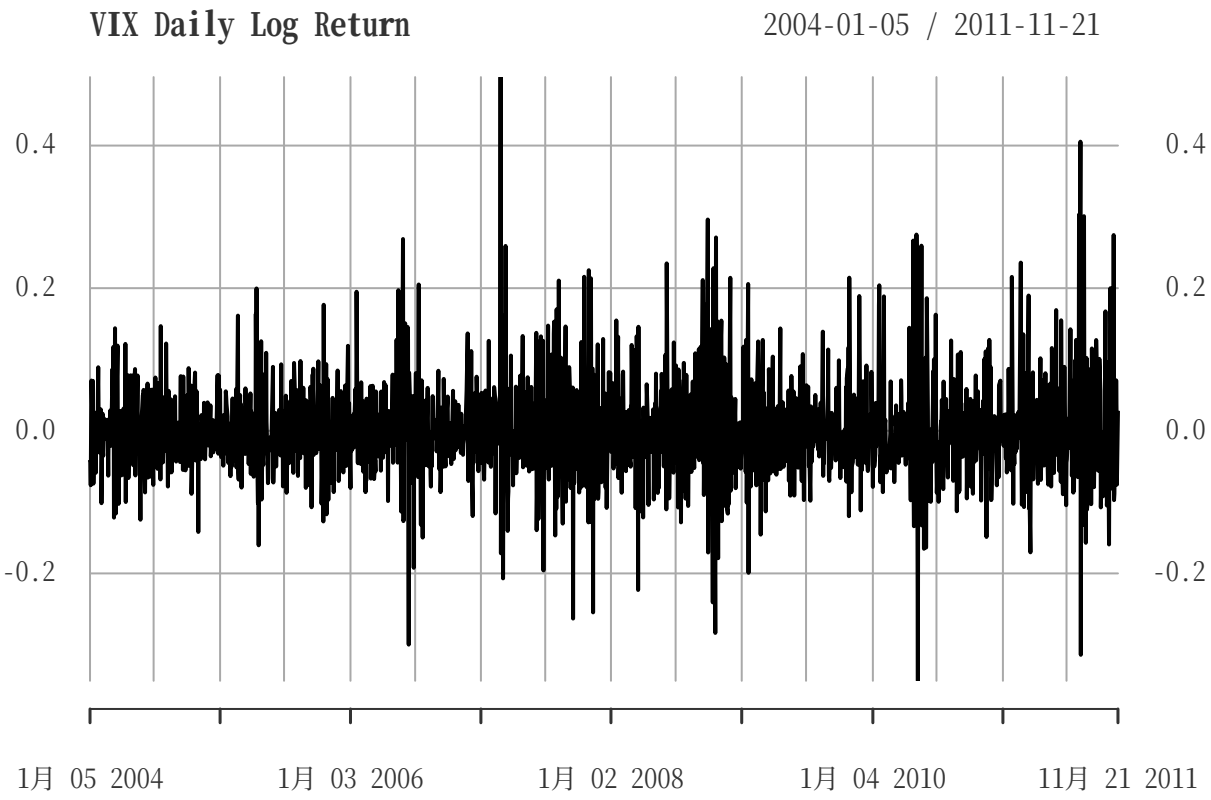


图 8.2: VIX 日对数收益率

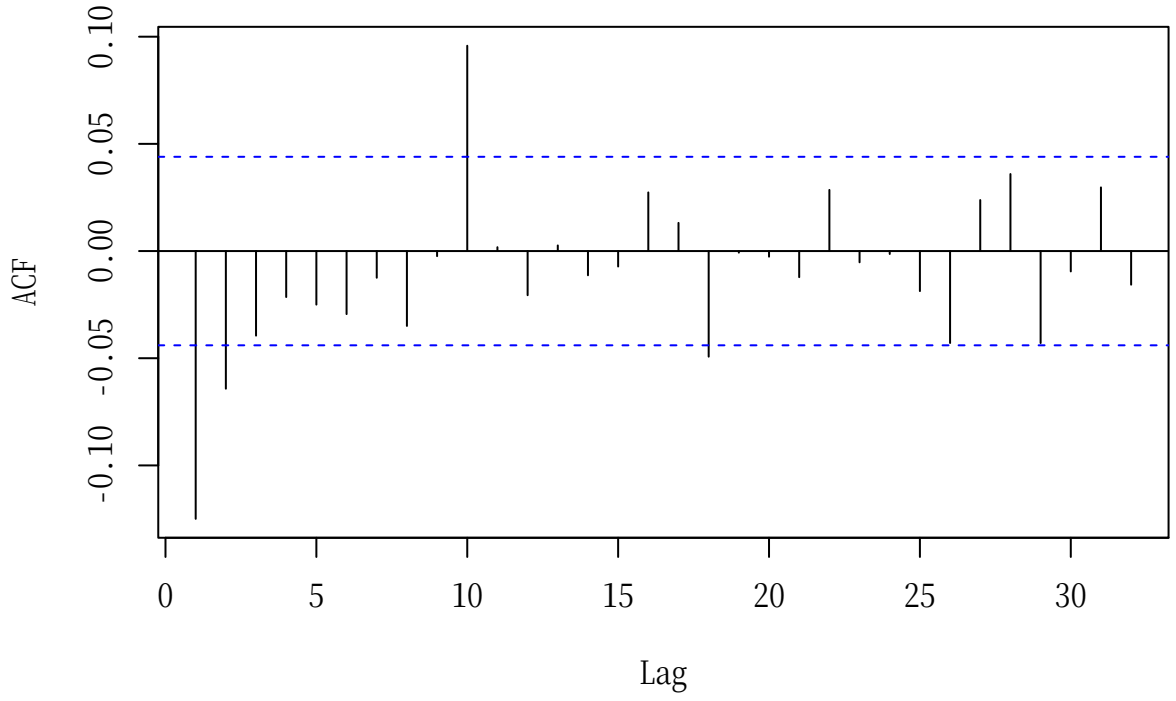


图 8.3: VIX 日对数收益率的 ACF

```
## arima(x = c(coredata(vix)), order = c(0, 1, 1))
##
## Coefficients:
##           ma1
##       -0.1500
## s.e.   0.0244
##
## sigma^2 estimated as 0.004561:  log likelihood = 2535.71,  aic = -5067.42
```

建立的模型为

$$X_t = X_{t-1} + \varepsilon_t - 0.1500\varepsilon_{t-1}$$

其中 $\hat{\sigma}^2 = 0.004561$。$\theta = 0.1500$。

检验残差是否白噪声：

```
Box.test(resm$residuals, lag=10, fitdf=1)
```

```
##
##  Box-Pierce test
##
## data:  resm$residuals
## X-squared = 47, df = 9, p-value = 3.925e-07
```

白噪声检验很显著，说明模型有所不足。从差分序列的 $\hat{\rho}_{10}$ 很大可以预见这个问题。

另行试验 ARIMA(0,1,10):

```
resm2 <- arima(c(coredata(vix)), order=c(0,1,10)); resm2
```

```
##
## Call:
## arima(x = c(coredata(vix)), order = c(0, 1, 10))
##
## Coefficients:
##           ma1      ma2      ma3      ma4      ma5      ma6      ma7      ma8
##       -0.1493  -0.0716  -0.0484  -0.0258  -0.0303  -0.0401  -0.0131  -0.0181
## s.e.   0.0223   0.0226   0.0228   0.0225   0.0226   0.0236   0.0231   0.0219
##          ma9     ma10
##       0.0092  0.0966
## s.e.  0.0245  0.0230
##
## sigma^2 estimated as 0.004453:  log likelihood = 2559.57,  aic = -5097.13
```

```
Box.test(resm2$residuals, lag=20, fitdf=10)
```

```
##
##  Box-Pierce test
##
```

```
## data:  resm2$residuals
## X-squared = 9.5782, df = 10, p-value = 0.4782
```

这个模型的白噪声检验通过，而且 AIC 也更好。

注意我们用了 8 年的数据，教材 (Tsay, 2013) 用的数据是 2008-05-01 到 2010-04-19 为两年数据。两年数据可能 ARIMA(0,1,1) 就足够。

8.2 时间序列成分分解

简单指数平滑，相当于只用一个变化的水平值 ℓ_t 拟合数据 y_t，

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1},$$

预测为

$$\hat{y}_{t+h|t} = \ell_t.$$

可以增加一个线性的趋势增长项 b_t，变成：

$$\begin{aligned}
\ell_t =& \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}),\\
b_t =& \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1},\\
\hat{y}_{t+h|t} =& \ell_t + hb_t.
\end{aligned}$$

可以增加季节项 s_t，变成：

$$\begin{aligned}
\ell_t =& \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}),\\
b_t =& \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1},\\
s_t =& \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},\\
\hat{y}_{t+h|t} =& \ell_t + hb_t + s_{t-m+h_m^+}.
\end{aligned}$$

其中 m 是季节频率，如月度数据为 12，季度数据为 4。$t-m+h_m^+$ 是 $t, t-1, \ldots, t-m+1$ 中与 $t-m+h$ 同月（季度）的值。

各项可以是相乘的。

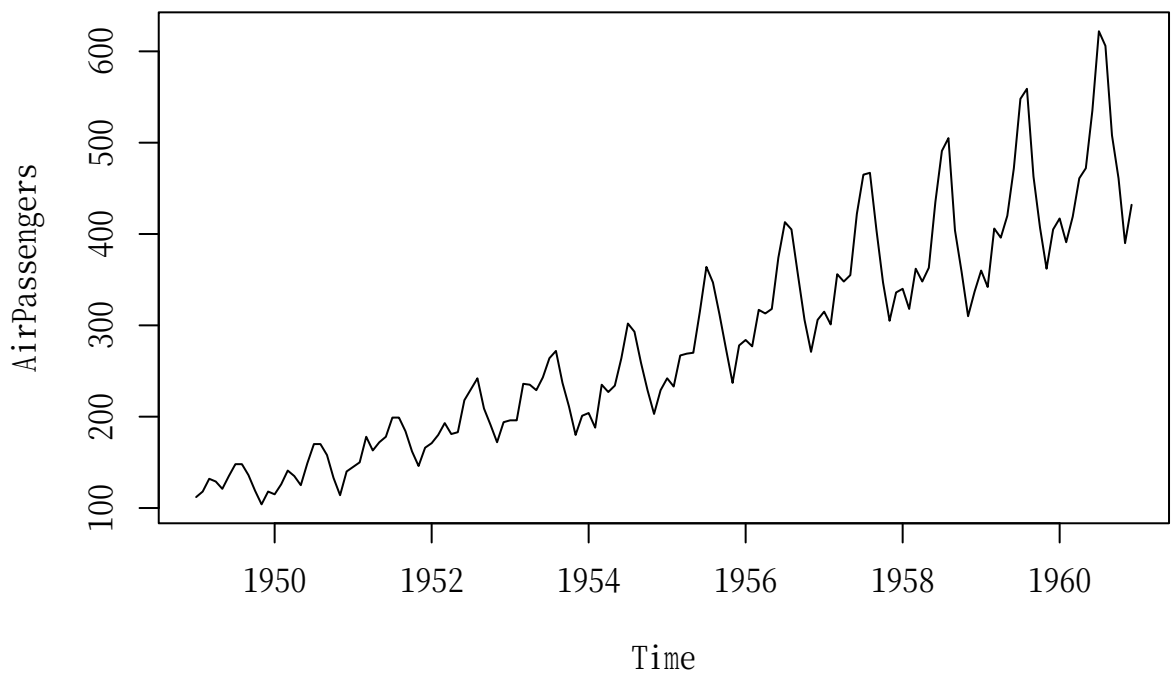
支持时间序列分解模型的 R 函数：

- stats 包的 `HoltWinters()` 函数提供这样的指数平滑方法。
- forecast 包的 `ets()` 函数提供了自动选择合适的指数平滑方法并进行预报的功能。
- stats 包的 `StructTS()` 使用最大似然估计方法估计这样的分解，仅支持加性模型。
- stats 包的 `stl()` 用局部多项式拟合方法进行分解。
- stats 包的 `decompose()` 用简单的滑动平均方法估计分解成分。

例 8.2. 考虑著名的航空数据的建模。AirPassengers 时间序列为 1949 年到 1960 年某航空公司月乘客人数。

图形：

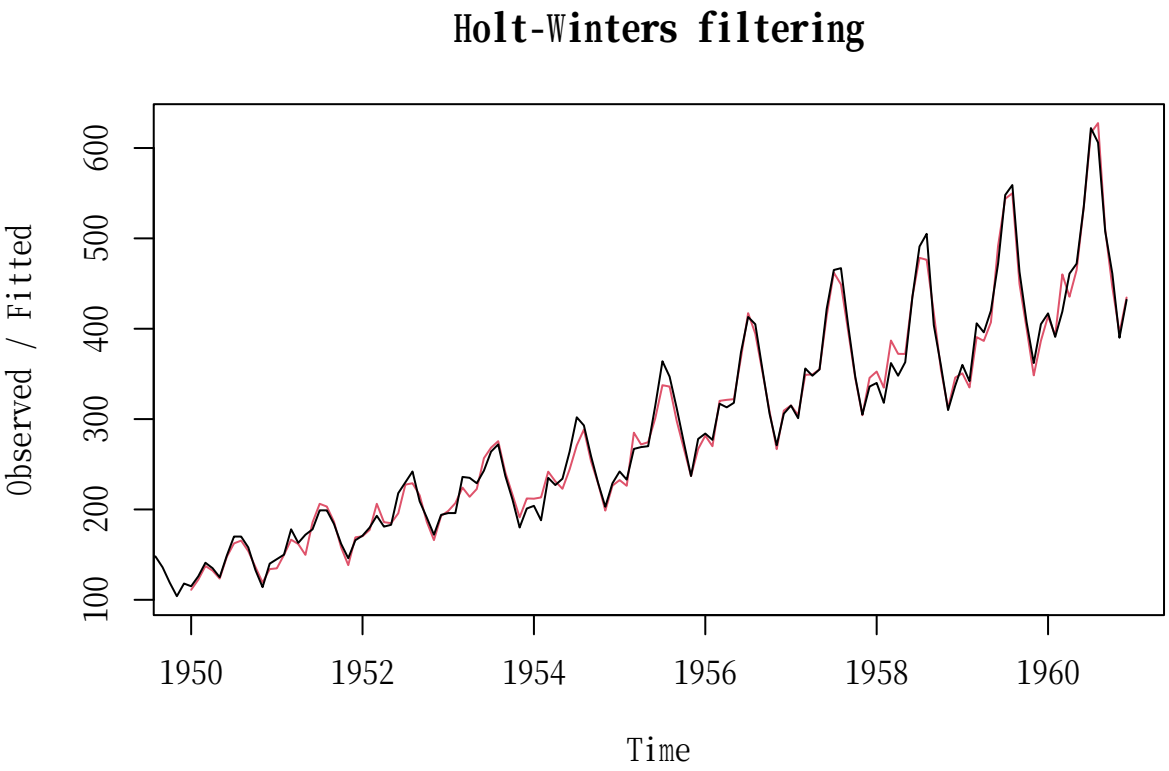
```
plot(AirPassengers)
```



数据包含趋势、季节项，并且应为乘性模型，或者取对数后建模。

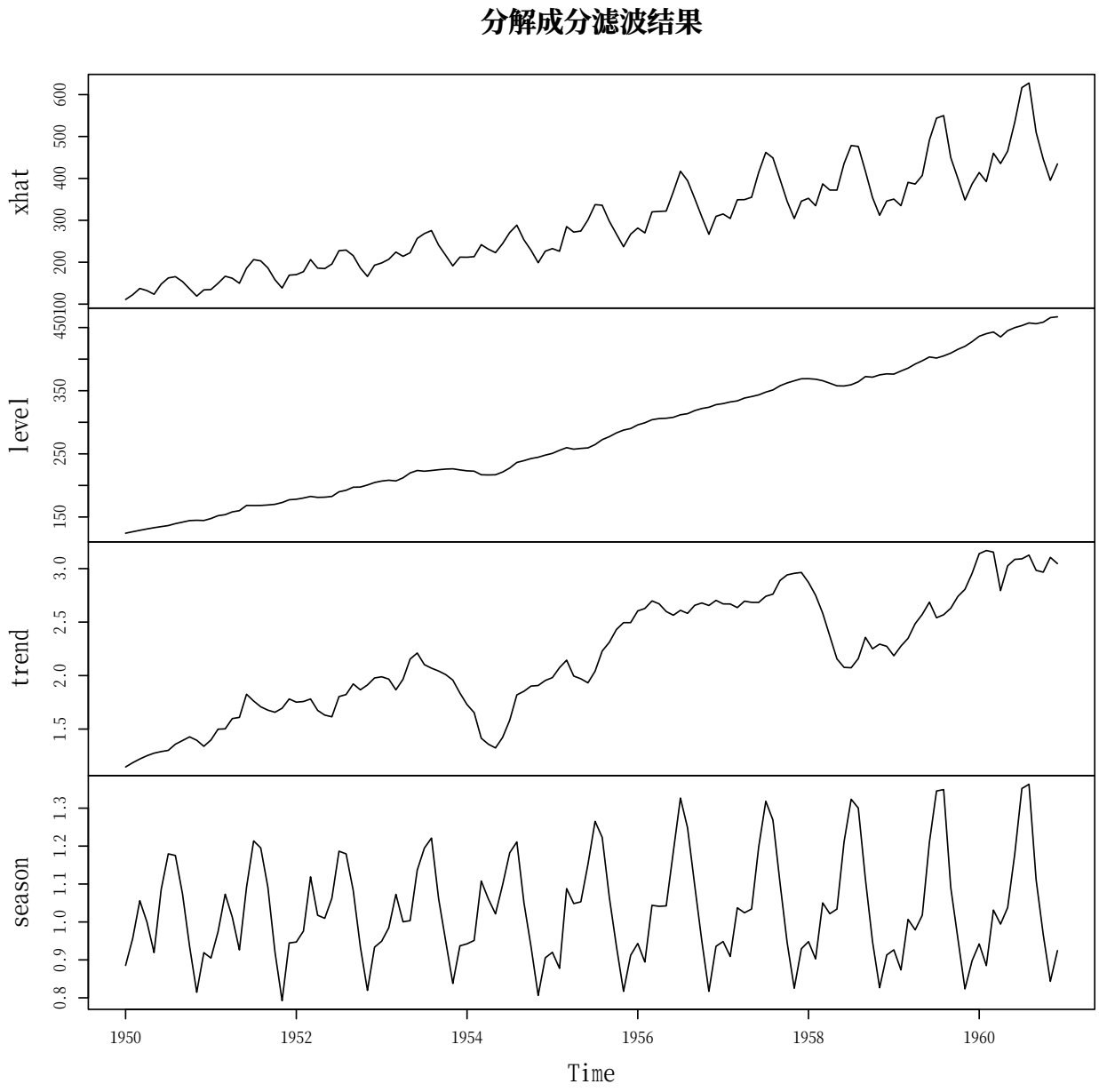
用 HoltWinters() 建模分解，并作拟合图：

```
al_hw <- HoltWinters(
  AirPassengers,
  seasonal = "multiplicative")
plot(al_hw)
```



各个分解成分的滤波结果：

```
plot(al_hw$fitted, main=" 分解成分滤波结果")
```



各成分为拟合值、l_t、b_t、s_t 的滤波估计。

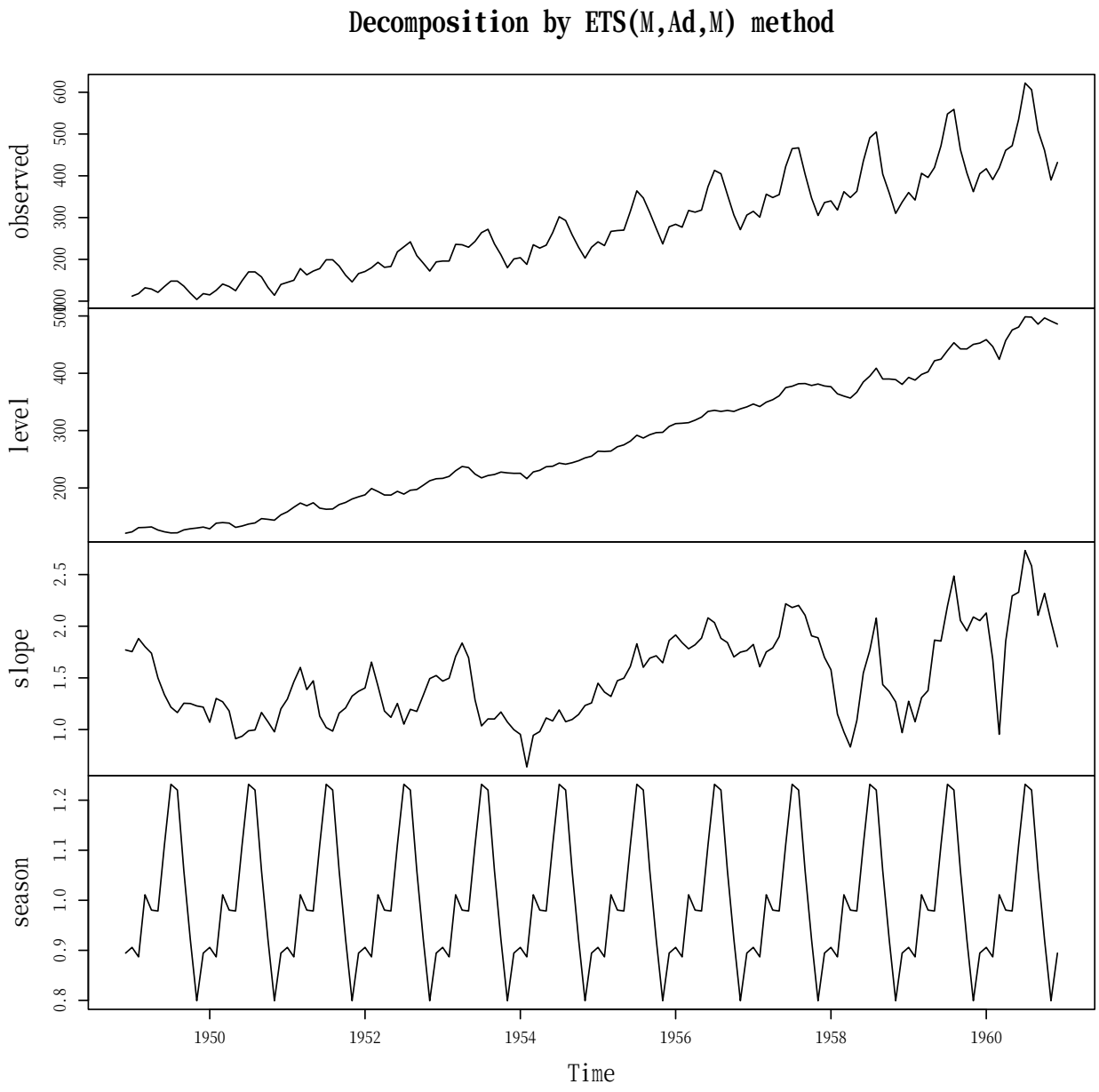
使用 `forecast::ets()`:

```
(al_ets <- forecast::ets(AirPassengers))
```

```
## ETS(M,Ad,M)
##
## Call:
##  forecast::ets(y = AirPassengers)
##
##   Smoothing parameters:
##     alpha = 0.7096
```

```
##     beta  = 0.0204
##     gamma = 1e-04
##     phi   = 0.98
##
##   Initial states:
##     l = 120.9939
##     b = 1.7705
##     s = 0.8944 0.7993 0.9217 1.0592 1.2203 1.2318
##            1.1105 0.9786 0.9804 1.011 0.8869 0.9059
##
##   sigma:  0.0392
##
##      AIC     AICc      BIC
## 1395.166 1400.638 1448.623
```

```
plot(al_ets)
```



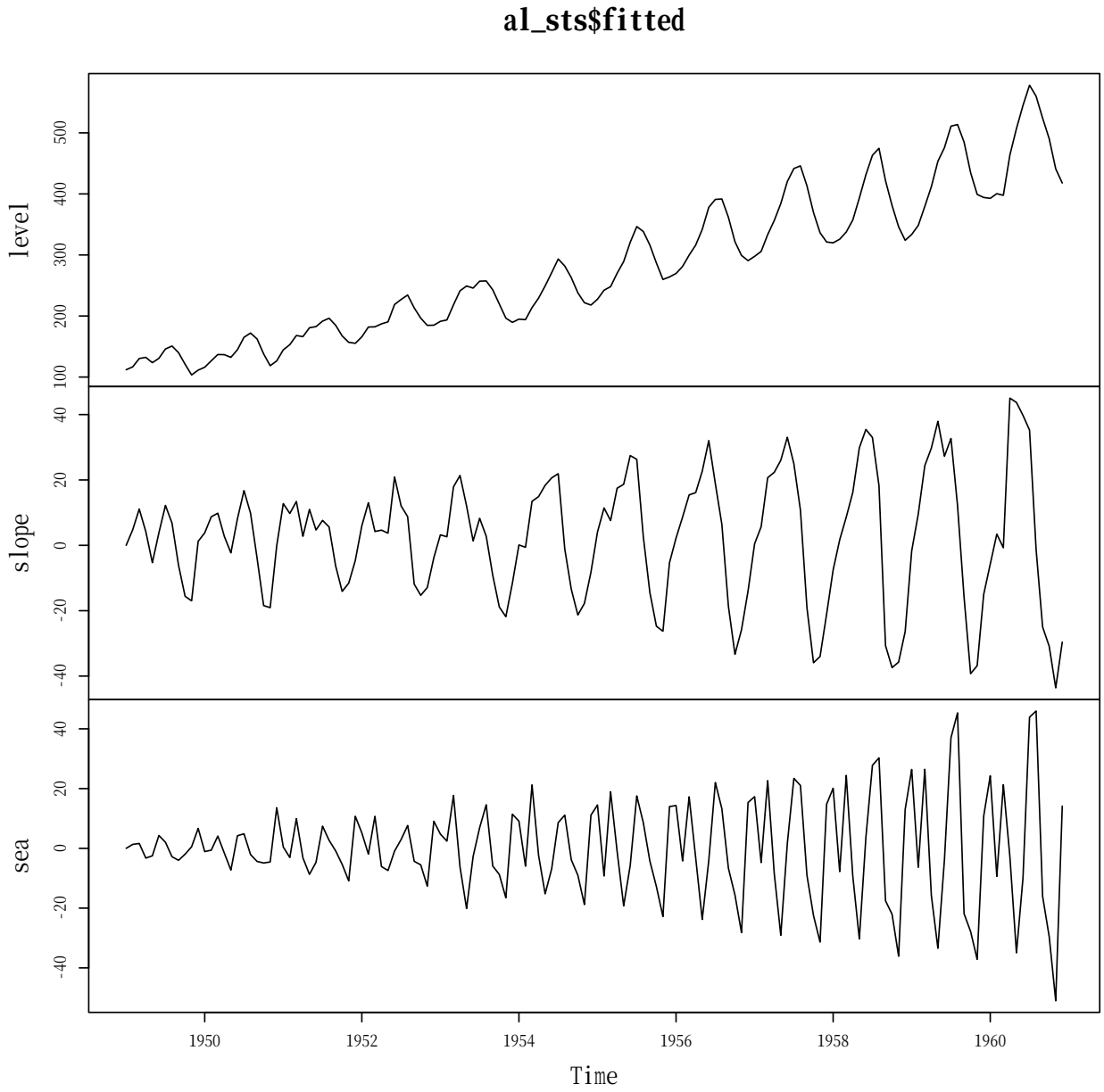
这个模型中误差项和季节项是乘性的，趋势项是加性的。

使用 StructTS():

```
(al_sts <- StructTS(AirPassengers))
```

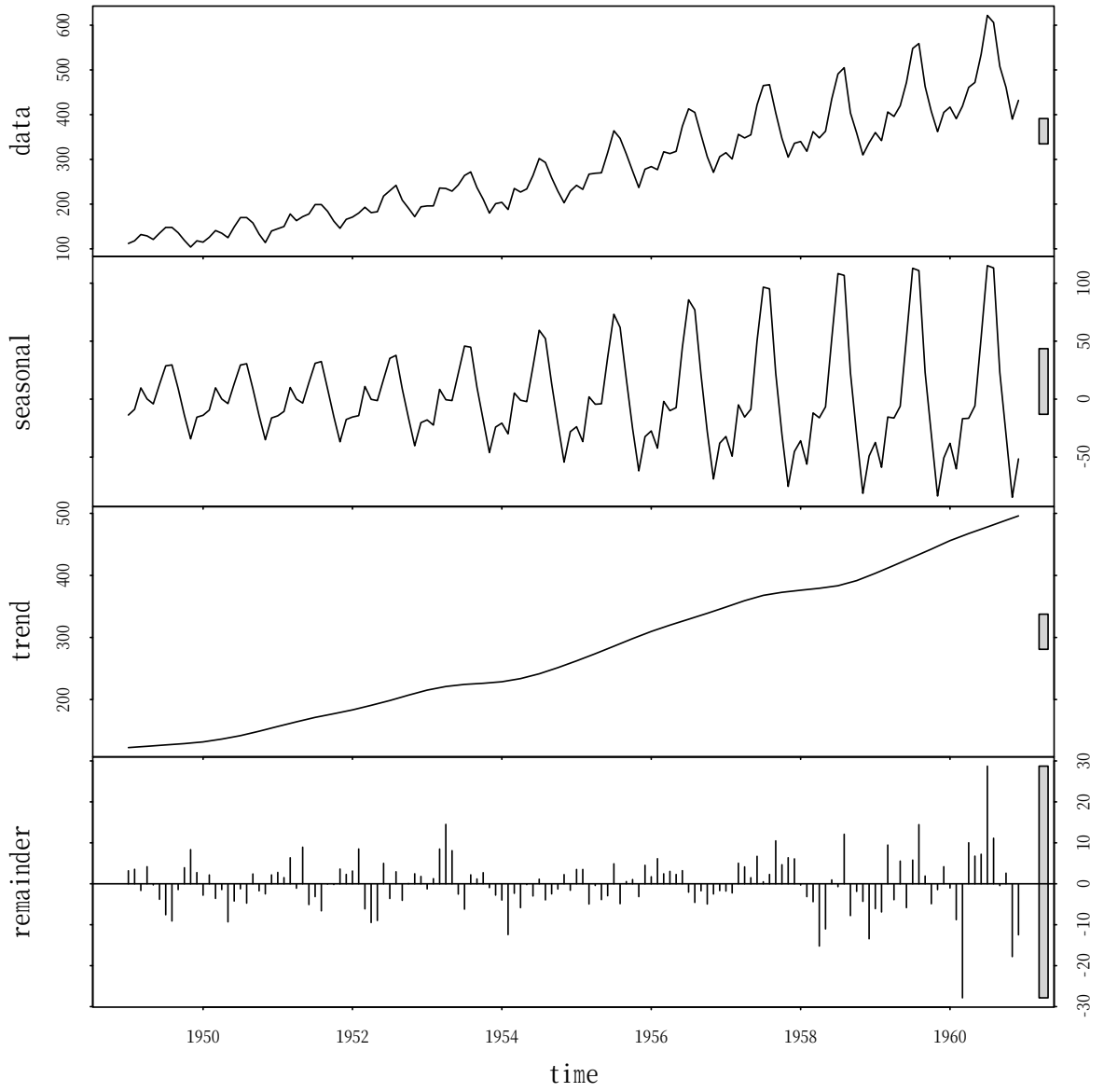
```
##
## Call:
## StructTS(x = AirPassengers)
##
## Variances:
##   level    slope     seas  epsilon
##    0.00   160.98    29.85     0.00
```

```
plot(al_sts$fitted)
```



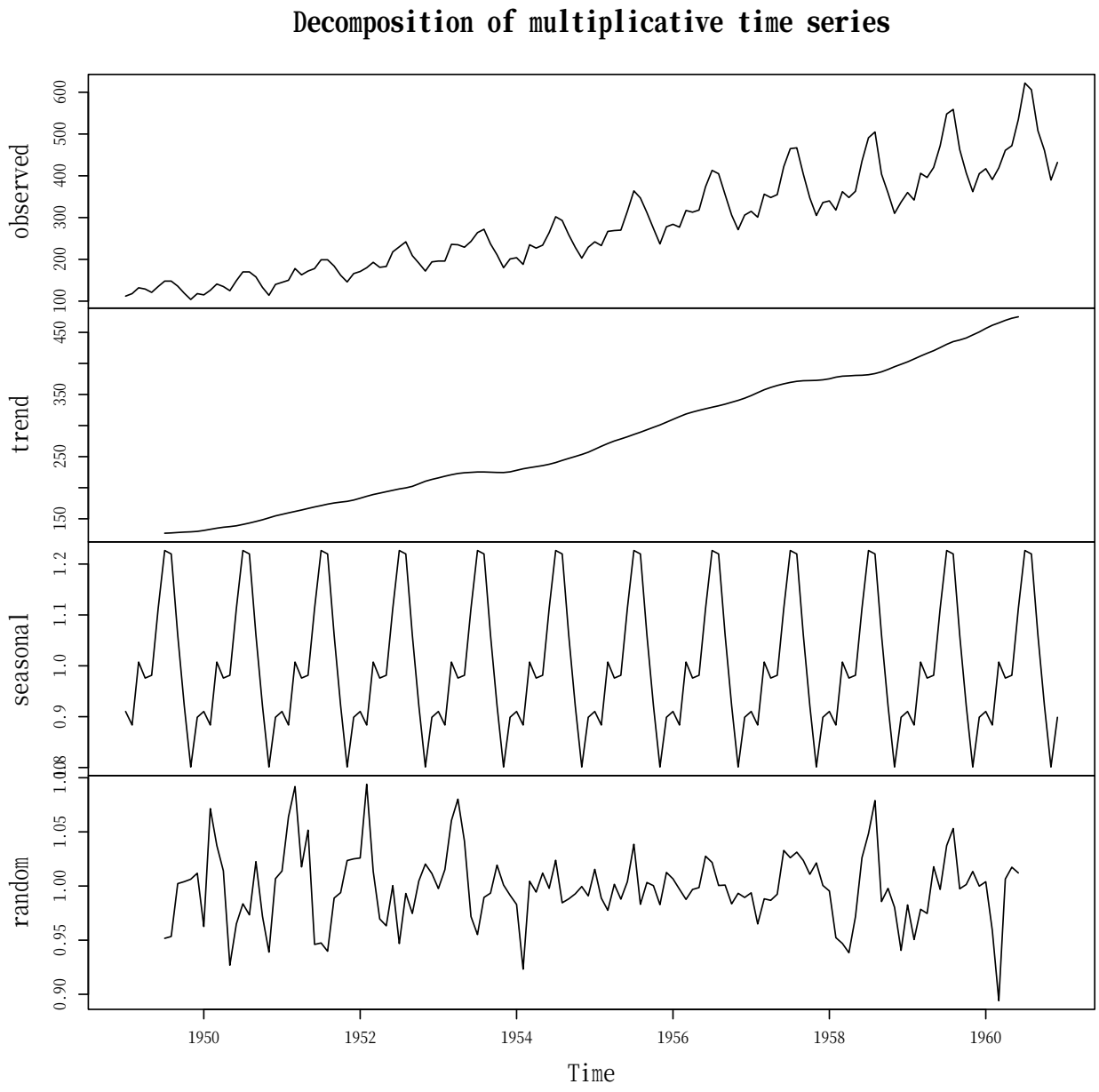
使用 stl():

```
al_stl <- stl(AirPassengers, s.window=7)
plot(al_stl)
```



使用 decompose():

```
al_dec <- decompose(AirPassengers,
  type = "multiplicative")
plot(al_dec)
```


Decomposition of multiplicative time series
observed
trend
seasonal
random
Time
1950
1952
1954
1956
1958
1960

Chapter 9

季节模型

经济和金融中的月度、季度数据一般有明显的周期，日数据也会有按照周、月、年周期的变化。这样的性质称为季节性，含有周期变化的时间序列称为季节时间序列。

如：可口可乐公司 1983 第 1 季度到 2009 第 3 季度公布的季度盈利数据。每个季度的盈利数据在季度结束后约一个月以后公布。共 107 个观测。考虑季度盈利的对数值。

```
da <- read_table(
  "q-ko-earns8309.txt",
  col_types=cols(pends=col_date("%Y%m%d"),
                 anntime=col_date("%Y%m%d"),
                 value = col_double()))
xts.koqtr <- xts(da[["value"]], ymd(da[["pends"]]))
ko <- ts(log(da[["value"]]), start=c(1983,1), frequency=4)
```

注：原始的 `q-ko-earns8309.txt` 第一行末尾有一个多余的空格，会使得 `read_table()` 函数认为还有第 4 列。已人为删除此空格。`read_table()` 函数应添加自动删除行首和行尾空格的功能。

季度盈利的对数值序列图：

```
plot(ko, type="l", main="Coca Kola Quarterly Log Earnings")
```

图9.1序列呈现出明显的周期为 4 的波动。如果是月度数据，周期为 12。盈利做了对数变换，其中一个理由是消除指数增长（倍数增长）现象，对数变化可以将指数增长变成线性增长。

农产品等与天气有关的衍生产品定价、能源期货定阶等与天气有关的金融产品研究中，季节性是重要的考虑因素。

经济研究中有时希望排除季节性影响，用某种方法去掉季节性的序列称为季节调整序列 (seasonally adjusted series)，如美国季节调整的 GNP 序列。X12-ARIMA 是一个成熟的季节调整方法。

季节因素的建模，也包括动态模型（类似于随机游动）和非随机模型（类似于固定线性趋势模型）。

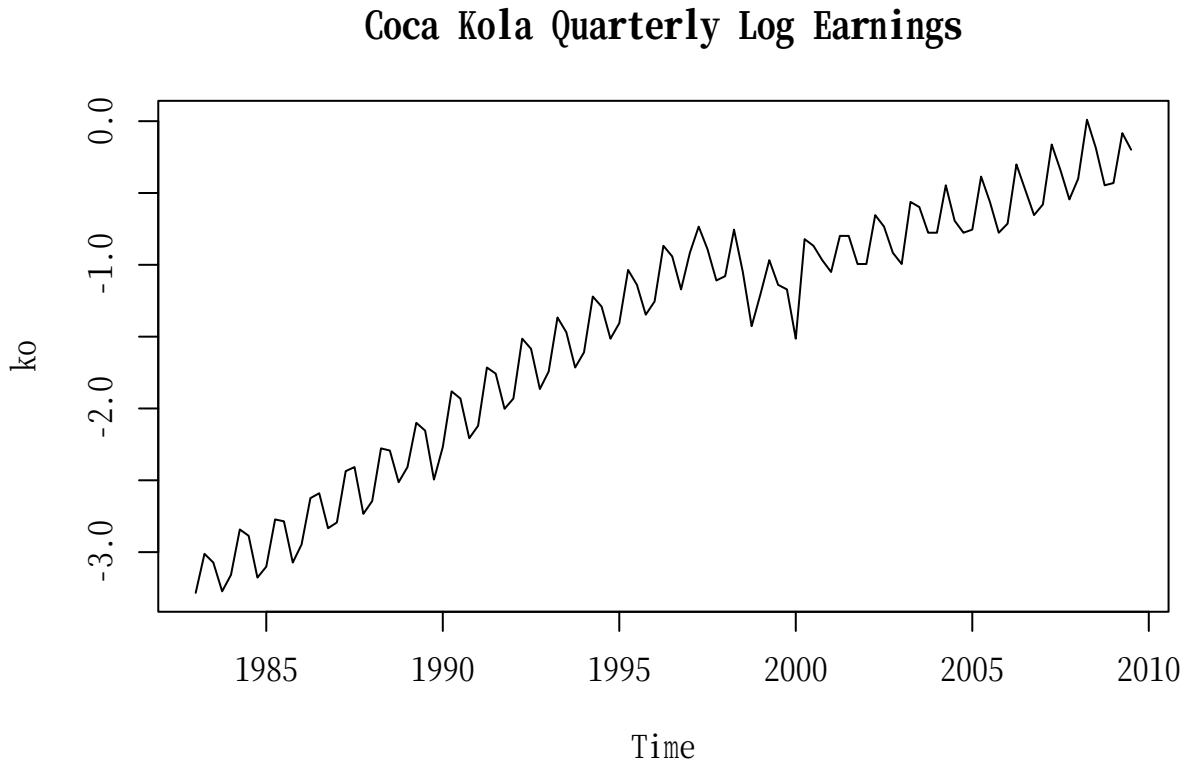


图 9.1: 可口可乐公司季度盈利对数值

9.1 季节差分

考虑可口可乐公司每股季度盈利对数值序列，记为 $\{x_t\}$，见图9.1。因为图形呈现出线性增长，明显是不平稳的，所以考虑其一阶差分：

```
d.ko <- diff(ko)
plot(d.ko, type="l", main="Coca Kola Quarterly Log Earnings diff()")
```

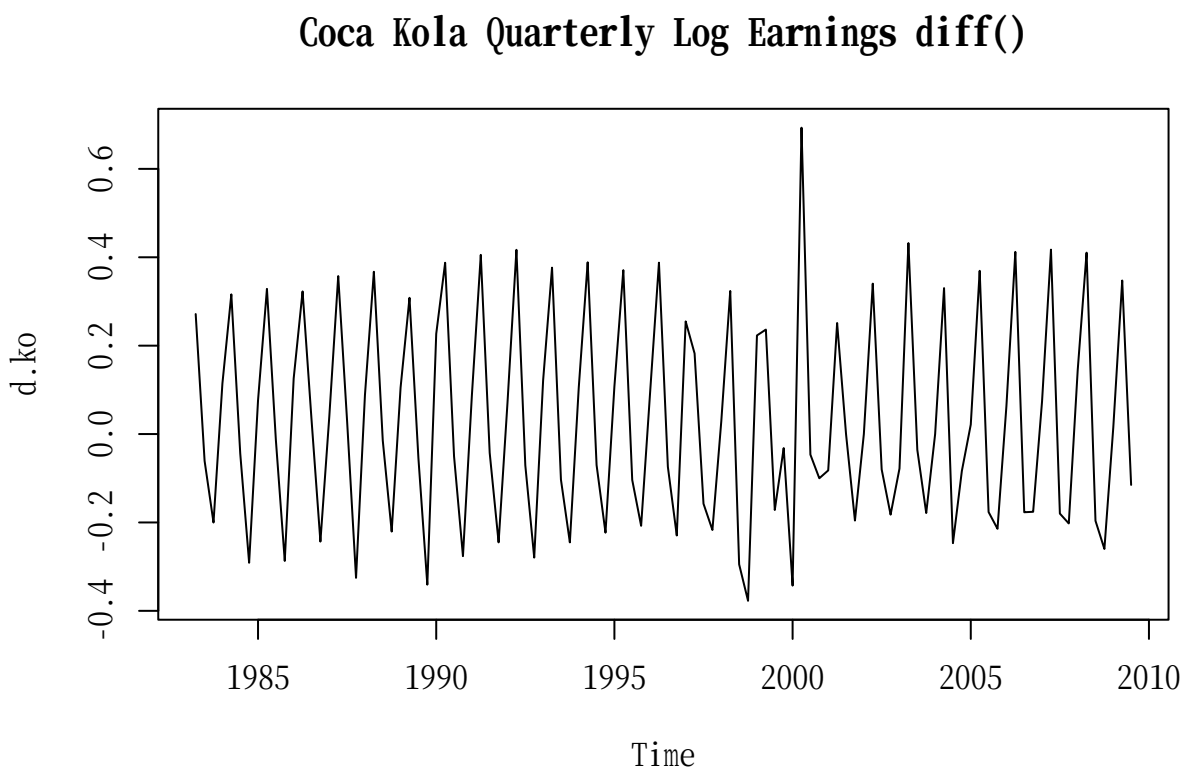


图 9.2: 可口可乐公司季度盈利对数值一阶差分

一阶差分后，消除了线性增长趋势，但是还有明显的季节性，这往往也是不平稳的表现，看其 ACF:

```
d.ko <- diff(ko)
forecast::Acf(d.ko, main="")
```

一阶差分的 ACF 衰减速度很慢，不适于用低阶的 ARMA 建模。一般 ARMA 序列应该是 ACF 衰减速度很快的。

消除季节性影响的方法是，第二年第一季度值减去第一年第一季度值，第二年第二季度值减去第一年第二季度值，第二年第三季度值减去第一年第三季度值，第二年第四季度值减去第一年第四季度值。有些类似于经济数据中指标的“同比”的概念。这称为季节差分，实际是计算 $(1-B^4)x_t = x_t - x_{t-4}$，R 中用 `diff(x, lag=4)` 计算这样的季节差分。例如：

```
tmp <- window(ko, start=c(1983,1), end=c(1985,4)); tmp
```

```
##           Qtr1      Qtr2      Qtr3      Qtr4
## 1983 -3.283414 -3.011862 -3.072613 -3.272804
```

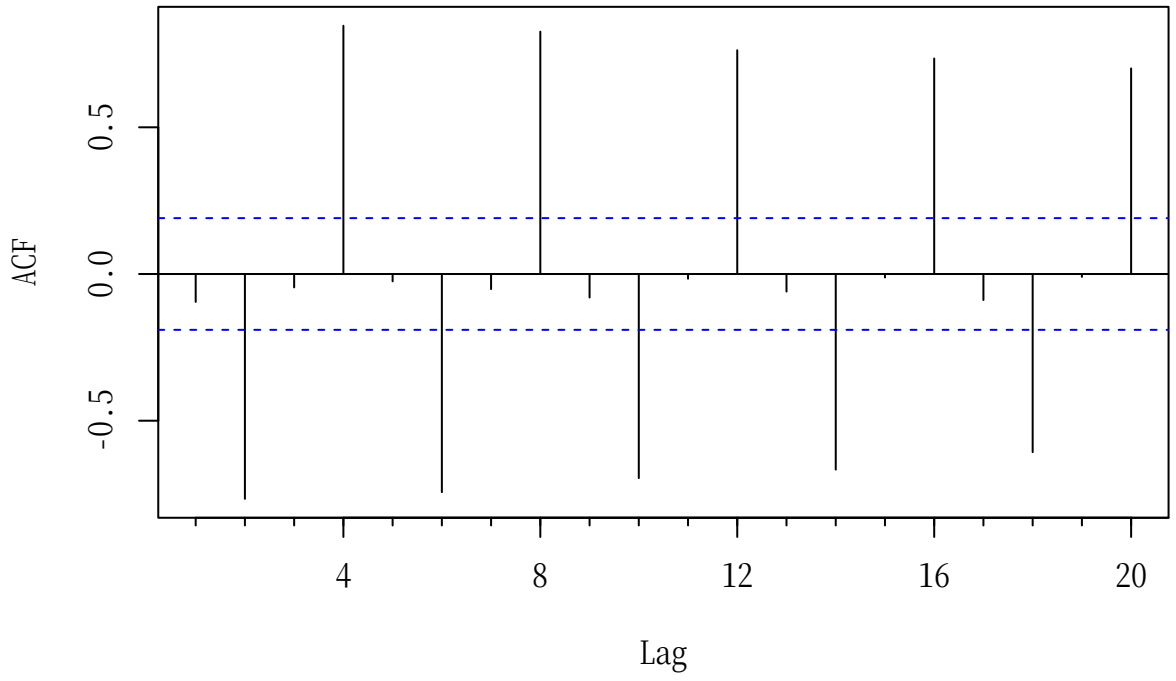


图 9.3: 可口可乐公司季度盈利对数值一阶差分的 ACF

```
## 1984 -3.158251 -2.842153 -2.885981 -3.177254
## 1985 -3.101093 -2.772589 -2.785471 -3.072613
```

```
diff(tmp, lag=4)
```

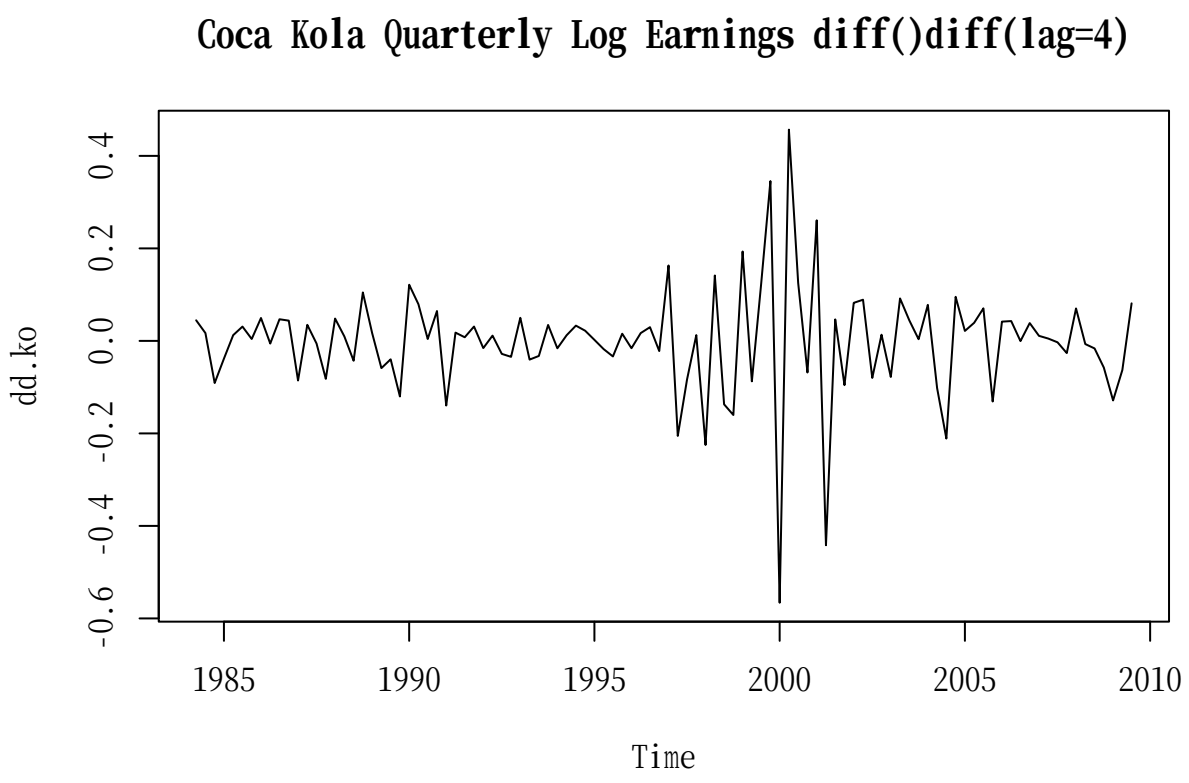
```
##            Qtr1       Qtr2       Qtr3       Qtr4
## 1984 0.12516314 0.16970847 0.18663191 0.09555002
## 1985 0.05715841 0.06956446 0.10051006 0.10464083
```

先做一阶差分 $(1-B)x_t$，然后再做一阶季度差分 $(1-B^4)(1-B)x_t = x_t - x_{t-1} - x_{t-4} + x_{t-5}$，得到新的序列 $\{y_t\}$：

```
dd.ko <- diff(diff(ko), lag=4)
str(dd.ko)
```

```
##  Time-Series [1:102] from 1984 to 2010: 0.0445 0.0169 -0.0911 -0.0384 0.0124 ...
```

```
plot(dd.ko, main="Coca Kola Quarterly Log Earnings diff()diff(lag=4)")
```



因为需要用到 x_{t-5}，所以变换后的序列从原来的第 6 个观测开始。从序列图形看已经消除了线性趋势与大部分季节波动。

查看一阶差分与一阶季度差分后的序列的 ACF:

```
forecast::Acf(dd.ko, main="")
```

此 ACF 的衰减已经比仅作一阶差分的序列的 ACF 衰减快多了，比较适合 ARMA 序列的 ACF 形状。注意 ACF 则滞后 1 和滞后 4（滞后 4 对应季节波动）都是显著的负值，这与一阶差分和季度差分有关。ACF 在滞后 5 也是显著的正值，部分理由是 $(1-B^4)(1-B)$ 中包含 B^5 项。这些 ACF 表现是许多经过一阶差分和季节差分的序列共同

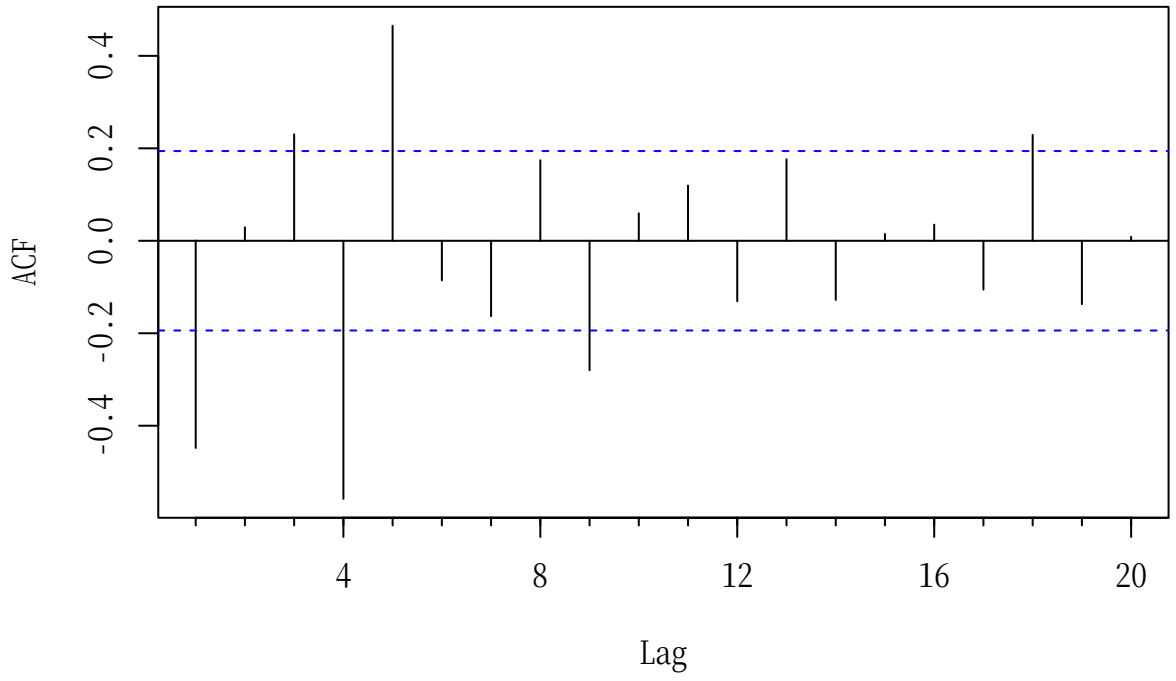


图 9.4: 可口可乐公司季度盈利对数值一阶差分和一阶季节差分的 ACF

的特征。

作 PACF 图：

```
pacf(dd.ko, main="")
```

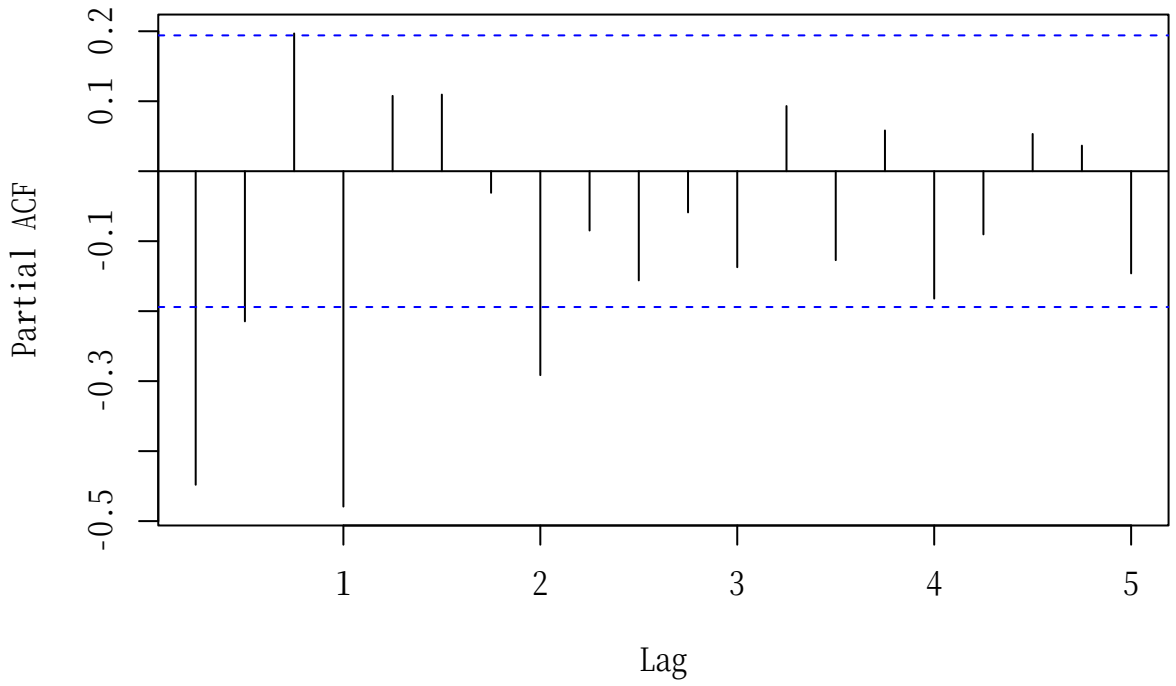


图 9.5: 可口可乐公司季度盈利对数值一阶差分和一阶季节差分的 PACF

PACF 也没有很快地截尾。有可能需要使用 ARMA 模型。

9.2 乘性季节模型

季节性的时间序列经过一阶差分和季节差分后的序列常常仍在滞后 1、4、5 这些位置呈现出自相关（如果是月度数据，则为滞后 1、12、13）。为此，对差分后序列建立 MA 序列，这样，$\{x_t\}$ 的模型为

$$(1-B)(1-B^s)x_t = (1-\theta B)(1-\Theta B^s)\varepsilon_t$$

其中 s 是周期，对上面的季度数据 $s=4$。这个模型称为航空模型，最早用在分析航空乘客数月度数据当中。

设 Y_t 为右边的 MA($s+1$ 模型)：

$$Y_t = (1-\theta B)(1-\Theta B^s)\varepsilon_t = \varepsilon_t - \theta\varepsilon_{t-1} - \Theta\varepsilon_{t-s} + \theta\Theta\varepsilon_{t-s-1}$$

易见 $EY_t = 0$,

$$\begin{aligned}
\gamma_0 =& \mathrm{Var}(Y_t) = (1+\theta^2)(1+\Theta^2)\sigma^2 \\
\gamma_1 =& -\theta(1+\Theta^2)\sigma^2 \\
\gamma_{s-1} =& \theta\Theta\sigma^2 \\
\gamma_s =& -\Theta(1+\theta^2)\sigma^2 \\
\gamma_{s+1} =& \theta\Theta\sigma^2 \\
\gamma_k =& 0,\ k \neq 0, 1, s-1, s, s+1
\end{aligned}$$

所以 $\{Y_t\}$ 的 ACF 为

$$\rho_1 = \frac{-\theta}{1+\theta^2},\ \rho_s = \frac{-\Theta}{1+\Theta^2}, \rho_{s-1} = \rho_{s+1} = \rho_1\rho_s = \frac{\theta\Theta}{(1+\theta^2)(1+\Theta^2)}$$

其它 $\rho_k = 0$。

设

$$\xi_t = (1-\theta B)\varepsilon_t, \quad \eta_t = (1-\Theta B^s)\varepsilon_t,$$

则

$$\rho_1^{(\xi)} = \frac{-\theta}{1+\theta^2}, \rho_s^{(\eta)} = \frac{-\Theta}{1+\Theta^2},$$

且 $\rho^{(\xi)}$ 和 $\rho^{(\eta)}$ 在其它滞后的值为零。比较 $\{Y_t\}$ 的 ACFρ_k 和 $\rho^{(\xi)}$、$\rho^{(\eta)}$ 发现：

$$\rho_1 = \rho_1^{(\xi)},\ \rho_s = \rho_s^{(\eta)},\ \rho_{s-1} = \rho_{s+1} = \rho_1^{(\xi)}\rho_s^{(\eta)}$$

其中 Y_t 的 ACF 在滞后 $s \pm 1$ 的值可以认为是 MA(1) 和 MA(4) 的 ACF 的交互作用。$\{Y_t\}$ 的模型称为乘性季节 MA 模型 (multiplicative MA model)。可以认为序列的同一季节两年之间的变化与相邻两个季节之间的变化时正交变化的。

航空模型与指数平滑也有关系。将原来的模型改写为

$$\frac{1-B}{1-\theta B}\left(\frac{1-B^s}{1-\Theta B^s}x_t\right) = \varepsilon_t$$

则可将其写成两步：

$$\frac{1-B}{1-\theta B}z_t = \varepsilon_t, \quad \frac{1-B^s}{1-\Theta B^s}x_t = z_t$$

第一个模型是一个 ARIMA(0,1,1) 模型，对应于指数平滑；第二个模型的输入是第一个模型的序列，对应于季节 s 的 $\mathrm{ARIMA}(0,1,1)_s$。所以航空模型可以看成是两个指数平滑的复合。

比航空模型略复杂一点儿的模型是增加一个滞后 2 的 MA 项，即

$$(1-B)(1-B^s)x_t = (1-\theta_1 B-\theta_2 B^2)(1-\Theta B^s)\varepsilon_t$$

在 $s > 4$ 较常出现。这时 ACF 不等于零的滞后位置是 $1, 2, s, s \pm 1, s \pm 2$。

另一个非乘积形式的带有季节因素的 MA 模型是

$$X_t = (1-\theta B-\Theta B^s)\varepsilon_t$$

但是乘积形式的更常见。

在 `arima()` 函数中，可以用 `seasonal=` 指定季节模型，包括季节 AR 阶、季节差分阶、季节 MA 阶以及周期。如

```
resm <- arima(
  ko, order=c(0,1,1), seasonal=list(order=c(0,1,1), period=4)
); resm
```

```
##
## Call:
## arima(x = ko, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 4))
##
## Coefficients:
##           ma1     sma1
##       -0.4096  -0.8203
## s.e.   0.0866   0.0743
##
## sigma^2 estimated as 0.00724:  log likelihood = 104.25,  aic = -202.5
```

结果模型为

$$(1-B)(1-B^4)x_t = (1-0.4096B)(1-0.8203B^4)\varepsilon_t$$

$\hat{\sigma}^2 = 0.00724$。注意 `arima()` 函数给出的 MA 多项式是 $1+\theta_1 B+\cdots+\theta_q B^q$ 形式的。

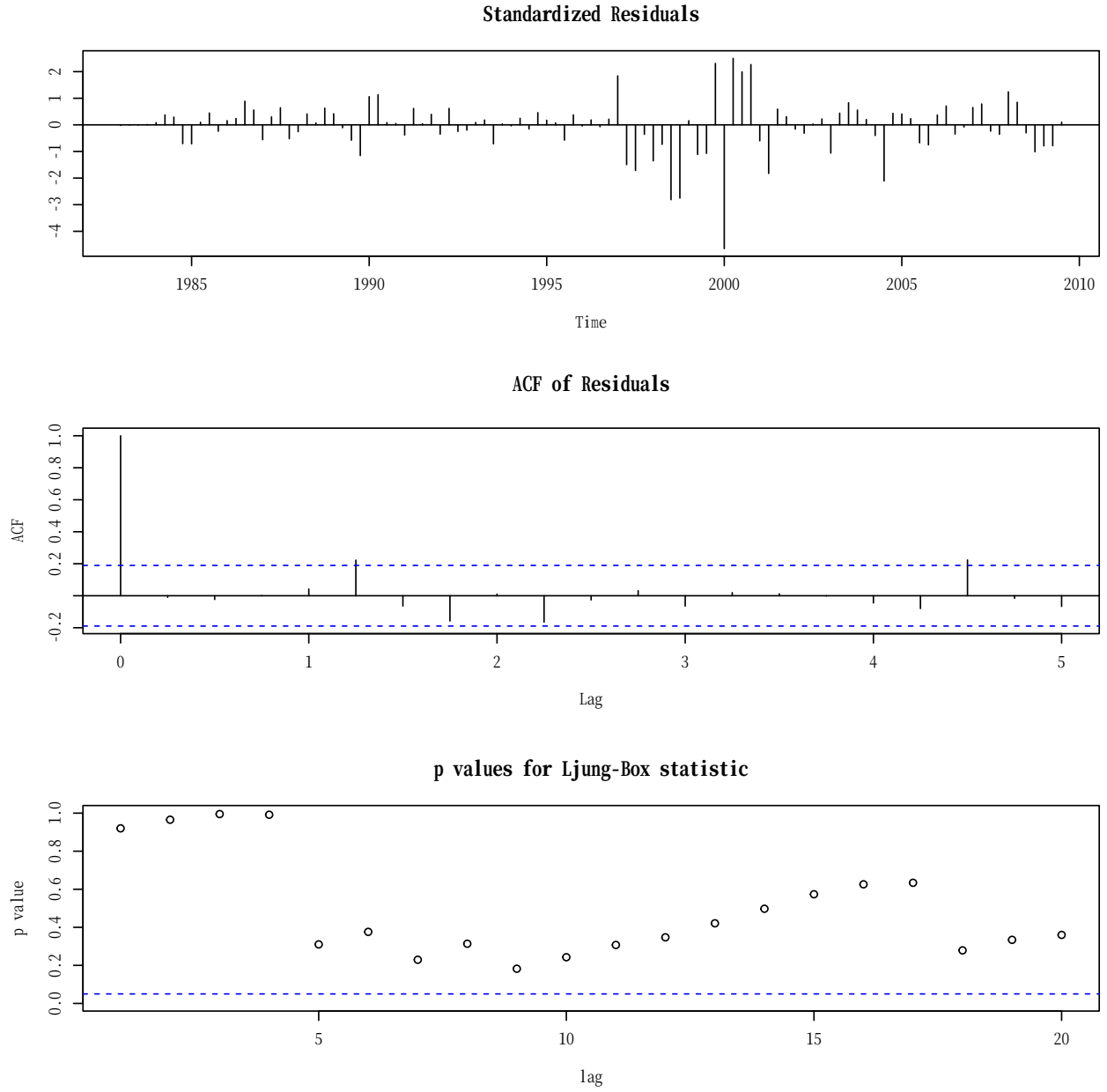
对模型残差做 LB 白噪声检验：

```
Box.test(resm$residuals, lag=12, fitdf=2)
```

```
##
##  Box-Pierce test
##
## data:  resm$residuals
## X-squared = 12.233, df = 10, p-value = 0.2698
```

结果不显著，表示承认模型合适。`arima()` 的结果还可以用 `tsdiag()` 函数检验残差是否白噪声，如:

```
tsdiag(resm, gof=20)
```



其中 gof= 指定最大滞后。诊断图中的第一个图时标准化残差；第二个图是残差的 ACF，除了滞后 0 之外应该落入两条水平线之间；第三个图是不同滞后值的 LB 白噪声检验的 p 值，应该位于检验水平 0.05 代表的水平线上方。

下面用航空模型做超前多步预报。用 1983 年到 2007 年这 25 年的 100 个观测值建模，对剩余的 7 个值作超前多步预报：

```
tmp.y <- window(ko, start=start(ko), end=c(2007,4))
resm2 <- arima(tmp.y, order=c(0,1,1), seasonal=list(order=c(0,1,1), period=4))
resm2
```

```
##
## Call:
## arima(x = tmp.y, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1), period = 4))
```

```
##
## Coefficients:
##           ma1     sma1
##       -0.4209  -0.8099
## s.e.   0.0874   0.0767
##
## sigma^2 estimated as 0.007432:  log likelihood = 95.78,  aic = -185.57
```

```
pred1 <- predict(resm2, n.ahead=7)
cbind(Observed=ko[101:107], Predicted=pred1$pred, SE=pred1$se)
```

```
##             Observed  Predicted         SE
## 2008 Q1 -0.400477567 -0.5060620 0.08621248
## 2008 Q2  0.009950331 -0.1237792 0.09962409
## 2008 Q3 -0.186329578 -0.2669296 0.11143307
## 2008 Q4 -0.446287103 -0.4501580 0.12210527
## 2009 Q1 -0.430782916 -0.4219704 0.13894879
## 2009 Q2 -0.083381609 -0.0396876 0.15111786
## 2009 Q3 -0.198450939 -0.1828380 0.16237749
```

这是每股季度盈利的对数值的预测。但是，如何得到每股季度盈利的预测？为了得到无偏估计，需要利用对数正态分布的性质进行期望得校正。如果随机变量 $\ln Y \sim \mathrm{N}(\mu, \sigma^2)$，则 $EY = \exp(\mu + \frac{1}{2}\sigma^2)$，$\mathrm{Var}(Y) = \exp(2\mu + \sigma^2)(e^{\sigma^2} - 1)$。

写成 R 函数：

```
lognorm_adjust <- function(pred_list){
  pred <- pred_list$pred
  se <- pred_list$se
  xnew <- exp(pred + 0.5*se^2)
  senew <- sqrt(exp(2*pred + se^2)*(exp(se^2)-1))
  list(pred = xnew, se=senew)
}
```

据此得到关于每股季度盈利的预测：

```
pred1adj <- lognorm_adjust(pred1)
pred2 <- pred1adj$pred
se2 <- pred1adj$se
cbind(Observed=c(coredata(xts.koqtr))[101:107],
      Predicted=round(pred2, 2), SE=round(se2, 2))
```

```
##         Observed Predicted   SE
## 2008 Q1     0.67      0.61 0.05
## 2008 Q2     1.01      0.89 0.09
## 2008 Q3     0.83      0.77 0.09
## 2008 Q4     0.64      0.64 0.08
## 2009 Q1     0.65      0.66 0.09
```

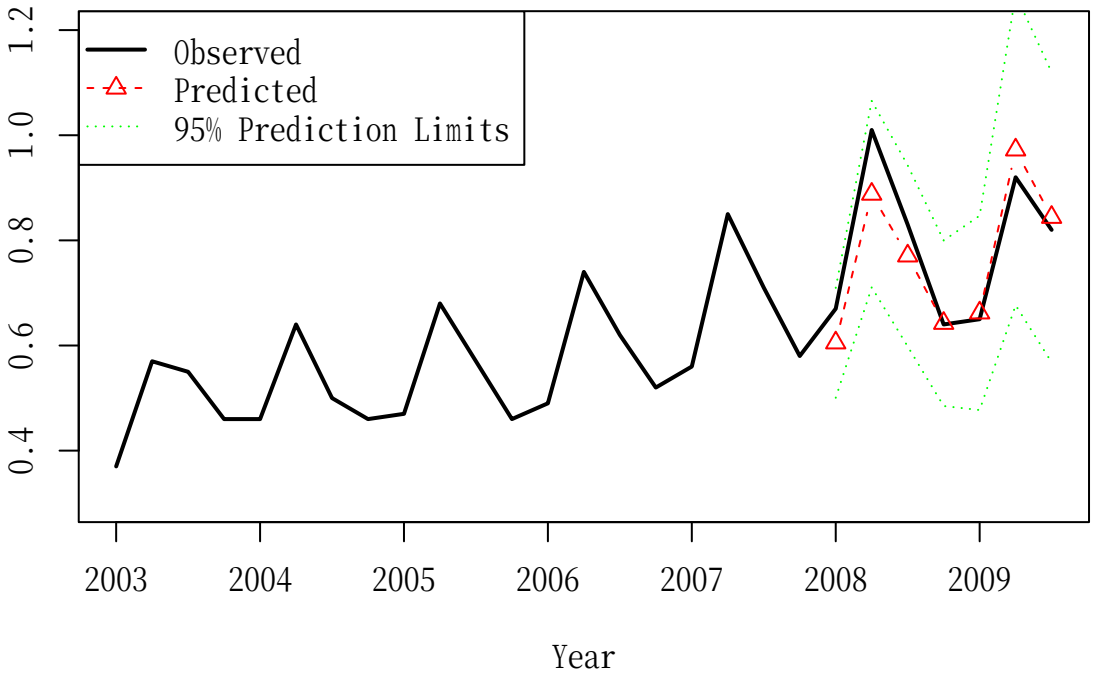
```
## 2009 Q2     0.92      0.97 0.15
## 2009 Q3     0.82      0.84 0.14
```

对预报效果作图，从 2003 年开始：

```
pred2 <- exp(pred1$pred + 0.5*pred1$se^2)
se2 <- sqrt(exp(2*pred1$pred + pred1$se^2)*(exp(pred1$se^2)-1))
cbind(Observed=c(coredata(xts.koqtr))[101:107], Predicted=round(pred2, 2), SE=round(se2, 2))
```

```
##         Observed Predicted   SE
## 2008 Q1     0.67      0.61 0.05
## 2008 Q2     1.01      0.89 0.09
## 2008 Q3     0.83      0.77 0.09
## 2008 Q4     0.64      0.64 0.08
## 2009 Q1     0.65      0.66 0.09
## 2009 Q2     0.92      0.97 0.15
## 2009 Q3     0.82      0.84 0.14
```

```
tmp.x <- ts(c(coredata(xts.koqtr["2003/"])), start=c(2003,1), frequency = 4)
tmp.p <- ts(c(pred2), start=c(2008,1), frequency = 4)
tmp.lb <- ts(c(pred2) - 2*c(se2), start=c(2008,1), frequency = 4)
tmp.ub <- ts(c(pred2) + 2*c(se2), start=c(2008,1), frequency = 4)
plot(tmp.x, ylim=c(0.3, 1.2), type="l", lwd=2,
     xlab="Year", ylab="")
lines(tmp.p, type="b", pch=2, lty=2, col="red")
lines(tmp.lb, lty=3, col="green")
lines(tmp.ub, lty=3, col="green")
legend("topleft", lty=c(1,2,3), lwd=c(2,1,1), pch=c(NA,2,NA),
       col=c("black", "red", "green"),
       legend=c("Observed", "Predicted", "95% Prediction Limits"))
```



9.3 季节哑变量

另一种表示季节性的方法是用非随机的回归项表示固定的季节模式。这样的模式虽然也可以通过季节差分消除，但是与动态模型和非随机线性趋势模型的关系类似，固定的季节模型不应该用季节差分处理。

非随机的季节因素用回归哑变量表示。$s = 4$ 时，用 3 个哑变量就可以表示 4 个不同季节的固定水平。为了判断非随机季节模型是否使用，可以先拟合动态的季节 ARIMA$(1,0,1)(1,0,1)_s$ 模型，当发现其中的季节因素可以忽略时，就可考虑采用非随机的季节模型。下面举例说明。

例 9.1. 考虑 CRSP 最高 10 分位资产组合的月简单收益率，从 1970 年 1 月到 2008 年 12 月共 39 年，468 个观测。

读入数据：

```
da <- read_table(
  "m-deciles08.txt",
  col_types=cols(.default = col_double(),
                 date=col_date("%Y%m%d")))
xts.dec10 <- xts(da[["CAP1RET"]], ymd(da[["date"]]))
dec10 <- ts(da[["CAP1RET"]], start=c(1970, 1), frequency=12)
```

收益率序列图：

```
plot(dec10, main="CRSP 10 Percentile Index Return Rate",
     xlab="Year", ylab="")
```

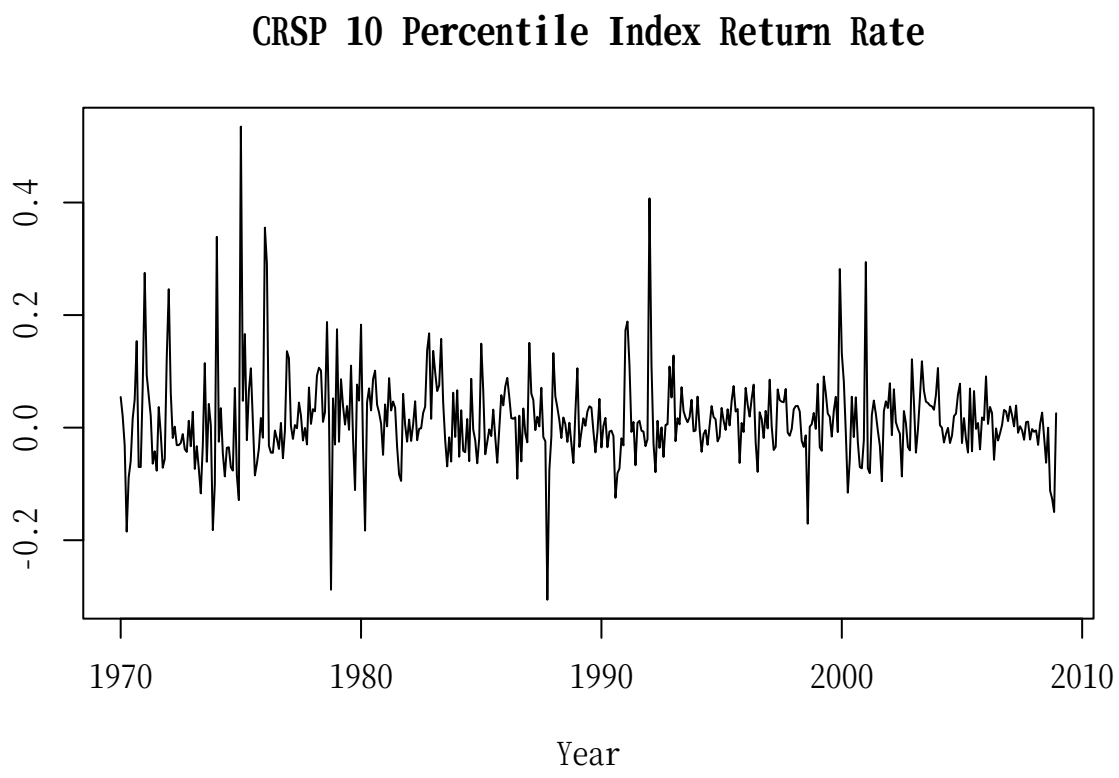


图 9.6: CRSP 10 分位指数的收益率

序列图本身没有表现出明显的趋势与周期性。取最近 3 年的数据来看：

```
plot(window(dec10, start=c(2006,1), end=c(2008,12)),
            main="CRSP 10 Percentile Index Return Rate",
     xlab="Year", ylab="")
```

也看不出周期性。

作 ACF 图：

```
forecast::Acf(dec10, main="", lag.max=36)
```

ACF 函数在 12 的倍数的滞后上显著不等于零，这体现出了周期性。来拟合 ARIMA$(1,0,1)(1,0,1)_{12}$:

```
resm1 <- arima(dec10, order=c(1,0,1),
               seasonal=list(order=c(1,0,1), period=12))
resm1
```

```
##
## Call:
```

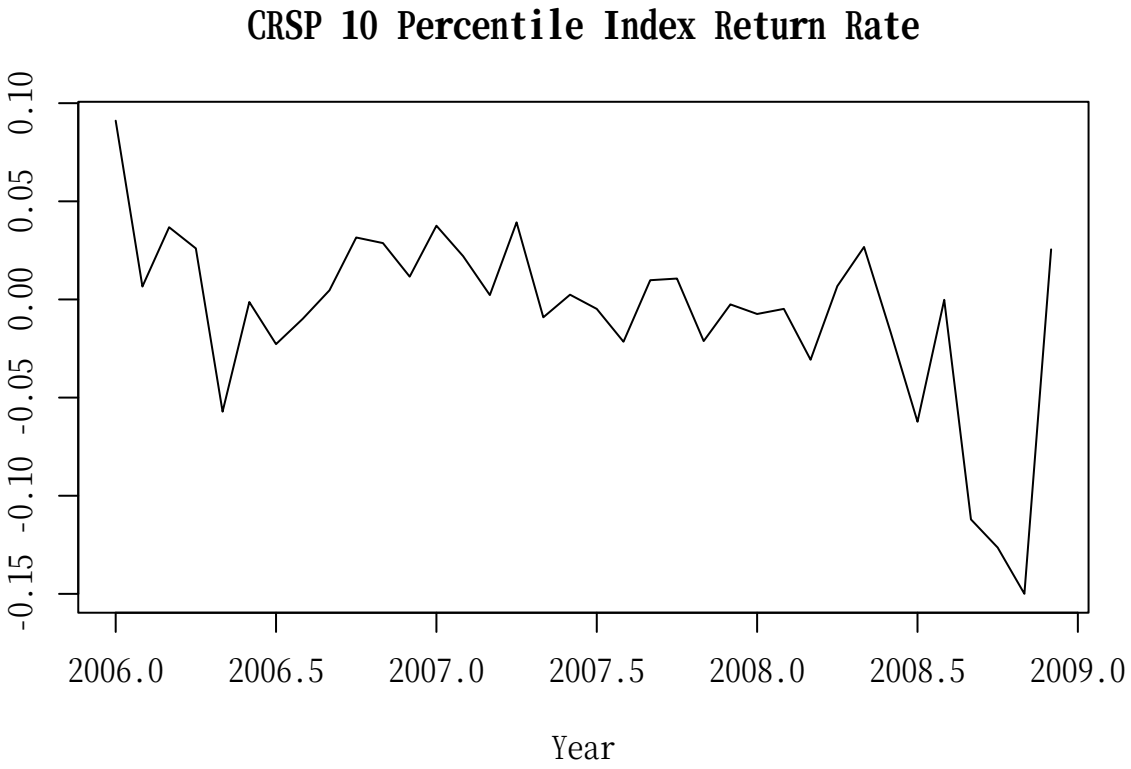


图 9.7: CRSP 10 分位指数的收益率 (近三年)

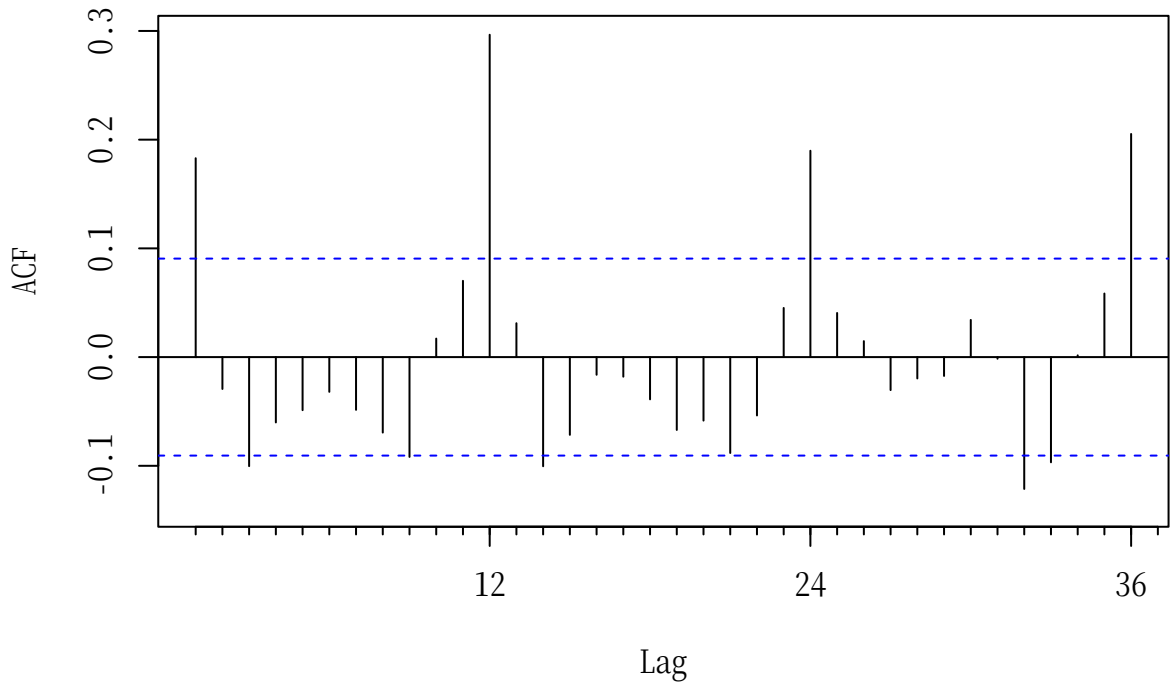


图 9.8: CRSP 10 分位指数收益率的 ACF

```
## arima(x = dec10, order = c(1, 0, 1), seasonal = list(order = c(1, 0, 1), period = 12))
##
## Coefficients:
##           ar1     ma1    sar1     sma1  intercept
##       -0.0639  0.2508  0.9882  -0.9142     0.0117
## s.e.   0.2205  0.2130  0.0092   0.0332     0.0125
##
## sigma^2 estimated as 0.004704:  log likelihood = 584.69,  aic = -1157.39
```

得到的模型可以写成

$$(1+0.0639B)(1-0.9882B^{12})(X_t-0.0117)=(1+0.2508B)(1-0.9142B^{12})\varepsilon_t$$

其中的 $1-0.9882B^{12}$ 项与 $1-0.9142B^{12}$ 近似可以消去，所以这个结果提示可能不需要使用动态季节模型。

采用非随机的哑变量模型来建模。为简单起见，仅考虑一月份的效应，因为实证发现一月份的收益率倾向为正值。

```
jan <- as.numeric(c(cycle(dec10))==1)
lm1 <- lm(c(dec10) ~ jan); summary(lm1)
```

```
##
## Call:
## lm(formula = c(dec10) ~ jan)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30861 -0.03475 -0.00176  0.03254  0.40671
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.002864   0.003333   0.859    0.391
## jan         0.125251   0.011546  10.848   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06904 on 466 degrees of freedom
## Multiple R-squared:  0.2016, Adjusted R-squared:  0.1999
## F-statistic: 117.7 on 1 and 466 DF,  p-value: < 2.2e-16
```

可以看出在以一月份为哑变量的线性回归中，自变量是显著的。拟合的模型为

$$\hat{x}_t=0.002864+0.1253\mathrm{Jan}_t$$

其中 Jan_t 当 t 的月份为一月份时等于 1，否则等于 0。

但是，这个模型是有缺陷的，因为线性回归假定随机误差项独立，而这里的随机误差项显然是有自相关的：

```
forecast::Acf(residuals(lm1), main="")
```

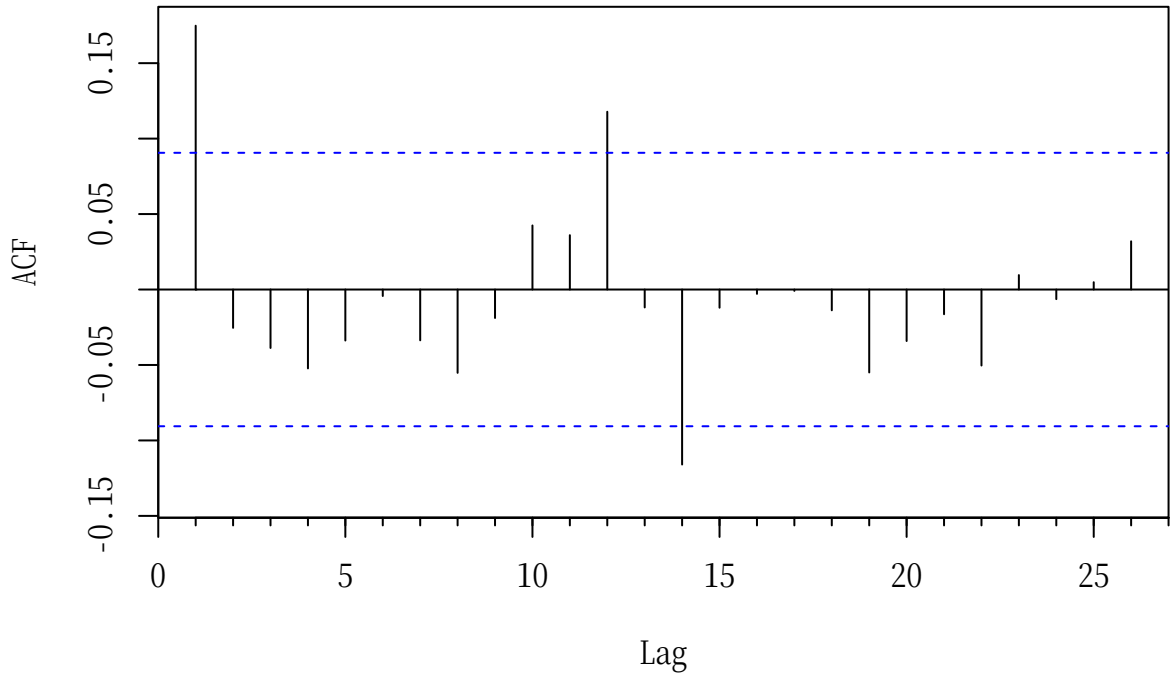


图 9.9: 固定季节项回归残差的 ACF

其滞后 1 还显著不等于 0。在滞后为 12 的倍数的位置已经不再显著。

Chapter 10

带时间序列误差的回归模型

10.1 方法示例

在统计学的数据分析中，线性回归分析是最常用的分析工具之一。线性回归以一元线性回归为例，模型如下

$$Y_t = \beta_0 + \beta_1 X_t + e_t,\ t = 1, 2, \ldots, T \tag{10.1}$$

其中自变量 $\{X_t\}$ 为常数列，β_0, β_1 为未知的系数，$\{e_t\}$ 为零均值独立同分布随机误差序列，方差为 σ_e^2，因变量 $\{Y_t\}$ 为随机变量列。参数 $\beta_0, \beta_1, \sigma_e^2$ 可以用最小二乘法估计，估计量无偏、相合，系数的估计渐近正态分布。为了对估计结果做假设检验或者用模型结果做预测，一般还假设 $\{e_t\}$ 为零均值独立同正态分布序列。

在金融研究中，$\{X_t\}$ 和 $\{Y_t\}$ 一般都是时间序列，而且 $\{e_t\}$ 也是时间序列，有序列相关性。这时，最小二乘估计不是最有效的估计甚至于可能不相合，基于 $\{e_t\}$ 独立同正态分布所做的标准误差估计、假设检验和预测都不再成立。

(Cochrane & Orcutt, 1949) 的文章指出，当 $\{e_t\}$ 彼此正相关时，回归系数的标准误差估计偏低，使得相应的 t 和 F 检验统计量的绝对值偏大。(Durbin & Watson, 1950) 和 (Durbin & Watson, 1951) 给出了一阶自相关的检验。Durbin-Watson 序列自相关检验使用检验统计量

$$\mathrm{DW} = \frac{\sum_{t=2}^{T}(\hat{e}_t - \hat{e}_{t-1})^2}{\sum_{t=1}^{T}\hat{e}_t^2} \approx 2(1 - \hat{\rho}_1)$$

其中 $\hat{e}_t$ 是回归残差，当无序列相关时 DW 统计量接近于 2。此统计量只用到一阶自相关。

当 $\{e_t\}$ 是平稳可逆 ARMA 序列时，可以将线性回归模型与平稳可逆 ARMA 序列同时估计，可以得到需要标准误差估计、假设检验和预测。R 的 `arima()` 函数提供了一个 `xreg=` 用来引入回归自变量。

如果不关心 $\{e_t\}$ 的具体模型，而只关心对回归系数的 SE 的正确估计以及假设检验的正确性，可以仅假设 $\{e_t\}$ 的协方差结构而不考虑 $\{e_t\}$ 的建模。这样的方法在混合线性模型中使用。

下面举例说明。考虑美国国债一年期与三年期利率，数据为周数据，从 1962-01-05 到 2009-10-04，共 2467 个观测。记一年期利率序列为 $\{x_{1t}\}$，三年期利率序列为 $\{x_{3t}\}$。

先读入数据：

```
da <- read_table(
  "w-gs1yr.txt",
  col_types=cols(.default=col_double()))
x1 <- da[["rate"]]
y1 <- xts(x1, make_date(da[["year"]], da[["mon"]], da[["day"]]))
da <- read_table2(
  "w-gs3yr.txt",
  col_types=cols(.default=col_double()))
```

```
## Warning: `read_table2()` was deprecated in readr 2.0.0.
## Please use `read_table()` instead.
```

```
x3 <- da[["rate"]]
y3 <- xts(x3, make_date(da[["year"]], da[["mon"]], da[["day"]]))
rm(da)
xts.gs <- cbind(gs1yr=y1, gs3yr=y3)
rm(y1, y3)
```

两个利率序列的时间序列图：

```
plot(xts.gs, type="l",
     main="US Federal Bonds 1 Year and 3 Years",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="auto",
     col=c("blue", "red"))
```

蓝色为一年期，红色为三年期。最后三年的图形：

```
plot(last(xts.gs, "3 years"), type="l",
     main="US Federal Bonds 1 Year and 3 Years",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="auto",
     col=c("blue", "red"))
```

这些数据作为时间序列呈现出不平稳的表现。对一年期利率作 ACF 估计：

```
forecast::Acf(x1, main="")
```

这样的 ACF 是有缓慢变化趋势的典型表现。

对一年期利率序列作单位根检验：

```
ar(diff(x1), method="mle")
```

```
##
## Call:
## ar(x = diff(x1), method = "mle")
##
## Coefficients:
```

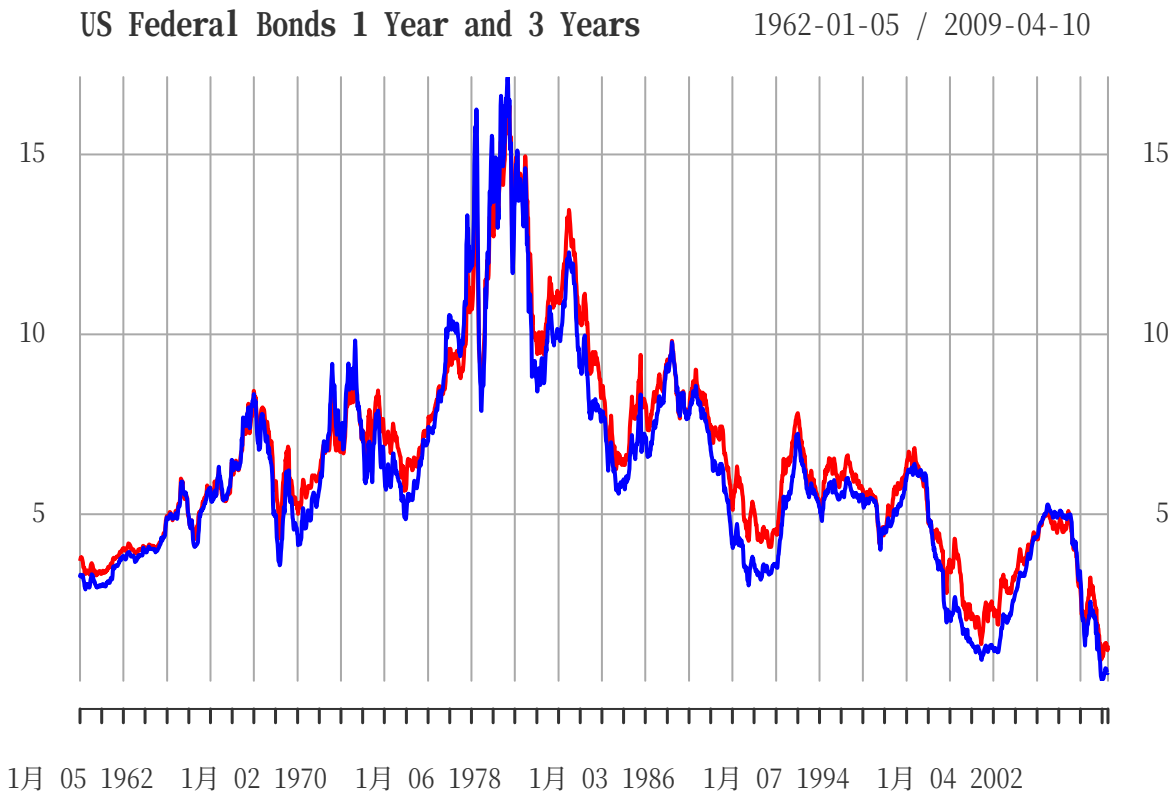


图 10.1: 美国一年期与三年期国债利率周数据

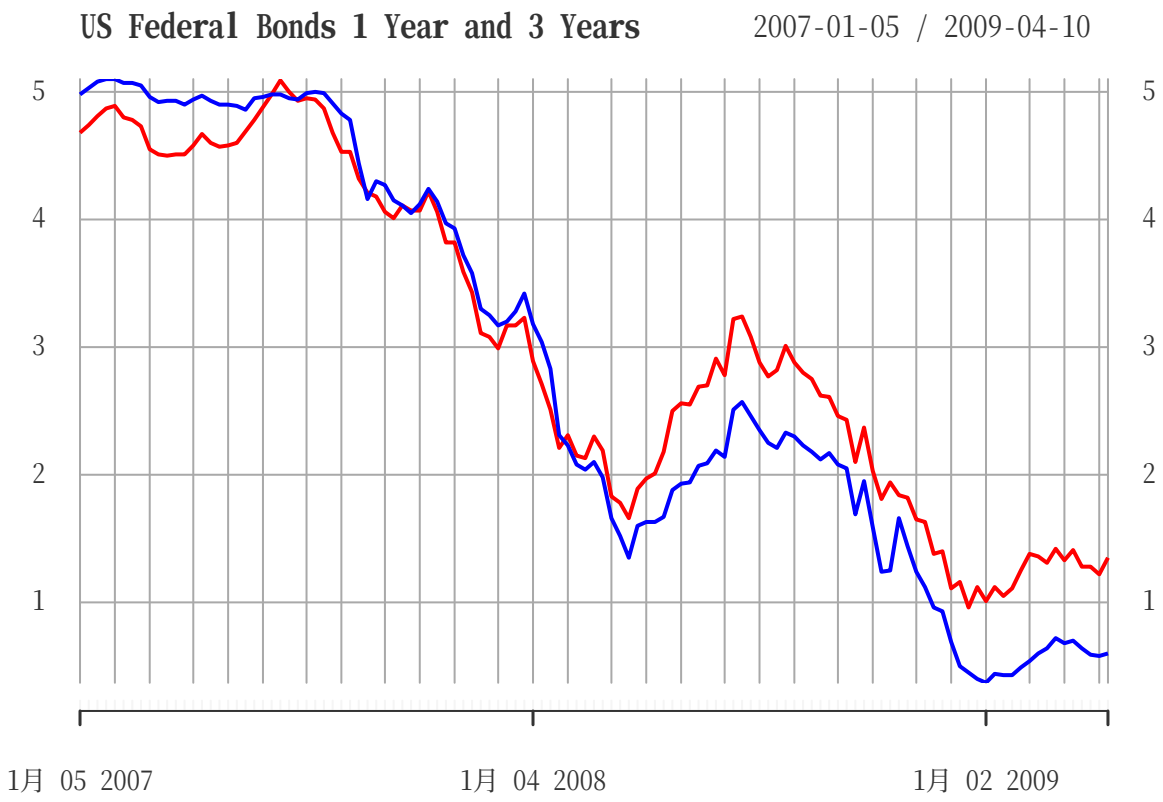


图 10.2: 美国一年期与三年期国债利率周数据

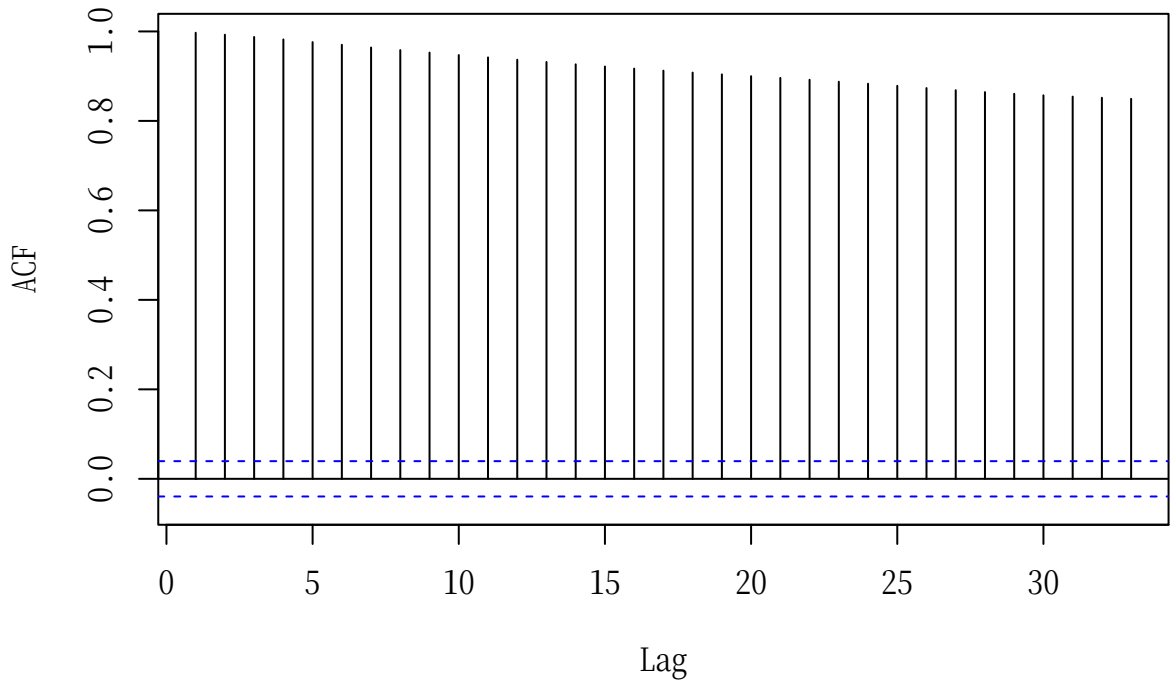


图 10.3: 一年期利率 ACF

```
##       1       2       3       4       5       6       7       8
##  0.3213  0.0372  0.0333  0.0577 -0.0093 -0.0008 -0.0813  0.0312
##       9
## -0.0469
##
## Order selected 9  sigma^2 estimated as  0.03103
```

```
fUnitRoots::adfTest(x1, lags=9, type="ct")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 9
##   STATISTIC:
##     Dickey-Fuller: -2.3813
##   P VALUE:
##     0.4169
##
## Description:
##  Wed Mar 23 16:19:14 2022 by user: Lenovo
```

单位根检验结果不显著，说明一年期利率序列是单位根过程。

如果以 x_{3t} 为因变量，以 x_{1t} 为自变量作线性回归：

$$x_{3t} = \beta_0 + \beta_1 x_{1t} + e_t$$

就会遇到 $\{e_t\}$ 序列相关的问题，甚至于 $\{e_t\}$ 可能有单位根造成方差线性增长，有异方差问题。

先用散点图考察两个序列的关系：

```
plot(x1, x3, type="p", pch=16, cex=0.5,
     xlab="1 Year", ylab="3 Years",
     main="US Federal Bonds 3 Years Rate Vs. 1 Year Rate")
```

散点图显示同期的一年期利率和三年期利率高度相关。

试拟合线性回归模型：

```
lm1 <- lm(x3 ~ x1); summary(lm1)
```

```
##
## Call:
## lm(formula = x3 ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

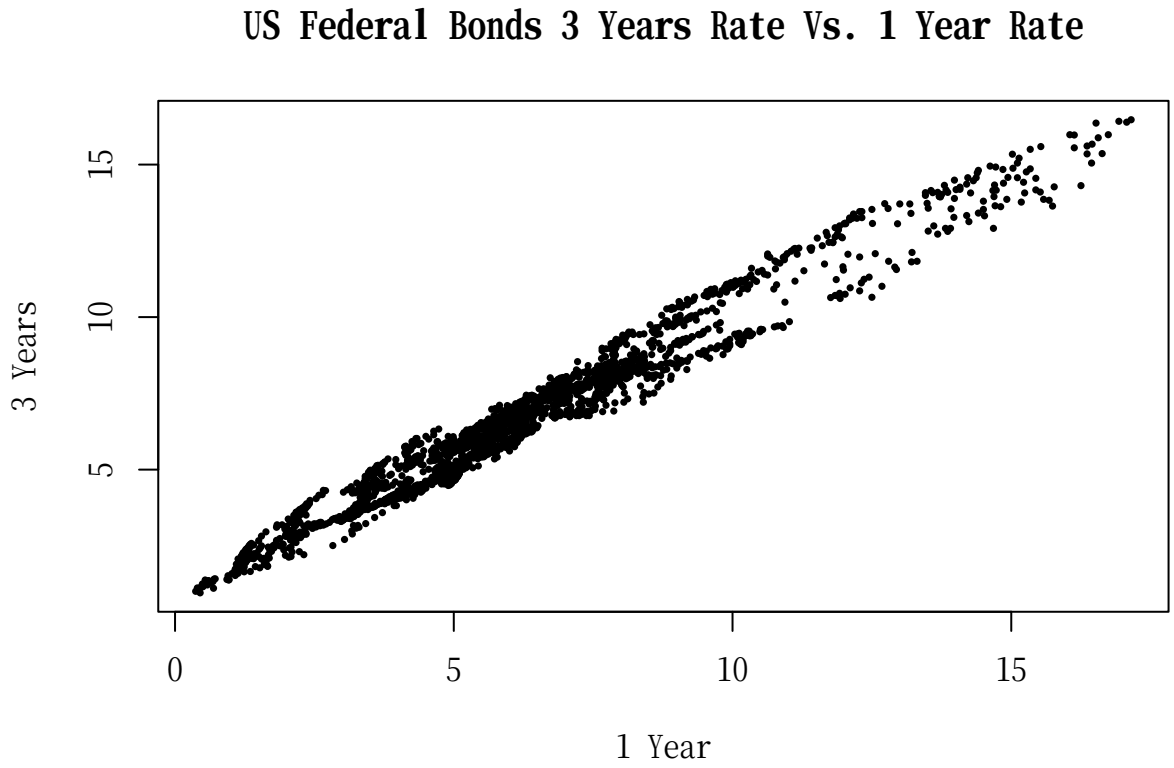


图 10.4: 美国一年期与三年期国债利率周数据散点图

```
## -1.82319 -0.37691 -0.01462  0.38661  1.35679
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.83214    0.02417   34.43   <2e-16 ***
## x1           0.92955    0.00357  260.40   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5228 on 2465 degrees of freedom
## Multiple R-squared:  0.9649, Adjusted R-squared:  0.9649
## F-statistic: 6.781e+04 on 1 and 2465 DF,  p-value: < 2.2e-16
```

注意：不懂统计的人会盲目相信统计软件给出的如此专业的输出结果。$R^2 = 0.9649$ 是一个十分令人欣喜的回归结果。但是，只有学习了相应的统计模型的有关知识并且知道模型的局限，才能正确地运用统计模型。上面的系数估计、σ_e^2 估计、SE 估计、t 检验等等都是基于随机误差项独立同分布假定的。如果假定不成立，σ_e^2 的估计会严重低估，SE 估计不相合，检验不准确。在经济和金融数据分析中一定要对回归模型结果进行诊断分析，最常用的是残差分析，其中一项是残差序列相关性的检验。

传统的残差序列相关性检验有 Durbin-Watson 检验，在时间序列分析软件的支持下，我们可以做 ACF 图，进行 Ljung-Box 白噪声检验。

```
forecast::Acf(lm1$residuals, main="")
```

残差的 ACF 表明结果仍有很强的自相关，甚至于可能有单位根。白噪声检验:

```
Box.test(lm1$residuals, lag=9)
```

```
##
##  Box-Pierce test
##
## data:  lm1$residuals
## X-squared = 19303, df = 9, p-value < 2.2e-16
```

结果显著，拒绝残差序列为白噪声的零假设。

对残差序列作单位根检验：

```
ar(diff(lm1$residuals), method="mle")
```

```
##
## Call:
## ar(x = diff(lm1$residuals), method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.2072  -0.0030  -0.0100   0.0911  -0.0641  -0.0642  -0.0475   0.0193
##       9       10       11       12
##  0.0163  -0.0509  -0.0174   0.0706
```

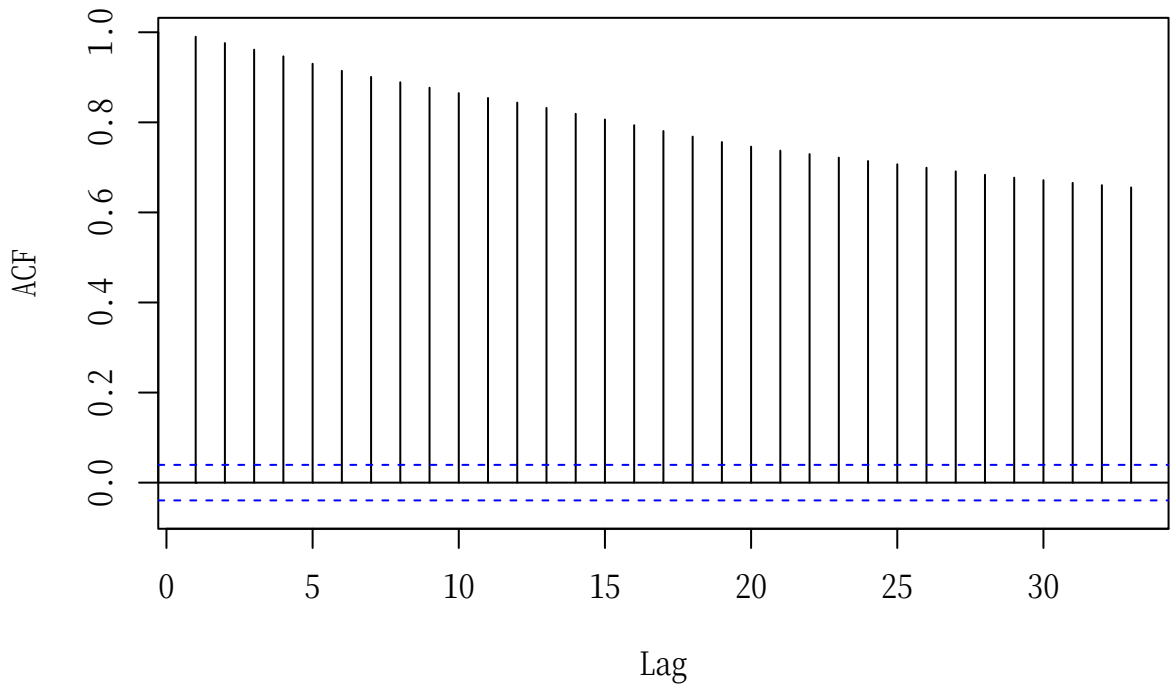


图 10.5: 三年期对一年期利率回归残差的 ACF

```
##
## Order selected 12  sigma^2 estimated as  0.005036
```

```
fUnitRoots::adfTest(lm1$residuals, lags=12, type="nc")
```

```
## Warning in fUnitRoots::adfTest(lm1$residuals, lags = 12, type = "nc"): p-value
## smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 12
##   STATISTIC:
##     Dickey-Fuller: -4.1201
##   P VALUE:
##     0.01
##
## Description:
##  Wed Mar 23 16:19:15 2022 by user: Lenovo
```

结果显著，拒绝残差有单位根的零假设。尽管如此，鉴于残差的高度自相关，还是不能轻易使用线性回归。

如果回归模型 $y_t = \beta_0 + \beta_1 x_t + e_t$ 中 y_t 和 x_t 都是单位根过程而 e_t 是线性时间序列，则称 $\{y_t\}$ 与 $\{x_t\}$ 序列存在协整关系 (cointegrated)。在目前的数据中，因为利率序列有单位根，残差虽然没有单位根但是 ACF 衰减很慢，不符合协整的要求。

为此，对利率序列作一阶差分，考虑利率差分之间的关系。分别记差分后的两个序列为 $\{c_{1t}\}$ 和 $\{c_{3t}\}$。

```
c1 <- diff(x1)
c3 <- diff(x3)
```

一年期利率的差分序列的 ACF:

```
forecast::Acf(c1, main="")
```

这个 ACF 的表现已经不是单位根过程，符合线性时间序列的 ACF 的表现。

一年期与三年期的两个差分序列的散点图：

```
plot(c1, c3, type="p", pch=16, cex=0.5,
     xlab="1 Year", ylab="3 Years",
     main="US Federal Bonds 3 Years Rate Diff1 Vs. 1 Year Rate Diff1")
```

两者仍有强烈的线性相关。作线性回归：

```
lm2 <- lm(c3 ~ c1); summary(lm2)
```

```
##
## Call:
```

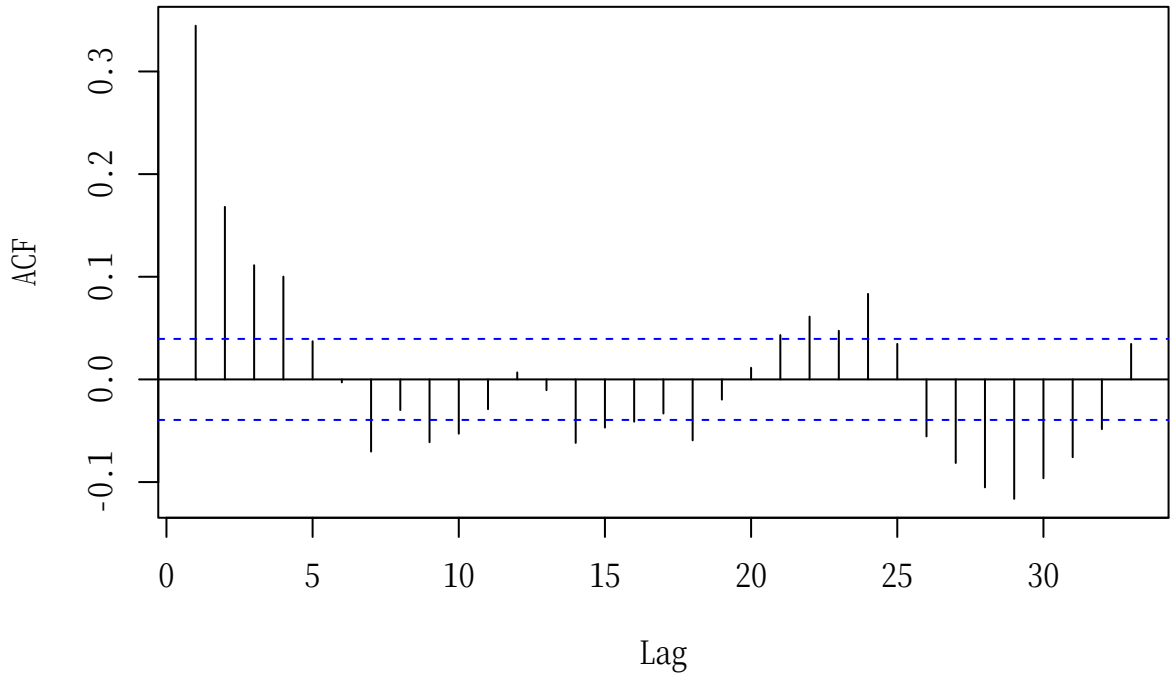


图 10.6: 一年期利率差分序列的 ACF

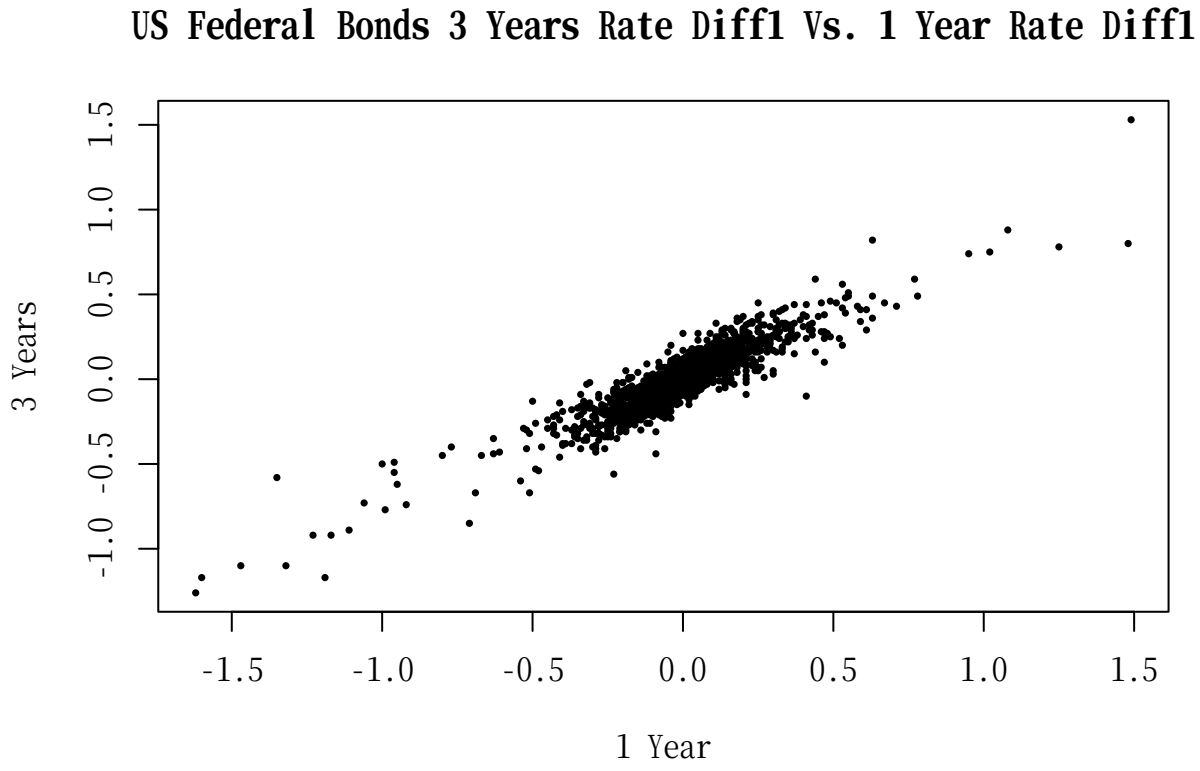


图 10.7: 一年期利率差分序列的 ACF

```
## lm(formula = c3 ~ c1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42459 -0.03578 -0.00117  0.03467  0.48921
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0001051  0.0013890  -0.076     0.94
## c1           0.7919323  0.0073391 107.906   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06897 on 2464 degrees of freedom
## Multiple R-squared:  0.8253, Adjusted R-squared:  0.8253
## F-statistic: 1.164e+04 on 1 and 2464 DF,  p-value: < 2.2e-16
```

结果表面上也很好，但仍需作序列相关性诊断分析。残差的 ACF：

```
forecast::Acf(lm2$residuals, main="")
```

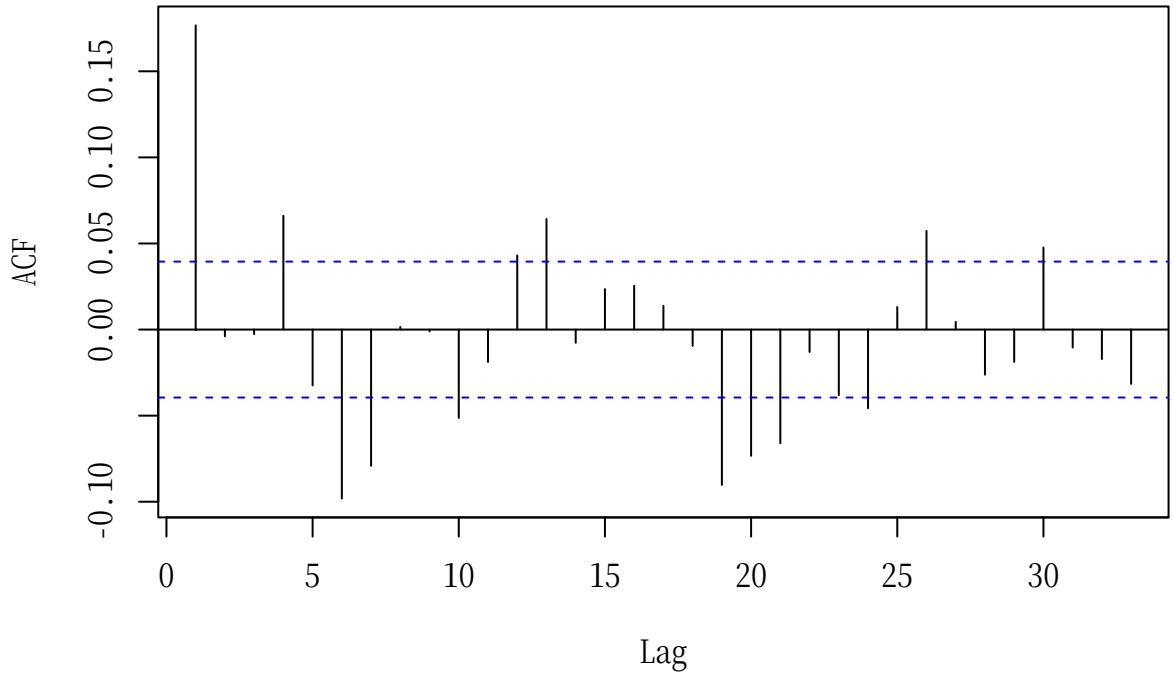


图 10.8: 差分之间回归残差的 ACF

这个 ACF 表现得与线性时间序列 ACF 相像，但是滞后 1 处仍显著不等于零。作 Ljung-Box 白噪声检验：

```
Box.test(lm2$residuals, lag=9)
```

```
##
##  Box-Pierce test
##
## data:  lm2$residuals
## X-squared = 129.57, df = 9, p-value < 2.2e-16
```

结果显著，说明回归模型中的误差项存在序列自相关，这样，上面的两个差分序列的回归结果中仅两个回归系数估计是可信的，误差方差估计、SE 估计、假设检验结果均不可用。

因为上面的回归残差的 ACF 主要在滞后 1 显著，其它位置虽然也有显著但值不大，所以考虑 e_t 用 MA(1)，回归变成

$$c_{3t} = \beta_1 c_{1t} + e_t, \quad e_t = \varepsilon_t + \theta\varepsilon_{t-1}$$

其中 $\{\varepsilon_t\}$ 为零均值独立同分布白噪声列。这里没有常数项是因为两个序列都是差分过的，如果原来的关系有常数项会在差分过程中被消去。

R 的 `arima()` 函数提供了将外生回归自变量与 AMRA 模型一同估计的功能。用 `xreg=` 指定外生回归自变量。

```
mres <- arima(c3, xreg=c1, order=c(0,0,1), include.mean=FALSE)
mres
```

```
##
## Call:
## arima(x = c3, order = c(0, 0, 1), xreg = c1, include.mean = FALSE)
##
## Coefficients:
##          ma1      c1
##       0.1823  0.7936
## s.e.  0.0196  0.0075
##
## sigma^2 estimated as 0.0046:  log likelihood = 3136.62,  aic = -6267.23
```

这拟合了如下的带有 MA(1) 误差的无截距项的一元线性回归模型：

$$c_{3t} = 0.7936 c_{1t} + e_t, \quad e_t = \varepsilon_t + 0.1823\varepsilon_{t-1}, \ \hat{\sigma}_\varepsilon^2 = 0.0046$$

从 $\hat{\beta}_1$ 和 $\hat{\theta}$ 的 SE 估计来看，两个系数都是显著不为零的。

小结：带时间序列误差的线性回归模型建模步骤

1. 拟合一个线性回归模型，并检验残差的序列相关性
2. 如果残差序列是单位根非平稳的，则对因变量和自变量都做一阶差分。然后对差分后的序列再进行第一步。如果残差序列是平稳的，则对残差序列识别一个 ARMA 模型，并相应地修改线性回归模型。

3. 用最大似然估计法对回归模型与 ARMA 模型进行联合估计，并对模型进行检验，看是否需要改进。主要可以使用 Ljung-Box 对残差进行白噪声检验。

10.2 arima 函数说明

R 中的 `stats::arima()` 函数：

- 支持平稳可逆 ARMA 建模，这时可以用 `include.mean=TRUE` 设定有均值参数。
- 支持 ARIMA 建模，如果差分阶大于零，则不允许 `include.mean=TRUE`，即单位根过程不允许带有漂移。
- 支持带有季节项的 ARIMA 建模。季节差分阶大于零时也不允许有漂移。
- 支持以平稳 ARMA 序列为误差项的回归建模。

平稳可逆 ARMA 模型，表示为 ARIMA$(p,0,q)(P,0,Q)_s$，模型公式为：

$$(1-\phi_1 B-\phi_2 B^2-\cdots-\phi_p B^p)(1-\Phi_1 B^s-\Phi_2 B^{2s}-\cdots-\Phi_P B^{Ps})(Y_t-\mu) \tag{10.2}$$

$$=(1+\theta_1 B+\theta_2 B^2+\cdots+\theta_q B^q)(1+\Theta_1 B^s+\Theta_2 B^{2s}+\cdots+\Theta_P B^{Qs})\varepsilon_t, \tag{10.3}$$

其中 s 是周期，μ 是 $\{Y_t\}$ 的期望值。

对于 ARIMA$(p,d,q)(P,D,Q)_s$ 模型 $\{Z_t\}$，则令 $Y_t=(1-B)^d(1-B^s)^D Z_t$，Y_t 服从(10.3)中 $\mu=0$ 时的模型。

如果 ARIMA$(p,0,q)(P,0,Q)_s$ 模型 $\{Z_t\}$ 有 `xreg` 选项，设输入的自变量或自变量矩阵（每列是一个自变量）的各个观测 x_t(标量或行向量)，则模型为

$$Z_t=\beta_0+x_t\beta+Y_t, \tag{10.4}$$

其中 $\{Y_t\}$ 服从(10.3)中 $\mu=0$ 时的模型。我们用模拟数据来进行这种情形的验证计算。

```
sima <- function(){
  # 样本量
  n <- 500
  # 自变量
  x <- runif(n, 10, 20)
  # 作为误差项的 ARMA
  y <- arima.sim(model=list(ar=0.3, ma=0.7), n=n)
  # 回归参数
  a <- 100; b <- 2
  # 输出的因变量
  z <- a + b*x + y
  cat("Z 的样本均值 =", mean(z), "\n")

  # 估计:
  res <- arima(z, order=c(1,0,1), xreg=x)
  print(res)
}
set.seed(101)
sima()
```

```
## Z的样本均值 = 129.9459
##
## Call:
## arima(x = z, order = c(1, 0, 1), xreg = x)
##
## Coefficients:
##          ar1     ma1  intercept       x
##       0.2927  0.7127    99.8338  1.9994
## s.e.  0.0511  0.0375     0.1643  0.0086
##
## sigma^2 estimated as 0.8742:  log likelihood = -676.45,  aic = 1362.89
```

上述结果中 intercept 是(10.4)中回归截距项 β_0 的估计而不再是 Z_t 的均值 μ 的估计。

如果包含回归自变量 $\{X_t\}$ 时因变量 $\{Z_t\}$ 有单位根，这时最好先将两者都预先差分后再建模，ARIMA 不会自动建立关于差分之后的自变量和因变量的回归模型，应使用模型

$$\Delta Z_t = a + b\Delta X_t + Y_t, \tag{10.5}$$

其中 $\{Y_t\}$ 服从平稳的 ARMA 模型，ΔZ_t 和 ΔX_t 预先计算。如果真实模型是(10.5)但在 arima() 函数中直接输入 Z_t 和 X_t 且指定差分阶 1，则模型估计结果将受到单位根影响给出错误结果。模拟验证如下：

```
simb <- function(){
  # 样本量
  n <- 500
  # 自变量
  dx <- runif(n, 10, 20)
  x <- cumsum(dx)
  # 作为误差项的 ARMA
  dy <- arima.sim(model=list(
    ar=0.3, ma=0.7), n=n)
  # 回归参数
  a <- 100; b <- 2
  # 输出的因变量
  dz <- a + b*dx + dy
  z <- cumsum(dz)

  # 估计，不预先差分：
  res <- arima(z, order=c(1,1,1), xreg=x)
  cat(" 不预先差分的错误结果：\n")
  print(res)

  # 估计，预先差分：
  res2 <- arima(dz, order=c(1,0,1), xreg=dx)
  cat("\n\n预先差分的正确结果：\n")
  print(res2)
```

```
}
set.seed(101)
simb()
```

```
## 不预先差分的错误结果：
##
## Call:
## arima(x = z, order = c(1, 1, 1), xreg = x)
##
## Coefficients:
##          ar1     ma1       x
##       0.9999  0.4827  1.9977
## s.e.  0.0002  0.0605  0.0094
##
## sigma^2 estimated as 1.29:  log likelihood = -776.24,  aic = 1560.49
##
##
## 预先差分的正确结果：
##
## Call:
## arima(x = dz, order = c(1, 0, 1), xreg = dx)
##
## Coefficients:
##          ar1     ma1  intercept      dx
##       0.2927  0.7127    99.8338  1.9994
## s.e.  0.0511  0.0375     0.1643  0.0086
##
## sigma^2 estimated as 0.8742:  log likelihood = -676.45,  aic = 1362.89
```

预先差分的缺点是 `predict()` 函数给出的预测也是关于 ΔZ_t 的，需要人为使用 $Z_{t+1} = Z_t + \Delta Z_t$ 进行转换。

如果 Z_t 和 X_t 包含单位根，而误差项 Y_t 为平稳的 ARMA，这时

$$Z_t = \beta_0 + \beta X_t + Y_t,$$

称 Z_t 和 X_t 有协整关系，arima 函数能正确给出估计，模拟验证如下：

```
simc <- function(){
  # 样本量
  n <- 500
  # 自变量
  dx <- runif(n, 10, 20)
  x <- cumsum(dx)
  # 作为误差项的 ARMA
  dy <- arima.sim(model=list(
    ar=0.3, ma=0.7), n=n)
```

```
  # 回归参数
  a <- 100; b <- 2
  # 输出的因变量
  z <- a + b*x + dy

  # 估计，不预先差分：
  res <- arima(z, order=c(1,0,1), xreg=x)
  cat(" 协整情况下的估计：\n")
  print(res)
}
set.seed(101)
simc()
```

```
## 协整情况下的估计：
##
## Call:
## arima(x = z, order = c(1, 0, 1), xreg = x)
##
## Coefficients:
##          ar1     ma1  intercept      x
##       0.2926  0.7128    99.7804  2e+00
## s.e.  0.0511  0.0374     0.2026  1e-04
##
## sigma^2 estimated as 0.8741:  log likelihood = -676.42,  aic = 1362.83
```

如果 X_t 与 Z_t 都有单位根且误差项也有单位根，这样的模型没有合理的定义，应使用预先差分的做法。

总之，在 `arima()` 函数中用 `xreg=` 引入回归自变量（外生变量）时，不要指定差分或者季节差分。

forecast 包的 `Arima()` 函数能实现 `stats::arima()` 的功能，但在差分阶大于零时允许有漂移项。forecast 包的 `auto.arima()` 函数可以自动定阶。

Chapter 11

长记忆模型

11.1 长记忆模型介绍

ACF 是时间序列建模的重要参考。对于 ARMA 序列，当滞后 $k \to \infty$ 时其样本 ACF 是负指数速度趋于零的。对于单位根非平稳列，其理论 ACF 无定义（因为自协方差是针对弱平稳列定义的），其样本 ACF 在样本量 $T \to \infty$ 时每个 $\hat{\rho}_k$ 都趋于 1 （$k > 0$）。

有一些平稳时间序列的 ACF 虽然也随滞后 $k \to \infty$ 趋于零，但是收敛到零的速度比较慢，只有负幂次 $k^{-\alpha}$ 这样的速度。这代表着序列的自相关性随着距离变远而减小得比较慢，称这样的序列是**长记忆**时间序列。注意，长记忆时间序列仍是弱平稳的，单位根非平稳列虽然远距离的自相关性很强但不称为长记忆。

长记忆时间序列的典型模型是分数差分弱平稳列，模型为

$$(1-B)^d X_t = \xi_t,\ -0.5 < d < 0.5 \tag{11.1}$$

其中 $\{\xi_t\}$ 是零均值独立同分布白噪声。

类似于 ARMA 序列平稳解的讨论，对算子多项式使用复变多项式的泰勒展开来求解。函数 $f(z) = (1-z)^{-d}$ 在 $z = 0$ 有如下的泰勒展开：

$$(1-z)^{-d} = \sum_{k=0}^{\infty} \psi_k z^k$$

其中

$$\psi_0 = 1,\ \psi_k = \frac{d(d+1)\ldots(d+k-1)}{k!}$$

类似地有

$$(1-z)^{d} = \sum_{k=0}^{\infty} \pi_k z^k$$

其中

$$\pi_0 = 1, \quad \pi_k = (-1)^k \frac{d(d-1)\ldots(d-k+1)}{k!}$$

若对实数 d 推广记号

$$\binom{d}{k}=\frac{d(d-1)(d-2)...(d-k+1)}{k!}$$

则有

$$\begin{aligned}(1-z)^{-d}=&\sum_{j=0}^{\infty}\binom{-d}{k}(-z)^k=\sum_{k=0}^{\infty}\frac{(-d)(-d-1)...(-d-k+1)}{k!}(-z)^k\\=&\sum_{k=0}^{\infty}\frac{d(d+1)...(d+k-1)}{k!}(+z)^k=\sum_{k=0}^{\infty}\psi_k z^k\end{aligned}$$

也有

$$(1-z)^d=\sum_{j=0}^{\infty}\binom{d}{k}(-z)^k=\sum_{k=0}^{\infty}(-1)^k\frac{d(d-1)...(d-k+1)}{k!}z^k=\sum_{k=0}^{\infty}\pi_k z^k$$

11.2 长记忆模型性质

模型(11.1)有如下性质：

11.2.1 MA 表示

若 $d<0.5$，则模型中的 $\{X_t\}$ 有弱平稳列解，并有无穷阶 MA 表示：

$$x_t=\sum_{k=0}^{\infty}\psi_k\xi_{t-k}$$

11.2.2 AR 表示

若 $d>-0.5$，则模型中的 $\{\xi_t\}$ 可以用 $\{X_t\}$ 表示为如下的无穷阶 AR 形式：

$$X_t+\sum_{k=1}^{\infty}\pi_k X_{t-k}=\xi_t$$

11.2.3 ACF 衰减速率

若 $-0.5<d<0.5$，则平稳解 $\{X_t\}$ 的 ACF 为

$$\rho_k=\frac{d(d+1)...(d+k-1)}{(-d+1)(-d+2)...(-d+k)},k=1,2,...$$

$\rho_1=\frac{d}{1-d}$，当 $k\to\infty$ 时

$$\rho_k=ck^{2d-1}+o(k^{2d-1})$$

是以负幂律缓慢衰减的 (c 为与 k 无关的常数)。注意 ARMA 序列的 ρ_k 以负指数速度快速衰减。

11.2.4 PACF

若 $-0.5<d<0.5$，则平稳解 $\{X_t\}$ 的 PACF 为

$$\phi_{kk}=\frac{d}{k-d},\ k=1,2,...$$

以 k^{-1} 速度衰减。

11.2.5 谱密度性质

弱平稳时间序列的谱密度 $f(\omega)$, $\omega \in [0,\pi]$ 是序列在不同的随机振动频率上的能量分布，是一个非负可积函数。若 $-0.5 < d < 0.5$，若平稳解的谱密度满足

$$f(\omega) \sim \omega^{-2d},\ \omega \to 0$$

这样，则 $0 < d < 0.5$ 时谱密度在 $\omega = 0$ 频率附近趋于正无穷，这在实际数据中的表现是数据存在缓慢的长期水平波动。ARAM 序列的谱密度是有界的。

如果将模型(11.1)中的白噪声 $\{\xi_t\}$ 推广为可以取 ARMA(p,q) 序列，这样的模型称为 ARFIMA(p,d,q) 模型，称为分数阶差分 ARMA 模型。参见 (Geweke & Porter-Hudak, 1983)。

R 的 fracdiff 软件包用来构建分数阶差分 ARMA 模型。

在金融时间序列建模中，如果样本 ACF 数值不大但是衰减特别缓慢，可以考虑长记忆模型。如果数值很大同时衰减慢则可能是单位根非平稳，或者具有很接近 1 的特征根的 ARMA 序列。

11.3 长记忆模型建模实例

例 11.1. 对 CRSP 价值加权指数和等权指数的日简单收益率数据，时间从 1970-01-02 到 2008-12-31，考虑日收益率的绝对值。

读入数据，计算日收益率的绝对值：

```
da <- read_table(
  "d-ibm3dx7008.txt",
  col_types=cols(.default=col_double(),
                 Date=col_date("%Y%m%d"))
)
xts.crspw <- xts(da[,-1], da$Date)
vw <- abs(da$vwretd)
ew <- abs(da$ewretd)
rm(da)
```

价值加权日收益率绝对值序列的 ACF:

```
forecast::Acf(vw, main="", lag.max=300)
```

等加权日收益率绝对值序列的 ACF:

```
forecast::Acf(ew, main="", lag.max=300)
```

这两个序列的 ACF 都比较小但是衰减缓慢，到之后 300 时仍显著不为零。都是正相关。

函数 `fracdiff::fdGPH()` 计算差分阶的 Geweke-Porter-Hudak 估计值：

```
fracdiff::fdGPH(vw)
```

```
## $d
## [1] 0.372226
##
```

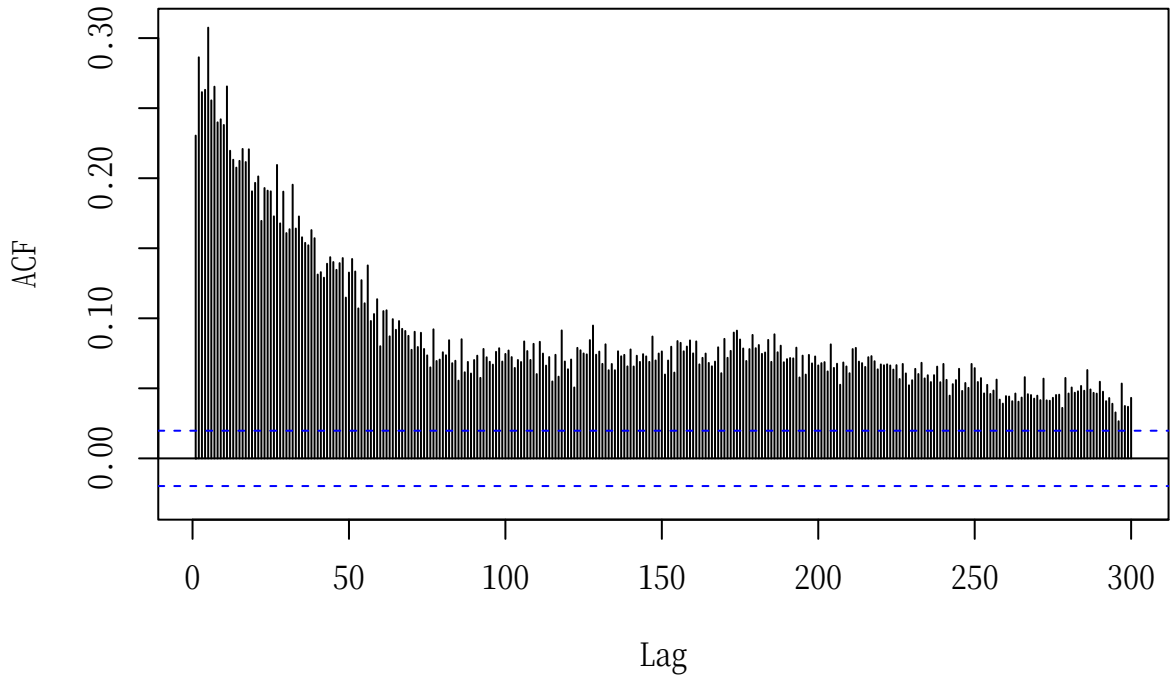


图 11.1: 价值加权日收益率绝对值序列的 ACF

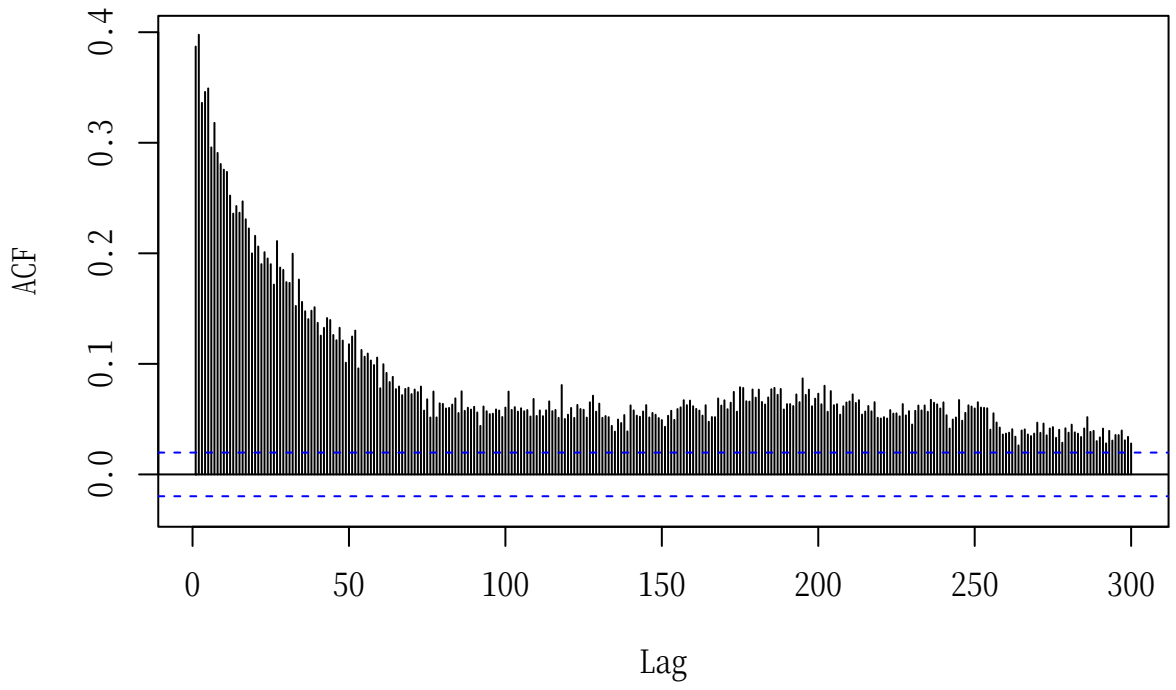


图 11.2: 等加权日收益率绝对值序列的 ACF

```
## $sd.as
## [1] 0.0698385
##
## $sd.reg
## [1] 0.06868857
```

估计的差分阶为 $d = 0.372$。

用 `fracdiff::fracdiff()` 函数进行 ARFIMA 模型估计：

```
mres <- fracdiff::fracdiff(vw, nar=1, nma=1)
summary(mres)
```

```
##
## Call:
##   fracdiff::fracdiff(x = vw, nar = 1, nma = 1)
##
## Coefficients:
##    Estimate Std. Error z value Pr(>|z|)
## d  0.490938   0.007997   61.39   <2e-16 ***
## ar 0.113389   0.005988   18.94   <2e-16 ***
## ma 0.575895   0.005946   96.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## sigma[eps] = 0.0065619
## [d.tol = 0.0001221, M = 100, h = 0.0003742]
## Log likelihood: 3.551e+04 ==> AIC = -71021.02 [4 deg.freedom]
```

其中选项 `nar` 和 `nma` 用来指定 AR 阶和 MA 阶。模型为

$$(1 - 0.1133B)(1 - B)^{0.4909} X_t = \varepsilon_t - 0.5759\varepsilon_{t-1},\ \sigma_\varepsilon = 0.006562$$

估计结果中差分阶 $d = 0.4909$ 已经接近非平稳的边缘 0.5 了。

注意：`fracdiff::fracdiff()` 的输出与 `arima()` 输出有差别，其 MA 系数的输出是 $1 - \theta_1 B - \cdots - \theta_q B^q$ 格式的。

Chapter 12

模型比较和平均

在实际金融事件序列数据的建模中，注意不存在所谓“正确的模型”，只能是从多个比较适合的模型中选择最合适的一个，或者将比较适合的多个模型的预测结果进行平均。

为了比较模型，有样本内比较和样本外比较两种方法。

12.1 样本内比较

如果建模目的是获得描述数据内在运动规律的参数模型，可以用样本内比较，利用全部数据建模，并比较不同模型的某个优良性指标，如 AIC、BIC、新息方差等。这些指标越小就认为模型越合适。

例 12.1. 考虑 CRSP 最高 10 分位资产组合的月简单收益率的不同模型，从 1970 年 1 月到 2008 年 12 月共 39 年，468 个观测。

读入数据：

```
da <- read_table(
  "m-deciles08.txt",
  col_types=cols(.default = col_double(),
                 date=col_date("%Y%m%d")))
xts.dec10 <- xts(da[["CAP1RET"]], ymd(da[["date"]]))
dec10 <- ts(da[["CAP1RET"]], start=c(1970, 1), frequency=12)
```

首先拟合 $\text{ARIMA}(1,0,1)(1,0,1)_{12}$:

```
resm1 <- arima(dec10, order=c(1,0,1),
               seasonal=list(order=c(1,0,1), period=12))
resm1
```

```
##
## Call:
## arima(x = dec10, order = c(1, 0, 1), seasonal = list(order = c(1, 0, 1), period = 12))
##
```

```
## Coefficients:
##           ar1     ma1    sar1     sma1  intercept
##       -0.0639  0.2508  0.9882  -0.9142     0.0117
## s.e.   0.2205  0.2130  0.0092   0.0332     0.0125
##
## sigma^2 estimated as 0.004704:  log likelihood = 584.69,  aic = -1157.39
```

其次，以是否一月份为哑变量，做一个哑变量回归模型，回归残差不继续建模：

```
jan <- as.numeric(c(cycle(dec10))==1)
lm1 <- lm(c(dec10) ~ jan); summary(lm1)
```

```
##
## Call:
## lm(formula = c(dec10) ~ jan)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30861 -0.03475 -0.00176  0.03254  0.40671
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.002864   0.003333   0.859    0.391
## jan         0.125251   0.011546  10.848   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06904 on 466 degrees of freedom
## Multiple R-squared:  0.2016, Adjusted R-squared:  0.1999
## F-statistic: 117.7 on 1 and 466 DF,  p-value: < 2.2e-16
```

最后，用带时间序列误差的回归，回归自变量是一月份效应哑变量：

```
jan <- as.numeric(c(cycle(dec10))==1)
resm2 <- arima(
  dec10, xreg=jan, seasonal=list(order=c(1,0,1), period=12))
resm2
```

```
##
## Call:
## arima(x = dec10, seasonal = list(order = c(1, 0, 1), period = 12), xreg = jan)
##
## Coefficients:
##          sar1    sma1  intercept     jan
##       -0.0920  0.2192     0.0027  0.1248
## s.e.   0.3585  0.3502     0.0037  0.0127
##
```

```
## sigma^2 estimated as 0.004671:  log likelihood = 591.56,  aic = -1173.12
```

三个模型的 σ_ε 估计分别为：

```
sqrt(0.004704)
```

```
## [1] 0.06858571
```

```
0.06904
```

```
## [1] 0.06904
```

```
sqrt(0.004671)
```

```
## [1] 0.06834471
```

差距很小，第一个模型和第三个模型的差距更小。从描述数据规律的角度三个模型都可接受。

12.2 样本外比较

如果建模的主要目的是预测，则应以预测误差小为比较的原则。这时，应该用前面的数据作为建模样本，后面的样本不参加建模，用模型对后面的样本进行预测并评估预测精度。这种模型比较的方法叫做回测检验 (backtesting)。

以超前一步预测问题为例。设有 $t = 1, 2, \dots, T$ 的样本，取 h 使得 h 和 $T - h$ 都比较大。回测检验步骤如下：

1. 对每个 $t = h, h+1, \dots, T-1$ 重复如下操作：用 $x_1, \dots, x_t$ 作为样本建立模型，用建立的模型预测 x_{t+1}，计算超前一步预测误差 $e_t(1)$。
2. 计算预测误差的均方平方根 (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{T-h}\sum_{t=h}^{T-1}[e_t(1)]^2}$$

3. 比较不同模型的 RMSE，以最小的一个为最优。

在建模时一般不对每一步都重新定阶。

其它的比较指标还有 MAE（平均绝对误差）

$$\text{MAE} = \frac{1}{T-h}\sum_{t=h}^{T-1}|e_t(1)|$$

以及偏差

$$\text{Bias} = \frac{1}{T-h}\sum_{t=h}^{T-1}e_t(1)$$

例 12.2. 以美国 GDP 季度数据为例。数据是从 1947 年第一季度到 2010 年第季度的美国 GDP 季节调整后序列，以 2005 年 GDP 为基础进行了通胀调整，单位为 2005 年的十亿美元。

读入数据，计算 GDP 的对数值和对数增长率：

```
da <- read_table("q-gdpc96.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Year = col_double(),
##   Mon = col_character(),
##   Day = col_character(),
##   gdp = col_double()
## )
```

```
gdp <- ts(log(da[["gdp"]]), start=c(1947,1), frequency=4)
dgdp <- diff(gdp)
```

对数 GDP 的时间序列图：

```
plot(gdp, xlab="Year", ylab="ln(GDP)")
```

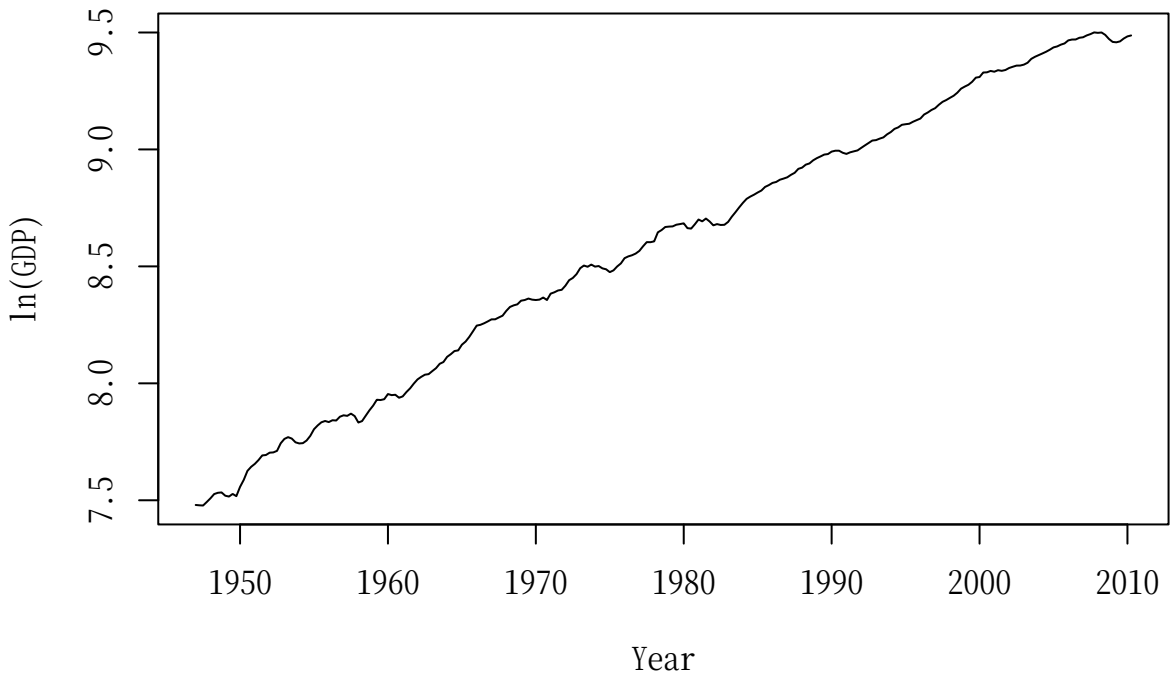


图 12.1: 对数 GDP 的时间序列

这个序列明显地有趋势。作其差分序列（对数增长率）的时间序列图：

```
plot(dgdp, xlab="Year", ylab="ln(GDP)")
abline(h=0, col="gray", lty=3)
```

这个图形没有明显区间，但是也不是白噪声的典型图形。

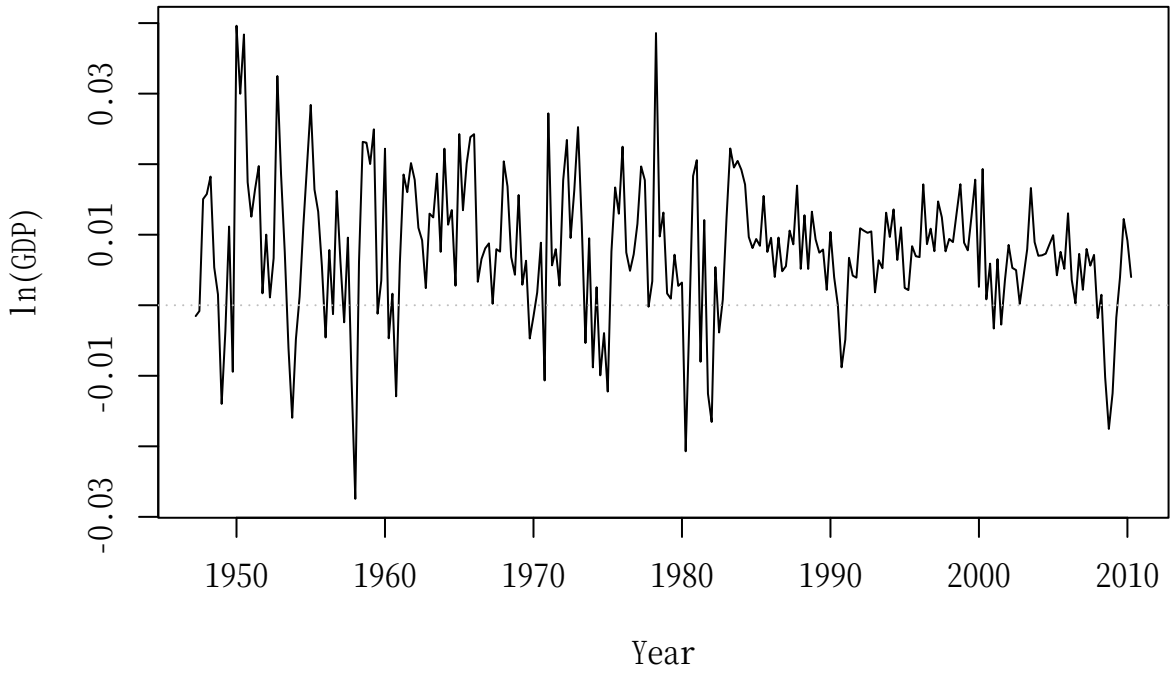


图 12.2: GDP 对数增长率的时间序列

作对数增长率的 ACF:

```
forecast::Acf(dgdp, main="")
```

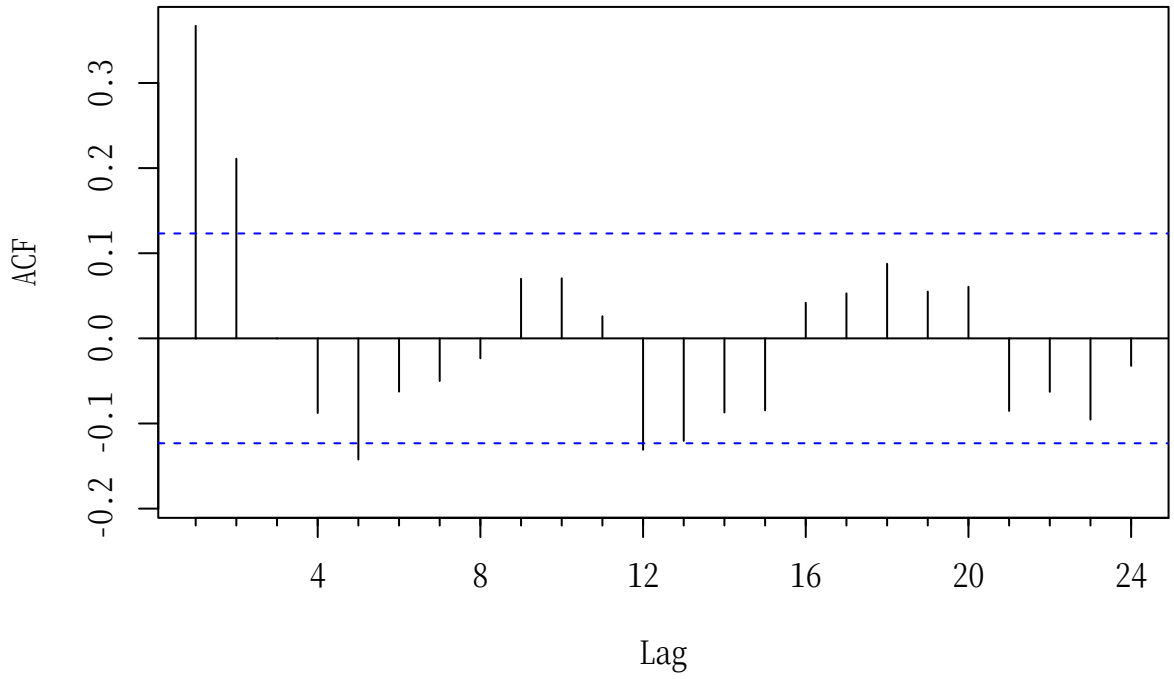


图 12.3: GDP 对数增长率的 ACF

可考虑 MA(2) 模型。

作对数增长率的 PACF:

```
pacf(dgdp, main="")
```

PACF 在滞后 12 处还显著。低阶可考虑 AR(3)。

用 AIC 对 AR 模型定阶：

```
ar(dgdp, method="mle")
```

```
##
## Call:
## ar(x = dgdp, method = "mle")
##
## Coefficients:
##       1        2        3
##  0.3461   0.1299  -0.1224
##
```

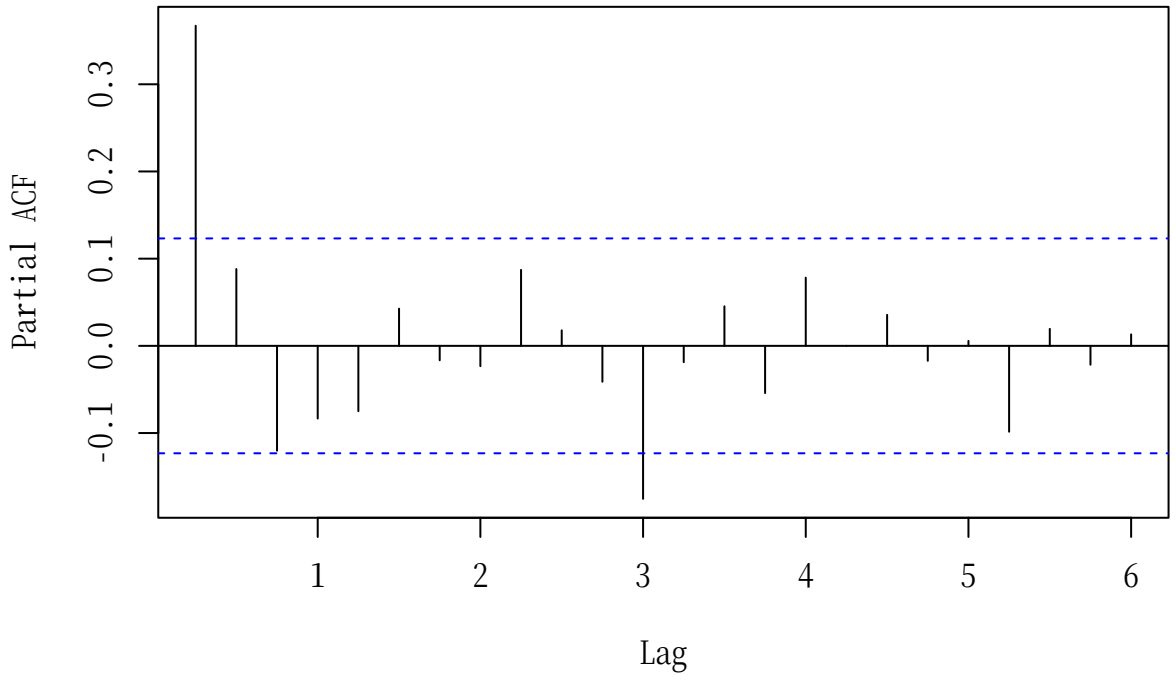


图 12.4: GDP 对数增长率的 PACF

```
## Order selected 3  sigma^2 estimated as  8.324e-05
```

选择了 3 阶。用 arima() 建模：

```
resm1 <- arima(dgdp, order=c(3,0,0)); resm1
```

```
##
## Call:
## arima(x = dgdp, order = c(3, 0, 0))
##
## Coefficients:
##          ar1     ar2      ar3  intercept
##       0.3462  0.1299  -0.1225     0.0079
## s.e.  0.0623  0.0655   0.0624     0.0009
##
## sigma^2 estimated as 8.323e-05:  log likelihood = 829.23,  aic = -1648.45
```

因为是季度序列，考虑模型 ARIMA$(3,0,0)(1,0,1)_4$：

```
resm2 <- arima(dgdp, order=c(3,0,0),
               seasonal=list(order=c(1,0,1), period=4))
resm2
```

```
##
## Call:
## arima(x = dgdp, order = c(3, 0, 0), seasonal = list(order = c(1, 0, 1), period = 4))
##
## Coefficients:
##          ar1     ar2      ar3    sar1     sma1  intercept
##       0.3305  0.1521  -0.1103  0.4966  -0.5865     0.0079
## s.e.  0.0633  0.0668   0.0635  0.2578   0.2357     0.0008
##
## sigma^2 estimated as 8.24e-05:  log likelihood = 830.47,  aic = -1646.93
```

从 AIC 来看第一个模型更好。

下面比较其一步预测误差，先写一个计算回测检验统计量的通用函数，这时从 (Tsay, 2013) 的伴随网站中提供的函数改进得到的。具体程序参见12.4。

两个模型的回测检验，从前面的 215 个开始建模并每次增加一个样本点计算一步预测误差：

```
cat("==== Model 1\n")
```

```
## ==== Model 1
```

```
res1 <- backtest(resm1, dgdp, n_min_sample=215, h=1)
print(res1$stats)
```

```
## # A tibble: 1 x 3
##   n_ahead    rmse     mae
##     <dbl>   <dbl>   <dbl>
```

```
## 1       1 0.00615 0.00443
```

```
cat("==== Model 2\n")
```

```
## ==== Model 2
```

```
res2 <- backtest(resm2, dgdp, n_min_sample=215, h=1)
print(res2$stats)
```

```
## # A tibble: 1 x 3
##   n_ahead    rmse     mae
##     <dbl>   <dbl>   <dbl>
## 1       1 0.00632 0.00455
```

用样本外一步预测均方误差平方根比较，模型一为 0.006153, 模型二为 0.006322，模型一略好，考虑到模型一还更简单，所以不论从样本内还是样本外比较都应选择模型一，即 AR(3) 模型。

12.3 模型平均

当有多个不同的模型都有相近的样本内和样本外表现时，可以用这些模型的预测的加权平均给出一个组合预测。这种技术称为**模型平均**。设有 m 个可用模型，都能给出超前一步预测的无偏预测（无偏是指超前一步预测误差期望为零），设第 i 个模型在时刻 j 给出的超前一步预测为 $\hat{x}_{i,h+1}$，则组合预测可定义为

$$\hat{x}_{h+1} = \sum_{i=1}^{m} w_i \hat{x}_{i,h+1}$$

其中 w_i 是非负的加权，$\sum_{i=1}^{m} w_i = 1$。

权重取法如等权，贝叶斯模型后验概率，等等。

12.4 附录：backtest 函数

改进版本的回测程序：

```
backtest <- function(
  modres, # 全集建模结果
  y,      # 要建模的时间序列
  h=1,    # 预测的最大步数
  n_min_sample, # 回测时最少的样本量
  time_y = NULL, # y 对应的时间标签
  x_list = NULL,  # 外生自变量，列表或数据框，序列长度与 y 相等
  fixed = NULL){
  library(tidyverse, quietly = TRUE)
  library(purrr, quietly = TRUE)
  library(xts, quietly = TRUE)

  ## 提取模型信息
  # (p,d,q):
```

```
order0 <- modres$arma[c(1, 6, 2)]
# 季节 (Q,D,Q)
orders <- modres$arma[c(3,7,4)]
# 周期
period <- modres$arma[5]

# fixed 选项，默认为与模型中输出结果相同
if(is.null(fixed)){
  fixed <- rep(NA, length(modres$mask))
  fixed[!modres$mask] <- coef(modres)[!modres$mask]
}

## 提取 include.mean 选项
get_include_mean <- function(){
  s <- as.character(as.expression(modres$call))
  s1 <- gsub("[ \n]", "", s)
  !grepl("include.mean=FALSE", s1, fixed=TRUE)[1]
}
include.mean <- get_include_mean()

## 将时间标签与数据分离，
## 时间标签存入 time_y，
## 数据向量存入 y0
if(is.ts(y) || is.xts(y)){
  y <- as.xts(y)
  y0 <- c(coredata(y))
  if(is.null(time_y)){
    time_y <- index(y)
  }
} else if(is.numeric(y)){
  y0 <- c(y)
  if(is.null(time_y)){
    time_y <- seq_along(y)
  }
} else {
  stop("Type of y should be numeric, or ts, or xts!")
}
stopifnot(length(time_y) == length(y))

stopifnot(is.numeric(h), length(h) == 1, h >= 1)

## 序号时间和真实时间的对应表
dtime <- tibble(
```

```
  t = seq_along(y0),
  time = time_y
)

## 真实数据序号与数值
dtrue <- tibble(
  t = seq_along(y0),
  y = y0
)

## 序列总长度
T <- length(y0)

## 如果有外生自变量且为时间序列，要求与 y 等长，且时间标签一致。转换为向量或矩阵。
## 为了适应多步预测，将 T+1 开始的 h-1 个值设置为 NA
make_xreg <- function(){
  xreg <- NULL

  if(!is.null(x_list)){
    if(is.numeric(x_list)) {
      x_list <- list(x = x_list)
    }

    x_list %>%
      # 每个列表元素长度必须等于输入时间序列长度
      walk(function(x){
        if(length(x) != length(y0))
          stop("Length of independent variable should equal length of time series!")
      } ) %>%
      # 选择 ts 和 xts 类型
      keep(function(x) is.ts(x) || is.xts(x)) %>%
      ## 对于时间序列，要求与输入时间序列时间完全匹配
      walk(function(x) {
        if(!all.equal(index(as.xts(x)), time_y))
          stop("Time of independent variable does not conform with time series!")
      })

    ## arima 函数和 predict.Arima 函数要求的 xreg 和 newxreg 选项为矩阵
    xreg <- as.matrix(as.data.frame(x_list))
    if(h>1){
      # 在多步预测时，会用到 T+1 到 T+h-1 处的外生自变量值，补为 NA
      xreg <- rbind(
        xreg,
```

```
        matrix(NA, nrow=h-1, ncol=ncol(xreg))
      )
    }
  } # if 非 NULL

  xreg
} # function
xreg <- make_xreg()

n_ahead <- seq(1, h, by=1)

# 对 t=n_min_sample, .+1, ..., T-1 作超前一步预测，以 h=1 为例。
# 先生成结果数据框
dres <- tibble(
  t = dtime$t,
  pred = vector("list", length=T))

for(n in seq(n_min_sample, T-1)){
  ytrain <- y0[1:n] # 本轮建模用的样本

  # 分无外生自变量和有外生自变量两种情形
  if(is.null(xreg)){
    # 没有外生自变量
    mod1 <- arima(
      ytrain,
      order = order0,
      seasonal = list(
        order = orders,
        period = period),
      fixed = fixed,
      include.mean=include.mean)
    pre <- predict(
      mod1,
      n.ahead=h,
      se.fit=TRUE)
  } else {
    # 有外生自变量
    xtrain <- xreg[1:n,,drop=FALSE]
    xtest <- xreg[(n+1):(n+h),,drop=FALSE]
    mod1 <- arima(
      ytrain,
      order = order0,
      seasonal = list(
```

```
          order = orders,
          period = period),
        xreg = xtrain,
        fixed = fixed,
        include.mean=include.mean)
      pre <- predict(
        mod1,
        newxreg = xtest,
        n.ahead = h,
        se.fit=TRUE)
    }

    ## 将基于 1:n 的预测结果作为子数据框存入到
    ## 结果 dres 的 [n,"pred"] 位置作为列表类型列
    dres[["pred"]][n] <- list(tibble(
      n_ahead = n_ahead,
      predict = as.vector(pre$pred),
      se = as.vector(pre$se) ) )
  } # for t

  ## 结果仅保留有预测的时间点，应为 T - n_min_sample 行
  dres <- dres[n_min_sample:(T-1), ]

  ## 获得 1:n_ahead 步的预测结果，与真实值按时间对齐后计算预测误差
  dtest <- dres %>%
    # 将每步预测的子数据框展开
    unnest(cols=c(pred)) %>%
    ungroup() %>%
    # 计算预测值对应的时间
    mutate(time_pred = t + n_ahead) %>%
    # 去掉 time，即所用的训练样本截止时间
    select(-t) %>%
    # 取预测值对应时间小于等于 T
    dplyr::filter(time_pred <= T) %>%
    # 将预测值对应的时间与真实值对应的时间对齐，
    inner_join(dtrue, by = c("time_pred" = "t")) %>%
    # 预测值对应的序号时间 time_pred 改名为 t
    rename(t = time_pred) %>%
    mutate(error = predict - y) %>%
    # 增加真实时间列 time
    inner_join(dtime, by = "t") %>%
    select(t, time, n_ahead, y, predict, se, error) %>%
    arrange(t, n_ahead)
```

```
  # RMSE 和 MAE
  stats <- dtest %>%
    select(n_ahead, error) %>%
    group_by(n_ahead) %>%
    summarise(
      rmse = sqrt(mean(error^2)),
      mae = mean(abs(error)))

  list(predict = dtest, stats = stats)
}
```

改进版本的测试程序：

```
gen_arma42 <- function(n=100){
  set.seed(101)

  y <- arima.sim(
    model=list(
      ar=c(-0.9, -1.4, -0.7, -0.6),
      ma=c(0.5, -0.4)),
    n = n,
    sd = sqrt(4))

  as.vector(y)
}

## ARMA(4,2) 无自变量，超前 1 步
test11 <- function(){
  y <- 100 + gen_arma42(n=100)
  mod <- arima(
    y, order = c(4, 0, 2),
    include.mean=TRUE
  )

  res <- backtest(
    mod, y,
    h=1,    # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
  )
  print(res)

  # 手工检查预测
  pre <- numeric(20)
```

```
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      include.mean=TRUE
    )
    pred1 <- predict(mod1)
    pre[t] <- pred1$pred
    se[t] <- pred1$se
  }
  stats <- res$stats
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}
## 成功。
## 将测试程序中两处 inlude.mean=TRUE 改为 FALSE 也成功。

## ARMA(4,2) 无自变量，超前 2 步
test12 <- function(){
  y <- 100 + gen_arma42(n=100)
  mod <- arima(
    y, order = c(4, 0, 2),
    include.mean=TRUE
  )

  res <- backtest(
    mod, y,
    h=2,    # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
  )
  print(res)

  # 手工检查预测，1 步
  pre <- numeric(20)
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      include.mean=TRUE
    )
    pred1 <- predict(mod1)
```

```
    pre[t] <- pred1$pred
    se[t] <- pred1$se
  }
  stats <- res$stats[1,]
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("1 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")

  # 手工检查预测, 2 步
  pre <- numeric(19)
  se <- numeric(19)
  for(t in 1:19){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      include.mean=TRUE
    )
    pred1 <- predict(mod1, n.ahead=2)
    pre[t] <- pred1$pred[2]
    se[t] <- pred1$se[2]
  }
  stats <- res$stats[2,]
  rmse <- sqrt(mean((y[82:100] - pre)^2))
  mae <- mean(abs(y[82:100] - pre))
  cat("2 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}
## 成功


## 对 ts 类型的测试
## ARMA(4,2) 无自变量, 超前 1 步
test21 <- function(){
  y <- 100 + gen_arma42(n=100)
  y <- ts(y, start=c(1900,1), frequency=4)
  mod <- arima(
    y, order = c(4, 0, 2),
    include.mean=TRUE
  )

  res <- backtest(
    mod, y,
    h=1,    # 预测的最大步数
```

```
    n_min_sample = 80 # 回测时最少的样本量
  )
  print(res)

  # 手工检查预测
  pre <- numeric(20)
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      include.mean=TRUE
    )
    pred1 <- predict(mod1)
    pre[t] <- pred1$pred
    se[t] <- pred1$se
  }
  stats <- res$stats
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}
## 成功

## ARMA(4,2) 无自变量，超前 2 步，xts 日数据，时间不连贯
test22 <- function(){
  y <- 100 + gen_arma42(n=100)
  time <- sort(sample(lubridate::ymd("2001-01-03") + 1:130, size=100))
  y <- xts(y, time)
  mod <- arima(
    y, order = c(4, 0, 2),
    include.mean=TRUE
  )

  res <- backtest(
    mod, y,
    h=2,    # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
  )
  print(res)

  # 手工检查预测，1 步
  pre <- numeric(20)
```

```
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      include.mean=TRUE
    )
    pred1 <- predict(mod1)
    pre[t] <- pred1$pred
    se[t] <- pred1$se
  }
  stats <- res$stats[1,]
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("1 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")

  # 手工检查预测, 2 步
  pre <- numeric(19)
  se <- numeric(19)
  for(t in 1:19){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      include.mean=TRUE
    )
    pred1 <- predict(mod1, n.ahead=2)
    pre[t] <- pred1$pred[2]
    se[t] <- pred1$se[2]
  }
  stats <- res$stats[2,]
  rmse <- sqrt(mean((y[82:100] - pre)^2))
  mae <- mean(abs(y[82:100] - pre))
  cat("2 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}
## 成功

gen_arima412 <- function(n=100){
  set.seed(101)

  y <- arima.sim(
    model=list(
      ar=c(-0.9, -1.4, -0.7, -0.6),
      ma=c(0.5, -0.4),
```

```
      order=c(4,1,2)),
    n = n,
    sd = sqrt(4))

  as.vector(y)[-1]
}

## ARIMA(4,1,2) 无自变量，超前 2 步
test31 <- function(){
  y <- 100 + gen_arima412(n=100)
  mod <- arima(
    y, order = c(4, 1, 2)  )

  res <- backtest(
    mod, y,
    h=2,    # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
  )
  print(res)

  # 手工检查预测，1 步
  pre <- numeric(20)
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 1, 2) )
    pred1 <- predict(mod1)
    pre[t] <- pred1$pred[1]
    se[t] <- pred1$se[1]
  }
  stats <- res$stats[1,]
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("1 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")

  # 手工检查预测，2 步
  pre <- numeric(19)
  se <- numeric(19)
  for(t in 1:19){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 1, 2))
    pred1 <- predict(mod1, n.ahead=2)
```

```
    pre[t] <- pred1$pred[2]
    se[t] <- pred1$se[2]
  }
  stats <- res$stats[2,]
  rmse <- sqrt(mean((y[82:100] - pre)^2))
  mae <- mean(abs(y[82:100] - pre))
  cat("2 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}
## 成功。

## 模拟 ARIMA(1,1,1)(1,1,1)_4
## $$(1-0.5B)(1 - 0.3B^4)(1-B)(1-B^4) y_t
##   = (1+0.3B)(1+0.1B^4) e_t$$
gen_arimas <- function(n=100){
  set.seed(101)

  y <- arima.sim(
    model=list(
      ar=c(0.5, 0, 0, 0.3, -0.15),
      ma=c(0.3, 0, 0, 0.1, 0.03),
      order=c(5,1,5)),
    n = n + 40,
    sd = sqrt(4))
  nn <- length(y)
  y <- y[1:(nn-4)] + y[5:nn]
  nn <- length(y)

  as.vector(y)[(nn-n+1):nn]
}


## ARIMA(1,1,1)(1,1,1)_4 无自变量, 超前 2 步
test41 <- function(){
  y <- 100 + gen_arimas(n=100)
  mod <- arima(
    y, order = c(1, 1, 1),
    seasonal = list(order = c(1,1,1), period=4))

  res <- backtest(
    mod, y,
    h=2,     # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
```

```
  )
  print(res)

  # 手工检查预测, 1 步
  pre <- numeric(20)
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(1,1,1),
      seasonal = list(order = c(1,1,1), period=4) )
    pred1 <- predict(mod1)
    pre[t] <- pred1$pred[1]
    se[t] <- pred1$se[1]
  }
  stats <- res$stats[1,]
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("1 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")

  # 手工检查预测, 2 步
  pre <- numeric(19)
  se <- numeric(19)
  for(t in 1:19){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(1,1,1),
      seasonal = list(order = c(1,1,1), period=4) )
    pred1 <- predict(mod1, n.ahead=2)
    pre[t] <- pred1$pred[2]
    se[t] <- pred1$se[2]
  }
  stats <- res$stats[2,]
  rmse <- sqrt(mean((y[82:100] - pre)^2))
  mae <- mean(abs(y[82:100] - pre))
  cat("2 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}


## 模拟稀疏系数 ARMA(5,5)
## 模拟 ARIMA(1,1,1)(1,1,1)_4
## $$(1-0.5B)(1 - 0.3B^4) y_t
##   = (1+0.3B)(1+0.1B^4) e_t$$
```

```r
gen_arma55 <- function(n=100){
  set.seed(101)

  y <- arima.sim(
    model=list(
      ar=c(0.5, 0, 0, 0.3, -0.15),
      ma=c(0.3, 0, 0, 0.1, 0.03),
      order=c(5,0,5)),
    n = n + 40,
    sd = sqrt(4))

  nn <- length(y)
  as.vector(y)[(nn-n+1):nn]
}

test51 <- function(){
  y <- 100 + gen_arma55(n=100)
  fixed <- rep(NA, 11)
  fixed[c(2,3,7,8)] <- 0
  mod <- arima(
    y, order = c(5,0,5),
    fixed = fixed, transform.pars = FALSE)

  res <- backtest(
    mod, y,
    h=2,    # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
  )
  print(res)

  # 手工检查预测, 1 步
  pre <- numeric(20)
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(5,0,5),
      fixed = fixed, transform.pars = FALSE )
    pred1 <- predict(mod1)
    pre[t] <- pred1$pred[1]
    se[t] <- pred1$se[1]
  }
  stats <- res$stats[1,]
  rmse <- sqrt(mean((y[81:100] - pre)^2))
```

```
  mae <- mean(abs(y[81:100] - pre))
  cat("1 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")

  # 手工检查预测, 2 步
  pre <- numeric(19)
  se <- numeric(19)
  for(t in 1:19){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(5,0,5),
      fixed = fixed, transform.pars = FALSE)
    pred1 <- predict(mod1, n.ahead=2)
    pre[t] <- pred1$pred[2]
    se[t] <- pred1$se[2]
  }
  stats <- res$stats[2,]
  rmse <- sqrt(mean((y[82:100] - pre)^2))
  mae <- mean(abs(y[82:100] - pre))
  cat("2 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}
## 成功


## 带有一个自变量的 ARMA(4,2), 超前 1 步
test61 <- function(n=100){
  yerr <- gen_arma42(n=n)
  x1 <- sample(1:10, size=n, replace=TRUE)
  y <- 100 + 2 * x1 + yerr
  mod <- arima(
    y, order = c(4, 0, 2),
    xreg = x1)

  res <- backtest(
    mod, y,
    x_list = list(x1),
    h=1,    # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
  )
  print(res)

  # 手工检查预测, 1 步
  pre <- numeric(20)
```

```
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      xreg = x1[1:(80+t-1)])
    pred1 <- predict(mod1, newxreg = x1[80+t])
    pre[t] <- pred1$pred[1]
    se[t] <- pred1$se[1]
  }
  stats <- res$stats[1,]
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("1 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}
## 成功

## 带有一个自变量的 ARMA(4,2), 超前 2 步
test62 <- function(n=100){
  yerr <- gen_arma42(n=n)
  x1 <- sample(1:10, size=n, replace=TRUE)
  y <- 100 + 2 * x1 + yerr
  mod <- arima(
    y, order = c(4, 0, 2),
    xreg = x1)

  res <- backtest(
    mod, y,
    x_list = list(x1),
    h=2,    # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
  )

  # 手工检查预测, 1 步
  pre <- numeric(20)
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      xreg = x1[1:(80+t-1)])
    pred1 <- predict(mod1, newxreg = x1[80+t])
    pre[t] <- pred1$pred[1]
    se[t] <- pred1$se[1]
```

```
  }
  stats <- res$stats[1,]
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("1 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")

  # 手工检查预测, 2 步
  pre <- numeric(19)
  se <- numeric(19)
  for(t in 1:19){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      xreg = x1[1:(80+t-1)])
    pred1 <- predict(mod1, newxreg = cbind(x1[80+t+(0:1)]), n.ahead=2)
    pre[t] <- pred1$pred[2]
    se[t] <- pred1$se[2]
  }
  stats <- res$stats[2,]
  rmse <- sqrt(mean((y[82:100] - pre)^2))
  mae <- mean(abs(y[82:100] - pre))
  cat("2 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}

## 带有两个个自变量的 ARMA(4,2), 超前 2 步
test63 <- function(n=100){
  yerr <- gen_arma42(n=n)
  x1 <- sample(1:10, size=n, replace=TRUE)
  x2 <- sample(11:20, size=n, replace=TRUE)
  xx <- cbind(x1, x2)
  y <- 100 + 2 * x1 + 0.5*x2 + yerr
  mod <- arima(
    y, order = c(4, 0, 2),
    xreg = xx)

  res <- backtest(
    mod, y,
    x_list = list(x1, x2),
    h=2,    # 预测的最大步数
    n_min_sample = 80 # 回测时最少的样本量
  )
```

```
  # 手工检查预测, 1 步
  pre <- numeric(20)
  se <- numeric(20)
  for(t in 1:20){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      xreg = xx[1:(80+t-1),,drop=FALSE])
    pred1 <- predict(mod1, newxreg = xx[80+t,,drop=FALSE])
    pre[t] <- pred1$pred[1]
    se[t] <- pred1$se[1]
  }
  stats <- res$stats[1,]
  rmse <- sqrt(mean((y[81:100] - pre)^2))
  mae <- mean(abs(y[81:100] - pre))
  cat("1 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")

  # 手工检查预测, 2 步
  pre <- numeric(19)
  se <- numeric(19)
  for(t in 1:19){
    mod1 <- arima(
      y[1:(80+t-1)], order = c(4, 0, 2),
      xreg = xx[1:(80+t-1),,drop=FALSE])
    pred1 <- predict(
      mod1, newxreg = xx[80+t+(0:1),,drop=FALSE], n.ahead=2)
    pre[t] <- pred1$pred[2]
    se[t] <- pred1$se[2]
  }
  stats <- res$stats[2,]
  rmse <- sqrt(mean((y[82:100] - pre)^2))
  mae <- mean(abs(y[82:100] - pre))
  cat("2 step ahead -- RMSE error: ", rmse - stats[["rmse"]],
      " MAE error: ", mae - stats[["mae"]], "\n")
}
## 成功

## 运行所有测试
run.all.tests <- function(){
  fun.list <- list(
    test11, test12,
    test21, test22,
    test31,
```

```
    test41,
    test51,
    test61, test62
  )
  for(fun in fun.list)
    fun()
}
```

蔡瑞胸教授的原始版本：

```
"backtest" <- function(m1, rt, orig, h, xre=NULL, fixed=NULL, inc.mean=TRUE){
  # m1: is a time-series model object
  # orig: is the starting forecast origin
  # rt: the time series
  # xre: the independent variables
  # h: forecast horizon
  # fixed: parameter constraint
  # inc.mean: flag for constant term of the model.
  #
  regor=c(m1$arma[1],m1$arma[6],m1$arma[2])
  seaor=list(order=c(m1$arma[3],m1$arma[7],m1$arma[4]),period=m1$arma[5])
  T=length(rt)
  if(!is.null(xre) && !is.matrix(xre)) xre=as.matrix(xre)
  ncx=ncol(xre)
  if(orig > T) orig=T
  if(h < 1) h=1
  rmse=rep(0,h)
  mabso=rep(0,h)
  nori=T-orig
  err=matrix(0,nori,h)
  jlast=T-1
  for (n in orig:jlast){
    jcnt=n-orig+1
    x=rt[1:n]
    if (!is.null(xre)){
      pretor=xre[1:n,]
      mm=arima(x,order=regor,seasonal=seaor,xreg=pretor,fixed=fixed,include.mean=inc.mean)
      nx=xre[(n+1):(n+h),]
      if(h==1)nx=matrix(nx,1,ncx)
      fore=predict(mm,h,newxreg=nx)
    }
    else {
      mm=arima(x,order=regor,seasonal=seaor,xreg=NULL,fixed=fixed,include.mean=inc.mean)
      fore=predict(mm,h,newxreg=NULL)
```

```
    }
    kk=min(T,(n+h))
    # nof is the effective number of forecats at the forecast origin n.
    nof=kk-n
    pred=fore$pred[1:nof]
    obsd=rt[(n+1):kk]
    err[jcnt,1:nof]=obsd-pred
  }
  #
  for (i in 1:h){
    iend=nori-i+1
    tmp=err[1:iend,i]
    mabso[i]=sum(abs(tmp))/iend
    rmse[i]=sqrt(sum(tmp^2)/iend)
  }
  print("RMSE of out-of-sample forecasts")
  print(rmse)
  print("Mean absolute error of out-of-sample forecasts")
  print(mabso)
  backtest <- list(origin=orig,error=err,rmse=rmse,mabso=mabso)
}
```

Chapter 13

线性时间序列案例学习—汽油价格

这一章用三个实例来详细讲解如何用 R 语言和线性时间序列模型分析实际数据，并展现线性时间序列模型的适用性与局限性。

数据为：

- 1997-01-06 到 2010-09-27 的美国普通汽油价格周数据；
- 1880 年 1 月到 2010 年 8 月全球温度异常值的月度数据；
- 美国失业率月度数据，包括首次申领失业救济金人数的序列以及不包括的序列。

这些数据是持续更新的，也反映了全球或美国经济的重要方面，其建模问题有足够的代表性。

用时间序列分析或者统计方法建模时，最常遇到的困难是如何选取一个适当的模型。当数据之间的动态相依性很复杂时，模型的形式难以确定；当有多个模型都表现很好时，模型难以选择。

George Box 教授关于建模问题有一句名言 (Box, 1976):

> 所有的模型都是错误的，但是其中有一些是有用的。

不同的研究目的可能偏向于不同的模型。

时间序列数据建模的一些指导原则：

- 数据仅是可利用信息的一部分，专业知识、常识、历史事件等都是需要考虑的可利用信息。
- 多个模型可能表现相近，这时并没有一个“正确的”模型，选择一个就可以。
- 在预测时，可以结合多个模型来改善预测效果。
- 建模的过程是从最简单的模型到逐步复杂，千万不能以为理论上越复杂、理解和掌握的人数越少的模型才是越好的模型。
- 模型应尽可能选择更简洁的模型，如果两个模型的表现相近，一定要选择更简单的一个。这也是避免过度拟合的要求。过度拟合会导致模型的外推预测能力丧失。

13.1 数据读入与探索性分析

原油价格和汽油价格对美国经济的重要影响：

- 20 实际 70 年代初期石油危机展示了石油资源的战略意义。

- 2008 年汽油价格高涨影响了居民生活：
  - 交通成本上涨
  - 供热成本上涨
  - 食品，服务价格上涨 ==> 通货膨胀
  - 降低了消费者在其它消费项目上的可支配收入，可能导致经济萧条

研究美国汽油价格以及原油价格对其影响。数据为美国普通石油价格周数据，从 1997-01-06 到 2010-09-27。时间是每周三, 共 717 个时间点; 美国原油价格周数据，从 1997-01-03 到 2010-09-24。时间是每周日。共 717 个时间点。原油价格比汽油价格早 3 天发布。

读入数据：石油价格和汽油价格分成了两个输入文件。石油价格的输入文件 `w-petroprice.txt` 又是一个用单个空格、多个空格、制表符、空格加制表符分隔的文件。这种文件千万不要自己制造出来。目前 readr 包还不支持这样的文件，用 R 原来的 `read.table()` 函数来读入：

```
da1 <- read.table(
  "w-petroprice.txt", header=TRUE)
xts.petrol <- xts(
  da1[,-(1:3)], make_date(da1[["Year"]], da1[["Mon"]], da1[["Day"]]))
da2 <- read_table(
  "w-gasoline.txt", col_names="pgas",
  col_types=cols(.default=col_double())
)
xts.gasoline <- xts(da2[[1]], index(xts.petrol)+3)
xts.pgs <- log(xts.gasoline[,1]) # 美国普通汽油价格对数值序列
pgs <- coredata(xts.pgs)[,1]
xts.pus <- log(xts.petrol[,"US"]) # 美国原油价格对数值序列
pus <- coredata(xts.pus)[,1]
## 一阶差分序列
dpgs <- diff(pgs)
dpus <- diff(pus)
```

如果希望用 readr 包来读入，最简单的解决办法是用文本编辑器编辑，将制表符都替换成单个空格；下面的 R 程序用了 R 语言的文件操作和正则表达式进行替换：

```
la <- readLines("w-petroprice.txt")
la <- gsub("\\s+", " ", la)
la <- gsub("^\\s+|\\s$", "", la)
da1 <- read_table(
  paste(la, collapse="\n"),
  col_types=cols(.default=col_double())
)
```

汽油价格对数值的时间序列图：

```
plot(
  xts.pgs,
  main="ln Gasoline Price",
```

```
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col="red"
)
```

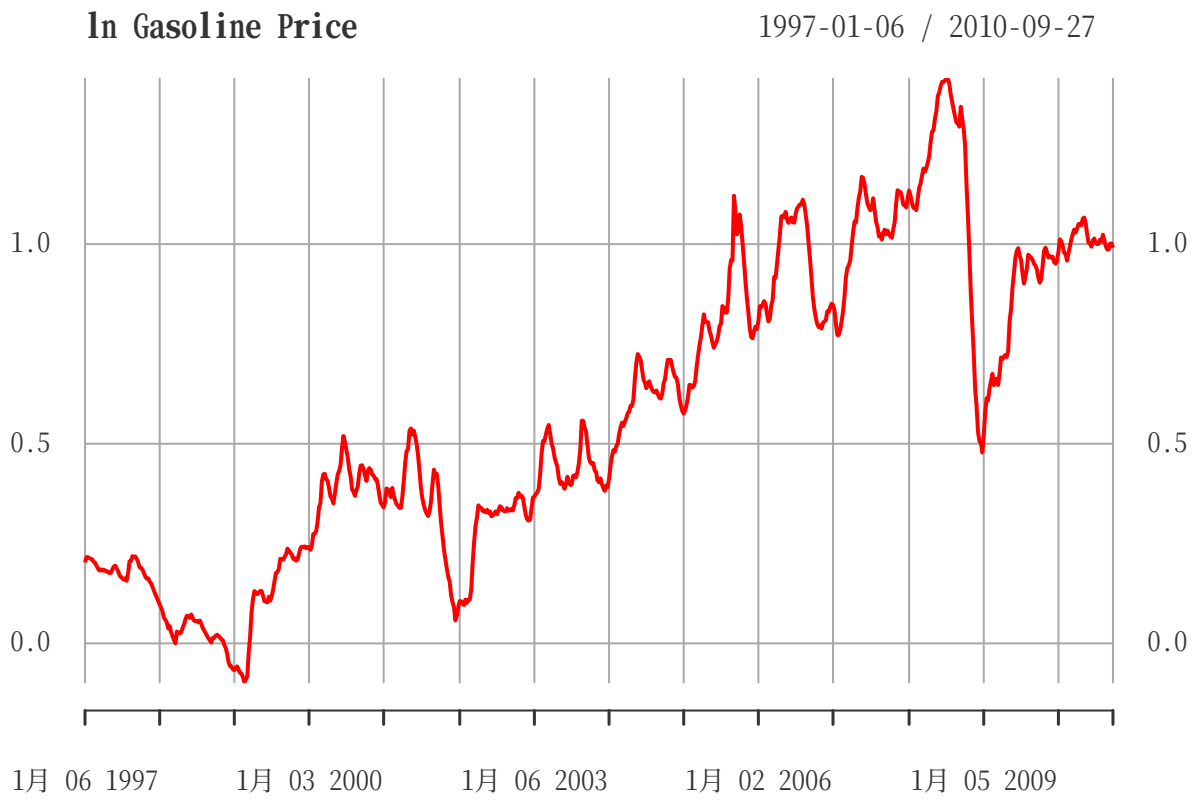


图 13.1: 汽油价格对数值序列

原油价格对数值的时间序列图：

```
plot(
  xts.pus,
  main="ln Petrol Price",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col="black"
)
```

将对数汽油价格与对数原油价格同时绘制：

```
xts.pgs.pus <- xts(cbind(pgs, pus), index(xts.pgs))
plot(
  xts.pgs.pus,
  main="ln Petrol Price and ln Gasoline Price",
```

图 13.2: 原油价格对数值序列

```
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col=c("red", "black")
)
```

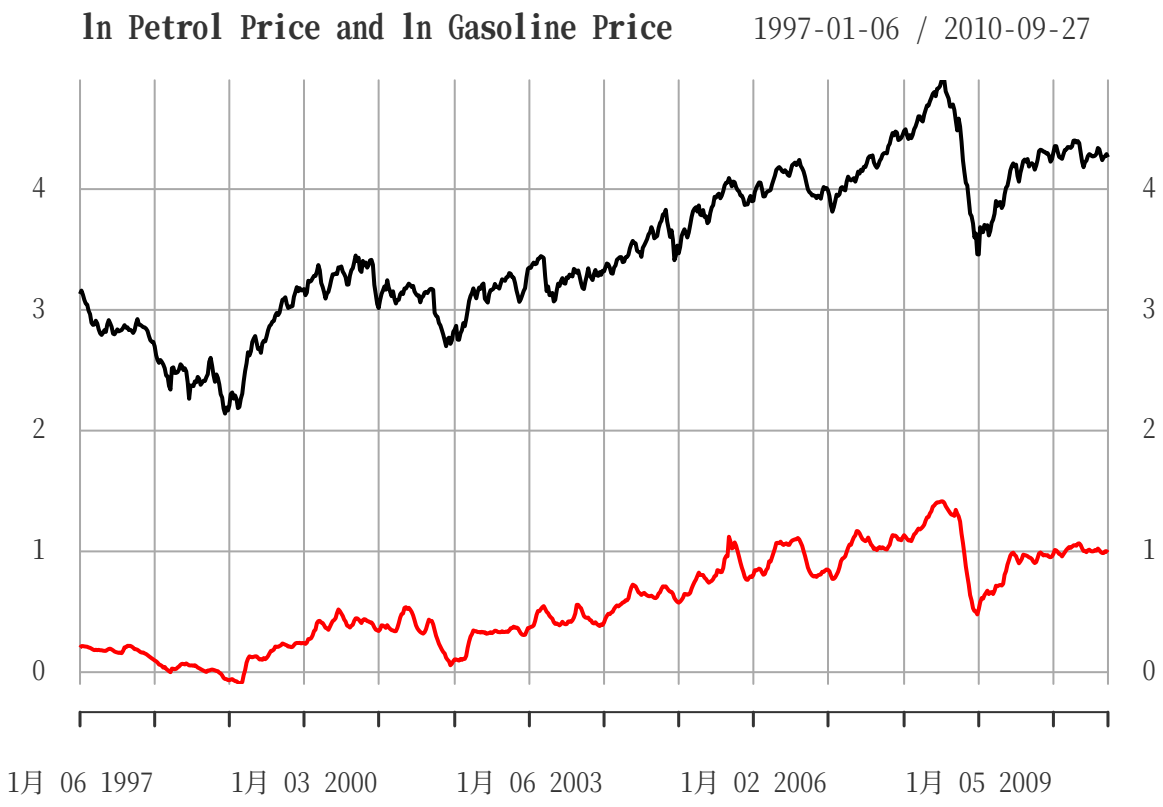


图 13.3: 汽油、原油价格对数值序列

为什么要用对数价格而不用原始价格？请看原始价格的图形：

```
xts.orig.pgs.pus <- xts(cbind(exp(pgs), exp(pus)), index(xts.pgs))
plot(
  xts.orig.pgs.pus,
  main="Petrol Price and Gasoline Price",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col=c("red", "black")
)
```

从图形可以看出，原始价格随时间而振幅变大，两个序列画在同一序列很难看出有同步变化。对数变换是最常见的克服指数增长、将比例关系转换成线性关系的变换。

对数汽油价格对对数原油价格的散点图：

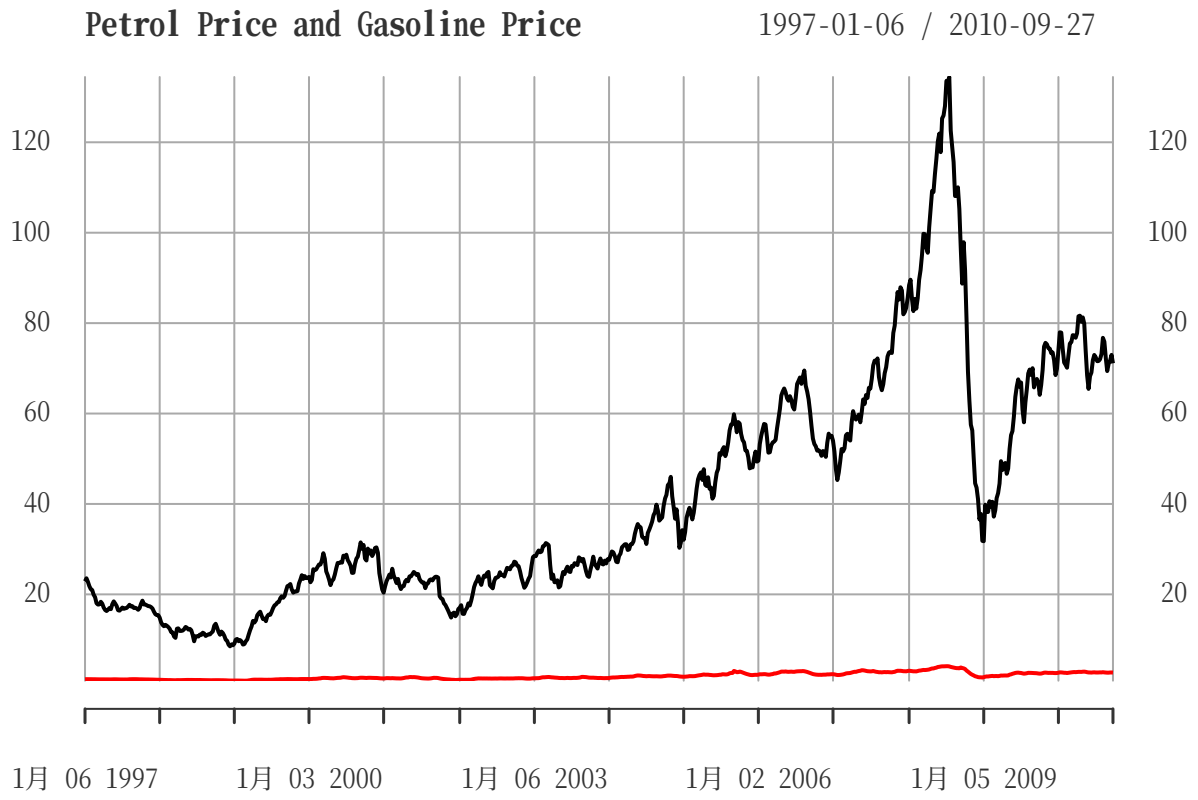


图 13.4: 汽油、原油价格原始序列

```
plot(pus, pgs, xlab="ln(Petrol Price)", ylab="ln(Gasoline Price)",
     pch=16, cex=0.3)
```

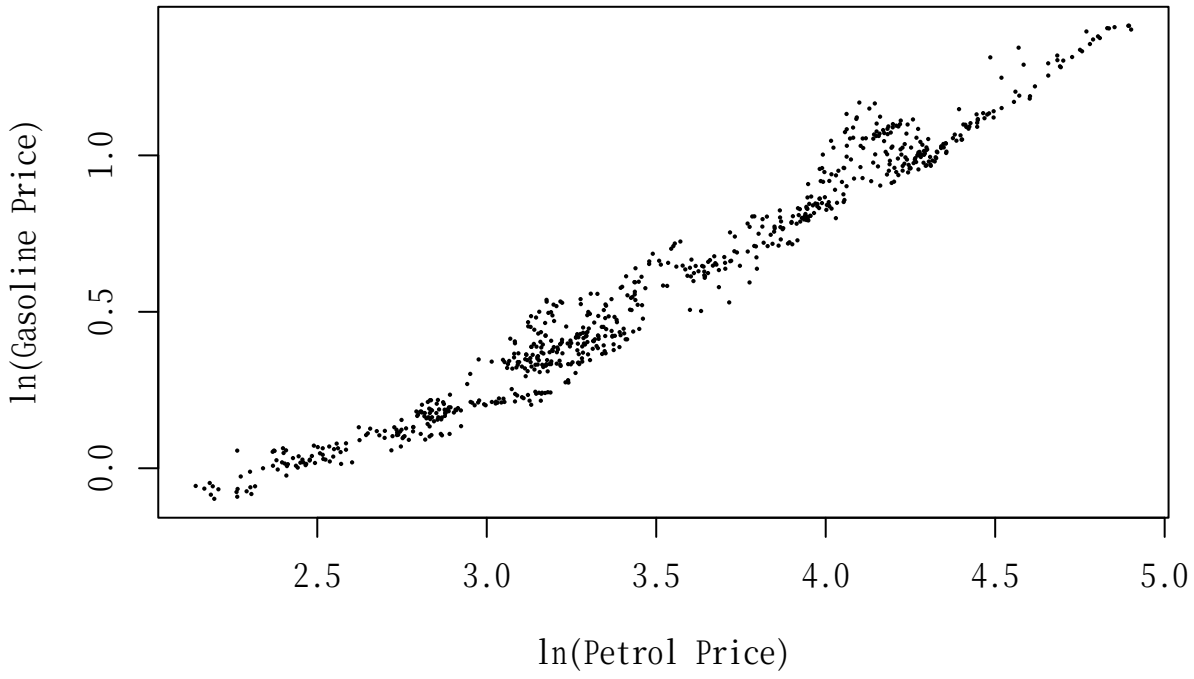


图 13.5: 对数汽油对对数原油价格

可以看出汽油价格与同期原油价格相关性很强，相关系数为：

```
cor.test(pus, pgs)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pus and pgs
## t = 143.56, df = 715, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9804472 0.9853826
## sample estimates:
##       cor
## 0.9830925
```

相关系数为 0.98。

因为对数价格呈现出缓慢增长随机趋势，所以使用其差分值（价格的对数变化率）作为建模数据。

汽油价格对数增长率的时间序列图：

```
plot(
  diff(xts.pgs),
  main="Gasoline Price Log Increase Rate",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col="red"
)
```

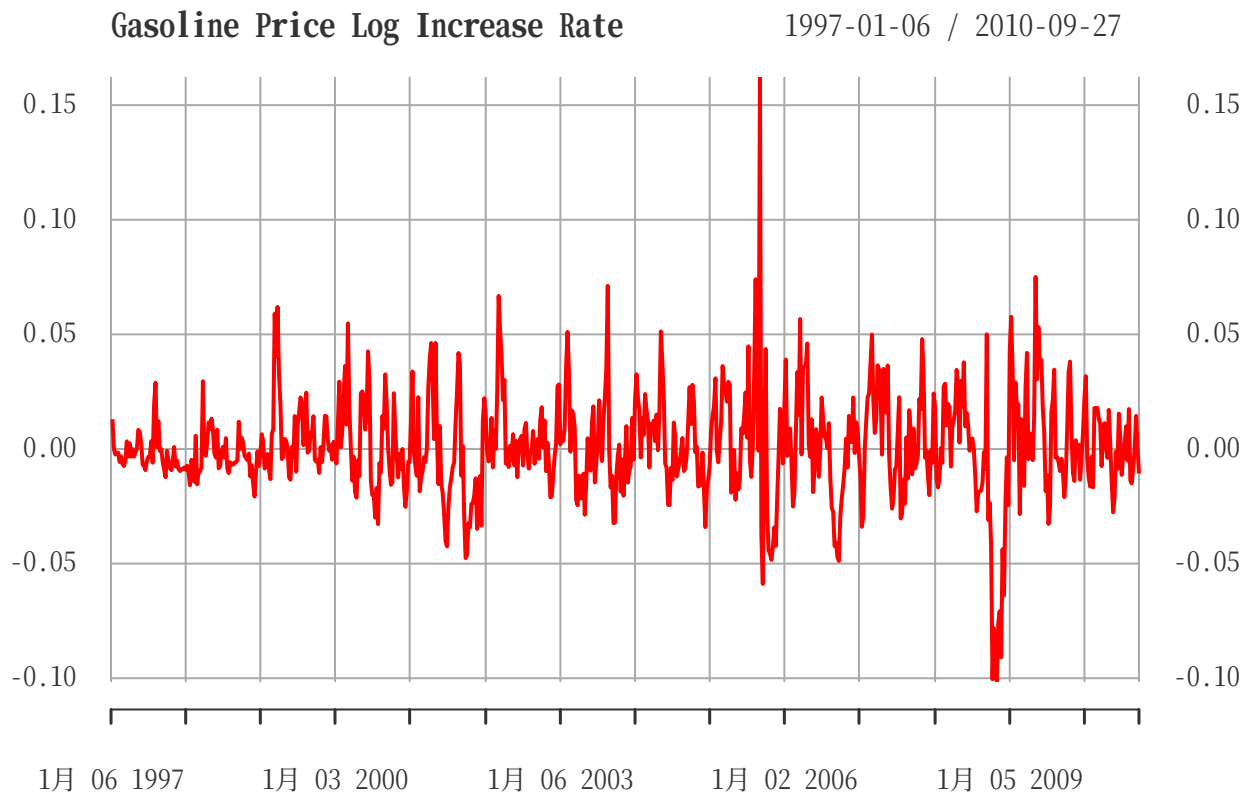


图 13.6: 汽油价格对数增长率序列

汽油价格对数增长率的直方图：

```
hist(dpgs, xlab="Gasoline Price Log Increase Rate")
```

汽油价格对数增长率的 ACF：

```
acf(dpgs, main="")
```

明显非白噪声，也不是低阶 MA 的典型形状。ACF 衰减很快，符合线性时间序列要求。

汽油价格对数增长率的 PACF：

```
pacf(dpgs, main="")
```

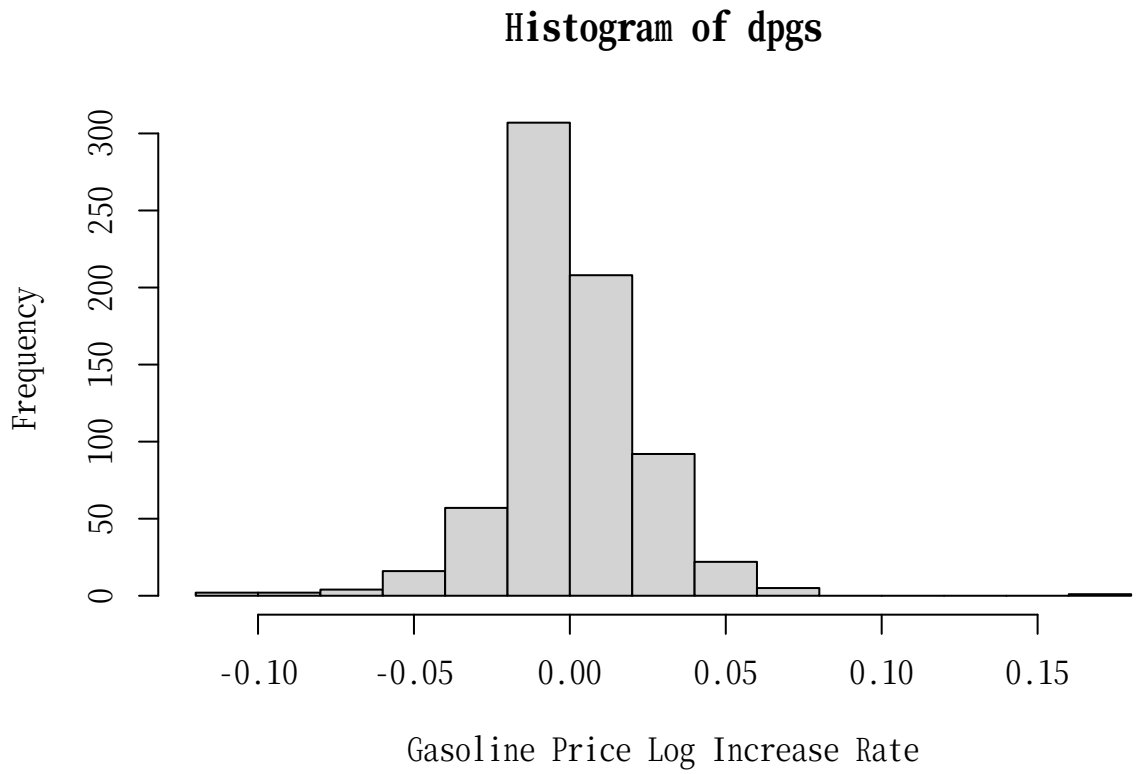


图 13.7: 汽油价格对数增长率分布

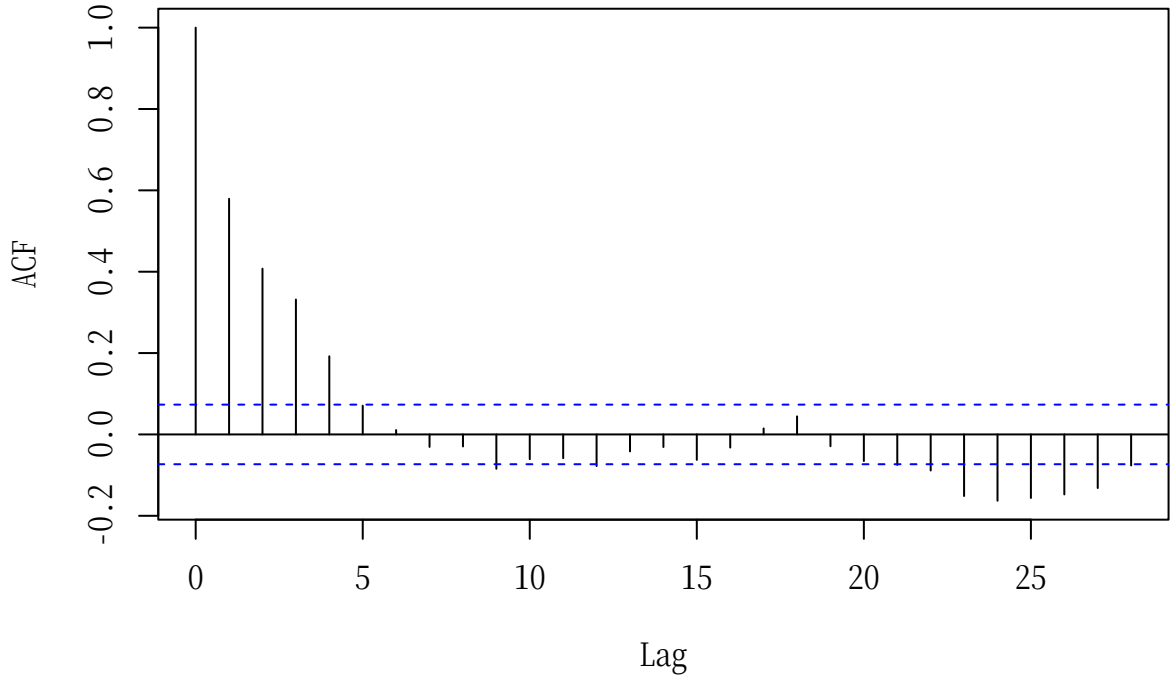


图 13.8: 汽油价格对数增长率 ACF

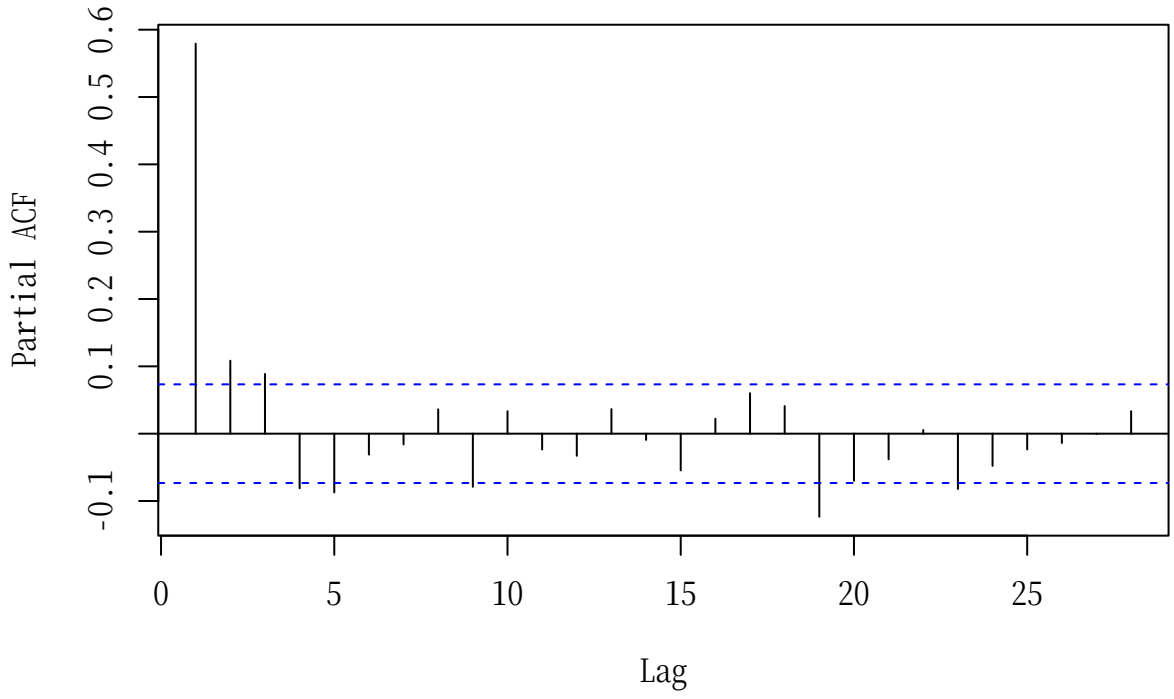


图 13.9: 汽油价格对数增长率 PACF

PACF 在滞后 1 到 5 都是显著的，在滞后 19 处也比较显著。用低阶 AR 有一定合理性。PACF 衰减很快，符合线性时间序列要求。

对数汽油价格序列的单位根检验：

```
fUnitRoots::adfTest(pgs, lags=5, type="c")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 5
##   STATISTIC:
##     Dickey-Fuller: -1.5863
##   P VALUE:
##     0.4678
##
## Description:
##  Wed Mar 23 16:19:26 2022 by user: Lenovo
```

检验结果是有单位根。但是，如果扣除一个非随机线性趋势项检验，就没有单位根：

```
fUnitRoots::adfTest(pgs, lags=5, type="ct")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 5
##   STATISTIC:
##     Dickey-Fuller: -3.7307
##   P VALUE:
##     0.02246
##
## Description:
##  Wed Mar 23 16:19:26 2022 by user: Lenovo
```

下面还是按照对数汽油价格是单位根序列来建模。

13.2 AR(5) 模型

记汽油价格对数增长率为 x_t。从时间序列图13.6来看，x_t 在 2005 年有一个高涨：

```
plot(
  diff(xts.pgs)["2005"],
  main="Gasoline Price Log Increase Rate",
  major.ticks="month", minor.ticks=NULL,
  grid.ticks.on="months",
  col="red"
)
```

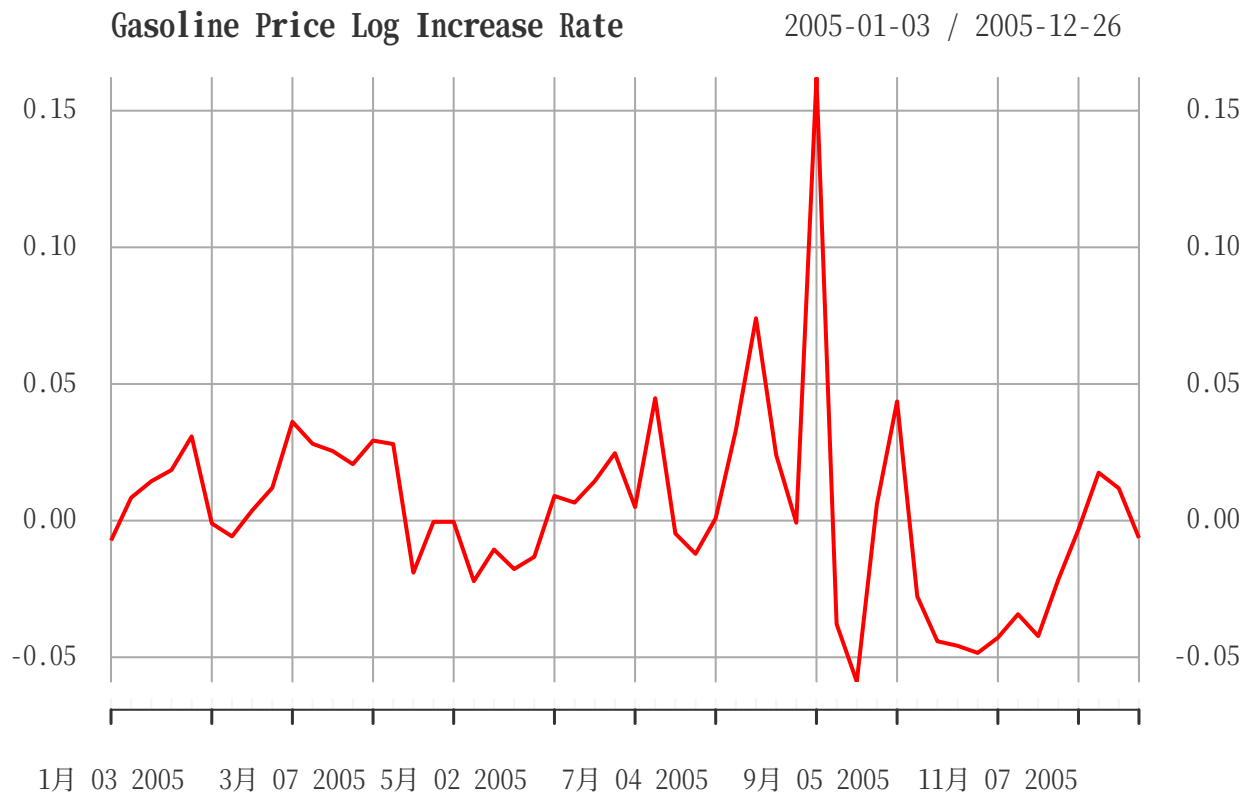


图 13.10: 汽油价格对数增长率序列 2005 年

在 2008 年有一个大跌：

```
plot(
  diff(xts.pgs)["2008"],
  main="Gasoline Price Log Increase Rate",
  major.ticks="month", minor.ticks=NULL,
  grid.ticks.on="months",
  col="red"
)
```

这些极端值为建模增加了困难。

从 x_t 的 ACF 和 PACF 来看，都是快速衰减，比较符合线性时间序列特征。

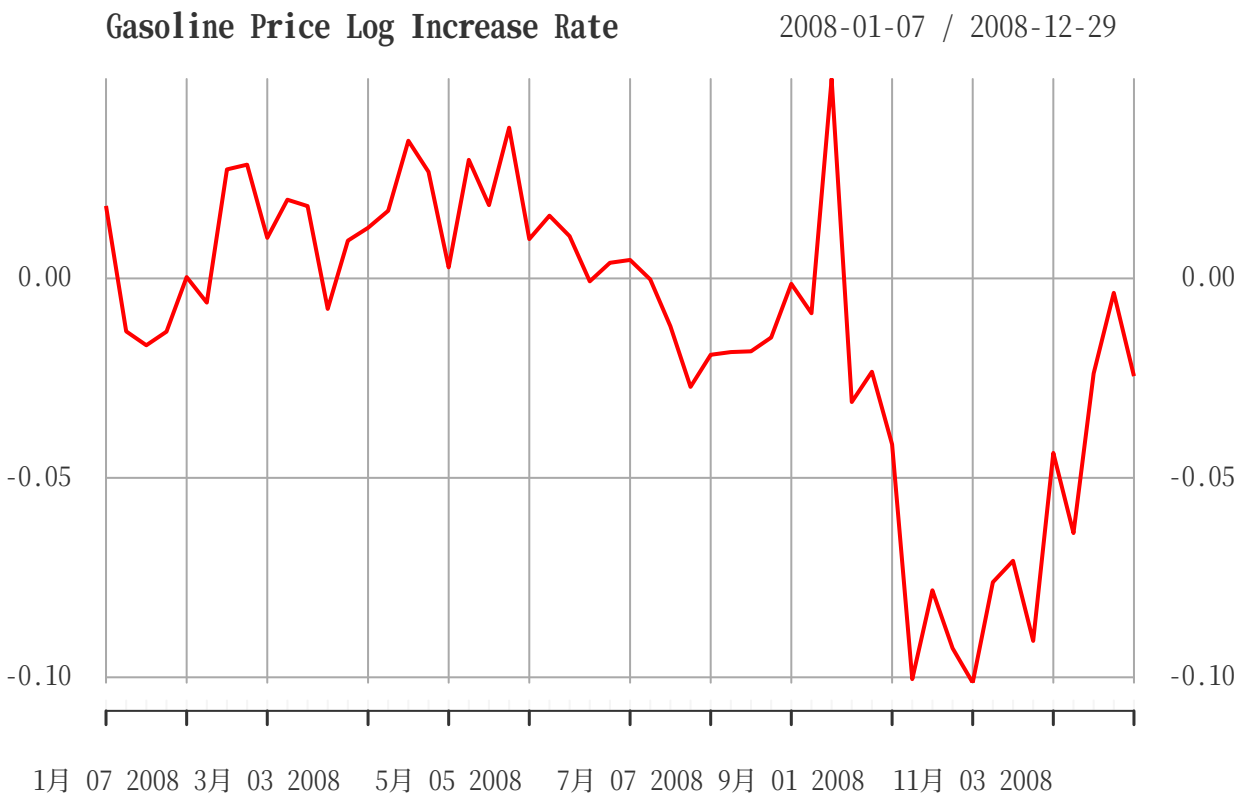


图 13.11: 汽油价格对数增长率序列 2008 年

使用 AIC 对 AR 模型定阶：

```
ar(dpgs, method="mle")
```

```
##
## Call:
## ar(x = dpgs, method = "mle")
##
## Coefficients:
##       1        2        3        4        5
##  0.5068   0.0786   0.1352  -0.0362  -0.0869
##
## Order selected 5  sigma^2 estimated as  0.000326
```

AIC 选择 5 阶，这也与 PACF 的观察结果吻合。

先建立一个含有常数项的 AR(5) 模型，再考虑常数项是否需要：

```
arima(dpgs, order=c(5,0,0), include.mean=TRUE)
```

```
##
## Call:
## arima(x = dpgs, order = c(5, 0, 0), include.mean = TRUE)
##
## Coefficients:
##          ar1     ar2     ar3      ar4      ar5  intercept
##       0.5067  0.0786  0.1352  -0.0362  -0.0867     0.0011
## s.e.  0.0372  0.0417  0.0415   0.0417   0.0372     0.0017
##
## sigma^2 estimated as 0.000326:  log likelihood = 1858.07,  aic = -3702.14
```

从 intercept 的估计值与 SE 的比较来看，均值是不显著的。（如果估计值落入正负二倍 SE 范围就不显著）。所以改为一个不含常数项的 AR(5) 模型：

```
resm3 <- arima(dpgs, order=c(5,0,0), include.mean=FALSE); resm3
```

```
##
## Call:
## arima(x = dpgs, order = c(5, 0, 0), include.mean = FALSE)
##
## Coefficients:
##          ar1     ar2     ar3      ar4      ar5
##       0.5073  0.0788  0.1355  -0.0360  -0.0862
## s.e.  0.0372  0.0417  0.0415   0.0417   0.0372
##
## sigma^2 estimated as 0.0003262:  log likelihood = 1857.85,  aic = -3703.71
```

以上模型为

$$x_t = 0.5073x_{t-1} + 0.0788x_{t-2} + 0.1355x_{t-3} - 0.0360x_{t-4} - 0.0862x_{t-5} + \varepsilon_t,\ \mathrm{Var}(\varepsilon_t) = 0.0003262$$

逐个看 AR 系数的显著性，可以发现 ar4 明显地不显著，ar2 也接近于不显著。删除第 4 个 AR 系数：

```
resm4 <- arima(
  dpgs, order=c(5,0,0), fixed=c(NA, NA, NA, 0, NA),
  include.mean=FALSE); resm4
```

```
## Warning in arima(dpgs, order = c(5, 0, 0), fixed = c(NA, NA, NA, 0, NA), : some
## AR parameters were fixed: setting transform.pars = FALSE
```

```
##
## Call:
## arima(x = dpgs, order = c(5, 0, 0), include.mean = FALSE, fixed = c(NA, NA,
##     NA, 0, NA))
##
## Coefficients:
##          ar1     ar2     ar3  ar4      ar5
##       0.5036  0.0789  0.1220    0  -0.1009
## s.e.  0.0370  0.0418  0.0385    0   0.0330
##
## sigma^2 estimated as 0.0003265:  log likelihood = 1857.48,  aic = -3704.96
```

修改后 AIC 值略有降低。这样限制稀疏模型有可能破坏稳定性条件，所以用 `polyroot()` 函数求特征根，看是否都在单位圆外：

```
abs(polyroot(c(1, -coefficients(resm4))))
```

```
## [1] 1.420831 1.606426 1.606426 1.420831 1.901977
```

特征根都在单位圆外且距离单位圆较远，说明线性时间序列模型是合适的。

如果进一步去掉滞后 2 系数：

```
resm5 <- arima(
  dpgs, order=c(5,0,0), fixed=c(NA, 0, NA, 0, NA),
  include.mean=FALSE); resm5
```

```
## Warning in arima(dpgs, order = c(5, 0, 0), fixed = c(NA, 0, NA, 0, NA), : some
## AR parameters were fixed: setting transform.pars = FALSE
```

```
##
## Call:
## arima(x = dpgs, order = c(5, 0, 0), include.mean = FALSE, fixed = c(NA, 0, NA,
##     0, NA))
##
## Coefficients:
##          ar1  ar2     ar3  ar4      ar5
```

```
##        0.5360     0  0.1513     0  -0.0930
## s.e.   0.0329     0  0.0354     0   0.0328
##
## sigma^2 estimated as 0.0003281:  log likelihood = 1855.7,  aic = -3703.4
```

AIC 值比仅去掉滞后 4 的模型变大了。所以应选择仅去掉滞后 4 的系数。

限制 $\phi_4 = 0$ 的 AR(5) 模型为：

$$x_t = 0.5036x_{t-1} + 0.0789x_{t-2} + 0.1220x_{t-3} - 0.1009x_{t-5} + \varepsilon_t,\ \mathrm{Var}(\varepsilon_t) = 0.0003265$$

称此模型为候选模型 1（MC1）。对此模型做诊断图：

```
tsdiag(resm4, gof=20)
```

从诊断图看，标准化残差仍有较大的异常值。ACF 已经基本符合白噪声要求。第三幅各个 Ljung-Box 白噪声检验均不显著，其中横坐标是检验所用的自相关系数个数，零假设为符合白噪声要求。这些诊断支持了候选模型 1。

13.3 ARMA(1,3) 模型

考虑汽油价格序列对数增长率的其它线性时间序列模型。注意到 PACF 中滞后 1 最显著，其它显著位置是刚刚超出 0.05 水平界限。所以考虑用 AR(1)；但是，因为 ACF 和 PACF 并不在低阶截尾，所以单用 AR 或者单用 MA 可能不够，尝试用 ARMA。

先看 AR(1) 的表现：

```
resm6 <- arima(
  dpgs, order=c(1,0,0),
  include.mean=FALSE)
forecast::Acf(residuals(resm6), lag.max=10, main="")
```

AR(1) 的 ACF 在滞后 3 有一个显著非零。尝试 ARMA(1,3):

```
resm7 <- arima(
  dpgs, order=c(1,0,3),
  include.mean=FALSE); resm7
```

```
##
## Call:
## arima(x = dpgs, order = c(1, 0, 3), include.mean = FALSE)
##
## Coefficients:
##          ar1      ma1     ma2     ma3
##       0.5917  -0.0919  0.0413  0.1547
## s.e.  0.0750   0.0774  0.0489  0.0435
##
## sigma^2 estimated as 0.0003273:  log likelihood = 1856.65,  aic = -3703.29
```

从系数估计值与 SE 的比较可以看出 MA 的系数 1、系数 2 都不显著。先去掉 MA 系数 2，拟合稀疏系数模型：

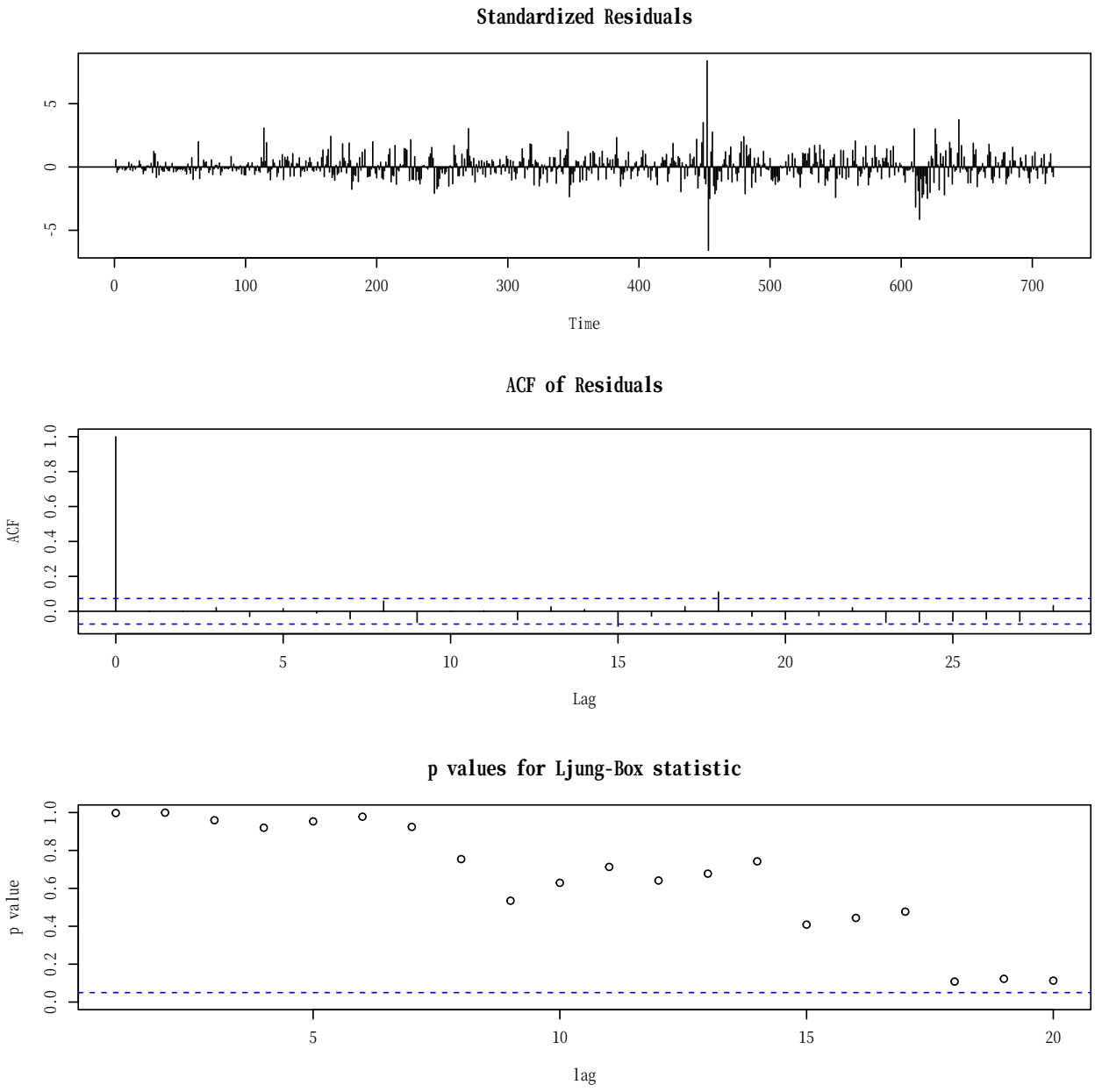


图 13.12: 候选模型 1 诊断

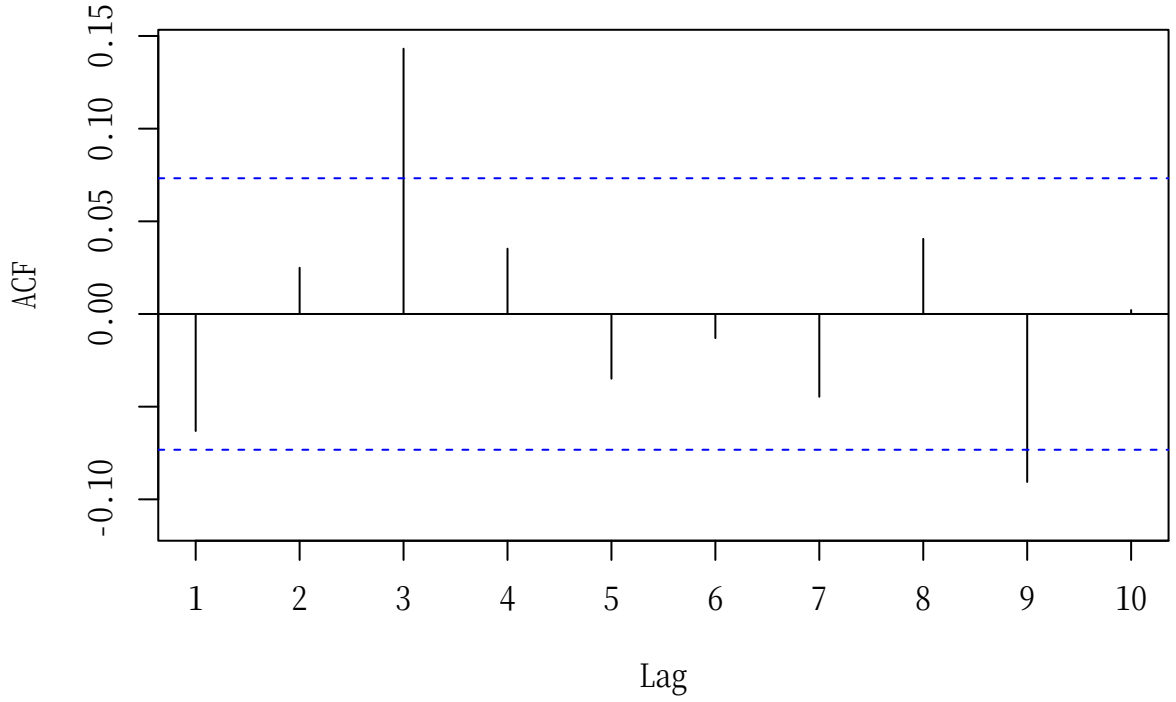


图 13.13: AR(1) 的残差 ACF

```
resm8 <- arima(
  dpgs, order=c(1,0,3), fixed=c(NA, NA, 0, NA),
  include.mean=FALSE); resm8
```

```
##
## Call:
## arima(x = dpgs, order = c(1, 0, 3), include.mean = FALSE, fixed = c(NA, NA,
##     0, NA))
##
## Coefficients:
##          ar1      ma1  ma2     ma3
##       0.6332  -0.1266    0  0.1411
## s.e.  0.0507   0.0595    0  0.0408
##
## sigma^2 estimated as 0.0003276:  log likelihood = 1856.3,  aic = -3704.6
```

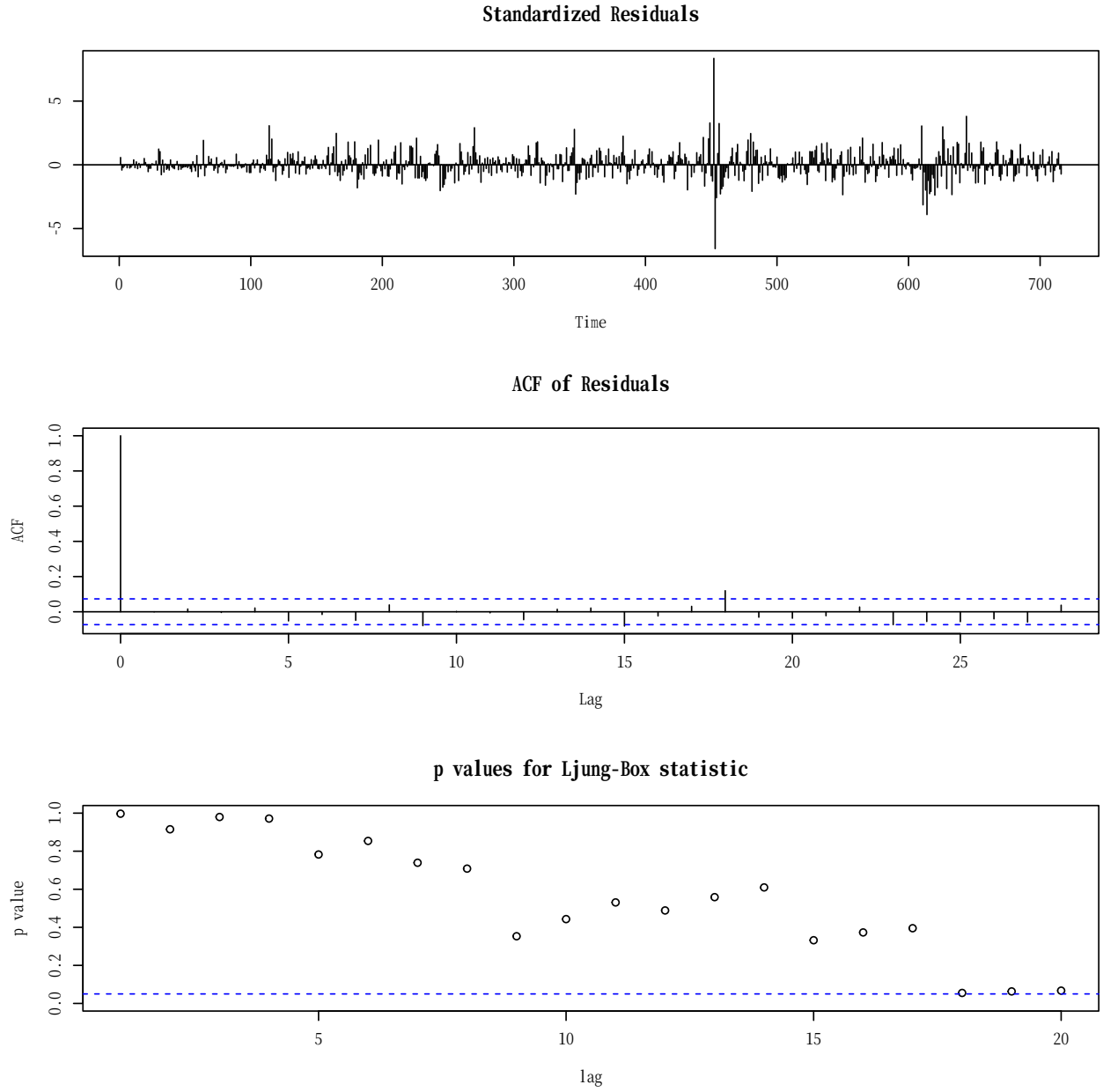
这个模型的 AIC 比候选模型 1 略差一点。MA 系数 1 显著性在两可之间，所以不继续精简。模型为

$$x_t = 0.6332x_{t-1} + \varepsilon_t - 0.1266\varepsilon_{t-1} + 0.1411\varepsilon_{t-3},\ \mathrm{Var}(\varepsilon_t) = 0.0003276$$

称此模型为候选模型 2（MC2）。

对候选模型 2 作诊断图：

```
tsdiag(resm8, gof=20)
```



标准化残差图仍有大异常值。ACF 已经符合白噪声表现；Ljung-Box 白噪声检验基本符合白噪声要求。

因为候选模型 1 和候选模型 2 是类似的模型，候选模型 1 的 AIC 较优，所以两者相比选择候选模型 1。

13.4 固定线性趋势模型

考虑用非随机线性趋势对汽油价格对数值序列建模。

```
tmp.t <- seq(717)
resm9 <- lm(pgs ~ tmp.t)
summary(resm9)
```

```
##
```

```
## Call:
## lm(formula = pgs ~ tmp.t)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.53400 -0.12170 -0.00928  0.12268  0.44298
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.664e-03  1.276e-02  -0.522    0.602
## tmp.t        1.627e-03  3.079e-05  52.848   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1707 on 715 degrees of freedom
## Multiple R-squared:  0.7962, Adjusted R-squared:  0.7959
## F-statistic:  2793 on 1 and 715 DF,  p-value: < 2.2e-16
```

回归结果显著。考察残差序列：

```
plot(residuals(resm9), type="l", xlab="Time", ylab="Residual")
```

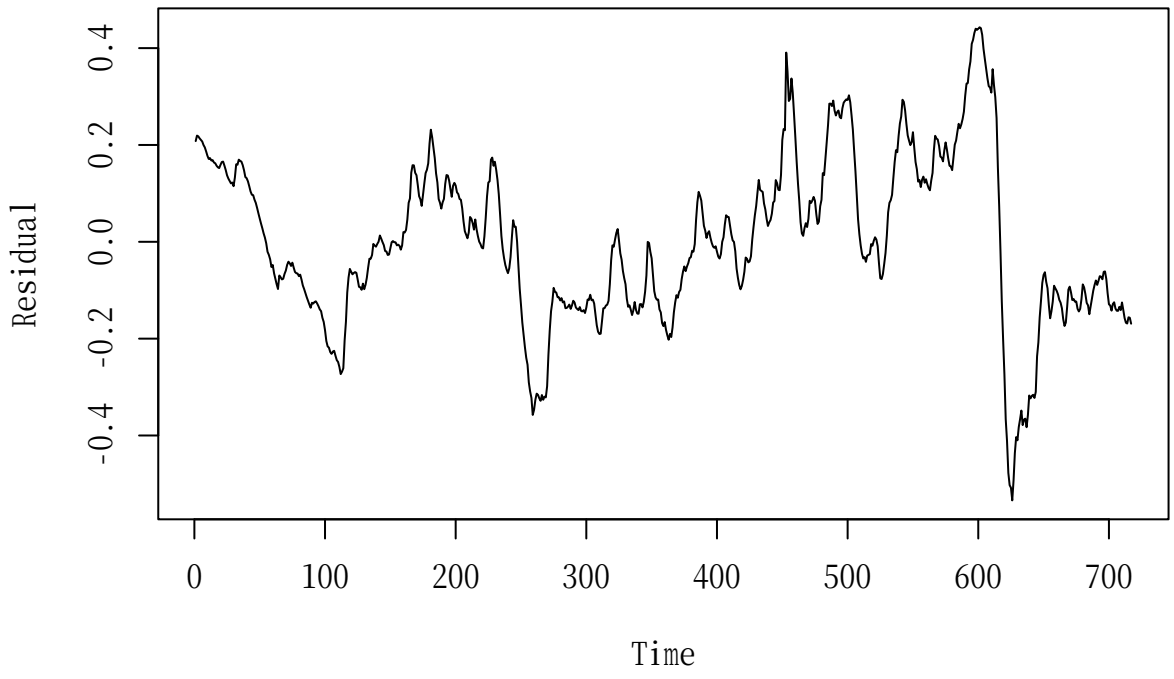


图 13.14: 非随机线性趋势模型残差序列

序列呈现出一定的缓慢随机水平变化，提示非平稳。残差的 ACF 图：

```
acf(residuals(resm9), main="", lag.max=20)
```

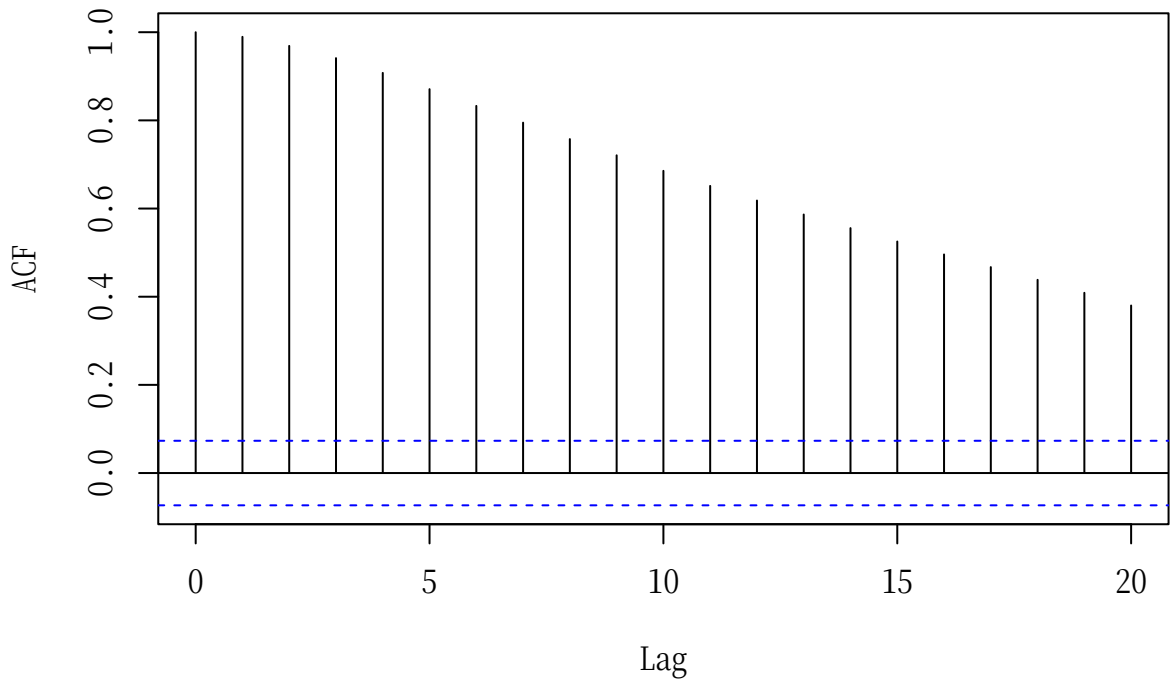


图 13.15: 非随机线性趋势模型残差的 ACF

说明残差序列的 ACF 是缓慢衰减的，不太适用线性时间序列加非随机线性趋势作为模型。

13.5 引入石油价格解释变量的模型

考虑用石油价格对数值作为解释变量，对汽油价格对数值序列建模。设 $\{x_t\}$ 为汽油价格对数增长率，$\{z_t\}$ 为原油价格对数增长率，使用带有误差自相关的回归模型。

先做简单回归：

$$x_t = a + bz_t + \varepsilon_t$$

```
resm10 <- lm(dpgs ~ dpus)
summary(resm10)
```

```
##
## Call:
## lm(formula = dpgs ~ dpus)
```

```
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.076840 -0.009456 -0.000279  0.008804  0.150721
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0006472  0.0006876   0.941    0.347
## dpus        0.2865347  0.0150791  19.002   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01839 on 714 degrees of freedom
## Multiple R-squared:  0.3359, Adjusted R-squared:  0.3349
## F-statistic: 361.1 on 1 and 714 DF,  p-value: < 2.2e-16
```

可以看出截距项不显著。尽管因为残差可能有自相关使得检验不一定精确，因为两个序列都是差分序列，原始序列相关系数很大，所以回归模型没有常数项是正常的。将上述回归模型改为不带截距项的模型：

$$x_t = bz_t + \varepsilon_t$$

```
resm11 <- lm(dpgs ~ -1 + dpus)
summary(resm11)
```

```
##
## Call:
## lm(formula = dpgs ~ -1 + dpus)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.076149 -0.008834  0.000365  0.009441  0.151350
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## dpus  0.28703    0.01507   19.05   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01839 on 715 degrees of freedom
## Multiple R-squared:  0.3366, Adjusted R-squared:  0.3357
## F-statistic: 362.8 on 1 and 715 DF,  p-value: < 2.2e-16
```

查看回归残差的 ACF:

```
forecast::Acf(resm11$residuals, main="")
```

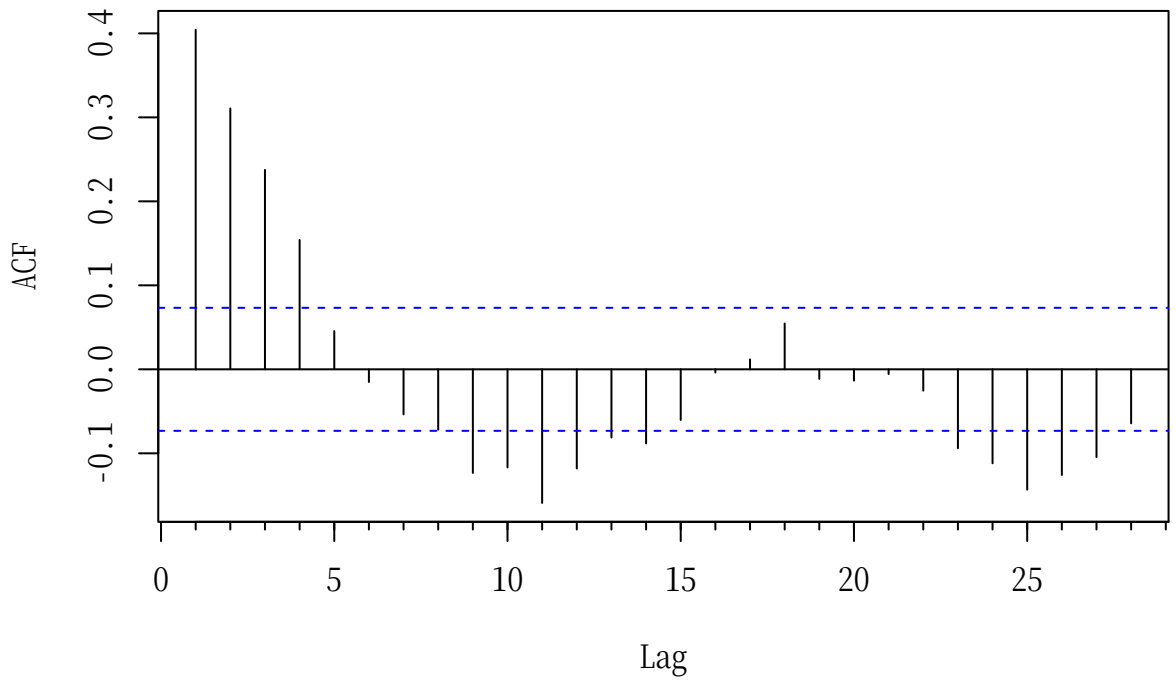


图 13.16: 回归模型的残差序列的 ACF

查看回归残差的 PACF:

```
pacf(resm11$residuals, main="")
```

回归残差的 ACF 和 PACF 都是快速衰减的，但都不截尾。从 PACF 看，基本在第 5 阶之后截尾了。

用 AIC 对残差的 AR 模型定阶：

```
ar(resm11$residuals, method="mle")
```

```
##
## Call:
## ar(x = resm11$residuals, method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6
##  0.3140   0.1575   0.0943   0.0346  -0.0568  -0.0625
##
## Order selected 6  sigma^2 estimated as  0.0002689
```

AIC 选择 6 阶。试用 `arima()` 函数估计 AR(6):

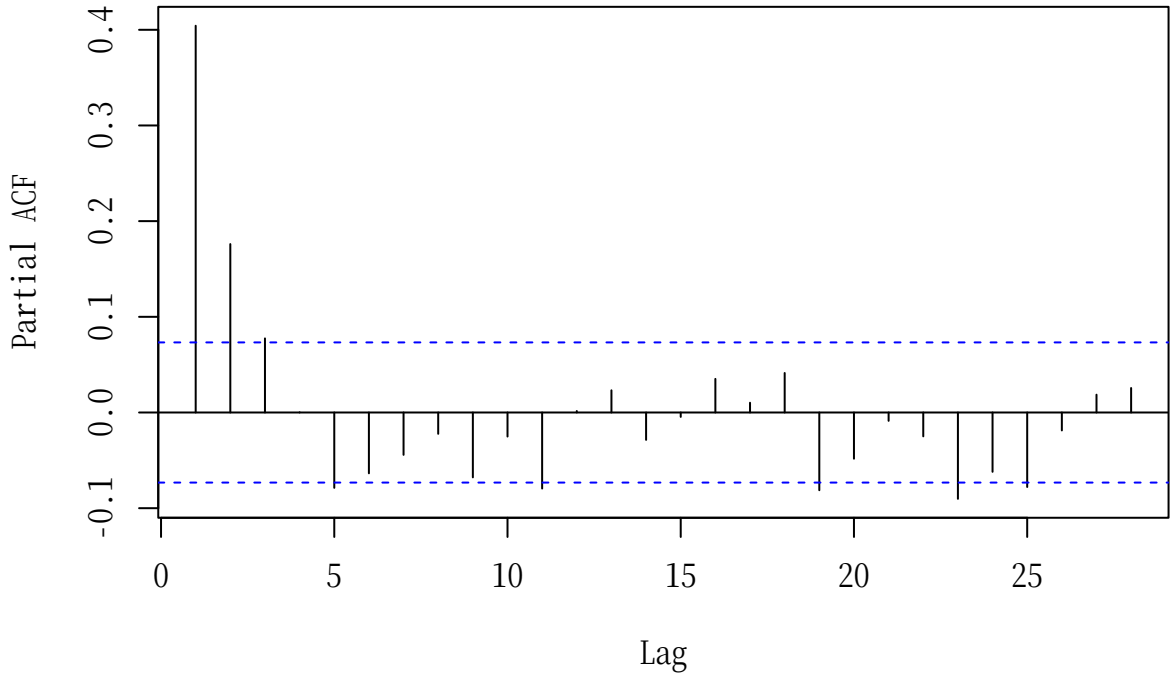


图 13.17: 回归模型的残差序列的 PACF

```
resm12 <- arima(resm11$residuals, order=c(6,0,0))
resm12
```

```
##
## Call:
## arima(x = resm11$residuals, order = c(6, 0, 0))
##
## Coefficients:
##          ar1     ar2     ar3     ar4      ar5      ar6  intercept
##       0.3140  0.1576  0.0946  0.0353  -0.0582  -0.0631     0.0006
## s.e.  0.0372  0.0390  0.0394  0.0394   0.0390   0.0372     0.0012
##
## sigma^2 estimated as 0.0002689:  log likelihood = 1927.08,  aic = -3838.16
```

从估计结果看，均值（intercept）不显著，滞后 6 系数不够显著，滞后 4 系数不显著。先拟合一个不带常数项的 AR(5):

```
resm13 <- arima(resm11$residuals, order=c(5,0,0), include.mean=FALSE)
resm13
```

```
##
## Call:
## arima(x = resm11$residuals, order = c(5, 0, 0), include.mean = FALSE)
##
## Coefficients:
##          ar1     ar2     ar3     ar4      ar5
##       0.3193  0.1563  0.0891  0.0256  -0.0781
## s.e.  0.0372  0.0391  0.0394  0.0391   0.0372
##
## sigma^2 estimated as 0.0002701:  log likelihood = 1925.51,  aic = -3839.02
```

这个精简的模型的 AIC 实际上改善了。因为滞后 4 系数不显著，做一个稀疏系数 AR(5):

```
resm14 <- arima(resm11$residuals, order=c(5,0,0),
                fixed=c(NA, NA, NA, 0, NA), include.mean=FALSE)
```

```
## Warning in arima(resm11$residuals, order = c(5, 0, 0), fixed = c(NA, NA, : some
## AR parameters were fixed: setting transform.pars = FALSE
```

```
resm14
```

```
##
## Call:
## arima(x = resm11$residuals, order = c(5, 0, 0), include.mean = FALSE, fixed = c(NA,
##     NA, NA, 0, NA))
##
## Coefficients:
##          ar1     ar2     ar3  ar4      ar5
##       0.3211  0.1592  0.0954    0  -0.0707
```

```
## s.e.  0.0371  0.0388  0.0382     0   0.0354
##
## sigma^2 estimated as 0.0002702:  log likelihood = 1925.29,  aic = -3840.59
```

AIC 继续有改善。将外生回归变量加入模型中，直接建立带有自相关误差项的回归模型：

```
resm15 <- arima(
  dpgs, xreg=dpus, order=c(5,0,0), include.mean=FALSE,
  fixed=c(NA, NA, NA, 0, NA, NA))
```

```
## Warning in arima(dpgs, xreg = dpus, order = c(5, 0, 0), include.mean = FALSE, :
## some AR parameters were fixed: setting transform.pars = FALSE
```

```
resm15
```

```
##
## Call:
## arima(x = dpgs, order = c(5, 0, 0), xreg = dpus, include.mean = FALSE, fixed = c(NA,
##     NA, NA, 0, NA, NA))
##
## Coefficients:
##          ar1     ar2     ar3  ar4      ar5    dpus
##       0.4037  0.1642  0.0961    0  -0.1014  0.1911
## s.e.  0.0386  0.0399  0.0386    0   0.0345  0.0136
##
## sigma^2 estimated as 0.0002532:  log likelihood = 1948.48,  aic = -3884.95
```

这样建立的模型可以写成：

$$
\begin{aligned}
x_t =& 0.1911 z_t + \xi_t, \\
\xi_t =& 0.4037\xi_{t-1} + 0.1642\xi_{t-2} + 0.0961\xi_{t-3} - 0.1014\xi_{t-5} + \varepsilon_t,\ \mathrm{Var}(\varepsilon_t) = 0.0002532
\end{aligned}
$$

称此模型为候选模型 3（MC3）。模型的诊断图形：

```
tsdiag(resm15, gof=20)
```

从诊断图看，除了标准化残差还有大异常值以外，残差的白噪声性质已经基本确认。说明候选模型 3 是合适的。

将候选模型 1 与候选模型 3 进行比较。这两个模型的比较是不公平的，因为候选模型 3 利用了额外的原油价格信息。候选模型 1 的残差方差估计是 0.0003281，AIC 值是 −3703.4；候选模型 3 的残差方差估计是 0.0002532，AIC 值是 −3884.95。模型 3 的模型方差估计降低了 23%，AIC 也有明显改善。后面可以看到模型 3 的预测效果也更好。

13.6 使用滞后石油价格解释变量的模型

数据中原油价格比汽油价格发布时间早 3 天，这样，利用回归模型预报时，仅能超前 1 到 3 天预报，超前 4 天时就没有对应的石油价格作为解释变量了。

考虑使用石油价格的前一个发布值作为解释变量，这样石油价格比汽油价格提前 10 天。

简单回归模型：

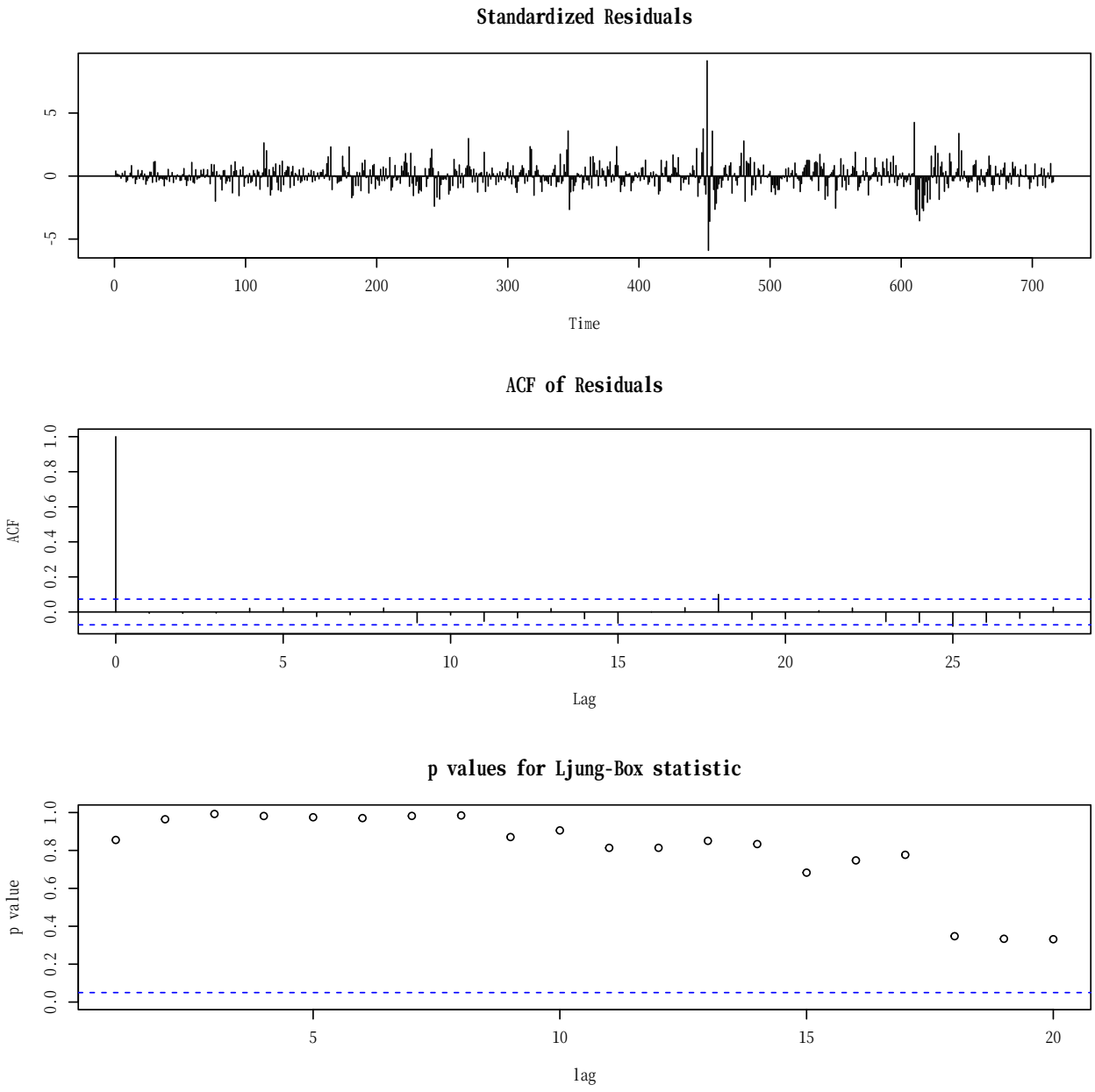


图 13.18: 候选模型 3 的诊断图形

$$x_t = bz_{t-1} + \varepsilon_t$$

```
dpus1 <- dpus[-length(dpus)]
dpgs1 <- dpgs[-1]
resm16 <- lm(dpgs1 ~ -1 + dpus1)
summary(resm16)
```

```
##
## Call:
## lm(formula = dpgs1 ~ -1 + dpus1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.088318 -0.011145 -0.000011  0.011391  0.161679
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## dpus1  0.18560    0.01716   10.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02093 on 714 degrees of freedom
## Multiple R-squared:  0.1408, Adjusted R-squared:  0.1395
## F-statistic:   117 on 1 and 714 DF,  p-value: < 2.2e-16
```

注意这个回归 $R^2 = 0.14$ 比间隔 3 天的回归的 $R^2 = 0.33$ 小了许多，预示着这个模型的预测效果远不如候选模型 3。

对回归残差作 ACF 图：

```
forecast::Acf(resm16$residuals, main="")
```

对回归残差作 PACF 图：

```
pacf(resm16$residuals, main="")
```

从残差的 ACF 和 PACF 看，虽然都是快速衰减的，但都不在低阶截尾。

用 AIC 为残差序列作 AR 定阶：

```
ar(resm16$residuals, method="mle")
```

```
##
## Call:
## ar(x = resm16$residuals, method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.2851   0.0805   0.2343  -0.0411  -0.0169  -0.0284  -0.0652   0.0577
```

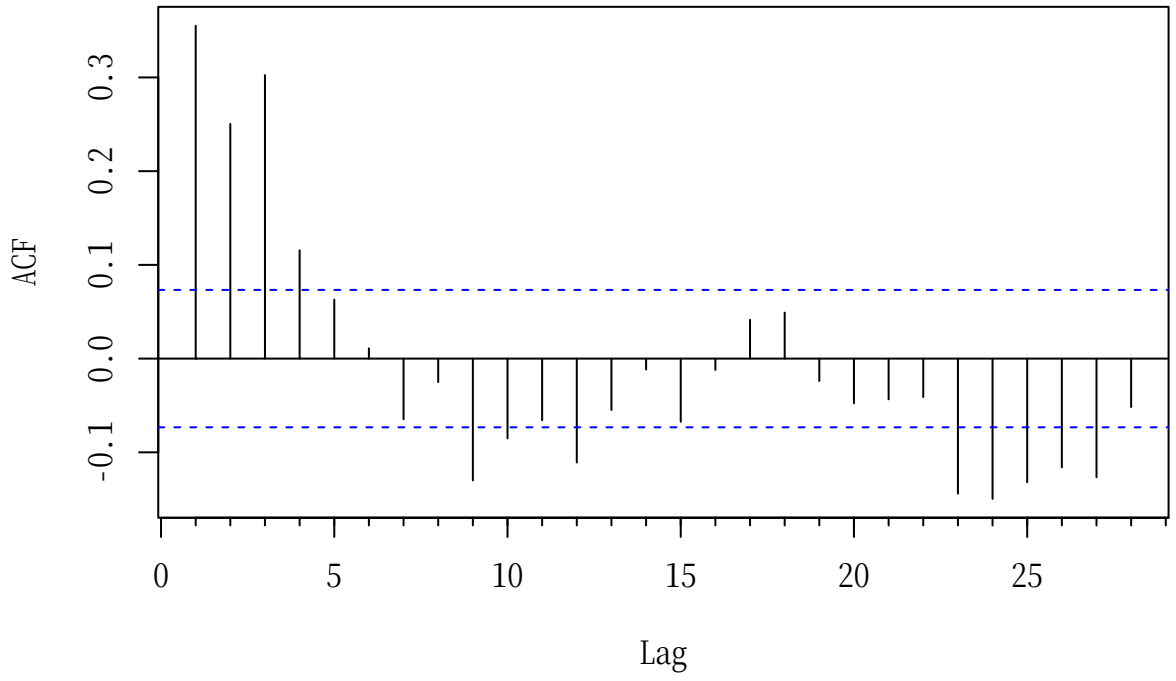


图 13.19: 滞后回归的残差 ACF

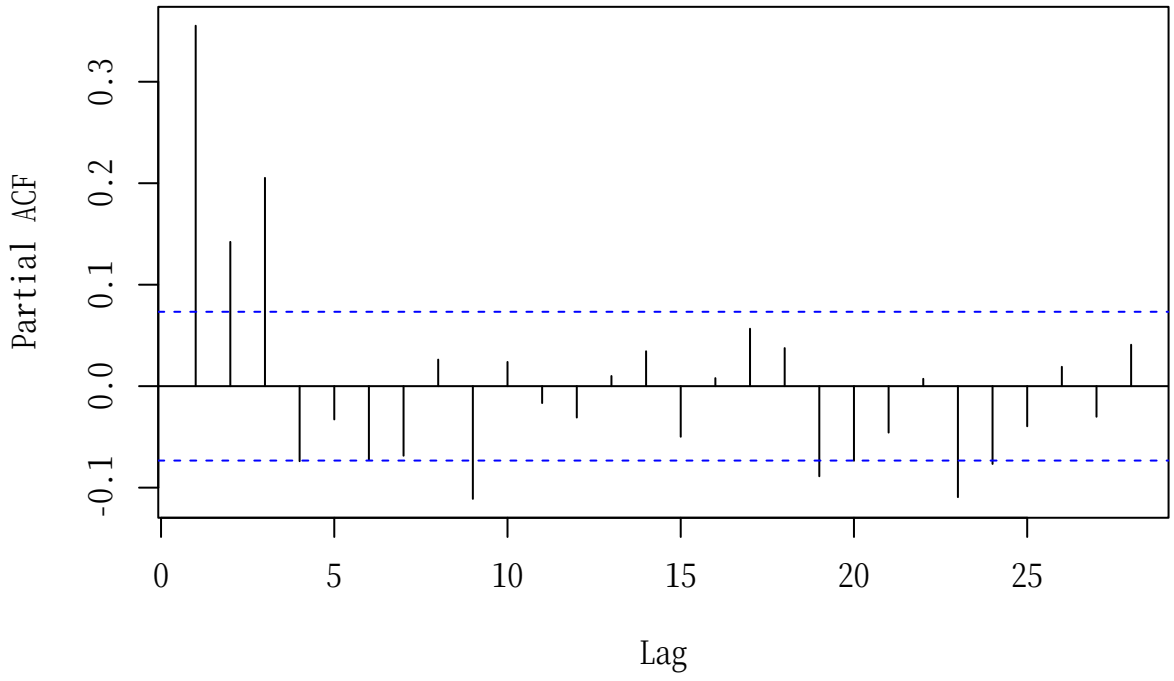


图 13.20: 滞后回归的残差 PACF

```
##       9
## -0.1101
##
## Order selected 9  sigma^2 estimated as  0.0003479
```

选择 9 阶。用 arima() 估计：

```
resm17 <- arima(
  dpgs1, xreg=dpus1, include.mean=FALSE,
  order=c(9,0,0))
resm17
```

```
##
## Call:
## arima(x = dpgs1, order = c(9, 0, 0), xreg = dpus1, include.mean = FALSE)
##
## Coefficients:
##          ar1     ar2     ar3      ar4      ar5      ar6      ar7     ar8
##       0.4559  0.0888  0.1679  -0.0468  -0.0653  -0.0195  -0.0362  0.0797
## s.e.  0.0425  0.0410  0.0423   0.0415   0.0416   0.0414   0.0410  0.0408
##           ar9   dpus1
##       -0.0882  0.0454
## s.e.   0.0373  0.0174
##
## sigma^2 estimated as 0.0003204:  log likelihood = 1861.55,  aic = -3701.1
```

查看各个系数的显著性，用估计值除以标准误差得到检验统计量，在等于零的零假设下近似服从标准正态分布：

```
arima.tstat <- function(mod) coef(mod)[coef(mod) != 0] / sqrt(diag(mod$var.coef))
round(sort(abs(arima.tstat(resm17))), 2)
```

```
##   ar6   ar7   ar4   ar5   ar8   ar2   ar9 dpus1   ar3   ar1
##  0.47  0.88  1.13  1.57  1.95  2.16  2.36  2.61  3.97 10.73
```

滞 6, 7, 4, 5, 8 的系数不显著。逐步进行改进，先去掉 ϕ_6：

```
fixed <- rep(NA_real_, 10)
fixed[6] <- 0
resm18 <- arima(
  dpgs1, xreg=dpus1, include.mean=FALSE,
  order=c(9,0,0), fixed=fixed)
```

```
## Warning in arima(dpgs1, xreg = dpus1, include.mean = FALSE, order = c(9, : some
## AR parameters were fixed: setting transform.pars = FALSE
```

```
resm18
```

```
##
## Call:
```

```
## arima(x = dpgs1, order = c(9, 0, 0), xreg = dpus1, include.mean = FALSE, fixed = fixed)
##
## Coefficients:
##          ar1     ar2     ar3      ar4      ar5  ar6      ar7     ar8      ar9
##       0.4574  0.0889  0.1646  -0.0473  -0.0718    0  -0.0423  0.0794  -0.0908
## s.e.  0.0424  0.0410  0.0417   0.0415   0.0393    0   0.0389  0.0408   0.0369
##        dpus1
##       0.0451
## s.e.  0.0174
##
## sigma^2 estimated as 0.0003205:  log likelihood = 1861.44,  aic = -3702.88
```

```
cat("\n\n")
```

```
round(sort(abs(arima.tstat(resm18))), 2)
```

```
##   ar7   ar4   ar5   ar8   ar2   ar9 dpus1   ar3   ar1
##  1.09  1.14  1.83  1.95  2.17  2.46  2.60  3.95 10.78
```

AIC 有改进。再去掉 ϕ_7:

```
fixed <- rep(NA_real_, 10)
fixed[c(6,7)] <- 0
resm19 <- arima(
  dpgs1, xreg=dpus1, include.mean=FALSE,
  order=c(9,0,0), fixed=fixed)
```

```
## Warning in arima(dpgs1, xreg = dpus1, include.mean = FALSE, order = c(9, : some
## AR parameters were fixed: setting transform.pars = FALSE
```

```
resm19
```

```
##
## Call:
## arima(x = dpgs1, order = c(9, 0, 0), xreg = dpus1, include.mean = FALSE, fixed = fixed)
##
## Coefficients:
##          ar1     ar2     ar3     ar4     ar5  ar6  ar7     ar8      ar9   dpus1
##       0.4618  0.0903  0.1617  -0.055  -0.078    0    0  0.0634  -0.0961  0.0431
## s.e.  0.0422  0.0411  0.0417   0.041   0.039    0    0  0.0381   0.0367  0.0172
##
## sigma^2 estimated as 0.0003211:  log likelihood = 1860.85,  aic = -3703.7
```

```
round(sort(abs(arima.tstat(resm19))), 2)
```

```
##   ar4   ar8   ar5   ar2 dpus1   ar9   ar3   ar1
##  1.34  1.66  2.00  2.20  2.50  2.62  3.87 10.94
```

AIC 有改进。再去掉 ϕ_4:

```
fixed <- rep(NA_real_, 10)
fixed[c(6,7,4)] <- 0
resm20 <- arima(
  dpgs1, xreg=dpus1, include.mean=FALSE,
  order=c(9,0,0), fixed=fixed)
```

```
## Warning in arima(dpgs1, xreg = dpus1, include.mean = FALSE, order = c(9, : some
## AR parameters were fixed: setting transform.pars = FALSE
```

```
resm20
```

```
##
## Call:
## arima(x = dpgs1, order = c(9, 0, 0), xreg = dpus1, include.mean = FALSE, fixed = fixed)
##
## Coefficients:
##          ar1     ar2     ar3  ar4      ar5  ar6  ar7     ar8      ar9   dpus1
##       0.4580  0.0894  0.1414    0  -0.0989    0    0  0.0604  -0.0938  0.0403
## s.e.  0.0425  0.0412  0.0392    0   0.0358    0    0  0.0381   0.0367  0.0173
##
## sigma^2 estimated as 0.0003219:  log likelihood = 1859.96,  aic = -3703.91
```

```
round(sort(abs(arima.tstat(resm20))), 2)
```

```
##   ar8   ar2 dpus1   ar9   ar5   ar3   ar1
##  1.59  2.17  2.33  2.55  2.76  3.60 10.78
```

AIC 有改进。再去掉 ϕ_8:

```
fixed <- rep(NA_real_, 10)
fixed[c(6,7,4,8)] <- 0
resm21 <- arima(
  dpgs1, xreg=dpus1, include.mean=FALSE,
  order=c(9,0,0), fixed=fixed)
```

```
## Warning in arima(dpgs1, xreg = dpus1, include.mean = FALSE, order = c(9, : some
## AR parameters were fixed: setting transform.pars = FALSE
```

```
resm21
```

```
##
## Call:
## arima(x = dpgs1, order = c(9, 0, 0), xreg = dpus1, include.mean = FALSE, fixed = fixed)
##
## Coefficients:
##          ar1     ar2     ar3  ar4      ar5  ar6  ar7  ar8      ar9   dpus1
##       0.4544  0.0877  0.1415    0  -0.0830    0    0    0  -0.0640  0.0406
## s.e.  0.0427  0.0413  0.0393    0   0.0345    0    0    0   0.0318  0.0176
```

```
##
## sigma^2 estimated as 0.000323:  log likelihood = 1858.7,  aic = -3703.4
```

```
round(sort(abs(arima.tstat(resm21))), 2)
```

```
##   ar9   ar2 dpus1   ar5   ar3   ar1
##  2.01  2.13  2.31  2.41  3.60 10.63
```

AIC 变差，所以退回到上一个模型，以 AR(9) 去掉滞后 4、6、7 系数的模型为候选模型 4（MC4）。注意，这个模型肯定不如模型 3，因为作为自变量的原油价格比汽油价格的发布时间早了 10 天。

13.7 样本外预测

下面比较 MC1、MC3、MC4 的样本外预测，对最后的 400 个样本点作超前一步预测。将这三个模型分别记为 M1、M2、M3。

$$
\begin{aligned}
M1: & x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \phi_3 x_{t-3} + \phi_5 x_{t-5} + \varepsilon_t \\
M2: & x_t = b z_t + \xi_t,\ \xi_t = \phi_1 \xi_{t-1} + \phi_2 \xi_{t-2} + \phi_3 \xi_{t-3} + \phi_5 \xi_{t-5} + \varepsilon_t \\
M3: & x_t = b z_{t-1} + \xi_t,\ \xi_t = \phi_1 \xi_{t-1} + \phi_2 \xi_{t-2} + \phi_3 \xi_{t-3} + \phi_5 \xi_{t-5} + \phi_8 \xi_{t-8} + \phi_9 \xi_{t-9} + \varepsilon_t
\end{aligned}
$$

为了统一起见，所有序列都放弃第一个观测。这样，每个序列有 715 个观测，对最后的 400 个做超前一步预测，最初仅使用 315 个观测建模。

```
dpuslag1 <- dpus[-length(dpus)]
dpus1 <- dpus[-1]
dpgs1 <- dpgs[-1]
nmin <- 316
nmax <- length(dpgs1)
fcst <- matrix(NA_real_, nmax, 3)
fixed <- rep(NA_real_, 10)
fixed[c(4,6,7)] <- 0
for(t1 in nmin:nmax){
  # t1 是要预测的时间点
  n <- t1-1 # 用来建模的序列长度
  # 利用 1 到 n 的样本建立三个模型
  m1 <- arima(dpgs1[1:n], order=c(5,0,0), include.mean=FALSE,
              fixed=c(NA, NA, NA, 0, NA), transform.pars=FALSE)
  m2 <- arima(dpgs1[1:n], xreg=dpus1[1:n],
              order=c(5,0,0), include.mean=FALSE,
              fixed=c(NA, NA, NA, 0, NA, NA), transform.pars=FALSE)
  m3 <- arima(dpgs1[1:n], xreg=dpuslag1[1:n],
              order=c(9,0,0), include.mean=FALSE,
              fixed=fixed, transform.pars=FALSE)

  # 分别获得超前一步预报
```

```
  fcst[t1,1] <- c(predict(m1, n.ahead=1, newxreg=NULL, se.fit=FALSE))
  fcst[t1,2] <- c(predict(m2, n.ahead=1, newxreg=dpus1[t1], se.fit=FALSE))
  fcst[t1,3] <- c(predict(m3, n.ahead=1, newxreg=dpuslag1[t1], se.fit=FALSE))
}
# 计算 RMSFE 和 MAFE
rmsfe <- rep(0.0, 3)
mafe <- rep(0.0, 3)
for(j in 1:3){
  rmsfe[j] <- sqrt( mean((fcst[nmin:nmax, j] - dpgs1[nmin:nmax])^2) )
  mafe[j] <- mean(abs(fcst[nmin:nmax, j] - dpgs1[nmin:nmax]))
}
rmsfe
```

```
## [1] 0.02171326 0.01925884 0.02169466
```

```
mafe
```

```
## [1] 0.01537896 0.01285303 0.01554937
```

结果的超前一步预测根均方误差和平均绝对预测误差列表如下：

```
tab1 <- data.frame(
  " 模型"=c("M1", "M2", "M3"),
  RMSFE = round(rmsfe, 5), MAFE=round(mafe, 5))
knitr::kable(tab1)
```

模型	RMSFE	MAFE
M1	0.02171	0.01538
M2	0.01926	0.01285
M3	0.02169	0.01555

看一看同期的汽油价格对数增长率的绝对值的分布：

```
summary(abs(dpgs1[nmin:nmax]))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.000000 0.006915 0.014015 0.018875 0.025504 0.162002
```

有一半的对数增长率在 0.014 以下，但是最好的 M2 模型的预报的平均绝对误差也达到 0.013。这提示模型预报效果不够理想。

三个模型比较，利用提前 3 天的石油价格的模型 M2 预测效果最好，不利用石油价格的 M1 和利用 10 天前石油价格的 M3 效果相近，说明利用 10 天前的石油价格作用不大。

M1 模型的预报图，黑色为真实值，红色为预报值：

```
xts.p1 <- xts(cbind(dpgs1[nmin:nmax], fcst[nmin:nmax,1]),
              index(xts.pgs)[-(1:2)][nmin:nmax])
plot(xts.p1, type="l",
     main="One-step ahead forecaste with M1",
```

```
     major.ticks="years", minor.ticks=NULL,
     col=c("black", "red"))
```

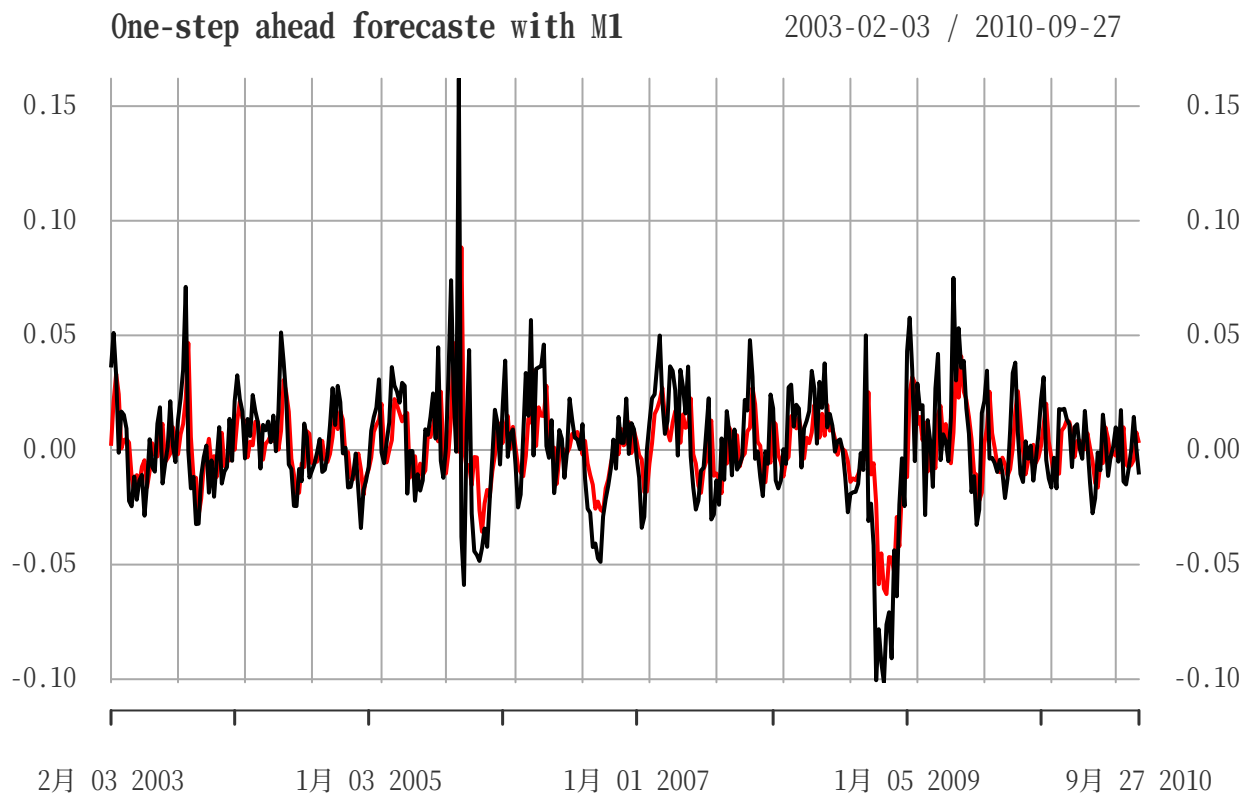


图 13.21: 汽油价格对数增长率用 M1 模型做一步预测

M2 模型的预报图，黑色为真实值，红色为预报值：

```
xts.p2 <- xts(cbind(dpgs1[nmin:nmax], fcst[nmin:nmax,2]),
              index(xts.pgs)[-(1:2)][nmin:nmax])
plot(xts.p2, type="l",
     main="One-step ahead forecaste with M2",
     major.ticks="years", minor.ticks=NULL,
     col=c("black", "red"))
```

如果从图形看，可以看出超前一步预测还是有一定效果的。

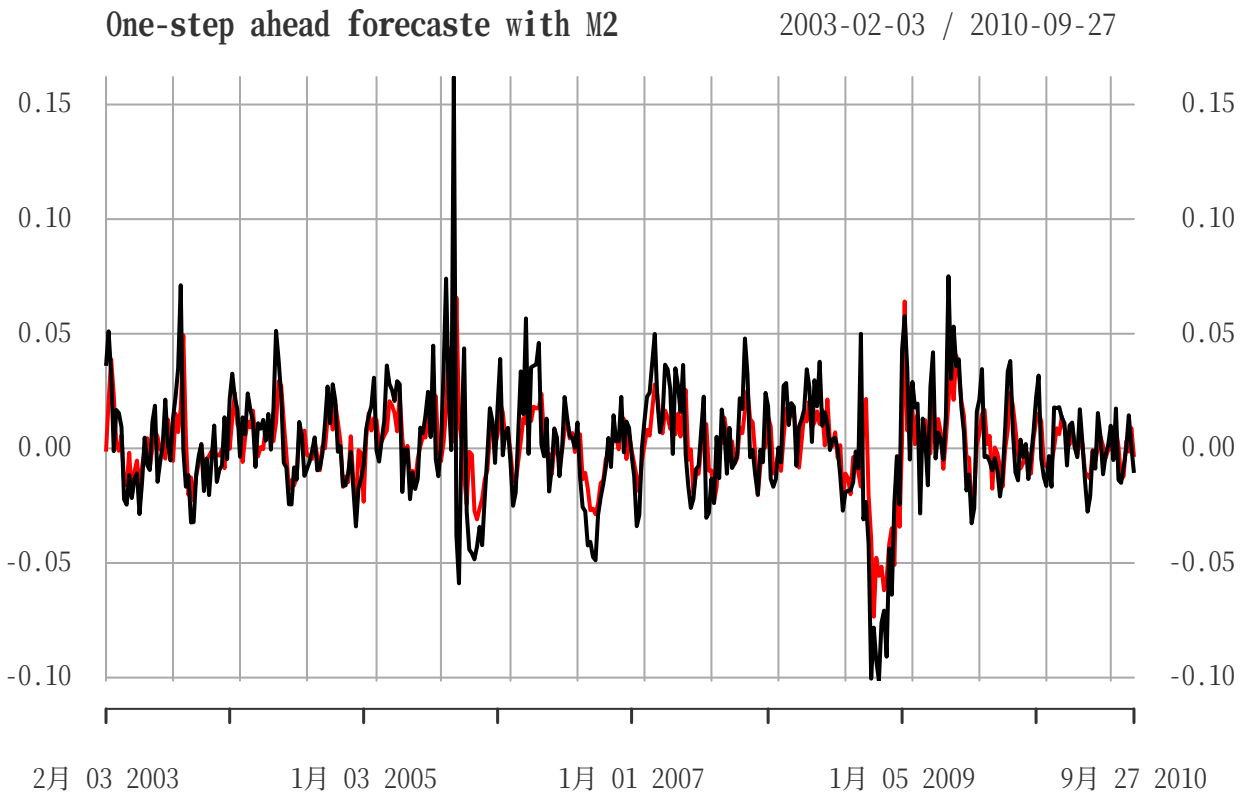


图 13.22: 汽油价格对数增长率用 M2 模型做一步预测

Chapter 14

线性时间序列案例学习—全球温度异常值

用时间序列方法对全球温度异常建模，目的不是证明全球变暖，而是：

- 演示线性时间序列模型的建模和预测方法；
- 比较不同的模型；
- 了解时间序列模型长期预测的局限性；
- 理解仅根据数据区分非随机趋势与单位根非平稳的困难。

全球温度异常值的数据来源有：

- GISS, Goddard Institute for Space Studies 隶属于 NASA(National Aeronautics and Space Administration)。
- NCDC，National Climatic Data Center 隶属于 NOAA(National Oceanic and Atmospheric Administation)。

这里使用的数据来自 GISS，NCDC 的数据也可以得到类似结果。数据为地面空气温度异常值的月度平均值，从 1880 年 1 月到 2010 年 8 月，单位是 0.01 摄氏度。

14.1 数据读入与探索性分析

读入数据，保存为 ts 时间序列对象：

```
Gtemp <- ts(
  scan("m-GLBTs.txt", quiet=TRUE),
  start=c(1880,1), frequency=12)
dG <- diff(Gtemp)
```

时间序列图：

```
plot(Gtemp, xlab="Year", ylab="Temperature", type="l")
```

可以看到从 1980 年代开始有明显的上升趋势。序列的波动性（方差）未见变大。

温度序列的 ACF:

```
acf(Gtemp, lag=36, main="")
```

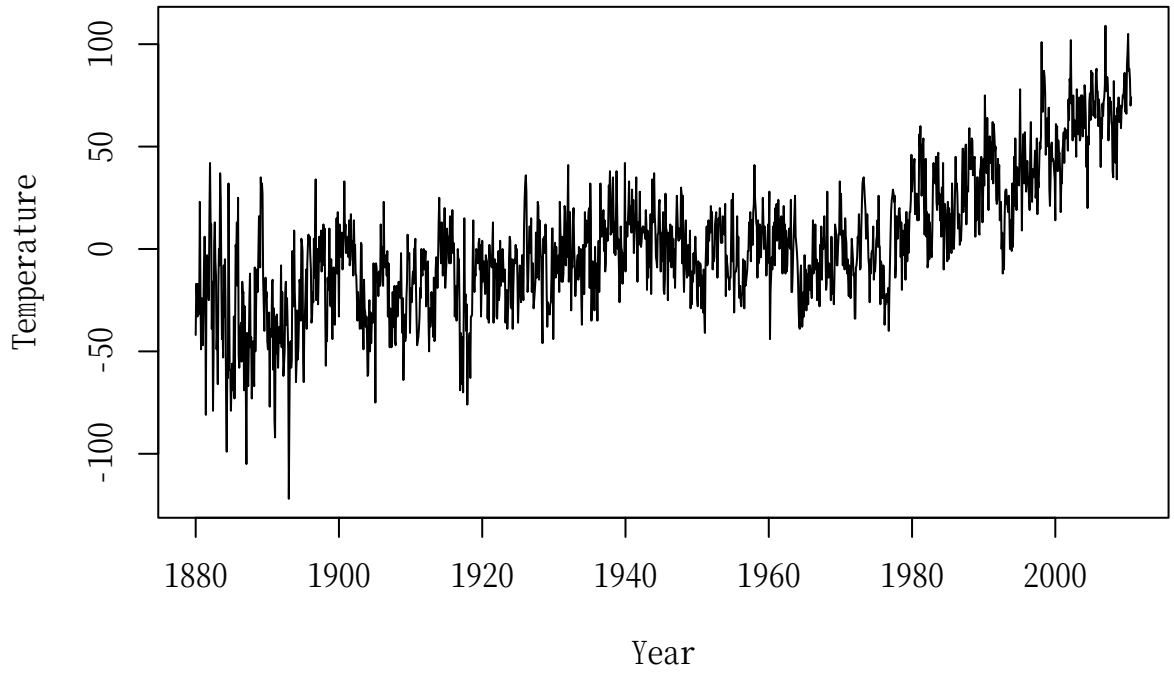


图 14.1: 全球温度异常值月度时间序列

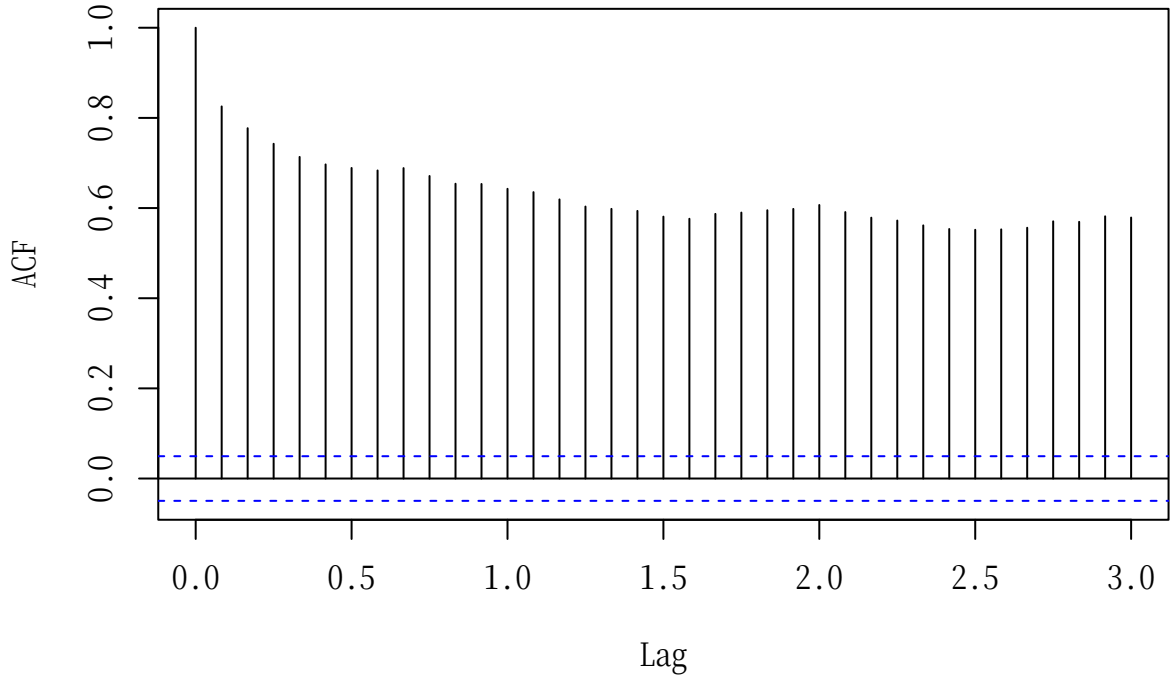


图 14.2: 全球温度异常值的 ACF

这样的 ACF 是缓慢下降的，一般为单位根非平稳或者有接近 1 的特征根的 ARMA 模型。

差分序列的时间序列图：

```
plot(dG, xlab="Year", ylab="diff(Temperature)", type="l")
```

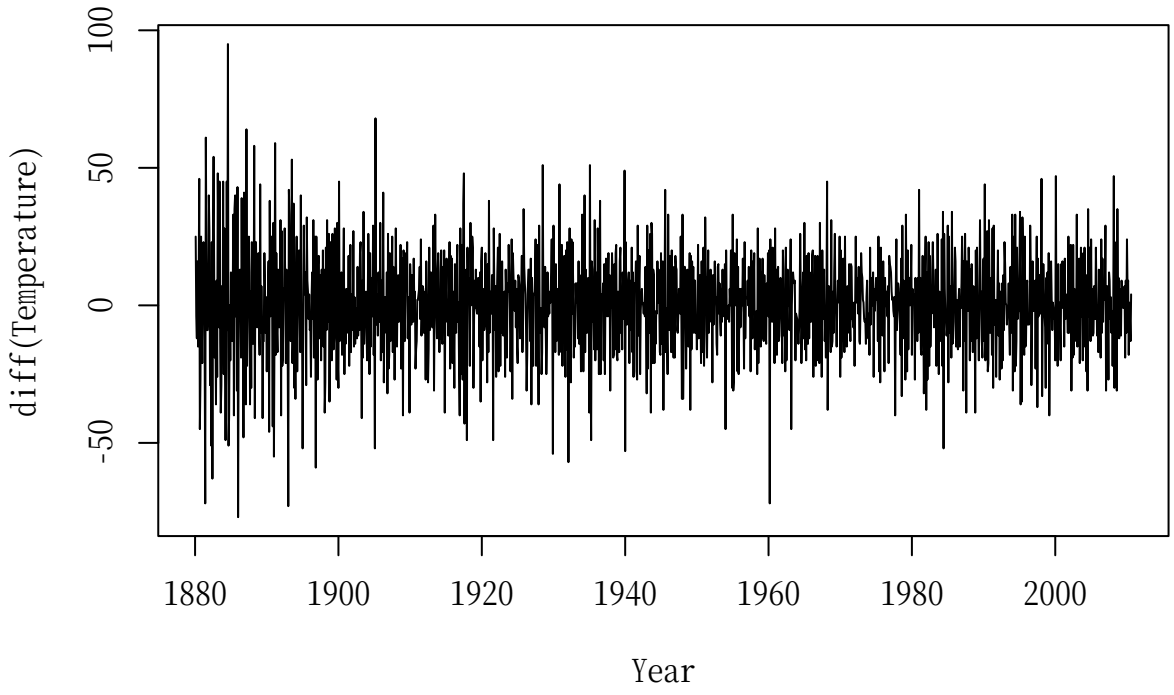


图 14.3: 全球温度异常值的差分序列

差分序列的 ACF:

```
acf(dG, lag=36, main="")
```

ACF 是快速衰减的，只有滞后 1 是明显显著的，其它滞后位置即使超过了 0.05 水平界限也仅是刚刚超过。可以考虑 MA(1) 模型。

差分序列的 PACF:

```
pacf(dG, lag=36, main="")
```

PACF 是快速衰减的，从 PACF 看不能用低阶 AR 建模。

14.2 单位根非平稳模型

做单位根检验：

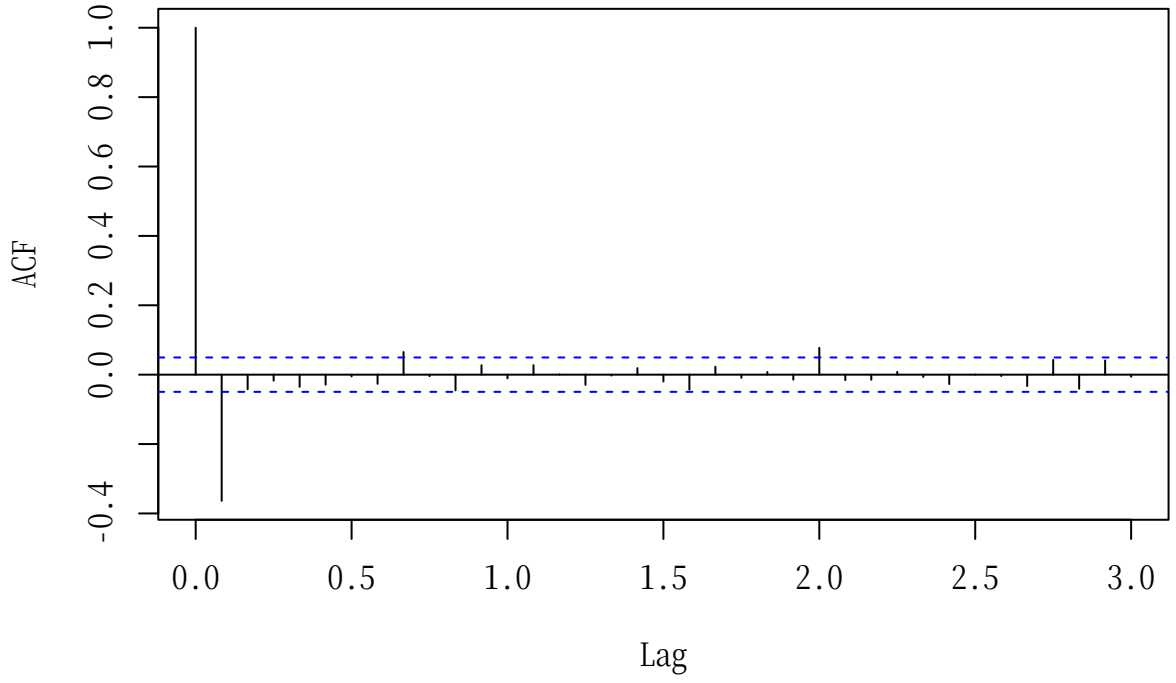


图 14.4: 全球温度异常值差分序列的 ACF

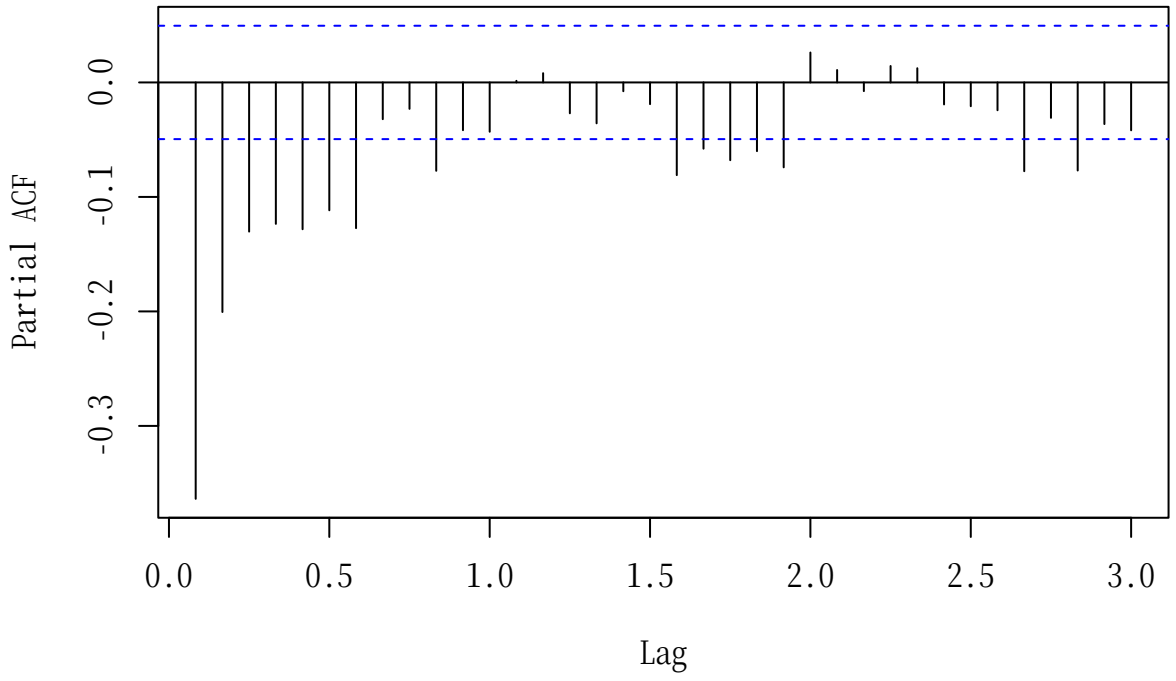


图 14.5: 全球温度异常值差分序列的 PACF

```
ar(dG, method="mle")
```

```
##
## Call:
## ar(x = dG, method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
## -0.5435  -0.3826  -0.3091  -0.2942  -0.2819  -0.2409  -0.2035  -0.0986
##       9       10       11       12
## -0.0892  -0.1048  -0.0658  -0.0425
##
## Order selected 12  sigma^2 estimated as  275.1
```

```
fUnitRoots::adfTest(Gtemp, lags=12, type="c")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 12
##   STATISTIC:
##     Dickey-Fuller: -2.7236
##   P VALUE:
##     0.07438
##
## Description:
##  Tue May 30 17:02:58 2023 by user: Lenovo
```

```
fUnitRoots::adfTest(Gtemp, lags=12, type="ct")
```

```
## Warning in fUnitRoots::adfTest(Gtemp, lags = 12, type = "ct"): p-value smaller
## than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 12
##   STATISTIC:
##     Dickey-Fuller: -5.5578
##   P VALUE:
```

```
##     0.01
##
## Description:
##  Tue May 30 17:02:58 2023 by user: Lenovo
```

检验结果不确定是否有单位根。

因为序列有明显趋势，而且 ACF 衰减缓慢，考虑用单位根非平稳模型，计算序列的差分。记 $\{G_t\}$ 为温度异常值序列，对 $\Delta G_t = G_t - G_{t-1}$ 建模。

用 AIC 作 AR 定阶：

```
ar(dG, method="mle")
```

```
##
## Call:
## ar(x = dG, method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
## -0.5435  -0.3826  -0.3091  -0.2942  -0.2819  -0.2409  -0.2035  -0.0986
##       9       10       11       12
## -0.0892  -0.1048  -0.0658  -0.0425
##
## Order selected 12  sigma^2 estimated as  275.1
```

AIC 给出 12 阶，因为是月度数据，所以滞后 12 出现是合理的。考察 AR(12) 的系数估计的 SE:

```
resm <- arima(Gtemp, order=c(12,1,0)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(12, 1, 0))
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##       -0.5359  -0.3774  -0.3081  -0.2924  -0.2773  -0.2366  -0.1995  -0.0969
## s.e.   0.0253   0.0286   0.0300   0.0309   0.0317   0.0321   0.0321   0.0317
##           ar9     ar10     ar11     ar12
##       -0.0913  -0.1153  -0.0644  -0.0422
## s.e.   0.0310   0.0301   0.0287   0.0253
##
## sigma^2 estimated as 275.2:  log likelihood = -6625.17,  aic = 13276.35
```

AR(12) 的模型无法精简。这个模型的阶数太高了。

考虑 MA 模型。MA(1):

```
resm <- arima(Gtemp, order=c(0,1,1)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(0, 1, 1))
##
## Coefficients:
##           ma1
##       -0.6038
## s.e.   0.0289
##
## sigma^2 estimated as 289.3:  log likelihood = -6664.07,  aic = 13332.14
```

MA(1) 模型的 AIC 为 13332.14，不如 AR(12) 的 13276.33。考虑 MA(2):

```
resm <- arima(Gtemp, order=c(0,1,2)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(0, 1, 2))
##
## Coefficients:
##           ma1      ma2
##       -0.5511  -0.1765
## s.e.   0.0241   0.0260
##
## sigma^2 estimated as 281.2:  log likelihood = -6641.88,  aic = 13289.76
```

MA(2) 的 AIC 为 13289.76, 还是不如 AR(12) 的 13276.33。

考虑 MA(12):

```
resm <- arima(Gtemp, order=c(0,1,12)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(0, 1, 12))
##
## Coefficients:
##           ma1      ma2      ma3      ma4      ma5      ma6     ma7     ma8
##       -0.5590  -0.1044  -0.0719  -0.0755  -0.0488  -0.0078  0.0027  0.0692
## s.e.   0.0254   0.0290   0.0292   0.0291   0.0291   0.0290  0.0302  0.0295
##           ma9     ma10    ma11     ma12
##       -0.0419  -0.0519  0.0191  -0.0474
## s.e.   0.0302   0.0279  0.0295   0.0268
##
## sigma^2 estimated as 269.7:  log likelihood = -6609.76,  aic = 13245.52
```

MA(12) 的 AIC 为 13245.52，远好于 AR(12) 的 13276.33。拟合精简的 MA(12):

```
fixed <- rep(NA_real_, 12)
fixed[7] <- 0
resm <- arima(Gtemp, order=c(0,1,12), fixed=fixed); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(0, 1, 12), fixed = fixed)
##
## Coefficients:
##           ma1      ma2      ma3      ma4      ma5      ma6  ma7     ma8
##       -0.5589  -0.1043  -0.0718  -0.0754  -0.0486  -0.0070    0  0.0699
## s.e.   0.0254   0.0289   0.0292   0.0291   0.0290   0.0275    0  0.0285
##           ma9     ma10    ma11     ma12
##       -0.0417  -0.0517  0.0191  -0.0471
## s.e.   0.0302   0.0279  0.0295   0.0266
##
## sigma^2 estimated as 269.7:  log likelihood = -6609.76,  aic = 13243.53
```

```
fixed[c(6,7)] <- 0
resm <- arima(Gtemp, order=c(0,1,12), fixed=fixed); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(0, 1, 12), fixed = fixed)
##
## Coefficients:
##           ma1      ma2      ma3     ma4      ma5  ma6  ma7     ma8      ma9
##       -0.5593  -0.1048  -0.0722  -0.076  -0.0516    0    0  0.0693  -0.0414
## s.e.   0.0254   0.0289   0.0294   0.029   0.0265    0    0  0.0285   0.0302
##          ma10    ma11     ma12
##       -0.0523  0.0182  -0.0473
## s.e.   0.0278  0.0293   0.0266
##
## sigma^2 estimated as 269.7:  log likelihood = -6609.8,  aic = 13241.59
```

```
fixed[c(6,7,11)] <- 0
resm <- arima(Gtemp, order=c(0,1,12), fixed=fixed); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(0, 1, 12), fixed = fixed)
##
## Coefficients:
##           ma1      ma2      ma3      ma4      ma5  ma6  ma7     ma8      ma9
```

```
##       -0.5580  -0.1049  -0.0742  -0.0761  -0.0504    0    0  0.0699  -0.0385
## s.e.   0.0253   0.0290   0.0292   0.0290   0.0264    0    0  0.0286   0.0299
##          ma10  ma11     ma12
##       -0.0459     0  -0.0398
## s.e.   0.0258     0   0.0237
##
## sigma^2 estimated as 269.8:  log likelihood = -6609.99,  aic = 13239.98
```

```
fixed[c(6,7,11,9)] <- 0
resm <- arima(Gtemp, order=c(0,1,12), fixed=fixed); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(0, 1, 12), fixed = fixed)
##
## Coefficients:
##           ma1      ma2      ma3      ma4      ma5  ma6  ma7     ma8  ma9
##       -0.5572  -0.1074  -0.0749  -0.0772  -0.0494    0    0  0.0544    0
## s.e.   0.0252   0.0289   0.0292   0.0287   0.0258    0    0  0.0258    0
##          ma10  ma11     ma12
##       -0.0615     0  -0.0440
## s.e.   0.0228     0   0.0235
##
## sigma^2 estimated as 270.1:  log likelihood = -6610.82,  aic = 13239.65
```

最后选择了一个 MA(12) 限制滞后 6、7、9、11 系数为零。考察这个模型的诊断：

```
fixed <- rep(NA_real_, 12)
fixed[c(6,7,11,9)] <- 0
mod1 <- arima(Gtemp, order=c(0,1,12), fixed=fixed)
tsdiag(mod1, gof=36)
```

残差有重尾现象，除此之外比较符合白噪声要求。在 Ljung-Box 检验使用自相关系数个数很多时会有拒绝白噪声假设的问题。

考虑用 ARMA。ARIMA(1,1,1):

```
resm <- arima(Gtemp, order=c(1,1,1)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(1, 1, 1))
##
## Coefficients:
##          ar1      ma1
##       0.3759  -0.8879
## s.e.  0.0430   0.0281
```

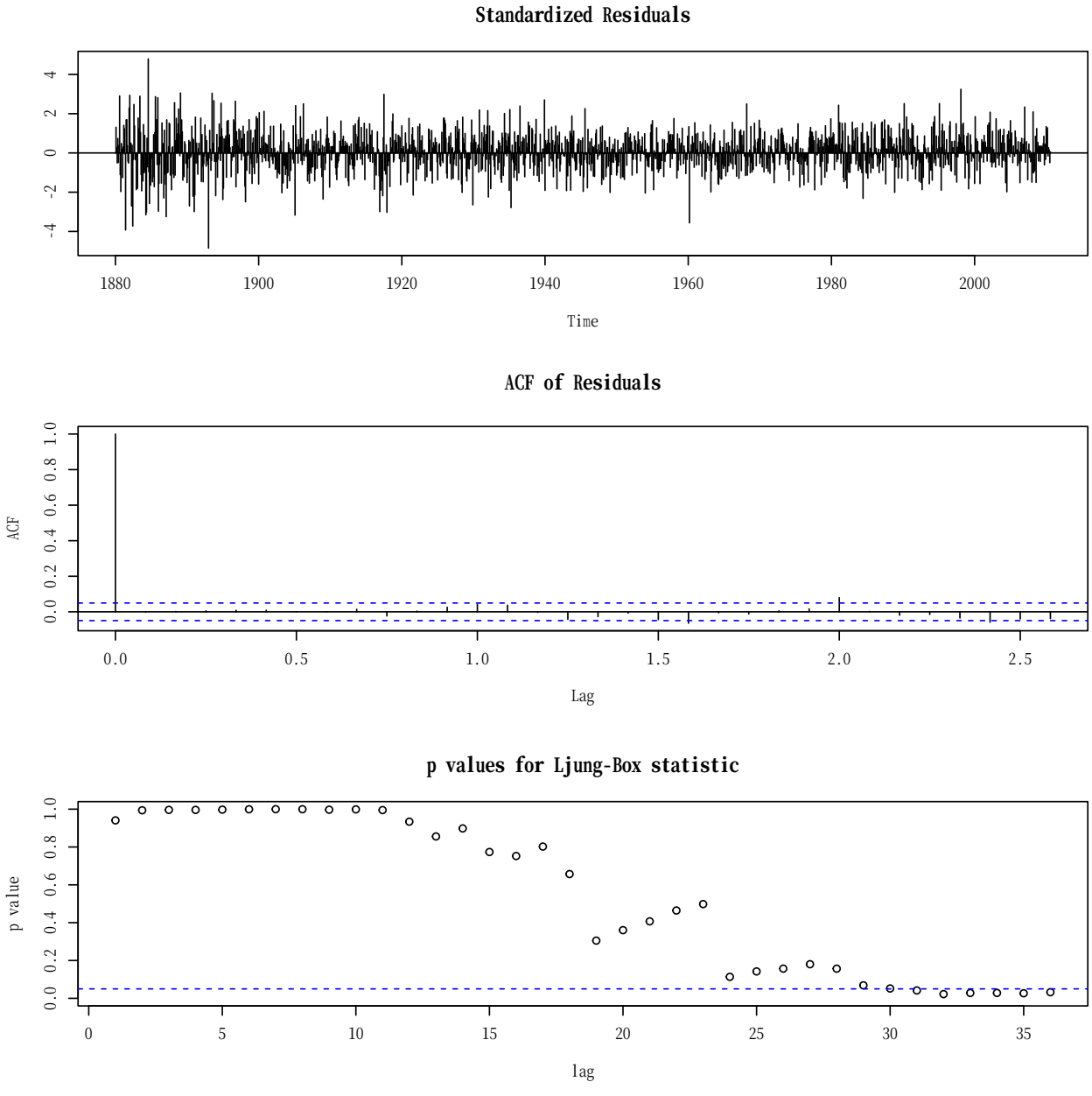
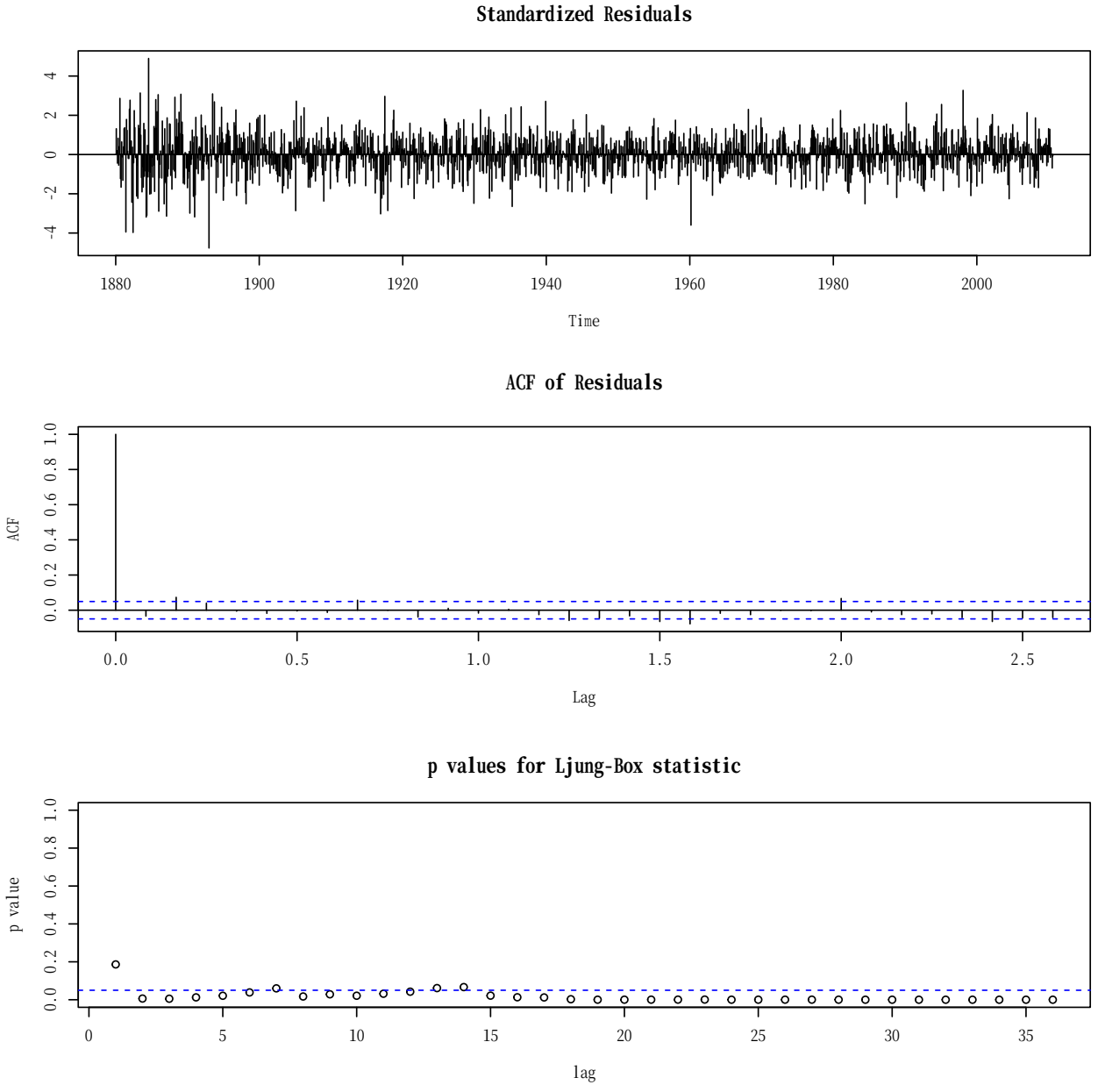


图 14.6: 稀疏系数 MA(12) 模型诊断

```
##
## sigma^2 estimated as 277:  log likelihood = -6630.33,  aic = 13266.65
```

效果不如刚才的 MA(12)。考察其模型诊断:

```
resm <- arima(Gtemp, order=c(1,1,1))
tsdiag(resm, gof=36)
```



模型诊断表明 ARIMA(1,1,1) 是不充分的。

考虑 ARIMA(1,1,2) 模型:

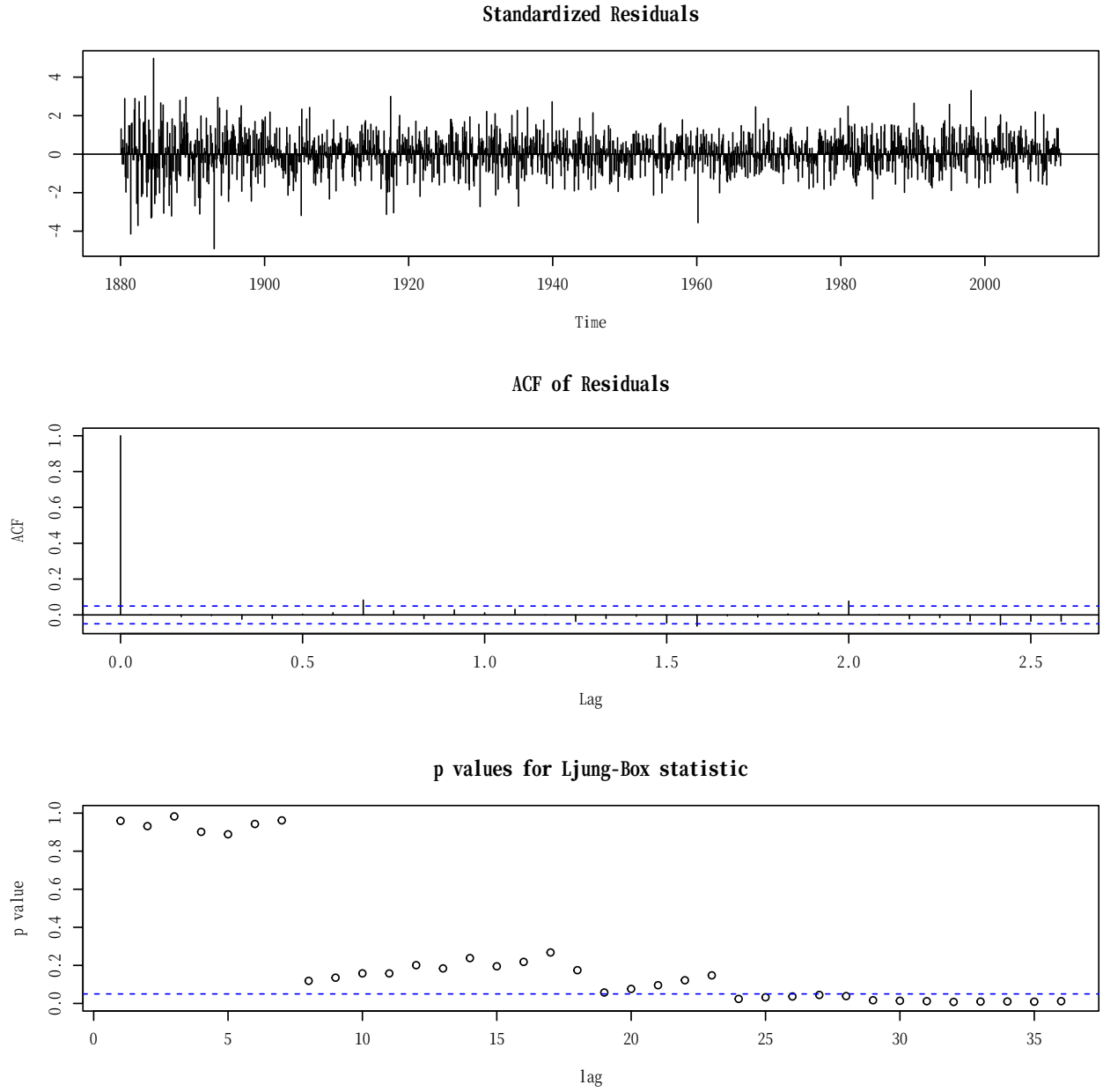
```
mod2 <- arima(Gtemp, order=c(1,1,2)); mod2
```

```
##
```

```
## Call:
## arima(x = Gtemp, order = c(1, 1, 2))
##
## Coefficients:
##          ar1      ma1     ma2
##       0.7387  -1.2973  0.3183
## s.e.  0.0406   0.0533  0.0492
##
## sigma^2 estimated as 272.1:  log likelihood = -6616.56,  aic = 13241.11
```

其 AIC 为 13241.11，比稀疏系数 MA(12) 的 13239.65 略差，但是模型更简单。作模型诊断：

```
mod2 <- arima(Gtemp, order=c(1,1,2))
tsdiag(mod2, gof=36)
```



Ljung-Box 检验的结果比 MA(12) 的结果差。

尝试加入一个季节 MA 项：

```
resm <- arima(Gtemp, order=c(1,1,2),
              seasonal=list(order=c(0,0,1), period=12))
resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(1, 1, 2), seasonal = list(order = c(0, 0, 1), period = 12))
##
## Coefficients:
```

```
##          ar1      ma1     ma2    sma1
##       0.7409  -1.3002  0.3206  0.0128
## s.e.  0.0400   0.0529  0.0490  0.0239
##
## sigma^2 estimated as 272:  log likelihood = -6616.41,  aic = 13242.81
```

加入的季节 MA 系数不显著，AIC 变差一些。继续尝试加入滞后 24 的季节 MA:

```
resm <- arima(Gtemp, order=c(1,1,2),
              seasonal=list(order=c(0,0,2), period=12))
resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(1, 1, 2), seasonal = list(order = c(0, 0, 2), period = 12))
##
## Coefficients:
##          ar1      ma1     ma2    sma1    sma2
##       0.7619  -1.3256  0.3426  0.0147  0.0717
## s.e.  0.0374   0.0514  0.0482  0.0257  0.0243
##
## sigma^2 estimated as 270.5:  log likelihood = -6612.04,  aic = 13236.07
```

```
mod2 <- arima(Gtemp, order=c(1,1,2),
              seasonal=list(order=c(0,0,2), period=12),
              fixed=c(NA, NA, NA, 0, NA))
mod2
```

```
##
## Call:
## arima(x = Gtemp, order = c(1, 1, 2), seasonal = list(order = c(0, 0, 2), period = 12),
##     fixed = c(NA, NA, NA, 0, NA))
##
## Coefficients:
##          ar1      ma1     ma2  sma1    sma2
##       0.7612  -1.3241  0.3416     0  0.0717
## s.e.  0.0379   0.0519  0.0485     0  0.0243
##
## sigma^2 estimated as 270.6:  log likelihood = -6612.2,  aic = 13234.4
```

此模型的 AIC 在比较的模型中最优。进行模型诊断：

```
tsdiag(mod2, gof=36)
```

模型诊断图表明模型基本充分，但也有一些缺陷：残差可能存在较多异常值，LB 白噪声检验结果接近于拒绝白噪声假设。在单位根非平稳模型中可以选定这个模型。模型写成：

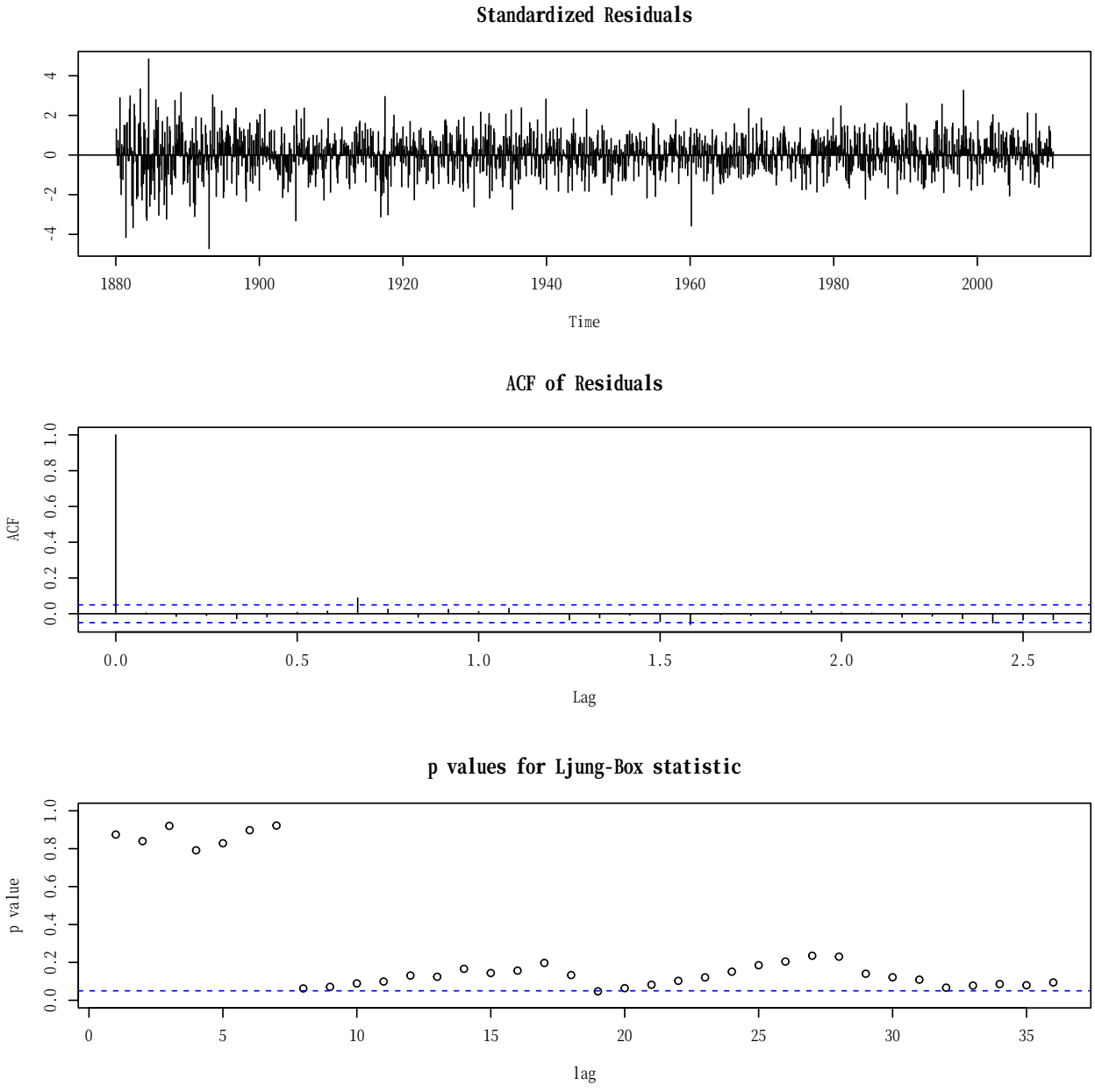


图 14.7: ARIMA$(1,1,2)(0,0,2)_{12}$ 模型诊断

$$(1-0.7612B)(1-B)G_t = (1+0.0717B^{24})(1-1.3241B+0.3416B^2)\varepsilon_t,\ \mathrm{Var}(\varepsilon_t)=270.6$$

AIC 值为 11234.4。

14.3 线性固定趋势模型

由于温度时间序列存在明显的上升趋势，一种候选模型是固定趋势模型。最简单的固定趋势是线性趋势：

$$G_t = \beta_0 + \beta_1 t + z_t$$

其中 $\{z_t\}$ 为线性时间序列。

先做简单的回归：

```
tt <- seq(length(Gtemp))
reslm <- lm(c(Gtemp) ~ tt); summary(reslm)
```

```
##
## Call:
## lm(formula = c(Gtemp) ~ tt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -92.055 -15.634   0.194  15.296  78.751
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -38.039763   1.134960  -33.52   <2e-16 ***
## tt            0.051560   0.001253   41.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.46 on 1566 degrees of freedom
## Multiple R-squared:  0.5195, Adjusted R-squared:  0.5192
## F-statistic:  1693 on 1 and 1566 DF,  p-value: < 2.2e-16
```

将估计的趋势叠加到原始序列图形上面：

```
plot(tt, Gtemp, type="l", xlab="Time", ylab="Temperature", main="Fixed Linear Trend")
abline(reslm, lwd=2, col="red")
```

这个模型拟合的效果还是不够精细。后面可以再设法改进。

考察回归的残差。残差序列图：

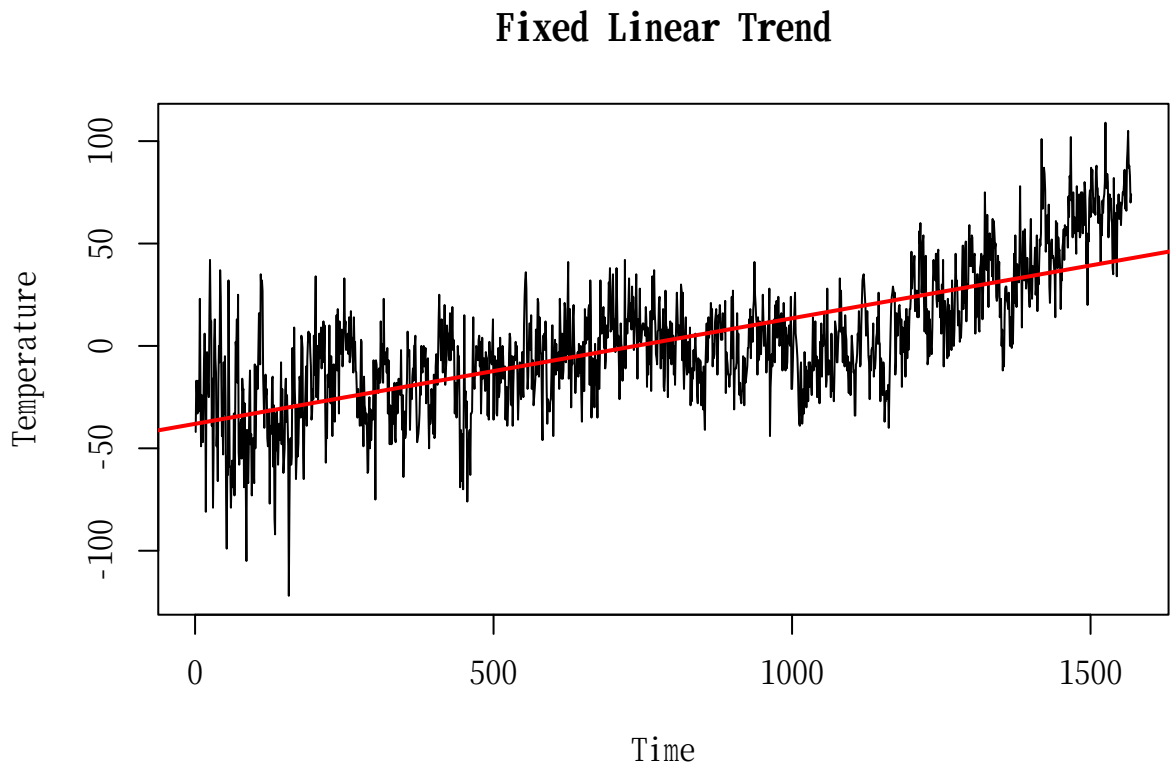


图 14.8: 非随机线性趋势

```
plot(reslm$residuals, type="l", xlab="Time", ylab="Residual")
abline(h=0, col="gray")
```

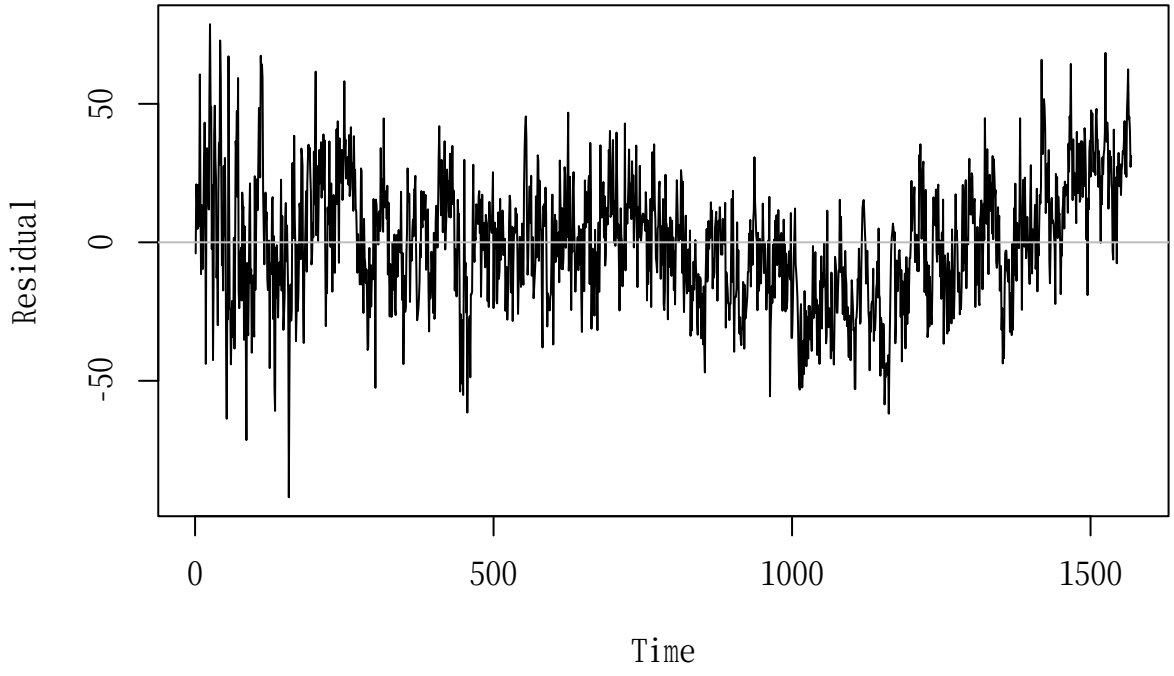


图 14.9: 非随机线性趋势残差序列

残差序列的 ACF：

```
acf(reslm$residuals, main="")
```

基本可以算是快速衰减，但不能用低阶 MA 表示。

残差序列的 PACF：

```
pacf(reslm$residuals, main="")
```

可以算是快速衰减，用低阶 AR 建模也有一定困难。

从回归残差看可以认为残差是线性平稳列，称这样的 G_t 为趋势平稳序列。

对回归残差采用 ARMA(2,1) 模型：

```
tt <- seq(length(Gtemp))
mod3 <- arima(Gtemp, xreg=tt, order=c(2,0,1)); mod3
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), xreg = tt)
```

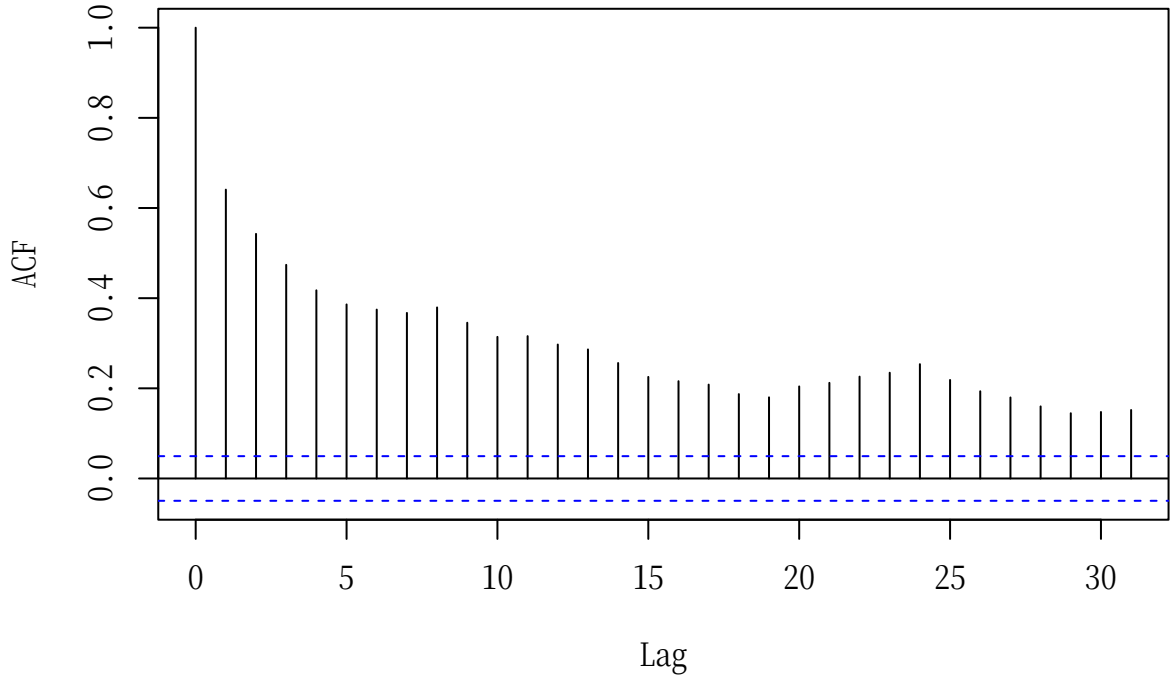


图 14.10: 非随机线性趋势残差序列 ACF

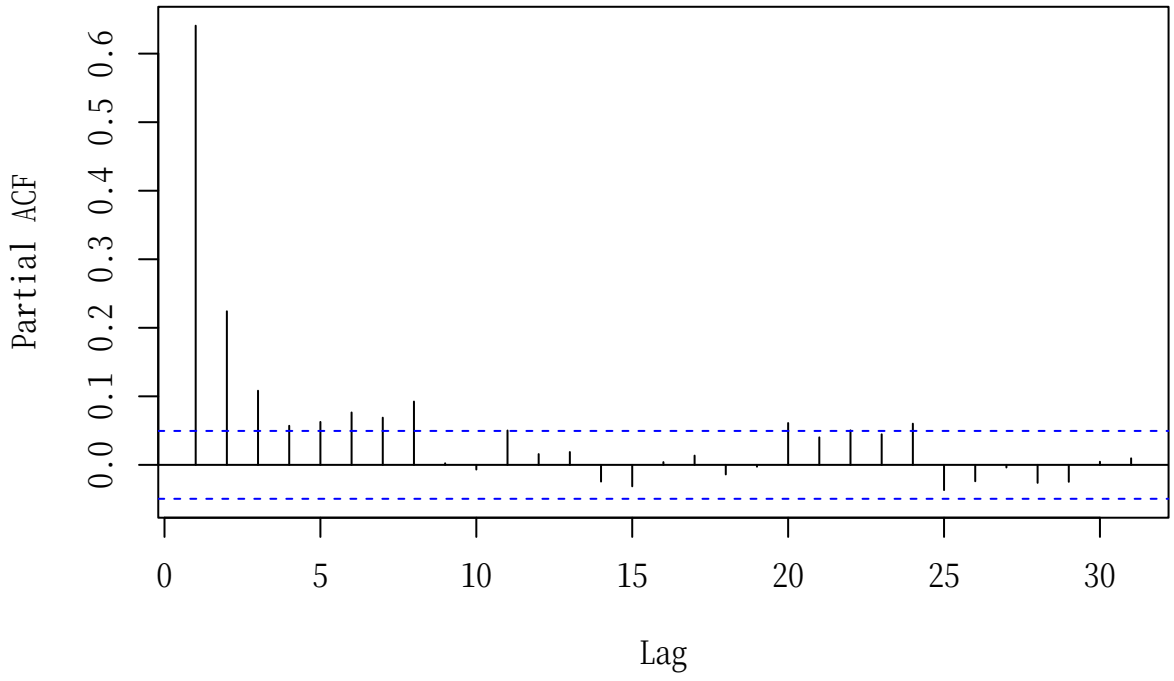


图 14.11: 非随机线性趋势残差序列 PACF

```
##
## Coefficients:
##          ar1      ar2      ma1  intercept      tt
##       1.2385  -0.2719  -0.7802   -38.8493  0.0530
## s.e.  0.0567   0.0477   0.0460     5.3548  0.0059
##
## sigma^2 estimated as 272.9:  log likelihood = -6622.99,  aic = 13257.97
```

模型可以写成

$$
\begin{aligned}
G_t =& -38.8493 + 0.0530t + z_t \\
z_t =& 1.2385 z_{t-1} - 0.2719 z_{t-2} + \varepsilon_t - 0.7802 \varepsilon_{t-1},\ \mathrm{Var}(\varepsilon_t) = 272.9
\end{aligned}
$$

对其进行模型诊断：

```
tsdiag(mod3)
```

结果基本可以接受，ACF 在滞后 24 处略超出 0.05 界限。考虑加入滞后 24 的 MA 项看是否有改进：

```
tt <- seq(length(Gtemp))
resm <- arima(Gtemp, xreg=tt, order=c(2,0,1),
              seasonal=list(order=c(0,0,2), period=12)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), seasonal = list(order = c(0, 0, 2), period = 12),
##     xreg = tt)
##
## Coefficients:
##          ar1      ar2      ma1    sma1    sma2  intercept      tt
##       1.1935  -0.2376  -0.7426  0.0043  0.0856   -38.6296  0.0528
## s.e.  0.0607   0.0494   0.0505  0.0265  0.0241     5.1662  0.0057
##
## sigma^2 estimated as 270.7:  log likelihood = -6616.71,  aic = 13249.42
```

```
mod4 <- arima(Gtemp, xreg=tt, order=c(2,0,1), fixed=c(NA, NA, NA, 0, NA, NA, NA),
              seasonal=list(order=c(0,0,2), period=12)); mod4
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), seasonal = list(order = c(0, 0, 2), period = 12),
##     xreg = tt, fixed = c(NA, NA, NA, 0, NA, NA, NA))
##
## Coefficients:
##          ar1      ar2      ma1  sma1    sma2  intercept      tt
##       1.1960  -0.2394  -0.7451     0  0.0856   -38.7150  0.0529
## s.e.  0.0587   0.0482   0.0486     0  0.0241     5.1843  0.0057
```

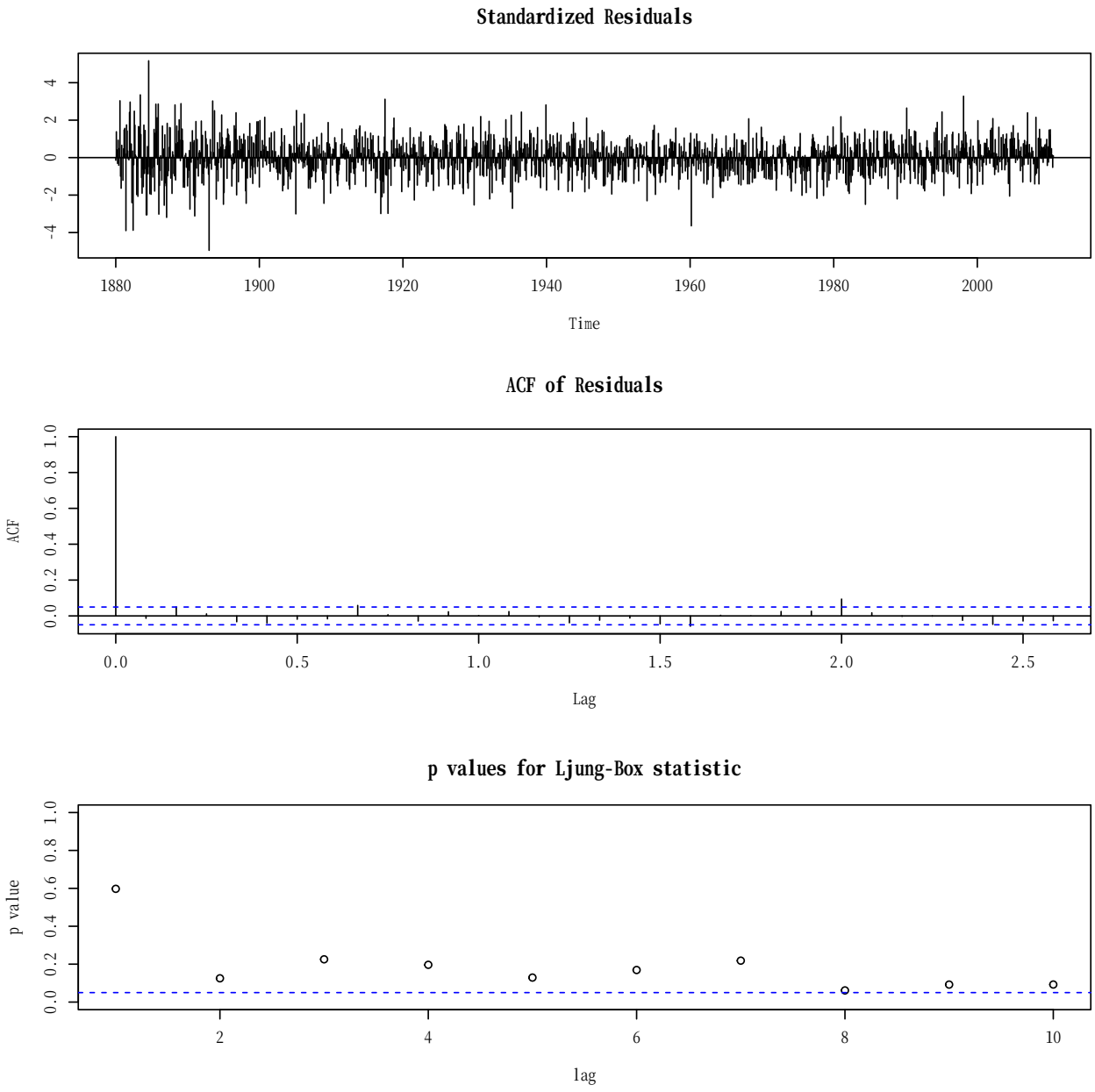


图 14.12: 线性趋势模型诊断

```
##
## sigma^2 estimated as 270.8:  log likelihood = -6616.72,  aic = 13247.45
```

从 AIC 来看是有明显改进的。但是，AIC 比较不如单位根非平稳模型的结果。模型为：

$$G_t = -38.7150 + 0.0529t + z_t$$
$$(1 - 1.1960B + 0.2394B^2)z_t = (1 - 0.7451B)(1 + 0.0856B^{24})\varepsilon_t,\ \mathrm{Var}(\varepsilon_t) = 270.8$$

对此模型的残差诊断图：

```
tsdiag(mod4, gof=36)
```

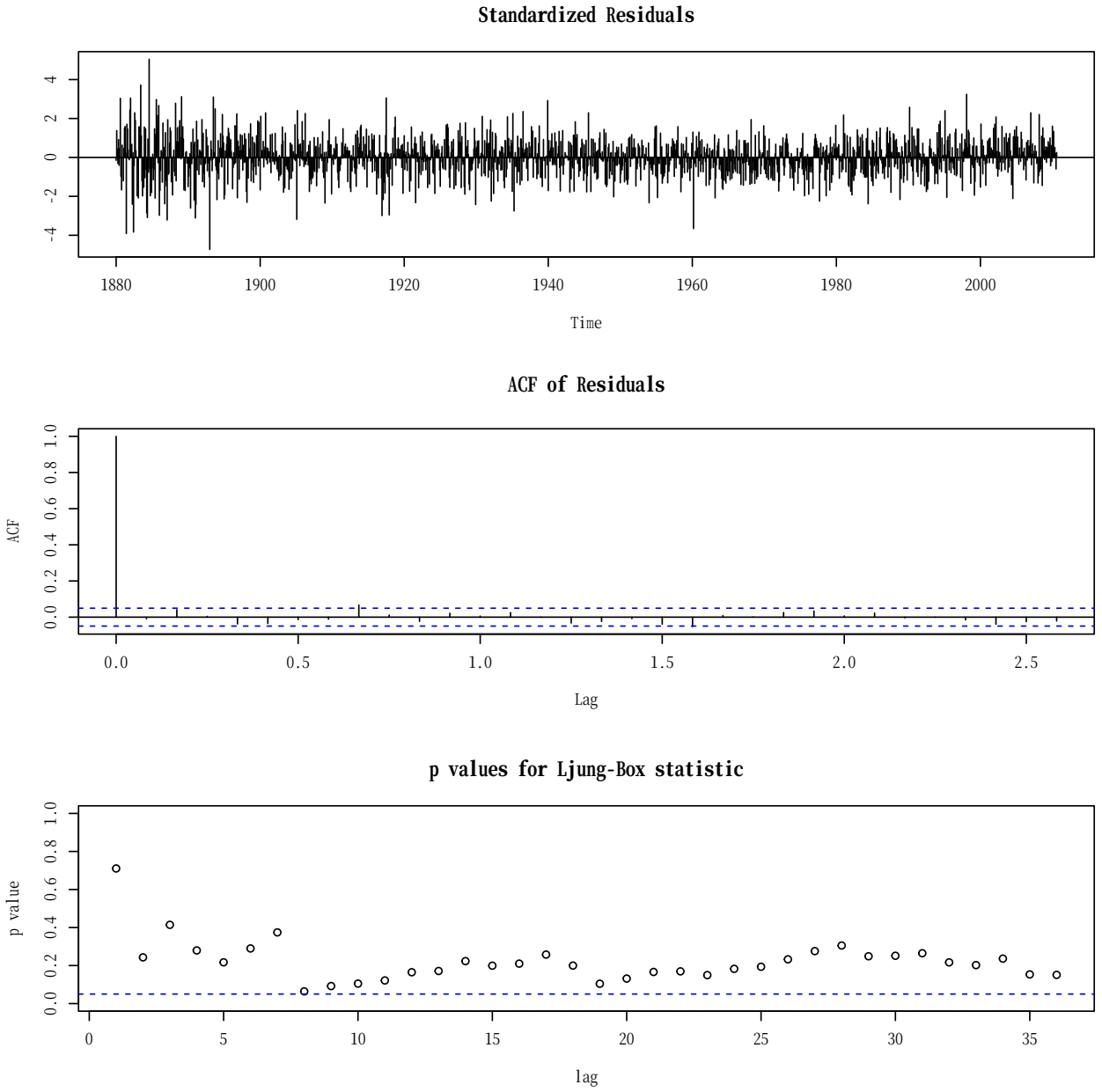


图 14.13: 趋势平稳改进模型的诊断图

结果表明模型比较充分。在趋势平稳假定下选择这个模型。

从估计结果看，平均每月气温增长 0.000529 摄氏度，平均每年气温增长 0.006348 摄氏度，平均 158 年增加 1 摄氏度。

14.4 二次固定趋势模型

温度异常值时间序列的增长趋势不像是线性的，在最后四十年增长更快。比较简单的非线性函数是二次多项式函数。模型为

$$G_t = \beta_0 + \beta_1 t + \beta_2 t^2 + z_t$$

其中 $\{z_t\}$ 为线性时间序列。

先做简单的回归：

```
tt <- seq(length(Gtemp))
reslm2 <- lm(c(Gtemp) ~ tt + I(tt^2)); summary(reslm2)
```

```
##
## Call:
## lm(formula = c(Gtemp) ~ tt + I(tt^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.53 -14.45   1.10  14.68  64.02
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.178e+01  1.613e+00 -13.505   <2e-16 ***
## tt          -1.058e-02  4.748e-03  -2.228    0.026 *
## I(tt^2)      3.960e-05  2.930e-06  13.517   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.26 on 1565 degrees of freedom
## Multiple R-squared:  0.5697, Adjusted R-squared:  0.5692
## F-statistic:  1036 on 2 and 1565 DF,  p-value: < 2.2e-16
```

将估计的趋势叠加到原始序列图形上面：

```
plot(tt, Gtemp, type="l", xlab="Time", ylab="Temperature", main="Fixed Quadratic Trend")
lines(tt, predict(reslm2), lwd=2, col="blue")
```

考察回归的残差。残差序列图：

```
plot(reslm2$residuals, type="l", xlab="Time", ylab="Residual")
abline(h=0, col="gray")
```

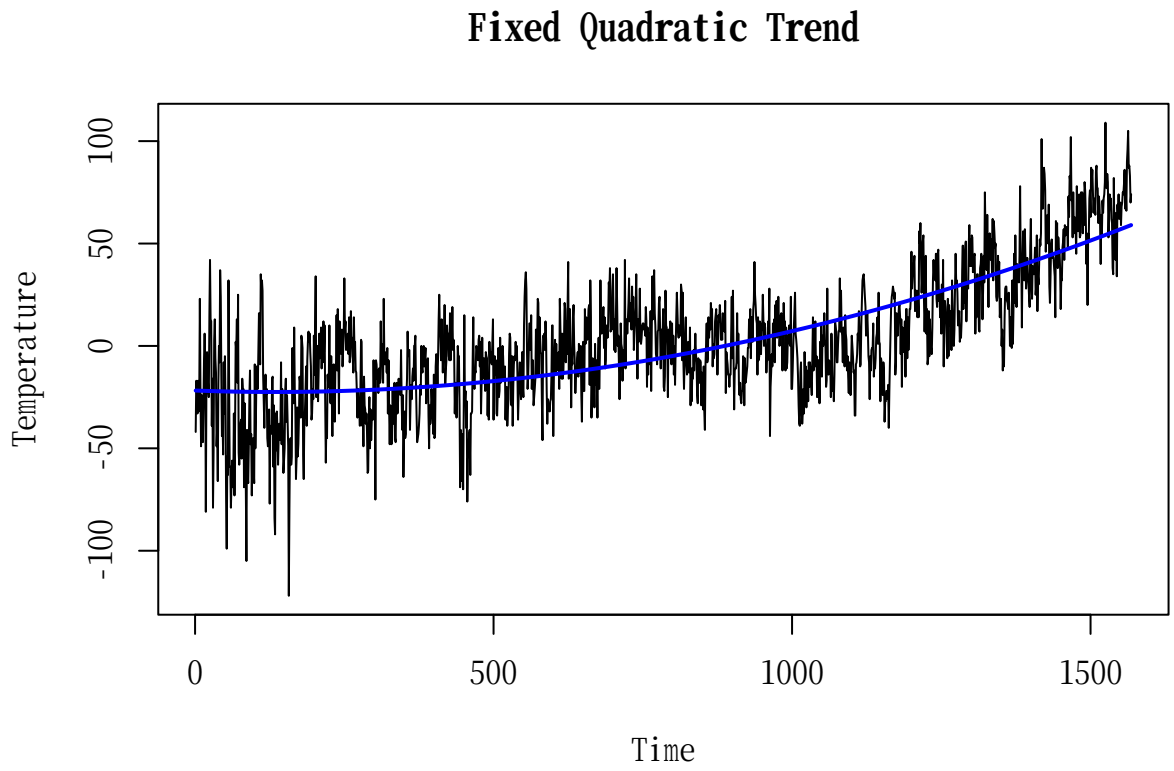


图 14.14: 非随机二次多项式趋势

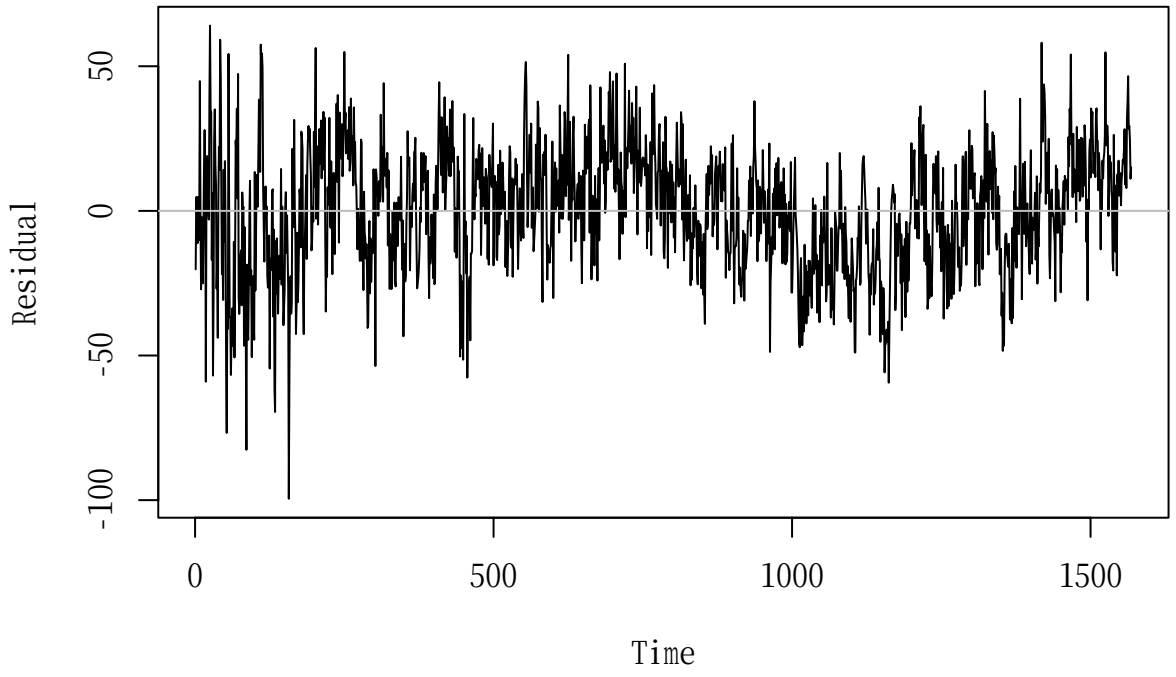


图 14.15: 非随机二次多项式趋势残差序列

残差序列仍呈现出非平稳。

残差序列的 ACF：

```
forecast::Acf(reslm2$residuals, main="")
```

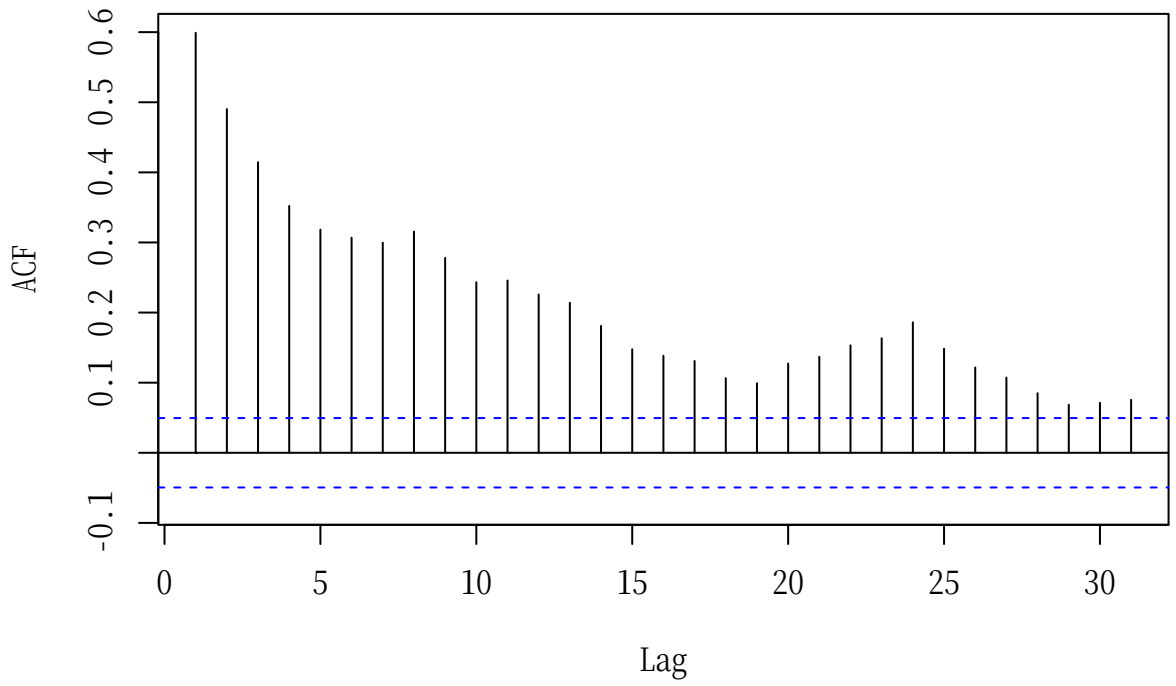


图 14.16: 非随机线性趋势残差序列 ACF

此 ACF 不能用低阶 MA 表示。PACF 为：

```
pacf(reslm2$residuals, main="")
```

PACF 表现像是低阶 AR 模型。

用 AIC 确定一个 AR 模型：

```
ar(reslm2$residuals, method="mle")
```

```
##
## Call:
## ar(x = reslm2$residuals, method = "mle")
##
## Coefficients:
##      1       2       3       4       5       6       7       8
## 0.4363  0.1413  0.0593  0.0071  0.0074  0.0286  0.0238  0.0901
##
```

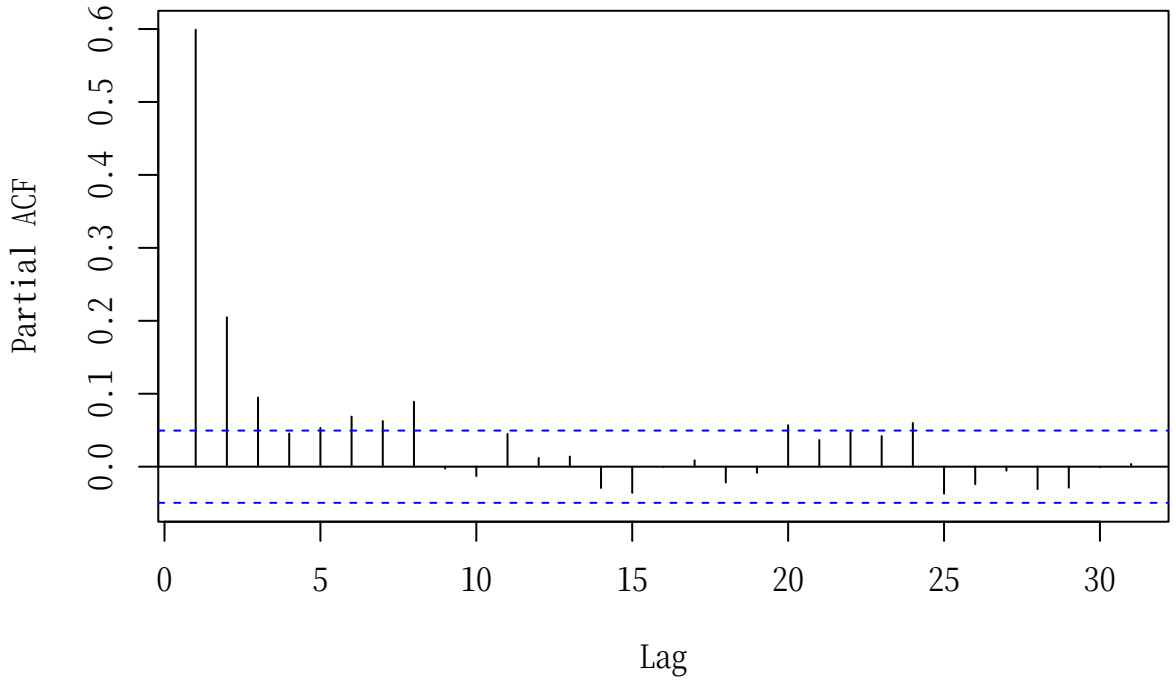


图 14.17: 非随机线性趋势残差序列 PACF

```
## Order selected 8  sigma^2 estimated as  268.5
```

用 arima() 函数联合估计：

```
tt <- seq(length(Gtemp))
tt2 <- (tt - mean(tt))^2/1E5
mod5 <- arima(Gtemp, xreg=cbind(tt, tt2), order=c(8,0,0)); mod5
```

```
##
## Call:
## arima(x = Gtemp, order = c(8, 0, 0), xreg = cbind(tt, tt2))
##
## Coefficients:
##          ar1     ar2     ar3     ar4     ar5     ar6     ar7     ar8  intercept
##       0.4361  0.1415  0.0594  0.0070  0.0075  0.0286  0.0240  0.0896   -46.8156
## s.e.  0.0251  0.0275  0.0277  0.0277  0.0277  0.0277  0.0275  0.0252     4.5405
##           tt     tt2
##       0.0522  4.0754
## s.e.  0.0044  1.0620
##
## sigma^2 estimated as 268.5:  log likelihood = -6610.02,  aic = 13244.03
```

对残差考虑 ARMA 建模。ARMA(1,1):

```
tt <- seq(length(Gtemp))
tt2 <- (tt - mean(tt))^2/1E5
resm <- arima(Gtemp, xreg=cbind(tt, tt2), order=c(1,0,1)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(1, 0, 1), xreg = cbind(tt, tt2))
##
## Coefficients:
##          ar1      ma1  intercept      tt     tt2
##       0.8476  -0.4199   -46.4231  0.0519  3.9788
## s.e.  0.0223   0.0410     3.6182  0.0035  0.8538
##
## sigma^2 estimated as 273.3:  log likelihood = -6623.93,  aic = 13259.85
```

效果不如 AR(8)。改为 ARMA(1,2)：

```
tt <- seq(length(Gtemp))
tt2 <- (tt - mean(tt))^2/1E5
resm <- arima(Gtemp, xreg=cbind(tt, tt2), order=c(1,0,2)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(1, 0, 2), xreg = cbind(tt, tt2))
```

```
##
## Coefficients:
##          ar1      ma1      ma2  intercept      tt     tt2
##       0.9003  -0.4637  -0.0954   -46.6156  0.0521  4.0167
## s.e.  0.0227   0.0363   0.0332     4.1794  0.0040  0.9814
##
## sigma^2 estimated as 272:  log likelihood = -6620.18,  aic = 13254.37
```

改为 ARMA(2,1)：

```
tt <- seq(length(Gtemp))
tt2 <- (tt - mean(tt))^2/1E5
resm <- arima(Gtemp, xreg=cbind(tt, tt2), order=c(2,0,1)); resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), xreg = cbind(tt, tt2))
##
## Coefficients:
##          ar1      ar2      ma1  intercept      tt     tt2
##       1.1946  -0.2460  -0.7454   -46.8246  0.0522  4.0737
## s.e.  0.0651   0.0513   0.0554     4.6574  0.0045  1.0875
##
## sigma^2 estimated as 271.1:  log likelihood = -6617.5,  aic = 13249
```

增加一个滞后 24 的季节 MA 项：

```
tt <- seq(length(Gtemp))
tt2 <- (tt - mean(tt))^2/1E5
resm <- arima(Gtemp, xreg=cbind(tt, tt2), order=c(2,0,1),
              seasonal=list(order=c(0,0,2), period=12))
resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), seasonal = list(order = c(0, 0, 2), period = 12),
##     xreg = cbind(tt, tt2))
##
## Coefficients:
##          ar1      ar2      ma1    sma1    sma2  intercept      tt     tt2
##       1.1527  -0.2156  -0.7104  0.0078  0.0848   -46.8806  0.0523  4.1021
## s.e.  0.0691   0.0529   0.0599  0.0265  0.0239     4.7049  0.0045  1.0983
##
## sigma^2 estimated as 268.9:  log likelihood = -6611.15,  aic = 13240.31
```

删去 SMA1 系数：

```
tt <- seq(length(Gtemp))
tt2 <- (tt - mean(tt))^2/1E5
fixed <- rep(NA_real_, 8)
fixed[4] <- 0
mod6 <- arima(Gtemp, xreg=cbind(tt, tt2), order=c(2,0,1),
              seasonal=list(order=c(0,0,2), period=12),
              fixed=fixed)
mod6
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), seasonal = list(order = c(0, 0, 2), period = 12),
##     xreg = cbind(tt, tt2), fixed = fixed)
##
## Coefficients:
##          ar1      ar2      ma1  sma1    sma2  intercept      tt     tt2
##       1.1568  -0.2183  -0.7141     0  0.0849   -46.8836  0.0522  4.1094
## s.e.  0.0666   0.0515   0.0574     0  0.0239     4.7189  0.0045  1.1014
##
## sigma^2 estimated as 268.9:  log likelihood = -6611.2,  aic = 13238.4
```

方程可以写成：

$$G_t = -46.88 + 0.0522t + 4.109\times10^{-5}(t-784.5)^2 + z_t \\ z_t = 1.16z_{t-1} - 0.22z_{t-2} + (1-0.7141B)(1+0.0849B^{24})\varepsilon_t \\ \mathrm{Var}(\varepsilon_t) = 268.9$$

模型的残差诊断：

```
forecast::Acf(mod6$residuals, lag.max=36, main="")
```

14.5 模型比较

前面考虑了多个模型，以下的三个模型是选择出来较好的，对模型残差的检验也能证明模型充分。如何比较效果相近的模型？

使用差分平稳模型（随机趋势模型）时，选用 $\mathrm{ARIMA}(1,1,2)(0,0,2)_{12}$ 且季节 MA 滞后 1 系数为零。

```
mc1 <- arima(
  Gtemp, order=c(1,1,2),
  seasonal=list(order=c(0,0,2), period=12),
  fixed=c(NA, NA, NA, 0, NA))
mc1
```

```
##
## Call:
## arima(x = Gtemp, order = c(1, 1, 2), seasonal = list(order = c(0, 0, 2), period = 12),
```

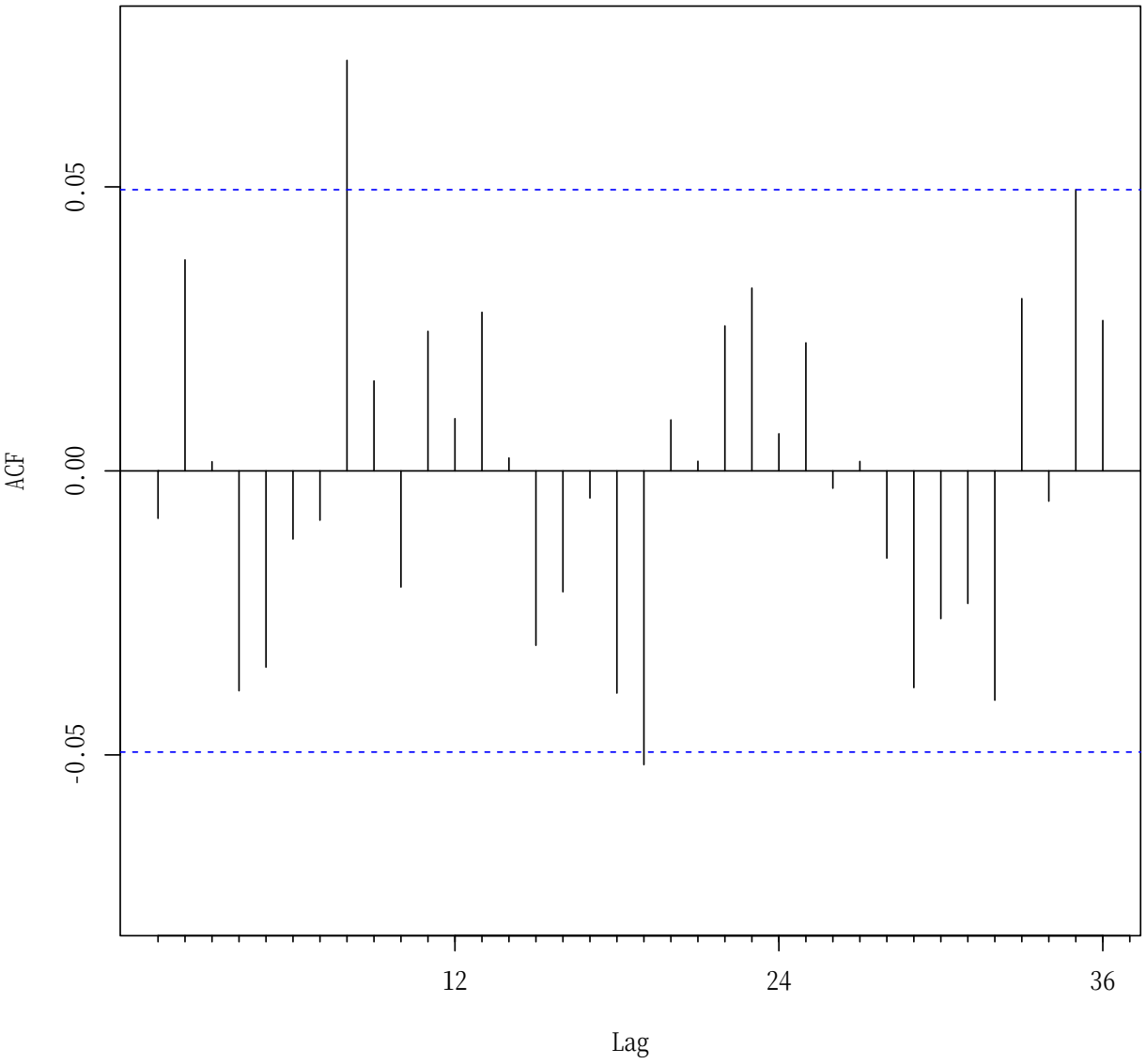


图 14.18: 二次趋势模型建模的残差诊断

```
##     fixed = c(NA, NA, NA, 0, NA))
##
## Coefficients:
##          ar1      ma1     ma2  sma1    sma2
##       0.7612  -1.3241  0.3416     0  0.0717
## s.e.  0.0379   0.0519  0.0485     0  0.0243
##
## sigma^2 estimated as 270.6:  log likelihood = -6612.2,  aic = 13234.4
```

使用非随机线性趋势模型时，残差选用 $\text{ARIMA}(2,0,1)(0,0,2)_{12}$ 且季节 MA 滞后 1 系数为零。

```
tt <- seq(length(Gtemp))
mc2 <- arima(
  Gtemp,
  xreg=tt,
  order=c(2,0,1),
  seasonal=list(order=c(0,0,2), period=12),
  fixed=c(NA, NA, NA, 0, NA, NA, NA))
mc2
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), seasonal = list(order = c(0, 0, 2), period = 12),
##     xreg = tt, fixed = c(NA, NA, NA, 0, NA, NA, NA))
##
## Coefficients:
##          ar1      ar2      ma1  sma1    sma2  intercept      tt
##       1.1960  -0.2394  -0.7451     0  0.0856   -38.7150  0.0529
## s.e.  0.0587   0.0482   0.0486     0  0.0241     5.1843  0.0057
##
## sigma^2 estimated as 270.8:  log likelihood = -6616.72,  aic = 13247.45
```

使用非随机二次多项式趋势模型时，残差选用 $\text{ARIMA}(2,0,1)(0,0,2)_{12}$ 且季节 MA 滞后 1 系数为零。

```
tt <- seq(length(Gtemp))
tt2 <- (tt - mean(tt))^2/1E5
fixed <- rep(NA_real_, 8)
fixed[4] <- 0
mc3 <- arima(
  Gtemp, xreg=cbind(tt, tt2), order=c(2,0,1),
  seasonal=list(order=c(0,0,2), period=12),
  fixed=fixed)
mc3
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), seasonal = list(order = c(0, 0, 2), period = 12),
```

```
##     xreg = cbind(tt, tt2), fixed = fixed)
##
## Coefficients:
##          ar1      ar2      ma1  sma1    sma2  intercept      tt     tt2
##       1.1568  -0.2183  -0.7141     0  0.0849   -46.8836  0.0522  4.1094
## s.e.  0.0666   0.0515   0.0574     0  0.0239     4.7189  0.0045  1.1014
##
## sigma^2 estimated as 268.9:  log likelihood = -6611.2,  aic = 13238.4
```

14.5.1 样本内比较

可以用对建模用数据的拟合情况比较模型优劣，在所用的参数个数相近的情况下比较是有意义的。

比较可以用：

- 模型残差更符合白噪声特征；
- AIC 或 BIC 更小；
- $\sigma^2_{\varepsilon_t}$ 更小。

从 AIC 来看，应该是 MC1 的随机趋势模型最好，其 AIC=13234 最小。

但是，不同的模型有完全不同的含义。对于随机趋势模型，它可以拟合出增长趋势，但是模型并不意味着以后也会持续增长，差分平稳模型完全可以表现出下降趋势。

对于非随机线性增长趋势模型，因为有一个显著的线性增长项，模型意味着以后温度也持续线性增长，增长速度为每年 0.00635 度，平均 157 年增长 1 摄氏度。

对于非随机二次增长趋势模型，因为有一个显著的二次增长项，模型意味着以后温度会加速增长，从数据末尾的 2010 年 8 月的异常值 74 开始，增加 1 摄氏度只需 87 年。

从现有的数据无法确定随机趋势模型还是非随机趋势模型更正确，相对于全球温度变化这种缓慢变化，现有的 130 数据还是太少了，不足以反映整个历史中或较近的历史中的全球气温变化。就好比我们在三年的时间观测到了股市为牛市，就断言后面也是牛市一样不可信。

14.5.2 样本外比较

样本外比较的思想是，建模时留出最后一部分时间不用于建模，建模后对留出的时间做超前一步预报，比较平均预报误差。

共有 1568 个月的数据，取最后的 200 个月的数据作为验证。

使用 MC1 作预测的函数：

```
pred1 <- function(y=Gtemp, n.pred=200, DEBUG=FALSE){
  T <- length(y)
  ypred <- rep(NA_real_, T)
  step <- 0
  for(h in (T-n.pred):(T-1)){
    step <- step + 1
    if(DEBUG){
```

```
      cat("pred1: step", step, "/", n.pred, "\n")
    }
    mod <- arima(y[1:h], order=c(1,1,2),
              seasonal=list(order=c(0,0,2), period=12),
              fixed=c(NA, NA, NA, 0, NA))
    ypred[h+1] <- predict(mod, n.ahead=1, se.fit=FALSE)
  }

  ypred
}
```

使用 MC2 作预测的函数：

```
pred2 <- function(y=Gtemp, n.pred=200, DEBUG=FALSE){
  T <- length(y)
  ypred <- rep(NA_real_, T)
  step <- 0
  for(h in (T-n.pred):(T-1)){
    step <- step + 1
    if(DEBUG){
      cat("pred2: step", step, "/", n.pred, "\n")
    }
    tt <- seq(h)
    mod <- arima(y[1:h],
                 xreg=tt,
                 order=c(2,0,1),
                 seasonal=list(order=c(0,0,2), period=12),
                 fixed=c(NA, NA, NA, 0, NA, NA, NA))
    ypred[h+1] <- predict(mod, n.ahead=1, se.fit=FALSE,
                          newxreg=c(h+1))
  }

  ypred
}
```

使用 MC3 作预测的函数：

```
pred3 <- function(y=Gtemp, n.pred=200, DEBUG=FALSE){
  T <- length(y)
  ypred <- rep(NA_real_, T)
  fixed <- rep(NA_real_, 8)
  fixed[4] <- 0
  step <- 0
  for(h in (T-n.pred):(T-1)){
    step <- step + 1
```

```
    if(DEBUG){
      cat("pred3: step", step, "/", n.pred, "\n")
    }
    tt <- seq(h)
    tt2 <- (tt - 784.5)^2 * 1E-5

    mod <- arima(y[1:h],
                 xreg=cbind(tt, tt2),
                 order=c(2,0,1),
                 seasonal=list(order=c(0,0,2), period=12),
                 fixed=fixed)
    ypred[h+1] <- predict(
      mod, n.ahead=1, se.fit=FALSE,
      newxreg=cbind(tt=h+1, tt2=(h+1-784.5)^2 * 1E-5))
  }

  ypred
}
```

对一种方法的预测计算预测根均方误差 (RMFSE) 和平均绝对预测误差 (MAFE) 的函数：

```
pred.err <- function(y=Gtemp, n.pred=200, ypred){
  T <- length(y)
  i1 <- T - n.pred + 1
  RMFSE <- sqrt( mean((y[i1:T] - ypred[i1:T])^2) )
  MAFE <- mean(abs(y[i1:T] - ypred[i1:T]))

  c(RMFSE=RMFSE, MAFE=MAFE)
}
```

计算这三个候选模型的预测误差指标：

```
time_start <- proc.time()[3]
pred.tab <- data.frame(
  " 模型"=paste("MC", 1:3, sep=""),
  RMFSE=rep(0.0, 3),
  MAFE=rep(0.0, 3)
)
funcs <- list(pred1, pred2, pred3)
for(i in 1:3){
  predf <- funcs[[i]]
  ypred <- predf(y=Gtemp, n.pred=200, DEBUG=TRUE)
  err <- pred.err(y=Gtemp, n.pred=200, ypred=ypred)
  pred.tab[i, "RMFSE"] <- err["RMFSE"]
  pred.tab[i, "MAFE"] <- err["MAFE"]
```

```
}
time_used <- proc.time()[3] - time_start
cat(" 回测检验花费时间: ", round(time_used/60), " 分钟\n")
## 回测检验花费时间:  10 分钟
```

预测误差比较的结果表格：

```
knitr::kable(pred.tab, digits=2)
```

模型	RMFSE	MAFE
MC1	14.53	11.17
MC2	15.34	11.97
MC3	14.87	11.51

样本外预测的结果也是随机趋势模型 MC1 最好。

14.6 长期预测

用 MC1 作超前 1 到 1200 步（共 100 年）的长期预测，并作图：

```
pred.long1 <- function(y=Gtemp, n.pred=1200){
  T <- length(y)
  ypred <- ts(c(y, rep(NA_real_, n.pred)),
              start=start(y), frequency = frequency(y))
  ntotal <- T + n.pred
  lb <- ts(rep(NA_real_, ntotal),
           start=start(y), frequency = frequency(y))
  ub <- lb
  mod <- arima(
    y, order=c(1,1,2),
    seasonal=list(order=c(0,0,2), period=12),
    fixed=c(NA, NA, NA, 0, NA))
  ypred0 <- predict(mod, n.ahead=n.pred, se.fit=TRUE)
  ypred[(T+1):(T+n.pred)] <- ypred0$pred
  lb[(T+1):(T+n.pred)] <- ypred0$pred - 2*ypred0$se
  ub[(T+1):(T+n.pred)] <- ypred0$pred + 2*ypred0$se

  list(pred=ypred, lb=lb, ub=ub)
}
```

```
plot.pred <- function(y=Gtemp, pred.long=pred.long1, n.pred=1200){
  ypred <- pred.long(y=Gtemp, n.pred=1200)
  ylim <- range(c(ypred$pred, ypred$lb, ypred$ub), na.rm=TRUE)
  times <- c(time(ypred$pred))
  plot(times, ypred$pred, type="l", xlab="year", ylab="Forcast",
       ylim=ylim)
```

```
  t1 <- length(y)+1
  t2 <- length(ypred$pred)
  lines(times[t1:t2], ypred$pred[t1:t2], col="red")
  lines(times[t1:t2], ypred$lb[t1:t2], lty=2, col="blue")
  lines(times[t1:t2], ypred$ub[t1:t2], lty=2, col="blue")
}
```

```
plot.pred(y=Gtemp, pred.long=pred.long1, n.pred=1200)
```

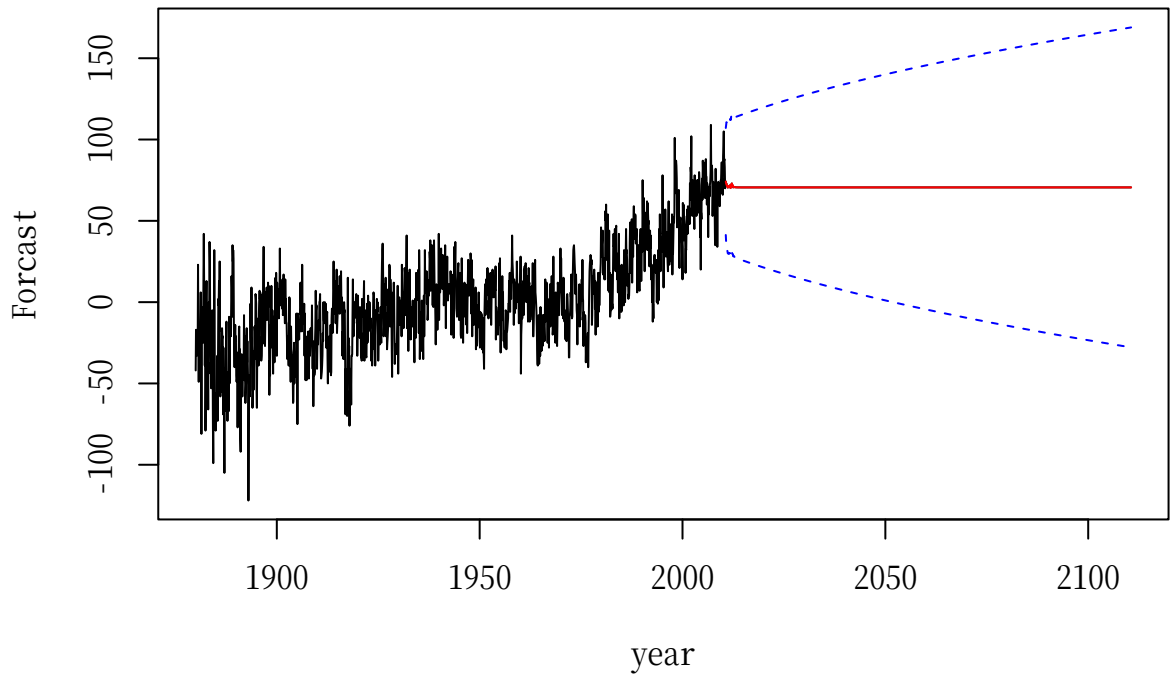


图 14.19: 随机趋势模型的 100 年长期预测

可见随机趋势的预测结果基本是用最后一个有数据的月份的数据作为预测值，预测的区间越来越宽。

用 MC3 作超前 1 到 1200 步（共 100 年）的长期预测，并作图：

```
pred.long3 <- function(y=Gtemp, n.pred=1200){
  T <- length(y)
  ypred <- ts(c(y, rep(NA_real_, n.pred)),
              start=start(y), frequency = frequency(y))
  ntotal <- T + n.pred
  lb <- ts(rep(NA_real_, ntotal),
           start=start(y), frequency = frequency(y))
  ub <- lb
```

```
  tt <- seq(ntotal)
  tt2 <- (tt - 784.5)^2/1E5
  fixed <- rep(NA_real_, 8)
  fixed[4] <- 0
  mod <- arima(
    y, xreg=cbind(tt, tt2)[1:T,], order=c(2,0,1),
    seasonal=list(order=c(0,0,2), period=12),
    fixed=fixed)
  ypred0 <- predict(mod, n.ahead=n.pred, se.fit=TRUE,
                    newxreg=cbind(tt, tt2)[(T+1):(T+n.pred),])
  ypred[(T+1):(T+n.pred)] <- ypred0$pred
  lb[(T+1):(T+n.pred)] <- ypred0$pred - 2*ypred0$se
  ub[(T+1):(T+n.pred)] <- ypred0$pred + 2*ypred0$se

  list(pred=ypred, lb=lb, ub=ub)
}
```

```
plot.pred(y=Gtemp, pred.long=pred.long3, n.pred=1200)
```

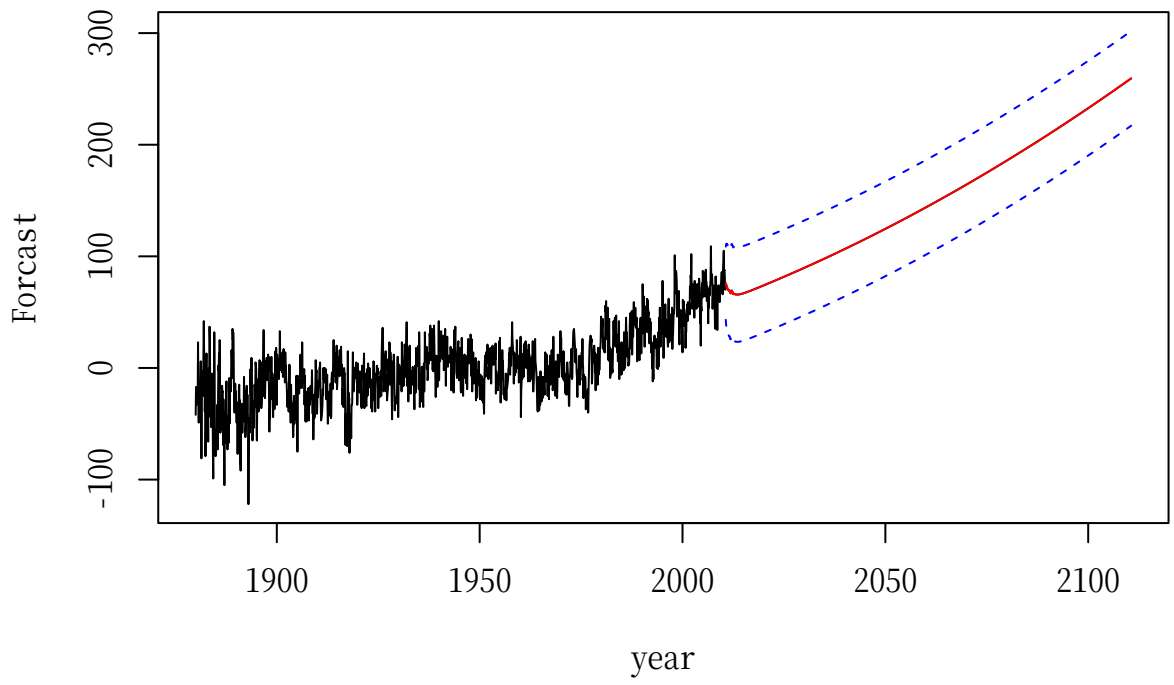


图 14.20: 非随机二次趋势模型的 100 年长期预测

非随机二次多项式趋势的模型给出了一个二次多项式增长长期预测，而且预测的区间长度略增长后就稳定不变。预测

方差最后趋于固定趋势回归的残差方差。

两个长期预测模型给出截然不同的结果，也无法证明哪一个结果正确，则两个结果都不能作为长期预测结果。

14.7 讨论

这个例子展示了如何从序列图、差分、ACF、PACF 等发现可能的建模方向，如何逐步改善模型，随机趋势与非随机趋势的比较与各自的建模步骤，如何比较模型，长期预报，等等。

下面对一些其他重要问题进行讨论。

14.7.1 模型的不确定性

所有的模型都是错误的，只不过其中一部分是有用的。模型是所分析过程的近似。

模型的不确定性一方面来自样本的随机性以及由此而来的参数估计的随机误差，还来自于不同模型选择的不确定性。统计推断时应尽可能考虑这些不确定性。时间序列分析中模型的不确定性可以通过模型平均来对抗。比如，全球温度异常模型的短期预测可以用三个模型的预测的简单平均计算，但也仅适用于短期预测。

14.7.2 短期预测与长期预测

目标是短期预测还是长期预测可能会影响模型选择，甚至于超前一步预测最优的模型不一定是超前两步预测最优的。文献中有自适应预测研究，见 (Tiao & Tsay, 1994)。

14.7.3 分段非随机趋势模型

从数据的时间序列图观察，在大约 1980 年以后斜率明显增加。所以，设计一个斜率分为两段的模型：

$$y_t = \beta_0 + (\beta_1 + \beta_2 I_t)t + \eta_t,$$

其中 I_t 为 $t > 1212$ 的示性函数，1212 对应于 1980 年 12 月，上述模型的回归斜率项在 1980 年 12 月之前为 β_1，在 1980 年 12 月之后为 $\beta_1 + \beta_2$。令 $x_t = I_t t$，则模型可以写成

$$y_t = \beta_0 + \beta_1 t + \beta_2 x_t + \eta_t.$$

```
tt <- seq(length(Gtemp))
xt <- ifelse(tt>1212, tt, 0)
resm <- arima(Gtemp, xreg=cbind(tt, xt), order=c(2,0,1))
resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), xreg = cbind(tt, xt))
##
## Coefficients:
##          ar1      ar2      ma1  intercept      tt      xt
##       1.1563  -0.2221  -0.7119   -29.0621  0.0313  0.0221
## s.e.  0.0716   0.0541   0.0627     4.0058  0.0056  0.0042
```

```
##
## sigma^2 estimated as 269.5:  log likelihood = -6612.92,  aic = 13239.83
```

增加季节项：

```
tt <- seq(length(Gtemp))
xt <- ifelse(tt>1212, tt, 0)
resm <- arima(Gtemp, xreg=cbind(tt, xt), order=c(2,0,1),
              seasonal=list(order=c(0,0,2), period=12))
resm
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), seasonal = list(order = c(0, 0, 2), period = 12),
##     xreg = cbind(tt, xt))
##
## Coefficients:
##          ar1      ar2      ma1    sma1    sma2  intercept      tt      xt
##       1.1168  -0.1940  -0.6788  0.0119  0.0820   -29.0241  0.0314  0.0220
## s.e.  0.0754   0.0556   0.0670  0.0263  0.0239     4.1381  0.0058  0.0044
##
## sigma^2 estimated as 267.4:  log likelihood = -6606.89,  aic = 13231.79
```

删去不显著的 Sma1 项：

```
tt <- seq(length(Gtemp))
xt <- ifelse(tt>1212, tt, 0)
fixed <- rep(NA_real_, 8)
fixed[4] <- 0
mod7 <- arima(Gtemp, xreg=cbind(tt, xt), order=c(2,0,1),
              seasonal=list(order=c(0,0,2), period=12),
              fixed=fixed)
mod7
```

```
##
## Call:
## arima(x = Gtemp, order = c(2, 0, 1), seasonal = list(order = c(0, 0, 2), period = 12),
##     xreg = cbind(tt, xt), fixed = fixed)
##
## Coefficients:
##          ar1      ar2      ma1  sma1    sma2  intercept      tt      xt
##       1.1220  -0.1973  -0.6835     0  0.0823   -29.2630  0.0317  0.0219
## s.e.  0.0727   0.0542   0.0643     0  0.0239     4.1411  0.0058  0.0044
##
## sigma^2 estimated as 267.5:  log likelihood = -6607,  aic = 13230
```

这个模型的 AIC 是现有的模型中最好的。从结果中 xt 自变量的估计值和标准误差比值看，这一项是显著的，即认为

$\beta_2 \neq 0$，说明分段是成立的。

残差诊断：

```
tsdiag(mod7, gof=36)
```

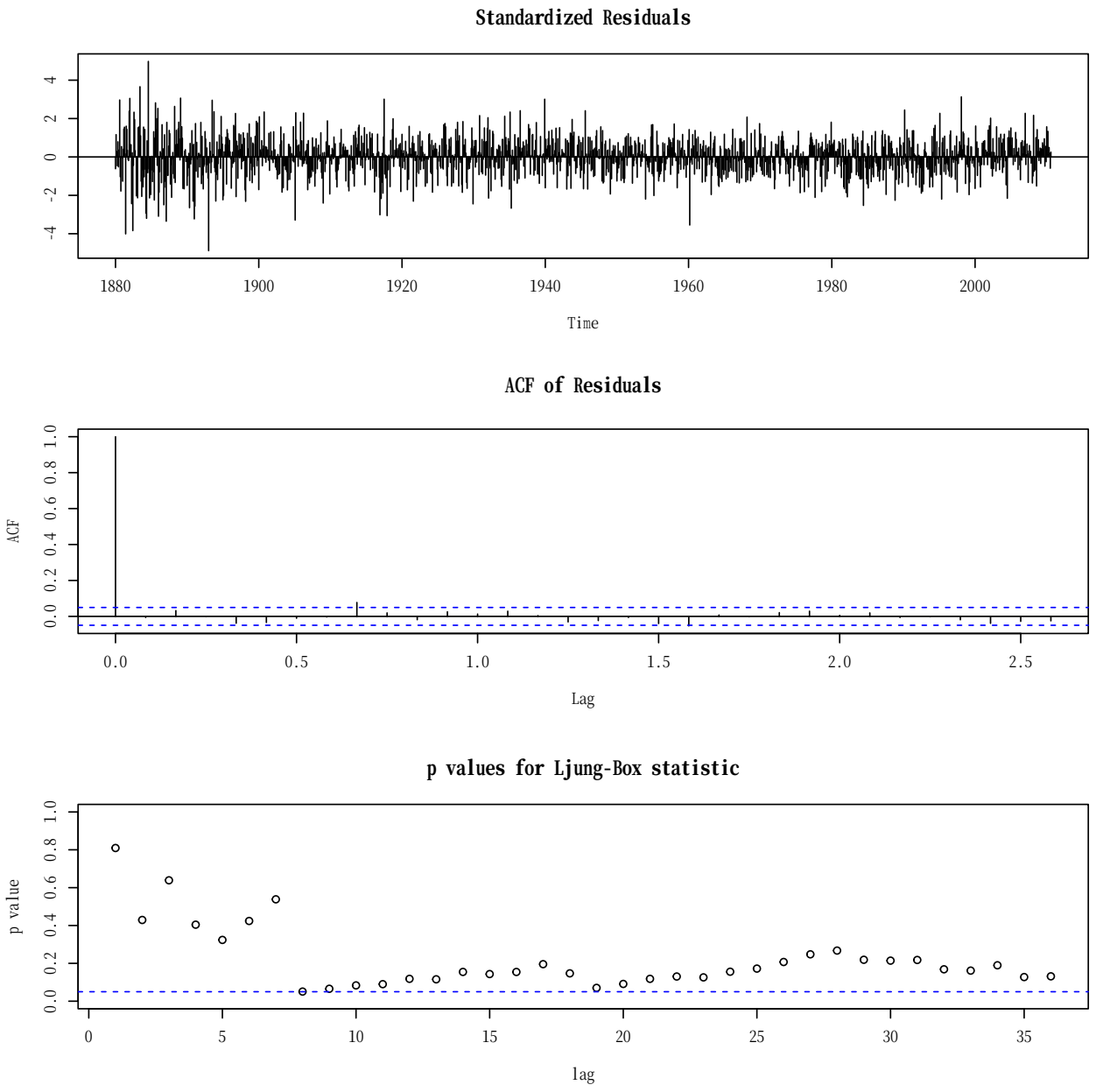


图 14.21: 分段模型的残差诊断

基本可以接受残差诊断结果。Ljung-Box 白噪声诊断：

```
Box.test(mod7$residuals, lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
```

```
##
## data:  mod7$residuals
## X-squared = 17.92, df = 12, p-value = 0.1181
```

这个模型从 1981 年开始温度的月升高率为 $(0.0317 + 0.0219) = 0.0536$，年升高率为 0.6432，即每年增加 0.006432 度，平均 155 年增加 1 摄氏度。

虽然这个模型的样本内比较 AIC 结果最好，但是并不推荐这种做法，因为过于追求样本内的拟合优度往往导致过度拟合，在样本外预测反而变差。既然斜率可以变化，那么未来斜率也可以变化，作长期预测时因为未来的斜率变化不可知所以也不能用来做长期预测。这里因为增长率加大以后样本量少所以不再比较样本外预测。

14.7.4 其它的数据

前面所用的全球气温异常数据来自 GISS。其它数据来源的数据建模也应得到类似结果。这里使用 NCDC 和 NOAA 的数据，单位为摄氏度。

读入数据：

```
da <- read_table("m-ncdc-noaa-glbtemp.txt", col_names=FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double()
## )
```

```
Gtemp2 <- ts(da[[3]][1:1568], start=c(1880, 1), frequency=12)
```

建立与 MC1 类似的模型：

```
mod8 <- arima(Gtemp2, order=c(1,1,2),
              seasonal=list(order=c(0,0,2), period=12),
              fixed=c(NA, NA, NA, 0, NA))
mod8
```

```
##
## Call:
## arima(x = Gtemp2, order = c(1, 1, 2), seasonal = list(order = c(0, 0, 2), period = 12),
##     fixed = c(NA, NA, NA, 0, NA))
##
## Coefficients:
##          ar1      ma1     ma2  sma1    sma2
##       0.5817  -1.2414  0.2639     0  0.0854
## s.e.  0.0704   0.0827  0.0781     0  0.0243
##
## sigma^2 estimated as 0.0881:  log likelihood = -321.11,  aic = 652.21
```

因为数据不同，这里的 AIC 不能与前一数据的模型的 AIC 比较。

模型诊断：

```
tsdiag(mod8, gof=36)
```

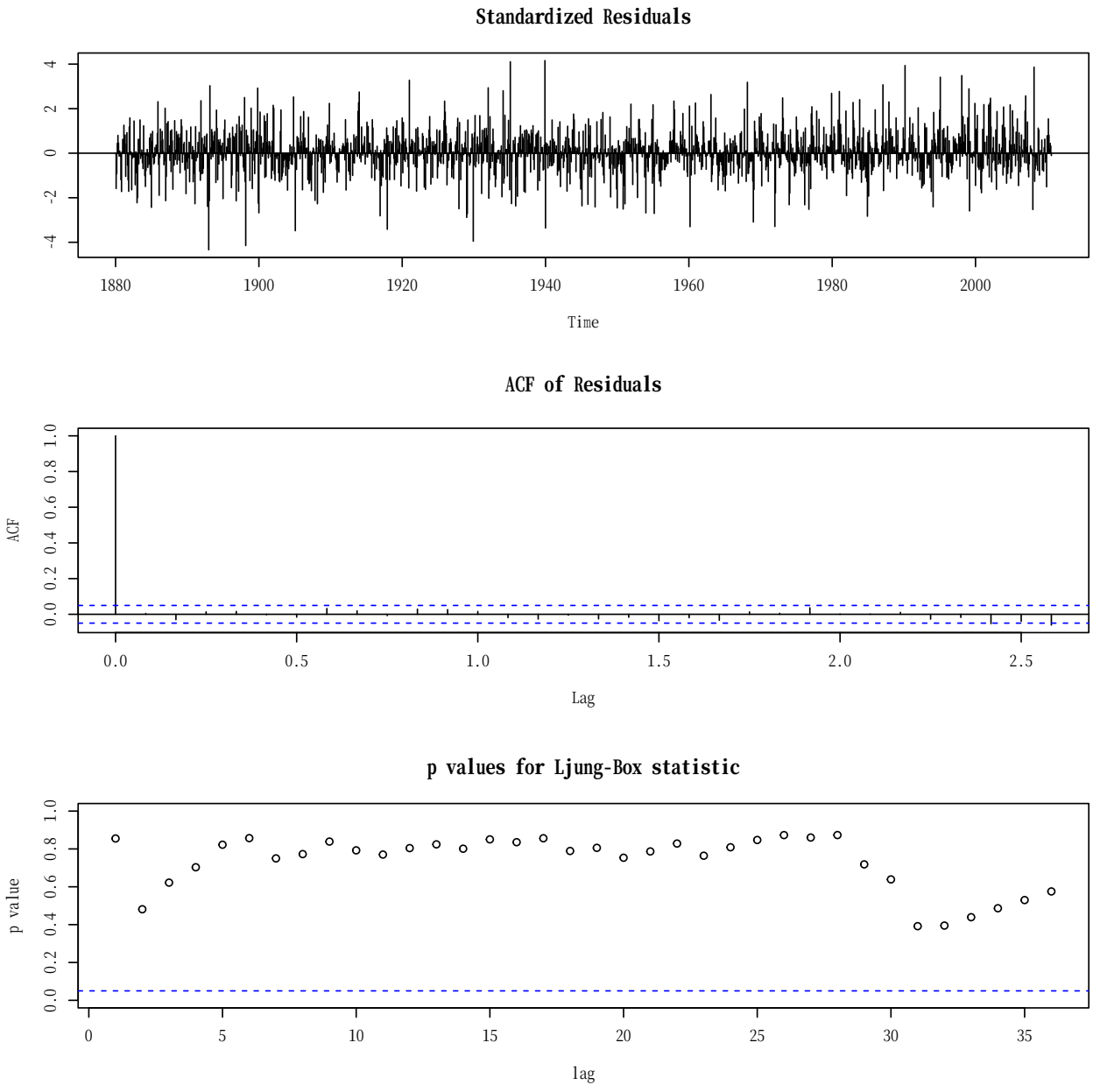


图 14.22: 用 NCDC 和 NOAA 数据建立随机趋势模型的诊断

诊断说明模型是可取的。

样本外比较略。

Chapter 15

线性时间序列案例学习—美国月失业率

失业率是每个国家、地区经济运行的重要指标。2011 年，美国的季节调整后的月度失业率在 9% 左右。

本章对美国月失业率数据进行建模和预测，使用不带解释变量和带解释变量的两种方法，解释变量是周首次申请失业救济金人数信息。

数据来自 Department of Labor, US Beareau of Labor Statistics。数据经过了季节调整。失业率为百分数 (单位是%)，是 16 周岁及以上的失业人口比例。每周首次申请失业救济金人数是每周新的申请失业保险的人数。失业率月度数据，从 1948 年 1 月到 2010 年 9 月，每月的第一天报告。周首次申请失业救济金人数是周数据，有效数据从 1967-01-07 到 2010-08-28，每周六报告。

本案例的目的是：

(1) 演示混合频率的时间序列的分析方法（这里是月频与周频）；
(2) 强调季节调整过的数据的建模有潜在的模型设置错误；
(3) 说明基于先验经验的试错法在模型设置中有帮助。

15.1 单变量时间序列模型

失业率数据从 1948 年 1 月到 2010 年 9 月，共 753 个观测值。

读入数据：

```
da <- read_table("m-unrate.txt", col_types = cols(.default = col_double()))
xts.unrate <- xts(
  da[["rate"]], with(da, make_date(Year, mon, dd)))
tclass(xts.unrate) <- "yearmon"
ts.unrate <- ts(da[["rate"]], start=c(1948,1), frequency = 12)
ts.dur <- diff(ts.unrate)
```

作时间序列图：

```
plot(
  xts.unrate, ylim=c(0, 12),
  main="US Monthly Unemployment Rate",
```

```
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col="red"
)
```

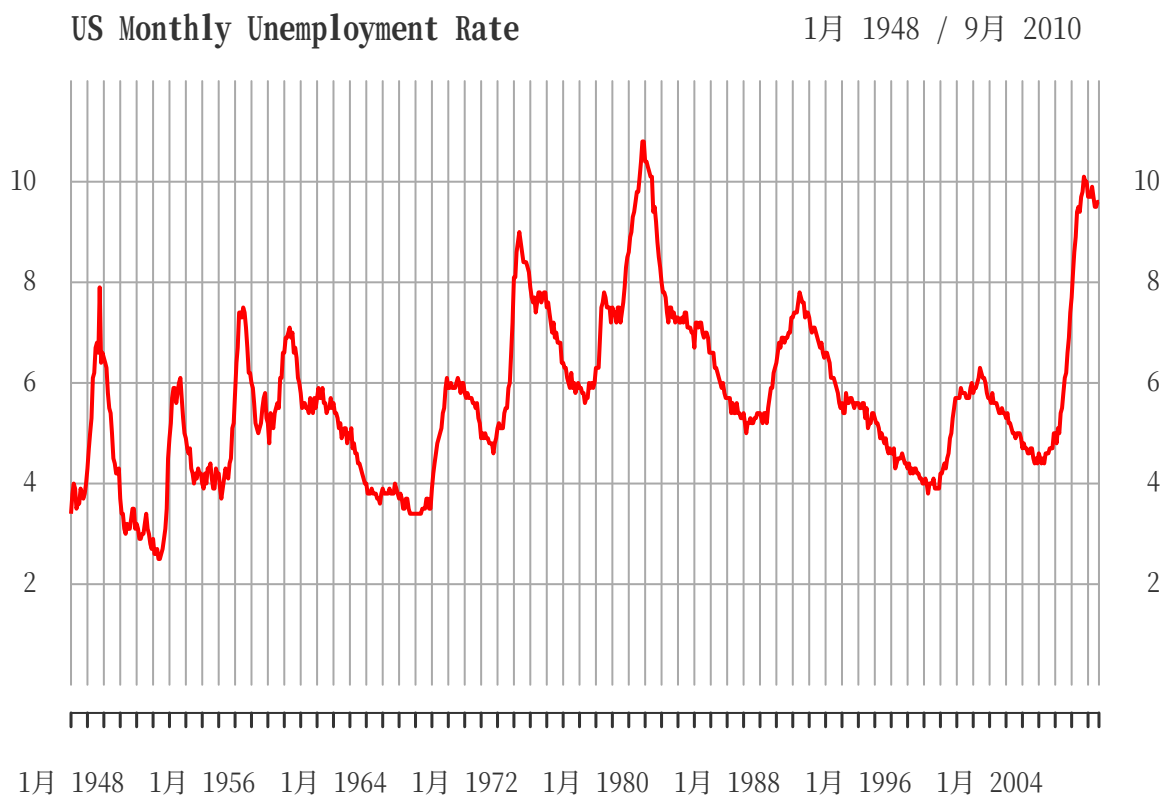


图 15.1: 美国月度失业率（经季节调整后）

分段作时间序列图：

```
plot(
  xts.unrate["1948/1968"], ylim=c(0, 12),
  main="US Monthly Unemployment Rate",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col="red"
)
```

```
plot(
  xts.unrate["1968/1988"], ylim=c(0, 12),
  main="US Monthly Unemployment Rate",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col="red"
```

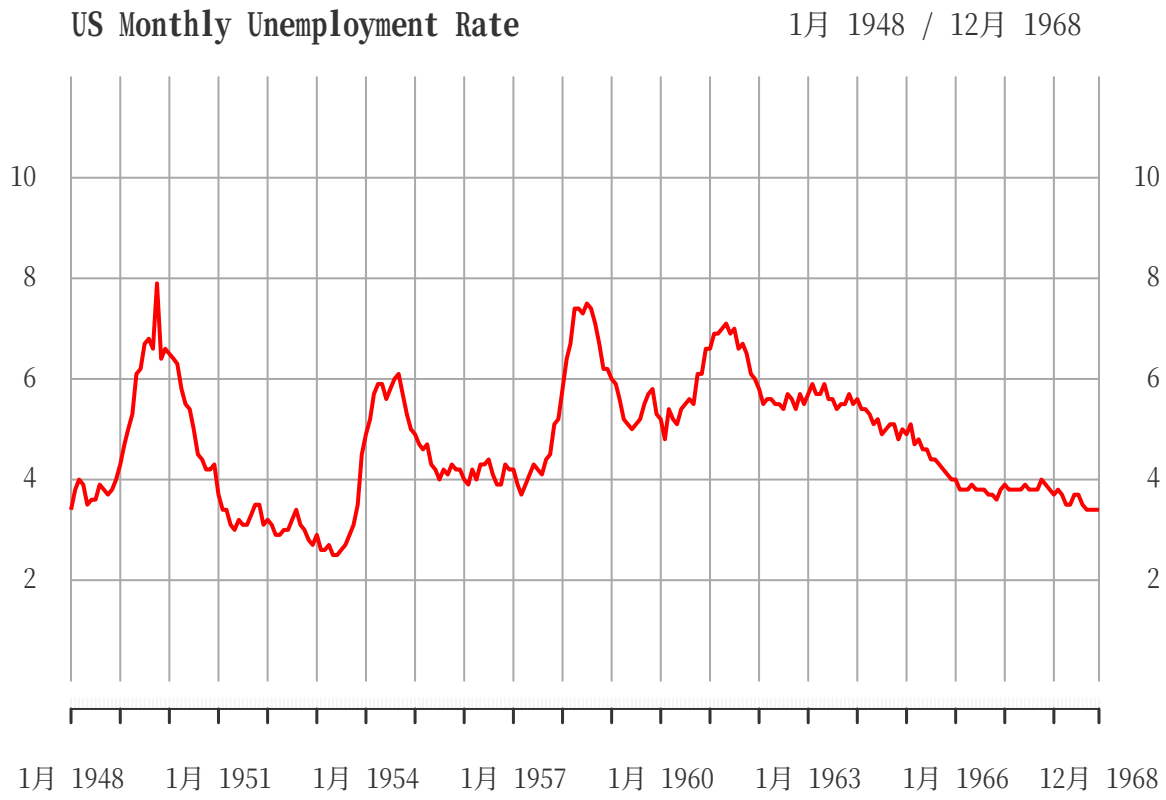


图 15.2: 美国月度失业率（经季节调整后）1948-1968

```
)
```

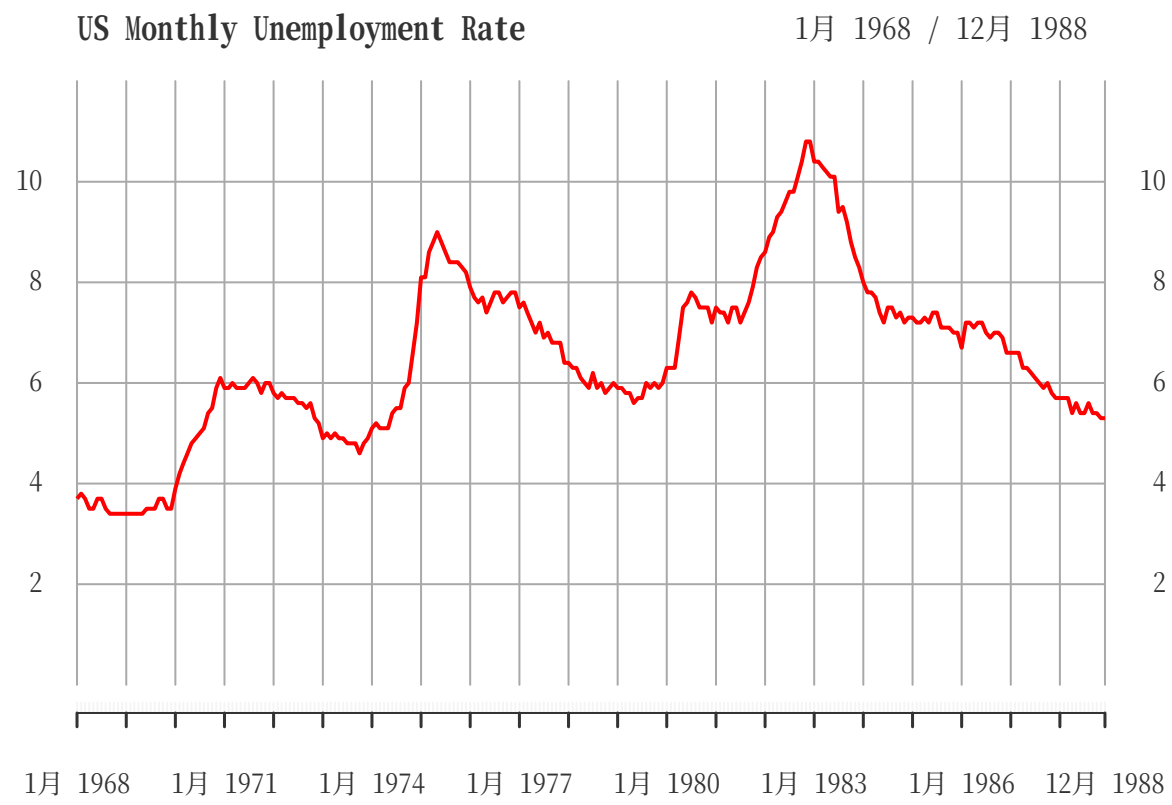


图 15.3: 美国月度失业率（经季节调整后）1968-1988

```
plot(
  xts.unrate["1988/"], ylim=c(0, 12),
  main="US Monthly Unemployment Rate",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years",
  col="red"
)
```

观察失业率的月度时间序列图发现：

(1) 失业率有周期性变化，但周期不固定；
(2) 失业率水平在数据的时间范围内可能有增长趋势；
(3) 失业率上升快，下降慢。这预示着失业率序列不服从线性时间序列。这里先不使用非线性时间序列。

记 x_t 为月度失业率序列，作其 ACF 图：

```
acf(ts.unrate, lag.max=36, main="")
```

这像是单位根非平稳。计算其差分序列，作 ACF 图：

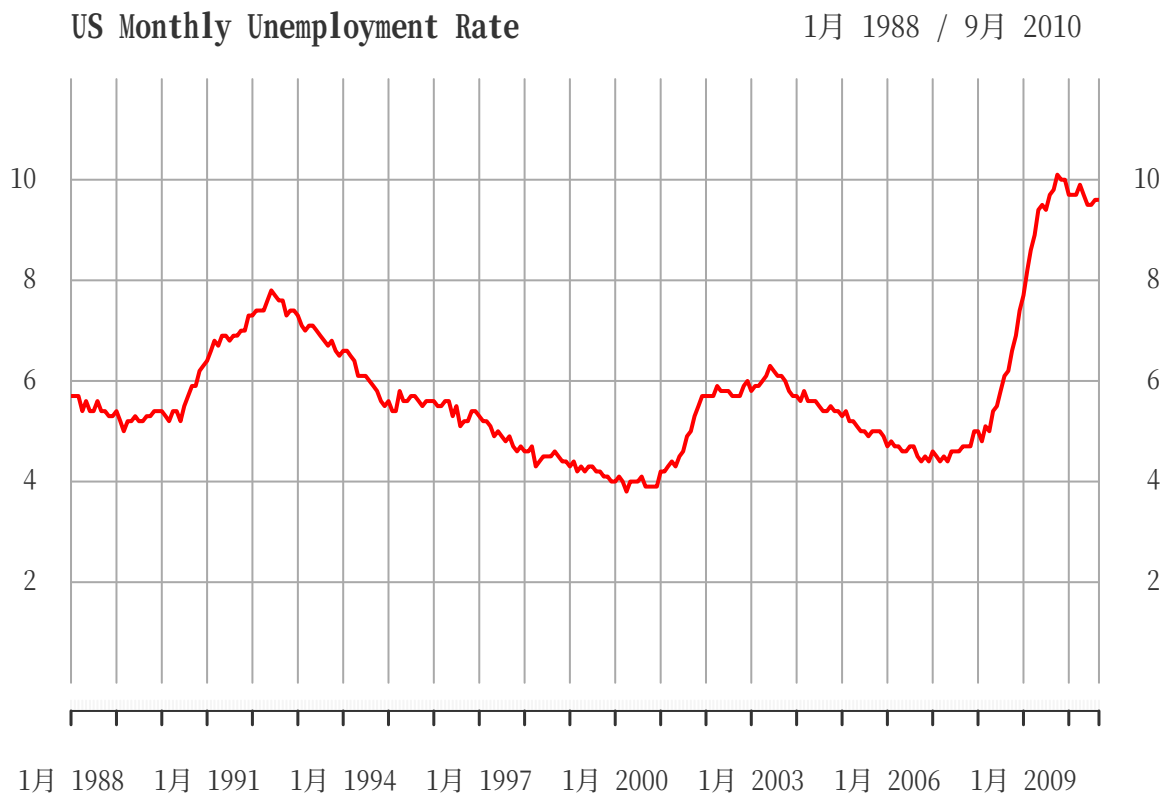


图 15.4: 美国月度失业率（经季节调整后）1988-2010

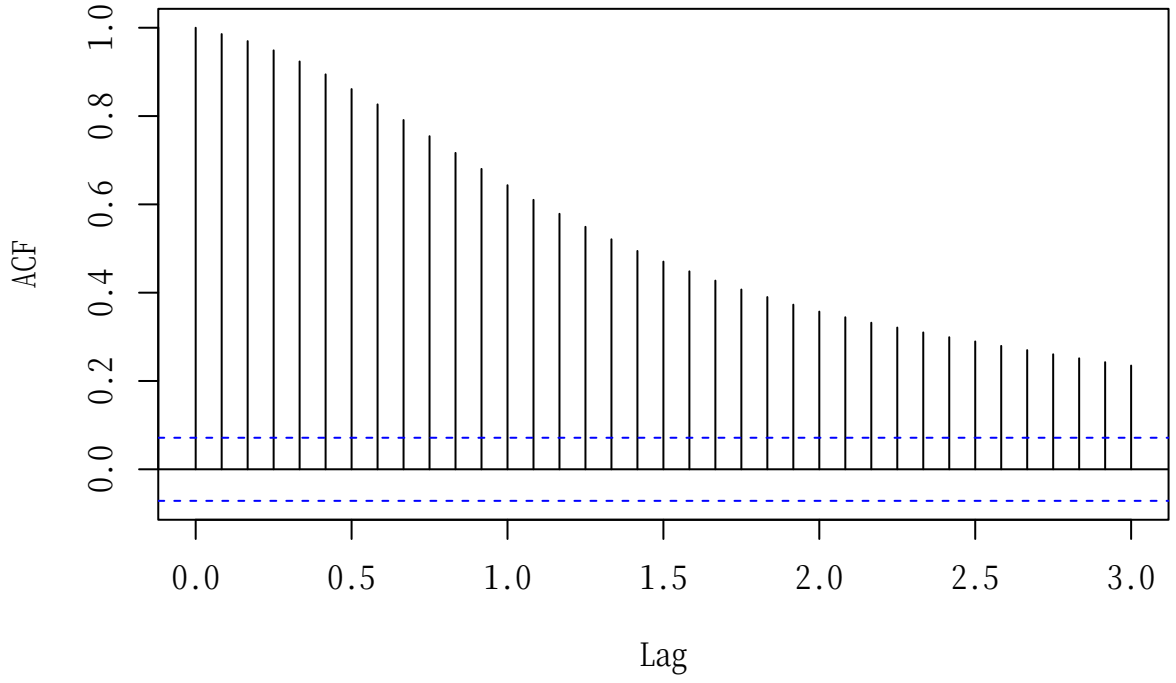


图 15.5: 美国月度失业率的 ACF

```
forecast::Acf(ts.dur, lag.max=36, main="")
```

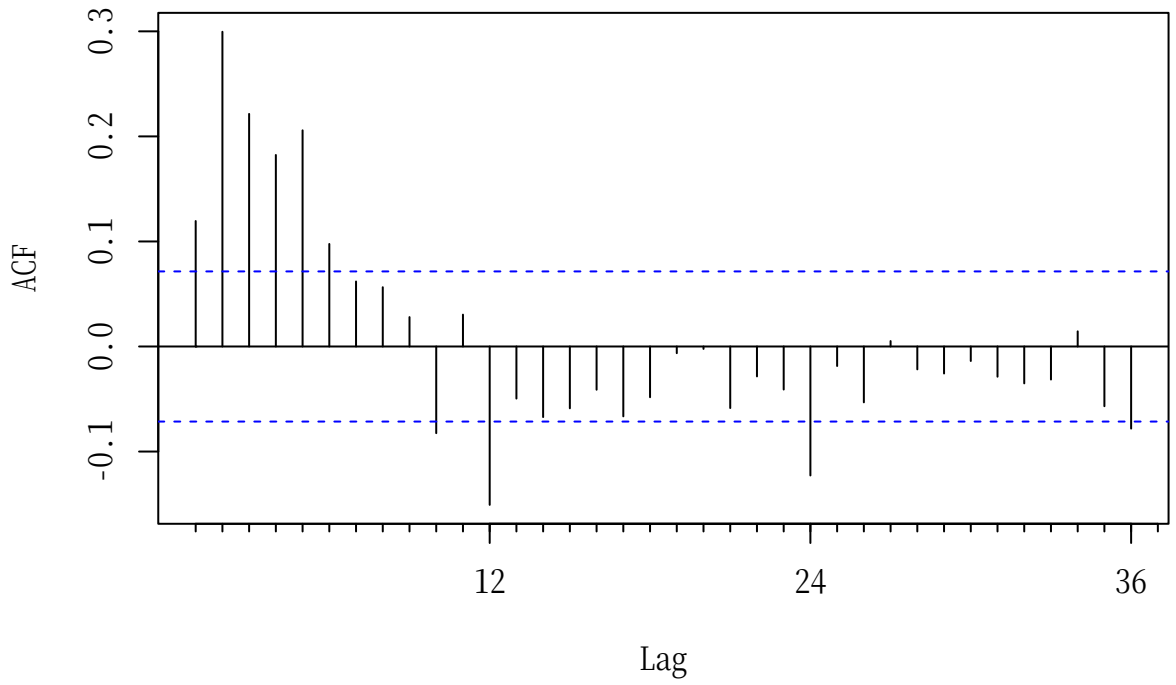


图 15.6: 美国月度失业率差分的 ACF

这不太像是低阶的 MA，在周期 12 的倍数有较高相关，怀疑是季节调整的残余周期性。

差分序列的 PACF:

```
pacf(ts.dur, lag.max=36, main="")
```

这也不像是低阶的 AR，而且在周期 12 的整倍数的偏自相关也超出。

对差分序列用 AIC 定阶建立 AR 模型：

```
ar(ts.dur, method="mle")
```

```
##
## Call:
## ar(x = ts.dur, method = "mle")
##
## Coefficients:
##       1        2        3        4        5        6        7        8
##  0.0114   0.2208   0.1536   0.1030   0.1319   0.0007  -0.0333   0.0047
##       9       10       11       12
## -0.0056  -0.1032   0.0302  -0.1174
```

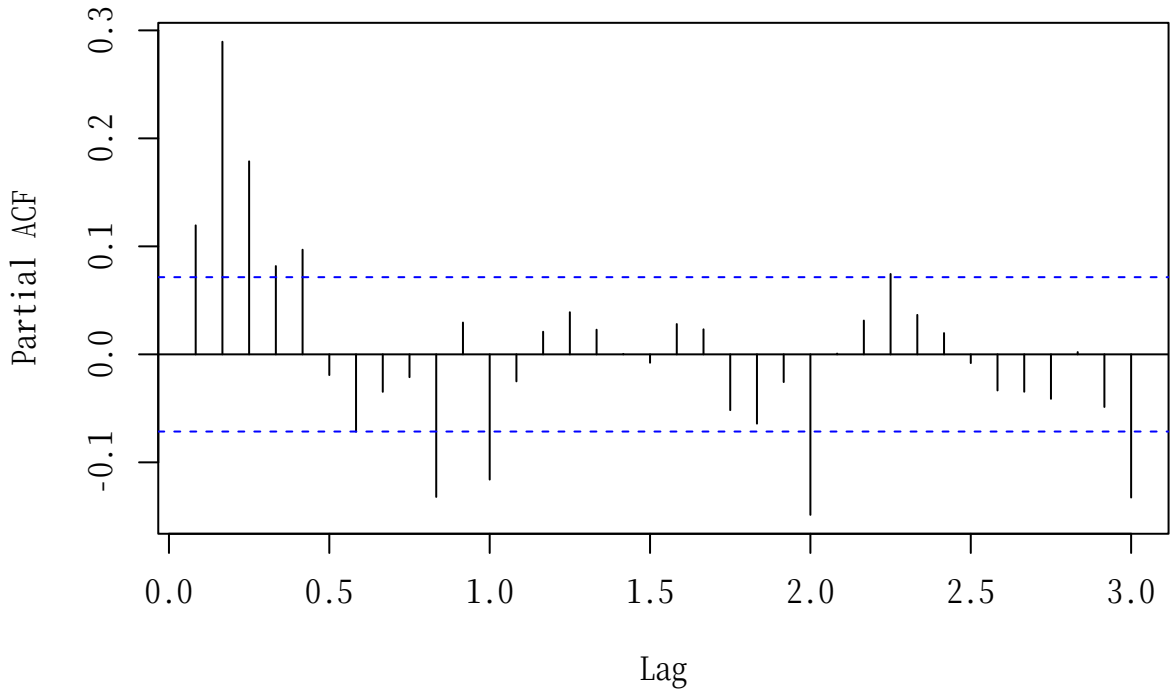


图 15.7: 美国月度失业率差分的 PACF

```
##
## Order selected 12  sigma^2 estimated as  0.03838
```

作单位根检验：

```
fUnitRoots::adfTest(ts.unrate, lags=12, type="ct")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 12
##   STATISTIC:
##     Dickey-Fuller: -2.8364
##   P VALUE:
##     0.2243
##
## Description:
##  Wed Mar 23 16:30:22 2022 by user: Lenovo
```

检验结果不显著，说明有单位根。

总结以上所见：

- 失业率序列很可能是单位根过程；
- 尽管已经进行了季节调整，差分序列的 ACF 和 PACF 在滞后为 12 倍数的地方还是较高，说明序列仍残余 12 个月周期的影响，可能还需要在模型中考虑增加季节项；
- 差分序列的 ACF 和 PACF 在滞后为 12 倍数处数值约 0.15，但是衰减很慢，这有些像是长记忆过程，当 ARMA 方程两边有相近特征根时可表现出类似长记忆性质；
- 除了季节性外，差分序列的 ACF 和 PACF 在 5 阶或 6 阶显著，PACF 指数震荡衰减；
- 可尝试 $p = 1$ 和 $q = 5$。取 $q = 5$ 是考虑到 PACF 的震荡衰减以及 ACF 在前 5 阶较高。取 $p = 1$ 是因为 ACF 也是指数衰减而不是快速截尾。

因此尝试如下的带有季节性的 $\text{ARIMA}(1,1,5)(1,0,1)_{12}$ 模型：

```
resm <- arima(
  ts.unrate, order=c(1,1,5),
  seasonal=list(order=c(1,0,1), period=12)
)
resm
```

```
##
## Call:
## arima(x = ts.unrate, order = c(1, 1, 5), seasonal = list(order = c(1, 0, 1),
##     period = 12))
##
```

```
## Coefficients:
##          ar1      ma1     ma2     ma3     ma4     ma5    sar1     sma1
##       0.7301  -0.7468  0.2194  0.0073  0.0383  0.0831  0.5978  -0.8469
## s.e.  0.0686   0.0776  0.0462  0.0501  0.0467  0.0431  0.0672   0.0477
##
## sigma^2 estimated as 0.03643:  log likelihood = 176.43,  aic = -334.87
```

删去不显著的 MA3 和 MA4 系数：

```
mod1 <- arima(
  ts.unrate, order=c(1,1,5),
  seasonal=list(order=c(1,0,1), period=12),
  fixed=c(NA, NA, NA, 0, 0, NA, NA, NA)
)
mod1
```

```
##
## Call:
## arima(x = ts.unrate, order = c(1, 1, 5), seasonal = list(order = c(1, 0, 1),
##     period = 12), fixed = c(NA, NA, NA, 0, 0, NA, NA, NA))
##
## Coefficients:
##          ar1      ma1     ma2  ma3  ma4     ma5    sar1     sma1
##       0.7536  -0.7744  0.2351    0    0  0.0990  0.6051  -0.8525
## s.e.  0.0569   0.0650  0.0365    0    0  0.0386  0.0654   0.0457
##
## sigma^2 estimated as 0.03649:  log likelihood = 175.75,  aic = -337.5
```

作模型残差检验：

```
tsdiag(mod1, gof=36)
```

结果表明模型充分。模型可以写成

$$(1-0.61B^{12})(1-0.75B)(1-B)x_t \tag{15.1}$$

$$=(1-0.85B^{12})(1-0.77B+0.24B^2+0.099B^5)\varepsilon_t, \tag{15.2}$$

$$\mathrm{Var}(\varepsilon_t)=0.0365, \quad \mathrm{AIC}=-337.5 \tag{15.3}$$

15.2 一个替代模型

因为考虑到需要有季节项，所以可以先建立一个仅有差分和季节项的模型，对残差分析再考虑增加其他 AR 或者 MA 成分。

```
resm2 <- arima(
  ts.unrate, order=c(0,1,0),
  seasonal=list(order=c(1,0,1), period=12)
```

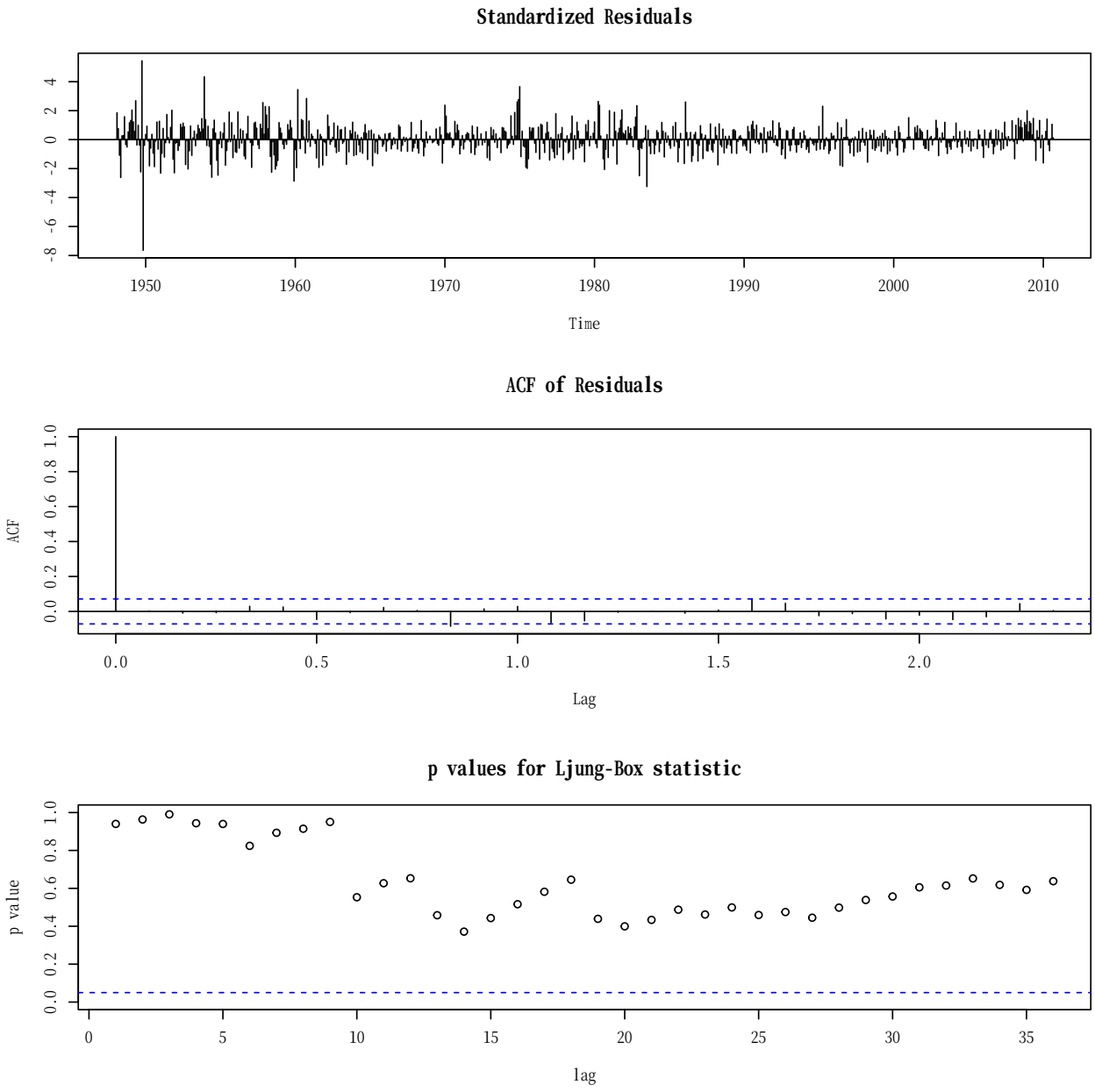


图 15.8: 失业率 ARIMA 模型的残差检验

```
)
resm2
```

```
##
## Call:
## arima(x = ts.unrate, order = c(0, 1, 0), seasonal = list(order = c(1, 0, 1),
##     period = 12))
##
## Coefficients:
##          sar1     sma1
##       0.6195  -0.8670
## s.e.  0.0658   0.0468
##
## sigma^2 estimated as 0.04267:  log likelihood = 116.9,  aic = -227.8
```

拟合的模型为

$$(1-0.62B^{12})x_t = (1-0.87B^{12})b_t,\ \sigma_b^2 = 0.04267$$

其中 b_t 表示残差序列。考察 b_t 的 ACF 和 PACF：

```
forecast::Acf(resm2$residuals, lag.max=36, main="")
```

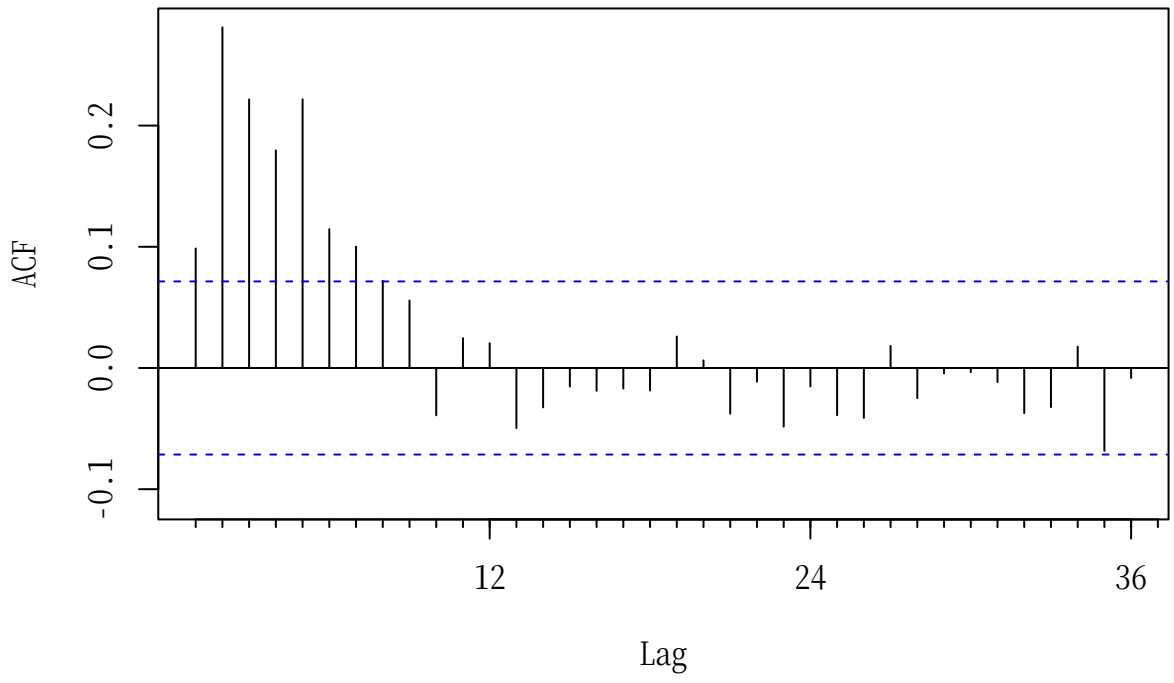


图 15.9: 单变量仅季节项的残差 ACF

残差 ACF 明显在前 6 阶不为零。

```
pacf(resm2$residuals, lag.max=36, main="")
```

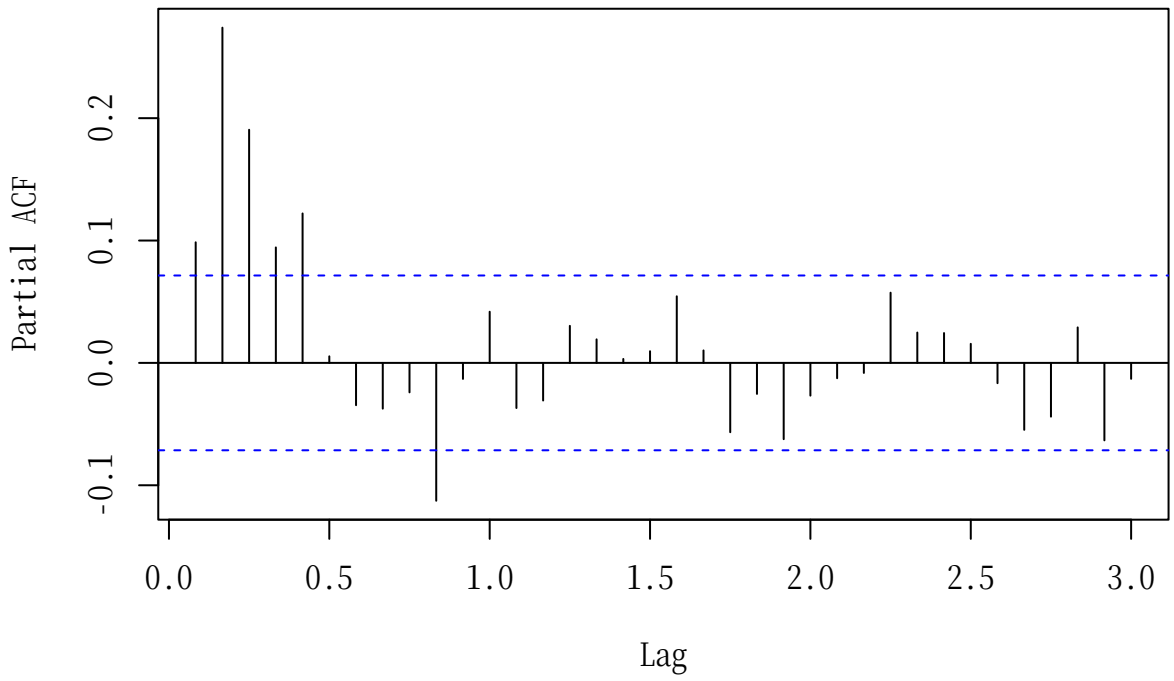


图 15.10: 单变量仅季节项的残差 PACF

PACF 呈现指数速度震荡衰减。所以，怀疑残差仍有 5 阶或 6 阶以上的 MA，

从 PACF 观察，尝试建立一个 AR(5)：

```
resm3 <- arima(
  ts.unrate, order=c(5,1,0),
  seasonal=list(order=c(1,0,1), period=12)
)
resm3
```

```
##
## Call:
## arima(x = ts.unrate, order = c(5, 1, 0), seasonal = list(order = c(1, 0, 1),
##     period = 12))
##
## Coefficients:
##           ar1     ar2     ar3     ar4     ar5    sar1     sma1
##       -0.0124  0.2101  0.1682  0.1024  0.1207  0.5624  -0.8233
```

```
## s.e.   0.0365  0.0366  0.0366  0.0370  0.0366  0.0723   0.0526
##
## sigma^2 estimated as 0.03663:  log likelihood = 174.57,  aic = -333.13
```

删去不显著的 ar1 系数：

```
fixed <- rep(NA_real_, 7); fixed[1] <- 0
mod2 <- arima(
  ts.unrate, order=c(5,1,0),
  seasonal=list(order=c(1,0,1), period=12),
  fixed=fixed
)
```

```
## Warning in arima(ts.unrate, order = c(5, 1, 0), seasonal = list(order = c(1, :
## some AR parameters were fixed: setting transform.pars = FALSE
```

```
mod2
```

```
##
## Call:
## arima(x = ts.unrate, order = c(5, 1, 0), seasonal = list(order = c(1, 0, 1),
##     period = 12), fixed = fixed)
##
## Coefficients:
##       ar1     ar2     ar3     ar4     ar5    sar1     sma1
##         0  0.2104  0.1652  0.0996  0.1194  0.5643  -0.8240
## s.e.    0  0.0366  0.0355  0.0362  0.0364  0.0724   0.0528
##
## sigma^2 estimated as 0.03664:  log likelihood = 174.51,  aic = -335.02
```

其 AIC 的值为-335，比 mod1 的 AIC=-337 略差一些。

模型可以写成：

$$(1-0.56B^{12})(1-0.21B^2-0.17B^3-0.10B^4-0.12B^5)(1-B)x_t \tag{15.4}$$

$$=(1-0.82B^{12})\varepsilon_t, \quad \sigma_\varepsilon^2=0.03664, \quad \text{AIC}=-335 \tag{15.5}$$

模型检验：

```
tsdiag(mod2, gof=36)
```

此模型也可以接受。

15.3 模型比较

比较(15.3)的 mod1 和(15.5)的 mod2。

- 从残差诊断来看，两个模型都很合适。

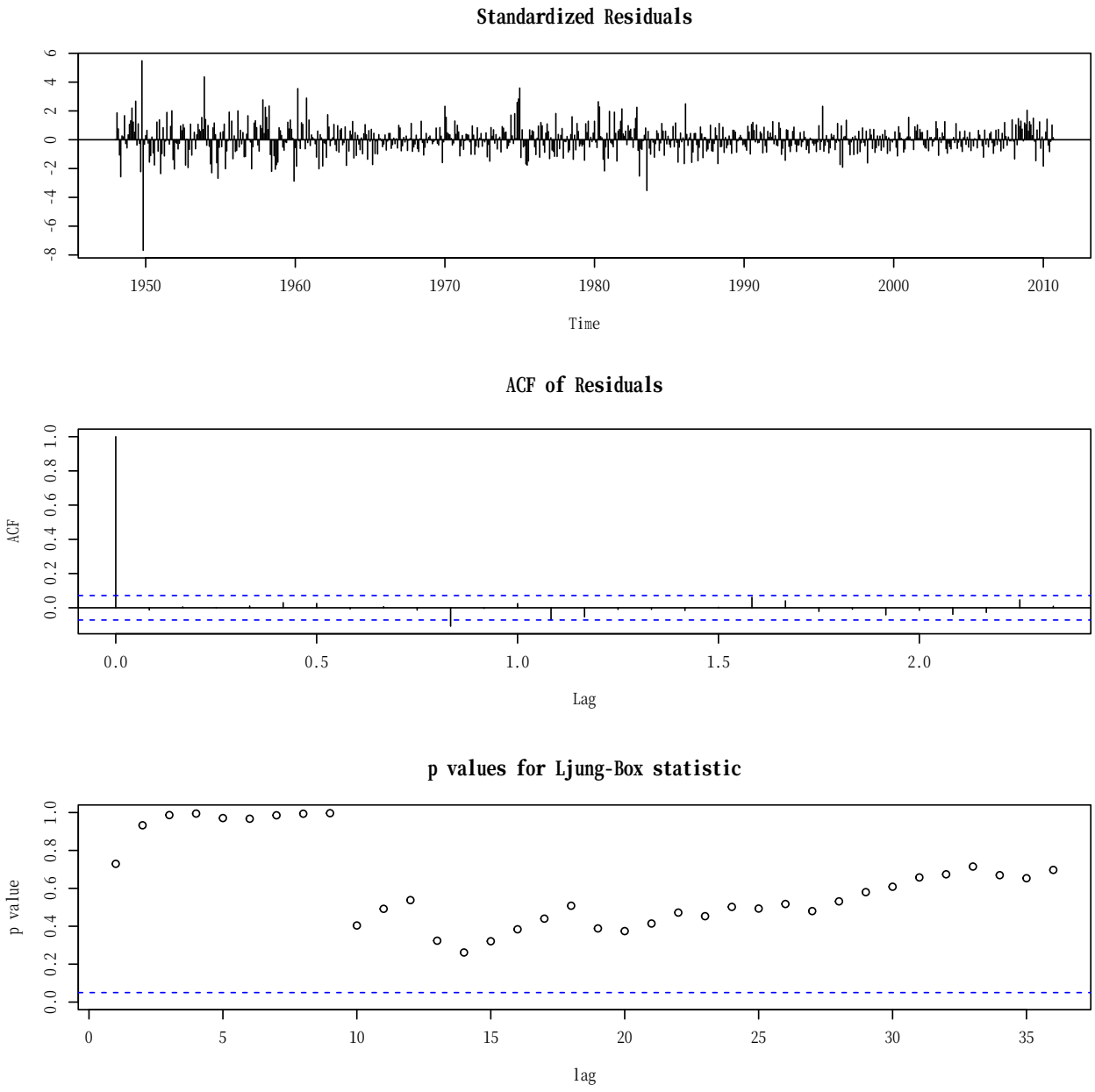


图 15.11: 单变量候选模型检验

- 从 AIC 比较，mod1 为 −337，mod2 为 −335，mod1 略占优。

下面进行样本外比较。保留样本的最后 53 个观测值，共有 753 个观测值，初始使用前 700 个建模，逐步地作超前一步预报。

使用 mod1 作超前一步预报的函数为：

```
pred1 <- function(y=ts.unrate, n.pred=73){
  T <- length(y)
  ypred <- rep(NA_real_, T)
  for(h in (T-n.pred):(T-1)){
    mod <- arima(
      y[1:h], order=c(1,1,5),
      seasonal=list(order=c(1,0,1), period=12),
      fixed=c(NA, NA, NA, 0, 0, NA, NA, NA)
    )
    ypred[h+1] <- predict(mod, n.ahead=1, se.fit=FALSE)
  }

  ypred
}
```

使用 mod2 作超前一步预报的函数为：

```
pred2 <- function(y=ts.unrate, n.pred=73){
  T <- length(y)
  ypred <- rep(NA_real_, T)
  fixed <- rep(NA_real_, 7); fixed[1] <- 0
  for(h in (T-n.pred):(T-1)){
    mod <- arima(
      y[1:h], order=c(5,1,0),
      seasonal=list(order=c(1,0,1), period=12),
      fixed=fixed, transform.pars=FALSE
    )
    ypred[h+1] <- predict(mod, n.ahead=1, se.fit=FALSE)
  }

  ypred
}
```

对一种方法的预测计算预测根均方误差 (RMFSE) 和平均绝对预测误差 (MAFE) 的函数：

```
pred.err <- function(y, n.pred=length(ypred), ypred){
  T <- length(y)
  i1 <- T - n.pred + 1
  RMFSE <- sqrt( mean((y[i1:T] - ypred[i1:T])^2) )
  MAFE <- mean(abs(y[i1:T] - ypred[i1:T]))
```

```
  c(RMFSE=RMFSE, MAFE=MAFE)
}
```

样本外预测比较：

```
err1 <- pred.err(y=ts.unrate, n.pred=73,
                 ypred=pred1(ts.unrate, n.pred=73))
err2 <- pred.err(y=ts.unrate, n.pred=73,
                 ypred=pred2(ts.unrate, n.pred=73))
err.tab1 <- data.frame(
  Model=c("mod1", "mod2"),
  RMFSE=c(err1[["RMFSE"]], err2[["RMFSE"]]),
  MAFE=c(err1[["MAFE"]], err2[["MAFE"]])
)
knitr::kable(err.tab1, digits=4)
```

Model	RMFSE	MAFE
mod1	0.1503	0.1182
mod2	0.1514	0.1185

两个模型的样本外预测效果也不相上下，mod1 略占优。事实上，如果不考虑季节因素，将 AR 改写成 MA 表示，两个模型十分接近。

15.4 使用首次申请失业救济金人数

这一节用周首次申请失业救济金人数作为解释变量来预测失业率。因为申请人数数据是从 1967 年 1 月开始才有，所以尽管失业率数据从 1948 年 1 月开始就有，我们要使用共同的数据存在的期间。

因为每个月的 1 号报告失业率，为了预报失业率只能使用前一个月的申请人数数据。实际使用的失业率数据跨度为 1967 年 2 月到 2010 年 9 月，周首次申请失业救济金人数数据跨度为 1967 年 1 月到 2010 年 8 月。用到的失业率的有效数据点数为 524 个。

周频数据与月频数据如何匹配？一种办法是将 x_t 对应的第 $t-1$ 月的所有周的申请人数加起来得到一个解释变量 c_{t-1}，就频率相同了。另一种办法是将第 $t-1$ 个月的前 4 周的申请人数 $w_{1,t-1}, w_{2,t-1}, w_{3,t-1}, w_{4,t-1}$ 都作为解释变量，如果有第 5 周也不用，c_{t-1} 不一定等于这四个变量的和。为避免数值计算量纲差距太大，申请人数原来以千人为单位，改为以百万人为单位。

```
da2 <- read_table("m-unrateic.txt",
                  col_types = cols(.default = col_double()))
da2[,5:9] <- da2[,5:9]/1000
xts.unic <- xts(
  da2[,-(1:3)], with(da2, make_date(year, mon, dd)))
tclass(xts.unic) <- "yearmon"
unrate <- da2[["rate"]]
head(da2)
```

```
## # A tibble: 6 x 9
```

```
##    year   mon    dd  rate  w1m1  w2m1  w3m1  w4m1  icm1
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  1967     2     1   3.8 0.208 0.207 0.217 0.204 0.836
## 2  1967     3     1   3.8 0.216 0.229 0.229 0.242 0.916
## 3  1967     4     1   3.8 0.31  0.241 0.245 0.247 1.04
## 4  1967     5     1   3.8 0.259 0.257 0.299 0.245 1.32
## 5  1967     6     1   3.9 0.254 0.231 0.23  0.228 0.943
## 6  1967     7     1   3.8 0.248 0.238 0.224 0.218 0.928
```

两个序列的图形：

```
plot(xts.unic[,c("rate", "icm1")],
     multi.panel=TRUE, yaxis.same=FALSE,
     main=" 月失业率与上个月申请失业救济金人数（单位：百万人）",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years",
     col=c("red", "green"))
```

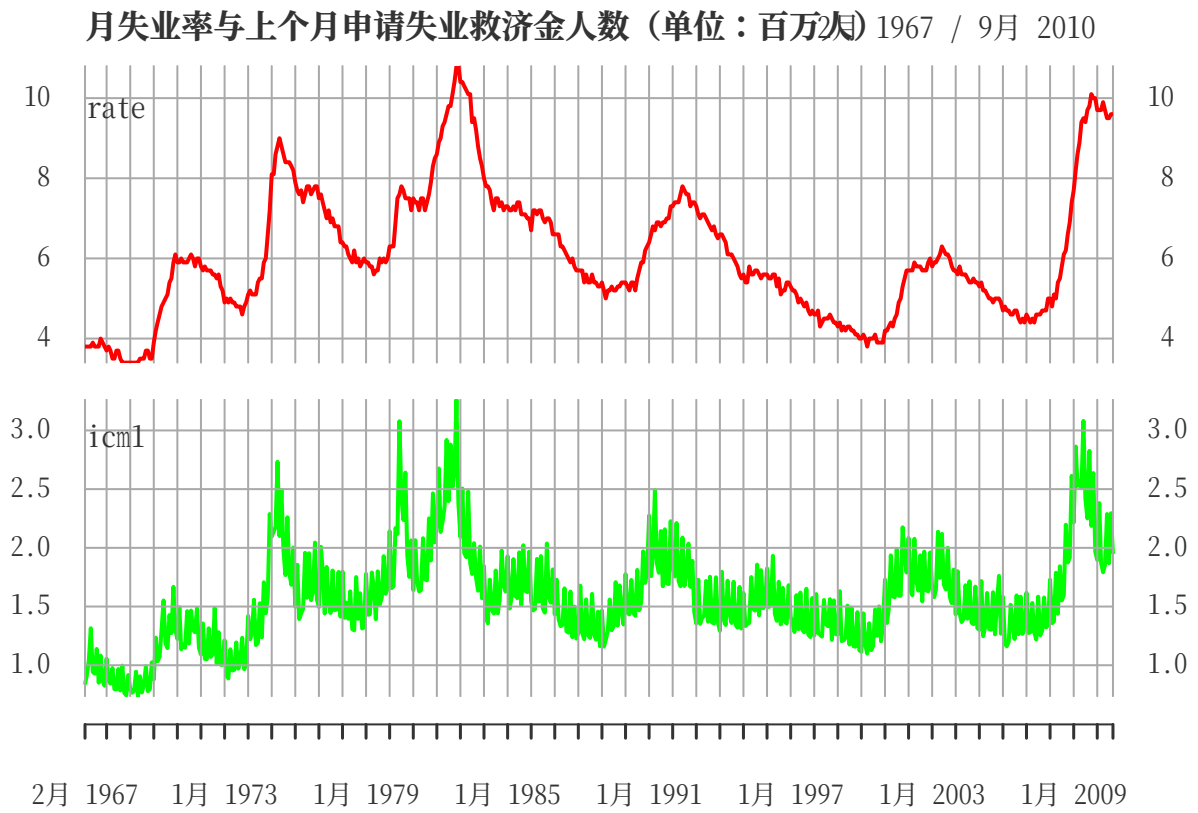


图 15.12: 月失业率与上个月申请失业救济金人数（单位：百万人）

失业率对上个月申请人数的散点图：

```
plot(da2[["icm1"]], da2[["rate"]], type="p",
     pch=16, col="gray",
     xlab=" 上个月申请失业救济金人数（单位：百万人）",
     ylab=" 月失业率")
```

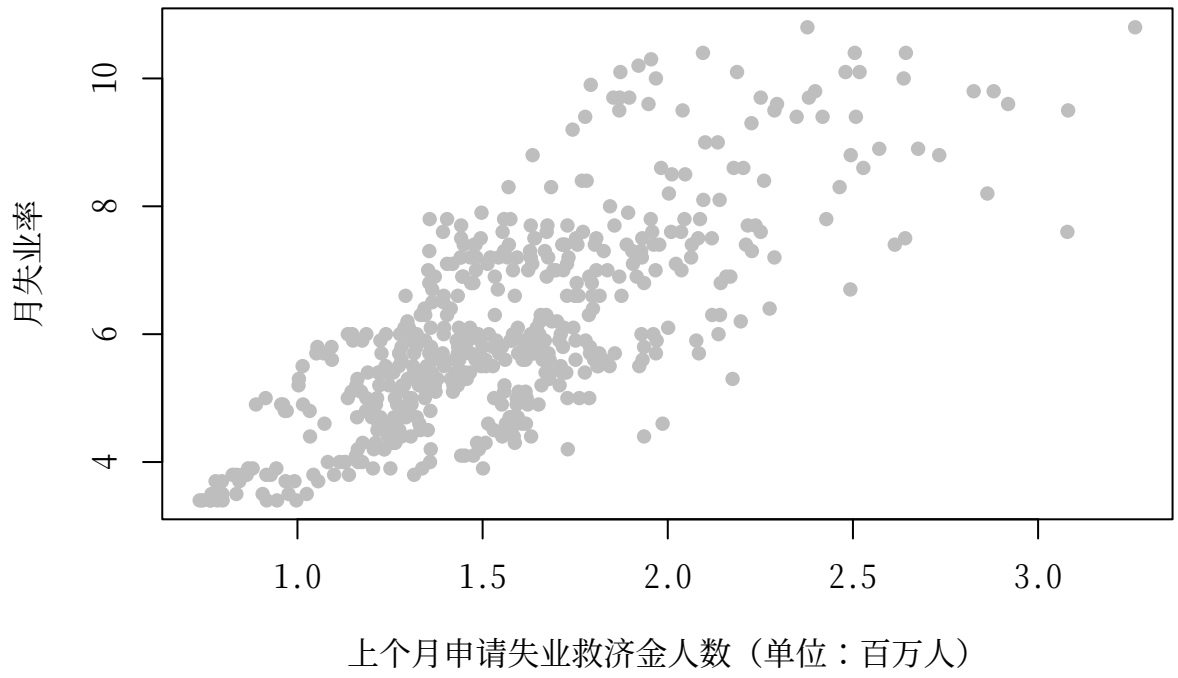


图 15.13: 月失业率对上个月申请失业救济金人数

两者存在一定的线性相关性。相关系数：

```
cor(da2[["icm1"]], da2[["rate"]])
```

```
## [1] 0.7570642
```

15.4.1 使用前一月申请人数总和

图15.13给出了从 1967 年 2 月到 2010 年 9 月的月失业率 x_t 的时序图，以及对应的前一个月的首次申请失业救济金人数的 c_{t-1} 序列的时序图。可以看出两者呈现一种共同的运动。

先拟合一个简单的一元线性回归模型：

```
lm1 <- lm(rate ~ icm1, data=da2)
summary(lm1)
```

```
##
## Call:
```

```
## lm(formula = rate ~ icm1, data = da2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8638 -0.7008 -0.1299  0.7366  3.1746
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.5202     0.1785   8.518   <2e-16 ***
## icm1          2.9047     0.1097  26.475   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.051 on 522 degrees of freedom
## Multiple R-squared:  0.5731, Adjusted R-squared:  0.5723
## F-statistic: 700.9 on 1 and 522 DF,  p-value: < 2.2e-16
```

结果的回归复相关系数平方为 0.57，说明有一定预报能力。模型为

$$x_t = 1.52 + 2.905c_{t-1} + n_t$$

其中 n_t 为误差项。

但是，这个回归有两个问题：

- 残差很可能有序列自相关，使得回归的检验是不可用的，估计也不是最有效的；
- 如果回归因变量和自变量存在单位根非平稳，结果可能是虚假的。

回归残差的 ACF:

```
lm1 <- lm(rate ~ icm1, data=da2)
acf(lm1$residuals, lag.max=36, main="")
```

残差的 ACF 衰减缓慢，可能由于原始序列有单位根。

回归残差的 PACF:

```
lm1 <- lm(rate ~ icm1, data=da2)
pacf(lm1$residuals, lag.max=36, main="")
```

前 3 个 PACF 很较高，后面也有多处超出界限。从回归残差的 ACF 和 PACF 来看不一定能用 ARMA 模型建模。

为克服模型设定上的困难，我们应用试错法和一些先验知识。

(1) 由于是月度数据，模型可能需要季节项。
(2) 失业率呈现出一定的非固定的周期波动，来源于商业周期，这表明失业率模型的一些特征根是复根，因此 AR 的阶在 2 以上；
(3) 回归残差的 ACF 都是正的，可考虑 MA 阶为整数，试选取 $q = 3$。

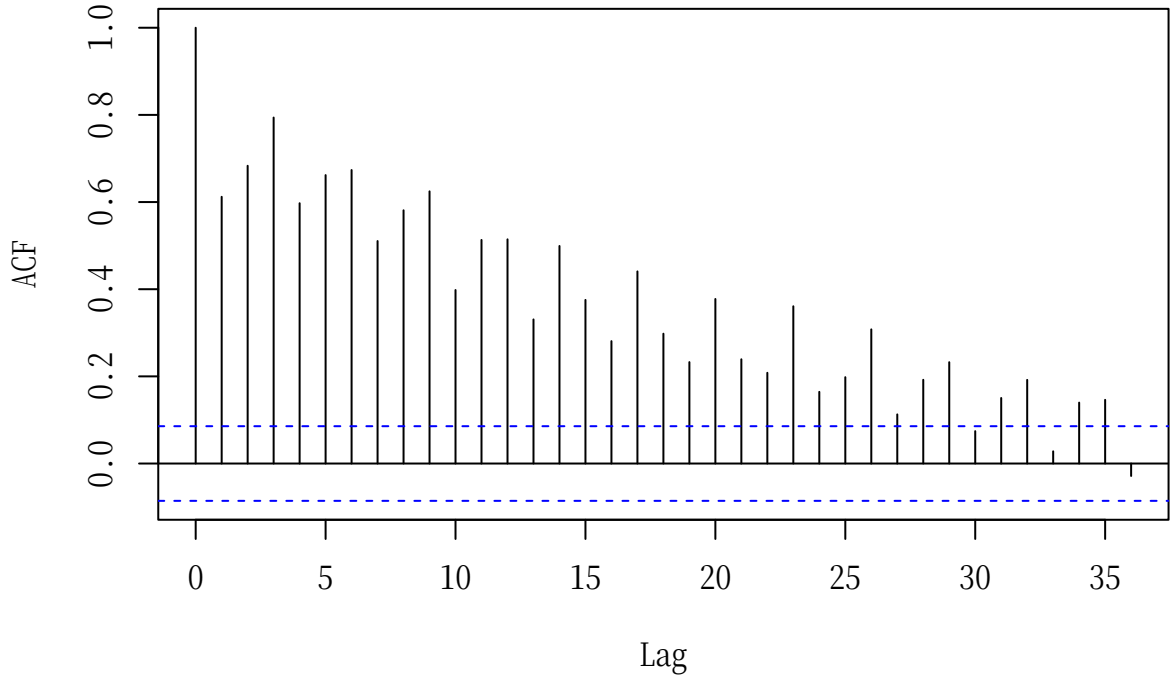


图 15.14: 以月申请人数为解释变量的回归残差的 ACF

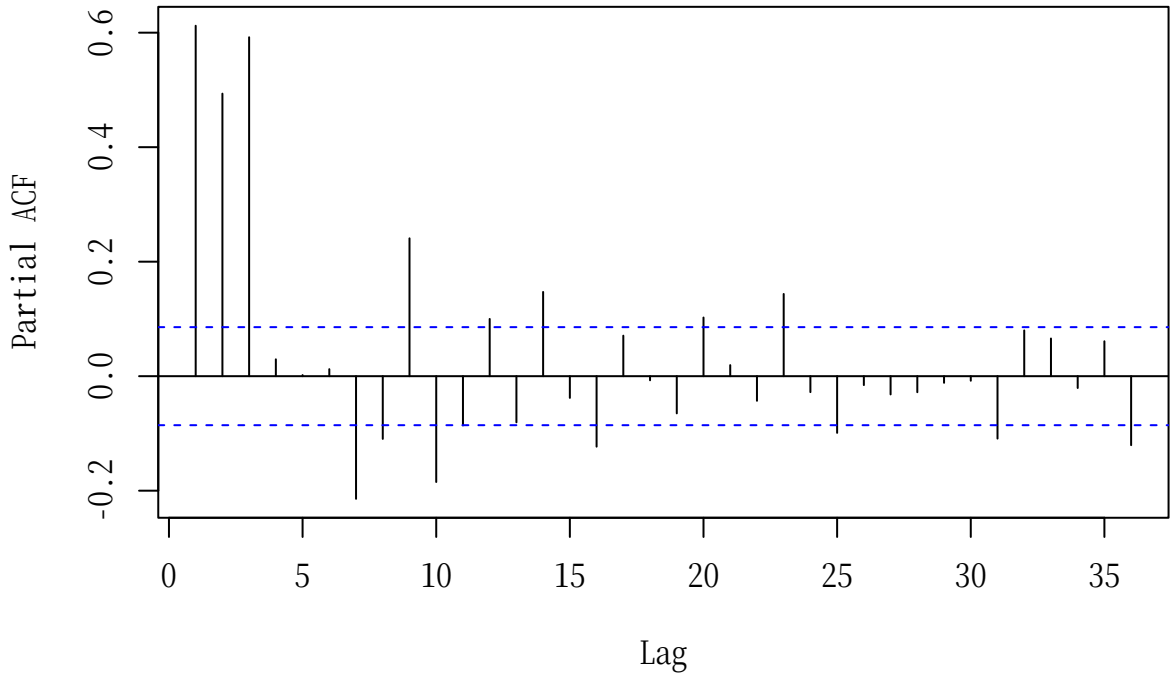


图 15.15: 以月申请人数为解释变量的回归残差的 PACF

暂定失业率带有月申请人数解释变量的模型为：

$$(1-\phi_1 B-\phi_2 B^2)(1-\Phi B^{12})(x_t-\beta_0-\beta_1 c_{t-1}) \tag{15.6}$$
$$=(1+\theta_1 B+\theta_2 B^2+\theta_3 B^3)(1+\Theta B^{12})a_t \tag{15.7}$$

```
resm4 <- arima(
  da2[["rate"]], xreg=da2[["icm1"]],
  order=c(2,0,3),
  seasonal=list(order=c(1,0,1), period=12)
); resm4
```

```
##
## Call:
## arima(x = da2[["rate"]], order = c(2, 0, 3), seasonal = list(order = c(1, 0,
##     1), period = 12), xreg = da2[["icm1"]])
##
## Coefficients:
##          ar1      ar2      ma1     ma2     ma3    sar1     sma1  intercept
##       1.8997  -0.9021  -0.8932  0.1458  0.0555  0.6501  -0.8520     6.0373
## s.e.  0.0332   0.0331   0.0543  0.0565  0.0466  0.0824   0.0586     0.3706
##       da2[["icm1"]]
##              0.0772
## s.e.         0.0212
##
## sigma^2 estimated as 0.02419:  log likelihood = 227.7,  aic = -435.39
```

因为数据不同，这里的 AIC 不与前面的单变量模型比较。结果表明 ma3 系数不显著，所以 MA 改为 $q=2$ 建模：

```
mod3 <- arima(
  da2[["rate"]], xreg=da2[["icm1"]],
  order=c(2,0,2),
  seasonal=list(order=c(1,0,1), period=12)
); mod3
```

```
##
## Call:
## arima(x = da2[["rate"]], order = c(2, 0, 2), seasonal = list(order = c(1, 0,
##     1), period = 12), xreg = da2[["icm1"]])
##
## Coefficients:
##          ar1      ar2      ma1     ma2    sar1     sma1  intercept
##       1.9123  -0.9145  -0.9100  0.1860  0.6465  -0.8483     6.1111
## s.e.  0.0283   0.0282   0.0527  0.0479  0.0823   0.0591     0.3748
##       da2[["icm1"]]
##              0.0782
## s.e.         0.0213
```

```
##
## sigma^2 estimated as 0.02426:  log likelihood = 226.97,  aic = -435.93
```

各个系数都显著。模型的残差诊断：

```
tsdiag(mod3, gof=36)
```

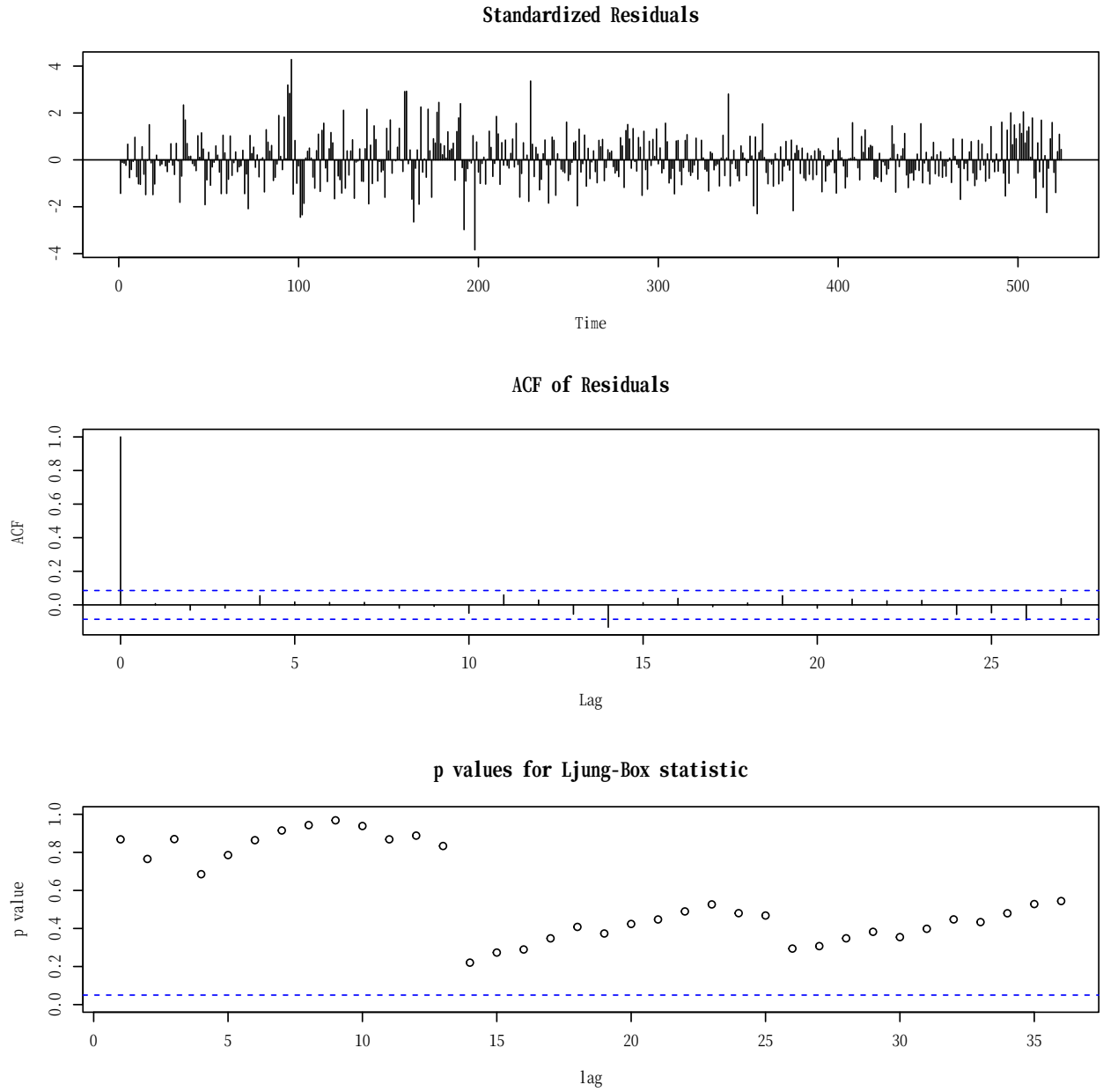


图 15.16: 用月申请人数为解释变量的 ARIMA 模型的诊断

除了残差有较多异常值以外，残差呈现出白噪声性质。

模型 mod3 可以写成：

$$(1-1.91B+0.91B^2)(1-0.65B^{12})(x_t-6.11-0.0782c_{t-1}) \tag{15.8}$$

$$=(1-0.91B+0.18B^2)(1-0.84B^{12})a_t,\ \mathrm{Var}(a_t)=0.02426,\ \mathrm{AIC}=-435.93 \tag{15.9}$$

模型中季节项的两个系数比较接近，这往往可以产生部分长记忆的现象。

模型中 AR 部分除了有一个对应于周期 12 的特征根以外，其它的特征根和对应的周期为：

```
polyroot(c(1, -coef(mod3)[1:2]))
```

```
## [1] 1.045522+0.019388i 1.045522-0.019388i
```

```
2*pi/Arg(polyroot(c(1, -coef(mod3)[1:2]))[1])/12
```

```
## [1] 28.23933
```

模型代表的周期为 28 个月。

15.4.2 使用前一月各周申请人数

这里使用前一个月的前四周的申请人数都作为自变量。下图为失业率与前一个月的前两周的申请人数的数据，三个序列走势是相近的：

```
plot(xts.unic[,c("rate", "w1m1", "w2m1")],
     multi.panel=TRUE, yaxis.same=FALSE,
     main=" 月失业率与上个月前两周申请失业救济金人数（单位：百万人）",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years",
     col=c("red", "green", "cyan"))
```

用四个解释变量作多元线性回归：

```
lm2 <- lm(rate ~ w1m1 + w2m1 + w3m1 + w4m1, data=da2)
summary(lm2)
```

```
##
## Call:
## lm(formula = rate ~ w1m1 + w2m1 + w3m1 + w4m1, data = da2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.40629 -0.67602 -0.06686  0.62419  2.55888
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5160     0.1651   3.125 0.001877 **
## w1m1          6.5221     2.1145   3.084 0.002148 **
## w2m1          9.6711     2.8630   3.378 0.000785 ***
```

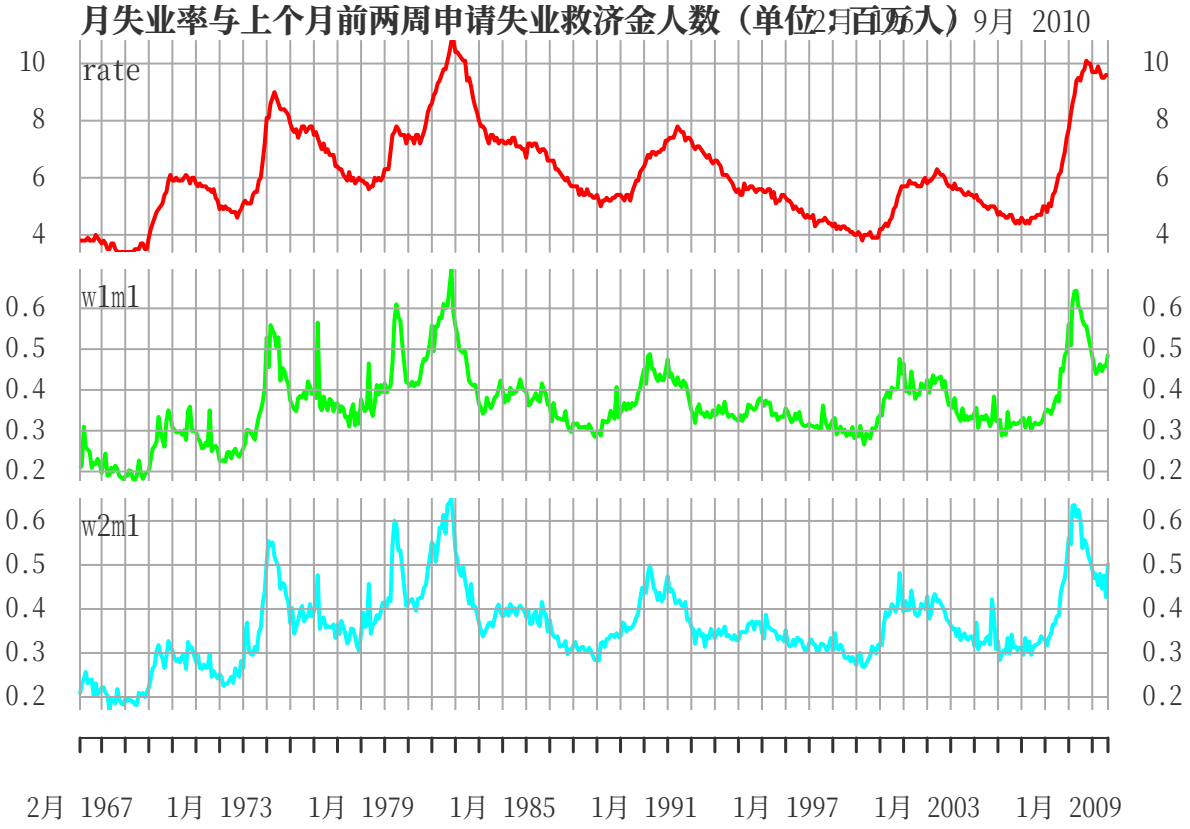


图 15.17: 月失业率与上个月前两周申请失业救济金人数（单位：百万人）

```
## w3m1         -2.4455     2.7980  -0.874 0.382506
## w4m1          1.6626     2.0624   0.806 0.420528
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8841 on 519 degrees of freedom
## Multiple R-squared:  0.6998, Adjusted R-squared:  0.6975
## F-statistic: 302.5 on 4 and 519 DF,  p-value: < 2.2e-16
```

复相关系数平方为 0.70，高于仅使用总数作为解释变量时的 0.57。当然，线性回归每增加一个解释变量其复相关系数平方都会增加，有过度拟合危险。这个回归的可能问题与用总数作为解释变量的情形类似，另外，模型中后第三、四周的解释变量不显著。这并不表明第三、四周的申请人数对下个月失业率没有预报能力，而是其预报能力可以被第一、二周的数据代替。比如，仅使用第三、四周建模：

```
lm2b <- lm(rate ~ w3m1 + w4m1, data=da2)
summary(lm2b)
```

```
##
## Call:
## lm(formula = rate ~ w3m1 + w4m1, data = da2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62420 -0.69034 -0.07828  0.63761  2.89772
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.6694     0.1707   3.920   0.0001 ***
## w3m1         10.2363     2.1342   4.796 2.11e-06 ***
## w4m1          4.7578     2.1012   2.264   0.0240 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9226 on 521 degrees of freedom
## Multiple R-squared:  0.6718, Adjusted R-squared:  0.6706
## F-statistic: 533.3 on 2 and 521 DF,  p-value: < 2.2e-16
```

结果仍是显著的。

下面仅使用第一、二周的申请人数预报下个月的失业率：

```
lm3 <- lm(rate ~ w1m1 + w2m1, data=da2)
summary(lm3)
```

```
##
## Call:
## lm(formula = rate ~ w1m1 + w2m1, data = da2)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.40653 -0.66933 -0.05949  0.63075  2.56954
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5127     0.1647   3.113  0.00195 **
## w1m1          6.4594     2.0709   3.119  0.00191 **
## w2m1          8.9609     2.0872   4.293  2.1e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8832 on 521 degrees of freedom
## Multiple R-squared:  0.6993, Adjusted R-squared:  0.6981
## F-statistic: 605.8 on 2 and 521 DF,  p-value: < 2.2e-16
```

复相关系数平方仍为 0.70，相比使用四个解释变量的模型基本没有下降。

仅使用前两周的数据预报还有一个额外的好处，就是给出预报的时间提前了，可以在前一个月 15 号的时候就预报下个月的失业率，如果使用总数则需要在月底才能预报下个月 1 号报告的失业率。

考察模型的残差 ACF：

```
acf(lm3$residuals, lag.max=36, main="")
```

回归残差 ACF 衰减缓慢，怀疑有单位根残余。相关系数都为正数。

```
pacf(lm3$residuals, lag.max=36, main="")
```

这个 PACF 比较符合截尾特征，取 $p = 3$ 可能合适。使用与 mod3 类似的设定：

```
mod4 <- arima(
  da2[["rate"]], xreg=as.matrix(da2[,c("w1m1", "w2m1")]),
  order=c(2,0,2),
  seasonal=list(order=c(1,0,1), period=12)
); mod4
```

```
##
## Call:
## arima(x = da2[["rate"]], order = c(2, 0, 2), seasonal = list(order = c(1, 0,
##     1), period = 12), xreg = as.matrix(da2[, c("w1m1", "w2m1")]))
##
## Coefficients:
##          ar1      ar2      ma1     ma2    sar1     sma1  intercept    w1m1
##       1.9172  -0.9197  -0.9958  0.2532  0.6111  -0.7915     5.6555  0.4265
## s.e.  0.0269   0.0268   0.0563  0.0507  0.1119   0.0883     0.3912  0.2721
##         w2m1
```

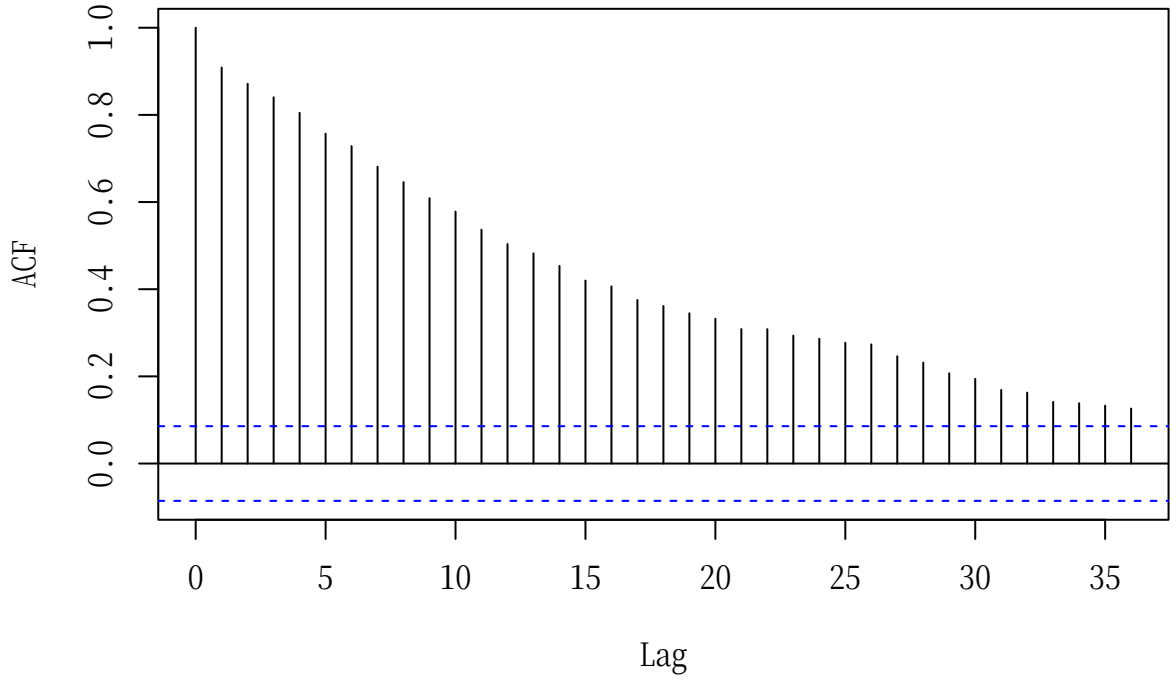


图 15.18: 使用两周申请人数作为解释变量的回归残差 ACF

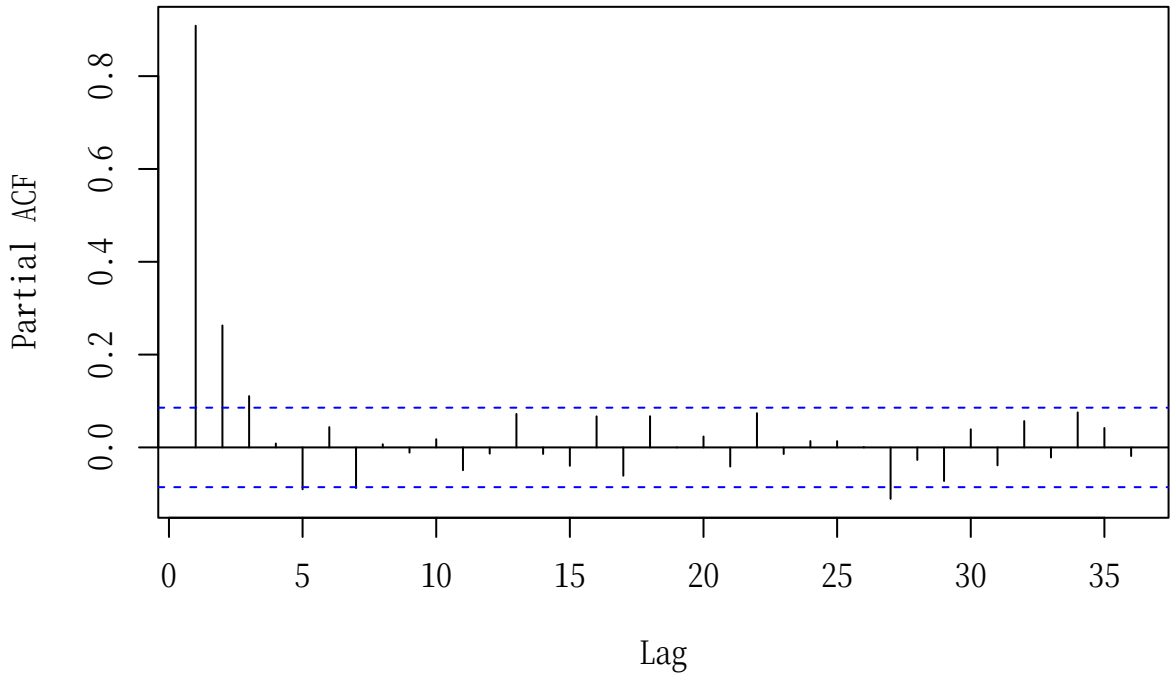


图 15.19: 使用两周申请人数作为解释变量的回归残差 PACF

```
##       0.9693
## s.e.  0.3206
##
## sigma^2 estimated as 0.024:  log likelihood = 230.29,  aic = -440.59
```

模型 AIC 为 −440.59，小于 mod3 的 −435.93。样本内比较 mod4 优于 mod3。

mod4 模型残差诊断：

```
tsdiag(mod4, gof=36)
```

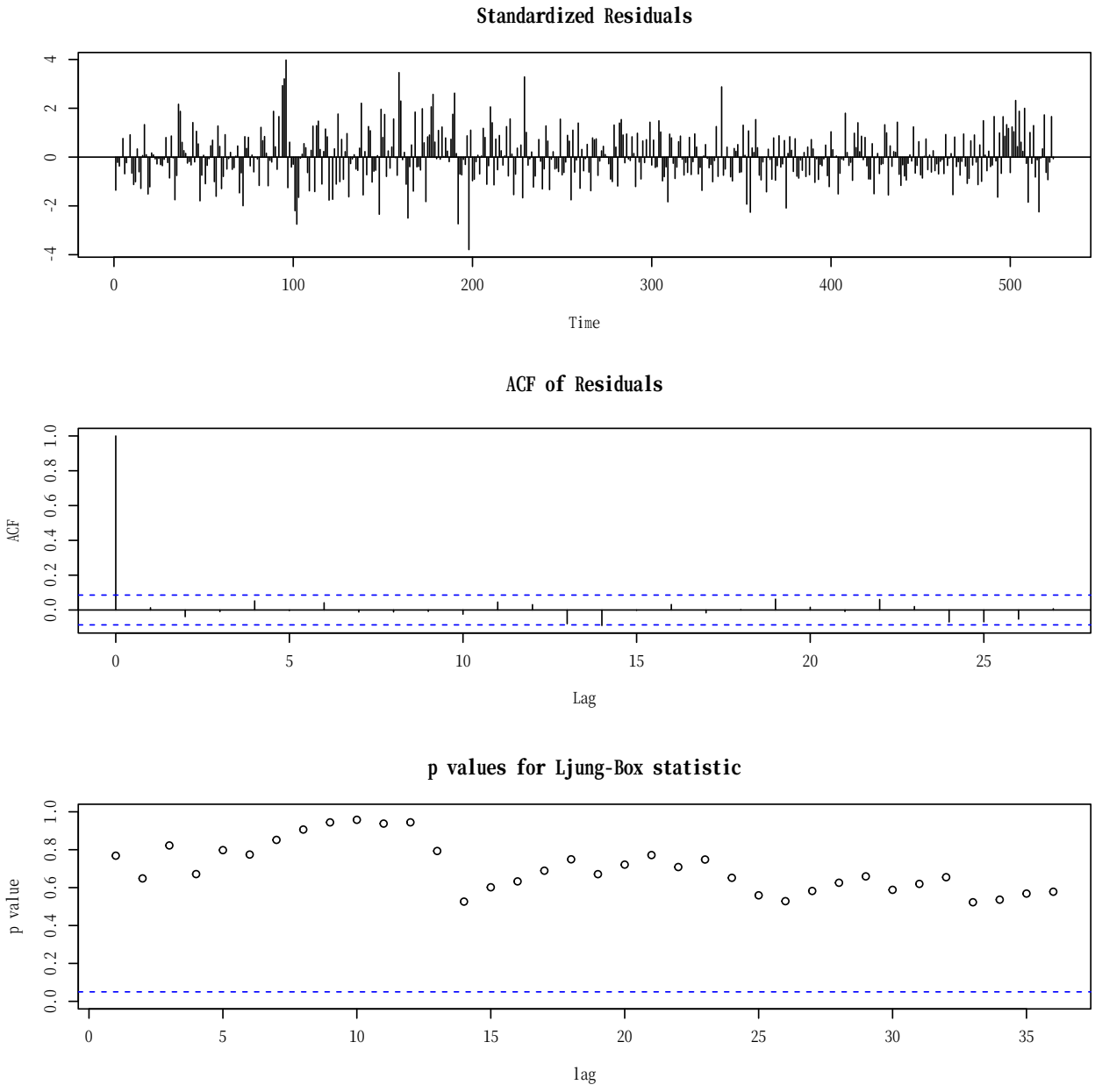


图 15.20: 使用前两周申请人数为解释变量的 ARIMA 模型的残差诊断

残差诊断可以接受。

模型 mod4 公式为：

$$(1-1.92B+0.92B^2)(1-0.61B^{12})(x_t-5.66-0.4265w_{1,t-1}-0.9693w_{2,t-1}) \tag{15.10}$$

$$=(1-1.00B+0.25B^2)(1-0.79B^{12})a_t,\ \mathrm{Var}(a_t)=0.024,\ \mathrm{AIC}=-4440.59 \tag{15.11}$$

仔细比较 mod3 和 mod4，可以发现回归残差的 ARMA 模型的系数基本相同。

15.5 模型再比较

前面的利用上个月首次申请失业救济金人数为解释变量的模型应该更准确地预测月失业率。为了比较，再建立一个不利用解释变量的单变量时间序列模型，时间期间取相同的可比的 1967 年 2 月到 2010 年 9 月。

```
mod5 <- arima(
  da2[["rate"]],
  order=c(1,1,2),
  seasonal=list(order=c(1,0,1), period=12)
); mod5
```

```
##
## Call:
## arima(x = da2[["rate"]], order = c(1, 1, 2), seasonal = list(order = c(1, 0,
##     1), period = 12))
##
## Coefficients:
##          ar1      ma1     ma2    sar1     sma1
##       0.9007  -0.8684  0.1700  0.6250  -0.8303
## s.e.  0.0310   0.0532  0.0467  0.0838   0.0615
##
## sigma^2 estimated as 0.0252:  log likelihood = 219.08,  aic = -426.17
```

模型 mod5 的残差诊断：

```
tsdiag(mod5, gof=36)
```

残差诊断表明模型 mod5 可接受。模型公式：

$$(1-0.90B)(1-B)(1-0.63B^{12})x_t=(1-0.87B+0.17B^2)(1-0.83B^{12})a_t \tag{15.12}$$

$$\mathrm{Var}(a_t)=0.0252,\ \mathrm{AIC}=-426.7 \tag{15.13}$$

用 AIC 作样本内比较：

```
err.tab2 <- data.frame(
  " 模型"=c(" 使用月申请人数", " 使用两周申请人数", " 不使用解释变量"),
  AIC=c(-435.93, -440.59, -426.17))
knitr::kable(err.tab2)
```

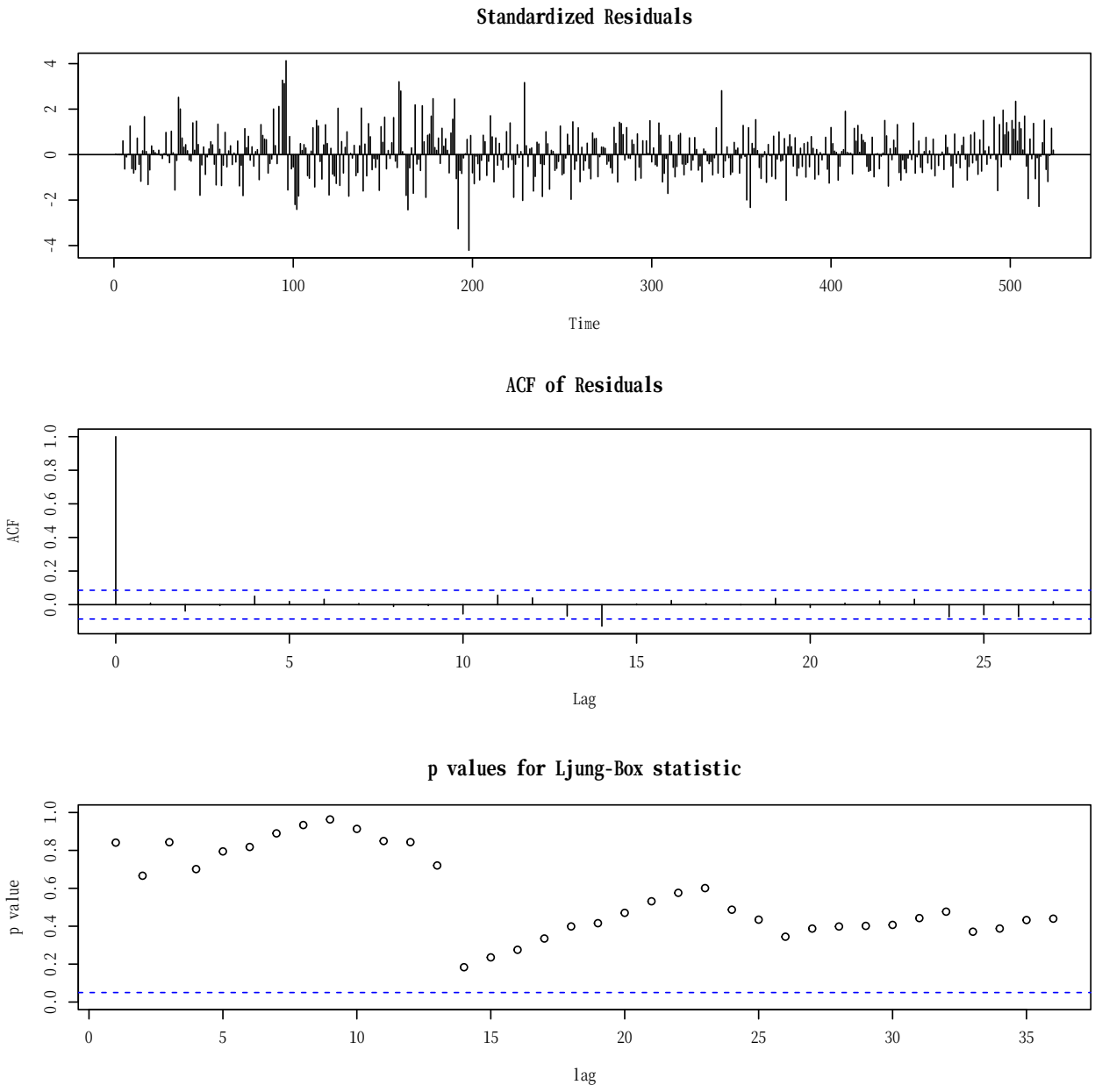


图 15.21: 时间区间缩短的一元时间序列的残差诊断

模型	AIC
使用月申请人数	-435.93
使用两周申请人数	-440.59
不使用解释变量	-426.17

用 AIC 作样本内比较，使用两周申请人数的模型 mod4 最优，不使用解释变量的模型 mod5 最差。

下面进行样本外比较。保留最后 53 个月数据用于预测比较。

使用 mod3 作超前一步预报的函数为：

```
pred3 <- function(da=da2, n.pred=53){
  T <- nrow(da)
  ypred <- rep(NA_real_, T)
  for(h in (T-n.pred):(T-1)){
    mod <- arima(
      da[["rate"]][1:h], xreg=da[["icm1"]][1:h],
      order=c(2,0,2),
      seasonal=list(order=c(1,0,1), period=12)
    )
    ypred[h+1] <- predict(mod, n.ahead=1, se.fit=FALSE,
                          newxreg=cbind(da[["icm1"]][h+1]))
  }

  ypred
}
```

使用 mod4 作超前一步预报的函数为：

```
pred4 <- function(da=da2, n.pred=53){
  T <- nrow(da)
  ypred <- rep(NA_real_, T)
  for(h in (T-n.pred):(T-1)){
    mod <- arima(
      da[["rate"]][1:h], xreg=as.matrix(da2[1:h,c("w1m1", "w2m1")]),
      order=c(2,0,2),
      seasonal=list(order=c(1,0,1), period=12)
    )
    ypred[h+1] <- predict(mod, n.ahead=1, se.fit=FALSE,
                          newxreg=cbind(da[["w1m1"]][h+1], da[["w2m1"]][h+1]))
  }

  ypred
}
```

使用 mod5 作超前一步预报的函数为：

```
pred5 <- function(da=da2, n.pred=53){
  T <- nrow(da)
  ypred <- rep(NA_real_, T)
  for(h in (T-n.pred):(T-1)){
    mod <- arima(
      da[["rate"]][1:h],
      order=c(1,1,2),
      seasonal=list(order=c(1,0,1), period=12)
    )
    ypred[h+1] <- predict(mod, n.ahead=1, se.fit=FALSE)
  }

  ypred
}
```

对一种方法的预测计算预测根均方误差 (RMFSE) 和平均绝对预测误差 (MAFE) 的函数：

样本外比较的计算：

```
pred.tab3 <- data.frame(
  " 模型"=c(" 使用月申请人数", " 使用两周申请人数", " 不使用解释变量"),
  AIC=c(-435.93, -440.59, -426.17),
  RMFSE=rep(0.0, 3),
  MAFE=rep(0.0, 3)
)
funcs <- list(pred3, pred4, pred5)
for(i in 1:3){
  predf <- funcs[[i]]
  ypred <- predf(da=da2, n.pred=53)
  err <- pred.err(y=da2[["rate"]], n.pred=53, ypred=ypred)
  pred.tab3[i, "RMFSE"] <- err["RMFSE"]
  pred.tab3[i, "MAFE"] <- err["MAFE"]
}
```

```
## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1
```

```
## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = da[["icm1"]][1:h], order = c(2, :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = as.matrix(da2[1:h, c("w1m1", :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = as.matrix(da2[1:h, c("w1m1", :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = as.matrix(da2[1:h, c("w1m1", :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = as.matrix(da2[1:h, c("w1m1", :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = as.matrix(da2[1:h, c("w1m1", :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = as.matrix(da2[1:h, c("w1m1", :
```

```
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = as.matrix(da2[1:h, c("w1m1", :
## possible convergence problem: optim gave code = 1

## Warning in arima(da[["rate"]][1:h], xreg = as.matrix(da2[1:h, c("w1m1", :
## possible convergence problem: optim gave code = 1
```

```
knitr::kable(pred.tab3, digits=4)
```

模型	AIC	RMFSE	MAFE
使用月申请人数	-435.93	0.1706	0.1454
使用两周申请人数	-440.59	0.1683	0.1372
不使用解释变量	-426.17	0.1684	0.1370

计算过程中出现多次建模时参数估计的最优化程序不收敛。

样本外比较与样本内比较结果很不同，样本内比较时，使用解释变量的模型明显较优。但是样本外比较时一元时间序列模型不差甚至于更优。这可能和仅超前一步预报有关，单位根模型在超前一步预报时基本上是用前一个数值作为超前一步预报的值，因为月失业率数据变化较慢，所以单位根模型也能得到较好的超前一步预测结果。

这个例子展示了样本内比较与样本外比较的差别。

三个模型有许多相似性：

第一，都有季节项，说明虽然序列都进行过季节调整，但仍残余季节影响；

第二，使用解释变量的模型(15.9)、(15.11) 虽然没有单位根，但是其特征根很接近于有单位根的情形：

```
abs(polyroot(c(1, -coef(mod3)[1:2])))
```

```
## [1] 1.045701 1.045701
```

```
abs(polyroot(c(1, -coef(mod4)[1:2])))
```

```
## [1] 1.042751 1.042751
```

这从一个侧面也解释了三个模型的超前一步预测效果相近的原因。

Part III

资产波动率模型

Chapter 16

资产波动率模型特征

金融数据中最关心的除了资产价格、收益率，就是资产波动率。资产波动率度量某项资产的风险，有多种定义。本章：

- 理解波动率特点；
- 学习 ARCH、GARCH 等波动率模型；
- 学习如何对波动率建模，如何应用波动率模型。

波动率是期权定价和资产分配的关键因素。波动率对计算风险管理中的 VaR（风险值）有重要作用。一些波动率指数已经成为金融工具，如 CBOE(Chicargo Board of Option Exchange) 比值的 VIX 波动率指数已经从 2004-03-26 开始成为期货产品。

波动率的数学模型已经提出多种，本章包括：

- (R. Engle, 1982) 的 ARCH 模型（自回归条件异方差模型）
- (Bollerslev, 1986) 的 GARCH 模型（广义自回归条件异方差模型）
- (Nelson, 1991) 的 EGARCH 模型（指数 GARCH 模型）
- Glosten 等 (1993) 和 Zakoian(1994) 提出的 TGARCH（门限 GARCH 模型）
- Engle 和 Ng(1993) 以及 Duan(1995) 提出的 NGARCH 模型（非对称 GARCH）
- Melino 和 Turnbull(1990)、Taylor(1994)、Harvey 等 (1994) 以及 Jacquier 等 (1994) 分别提出的随机波动率 (Stochastic Volatility, SV) 模型

16.1 波动率的特征

这是原书 (Tsay, 2013)§4.1 内容。

波动率 (volatility) 指的是资产价格的波动强弱程度，类似于概率论中随机变量标准差的概念。波动率不能直接观测，可以从资产收益率中看出波动率的一些特征。

- 存在波动率聚集 (volatility clustering)
- 波动率随时间连续变化，一般不出现波动率的跳跃式变动
- 波动率一般在固定的范围内变化，意味着动态的波动率是平稳的
- 在资产价格大幅上扬和大幅下跌两种情形下，波动率的反映不同，大幅下跌时波动率一般也更大，这种现象称为杠杆效应（leverage effect）

这些性质对波动率模型的提出、改进有重要意义，许多新的波动率模型都是诊断原有模型不能反映上面的某型特征而提出的。例如，EGARCH 模型和 TGARCH 模型可以反映出波动率在价格上扬和下跌时的不对称性。

不同的波动率计算方法使用不同的数据源。例如，对 IBM 股票有如下三种数据源：

- 每个交易日的日收益率；
- 伴随 IBM 股票的期权数据
- 盘中交易和报价的分笔数据；

分别可以计算如下三种不同的波动率：

- 作为日收益率的条件标准差（或条件方差），建模计算。本章的模型是针对这种波动率定义。
- 隐含波动率：根据期权的理论公式如 BS 公式，从股票价格和期权价格数据反解出模型中的波动率，这样的得到的波动率称为隐含波动率。隐含波动率倾向于比用日收益率建模得到的波动率数值要大。CBOE 的 VIX 指数就是隐含波动率。
- 实际波动率：利用一天内所有的收益率数据，如每 5 分钟的收益率，估计一天收益率的方差（波动率平方）。已实现波动率（realized volatility, RV）是这样的波动率。

类似于利率，度量波动率的时间区间一般也取为一年，波动率一般是年化波动率。如果有了日波动率，可以将其乘以 $\sqrt{252}$ 转换成年化的波动率。

16.2 波动率模型的结构

这是原书 (Tsay, 2013)§4.2 内容。

波动率是收益率的条件标准差。设 r_t 是某种资产在时刻 t 的基于某时间单位（如天、月、年）的对数收益率，一般认为 $\{r_t\}$ 序列是前后不相关的，或者是低阶相关的，表现为其 ACF 基本都在零上下的界限内波动，或者仅前一两个略超出界限。但是，$\{r_t\}$ 序列一般也不是前后独立的。

以 Intel 公司股票从 1973-1 到 2009-12 的月度对数收益率为例，有 444 个观测值。读入数据：

```
d.intel <- read_table(
  "m-intcsp7309.txt",
  col_types=cols(
    .default=col_double(),
    date=col_date("%Y%m%d")
  ))
```

前 20 行数据：

```
knitr::kable(d.intel[1:20,])
```

date	intc	sp
1973-01-31	0.010050	-0.017111
1973-02-28	-0.139303	-0.037490
1973-03-30	0.069364	-0.001433
1973-04-30	0.086486	-0.040800
1973-05-31	-0.104478	-0.018884
1973-06-29	0.133333	-0.006575
1973-07-31	0.625000	0.037982
1973-08-31	0.117647	-0.036685
1973-09-28	0.234818	0.040096
1973-10-31	0.144262	-0.001291
1973-11-30	-0.240688	-0.113861
1973-12-31	0.188679	0.016569
1974-01-31	0.139683	-0.010046
1974-02-28	0.155989	-0.003624
1974-03-29	-0.163855	-0.023280
1974-04-30	0.162824	-0.039051
1974-05-31	0.148699	-0.033551
1974-06-28	-0.148867	-0.014665
1974-07-31	-0.448669	-0.077791
1974-08-30	-0.151724	-0.090279

将其转换成对数收益率后保存成 xts 类型，并将日期类型改为 `yearmon`，生成一个 ts 类型副本:

```
xts.intel <- xts(
  log(1 + d.intel[["intc"]]), d.intel[["date"]]
)
tclass(xts.intel) <- "yearmon"
ts.intel <- ts(c(coredata(xts.intel)), start=c(1973,1), frequency=12)
```

作 Intel 对数收益率的时间序列图:

```
plot(ts.intel, ylab="log return",
     main="Intel Stock Price Monthly Log Return")
```

作对数收益率序列的 ACF 图:

```
forecast::Acf(ts.intel, main="ACF of log return")
```

从 $\{r_t\}$ 的 ACF 图来看，除了滞后 7 和滞后 14 两个点略微出界，其它的自相关都在零水平线的界限内，可以认为基本是白噪声（这里的白噪声特指零均值、不相关的弱平稳时间序列）。

作 Ljung-Box 白噪声检验:

```
Box.test(ts.intel, lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
```

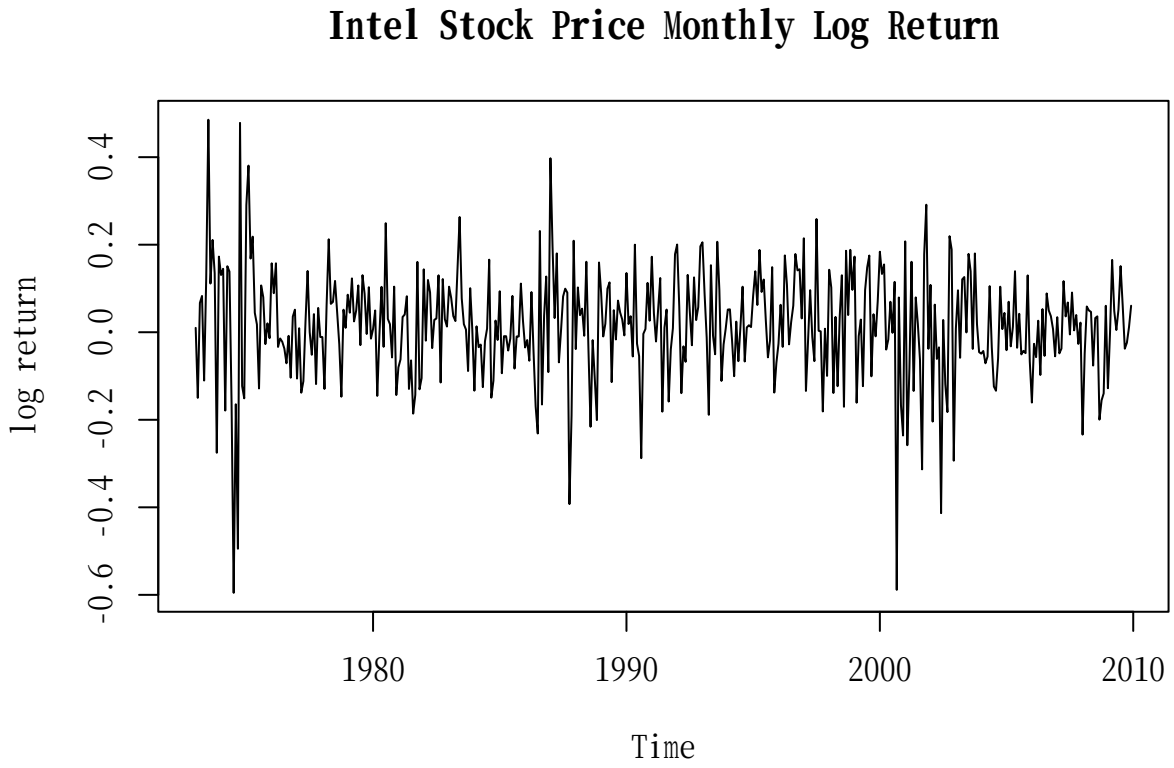


图 16.1: Intel 对数收益率时间序列

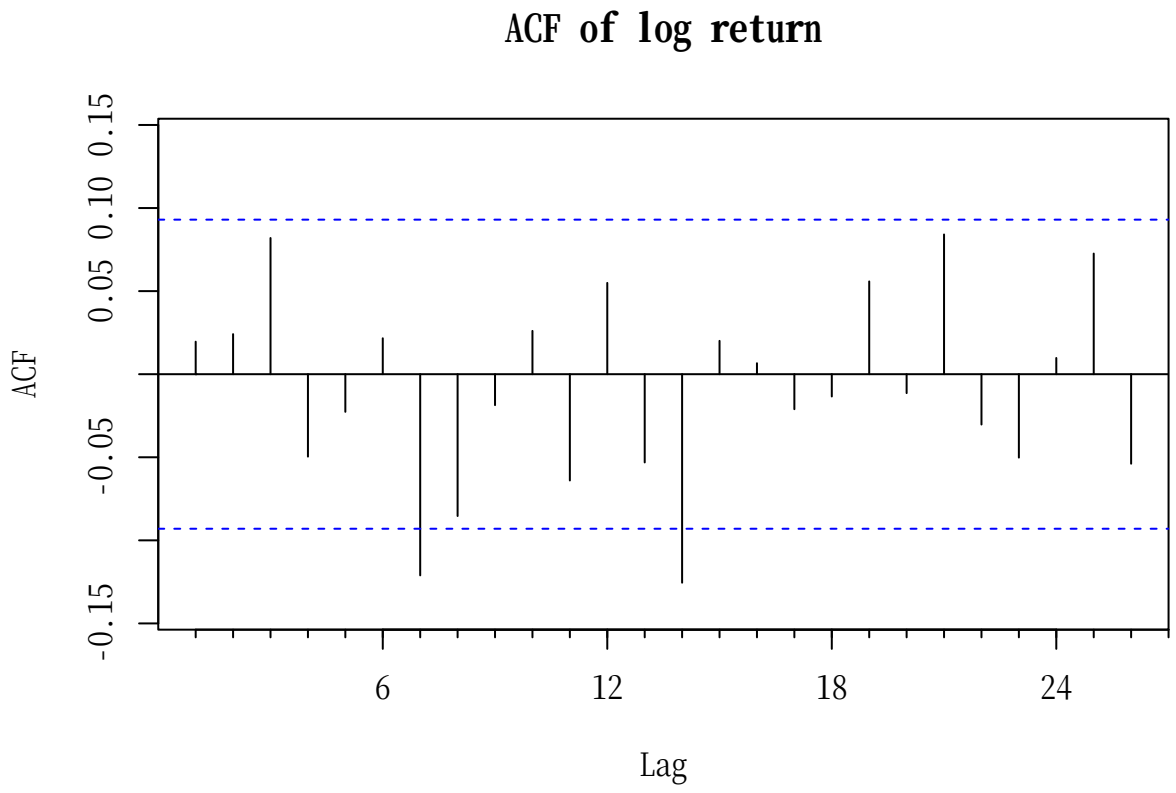


图 16.2: Intel 对数收益率的 ACF 估计

```
##
## data:  ts.intel
## X-squared = 18.676, df = 12, p-value = 0.09665
```

在 0.05 水平下通过了白噪声检验。

可以检验 $\{r_t\}$ 序列的均值是否等于零：

```
t.test(as.vector(ts.intel))
```

```
##
##  One Sample t-test
##
## data:  as.vector(ts.intel)
## t = 2.3788, df = 443, p-value = 0.01779
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.00249032 0.02616428
## sample estimates:
## mean of x
## 0.0143273
```

p 值小于 0.05，可认为平均收益率显著大于 0。

考虑 $\{|r_t|\}$ 序列。作其 ACF：

```
forecast::Acf(abs(ts.intel),
  main="Intel 对数收益率绝对值的 ACF 估计")
```

可以看出，多个自相关值超出了界限。作 Ljung-Box 白噪声检验：

```
Box.test(abs(ts.intel), lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  abs(ts.intel)
## X-squared = 124.91, df = 12, p-value < 2.2e-16
```

结果显著地拒绝了白噪声零假设。这说明 $\{r_t\}$ 表现为平稳、不相关，但并不前后独立。

作 $\{r_t^2\}$ 的 ACF:

```
forecast::Acf(ts.intel^2, main="Intel 对数收益率平方的 ACF 估计")
```

对 $\{r_t^2\}$ 序列作 Ljung-Box 白噪声检验：

```
Box.test(ts.intel^2, lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
##
```

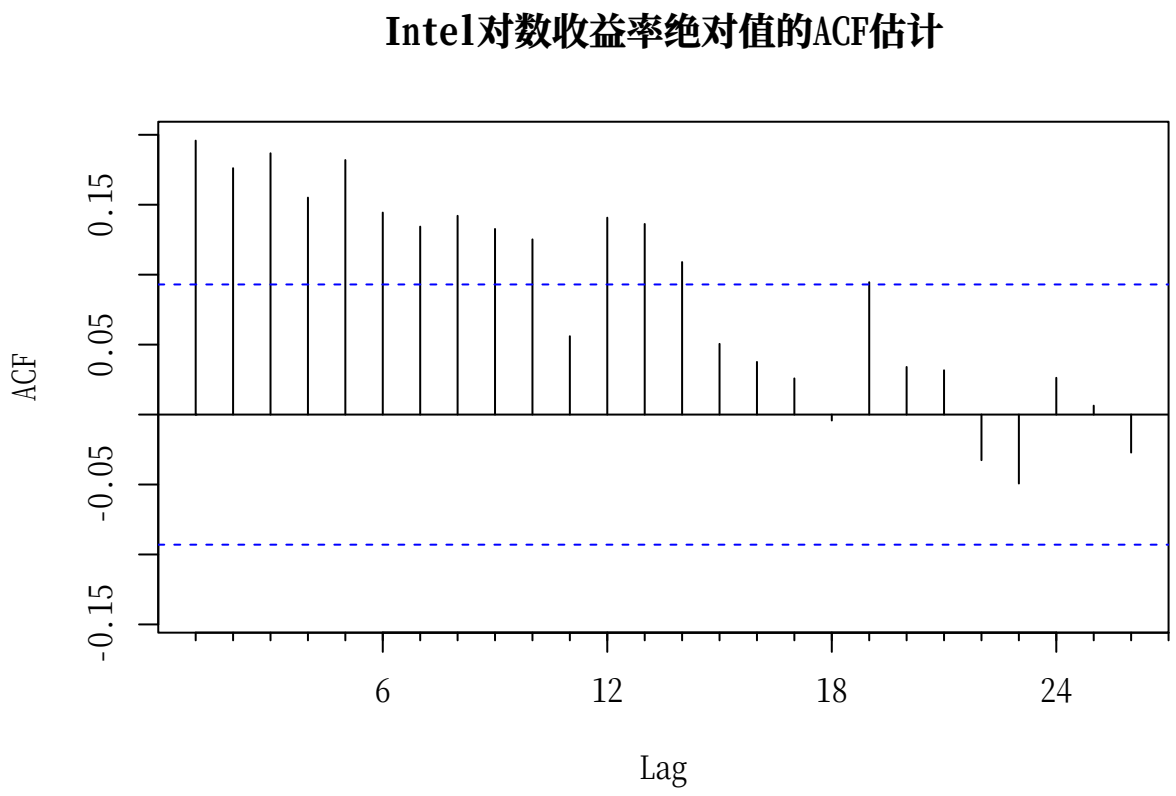


图 16.3: Intel 对数收益率绝对值的 ACF 估计

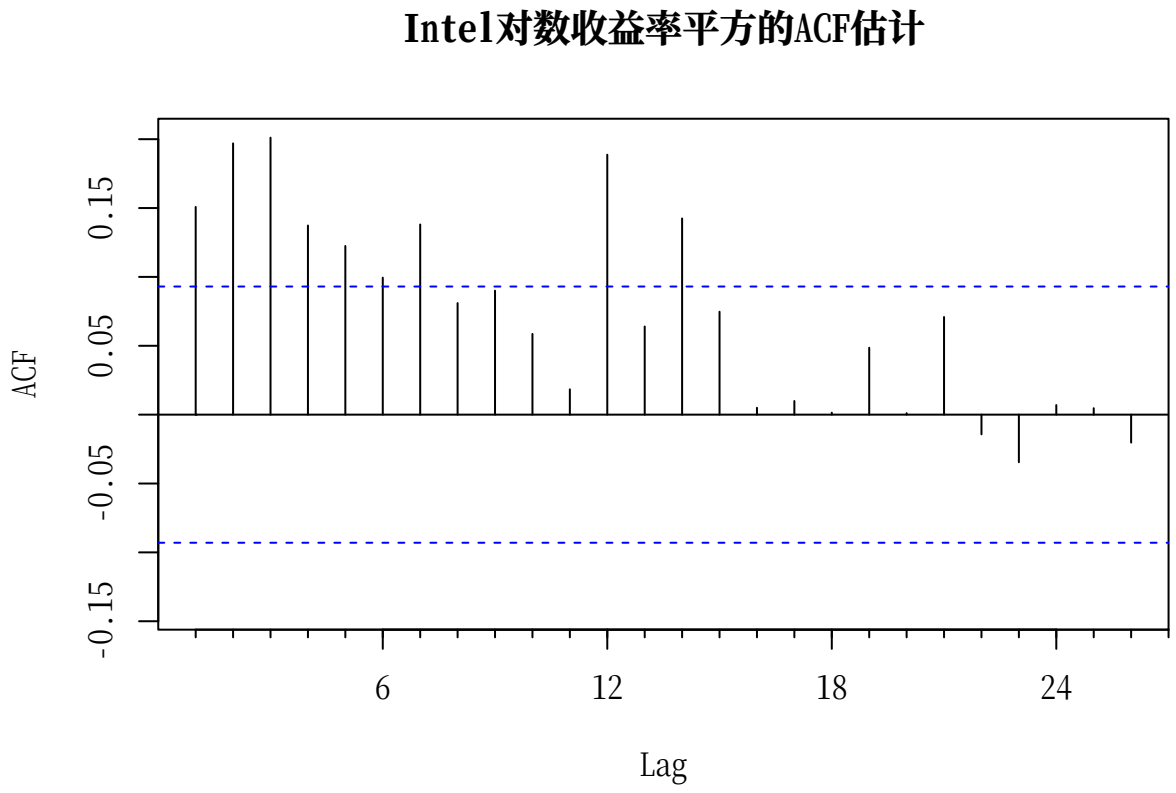


图 16.4: Intel 对数收益率平方的 ACF 估计

```
## data:  ts.intel^2
## X-squared = 98.783, df = 12, p-value = 9.992e-16
```

$\{r_t^2\}$ 的 ACF 估计和白噪声检验都说明它存在相关性，这说明 $\{r_t\}$ 不是独立序列。

一元波动率模型就是试图刻画收益率这种本身不相关或低阶自相关，但是不独立的性质。用 F_{t-1} 表示截止到 $t-1$ 时刻的收益率信息，尤其是包括这些收益率的线性组合，考察 r_t 在 F_{t-1} 条件下的条件均值和条件方差：

$$\mu_t = E(r_t|F_{t-1}), \quad \sigma_t^2 = \mathrm{Var}(r_t|F_{t-1}) = E[(r_t-\mu_t)^2|F_{t-1}] \tag{16.1}$$

通过分析实例的经验可知，(16.1)中 $\{r_t\}$ 通常比较简单，如平稳 ARMA(p,q) 序列。例如，对 Intel 股票价格对数收益率序列 $\{r_t\}$，可设 $r_t = \mu_t + a_t$, 其中 $\mu_t = \mu$ 为常数，a_t 为不相关的白噪声列。

对一般的对数收益率 $\{r_t\}$，设其服从 ARMA(p,q) 模型：

$$r_t = \phi_0 + \sum_{j=1}^{p}\phi_j r_{t-j} + a_t + \sum_{j=1}^{q}\theta_j a_{t-j}$$

其中 $\{a_t\}$ 为不相关的白噪声列（注意第6章的 ARMA 定义中要求独立同分布，这里放宽要求）。于是

$$\mu_t = E(r_t|F_{t-1}) = \phi_0 + \sum_{j=1}^{p}\phi_j r_{t-j} + \sum_{j=1}^{q}\theta_j a_{t-j} = r_t - a_t \tag{16.2}$$

这里我们对白噪声列假定 $a_t = r_t - E(r_t|F_{t-1})$，称这样的白噪声列 $\{a_t\}$ 为平稳列 $\{r_t\}$ 的**新息列** (innovation) 或**扰动列**。r_t 可分解为

$$r_t = \mu_t + a_t.$$

如果可以获得其他的解释变量（外生变量），可以建立模型 $r_t = \mu_t + a_t$，其中

$$\mu_t = \phi_0 + \sum_{i=1}^{k}\beta_i x_{i,t-1} + \sum_{j=1}^{p}\phi_j y_{t-j} + \sum_{j=1}^{q}\theta_j a_{t-j} \tag{16.3}$$

其中 $x_{i,t-1}$ 是第 i 个解释变量在 $t-1$ 时刻的值，y_{t-j} 是剔除解释变量影响后的 r_{t-j} 的值。(16.3)是第10章模型的一个应用。

μ_t 服从的 ARMA(p,q) 的阶与数据的采样频率有关，股票指数的日频数据往往有较小的前后相关性，月度数据则可能没有任何显著的前后相关。

模型(16.3)中的自变量比较灵活，例如，可以取日期星期一哑变量为自变量，考察所谓的周末效应。在资产定价模型 (Capital Asset Pricing Model, CAPM) 中，r_t 的的方程可以写成

$$r_t = \phi_0 + \beta r_{m,t} + a_t$$

其中 $r_{m,t}$ 是市场收益率，一般用综合指数收益率代替。

综合(16.2)和(16.3)，都有

$$\sigma_t^2 = \mathrm{Var}(r_t|F_{t-1}) = \mathrm{Var}(a_t|F_{t-1}) = E(a_t^2|F_{t-1}). \tag{16.4}$$

这里的 σ_t 就是波动率，是收益率的条件标准差。如果假设模型中的白噪声 $\{a_t\}$ 是独立序列，则 $\sigma_t^2 \equiv \sigma^2$，波动率就没有建模的可能。这里假定 $\{a_t\}$ 是零均值不相关平稳列，满足 $E(a_t|F_{t-1}) = 0$，但不是独立序列。

本章的问题就是对 σ_t^2 建模，这种模型叫做条件异方差模型。条件异方差模型分为两类：

- 用确定函数来刻画 σ_t^2 的变化，ARCH 和 GARCH 模型属于这一类；
- 用随机方程描述 σ_t^2 的变化，随机波动率 (SV) 模型属于这一类。

(16.2)和(16.3)中的 μ_t 的模型称为 r_t 的**均值方程**，σ_t^2 的模型称为 r_t 的**波动率方程**。条件异方差模型就是在原来对 r_t 的均值 μ_t 建模的基础上，再增加一个描述资产收益率的条件方差随时间变化的模型。这可以更精确刻画给定 $r_1, r_2, \dots, r_t$ 后 r_{t+1} 所服从的条件分布。

16.3 波动率模型的建立

这是原书 (Tsay, 2013)§4.3 内容。

对资产收益率序列建立波动率模型需要如下四个步骤：

1. 通过检验序列的自相关性建立均值的方程，必要时还可以引入适当的解释变量；
2. 对均值方程的残差作白噪声检验，通过后，对残差检验 ARCH 效应；
3. 如果 ARCH 效应检验结果显著，则指定一个波动率模型，对均值方程和波动率方程进行联合估计；
4. 对得到的模型进行验证，需要时做改进。

关于均值方程，资产收益率一般没有自相关（注意，这并不是独立）或者仅有弱的自相关。如果样本均值显著不等于零，需要从数据中减去样本均值，这称为**中心化**。对某些日收益率或更高频的序列可能需要建立简单的 AR 模型。某些情况下可以加入额外的解释变量或者与日期有关的解释变量，比如反映周末的哑变量，反映一月份的哑变量，等等。例如，在 Intel 股票月对数收益率的均值方程建模时，其均值方程为一个常数。

16.4 ARCH 效应的检验

这是原书 (Tsay, 2013)§4.4 内容。

为了检验 ARCH 效应，先建立均值模型，拟合 μ_t，计算残差 $a_t = r_t - \mu_t$。用残差序列的平方 $\{a_t^2\}$ 作 ARCH 效应检验。

有两种检验方法。一种是对 $\{a_t^2\}$ 作 Ljung-Box 白噪声检验，检验不显著时没有 ARCH 效应，检验显著时有 ARCH 效应。

另一种检验方法是 (R. Engle, 1982) 提出的。考虑如下的最小二乘问题：

$$a_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2 + e_t,\ t = m+1, \dots, T$$

其中 T 为样本量，m 是适当的 AR 阶数，e_t 为回归残差。零假设为

$$H_0:\ \alpha_1 = \cdots = \alpha_m = 0$$

拒绝 H_0 时有 ARCH 效应。这称为 Engle 的拉格朗日乘子法检验。

用普通最小二乘方法估计上述回归问题并计算残差 $\hat{e}_t$。令

$$\mathrm{SSR}_0 = \sum_{t=m+1}^{T} (a_t^2 - \bar{\omega})^2$$

其中 $\bar{\omega} = \frac{1}{T}\sum_{t=1}^{T} a_t^2$ 是 $\{a_t^2\}$ 的样本均值。令

$$\mathrm{SSR}_1 = \sum_{t=m+1}^{T} \hat{e}_t^2$$

令

$$F = \frac{(\mathrm{SSR}_0 - \mathrm{SSR}_1)/m}{\mathrm{SSR}_1/(T-2m-1)}$$

则在 H_0 成立时 F 近似服从 $F(m, T-2m-1)$。当 T 很大时 F 的分母渐近一个常数，mF 近似服从 $\chi^2(m)$，用 $\chi^2(m)$ 分布计算 F 的右侧 p 值进行检验。

执行 Engle 的 ARCH 效应检验的函数定义如下：

```
"archTest" <- function(x, m=10){
  # Perform Lagrange Multiplier Test for ARCH effect of a time series
  # x: time series, residual of mean equation
  # m: selected AR order

  y <- (x - mean(x))^2
  T <- length(x)
  atsq <- y[(m+1):T]
  xmat <- matrix(0, T-m, m)
  for (j in 1:m){
    xmat[,j] <- y[(m+1-j):(T-j)]
  }
  lmres <- lm(atsq ~ xmat)
  summary(lmres)
}
```

16.4.1 Intel 公司股票月对数收益率 ARCH 效应的检验

下面对 Intel 公司股票月对数收益率序列检验 ARCH 效应。因为 r_t 的均值方程仅有常数项，令 $a_t = r_t - \bar{r}$。先对 $\{a_t^2\}$ 作 Ljung-Box 白噪声检验：

```
Box.test((ts.intel - mean(ts.intel))^2,
         lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  (ts.intel - mean(ts.intel))^2
## X-squared = 92.939, df = 12, p-value = 1.332e-14
```

结果高度显著，说明有 ARCH 效应。

用 Engle 的拉格朗日乘子法检验 ARCH 效应：

```
archTest(ts.intel - mean(ts.intel), m=12)
```

```
##
## Call:
## lm(formula = atsq ~ xmat)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.07440 -0.01153 -0.00658  0.00395  0.35255
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.005977   0.002249   2.658 0.008162 **
## xmat1        0.093817   0.048147   1.949 0.052013 .
## xmat2        0.153085   0.048102   3.183 0.001569 **
## xmat3        0.146087   0.048614   3.005 0.002815 **
## xmat4        0.023539   0.049126   0.479 0.632075
## xmat5        0.007347   0.049107   0.150 0.881139
## xmat6        0.010342   0.047027   0.220 0.826050
## xmat7        0.057183   0.047027   1.216 0.224681
## xmat8        0.014320   0.047079   0.304 0.761149
## xmat9        0.007157   0.046968   0.152 0.878965
## xmat10      -0.019742   0.046566  -0.424 0.671810
## xmat11      -0.057537   0.046041  -1.250 0.212116
## xmat12       0.161945   0.045965   3.523 0.000473 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03365 on 419 degrees of freedom
## Multiple R-squared:  0.1248, Adjusted R-squared:  0.0997
## F-statistic: 4.978 on 12 and 419 DF,  p-value: 9.742e-08
```

检验的 p 值为 9.7×10^{-8}，高度显著，说明有 ARCH 效应。

注意程序中 `Box.test()` 输入的是 a_t^2，`archTest()` 输入的是 a_t，没有平方。

16.4.2 美元对欧元汇率日对数收益率 ARCH 效应的检验

其它金融时间序列也存在 ARCH 效应。考虑美元对欧元汇率日对数收益率，从 1999-01-04 到 2010-08-20。

读入数据:

```
d.useu <- read_table(
  "d-useu9910.txt",
  col_types=cols(.default=col_double())
)
str(d.useu)
```

```
## spc_tbl_ [2,930 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ year: num [1:2930] 1999 1999 1999 1999 1999 ...
##  $ mon : num [1:2930] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day : num [1:2930] 4 5 6 7 8 11 12 13 14 15 ...
##  $ rate: num [1:2930] 1.18 1.18 1.16 1.17 1.16 ...
##  - attr(*, "spec")=
```

```
##   .. cols(
##   ..   .default = col_double(),
##   ..   year = col_double(),
##   ..   mon = col_double(),
##   ..   day = col_double(),
##   ..   rate = col_double()
##   .. )
```

共 2930 个观测。保存成 xts，计算对数收益率：

```
xts.useu <- with(d.useu, xts(rate, make_date(year, mon, day)))
xts.useu.lnrtn <- diff(log(xts.useu))[-1,]
eu <- c(coredata(xts.useu.lnrtn))
```

对数收益率的时间序列图：

```
plot(xts.useu.lnrtn, main="",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years",
     format.labels="%Y")
```

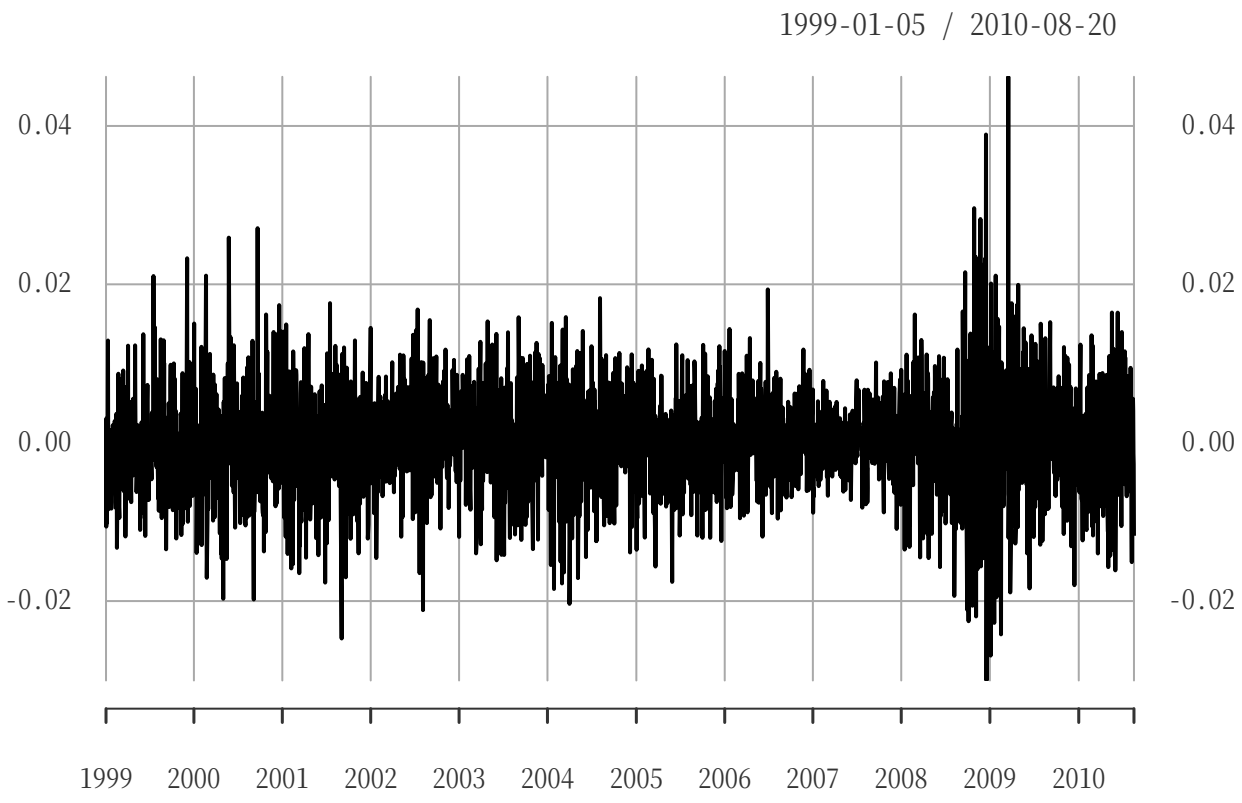


图 16.5: 美元对欧元汇率日对数收益率

此序列有波动率聚集现象，在 2008 年下半年和 2009 年上半年有较高的波动率。

作 ACF 图：

```
forecast::Acf(eu, main="")
```

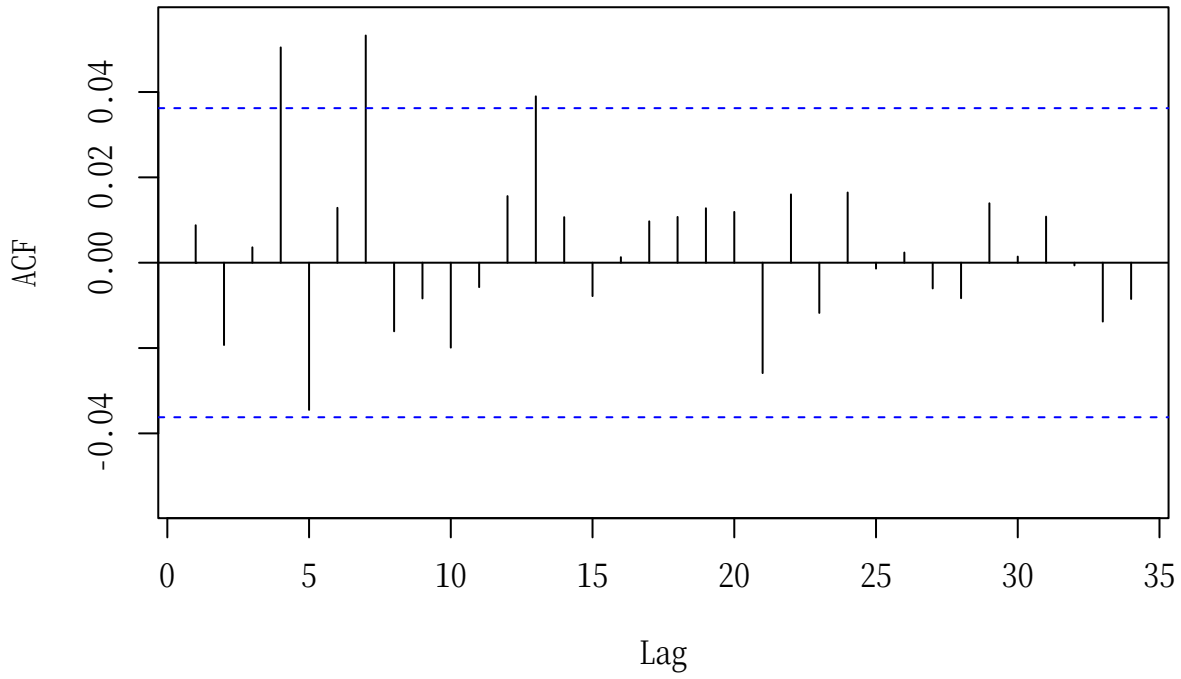


图 16.6: 美元对欧元汇率日对数收益率的 ACF

基本符合白噪声的 ACF 表现。

作 Ljung-Box 白噪声检验：

```
Box.test(eu, lag=20, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  eu
## X-squared = 30.585, df = 20, p-value = 0.06091
```

在 0.05 水平下可以承认对数收益率为白噪声列。

检验对数收益率均值是否等于零：

```
t.test(eu)
```

```
##
##  One Sample t-test
##
```

```
## data:  eu
## t = 0.20217, df = 2928, p-value = 0.8398
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.0002122342  0.0002610303
## sample estimates:
##    mean of x
## 2.439805e-05
```

结果说明对数收益率均值为零。所以 r_t 的均值方程为 $r_t = a_t$（$\mu_t = 0$）。

对 r_t^2 作 ACF 图：

```
forecast::Acf(eu^2, main="")
```

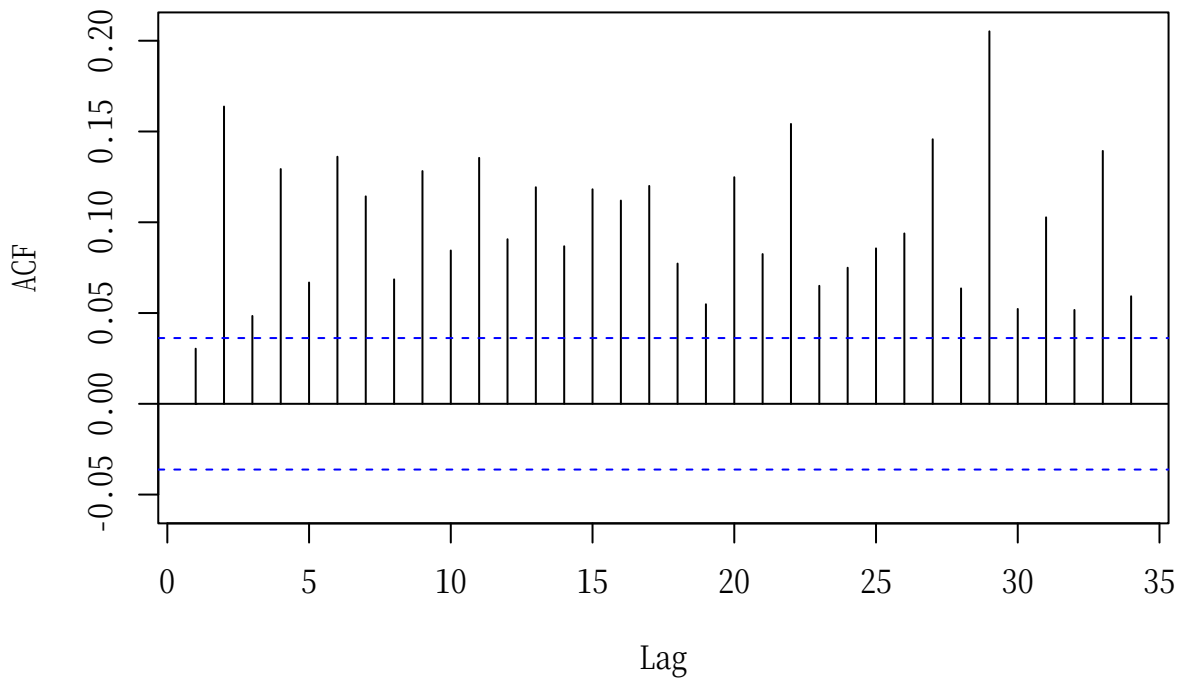


图 16.7: 美元对欧元汇率日对数收益率平方的 ACF

此 ACF 明显非白噪声，且不像是低阶 MA。

对 r_t^2 作 PACF 图：

```
pacf(eu^2, main="")
```

此 PACF 表现不像是低阶的 AR。

对 r_t^2 作 Ljung-Box 白噪声检验：

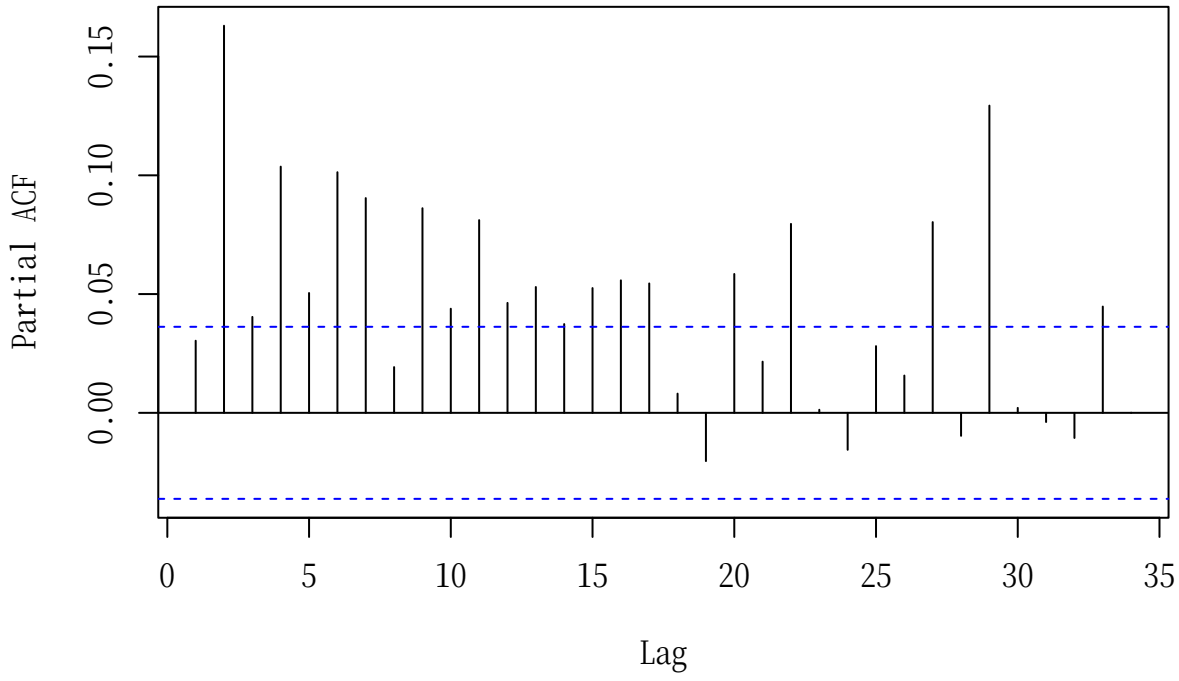


图 16.8: 美元对欧元汇率日对数收益率平方的 PACF

```
Box.test(eu^2, lag=20, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  eu^2
## X-squared = 661.45, df = 20, p-value < 2.2e-16
```

检验结果拒绝白噪声假设，说明汇率的日对数收益率有 ARCH 效应。

对 r_t^2 序列作 Engle 的拉格朗日乘子法检验：

```
archTest(eu, m=20)
```

```
##
## Call:
## lm(formula = atsq ~ xmat)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -3.229e-04 -3.369e-05 -1.949e-05  9.360e-06  2.100e-03
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.281e-05  2.535e-06   5.054 4.60e-07 ***
## xmat1       -3.022e-02  1.858e-02  -1.626 0.103966
## xmat2        9.441e-02  1.859e-02   5.080 4.02e-07 ***
## xmat3       -1.226e-02  1.867e-02  -0.657 0.511513
## xmat4        5.309e-02  1.864e-02   2.848 0.004428 **
## xmat5        2.668e-03  1.864e-02   0.143 0.886202
## xmat6        7.200e-02  1.862e-02   3.868 0.000112 ***
## xmat7        5.625e-02  1.866e-02   3.015 0.002594 **
## xmat8       -1.599e-03  1.869e-02  -0.086 0.931828
## xmat9        6.060e-02  1.867e-02   3.245 0.001188 **
## xmat10       2.794e-02  1.867e-02   1.497 0.134592
## xmat11       6.413e-02  1.867e-02   3.435 0.000602 ***
## xmat12       4.020e-02  1.867e-02   2.153 0.031439 *
## xmat13       4.375e-02  1.869e-02   2.341 0.019299 *
## xmat14       2.900e-02  1.867e-02   1.553 0.120458
## xmat15       4.927e-02  1.863e-02   2.645 0.008222 **
## xmat16       5.349e-02  1.865e-02   2.868 0.004163 **
## xmat17       5.702e-02  1.865e-02   3.058 0.002251 **
## xmat18       1.873e-03  1.868e-02   0.100 0.920136
## xmat19      -1.836e-02  1.859e-02  -0.987 0.323583
## xmat20       5.844e-02  1.859e-02   3.144 0.001683 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.483e-05 on 2888 degrees of freedom
## Multiple R-squared:  0.09265,    Adjusted R-squared:  0.08636
## F-statistic: 14.74 on 20 and 2888 DF,  p-value: < 2.2e-16
```

结果说明有显著的 ARCH 效应。

Chapter 17

ARCH 模型

这是原书 (Tsay, 2013)§4.5 内容。

17.1 ARCH 模型公式

(R. Engle, 1982) 提出了 ARCH 模型（自回归条件异方差模型），这是对将波动率定义为条件标准差，第一次提出的波动率的理论模型。基本思想是：

1. 资产收益率的扰动序列 $a_t = r_t - E(r_t|F_{t-1})$ 是前后不相关的，但是前后不独立。
2. a_t 的不独立性，描述为 $\mathrm{Var}(r_t|F_{t-1}) = \mathrm{Var}(a_t|F_{t-1})$ 可以用 a_t^2 的滞后值的线性组合表示。

其中 $F_t = \sigma(\{r_t, r_{t-1}, ...\})$。

具体地，ARCH(m) 模型为

$$\begin{aligned} a_t =& \sigma_t \varepsilon_t, \\ \sigma_t^2 =& \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2. \end{aligned} \tag{17.1}$$

其中 $\{\varepsilon_t\}$ 是零均值单位方差的独立同分布白噪声，$\alpha_0 > 0$，$\alpha_j \geq 0$, $j = 1, 2, ..., m$，另外 $\{\alpha_j\}$ 还需要满足一些条件使得 $\mathrm{Var}(a_t)$ 有限，类似于 AR(p) 序列的平稳性的特征根条件。

在(17.1)的波动率方程的右侧，仅出现了截止到 $t-1$ 时刻的 $a_{t-1}, ..., a_{t-m}$ 的确定性函数而没有新增的随机扰动，所以称 ARCH 模型为确定性的波动率模型，这意味着 σ_t^2 关于 F_{t-1} 可测，即在 $t-1$ 时刻可以确定条件方差 σ_t^2 的值。

ε_t 的分布常取为标准正态分布，标准化的 t 分布，广义误差分布 (Generalized Error Distribution)，有些情况下还取为有偏的分布。

如果 $\varepsilon_t \sim \mathrm{N}(0,1)$，记 $\mu_t = E(r_t|F_{t-1})$，考虑 $p = 1$ 情形，则

$$r_t|F_{t-1} \sim \mathrm{N}(\mu_t, \sigma_t^2) = \mathrm{N}(\mu_t, \alpha_0 + \alpha_1 a_{t-1}^2).$$

所以在 ARCH 模型中 ε_t 的分布称为 “条件分布”，即在 F_{t-1} 条件下 r_t 的条件分布的类型。

因为 $a_t = r_t - E(r_t|F_{t-1})$ 所以 $Ea_t = 0$，$E(a_t|F_{t-1}) = 0$。由(17.1)中 $\{\varepsilon_t\}$ 独立可知 ε_t 与 F_{t-1} 独立，从而与 σ_t^2

独立，于是

$$\begin{aligned}\mathrm{Var}(r_t|F_{t-1}) =&E[(r_t - E(r_t|F_{t-1}))^2|F_{t-1}] = E(a_t^2|F_{t-1}) &(17.2)\\ =&E(\varepsilon_t^2\sigma_t^2|F_{t-1}) = \sigma_t^2 E(\varepsilon_t^2|F_{t-1}) = \sigma_t^2 &(17.3)\\ =&\alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2. &(17.4)\end{aligned}$$

即(17.1)给出了 r_t 的条件方差方程。

因为系数 α_j 都是非负数，所以历史值 a_{t-j}^2 较大意味着 a_t 的条件方差较大，于是在 ARCH 模型框架下，大的扰动后面倾向于会出现较大的扰动。这里“倾向于”不是指一定会出现大的扰动，因为 a_{t-j}^2 较大使得条件方差 σ_t^2 较大，而方差大只能说出现较大的 a_t 的概率变大，而不是一定会出现大的扰动 a_t。这种现象能够解释资产收益率的波动率聚集现象。

注意：有些作者用 $h_t = \sigma_t^2$ 作为条件方差的记号，这时扰动 $a_t = \varepsilon_t\sqrt{h_t}$。

17.2 ARCH 模型的性质

以最简单的 ARCH(1) 为例讨论模型的性质。模型为

$$a_t = \sigma_t\varepsilon_t,\ \sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2. \tag{17.5}$$

其中 $\alpha_0 > 0, 0 < \alpha_1 < 1$。$\alpha_1 > 0$ 是因为如果等于零就不能算 ARCH(1)，$\alpha_1 < 1$ 是为了 a_t 方差有限。

17.2.1 新息性质

令 $F_t = \sigma(\{r_t, r_{t-1}, \ldots\})$，按 a_t 定义，

$$a_t = r_t - E(r_t|F_{t-1}),$$

$\{a_t\}$ 称为 $\{r_t\}$ 的**新息序列** (innovation series)。

这样的 a_t 满足

$$\begin{aligned}E(a_t|F_{t-1}) =&E[r_t - E(r_t|F_{t-1})|F_{t-1}]\\ =&E(r_t|F_{t-1}) - E(r_t|F_{t-1}) = 0,\end{aligned}$$

称这样的序列为（关于 $\{F_t\}$ 的）鞅差序列。鞅差序列必为不相关列。

易见

$$E(a_t) = E[E(a_t|F_{t-1})] = 0.$$

适当条件下 $\{a_t\}$ 的方差恒定，$\{a_t\}$ 是零均值白噪声列。

来计算 $\{a_t\}$ 的无条件方差。

由模型可知 σ_t^2 关于 F_{t-1} 可测，且 ε_t 与 F_{t-1} 独立，故有

$$\begin{aligned}\mathrm{Var}(a_t) =&E(a_t^2) = E[E(a_t^2|F_{t-1})]\\ =&E[E(\sigma_t^2\varepsilon_t^2|F_{t-1})] = E[\sigma_t^2 E(\varepsilon_t^2|F_{t-1})]\\ =&E[\sigma_t^2 E(\varepsilon_t^2)] = E(\sigma_t^2)\\ =&\alpha_0 + \alpha_1 E(a_{t-1}^2).\end{aligned}$$

因为 $\mathrm{Var}(a_t)$ 为常数，所以

$$\mathrm{Var}(a_t) = E(a_t^2) = \frac{\alpha_0}{1-\alpha_1}.$$

这要求 $0 < \alpha_1 < 1$。

17.2.2 对应的 AR 模型

所谓的 ARCH 效应，在数据上就是 $\{r_t\}$ 本身表现为白噪声，但是 $\{r_t^2\}$ 具有明显的相关性。以(17.5)为例，若 $r_t = a_t$，可以证明 r_t^2 服从一个 AR(1) 模型。

令

$$v_t = a_t^2 - \sigma_t^2.$$

则

$$\begin{aligned} a_t^2 =& \sigma_t^2 + (a_t^2 - \sigma_t^2) = \sigma_t^2 + v_t && (17.6)\\ =& \alpha_0 + \alpha_1 a_{t-1}^2 + v_t. && (17.7) \end{aligned}$$

因为 $0 < \alpha_1 < 1$，如果 $\{v_t\}$ 是白噪声列，则上式为关于 $\{a_t^2\}$ 的一个 AR(1) 模型。下面证明适当条件下 $\{v_t\}$ 是白噪声列。

由于

$$v_t = a_t^2 - \sigma_t^2 = a_t^2 - E(a_t^2|F_{t-1}),$$

所以 $\{v_t\}$ 是鞅差序列，是不相关列，且 $E(v_t) \equiv 0$。考虑 $\mathrm{Var}(v_t)$。

$$\begin{aligned} \mathrm{Var}(v_t) =& E(v_t^4) = E[(a_t^2 - \sigma_t^2)^2]\\ =& E[(\sigma_t^2\epsilon_t^2 - \sigma_t^2)^2]\\ =& E[\sigma_t^4(\epsilon_t^2 - 1)^2]\\ =& E\{E[\sigma_t^4(\epsilon_t^2 - 1)^2|F_{t-1}]\}\\ =& E\{\sigma_t^4 E[(\epsilon_t^2 - 1)^2|F_{t-1}]\}\\ =& E\{\sigma_t^4 E[(\epsilon_t^2 - 1)^2]\}\\ =& E[(\epsilon_t^2 - 1)^2]E(\sigma_t^4). \end{aligned}$$

记 $c_1 = E[(\epsilon_t^2 - 1)^2]$，则

$$\mathrm{Var}(v_t) = c_1 E(\sigma_t^4).$$

来求 $E(\sigma_t^4)$，先考虑

$$\begin{aligned} E(a_t^4) =& E(\varepsilon_t^4\sigma_t^4) = E[E(\varepsilon_t^4\sigma_t^4|F_{t-1})]\\ =& E[\sigma_t^4 E(\varepsilon_t^4|F_{t-1})] = E[\sigma_t^4 E(\varepsilon_t^4)]\\ =& E(\varepsilon_t^4)E(\sigma_t^4). \end{aligned}$$

记 $c_2 = E(\varepsilon_t^4)$，则 $E(\sigma_t^4) = c_2^{-1}E(a_t^4)$。而

$$\begin{aligned} E(a_t^4) =& c_2 E(\sigma_t^4)\\ =& c_2 E[(\alpha_0 + \alpha_1 a_{t-1}^2)^2]\\ =& c_2\alpha_0 + 2c_2\alpha_0\alpha_1 E(a_{t-1}^2) + c_2\alpha_1^2 E(a_{t-1}^4)\\ =& c_2\alpha_0 + \frac{2c_2\alpha_0^2\alpha_1}{1-\alpha_1} + c_2\alpha_1^2 E(a_{t-1}^4). \end{aligned}$$

这构成了 $E(a_t^4)$ 的递推式。令 $E(a_t^4) = E(a_{t-1}^4)$，可解得

$$E(a_t^4) = \frac{c_2\alpha_0 + \frac{2c_2\alpha_0^2\alpha_1}{1-\alpha_1}}{1 - c_2\alpha_1^2},\ t = 0, 1, 2, \ldots. \tag{17.8}$$

只要 $E(a_0^4)$ 满足上式，且参数 α_0, α_1，c_1, c_2 使得上式为正数值，则 $E(a_t^4)$ 为常数，从而 $\mathrm{Var}(v_t)$ 为常数，从而 $\{v_t\}$ 是零均值白噪声列。于是 $\{a_t^2\}$ 服从一个 AR(1) 模型。

为了 $E(a_0^4)$ 满足条件，可设 σ_0 与 $\{r_t, \varepsilon_t, t \geq 0\}$ 都独立，且满足

$$E(\sigma_0^2) = \frac{\alpha_0}{1-\alpha_1},$$

且 $E(a_0^4) = c_2 E(\sigma_0^4)$ 满足(17.8)，则 a_t^2 满足 AR(1) 模型(17.7)。

17.2.3 高阶矩

前面关于 $\{a_t^2\}$ 的 AR 模型的讨论用到了高阶矩。有些应用也需要利用 a_t 的高阶矩，例如，为了研究 a_t 是否厚尾分布，需要计算峰度，峰度依赖于四阶矩。

高阶矩的存在要求(17.1)中的 ε_t 与 $\{\alpha_j\}$ 满足一定的约束性条件。若(17.1)中的 ε_t 服从标准正态分布，则其有任意阶矩，这时条件四阶矩

$$\begin{aligned} E(a_t^4|F_{t-1}) =& E(\sigma_t^4 \varepsilon_t^4 | F_{t-1}) = \sigma_t^4 E(\varepsilon_t^4 | F_{t-1}) \\ =& 3\sigma_t^4 = 3(\alpha_0 + \alpha_1 a_{t-1}^2)^2, \end{aligned}$$

从而无条件的四阶矩为

$$\begin{aligned} E(a_t^4) =& E[E(a_t^4|F_{t-1})] = 3E[(\alpha_0 + \alpha_1 a_{t-1}^2)^2] \\ =& 3\left[\alpha_0^2 + 2\alpha_0\alpha_1 E(a_{t-1}^2) + \alpha_1^2 E(a_{t-1}^4)\right]. \end{aligned}$$

这构成了 $E(a_t^4)$ 的递推式，令 $Ea_t^4 = Ea_{t-1}^4$，于是

$$(1 - 3\alpha_1^2) Ea_t^4 = 3\left(\alpha_0 + 2\alpha_0\alpha_1 \frac{\alpha_0}{1-\alpha_1}\right),$$

这要求 $1 - 3\alpha_1^2 > 0$，即 $0 < \alpha_1 < \frac{\sqrt{3}}{3}$。这时

$$Ea_t^4 = \frac{3\alpha_0^2(1+\alpha_1)}{(1-\alpha_1)(1-3\alpha_1^2)}.$$

只要 $E(a_0^4)$ 满足上式，则上式对 $t \geq 0$ 都成立。由此可以计算 a_t 的峰度为

$$\frac{Ea_t^4}{[\mathrm{Var}(a_t^2)]^2} = 3\frac{1-\alpha_1^2}{1-3\alpha_1^2} > 3.$$

a_t 的超额峰度为

$$\frac{Ea_t^4}{[\mathrm{Var}(a_t^2)]^2} - 3 = \frac{6\alpha_1^2}{1-3\alpha_1^2} > 0.$$

这说明 $\{a_t\}$ 序列是厚尾（重尾）分布，其样本比均值和方差相同的正态序列有更多的离群值（outliers）。这与实证分析中对资产收益率的分布观察结果一致。

17.2.4 一般 ARCH 模型的性质

对一般的 ARCH(m) 模型，其性质与 ARCH(1) 类似，但公式更复杂。

模型可推广为二次型表示，见 (Tsay, 2010) 第 3 章。

17.3 ARCH 模型的优缺点

优点：

1. 可以产生波动率聚集；
2. 扰动 a_t 具有厚尾分布。

缺点：

1. 因为假定 a_{t-j} 通过 a_{t-j}^2 影响波动率 σ_t，所以正的扰动和负的扰动对波动率影响相同，但是实际的资产收益率中正负扰动对波动率影响不同，较大的负扰动比正扰动引起的波动更大。
2. ARCH 模型对模型参数有较严格的约束条件，即使是 ARCH(1)，为了能计算峰度，也需要 $\alpha_1 \in (0, \frac{\sqrt{3}}{3})$，高阶的 ARCH($m$) 的约束条件更为复杂。这对带高斯新息的 ARCH 模型通过超额峰度表达厚尾性是一个限制。
3. 只能描述条件方差的变化，但是不能解释变化的原因。
4. 由模型做的波动率预测会偏高。
5. 可能需要较大的 m。

17.4 ARCH 模型的建模步骤

17.4.1 定阶

在 ARCH 效应检验显著后，可以通过考察 $\{a_t^2\}$ 序列的 PACF 来对 ARCH 模型定阶。下面解释理由。

首先，模型为

$$\sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2$$

因为 $E(a_t^2|F_{t-1}) = \sigma_t^2$，所以认为近似有

$$a_t^2 \approx \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2$$

这样可以用 $\{a_t^2\}$ 序列的 PACF 的截尾性来估计 ARCH 阶 m。

另一方面，令 $\eta_t = a_t^2 - \sigma_t^2$，可以证明 $\{\eta_t\}$ 为零均值不相关白噪声列，则 a_t^2 有模型

$$a_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2 + \eta_t$$

这是 $\{a_t^2\}$ 的 AR(m) 模型，但不要求 $\{\eta_t\}$ 独立同分布。从这个模型用最小二乘法估计 $\{\alpha_j\}$ 是相合估计，但不是有效（方差最小）估计。因此从 $\{a_t^2\}$ 的 PACF 估计 m 是合理的。

作为例子，考虑 §16.4.2的美元对欧元汇率日对数收益率问题，画出对数收益率平方序列的 PACF:

```
d.useu <- read_table(
  "d-useu9910.txt",
  col_types=cols(.default=col_double())
)
xts.useu <- with(d.useu, xts(rate, make_date(year, mon, day)))
xts.useu.lnrtn <- diff(log(xts.useu))[-1,]
eu <- c(coredata(xts.useu.lnrtn))
```

```
pacf(eu^2, main="")
```

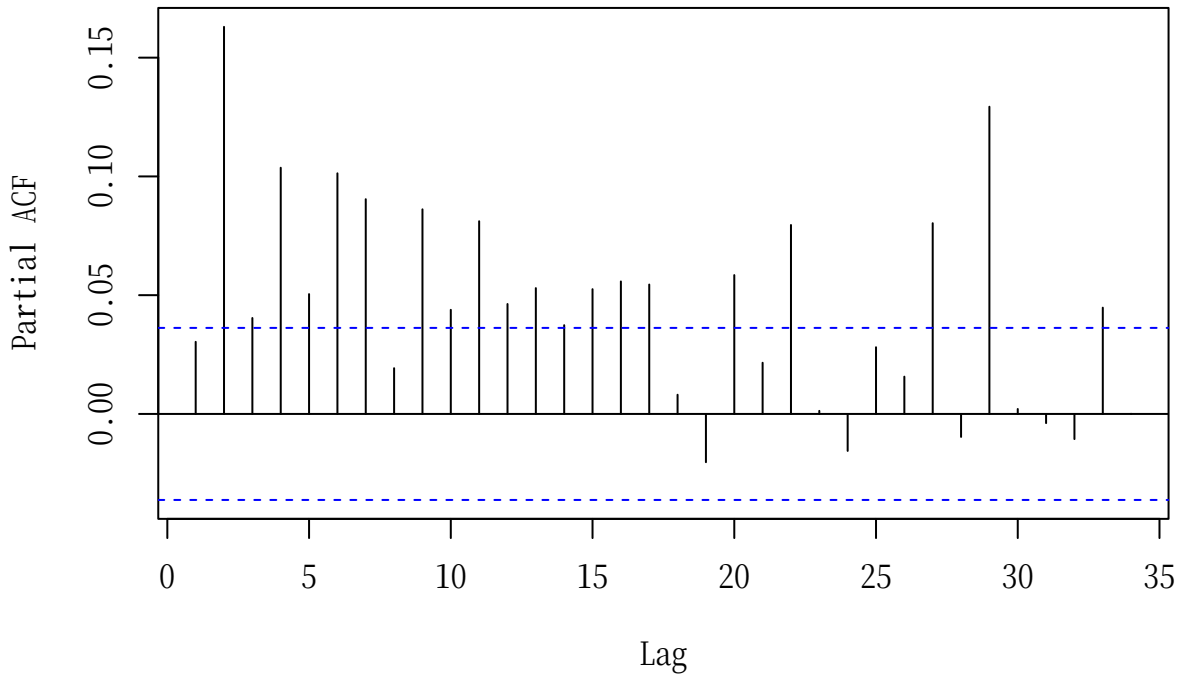


图 17.1: 美元对欧元日对数收益率平方的 PACF

结果提示需要用高阶（如 $m = 29$）的 ARCH 模型，从 AR 模型建模经验，有可能采用类似 ARMA 格式的模型会减少参数使用。GARCH 模型可以进行这种改进。

17.4.2 模型估计

对 ARCH(m) 模型，考虑最大似然估计或条件最大似然估计。扰动 $a_t = \varepsilon_t \sigma_t$, $\sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2$，记模型参数 $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_m)^T$。模型的似然函数与假定的 ε_t 的分布有关，存在多种似然函数形式。

似然函数即 $a_1, \ldots, a_T$ 的联合密度为

$$\begin{aligned} & f(a_1, \ldots, a_T|\alpha) \\ =& f(a_T|F_{T-1}, \alpha) f(a_{T-1}|F_{T-2}, \alpha) \ldots f(a_{m+1}|F_m, \alpha) f(a_1, \ldots, a_m|\alpha). \end{aligned}$$

其中 $f(a_t|F_{t-1}, \alpha)$ 表示给定 $t-1$ 时刻已知值条件下 a_t 的条件密度。

17.4.2.1 正态假设

当假定 ε_t 为独立同标准正态分布分布随机变量列时，在已知 $a_1, \ldots, a_{t-1}$ 条件下，

$$\sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \cdots + \alpha_m a_{t-m}^2$$

看成已知数，$a_t=\varepsilon_t\sigma_t$ 的条件分布为 $\mathrm{N}(0,\sigma_t^2)$，于是似然函数为

$$f(a_1,\cdots,a_T|\alpha)=\prod_{t=m+1}^{T}\frac{1}{\sqrt{2\pi}\sigma_t}\exp\left(-\frac{1}{2}\frac{a_t^2}{\sigma_t^2}\right)f(a_1,\cdots,a_m|\alpha)$$

其中 $f(a_1,\cdots,a_m|\alpha)$ 形式比较复杂，常常从似然函数中去掉此项，变成条件似然函数，当 T 较大时去掉此项的影响很小。条件似然函数为

$$f(a_{m+1},\cdots,a_T|\alpha)=\prod_{t=m+1}^{T}\frac{1}{\sqrt{2\pi}\sigma_t}\exp\left(-\frac{1}{2}\frac{a_t^2}{\sigma_t^2}\right)$$

条件对数似然函数为

$$\ell(a_{m+1},\cdots,a_T|\alpha)=-\frac{1}{2}\sum_{t=m+1}^{T}\left[\ln\sigma_t^2+\frac{a_t^2}{\sigma_t^2}\right]+\text{常数项}\tag{17.9}$$

其中 $\sigma_t^2=\alpha_0+\alpha_1a_{t-1}^2+\cdots+\alpha_ma_{t-m}^2$ 是 α 的函数，给定一组 α，就可以计算 $\sigma_t^2,t=m+1,\cdots,T$，从而计算似然函数值。最大化条件对数似然函数得到的估计称为正态假设下的条件最大似然估计。

17.4.2.2 t 分布假设

因为收益率分布厚尾，有些应用中假设 $\{a_t\}$ 的方差模型中的 ε_t 服从标准化 t 分布，自由度为 v。设 ξ 为服从 $\mathrm{t}(v)$ 分布的随机变量，则 $v>2$ 时 $E\xi=0,\mathrm{Var}(\xi)=v/(v-2)$, 令 $\eta=\frac{\xi}{\sqrt{v/(v-2)}}$，称 η 的分布为标准化 $\mathrm{t}(v)$ 分布，设 ε_t 分布为标准化 $\mathrm{t}(v)$ 分布，其分布密度为

$$\begin{aligned}f(\varepsilon_t|v)=&\frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{v}{2}\right)\sqrt{(v-2)\pi}}\left(1+\frac{\varepsilon_t^2}{v-2}\right)^{-\frac{v+1}{2}},\\&\varepsilon_t\in(-\infty,\infty),\quad(v>2).\end{aligned}\tag{17.10}$$

这时，由 $a_t=\varepsilon_t\sigma_t$，其中 $\sigma_t^2=\mathrm{Var}(a_t|F_{t-1})$，得条件似然函数

$$f(a_{m+1},\cdots,a_T|\alpha,a_1,\cdots,a_m)\tag{17.11}$$

$$=\left[\frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{v}{2}\right)\sqrt{(v-2)\pi}}\right]^{T-m}\prod_{t=m+1}^{T}\frac{1}{\sigma_t}\left(1+\frac{\varepsilon_t^2}{(v-2)\sigma_t^2}\right)^{-\frac{v+1}{2}}.\tag{17.12}$$

可以先验地取定自由度 v 的值，在(17.12)中关于 α 求最大值；也可以将 v 和 α 一起在(17.12)中求最大值。这样得到的参数估计称为在 t 分布下得条件最大似然估计。

先验地设定自由度 v 时，通常取 v 在 3 到 6 之间。（条件）对数似然函数为

$$\ell(a_{m+1},\cdots,a_T|\alpha,a_1,\cdots,a_m)\tag{17.13}$$

$$=-\sum_{t=m+1}^{T}\left\{\frac{v+1}{2}\ln\left(1+\frac{a_t^2}{(v-2)\sigma_t^2}\right)+\frac{1}{2}\ln\sigma_t^2\right\}+\text{常数项}.\tag{17.14}$$

其中 $\sigma_t^2=\alpha_0+\alpha_1a_{t-1}^2+\cdots+\alpha_ma_{t-m}^2$ 依赖于 $\alpha=(\alpha_0,\alpha_1,\cdots,\alpha_m)^T$。

将自由度 v 也看作未知参数时，（条件）对数似然函数为

$$\ell(a_{m+1},\cdots,a_T|v,\alpha,a_1,\cdots,a_m)\tag{17.15}$$

$$=(T-m)\left[\ln\Gamma\left(\frac{v+1}{2}\right)-\ln\Gamma\left(\frac{v}{2}\right)-\frac{1}{2}\ln((v-2)\pi)\right]\tag{17.16}$$

$$+\ell(a_{m+1},\cdots,a_T|\alpha,a_1,\cdots,a_m).\tag{17.17}$$

其中最后一项就是(17.14)的值。

17.4.2.3 有偏 t 分布假设

资产收益率分布除了厚尾之外还常常有偏。可以修改 t 分布使其变成标准化的有偏的单峰密度。有多种方法可以做这种修改，这里使用 (Fernandez & Steel, 1998) 的做法。该方法可以在任何连续单峰且关于 0 对称的一元分布中引入有偏性。将 t(v) 分布进行有偏化后密度为

$$g(\varepsilon|v,\xi)=\begin{cases}\frac{2c_2}{\xi+\frac{1}{\xi}}f(\xi(c_2\varepsilon_t+c_1)\,|\,v), & \varepsilon_t<-\frac{c_1}{c_2},\\ \frac{2c_2}{\xi+\frac{1}{\xi}}f((c_2\varepsilon_t+c_1)/\xi\,|\,v), & \varepsilon_t\geq-\frac{c_1}{c_2}.\end{cases}$$

其中 $f(\cdot|v)$ 是(17.10)定义的标准化 t(v) 分布密度，参数 v 是自由度，可以控制厚尾程度；参数 ξ^2 是密度在峰值右边的面积与在峰度左边的面积之比，代表了有偏的程度，当 $\xi=1$ 时还是标准化 t(v) 分布，当 $\xi>1$ 时右偏，当 $\xi<1$ 时左偏，

$$\begin{aligned}c_1=&\frac{\Gamma\left(\frac{v-1}{2}\right)\sqrt{v-2}}{\sqrt{\pi}\Gamma\left(\frac{v}{2}\right)}\left(\xi-\frac{1}{\xi}\right),\\ c_2^2=&\xi^2-\frac{1}{\xi^2}-1-c_1^2.\end{aligned}$$

对标准化 t 分布，取 $v=5,10,30$，作密度图：

```
dtstd <- function(x, df)
  sqrt(df/(df-2)) * dt(x*sqrt(df/(df-2)), df)
```

```
curve(dtstd(x, 5), -3, 3, lwd=1, lty=1, col="black",
      xlab="x", ylab=" 密度")
curve(dtstd(x, 10), -3, 3, lwd=1, lty=2, col="red", add=TRUE)
curve(dtstd(x, 30), -3, 3, lwd=1, lty=3, col="blue", add=TRUE)
legend("topleft", lty=c(1,2,3), col=c("black", "red", "blue"),
       legend=c(expression(v==5), expression(v==10), expression(v==30)))
```

对有偏的 t 分布，取 $v=5$，$\xi=0.75,1,1.5$, 作密度图：

```
dtskew <- function(x, df, xi){
  y <- numeric(length(x))
  par1 <- gamma((df-1)/2)*sqrt(df-2)/sqrt(pi)/gamma(df/2)*(xi - 1/xi)
  par2 <- sqrt(xi^2 + 1/xi^2 - 1 - par1^2)
  c1 <- 2*par2/(xi + 1/xi)
  sele <- x < -par1/par2
  y[sele] <- c1 * dtstd(xi*(par2*x[sele] + par1), df)
  y[!sele] <- c1 * dtstd((par2*x[!sele] + par1)/xi, df)

  y
}
```

```
curve(dtskew(x, 5, 1.0), -3, 3, ylim=c(0, 0.6),
      lwd=1, lty=1, col="black",
      xlab="x", ylab=" 密度")
curve(dtskew(x, 5, 0.75), -3, 3, lwd=1, lty=2, col="red", add=TRUE)
```

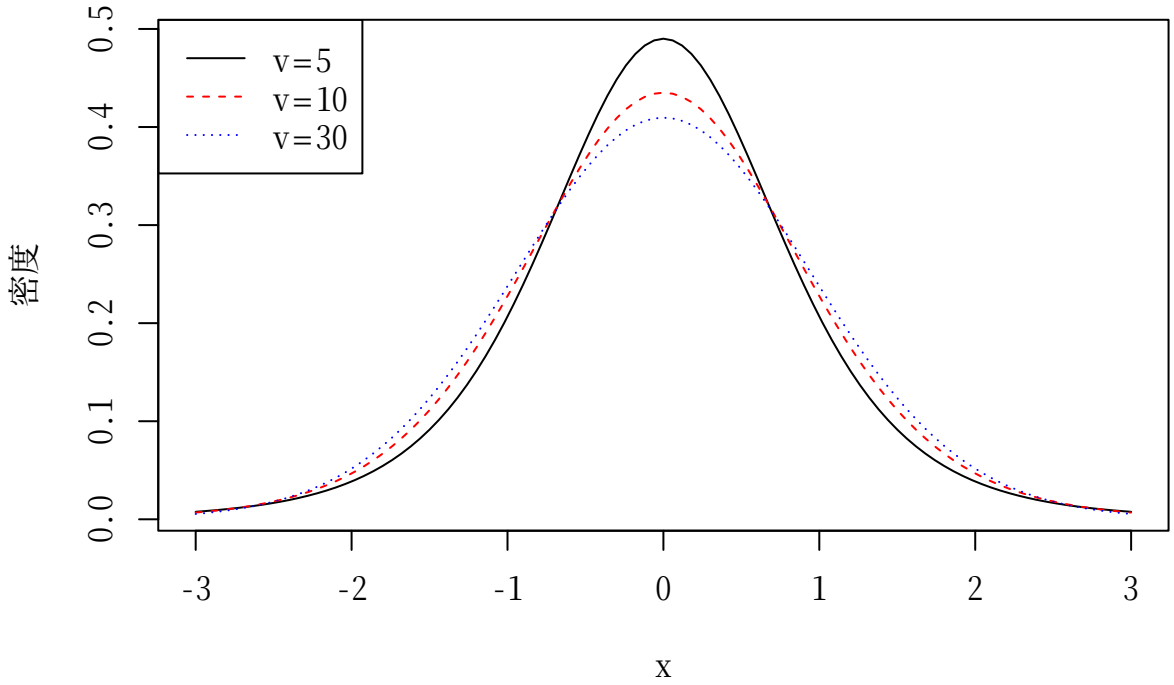


图 17.2: 标准化 t 分布密度图

```
curve(dtskew(x, 5, 1.5), -3, 3, lwd=1, lty=3, col="blue", add=TRUE)
legend("topleft", lty=c(1,2,3), col=c("black", "red", "blue"),
       legend=c(expression(xi==1), expression(xi==0.75), expression(xi==1.5)))
```

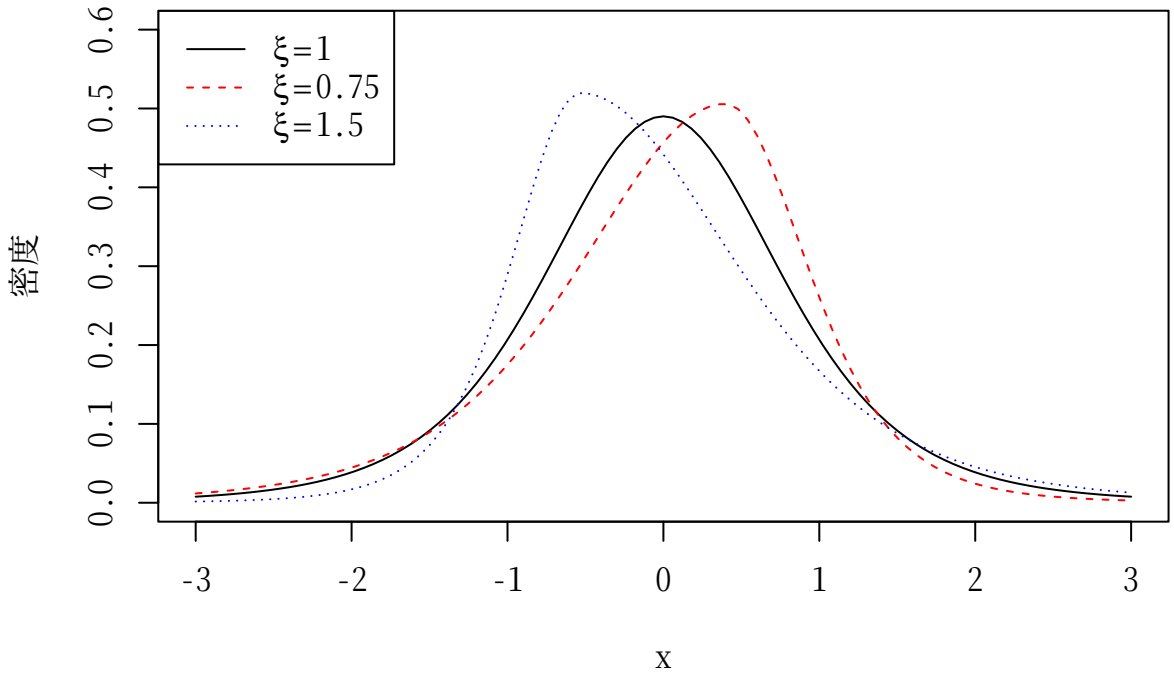


图 17.3: 有偏 t 分布密度图

17.4.2.4 广义误差分布假设

ε_t 的另一种可取分布是广义误差分布 (GED)，密度为

$$f(x|v) = \frac{v}{\lambda 2^{1+\frac{1}{v}}\Gamma(\frac{1}{v})} e^{-\frac{1}{2}|\frac{x}{\lambda}|^v},\ x \in (-\infty, \infty)\ (0 < v \leq \infty).$$

其中 $v = 2$ 时即标准正态分布，$0 < v < 2$ 时为厚尾分布。

$$\lambda = \left[2^{-\frac{2}{v}} \frac{\Gamma(\frac{1}{v})}{\Gamma(\frac{3}{v})}\right]^{\frac{1}{2}}.$$

17.4.3 模型验证

对一个建立好的 ARCH 模型，可计算标准化残差

$$\tilde{a}_t = \frac{a_t}{\sigma_t},$$

其中 a_t 是均值方程的残差，σ_t 是波动率方程拟合的值。$\{\tilde{a}_t\}$ 应表现为零均值、单位标准差的独立同分布序列。

对 $\{\tilde{a}_t\}$ 作 Ljung-Box 白噪声检验，可以考察均值方程的充分性。对 $\{\tilde{a}_t^2\}$ 作 Ljung-Box 白噪声检验，可以考察波动率方程的充分性。$\{\tilde{a}_t\}$ 的偏度、峰度、QQ 图可以用来与 ε_t 的假定分布比较，以检验模型假定的正确性。

R 的 fGarch 扩展包可以估计 ARCH 模型，并提供了多种诊断图。rugarch 包也可以估计和作图。

17.4.4 预测

ARCH 模型的预测类似 AR 模型的预测。从预测原点 h 出发，对 σ_t^2 序列作超前一步预测，即预测 σ_{h+1}^2，有

$$\sigma_h^2(1) = \sigma_{h+1}^2 = \alpha_0 + \alpha_1 a_h^2 + \cdots + \alpha_m a_{h+1-m}^2.$$

要做超前 2 步预测时，因为 a_{h+1} 未知，有 $E(a_{h+1}^2|F_h) = \sigma_h^2(1)$, 所以

$$\sigma_h^2(2) = \alpha_0 + \alpha_1\sigma_h^2(1) + \alpha_2 a_h^2 + \cdots + \alpha_m a_{h+2-m}^2.$$

一般地，$\sigma_h^2(\ell)$ 可以滚动计算，

$$\sigma_h^2(\ell) = \alpha_0 + \sum_{j=1}^{m} \alpha_j \sigma_h^2(\ell - j).$$

其中 $l - j \le 0$ 时 $\sigma_h^2(\ell - j) = a_{m+\ell-j}^2$。

17.5 ARCH 模型建模实例

两个实例，Intel 公司股票月对数收益率序列的波动率建模，美元对欧元的汇率的日对数收益率的波动率建模。

17.5.1 Intel 公司股票 ARCH 建模实例

继续使用 1973 年到 2009 年 Intel 公司股票的月对数收益率。ARCH 效应检验一节进行了 ARCH 效应检验，证明有 ARCH 效应。

数据读入：

```
d.intel <- read_table(
  "m-intcsp7309.txt",
  col_types=cols(
    .default=col_double(),
    date=col_date("%Y%m%d")
  ))
xts.intel <- xts(
  log(1 + d.intel[["intc"]]), d.intel[["date"]]
)
tclass(xts.intel) <- "yearmon"
ts.intel <- ts(c(coredata(xts.intel)), start=c(1973,1), frequency=12)
at <- ts.intel - mean(ts.intel)
```

序列的均值模型是常数均值。减去均值之后的残差的平方的 ACF:

```
forecast::Acf(at^2, lag.max=36, main="")
```

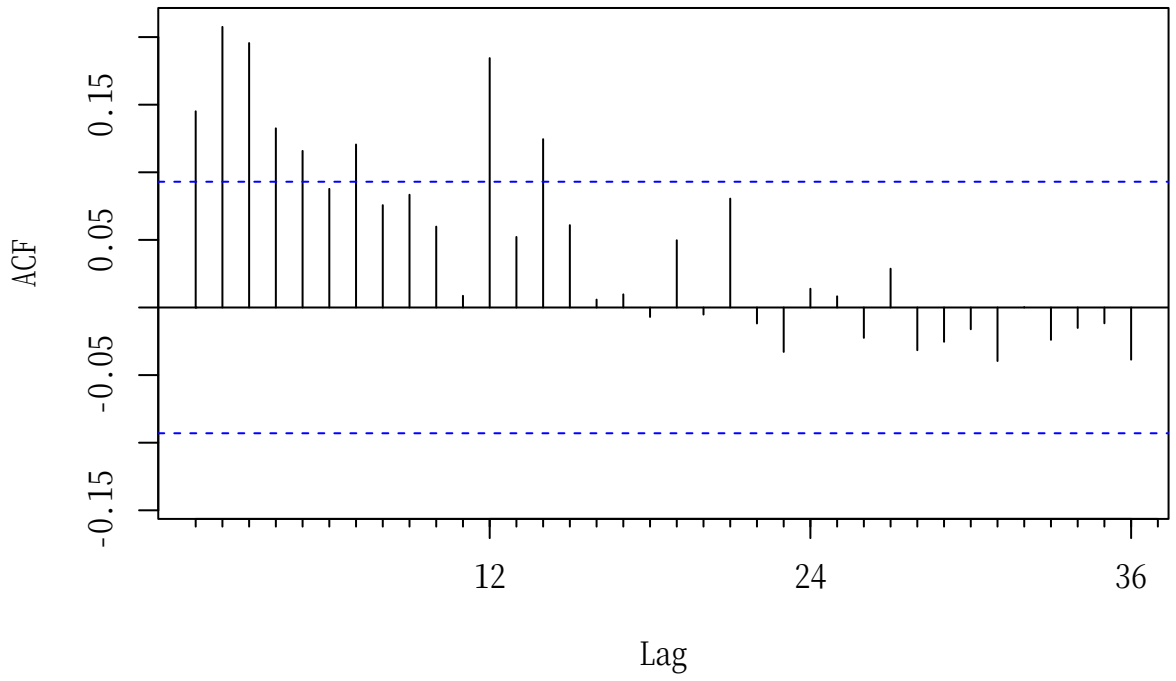


图 17.4: Intel 股票建模中心化的残差平方的 ACF

在频率 12 处较高，另外在滞后 1、2、3 上较突出。

残差的平方的 PACF:

```
pacf(at^2, lag.max=36, main="")
```

残差平方的 PACF 在滞后 12 处较高，另外在滞后 1 到 3 较高。可考虑建立 ARCH(3) 作为波动率方程。

设 r_t 为收益率，拟建立如下的均值方程和波动率方程：

$$\begin{aligned} r_t =& \mu + a_t, \quad a_t = \varepsilon_t \sigma_t, \\ \sigma_t^2 =& \alpha_0 + \alpha_1 a_{t-1}^2 + \alpha_2 a_{t-2}^2 + \alpha_3 a_{t-3}^2. \end{aligned}$$

使用 fGarch 包的 `garchFit()` 函数建立 ARCH 模型。

```
library(fGarch)
mod1 <- garchFit(~ 1 + garch(3,0), data=c(ts.intel), trace=FALSE)
summary(mod1)
```

```
##
## Title:
```

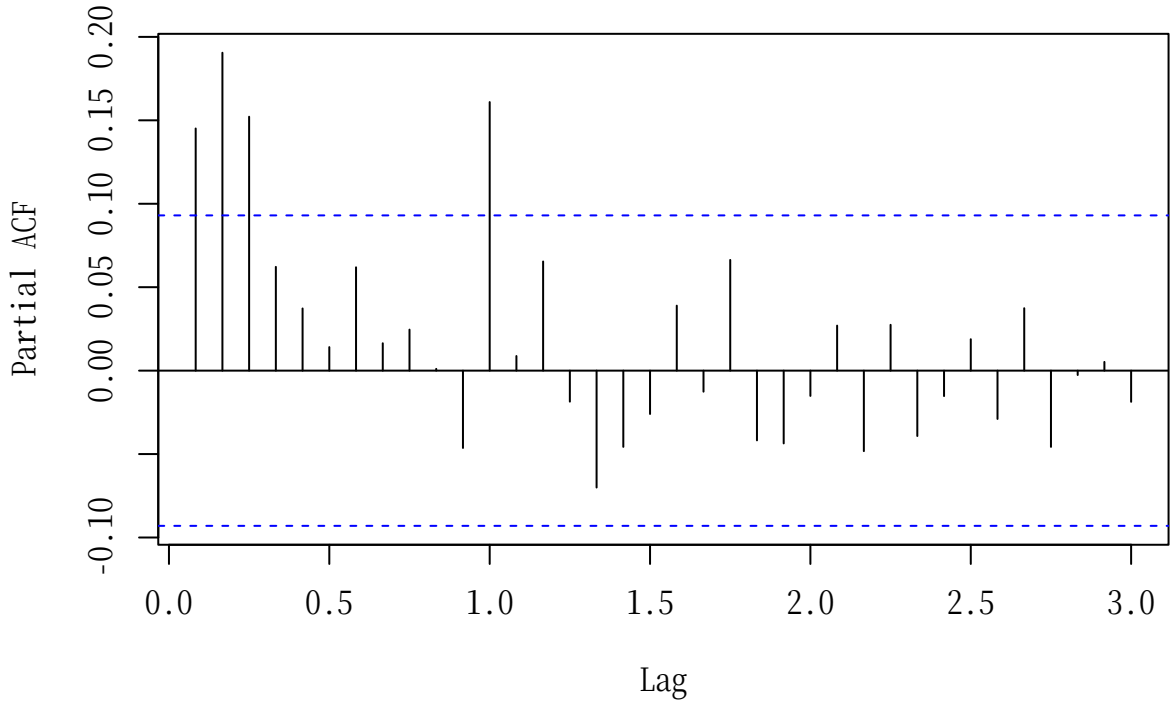


图 17.5: Intel 股票建模中心化的残差平方的 PACF

```
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(3, 0), data = c(ts.intel), trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(3, 0)
## <environment: 0x0000013a37bceb38>
##  [data = c(ts.intel)]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1    alpha2    alpha3
## 0.012567  0.010421  0.232889  0.075069  0.051994
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.012567    0.005515    2.279   0.0227 *
## omega   0.010421    0.001238    8.418   <2e-16 ***
## alpha1  0.232889    0.111541    2.088   0.0368 *
## alpha2  0.075069    0.047305    1.587   0.1125
## alpha3  0.051994    0.045139    1.152   0.2494
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  303.9607    normalized:  0.6845963
##
## Description:
##  Tue May 30 17:03:32 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  203.3622  0
##  Shapiro-Wilk Test  R    W      0.9635971 4.89862e-09
##  Ljung-Box Test     R    Q(10)  9.260779  0.5075465
##  Ljung-Box Test     R    Q(15)  19.36748  0.1975619
```

```
##  Ljung-Box Test     R    Q(20)  20.46983  0.4289059
##  Ljung-Box Test     R^2  Q(10)  7.322124  0.6947246
##  Ljung-Box Test     R^2  Q(15)  27.41529  0.0255293
##  Ljung-Box Test     R^2  Q(20)  28.1511   0.1058705
##  LM Arch Test       R    TR^2   25.23346  0.01375453
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -1.346670 -1.300546 -1.346920 -1.328481
```

程序中 `1 + garch(3,0)` 中 `garch(3,0)` 表示 σ_t^2 的 ARCH(3) 模型，`1` 表示均值方程是一个常数。输出结果中 `mu` 为均值方程的均值，`omega` 为 α_0，`alpha1` 为 α_1。所以得到的均值方程和波动率方程为：

$$\begin{aligned} r_t =& 0.0126 + a_t, \quad a_t = \varepsilon_t \sigma_t, \quad \varepsilon_t \text{ i.i.d} \mathrm{N}(0,1), \\ \sigma_t^2 =& 0.0104 + 0.02329 a_{t-1}^2 + 0.0751 a_{t-2}^2 + 0.0520 a_{t-3}^2. \end{aligned}$$

因为结果中 α_2 和 α_3 的估计值是不显著的，可拟合 ARCH(1) 模型为波动率方程：

```
mod2 <- garchFit( ~ 1 + garch(1,0), data=c(ts.intel), trace=FALSE)
summary(mod2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 0), data = c(ts.intel), trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 0)
## <environment: 0x0000013a39cef4f8>
##  [data = c(ts.intel)]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1
## 0.013130  0.011046  0.374976
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
```

```
## mu      0.013130    0.005318    2.469  0.01355 *
## omega   0.011046    0.001196    9.238  < 2e-16 ***
## alpha1  0.374976    0.112620    3.330  0.00087 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  299.9247    normalized:  0.675506
##
## Description:
##  Tue May 30 17:03:32 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  144.3782  0
##  Shapiro-Wilk Test  R    W      0.9678175 2.670339e-08
##  Ljung-Box Test     R    Q(10)  12.12247  0.2769434
##  Ljung-Box Test     R    Q(15)  22.30704  0.1000021
##  Ljung-Box Test     R    Q(20)  24.33411  0.228102
##  Ljung-Box Test     R^2  Q(10)  16.57804  0.08423799
##  Ljung-Box Test     R^2  Q(15)  37.44344  0.001089753
##  Ljung-Box Test     R^2  Q(20)  38.8139   0.007031666
##  LM Arch Test       R    TR^2   27.32894  0.006926884
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -1.337499 -1.309824 -1.337589 -1.326585
```

这个模型的 AIC 为 -1.3375，比上一个模型的 -1.3467 要差一些。

输出中给出了标准化残差 $\tilde{a}_t$ 的 Ljung-Box 白噪声检验结果，滞后 10 的 p 值为 0.2769, 承认白噪声；$\tilde{a}_t^2$ 的滞后 10 的 Ljung-Box 白噪声检验结果 p 值为 0.08, 在 0.05 水平下也可以承认白噪声。

Jarque-Bera 检验是正态分布的偏度峰度检验，Shapiro-Wilk 检验是正态性检验，零假设为 ε_t 服从正态分布，这两个检验的结果表明标准化残差不服从正态分布。

模型方程：

$$r_t = 0.0131 + a_t, \quad a_t = \varepsilon_t \sigma_t, \quad \varepsilon_t \text{ i.i.dN}(0,1) \tag{17.18}$$

$$\sigma_t^2 = 0.0110 + 0.3750 a_{t-1}^2 \tag{17.19}$$

但是，两个不同模型给出了不同的均值方程。这说明均值方程和波动率方程的参数是同时估计的，不是先估计均值方程然后利用均值方程的残差估计波动率方程。

为了进行模型验证，可计算标准化残差

$$\tilde{a}_t = \frac{a_t}{\sigma_t}, t = m+1, m+2, \ldots, T,$$

其中 σ_t 从估计的 ARCH 模型中递推地计算。$\{\tilde{a}_t\}$ 应表现得像是独立同分布零均值白噪声列。

计算标准化残差 $\{\tilde{a}_t\}$：

```
resi <- residuals(mod2, standardize=TRUE)
```

fGarch 包的 plot 函数可以对建模结果绘制多种诊断图形，包括：

1. 输入序列的时间序列图；
2. 估计的波动率（条件标准差）折线图；
3. 输入序列图，与正负 2 倍波动率叠加；
4. 输入的 ACF 图；
5. 输入的平方（即 r_t^2）的 ACF 图；
6. 互相关系数图（文档无说明）；
7. 拟合残差图；
8. 多个波动率图（文档无说明）；
9. 标准化残差图；
10. 标准化残差的 ACF 图；
11. 标准化残差平方的 ACF 图；
12. r_t^2 和 r_t 的互相关系数函数图；
13. 标准化残差相对于 ε_t 理论分布（条件分布）的 QQ 图。

标准化残差的时序图：

```
plot(mod2, which=9)
```

标准化残差 $\{\tilde{a}_t\}$ 的 ACF:

```
plot(mod2, which=10)
```

$\{\tilde{a}_t^2\}$ 的 ACF:

```
plot(mod2, which=11)
```

除了在滞后 11、滞后 21 还略高以外已经没有了低阶的波动率相关。

$\{\tilde{a}_t^2\}$ 的 PACF:

```
pacf(resi^2, lag.max=36, main="")
```

仅在高阶的滞后 11、滞后 21 还较高。作为低阶模型，ARCH(1) 作为波动率方程已经比较适合。

标准化残差相对于标准正态分布的 QQ 图：

```
plot(mod2, which=13)
```

这个图形说明标准化残差相对于正态分布有重尾。正态分布作为条件分布不够合适。

显示拟合的条件标准差序列（波动率序列）：

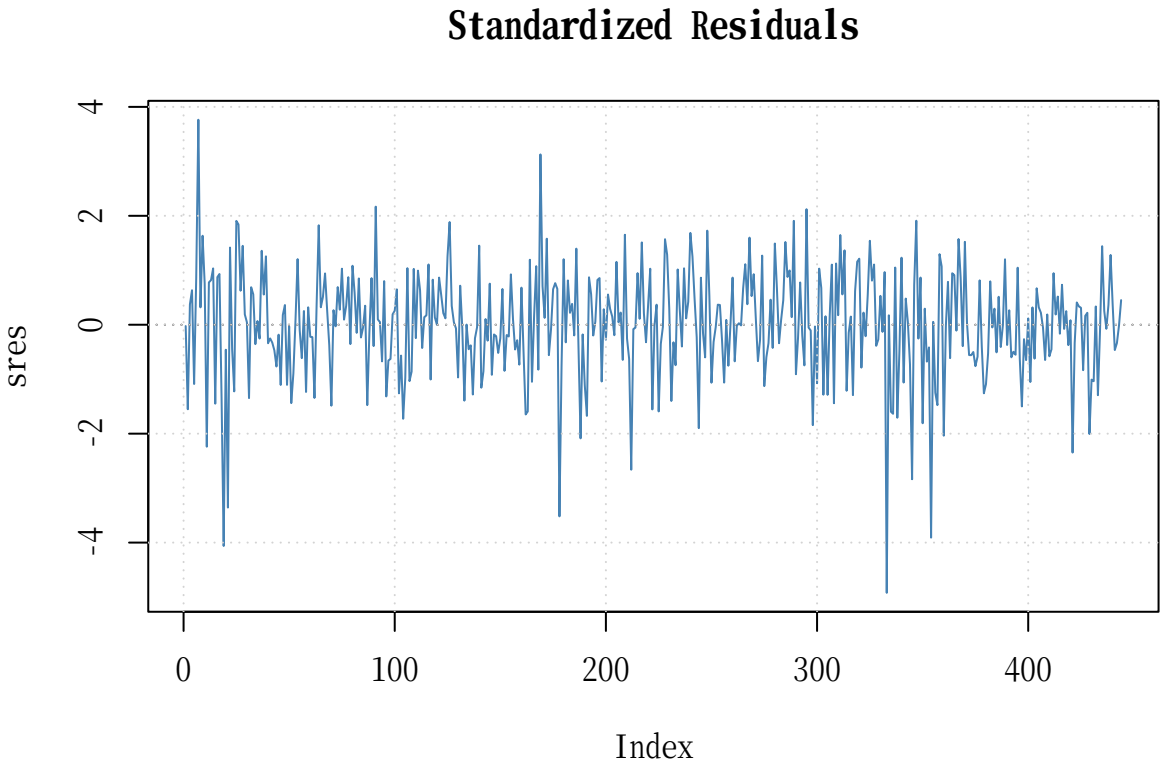


图 17.6: Intel 股票建模标准化残差

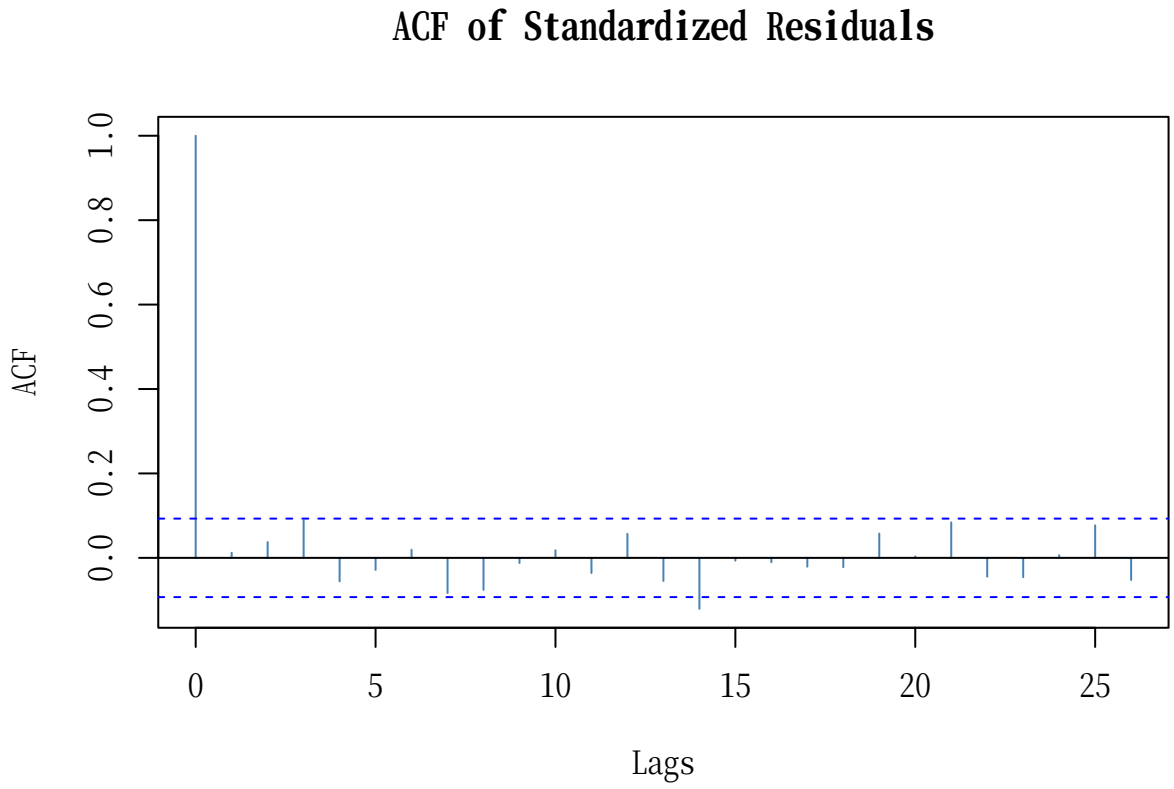


图 17.7: Intel 股票建模标准化残差的 ACF

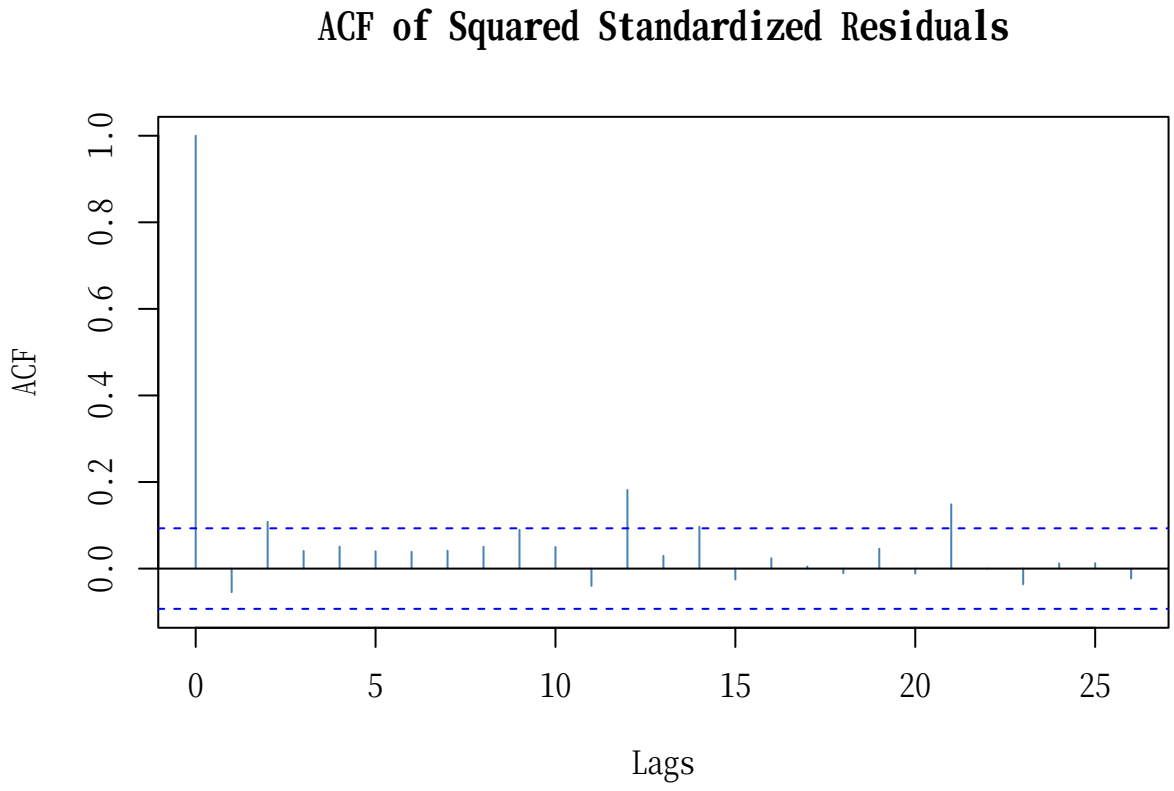


图 17.8: Intel 股票建模标准化残差平方的 ACF

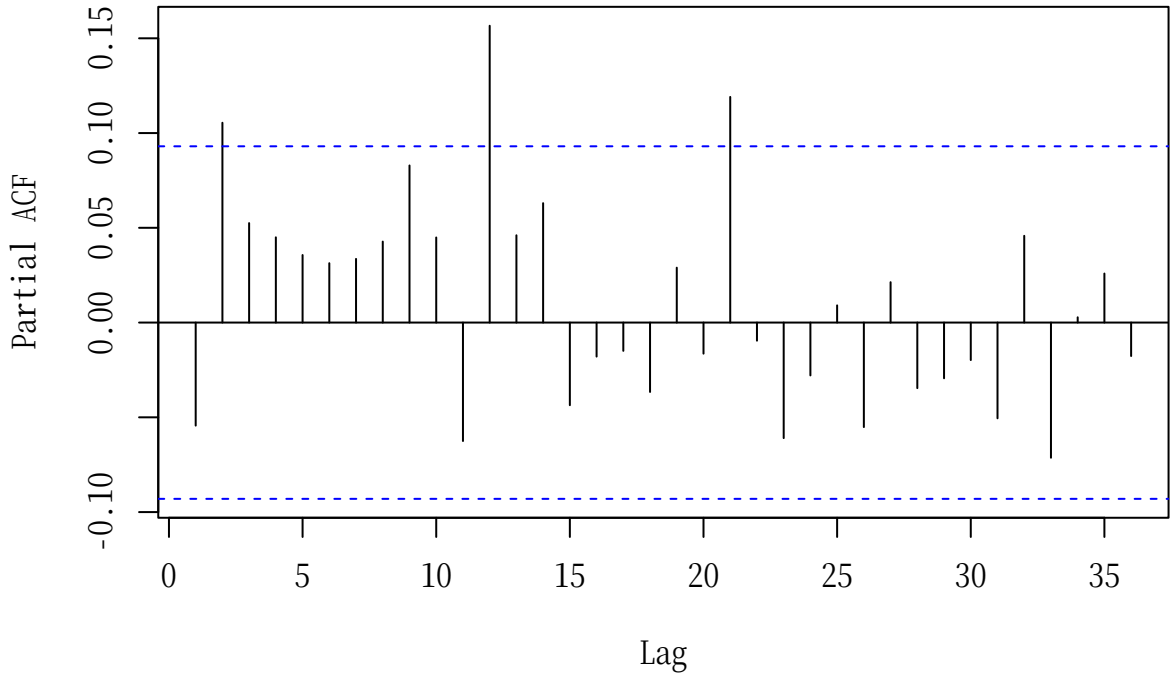


图 17.9: Intel 股票建模标准化残差平方的 PACF

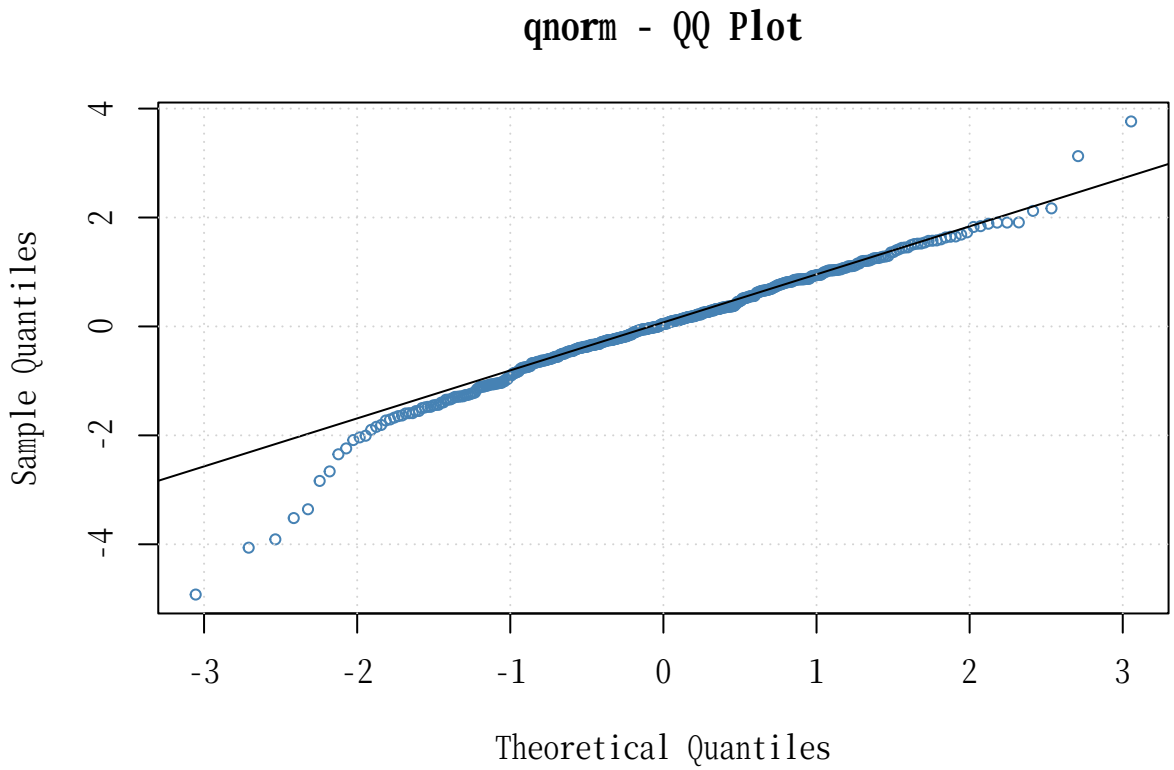


图 17.10: Intel 股票标准化残差正态 QQ 图

```
plot(mod2, which=2)
```

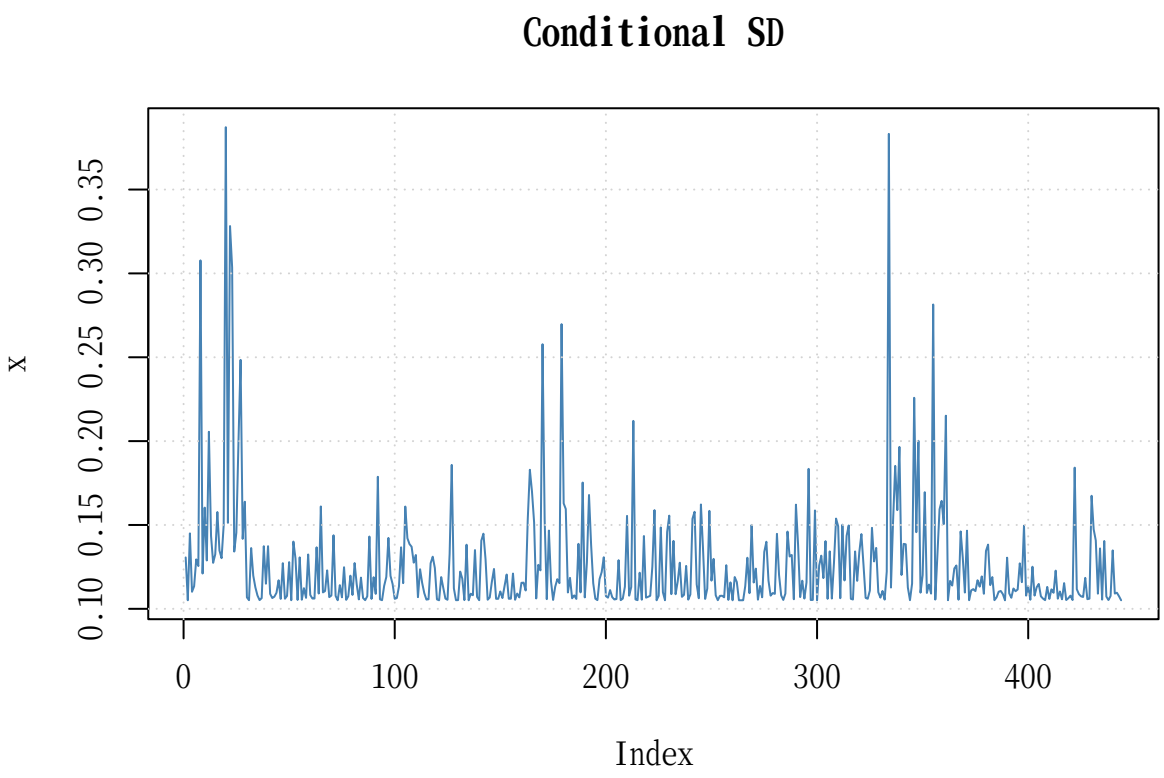


图 17.11: Intel 股票建模波动率

获得拟合的波动率用 `volatility()` 函数：

```
mod2v <- volatility(mod2)
head(mod2v)
```

```
## [1] 0.1306624 0.1051189 0.1450052 0.1101684 0.1134644 0.1294741
```

获得拟合的均值用 `fitted()` 函数。按理说这个模型的均值方程应该是常数 0.0131，这里 `fitted()` 返回的是原始的 r_t 序列。

```
mod2f <- fitted(mod2)
head(mod2f)
```

```
##             1            2           3            4            5           6
##  0.009999835 -0.150012753  0.067064079  0.082948635 -0.110348491  0.125162849
```

用 `predict()` 函数作超前若干步的预测，如：

```
mod2p <- predict(mod2, n.ahead=12)
mod2p
```

```
##    meanForecast meanError standardDeviation
```

```
## 1    0.01313002 0.1090512          0.1090512
## 2    0.01313002 0.1245214          0.1245214
## 3    0.01313002 0.1298481          0.1298481
## 4    0.01313002 0.1317900          0.1317900
## 5    0.01313002 0.1325108          0.1325108
## 6    0.01313002 0.1327801          0.1327801
## 7    0.01313002 0.1328809          0.1328809
## 8    0.01313002 0.1329187          0.1329187
## 9    0.01313002 0.1329329          0.1329329
## 10   0.01313002 0.1329382          0.1329382
## 11   0.01313002 0.1329402          0.1329402
## 12   0.01313002 0.1329410          0.1329410
```

预测包括均值的预测（显然是用 μ 预测的）、均值预测的标准误差、波动率的预测（是用 a_t 的值滚动计算的）。波动率长期预测接近于 ARCH 模型的无条件标准差。

模型(17.19)的一些特点：

第一，Intel 股票月对数收益率是 1.31%，年对数收益率是 15.72%，这是很了不起的。

直接计算收益率均值：

```
mean(ts.intel)
```

```
## [1] 0.0143273
```

第二，$\alpha_1^2 = 0.3750^2 = 0.14 < 1/3$，说明 a_t 序列有无条件四阶矩。

第三，r_t 的无条件标准差为

$$\sqrt{0.0110/(1-0.3750)} = 0.1327$$

直接从原始数据计算：

```
sd(ts.intel)
```

```
## [1] 0.1269101
```

结果相近但不相同。

第四，模型可以用来估计和预测 Intel 股票收益率的月波动率。

17.5.2 Intel 股票问题改用 t 分布

在 fGarch 的 `garchFit()` 函数中加选项 `cond.dist="std"`。

```
mod3 <- garchFit(~ 1 + garch(1,0), data=ts.intel,
  cond.dist="std", trace=FALSE)
summary(mod3)
```

```
##
## Title:
##  GARCH Modelling
```

```
##
## Call:
##  garchFit(formula = ~1 + garch(1, 0), data = ts.intel, cond.dist = "std",
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 0)
## <environment: 0x0000013a392f78a0>
##  [data = ts.intel]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##       mu     omega    alpha1     shape
## 0.017202  0.011816  0.277476  5.970256
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.017202    0.005195    3.311 0.000929 ***
## omega   0.011816    0.001560    7.574 3.62e-14 ***
## alpha1  0.277476    0.107183    2.589 0.009631 **
## shape   5.970256    1.529521    3.903 9.49e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  315.0899    normalized:  0.709662
##
## Description:
##  Tue May 30 17:03:32 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  157.78    0
##  Shapiro-Wilk Test  R    W      0.9663975 1.48822e-08
##  Ljung-Box Test     R    Q(10)  12.8594   0.2316394
##  Ljung-Box Test     R    Q(15)  23.40633  0.07588549
##  Ljung-Box Test     R    Q(20)  25.374    0.1874954
```

```
##  Ljung-Box Test      R^2  Q(10)  19.96094  0.02962426
##  Ljung-Box Test      R^2  Q(15)  42.55552  0.0001845072
##  Ljung-Box Test      R^2  Q(20)  44.06742  0.001473957
##  LM Arch Test        R    TR^2   29.76073  0.003033495
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -1.401306 -1.364407 -1.401466 -1.386755
```

因为已经采用条件 t 分布，所以结果中的 JB 检验和 SW 检验没有意义。

模型为

$$r_t = 0.0172 + a_t, \quad a_t = \varepsilon_t \sigma_t, \quad \varepsilon_t \text{ i.i.d} t^*(5.97) \tag{17.20}$$

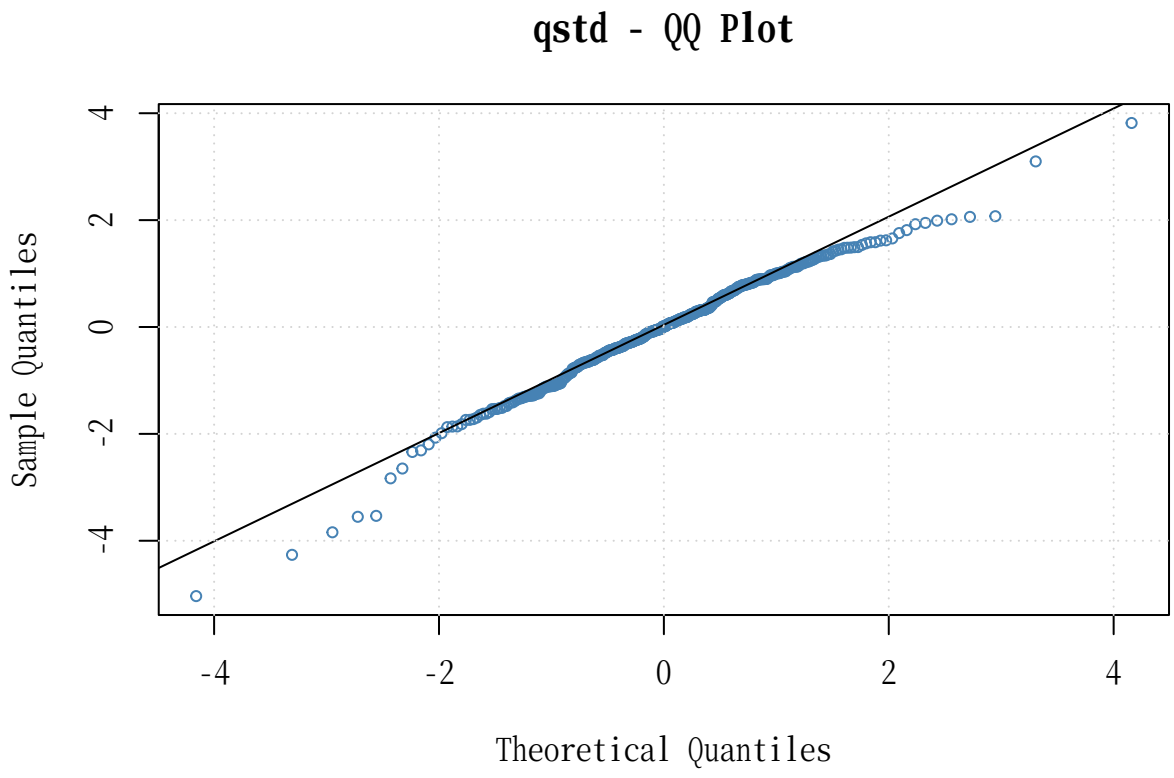
$$\sigma_t^2 = 0.0118 + 0.2775 a_{t-1}^2 \tag{17.21}$$

其中 t^* 表示标准化 t 分布。

a_t 的无条件标准差按模型估计为 $\sqrt{0.0118/(1-0.2775)} = 0.1278$，正态时为 0.1327，原始数据直接估计为 0.1269。标准化残差 $\tilde{a}_t$ 的滞后 10 的 Ljung-Box 白噪声检验 p 值为 0.23，可承认白噪声；标准化残差平方 $\tilde{a}_t^2$ 的滞后 10 的 Ljung-Box 白噪声检验 p 值为 0.03，仍有一定的残余波动率相关性。

作标准化残差相对于标准化 t 分布的 QQ 图：

```
plot(mod3, which=13)
```



可见这个分布比正态分布更符合，但仍存在左偏，有可能需要使用有偏 t 分布。

改用 t 分布以后，α_1 估计值从原来正态时的 0.3750 降低到了 0.2775，说明厚尾的 ε_t 分布降低了 ARCH 效应。本问题采用正态分布和 t 分布的结果差别不大。后面将指出，Intel 月对数收益率的波动率方程更合适的模型是 GARCH(1,1)。

fGarch 包的 `garchFit()` 函数支持多种条件分布，默认为正态分布，用 `cond.dist=` 指定分布：

- `"norm"`: 正态；
- `"snorm"`: 有偏正态;
- `"ged"`: 广义误差分布；
- `"sged"`: 有偏广义误差分布;
- `"std"`: t 分布;
- `"sstd"`: 有偏 t 分布；
- `"snig"`
- `"QMLE"`：拟最大似然估计，仍假设正态但是采用稳健标准误差估计；

17.5.3 欧元汇率 ARCH 建模实例

考虑 1999-01-04 到 2010-08-20 的欧元对美元汇率的日对数收益率。见 §16.4.2。均值方程为 $r_t = a_t$，并显示数据有 ARCH 效应。该序列是纯条件异方差模型的典型例子。

```
d.useu <- read_table(
  "d-useu9910.txt",
  col_types=cols(.default=col_double())
)
xts.useu <- with(d.useu, xts(rate, make_date(year, mon, day)))
xts.useu.lnrtn <- diff(log(xts.useu))[-1,]
eu <- c(coredata(xts.useu.lnrtn))
```

a_t^2 的 ACF:

```
forecast::Acf(eu^2, lag.max=36, main="")
```

呈现出低的较长期的相关。

a_t^2 的 PACF:

```
pacf(eu^2, lag.max=36, main="")
```

在低阶到滞后 11 可以，但是高阶仍有较大的值。考虑建立 ARCH(11) 模型。采用条件高斯似然函数：

```
mod4 <- garchFit(~ 1 + garch(11,0), data=eu, trace=FALSE)
summary(mod4)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(11, 0), data = eu, trace = FALSE)
```

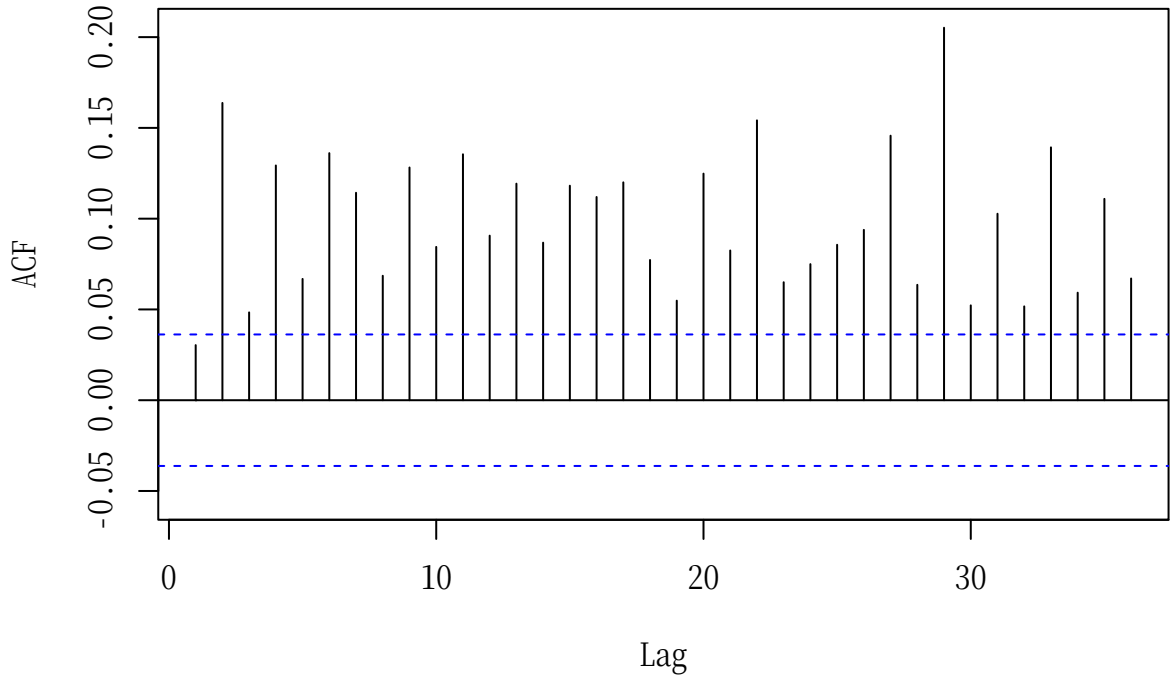


图 17.12: 欧元汇率平方的 ACF

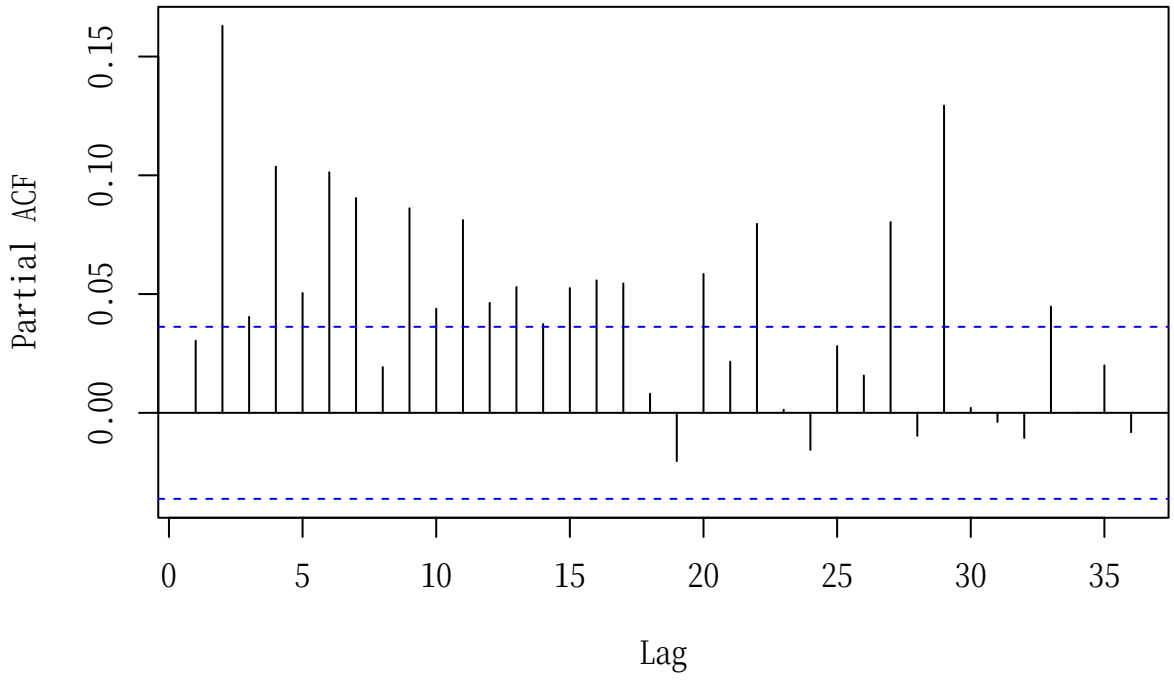


图 17.13: 欧元汇率平方的 PACF

```
##
## Mean and Variance Equation:
##  data ~ 1 + garch(11, 0)
## <environment: 0x0000013a350a5f58>
##  [data = eu]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1      alpha2      alpha3      alpha4
## 1.2646e-04  1.8902e-05  1.6608e-02  4.4563e-02  2.7210e-02  8.0370e-02
##     alpha5      alpha6      alpha7      alpha8      alpha9     alpha10
## 5.0112e-02  9.2192e-02  7.5283e-02  6.9536e-02  3.3466e-02  2.7824e-02
##    alpha11
## 3.8774e-02
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      1.265e-04   1.110e-04    1.140 0.254412
## omega   1.890e-05   1.727e-06   10.944  < 2e-16 ***
## alpha1  1.661e-02   1.575e-02    1.055 0.291563
## alpha2  4.456e-02   2.085e-02    2.137 0.032592 *
## alpha3  2.721e-02   1.700e-02    1.601 0.109370
## alpha4  8.037e-02   2.362e-02    3.402 0.000669 ***
## alpha5  5.011e-02   2.127e-02    2.355 0.018499 *
## alpha6  9.219e-02   2.274e-02    4.053 5.05e-05 ***
## alpha7  7.528e-02   2.406e-02    3.129 0.001755 **
## alpha8  6.954e-02   2.455e-02    2.832 0.004622 **
## alpha9  3.347e-02   2.022e-02    1.655 0.097835 .
## alpha10 2.782e-02   1.820e-02    1.528 0.126400
## alpha11 3.877e-02   1.906e-02    2.035 0.041890 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  10698.47    normalized:  3.652603
##
## Description:
##  Tue May 30 17:03:43 2023 by user: Lenovo
```

```
##
##
## Standardised Residuals Tests:
##                                Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  360.8056  0
##  Shapiro-Wilk Test  R    W      0.9891734 3.893905e-14
##  Ljung-Box Test     R    Q(10)  15.77628  0.1062181
##  Ljung-Box Test     R    Q(15)  21.50103  0.1215706
##  Ljung-Box Test     R    Q(20)  24.77444  0.2101971
##  Ljung-Box Test     R^2  Q(10)  4.801169  0.9040581
##  Ljung-Box Test     R^2  Q(15)  16.73082  0.3352091
##  Ljung-Box Test     R^2  Q(20)  27.5606   0.1202158
##  LM Arch Test       R    TR^2   11.96808  0.4482465
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -7.296329 -7.269777 -7.296368 -7.286766
```

通过对标准化残差 $\tilde{a}_t$ 和 $\tilde{a}_t^2$ 的 Ljung-Box 白噪声检验来看，模型是充分的。但是，Jarque-Bera 检验是正态分布的偏度峰度检验，Shapiro-Wilk 检验是正态性检验，这两个检验表明标准化残差不服从正态分布。

模型为

$$\begin{aligned} r_t =& 0.0013 + \sigma_t \varepsilon_t,\ \varepsilon_t \text{ i.i.d.N}(0,1) \\ \sigma_t^2 =& 1.89 \times 10^{-5} + 0.0166 a_{t-1}^2 + \cdots + 0.03877 a_{t-11}^2 \end{aligned}$$

下一节会指出，改用 GARCH 模型可以得到更精简的模型。

17.6 补充推导

17.6.1 新息性质

设 $\{r_t\}$ 为二阶矩过程，令

$$\begin{aligned} F_t =& \sigma(\{r_t, r_{t-1}, \dots\}), \\ a_t =& r_t - E(r_t|F_{t-1}). \end{aligned}$$

则 $\{a_t\}$ 为二阶矩过程。

$$\begin{aligned} E(a_t|F_{t-1}) =& E(r_t|F_{t-1}) - E[E(r_t|F_{t-1})|F_{t-1}] \\ =& 0. \end{aligned}$$

称满足上式的序列 $\{a_t\}$ 为关于 $\{F_t\}$ 的**鞅差序列**。

对鞅差序列，有

$$E[a_t] = E[E(a_t|F_{t-1})] = 0.$$

对 $t > s$ 有

$$\begin{aligned} E[a_t|F_s] =& E[E(a_t|F_{t-1})|F_s] \\ =& E[0|F_s] = 0. \end{aligned}$$

于是

$$\begin{aligned}&\mathrm{Cov}(a_t, a_s) = E(a_t a_s)\\ =&E[E(a_t a_s|F_s)] = E[a_s E(a_t|F_s)] = 0.\end{aligned}$$

所以二阶矩有限的鞅差序列是不相关列。如果 $\mathrm{Var}(a_t)$ 恒定，则 $\{a_t\}$ 是白噪声列。

17.6.2 白噪声性质

当 $\{a_t\}$ 服从 ARCH(1) 模型时，设时间 t 从 0 开始，令

$$F_t = \sigma(\{r_0, \varepsilon_0, \dots, r_t, \varepsilon_t\}),$$

其中 $\{\varepsilon_t\}$ iid $(0,1)$，且 ε_t 与 $r_0, \dots, r_{t-1}$ 独立，即 ε_t 与 F_{t-1} 独立。设

$$\sigma_0^2 = \frac{\alpha_0}{1-\alpha_1}.$$

这时，$\{a_t, F_t, t \geq 0\}$ 仍为鞅差序列。

注意

$$\begin{aligned}a_0^2 =&\sigma_0^2\varepsilon_0^2 \in F_0,\\ \sigma_1^2 =&\alpha_0 + \alpha_1 a_0^2 \in F_0,\\ a_1^2 =&\sigma_0^2\varepsilon_1^2 \in F_1,\\ \sigma_2^2 =&\alpha_0 + \alpha_1 a_1^2 \in F_1,\end{aligned}$$

归纳可知

$$a_t \in F_t, \quad \sigma_t \in F_{t-1}.$$

已知 $\{a_t\}$ 是零均值不相关列。来计算 $\mathrm{Var}(a_t)$。

$$\begin{aligned}&\mathrm{Var}(a_t) = E(a_t^2) = E(\sigma_t^2\varepsilon_t^2)\\ =&E\{E(\sigma_t^2\varepsilon_t^2|F_{t-1})\} = E\{\sigma_t^2 E(\varepsilon_t^2|F_{t-1})\}\\ =&E\{\sigma_t^2 E(\varepsilon_t^2)\} = E(\sigma_t^2).\end{aligned}$$

注意

$$\begin{aligned}E(\sigma_t^2) =&\alpha_0 + \alpha_1 E(a_{t-1}^2)\\ =&\alpha_0 + \alpha_1 E(\sigma_{t-1}^2),\end{aligned}$$

为了 $E(\sigma_t^2)$ 不依赖于 t，必须且只需

$$E(\sigma_t^2) = \frac{\alpha_0}{1-\alpha_1},$$

这只要取 σ_0^2 等于此值即可。这时，

$$\mathrm{Var}(a_t) = E(\sigma_t^2) = \frac{\alpha_0}{1-\alpha_1},\ t = 0, 1, 2, \dots.$$

也可以取 σ_0^2 为期望等于此值的随机变量且与 F_t 独立（$\forall t \geq 0$）。

所以，适当条件下 ARCH(1) 中的 $\{a_t\}$ 是零均值白噪声列。此结论可以推广到 ARCH(p) 模型。

Chapter 18

GARCH 模型

本章来自 (Tsay, 2013)§4.6-4.8 内容。

ARCH 模型用来描述波动率能得到很好的效果，但实际建模时可能需要较高的阶数，比如 §17.5.3的欧元汇率波动率建模用了 11 阶的 ARCH 模型。考虑类似从 AR 推广到 ARMA 的模型变化。

18.1 模型方程

(Bollerslev, 1986) 提出了 ARCH 模型的一种重要推广模型，称为 GARCH 模型。对于一个对数收益率序列 r_t，令 $a_t = r_t - \mu_t = r_t - E(r_t|F_{t-1})$ 为其新息序列，称 $\{a_t\}$ 服从 GARCH(m,s) 模型，如果 a_t 满足

$$a_t = \sigma_t \varepsilon_t, \quad \sigma_t^2 = \alpha_0 + \sum_{i=1}^{m} \alpha_i a_{t-i}^2 + \sum_{j=1}^{s} \beta_j \sigma_{t-j}^2 \tag{18.1}$$

其中 $\{\varepsilon_t\}$ 为零均值单位方差的独立同分布白噪声列，$\alpha_0 > 0$, $\alpha_i \geq 0$, $\beta_j \geq 0$, $0 < \sum_{i=1}^{m} \alpha_i + \sum_{j=1}^{s} \beta_j < 1$，这最后一个条件用来保证满足模型的 a_t 的无条件方差有限且不变，而条件方差 σ_t^2 可以随时间 t 而变。

18.2 与 ARMA 模型比较

模型(18.1)表面上像是如下的 $x_t = \sigma_t^2$ 的类似于 ARMA(s,m) 的模型：

$$X_t^2 - \sum_{j=1}^{s} \beta_j X_{t-j}^2 = \alpha_0 + \sum_{i=1}^{m} \alpha_i a_{t-i}^2,$$

但是这里的 σ_{t-i}^2 和 a_{t-i}^2 是有关系的，σ_{t-i}^2 是 a_{t-i} 的条件方差，ARMA 模型中的 x_{t-i} 与模型中的白噪声 ε_{t-i} 并没有这样的关系。

为了利用 GARCH 模型与 ARMA 模型的相似性，令 $\alpha_i = 0$, 当 $i > m$；令 $\beta_j = 0$, 当 $j > s$。令 $\eta_t = a_t^2 - \sigma_t^2$，下一小节证明了 $Ea_t = 0$，而 $\sigma_t^2 = \mathrm{Var}(a_t|F_{t-1}) = E(a_t^2|F_{t-1})$, 当 a_t 为严平稳列时 η_t 是鞅差序列，这是比宽白噪声严一些，比零均值独立同分布白噪声宽一些的条件。将 $\sigma_{t-i}^2 = a_{t-i}^2 - \eta_{t-i}$ 代入模型(18.1)得

$$a_t^2 = \alpha_0 + \sum_{i=1}^{\max(m,s)} (\alpha_i + \beta_i) a_{t-i}^2 + \eta_t - \sum_{j=1}^{s} \beta_j \eta_{t-j}. \tag{18.2}$$

这就是关于 $\{a_t^2\}$ 的 ARMA($\max(m,s)$, s) 模型，由 ARMA 模型的无条件期望的公式得

$$Ea_t^2 = \frac{\alpha_0}{1-\sum_{i=1}^{\max(m,s)}(\alpha_i+\beta_i)} = \frac{\alpha_0}{1-\sum_{i=1}^{m}\alpha_i-\sum_{j=1}^{s}\beta_j}.$$

这要求分母为正，即要求 $\sum_{i=1}^{m}\alpha_i+\sum_{j=1}^{s}\beta_j<1$。这时 a_t 的无条件方差 $\mathrm{Var}(a_t)$ 也等于上式。

18.3 GARCH 模型的性质

下面以最简单的 GARCH(1,1) 为例研究 GARCH 模型的性质。令 F_{t-1} 表示截止到 $t-1$ 时刻的 a_{t-i} 和 σ_{t-j} 所包含的信息。模型为

$$a_t = \sigma_t\varepsilon_t, \quad \varepsilon_t \text{ i.i.d. } \mathrm{WN}(0,1), \tag{18.3}$$

$$\sigma_t^2 = \alpha_0+\alpha_1 a_{t-1}^2+\beta_1\sigma_{t-1}^2. \tag{18.4}$$

为了计算无条件均值 Ea_t，先计算条件期望

$$E(a_t|F_{t-1}) = E(\sigma_t\varepsilon_t|F_{t-1}) = \sigma_t E(\varepsilon_t|F_{t-1}) = 0.$$

这里用了 $\sigma_t\in F_{t-1}$ 而 ε_t 与 F_{t-1} 独立。于是

$$Ea_t = E[E(a_t|F_{t-1})] = 0.$$

即 GARCH 模型的新息 a_t 的无条件期望为零。

来计算 a_t 的无条件方差。设模型(18.1)的 $\{a_t\}$ 序列存在严平稳解，则

$$\begin{aligned}\mathrm{Var}(a_t) =& E(a_t^2) = E[E(a_t^2|F_{t-1})] = E[E(\sigma_t^2\varepsilon_t^2|F_{t-1})]\\ =& E[\sigma_t^2E(\varepsilon_t^2|F_{t-1})] = E[\sigma_t^2E(\varepsilon_t^2)]\\ =& E[\sigma_t^2] = E[\alpha_0+\alpha_1a_{t-1}^2+\beta_1\sigma_{t-1}^2]\\ =& \alpha_0+\alpha_1E(a_{t-1}^2)+\beta_1E[E(a_{t-1}^2|F_{t-2})]\\ =& \alpha_0+(\alpha_1+\beta_1)E(a_{t-1}^2).\end{aligned}$$

令 $Ea_t^2 = Ea_{t-1}^2$，解得

$$\mathrm{Var}(a_t) = Ea_t^2 = \frac{\alpha_0}{1-\alpha_1-\beta_1}.$$

GARCH(1,1) 模型的性质：

第一，像 ARCH 模型一样，a_t 存在波动率聚集，一个较大的 a_{t-1} 或 σ_{t-1} 使得 1 步以后的条件方差变大，从而倾向于出现较大的对数收益率。

第二，当 ε_t 为标准正态分布时，在如下条件下 a_t 有无条件四阶矩：

$$1-2\alpha_1^2-(\alpha_1+\beta_1)^2>0.$$

这时超额峰度为

$$\frac{Ea_t^4}{(Ea_t^2)^2}-3 = \frac{2\left[1-(\alpha_1+\beta_1)^2+\alpha_1^2\right]}{1-(\alpha_1+\beta_1)^2-2\alpha_1^2}>0.$$

即 a_t 分布厚尾。但是，对实际数据建模时即使使用条件 t 分布，对数据的厚尾性的拟合仍可能不足。

第三，GARCH 模型给出了一个比较简单的波动率模型。

第四，因为 σ_t^2 对 a_{t-i} 的依赖是通过 a_{t-i}^2，所以一个取正值的扰动 a_{t-i} 和一个取负值的 a_{t-i}，只要绝对值相等，对后续波动率的影响就是相等的，不能体现杠杆效应。

18.4 预测

可以用类似 ARMA 预测的方法预测波动率。仍以 GARCH(1,1) 为例，由模型(18.4)，基于截止到 h 时刻的观测作超前一步预测：

$$\sigma_{h+1}^2 = \alpha_0 + \alpha_1 a_h^2 + \beta_1 \sigma_h^2 \in F_h.$$

所以

$$\sigma_h^2(1) = E(\sigma_{h+1}^2|F_h) = \sigma_{h+1}^2 = \alpha_0 + \alpha_1 a_h^2 + \beta_1 \sigma_h^2. \tag{18.5}$$

对 σ_{h+2}^2，利用 $a_t^2 = \sigma_t^2 \varepsilon_t^2$，有

$$\begin{aligned}\sigma_{h+2}^2 =& \alpha_0 + \alpha_1 a_{h+1}^2 + \beta_1 \sigma_{h+1}^2 \\ =& \alpha_0 + \alpha_1 \sigma_{h+1}^2 \varepsilon_{h+1}^2 + \beta_1 \sigma_{h+1}^2 \\ =& \alpha_0 + (\alpha_1 \varepsilon_{h+1}^2 + \beta_1)\sigma_{h+1}^2\end{aligned}$$

于是

$$\begin{aligned}\sigma_h^2(2) =& E(\sigma_{h+2}^2|F_h) = \alpha_0 + E(\alpha_1 \varepsilon_{h+1}^2 + \beta_1|F_h)\sigma_{h+1}^2 \\ =& \alpha_0 + (\alpha_1 + \beta_1)\sigma_h^2(1).\end{aligned}$$

类似地，对 $\ell \geq 2$ 有

$$\sigma_{h+\ell}^2 = \alpha_0 + \alpha_1 \varepsilon_{h+\ell-1}^2 \sigma_{h+\ell-1}^2 + \beta_1 \sigma_{h+\ell-1}^2 = \alpha_0 + (\alpha_1 \varepsilon_{h+\ell-1}^2 + \beta_1)\sigma_{h+\ell-1}^2,$$

于是

$$\begin{aligned}\sigma_h^2(\ell) =& E\left\{\sigma_{h+\ell}^2|F_h\right\} = \alpha_0 + E\left\{(\alpha_1 \varepsilon_{h+\ell-1}^2 + \beta_1)\sigma_{h+\ell-1}^2|F_h\right\} && (18.6)\\ =& \alpha_0 + E\left\{E\left[(\alpha_1 \varepsilon_{h+\ell-1}^2 + \beta_1)\sigma_{h+\ell-1}^2|F_{h+\ell-2}\right]|F_h\right\} && (18.7)\\ =& \alpha_0 + E\left\{\sigma_{h+\ell-1}^2 E\left[\alpha_1 \varepsilon_{h+\ell-1}^2 + \beta_1|F_{h+\ell-2}\right]|F_h\right\} \quad (\text{注意}\sigma_{h+\ell-1}^2 \in F_{h+\ell-2}) && (18.8)\\ =& \alpha_0 + \left\{\sigma_{h+\ell-1}^2(\alpha_1 + \beta_1)|F_h\right\} && (18.9)\\ =& \alpha_0 + (\alpha_1 + \beta_1)\sigma_h^2(\ell - 1). && (18.10)\end{aligned}$$

预测公式与自回归系数为 $(\alpha_1 + \beta_1)$ 的 ARMA(1,1) 的超前预测公式相同。

从 $\ell = 2$ 迭代计算得

$$\sigma_h^2(\ell) = \frac{\alpha_0[1 - (\alpha_1 + \beta_1)^{\ell-1}]}{1 - (\alpha_1 + \beta_1)} + (\alpha_1 + \beta_1)^{(\ell-1)}\sigma_h^2(1).$$

只要 $\alpha_1 + \beta_1 < 1$ 就有

$$\sigma_h^2(\ell) \to \frac{\alpha_0}{1 - \alpha_1 - \beta_1} = \mathrm{Var}(a_t).$$

即超前多步条件方差预测趋于 a_t 的无条件方差。

18.5 模型估计

ARCH 模型的建模步骤也适用于 GARCH 模型的建模。GARCH 模型的定阶方法研究不多，一般用试错法尝试较低阶的 GARCH 模型，如 GARCH(1,1), GARCH(2,1), GARCH(1,2) 等。许多情况下 GARCH(1,1) 就能解决问题。

为了估计参数，可以假定初始的 σ_t^2 已知，递推计算后续的 σ_t^2 并计算条件似然函数，求条件似然函数的最大值点得到参数估计。有时用 a_t 的样本方差作为初始的 σ_t 的值。

为了检验模型的充分性，可以计算标准化残差

$$\tilde{a}_t = \frac{a_t}{\sigma_t},$$

通过对 $\tilde{a}_t$ 和 $\tilde{a}_t^2$ 的白噪声检验确认模型可以接受，还可以做 $\tilde{a}_t$ 相对于条件分布的 QQ 图以检验模型假设的条件分布的拟合优度。

18.5.1 Intel 公司股票收益率的波动率建模实例

继续使用 §17.5.1中的 Intel 公司股票从 1973-1 到 2009-12 的月度对数收益率数据，有 444 个观测值。§17.5.1中的 ARCH(1) 模型(17.19)在模型检验中有一些不足，比如关于标准化残差平方的 Ljung-Box 检验的滞后 15 和滞后 20 检验是显著的。尝试用 GARCH(1,1) 模型来改进。记 r_t 为对数收益率序列。

数据读入：

```
d.intel <- read_table(
  "m-intcsp7309.txt",
  col_types=cols(
    .default=col_double(),
    date=col_date("%Y%m%d")
  ))
xts.intel <- xts(
  log(1 + d.intel[["intc"]]), d.intel[["date"]]
)
tclass(xts.intel) <- "yearmon"
ts.intel <- ts(c(coredata(xts.intel)), start=c(1973,1), frequency=12)
at <- ts.intel - mean(ts.intel)
```

对数收益率的时间序列图：

```
plot(ts.intel, ylab="log return", main="Intel Stock Price Monthly Log Return")
```

18.5.2 使用条件正态分布

采用正态条件分布建立 GARCH(1,1) 模型：

```
library(fGarch, quietly = TRUE)
mod1 <- garchFit(~ 1 + garch(1,1), data=ts.intel, trace=FALSE)
summary(mod1)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = ts.intel, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
```

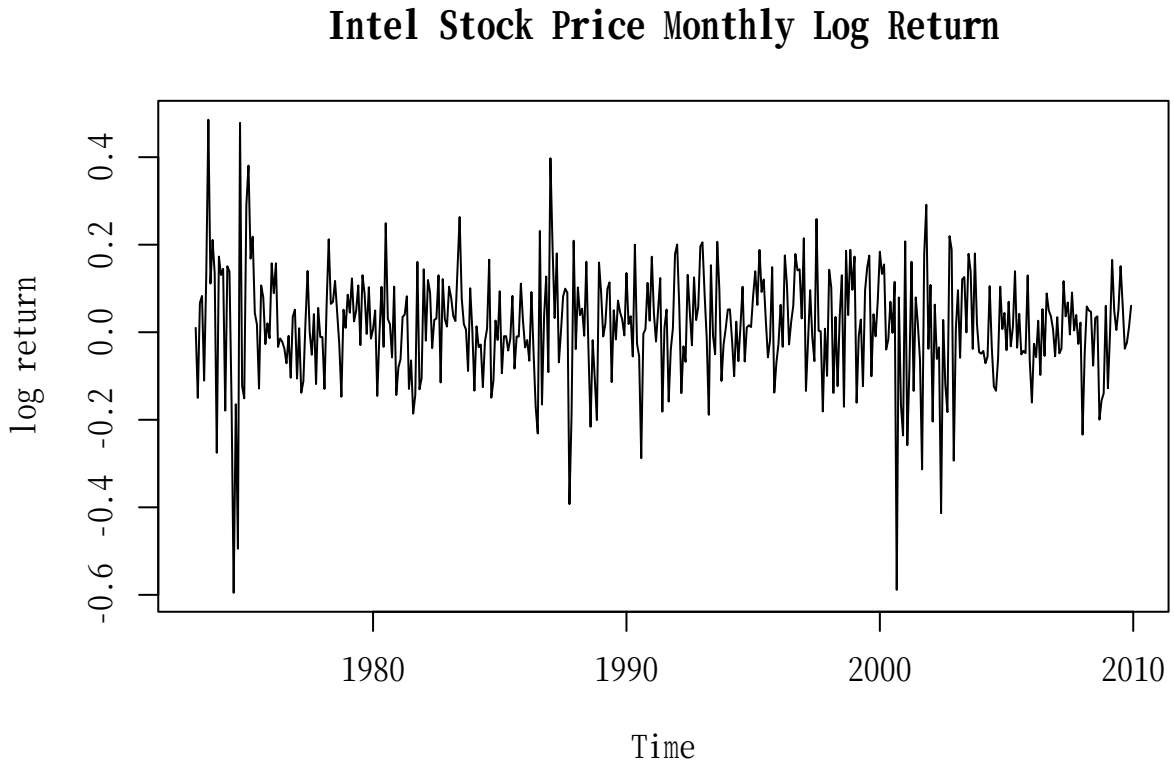


图 18.1: Intel 月对数收益率时间序列

```
## <environment: 0x000001f25d723a10>
##  [data = ts.intel]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1       beta1
## 0.01126568  0.00091902  0.08643831  0.85258554
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.0112657   0.0053931    2.089  0.03672 *
## omega  0.0009190   0.0003888    2.364  0.01808 *
## alpha1 0.0864383   0.0265439    3.256  0.00113 **
## beta1  0.8525855   0.0394322   21.622  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  312.3307    normalized:  0.7034475
##
## Description:
##  Tue May 30 17:03:47 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  174.904   0
##  Shapiro-Wilk Test  R    W      0.9709615 1.030281e-07
##  Ljung-Box Test     R    Q(10)  8.016844  0.6271916
##  Ljung-Box Test     R    Q(15)  15.5006   0.4159946
##  Ljung-Box Test     R    Q(20)  16.41549  0.6905368
##  Ljung-Box Test     R^2  Q(10)  0.8746345 0.9999072
##  Ljung-Box Test     R^2  Q(15)  11.35935  0.7267295
##  Ljung-Box Test     R^2  Q(20)  12.55994  0.8954573
##  LM Arch Test       R    TR^2   10.51401  0.5709617
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
```

```
## -1.388877 -1.351978 -1.389037 -1.374326
```

对标准化残差及其平方的白噪声检验结果都通过了。比 §17.5.1要好，而且 ACI 值 −1.3889 比(17.19)的 AIC 值 −1.3375 也更好。条件分布的正态性检验仍通不过。

模型可以写成：

$$r_t = 0.0113 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \epsilon_t \text{ i.i.d. } \sim \mathrm{N}(0,1) \tag{18.11}$$

$$\sigma_t^2 = 0.00092 + 0.0864 a_{t-1}^2 + 0.8526 \sigma_{t-1}^2 \tag{18.12}$$

σ_t^2 对过去的依赖主要来源于 $\beta_1 = 0.85$。

拟合的波动率图形：

```
##plot(mod1, which=2)
vola <- volatility(mod1)
plot(ts(vola, start=start(ts.intel), frequency=frequency(ts.intel)),
     xlab=" 年", ylab=" 波动率")
abline(h=sd(ts.intel), col="green")
```

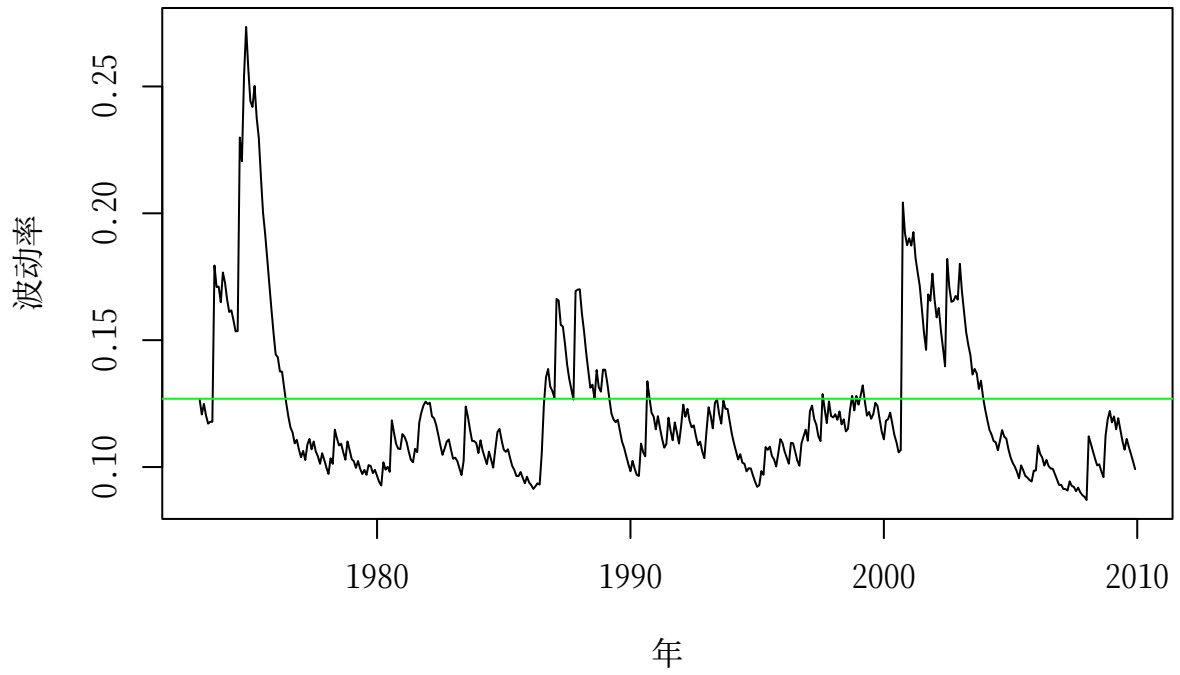


图 18.2: Intel 股票建模拟合波动率

可以看出波动率在 1973-1974 石油危机期间和 2000 年的互联网泡沫期间最高。波动率图形中绿色横线为样本标准差，值为：

```
sd(ts.intel)
```

```
## [1] 0.1269101
```

按模型计算的对数收益率无条件标准差为:

```
tmpx <- coef(mod1)
unname(sqrt(tmpx["omega"]/(1 - tmpx["alpha1"] - tmpx["beta1"])))
```

```
## [1] 0.122767
```

模型计算的无条件标准差与样本标准差很接近。

标准化残差的时间序列图:

```
##plot(mod1, which=9)
resi <- residuals(mod1, standardize=TRUE)
plot(ts(resi, start=start(ts.intel), frequency=frequency(ts.intel)),
     xlab=" 年", ylab=" 标准化残差")
```

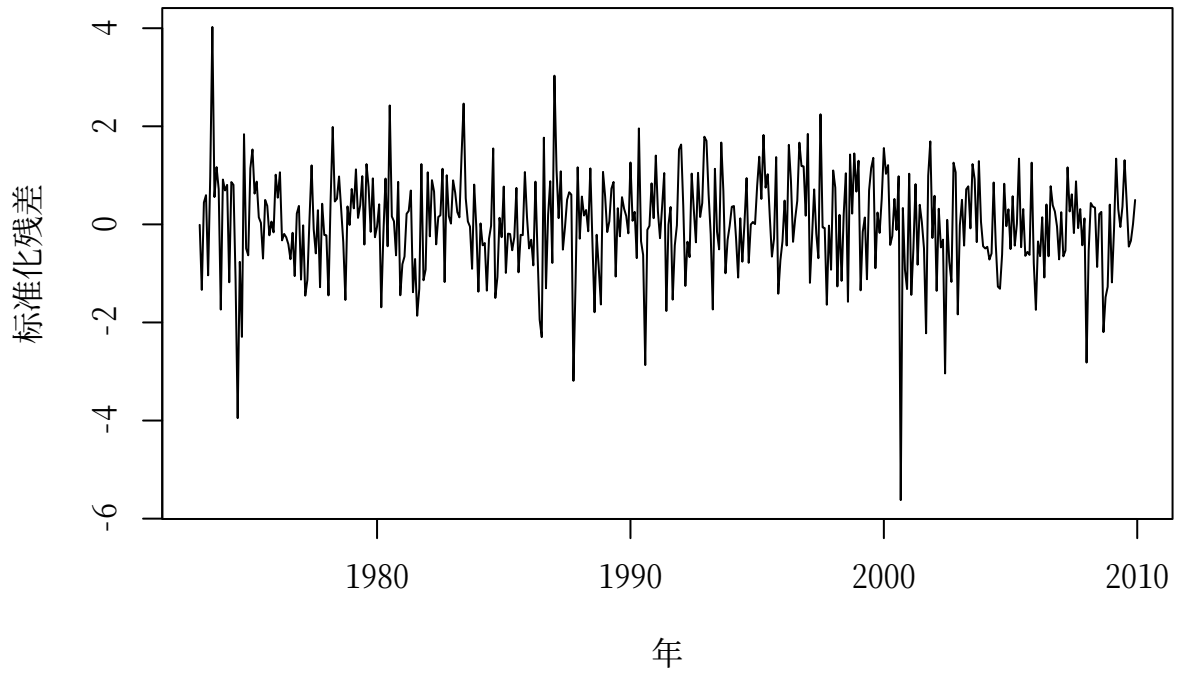


图 18.3: Intel 股票建模标准化残差

除了个别异常值，标准化残差看不出不符合独立同分布序列的特征。

$\tilde{a}_t$ 的 ACF:

```
plot(mod1, which=10)
```

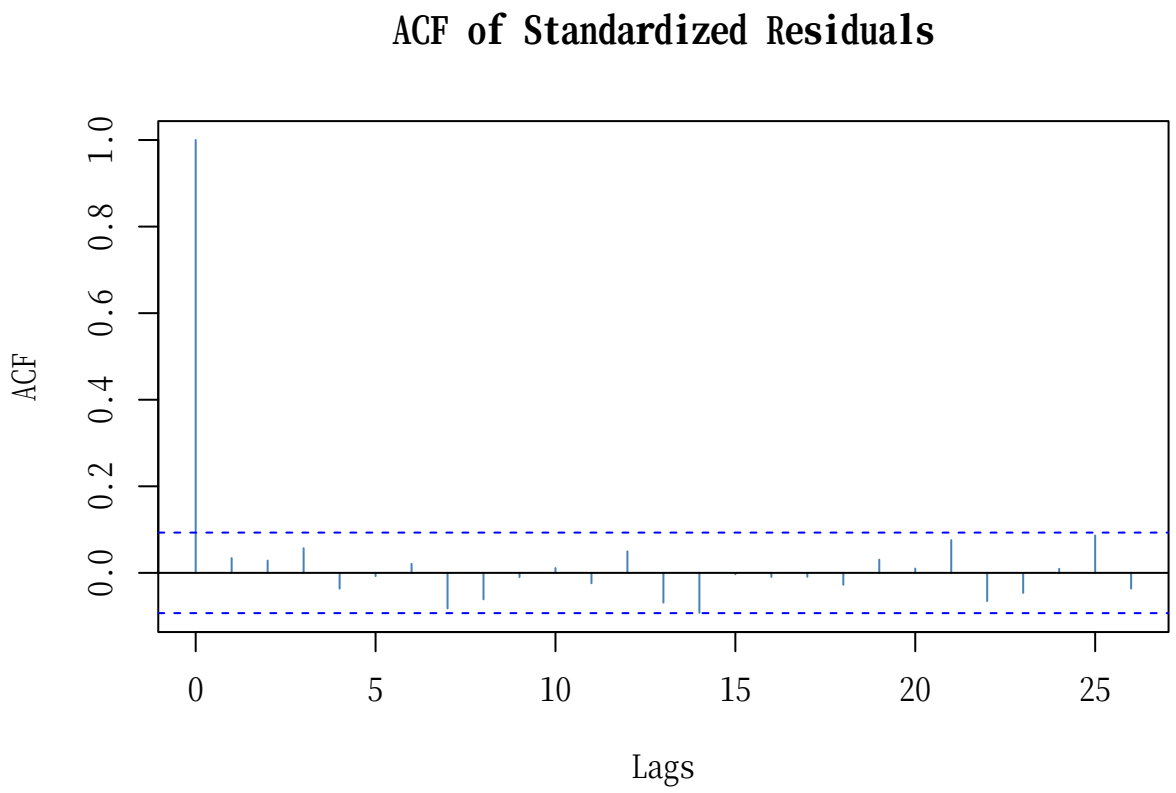


图 18.4: Intel 股票建模标准化残差的 ACF

$\tilde{a}_t^2$ 的 ACF:

```
plot(mod1, which=11)
```

$\tilde{a}_t^2$ 的 ACF 在滞后 12 处略超出。总的来说模型是满意的。

用 $\hat{\mu} + 2\hat{\sigma}_t$ 可以作为 r_t 的近似 95% 置信区间，将置信下限和置信下限分别延 t 轴方向连成曲线，得到如下图形：

```
plot(mod1, which=3)
```

可见对数收益率的取值基本都在预测区间之内。预测是滚动的超前一步预测。

上图直接提取数据也可以编程做出，如：

```
vola <- volatility(mod1)
hatmu <- coef(mod1)["mu"]
lb.intel <- hatmu - 2*vola
ub.intel <- hatmu + 2*vola
ylim <- range(c(ts.intel, lb.intel, ub.intel))
x.intel <- c(time(ts.intel))
plot(x.intel, c(ts.intel), type="l",
```

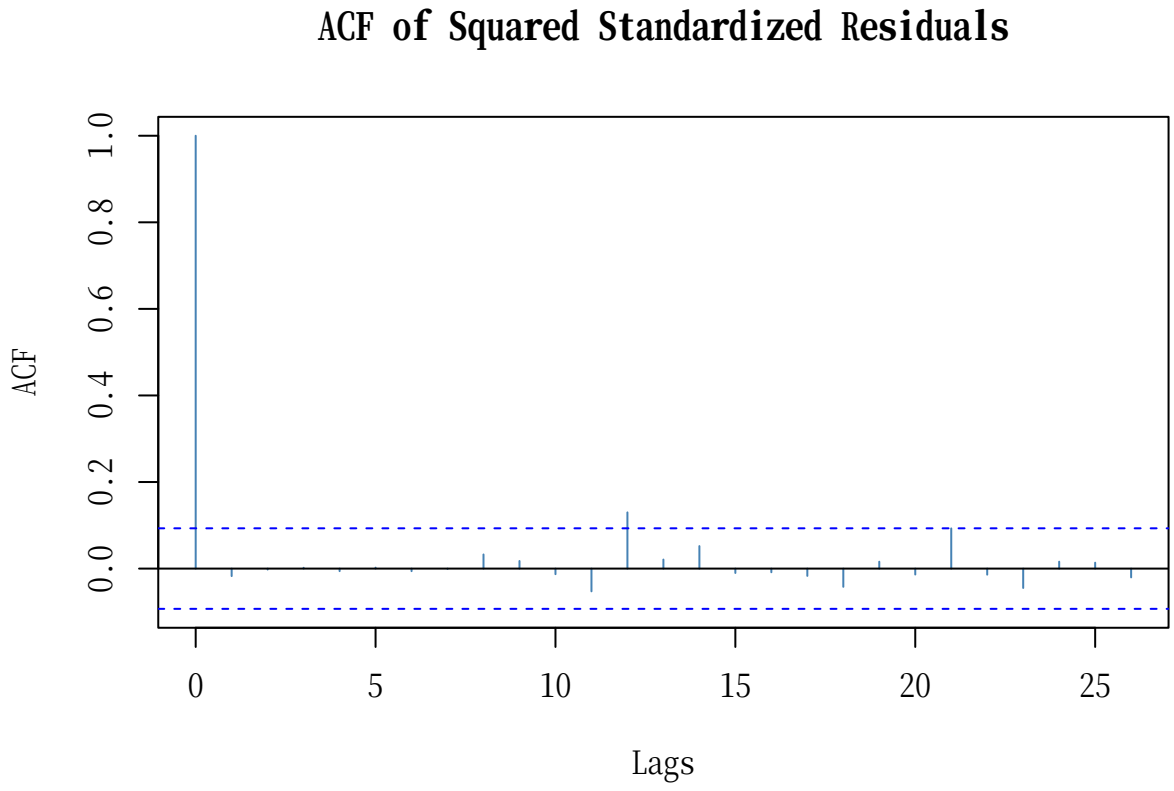


图 18.5: Intel 股票建模标准化残差平方的 ACF

图 18.6: Intel 股票对数收益率的逐点 95% 预测界限

```
     xlab=" 年", ylab=" 对数收益率")
lines(x.intel, c(lb.intel), col="red")
lines(x.intel, c(ub.intel), col="red")
```

tseries 包的 garch() 函数也可以用来拟合 GARCH 模型，只能使用条件正态分布。函数结果支持一系列的信息提取函数，如 summary()。

18.5.3 使用条件 t 分布

应用条件 t 分布的新息，拟合模型：

```
library(fGarch, quietly = TRUE)
mod2 <- garchFit(~ 1 + garch(1,1), data=ts.intel,
                 cond.dist="std", trace=FALSE)
summary(mod2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = ts.intel, cond.dist = "std",
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x000001f25c581f98>
##  [data = ts.intel]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##        mu      omega     alpha1      beta1      shape
## 0.0165076  0.0011576  0.1059029  0.8171298  6.7723926
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.0165076   0.0051031    3.235 0.001217 **
## omega  0.0011576   0.0005782    2.002 0.045287 *
## alpha1 0.1059029   0.0372046    2.846 0.004420 **
## beta1  0.8171298   0.0580150   14.085  < 2e-16 ***
```

```
## shape  6.7723926   1.8572644    3.646 0.000266 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  326.2264    normalized:  0.734744
##
## Description:
##  Tue May 30 17:03:47 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  203.4936  0
##  Shapiro-Wilk Test  R    W      0.9687607 3.970513e-08
##  Ljung-Box Test     R    Q(10)  7.877796  0.6407723
##  Ljung-Box Test     R    Q(15)  15.55225  0.4124162
##  Ljung-Box Test     R    Q(20)  16.5048   0.6848548
##  Ljung-Box Test     R^2  Q(10)  1.066053  0.9997694
##  Ljung-Box Test     R^2  Q(15)  11.49886  0.7164967
##  Ljung-Box Test     R^2  Q(20)  12.61507  0.8932826
##  LM Arch Test       R    TR^2   10.80749  0.5454849
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -1.446966 -1.400841 -1.447215 -1.428776
```

模型的检验也比较满意。模型可以写成：

$$r_t = 0.0165 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \epsilon_t \text{ i.i.d. } \sim t^*(6.77) \tag{18.13}$$

$$\sigma_t^2 = 0.00116 + 0.1059 a_{t-1}^2 + 0.8171 \sigma_{t-1}^2 \tag{18.14}$$

其中 t^* 表示标准化 t 分布。AIC 为 -1.447，正态分布时的 AIC 为 -1.389，t 分布结果的样本内比较占优。

模型(18.14)隐含的 a_t 的无条件标准差为

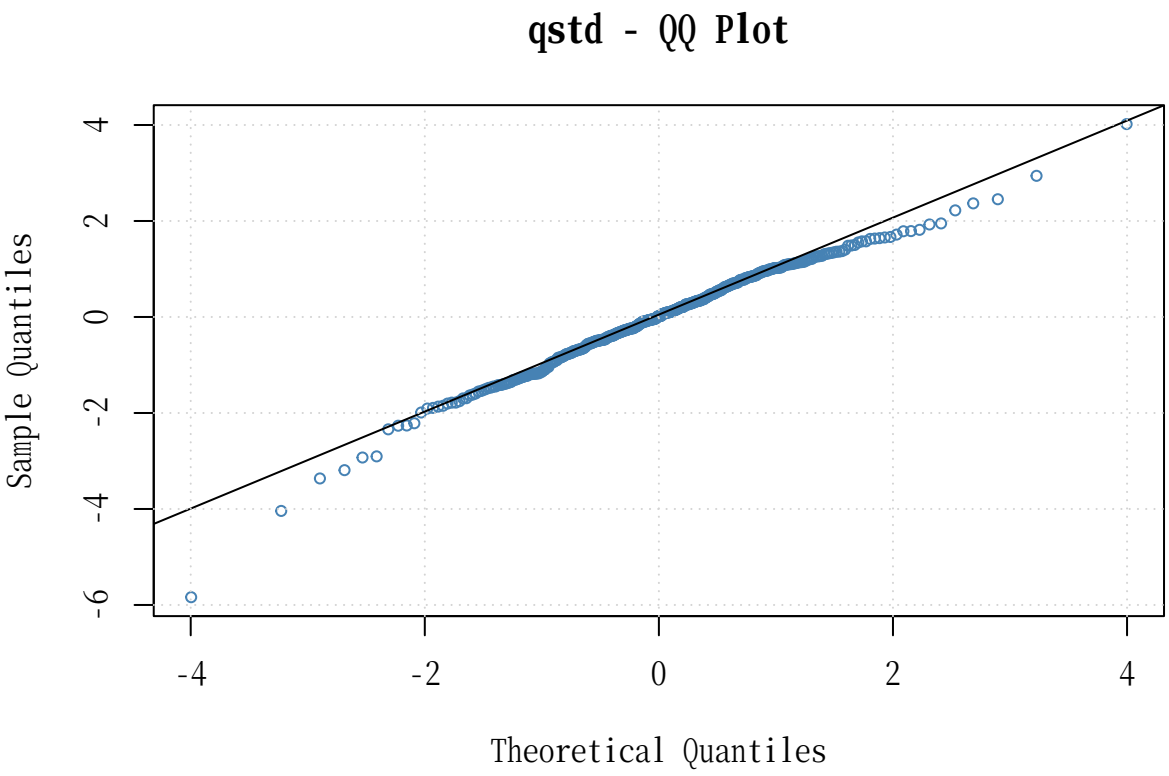
```
tmpx <- coef(mod2)
unname(sqrt(tmpx["omega"]/(1 - tmpx["alpha1"] - tmpx["beta1"])))
```

```
## [1] 0.1226396
```

即 0.1226，样本标准差为 0.1269，正态时模型隐含的无条件标准差为 0.1228，不同分布隐含的无条件标准差基本相同。

标准化残差相对于标准化 t 分布的 QQ 图：

```
plot(mod2, which=13)
```



可见分布拟合不够好，仍有重尾和左偏。

18.5.4 使用条件有偏 t 分布

对数收益率序列 r_t 的样本偏度为

```
skewness.test <- function(x, na.rm=TRUE){
  if(na.rm) x <- x[!is.na(x)]
  z <- (x - mean(x))/sd(x)
  n <- length(x)
  sk <- n/(n-1)/(n-2) * sum(z^3)
  t.sk <- sk / sqrt(6/n)
  p.sk <- 2*(1 - pnorm(abs(t.sk)))
  c(estimate=sk, statistic=t.sk, pvalue=p.sk)
}
skewness.test(as.vector(ts.intel))
```

```
##      estimate     statistic        pvalue
## -5.563719e-01 -4.786092e+00  1.700598e-06
```

偏度估计为 -0.56，偏度为零的零假设的检验 p 值很小，所以收益率分布是显著左偏（负偏）的。这样，ε_t 的分布最好采用左偏的分布。

在 fGarch 的 garchFit() 中指定 cond.dist="sstd"，则条件分布为有偏的标准化 t 分布：

```
library(fGarch, quietly = TRUE)
mod3 <- garchFit(~ 1 + garch(1,1), data=ts.intel,
  cond.dist="sstd", trace=FALSE)
summary(mod3)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = ts.intel, cond.dist = "sstd",
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x000001f25d0e0ad8>
##  [data = ts.intel]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##        mu      omega     alpha1      beta1       skew      shape
## 0.0133343  0.0011621  0.1049294  0.8177869  0.8717222  7.2344212
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.0133343   0.0053430    2.496 0.012572 *
## omega  0.0011621   0.0005587    2.080 0.037520 *
## alpha1 0.1049294   0.0358862    2.924 0.003456 **
## beta1  0.8177869   0.0559866   14.607  < 2e-16 ***
## skew   0.8717222   0.0629130   13.856  < 2e-16 ***
## shape  7.2344212   2.1018126    3.442 0.000577 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  328.0995    normalized:  0.7389628
##
```

```
## Description:
##  Tue May 30 17:03:47 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  195.2181  0
##  Shapiro-Wilk Test  R    W      0.9692506 4.892633e-08
##  Ljung-Box Test     R    Q(10)  7.882124  0.6403498
##  Ljung-Box Test     R    Q(15)  15.62496  0.4074053
##  Ljung-Box Test     R    Q(20)  16.57741  0.6802192
##  Ljung-Box Test     R^2  Q(10)  1.078434  0.9997569
##  Ljung-Box Test     R^2  Q(15)  11.95157  0.6826911
##  Ljung-Box Test     R^2  Q(20)  13.03793  0.8757507
##  LM Arch Test       R    TR^2   11.18828  0.5128558
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -1.450899 -1.395550 -1.451257 -1.429071
```

模型为：

$$r_t = 0.0133 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \epsilon_t \text{ i.i.d. } \sim t^*_{0.87,7.23} \tag{18.15}$$

$$\sigma_t^2 = 0.00116 + 0.1049 a_{t-1}^2 + 0.8178 \sigma_{t-1}^2 \tag{18.16}$$

其中 $t^*_{\xi,d}$ 表示标准化的有偏的 t 分布，偏度参数为 ξ，自由度为 d。

输出的检验结果表明模型充分。AIC 为 -1.451，相比对称的标准化 t 分布的 -1.447，和正态分布时的 AIC 为 -1.389，有偏 t 分布的 AIC 较优。

输出中偏度参数 ξ 的检验是 $H_0 : \xi = 0$ 的检验，但是我们需要的是 $H_0 : \xi = 1$ 的检验。计算 z 检验如下：

```
z <- (0.8717220 - 1) / 0.0629129
pv <- 2*(1 - pnorm(abs(z)))
c(statistic=z, pvalue=pv)
```

```
##   statistic      pvalue
## -2.03897770  0.04145225
```

可以认为偏度参数 ξ 是显著小于 1 的，模型中的条件分布是左偏的。也可以计算估计量加减两倍标准误差作为参数的近似 95% 置信区间，如

$$0.8717 \pm 2 \times 0.0629 \approx (0.745, 0.9975),$$

区间在 1 的左侧，说明在 0.05 水平下参数显著地小于 1。

作标准化残差相对于 $t^*(0.87, 7.23)$ 分布的 QQ 图：

```
plot(mod3, which=13)
```

从 QQ 图来看标准化残差比理论分布略重尾，但左偏已经不明显，比正态分布和标准化 t 分布拟合更好。

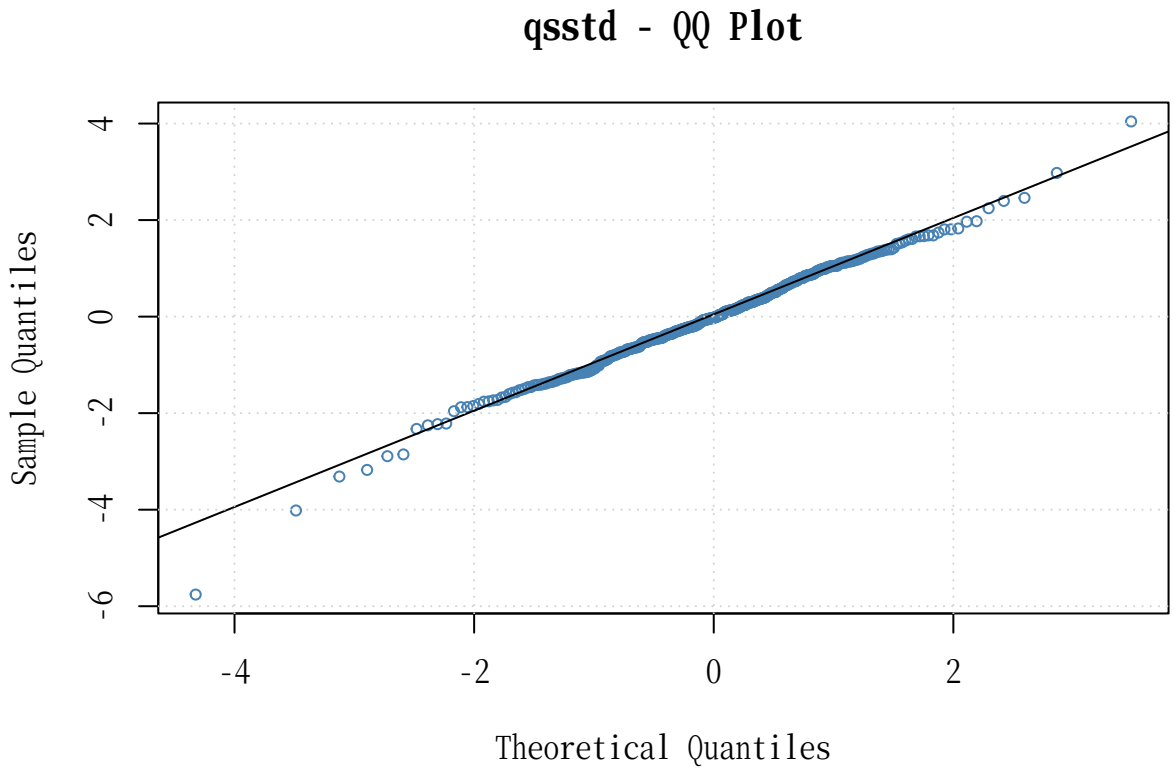


图 18.7: 标准化残差在有偏 t 分布假设时的 QQ 图

18.5.5 讨论和比较

对 Intel 的月对数收益率建模，同一采用了常数均值作为均值模型，GARCH(1,1) 作为波动率模型，但采用了三种不同的条件分布：

- 正态分布;
- t 分布；
- 有偏 t 分布。

从对 $\tilde{a}_t$ 和 $\tilde{a}_t^2$ 的白噪声检验来看三个模型都很好地拟合了数据。如果采用 AIC 来选择，那么有偏 t 分布较优。如果采用 BIC 来选择，则会选择对称 t 分布，实际上 BIC 倾向于参数较少的模型，而且有偏 t 分布拟合得到的参数 ξ 与 1（对称时）差距并不太大。这也说明不同的比较准则可能会选择不同的模型。

将三个模型估计的波动率绘制在同一坐标系中进行比较，可以看到三个模型拟合的波动率基本相同：

```
x.intel <- as.vector(time(ts.intel))
vola1 <- volatility(mod1)
vola2 <- volatility(mod2)
vola3 <- volatility(mod3)
matplot(x.intel, cbind(vola1, vola2, vola3),
        type="l",
        lty=1, col=c("green", "blue", "red"),
        xlab=" 年", ylab=" 波动率")
legend("top", lty=1, col=c("green", "blue", "red"),
       legend=c(" 标准正态分布", " 对称标准化 t 分布", " 有偏标准化 t 分布"))
```

下面比较三个模型的波动率超前 1 到 12 步的预测结果，预测基于截止到 2009 年 12 月的数据：

```
p1 <- predict(mod1, n.ahead=12)[["standardDeviation"]]
p2 <- predict(mod2, n.ahead=12)[["standardDeviation"]]
p3 <- predict(mod3, n.ahead=12)[["standardDeviation"]]
pred.tab <- tibble(
  " 预测步数"=1:12,
  " 正态"=p1,
  " 对称 t"=p2,
  " 有偏 t"=p3
)
knitr::kable(pred.tab, digits=4)
```

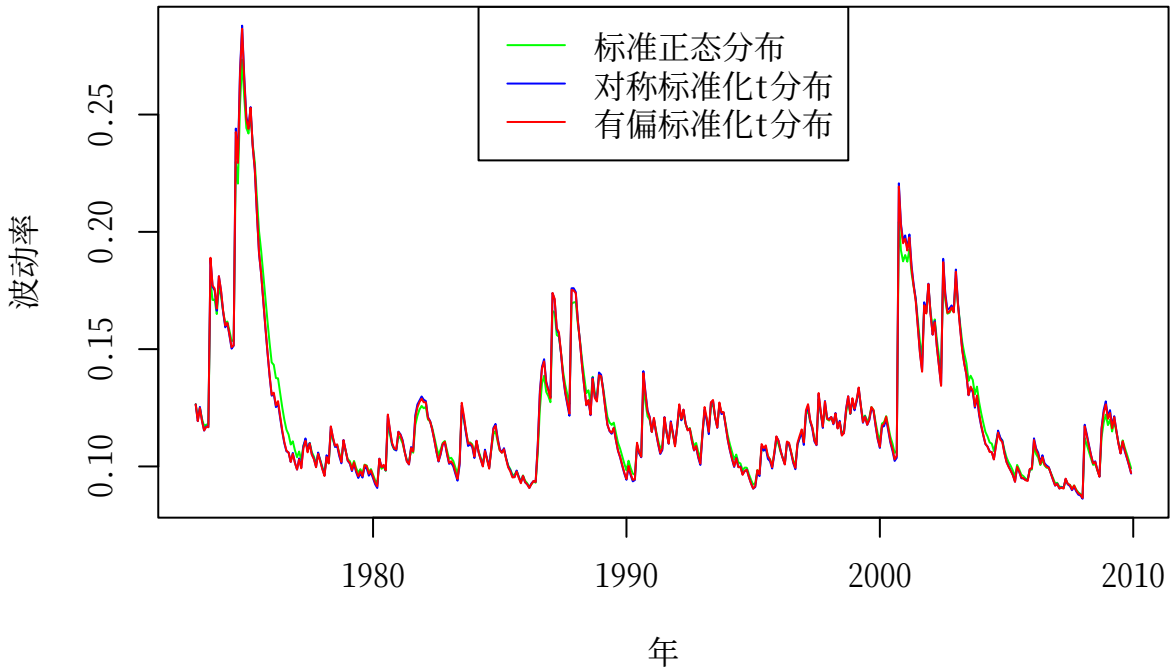


图 18.8: Intel 股票三种模型估计的波动率比较

预测步数	正态	对称 t	有偏 t
1	0.0975	0.0951	0.0955
2	0.0993	0.0975	0.0979
3	0.1009	0.0997	0.1000
4	0.1023	0.1016	0.1019
5	0.1037	0.1034	0.1037
6	0.1050	0.1050	0.1053
7	0.1061	0.1065	0.1067
8	0.1072	0.1078	0.1080
9	0.1082	0.1090	0.1092
10	0.1092	0.1101	0.1103
11	0.1100	0.1111	0.1113
12	0.1109	0.1121	0.1122

三个模型的波动率多步预报基本相同。

18.5.6 预测的评估

波动率的三种定义都不是直接可观测的，基本都来自模型估计。所以，比较不同波动率模型的预测结果具有挑战性。有人用样本外预测进行评估，比较波动率平方的预测值 $\sigma_h^2(\ell)$ 与同期的扰动值 $a_{h+\ell}^2$，但计算两者的相关系数一般是数值很小的。这是因为虽然 a_t^2 是 σ_t^2 的无偏估计，但仅用单个点作为估计误差太大，没有实际意义。有人用已实现波动率作为标准进行评估，这就需要将已实现波动率作为真实的波动率，也没有充分的依据。

评估的更多信息参见 Anderson 和 Bollerslev(1998)。

18.5.7 使用 rugarch 包

rugarch 扩展包提供了更多的 GARCH 变种（改进）模型的估计功能，我们用 rugarch 包估计 intel 股票月度对数收益率的 GARCH 模型，使用有偏 t 分布：

```
library(rugarch)
spec1 <- ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=TRUE  ),
  variance.model = list(
    model = "sGARCH", # standard GARCH model
    garchOrder = c(1,1) ),
  distribution.model="sstd" )
mod2ru <- ugarchfit(spec = spec1, data = ts.intel)
show(mod2ru)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
```

```
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.013349    0.005344   2.4980 0.012491
## omega   0.001161    0.000559   2.0783 0.037684
## alpha1  0.104594    0.035883   2.9149 0.003558
## beta1   0.817922    0.056054  14.5918 0.000000
## skew    0.871842    0.062835  13.8750 0.000000
## shape   7.249880    2.107754   3.4396 0.000583
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.013349    0.005724   2.3321 0.019694
## omega   0.001161    0.000508   2.2865 0.022227
## alpha1  0.104594    0.030754   3.4010 0.000671
## beta1   0.817922    0.048373  16.9088 0.000000
## skew    0.871842    0.061014  14.2891 0.000000
## shape   7.249880    1.994303   3.6353 0.000278
##
## LogLikelihood : 328.1036
##
## Information Criteria
## ------------------------------------
##
## Akaike       -1.4509
## Bayes        -1.3956
## Shibata      -1.4513
## Hannan-Quinn -1.4291
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.5396  0.4626
## Lag[2*(p+q)+(p+q)-1][2]    0.7088  0.6032
## Lag[4*(p+q)+(p+q)-1][5]    1.8503  0.6542
## d.o.f=0
```

```
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2528  0.6151
## Lag[2*(p+q)+(p+q)-1][5]    0.3020  0.9835
## Lag[4*(p+q)+(p+q)-1][9]    0.4598  0.9988
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]  0.005877 0.500 2.000  0.9389
## ARCH Lag[5]  0.038486 1.440 1.667  0.9966
## ARCH Lag[7]  0.068421 2.315 1.543  0.9997
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.337
## Individual Statistics:
## mu     0.15846
## omega  0.13798
## alpha1 0.10627
## beta1  0.10705
## skew   0.04290
## shape  0.06739
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.49 1.68 2.12
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias          0.17515 0.8610
## Negative Sign Bias 0.26118 0.7941
## Positive Sign Bias 0.07641 0.9391
## Joint Effect       0.09363 0.9926
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
```

```
##   group statistic p-value(g-1)
## 1    20     16.27       0.6392
## 2    30     29.78       0.4249
## 3    40     35.28       0.6403
## 4    50     52.17       0.3517
##
##
## Elapsed time : 0.1062911
```

结果与18.5.4中用 fGARCH 包计算的结果基本一致。`distribution.model` 的默认条件分布是`"norm"`，即条件正态分布。

结果中包含参数估计和标准误差估计、检验，以及稳健标准误差估计结果。各个信息准则 (Akaike, Bayes, Shibata, Hannan-Quinn) 的公式为：

$$
\begin{aligned}
\text{AIC} =& \frac{-2\ln L}{T} + \frac{2(m+s)}{T}, \\
\text{BIC} =& \frac{-2\ln L}{T} + \frac{(m+s)\ln T}{T}, \\
\text{SIC} =& \frac{-2\ln L}{T} + \ln\frac{T+2(m+s)}{T}, \\
\text{HQIC} =& \frac{-2\ln L}{T} + \frac{2(m+s)\ln\ln T}{T}.
\end{aligned}
$$

为了检验标准化残差和其平方是否白噪声，使用了 Fisher and Gallagher(2012) 提出的加权 Ljung-Box 检验法，这种方法考虑到了输入不是原始数据而是建模残差对统计量分布的影响。结果表明标准化残差和其平方都是白噪声。加权 ARCH LM 检验也是这样的对标准化残差进行的修改的 ARCH 效应检验。

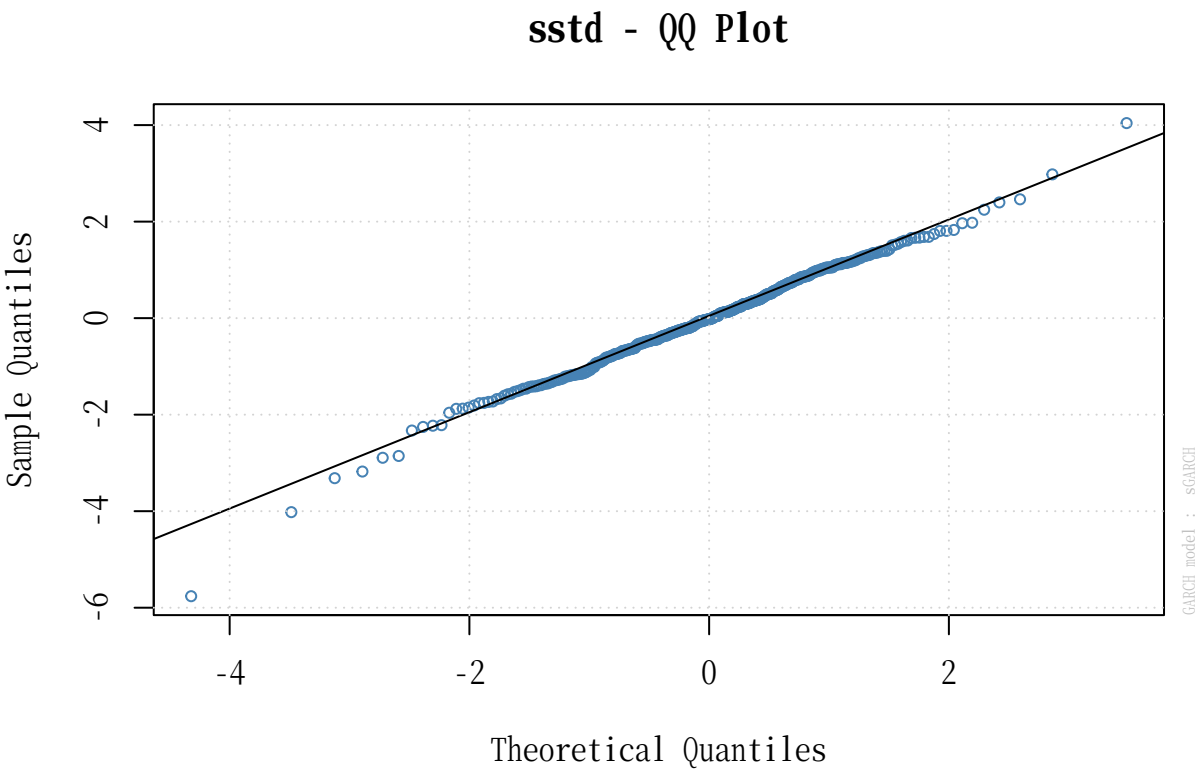
符号偏差检验 (Sign Bias Test) 来自 Engle and Ng(1993)，用于检验有没有杠杆效应。

修正的 Pearson 拟合优度检验用来检验标准化残差与假设的 ε_t 分布的一致性。这里的结果是拟合充分。

Nyblom 检验用来检验模型参数是否满足平稳性条件。联合检验的统计量 1.337，没有超过 5% 的临界值 1.68，所以结果不显著，可以认为模型参数满足平稳性条件。

rugarch 也可以用 `plot()` 函数对建模拟合结果制作各种诊断图形，如标准化残差相对于理论分布的 QQ 图：

```
plot(mod2ru, which=9)
```



支持的作图类型包括：

1. 输入序列叠加正负两倍波动率；
2. 输入序列叠加 1% 的 VaR 限；
3. 波动率与收益率绝对值的叠加图形；
4. 输入序列的 ACF 图；
5. 输入序列平方的 ACF 图；
6. 输入序列绝对值的 ACF 图；
7. 输入序列和平方序列的互相关函数图；
8. 标准化残差的直方图和参数密度估计图；
9. 标准化残差相对于假设的理论分布的 QQ 图；
10. 标准化残差的 ACF 图；
11. 标准化残差平方的 ACF 图；
12. 不同的新息符号对波动率影响的曲线图。

18.6 两步估计法

参考(18.2)对 ARMA 的模仿：

$$a_t^2 = \alpha_0 + \sum_{i=1}^{\max(m,s)} (\alpha_i + \beta_i) a_{t-i}^2 + \eta_t - \sum_{j=1}^{s} \beta_j \eta_{t-j},$$

提出如下的两步估计方法来估计 GARCH 模型。

第一，忽略 ARCH 效应，用线性时间序列建模方法（如最大似然估计）对收益率序列建立均值方程。残差序列用 a_t 表示。

第二，将 $\{a_t^2\}$ 作为观测序列，可以用最大似然估计方法估计式(18.2)中的参数。用 $\hat{\phi}_i$ 和 $\hat{\theta}_i$ 分别表示 AR 和 MA 部分的系数估计值。则 GARCH 模型参数估计为 $\hat{\beta}_i = -\hat{\theta}_i$, $\hat{\alpha}_i = \hat{\phi}_i + \hat{\theta}_i$。这种估计方法只是一种近似，没有理论结果证明其合理性，但是一些经验显示这样的估计往往能够给出不错的近似结果，尤其是大中样本情形。

例如，对 Intel 月对数收益率序列，先用常数作为均值方程，计算残差 a_t:

```
hatmu <- mean(ts.intel); hatmu
```

```
## [1] 0.0143273
```

```
at <- ts.intel - hatmu
at2 <- at^2
```

关于 a_t^2 用 ARMA(1,1) 估计：

```
mod4 <- arima(at2, order=c(1,0,1)); mod4
```

```
##
## Call:
## arima(x = at2, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.9119  -0.7915     0.0161
## s.e.  0.0430   0.0635     0.0039
##
## sigma^2 estimated as 0.001223:  log likelihood = 858.64,  aic = -1709.28
```

这里 $\hat{\phi}_1 = 0.9119$, $\hat{\theta}_1 = -0.7915$，所以 $\hat{\beta}_1 = 0.7915$, $\hat{\alpha}_1 = 0.1204$。

$$\hat{\alpha}_0 = \hat{\mu}(1 - \hat{\phi}_1) = 0.0161(1 - 0.9919) = 0.00013$$

(注意 `arima()` 函数输出的 `intecept` 是模型均值而不是 ϕ_0 参数)

这样的两步法得到的模型估计为：

$$r_t = 0.0143 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \sigma_t^2 = 0.00013 + 0.1204 a_{t-1}^2 + 0.7915 \sigma_{t-1}^2$$

与 `garchFit()` 得到的估计结果十分接近。

这样得到的误差波动率的估计为：

```
vola4 <- sqrt(at2 - mod4$residuals)
```

与有偏 t 分布的 GARCH 拟合的波动率计算相关系数：

```
cor(vola3, vola4)
```

```
## [1] 0.9976243
```

两者十分相似。

18.7 IGARCH 模型

GARCH 模型可以写成(18.2)这样的关于 a_t^2 的 ARMA 形式，其中 $\eta_t = a_t^2 - \sigma_t^2$ 是模型的扰动，是鞅差白噪声：

$$a_t^2 = \alpha_0 + \sum_{i=1}^{\max(m,s)} (\alpha_i + \beta_i) a_{t-i}^2 + \eta_t - \sum_{j=1}^{s} \beta_j \eta_{t-j}, \tag{18.17}$$

如果这个模型中的 AR 部分有单位根（有特征根等于 1），则对应的模型不再满足 GARCH 模型条件，称为 IGARCH 模型，或单位根 GARCH 模型。类似于 ARIMA 模型，IGARCH 模型的扰动 $\eta_t = a_t^2 - \sigma_t^2$ 对 a_t^2 的影响是持久的、不衰减的。

IGARCH(1,1) 模型可以写成

$$a_t = \sigma_t \varepsilon_t, \quad \sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + (1 - \alpha_1)\sigma_{t-1}^2.$$

Intel 股票的月对数收益率建立 IGARCH(1,1) 模型：

```
library(rugarch)
speci <- ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=TRUE  ),
  variance.model = list(
    model = "iGARCH", # itegrated GARCH model
    garchOrder = c(1,1) ) )
mod5ru <- ugarchfit(spec = speci, data = xts.intel)
show(mod5ru)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : iGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.009665    0.005340   1.8100 0.070298
## omega   0.000345    0.000182   1.8990 0.057561
## alpha1  0.127533    0.033626   3.7927 0.000149
## beta1   0.872467          NA       NA       NA
##
## Robust Standard Errors:
```

```
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.009665    0.006240   1.5488 0.121425
## omega   0.000345    0.000191   1.8060 0.070925
## alpha1  0.127533    0.031820   4.0079 0.000061
## beta1   0.872467          NA       NA       NA
##
## LogLikelihood : 308.1861
##
## Information Criteria
## ------------------------------------
##
## Akaike       -1.3747
## Bayes        -1.3470
## Shibata      -1.3748
## Hannan-Quinn -1.3638
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.6333  0.4261
## Lag[2*(p+q)+(p+q)-1][2]    0.8602  0.5459
## Lag[4*(p+q)+(p+q)-1][5]    1.8181  0.6620
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2326  0.6296
## Lag[2*(p+q)+(p+q)-1][5]    0.5702  0.9463
## Lag[4*(p+q)+(p+q)-1][9]    0.8666  0.9911
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1295 0.500 2.000  0.7190
## ARCH Lag[5]    0.2649 1.440 1.667  0.9495
## ARCH Lag[7]    0.3838 2.315 1.543  0.9877
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.1698
```

```
## Individual Statistics:
## mu     0.22485
## omega  0.03416
## alpha1 0.21842
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          0.846 1.01 1.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## -----------------------------------
##                     t-value   prob sig
## Sign Bias           0.24275 0.8083
## Negative Sign Bias 0.06684 0.9467
## Positive Sign Bias 0.25664 0.7976
## Joint Effect        0.10895 0.9907
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     27.71      0.08914
## 2    30     37.08      0.14416
## 3    40     57.98      0.02571
## 4    50     59.38      0.14717
##
##
## Elapsed time : 0.03541899
```

模型为

$$\begin{aligned} r_t =& 0.009665 + a_t,\ a_t = \sigma_t \varepsilon_t, \\ \sigma_t^2 =& 0.000345 + 0.1275 a_{t-1}^2 + 0.8725 \sigma_{t-1}^2. \end{aligned}$$

标准化残差及其平方的白噪声检验通过，残余 ARCH 效应检验通过。条件分布的拟合优度检验中有三个通过，有一个没有通过（水平 0.05）。

模型的 AIC$=-1.3747$，使用有偏 t 分布的标准 GARCH(1,1) 的 AIC$=-1.4509$，不建议使用 IGARCH。

18.8 GARCH-M 模型

有些金融资产的收益率的条件均值受到其波动率的影响，称为风险溢价。GARCH-M 模型就是用来描述这样的现象, M 表示条件均值依赖于 GARCH 模型。一种简单的 GARCH-M(1,1) 模型为

$$r_t = \mu + c\sigma_t^2 + a_t, \quad a_t = \sigma_t \varepsilon_t, \tag{18.18}$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \tag{18.19}$$

模型中的收益率条件均值为 $E(r_t|F_{t-1}) = \mu + c\sigma_t^2$ 需要用条件方差 $\sigma_t^2 = \text{Var}(r_t|F_{t-1})$ 描述。参数 c 称为**风险溢价参数**, 如果 c 为正值则收益率与波动率正相关。

文献中还有其他的风险溢价模型形式，如

$$r_t = \mu + c\sigma_t + a_t$$
$$r_t = \mu + c\ln\sigma_t^2 + a_t$$

式(18.19)中的收益率 r_t 不再是不相关列，而是序列相关的，相关性来自 σ_t^2 的序列相关性。风险溢价的存在是股票收益率具有序列相关性的原因之一。

18.8.1 Intel 股票月对数收益率的 GARCH-M 建模

用 rugarch 包估计 GARCH-M 模型：

```
library(rugarch)
specm <- ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=TRUE,
    archm=TRUE, archpow=2),
  variance.model = list(
    model = "sGARCH", # standard GARCH model
    garchOrder = c(1,1) ) )
mod6ru <- ugarchfit(spec = specm, data = xts.intel)
show(mod6ru)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu     -0.000181    0.013692 -0.013241 0.989436
## archm   0.855449    0.949115  0.901312 0.367422
## omega   0.000944    0.000392  2.409028 0.015995
## alpha1  0.087240    0.026657  3.272698 0.001065
## beta1   0.849721    0.039232 21.658619 0.000000
##
```

```
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     -0.000181    0.015260 -0.01188 0.990521
## archm   0.855449    1.097417  0.77951 0.435678
## omega   0.000944    0.000465  2.02972 0.042385
## alpha1  0.087240    0.029543  2.95293 0.003148
## beta1   0.849721    0.041844 20.30706 0.000000
##
## LogLikelihood : 312.7599
##
## Information Criteria
## ------------------------------------
##
## Akaike       -1.3863
## Bayes        -1.3402
## Shibata      -1.3866
## Hannan-Quinn -1.3681
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.6479  0.4209
## Lag[2*(p+q)+(p+q)-1][2]    0.8671  0.5434
## Lag[4*(p+q)+(p+q)-1][5]    2.1293  0.5882
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1373  0.7110
## Lag[2*(p+q)+(p+q)-1][5]    0.1496  0.9958
## Lag[4*(p+q)+(p+q)-1][9]    0.3173  0.9997
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3] 0.0002648 0.500 2.000  0.9870
## ARCH Lag[5] 0.0167638 1.440 1.667  0.9990
## ARCH Lag[7] 0.0530451 2.315 1.543  0.9999
##
## Nyblom stability test
```

```
## -----------------------------------
## Joint Statistic:  1.5298
## Individual Statistics:
## mu     0.15304
## archm  0.17085
## omega  0.07592
## alpha1 0.09511
## beta1  0.07486
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## -----------------------------------
##                    t-value   prob sig
## Sign Bias           0.2498 0.8028
## Negative Sign Bias  0.4723 0.6370
## Positive Sign Bias  0.1415 0.8875
## Joint Effect        0.2692 0.9657
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     17.35       0.5661
## 2    30     22.76       0.7876
## 3    40     30.77       0.8236
## 4    50     49.47       0.4544
##
##
## Elapsed time : 0.1170411
```

注意输入数据没有乘以 100。其中的 archm 项不显著，所以没有风险溢价效应。

18.8.2 标普 500 指数月超额收益率的 GARCH-M 建模

考虑标普 500 指数从 1926 年到 1991 年的月超额收益率。共 792 个观测。

```
x.sp <- scan("sp500.txt", quiet=TRUE)*100
ts.sp <- ts(x.sp, start=c(1926,1), frequency = 12)
```

```
plot(ts.sp, main=" 标普 500 的月超额收益率 (%)", xlab=" 年", ylab=" 超额收益率")
```

用 GARCH(1,1) 模型：

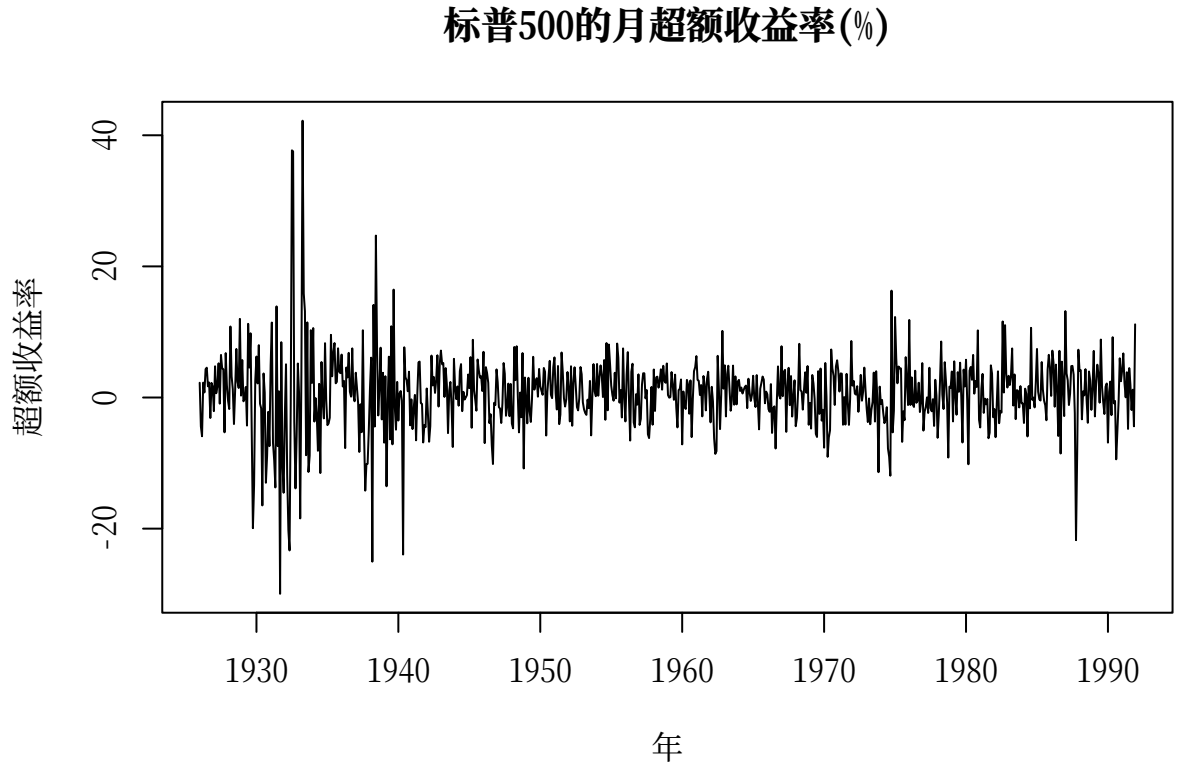


图 18.9: 标普 500 的月超额收益率 (%)

```
library(rugarch)
spec.sp1 <- ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=TRUE  ),
  variance.model = list(
    model = "sGARCH", # standard GARCH model
    garchOrder = c(1,1) ) )
mod7ru <- ugarchfit(spec = spec.sp1, data = ts.sp)
show(mod7ru)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.74496    0.153760   4.8449 0.000001
## omega    0.80400    0.284208   2.8289 0.004671
## alpha1   0.12226    0.022102   5.5315 0.000000
## beta1    0.85434    0.021813  39.1665 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.74496    0.171721   4.3382 0.000014
## omega    0.80400    0.336108   2.3921 0.016753
## alpha1   0.12226    0.028162   4.3413 0.000014
## beta1    0.85434    0.026423  32.3330 0.000000
##
## LogLikelihood : -2377.839
##
## Information Criteria
## ------------------------------------
##
## Akaike       6.0147
```

```
## Bayes        6.0384
## Shibata      6.0147
## Hannan-Quinn 6.0238
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.6270  0.4284
## Lag[2*(p+q)+(p+q)-1][2]    0.6322  0.6347
## Lag[4*(p+q)+(p+q)-1][5]    2.7320  0.4582
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.6221  0.4303
## Lag[2*(p+q)+(p+q)-1][5]    1.8668  0.6502
## Lag[4*(p+q)+(p+q)-1][9]    3.6708  0.6450
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.6535 0.500 2.000  0.4189
## ARCH Lag[5]    1.3390 1.440 1.667  0.6356
## ARCH Lag[7]    1.9550 2.315 1.543  0.7269
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.0291
## Individual Statistics:
## mu     0.08927
## omega  0.20235
## alpha1 0.27140
## beta1  0.16486
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.07 1.24 1.6
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
```

```
##                    t-value      prob sig
## Sign Bias           3.1027 0.0019860 ***
## Negative Sign Bias  0.9191 0.3583459
## Positive Sign Bias  0.7084 0.4788764
## Joint Effect       17.6089 0.0005296 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     30.73      0.04324
## 2    30     40.27      0.07958
## 3    40     48.40      0.14373
## 4    50     61.41      0.10989
##
##
## Elapsed time : 0.04562092
```

模型的标准化残差及其平方的白噪声检验通过，正态性假设中有一个没有通过。得到的模型为

$$\begin{aligned} 100r_t =& 0.7450 + a_t, \quad a_t = \sigma_t \varepsilon_t, \\ \sigma_t^2 =& 0.8040 + 0.1223 a_{t-1}^2 + 0.8543 \sigma_{t-1}^2. \end{aligned}$$

模型很接近 IGARCH 条件。

尝试对标普 500 超额收益率拟合 GARCH-M(1,1) 模型：

```
library(rugarch)
spec.sp2 <- ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=TRUE,
    archm=TRUE, archpow=2),
  variance.model = list(
    model = "sGARCH", # standard GARCH model
    garchOrder = c(1,1) ) )
mod8ru <- ugarchfit(spec = spec.sp2, data = ts.sp)
show(mod8ru)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
```

```
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.542048    0.236018   2.2966 0.021639
## archm   0.010081    0.008885   1.1346 0.256536
## omega   0.829648    0.293212   2.8295 0.004662
## alpha1  0.123124    0.022286   5.5247 0.000000
## beta1   0.852261    0.022400  38.0479 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.542048    0.243967   2.2218 0.026296
## archm   0.010081    0.008913   1.1311 0.258024
## omega   0.829648    0.347874   2.3849 0.017083
## alpha1  0.123124    0.027802   4.4286 0.000009
## beta1   0.852261    0.027061  31.4946 0.000000
##
## LogLikelihood : -2377.192
##
## Information Criteria
## ------------------------------------
##
## Akaike       6.0156
## Bayes        6.0451
## Shibata      6.0156
## Hannan-Quinn 6.0270
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.7246  0.3946
## Lag[2*(p+q)+(p+q)-1][2]    0.7501  0.5869
## Lag[4*(p+q)+(p+q)-1][5]    3.0243  0.4026
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.7589  0.3837
```

```
## Lag[2*(p+q)+(p+q)-1][5]    1.8690  0.6497
## Lag[4*(p+q)+(p+q)-1][9]    3.4842  0.6771
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.5048 0.500 2.000  0.4774
## ARCH Lag[5]    1.0892 1.440 1.667  0.7066
## ARCH Lag[7]    1.6862 2.315 1.543  0.7833
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.327
## Individual Statistics:
## mu     0.06463
## archm  0.03109
## omega  0.21419
## alpha1 0.26216
## beta1  0.16482
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias           3.5833 3.601e-04 ***
## Negative Sign Bias  1.1596 2.466e-01
## Positive Sign Bias  0.6211 5.347e-01
## Joint Effect       22.1805 5.983e-05 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     23.86      0.20163
## 2    30     41.11      0.06741
## 3    40     40.63      0.39861
## 4    50     56.86      0.20554
##
##
```

```
## Elapsed time : 0.1301072
```

拟合的模型为：

$$\begin{aligned}\mu_t =& 0.54 + 0.010\sigma_t^2 + a_t, \quad a_t = \sigma_t \varepsilon_t,\\ \sigma_t^2 =& 0.83 + 0.12 a_{t-1}^2 + 0.85\sigma_{t-1}^2.\end{aligned}$$

其中 $c = 0.010$ 不显著，说明标普 500 的月超额收益率不存在 GARCH-M 现象，没有风险溢价。从 AIC 比较来看，标准 GARCH 模型的 AIC 更低。

Chapter 19

改进的 GARCH 模型

本章讲 GARCH 模型的一些有针对性的改进。来自 (Tsay, 2013)§4.9-4.11 内容。

19.1 EGARCH 模型

19.1.1 模型

(Nelson, 1991) 提出的指数 GARCH(EGARCH) 模型允许正负资产收益率对波动率有不对称的影响。考虑如下变换

$$g(\varepsilon_t) = \alpha\varepsilon_t + \gamma\left[|\varepsilon_t| - E|\varepsilon_t|\right], \tag{19.1}$$

其中 α 和 γ 是实常数。$\{\varepsilon_t\}$ 和 $\{|\varepsilon_t| - E|\varepsilon_t|\}$ 都分别是零均值独立同分布白噪声，分布为连续分布。易见 $Eg(\varepsilon_t) = 0$。由下式可见 $g(\varepsilon_t)$ 的分布是非对称的：

$$g(\varepsilon_t) = \begin{cases} (\alpha+\gamma)\varepsilon_t - \gamma E|\varepsilon_t|, & \varepsilon_t \geq 0, \\ (\alpha-\gamma)\varepsilon_t - \gamma E|\varepsilon_t|, & \varepsilon_t < 0. \end{cases} \tag{19.2}$$

当 $\varepsilon_t \sim \mathrm{N}(0,1)$ 时 $E|\varepsilon_t| = \sqrt{\frac{2}{\pi}}$。对式(17.10)中的标准 t 分布，有

$$E|\varepsilon_t| = \frac{2\sqrt{v-2}\Gamma(\frac{v+1}{2})}{(v-1)\Gamma(\frac{v}{2})\sqrt{\pi}}.$$

EGARCH(m,s) 模型可以用滞后算子的形式写成

$$a_t = \sigma_t\varepsilon_t, \quad \ln\sigma_t^2 = \alpha_0' + \frac{1+\alpha_2 B + \cdots + \alpha_m B^{m-1}}{1-\beta_1 B - \cdots - \beta_s B^s} g(\varepsilon_{t-1}) \tag{19.3}$$

α_0' 为常数，其中 B 是滞后算子，多项式 $1+\alpha_2 z + \cdots + \alpha_m z^{m-1}$ 和 $1-\beta_1 z - \cdots - \beta_m z^m$ 的根都在单位圆外且两个多项式没有公因子。

注意这里的模型阶相当于 GARCH(m,s)。

记 $\xi_t = \ln\sigma_t^2$，则(19.3)给出的 ξ_t 为一个平稳线性 ARMA$(s, m-1)$ 序列，以零均值独立同分布白噪声 $g(\varepsilon_{t-1})$ 为新息；但是，$\ln\sigma_t^2$ 通过 $\varepsilon_{t-j} = a_{t-j}/\sigma_{t-j}$ 对 $\{a_t\}$ 序列依赖。原始的 GARCH 模型的 σ_t^2 的方程是直接依赖于 a_{t-j}^2 的，$\pm a_{t-j}$ 对 σ_t^2 影响相同。

易见 $E\ln\sigma_t^2=\alpha_0'$。

更一般地，可以令 $g(\cdot)$ 中的 γ 随滞后 j 变化，模型变成：

$$a_t=\sigma_t\varepsilon_t, \tag{19.4}$$

$$\ln\sigma_t^2=\alpha_0+\sum_{j=1}^{m}\left[\alpha_j\varepsilon_{t-j}+\gamma_j(|\varepsilon_{t-j}|-E|\varepsilon_{t-j}|)\right]+\sum_{i=1}^{s}\beta_i\ln\sigma_{t-i}^2. \tag{19.5}$$

在(19.5)中，α_j 代表了对数收益率的正负扰动对波动率的不同影响，如果 $\alpha_j=0$，则正负扰动对波动率影响就相同了。

EGARCH 与 GARCH 模型的区别还有：

(1) 使用条件方差的对数建模，因为对数值可正可负，这就取消了 GARCH 模型对系数必须非负的限制。
(2) $g(\varepsilon_{t-j})=g(a_{t-j}/\sigma_{t-j})$ 的使用使得波动率对 a_{t-j} 的依赖关系与 a_{t-j} 的正负号有关，可以用来描述正负收益率对波动率的不同的影响，即杠杆效应。

19.1.2 EGARCH(1,1) 模型

以最简单的 EGARCH(1,1) 为例来讨论。模型为

$$a_t=\sigma_t\varepsilon_t, \tag{19.6}$$

$$\ln\sigma_t^2=\alpha_0+[\alpha_1\varepsilon_{t-1}+\gamma_1(|\varepsilon_{t-1}|-E|\varepsilon_{t-1}|)]+\beta_1\ln\sigma_{t-1}^2. \tag{19.7}$$

其中 ε_t iid N(0,1), 这个模型的阶类似 GARCH(1,1)，模型实际上是关于 $\ln\sigma_t^2$ 的一个 AR(1) 模型。

与一个 GARCH(1,1) 模型对比：

$$\sigma_t^2=\alpha_0+\alpha_1a_{t-1}^2+\beta_1\sigma_{t-1}^2$$

所以 EGARCH 模型用 $\ln\sigma_t^2$ 替换了 σ_t^2，用 $g(\varepsilon_{t-1})$ 替换了 $\alpha_1a_{t-1}^2$，用 $\ln\sigma_{t-1}^2$ 替换了 σ_{t-1}^2。注意 $g(\varepsilon_{t-1})=g(a_{t-1}/\sigma_{t-1})$。

(19.7)中，设 $\alpha_1<0$，$\varepsilon_{t-1}\geq 0$ 时，$\ln\sigma_t^2$ 变小，$\varepsilon_{t-1}<0$ 时，$\ln\sigma_t^2$ 变大，对于相同的 $|a_{t-1}|$, 负值比正值产生更大的 $\ln\sigma_t^2$。

波动率方程可以写成

$$\sigma_t^2=\sigma_{t-1}^{2\beta_1}e^{\alpha_0''}\cdot\begin{cases}\exp\left[(\gamma_1+\alpha_1)\frac{|a_{t-1}|}{\sigma_{t-1}}\right], & a_{t-1}\geq 0\\ \exp\left[(\gamma_1-\alpha_1)\frac{|a_{t-1}|}{\sigma_{t-1}}\right], & a_{t-1}<0\end{cases}$$

其中 $\alpha_0''=\alpha_0-\gamma_1E|\varepsilon_{t-1}|$。

系数 $\gamma_1+\alpha_1$ 和 $\gamma_1-\alpha_1$ 表明当 $\alpha_1\neq 0$ 时模型对正的和负的 a_{t-1} 的响应是非对称的，由于一般负的扰动造成的波动更大，所以 α_1 应该是负的才比较符合实际数据。

19.1.3 实例 1：CRSP 价值加权指数超额收益率

例子来自 (Nelson, 1991) 的文章。考虑 CRSP 价值加权指数的超额收益率日数据，从 1962 年 7 月到 1987 年 12 月，共 6408 个观测值，超额收益率是价值加权指数的收益率减去国债的月收益率，假定一个月每天的国债收益率相等。

用 r_t 表示超额收益率，所用的模型为

$$\begin{aligned} r_t =& \phi_0 + \phi_1 r_{t-1} + c\sigma_t^2 + a_t, \quad a_t = \sigma_t \varepsilon_t, \\ \ln \sigma_t^2 =& \alpha_0 + \ln(1+\omega N_t) + \frac{1+\beta B}{1-\alpha_1 B - \alpha_2 B^2} g(\varepsilon_{t-1}), \\ g(\varepsilon_{t-1}) =& \theta \varepsilon_{t-1} + \gamma(|\varepsilon_{t-1}| - E|\varepsilon_{t-1}|). \end{aligned}$$

均值方程是带有回归自变量的 AR(1) 模型，回归自变量是波动率平方，这个回归自变量代表了风险溢价，c 是风险溢价参数。波动率方程中的回归自变量 N_t 是第 $t-1$ 交易日到第 t 交易日之间没有交易的天数，这一项解释了放假休市若干天以后增加的波动。ε_t 使用广义误差分布 (GED), 波动率方程是是 EGARCH(2,2) 模型。这里所用的符号与19.1.1有所不同。θ 代表了杠杆效应。

(Nelson, 1991) 论文中参数估计值与标准误差列表如下：

参数	估计值	标准误差
α_0	-10.06	0.346
ω	0.183	0.028
γ	0.156	0.023
α_1	1.929	0.015
α_2	-0.929	0.015
β	-0.978	0.006
θ	-0.118	0.009
ϕ_0	3.5×10^{-4}	9.9×10^{-5}
ϕ_1	0.205	0.012
c	-3.361	2.026
v	1.576	0.032

估计中只有风险溢价参数 c 是不显著，其它参数，包括代表了不对称性的 θ，都是显著的，θ 估计为负，这表明模型的波动率收到负的扰动影响更大。

19.1.4 实例 2：IBM 股票月对数收益率

考虑 IBM 股票月对数收益率建模，从 1967 年 1 月到 2009 年 12 月，561 个观测值。建立 EGARCH(1,1) 模型，使用(19.7)的模型形式，模型为

$$\begin{aligned} r_t =& \mu + a_t, \\ a_t =& \sigma_t \varepsilon_t, \\ \ln \sigma_t^2 =& \alpha_0 + [\alpha_1 \varepsilon_{t-1} + \gamma_1(|\varepsilon_{t-1}| - E|\varepsilon_{t-1}|)] + \beta_1 \ln \sigma_{t-1}^2. \end{aligned}$$

读入数据：

```
da <- read_table(
  "m-ibmsp6709.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
xts.ibm <- xts(log(1 + da[["ibm"]]), da[["date"]])
tclass(xts.ibm) <- "yearmon"
ts.ibm <- ts(log(1 + da[["ibm"]]), start=c(1967,1), frequency=12)
```

对数收益率的时间序列图：

```
plot(xts.ibm,
  format.labels="%Y",
  xlab="Year", ylab="Log return")
```

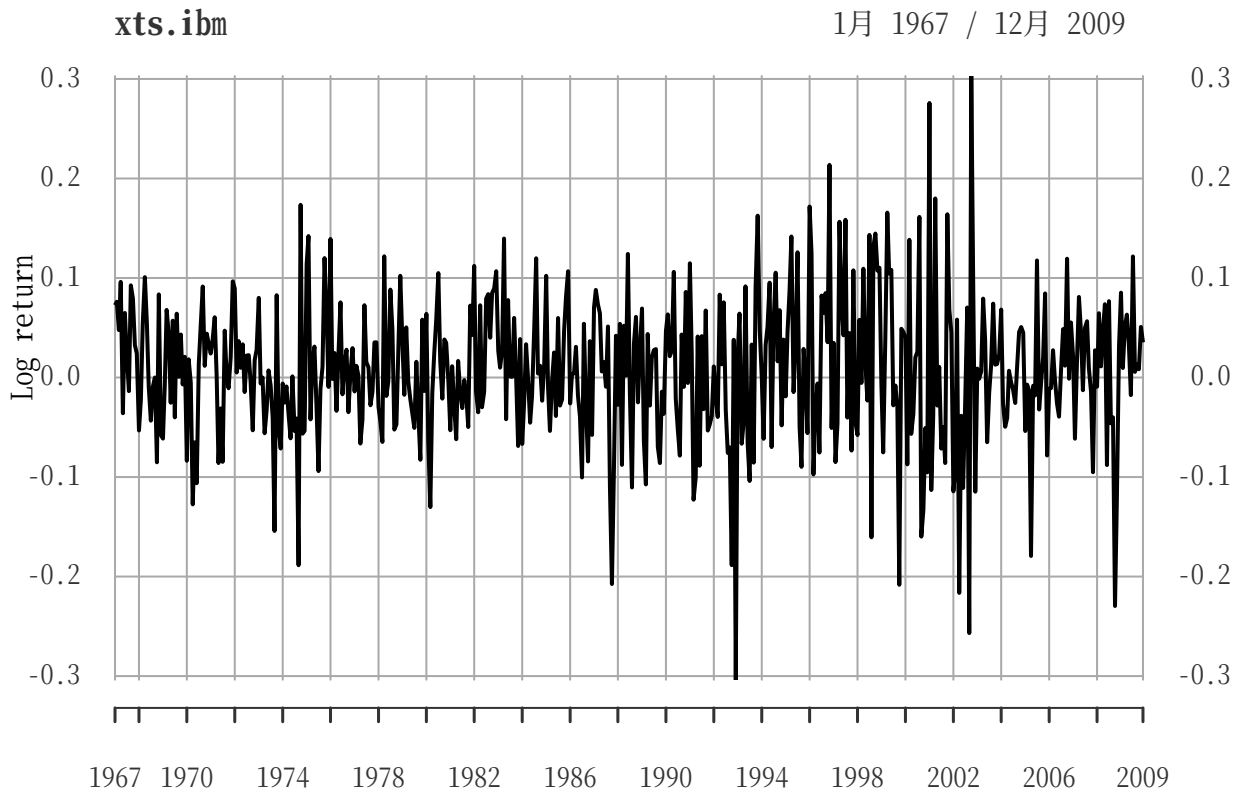


图 19.1: IBM 股票月对数收益率

ACF 图：

```
forecast::Acf(xts.ibm, lag.max = 36, main="")
```

从对数收益率的 ACF 来看基本是白噪声。作 Ljung-Box 白噪声检验：

```
Box.test(xts.ibm, lag=12, type="Ljung")
```

```
##
```

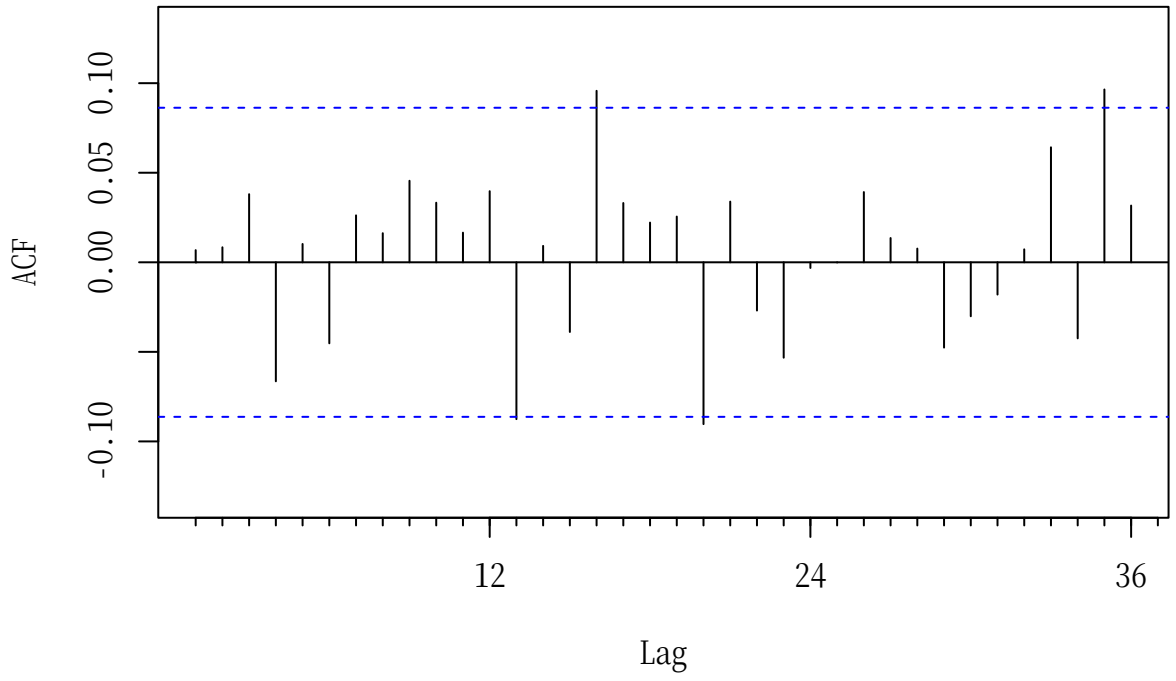


图 19.2: IBM 股票月对数收益率 ACF

```
##  Box-Ljung test
##
## data:  xts.ibm
## X-squared = 7.4042, df = 12, p-value = 0.8298
```

基本可以认为是（宽）白噪声。

a_t^2 的 ACF：

```
forecast::Acf(xts.ibm^2, lag.max = 36, main="")
```

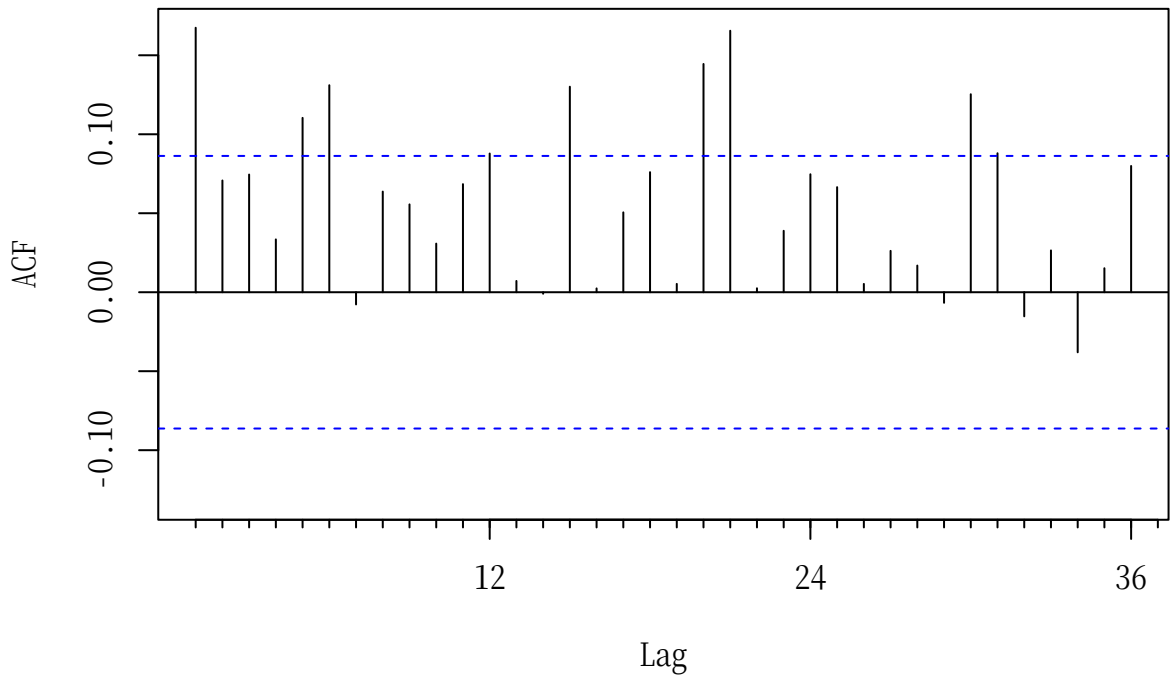


图 19.3: IBM 股票月对数收益率平方 ACF

这表明有轻微的 ARCH 效应。

用 rugarch 扩展包可以估计 EGARCH 模型，程序如：

```
library(rugarch)
spec1 <- ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=TRUE  ),
  variance.model = list(
    model = "eGARCH", # exponential GARCH model
    garchOrder = c(1,1) ) )
```

```
mod1ru <- ugarchfit(spec = spec1, data = xts.ibm)
show(mod1ru)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.006649    0.002963   2.2442 0.024820
## omega  -0.423208    0.223673  -1.8921 0.058480
## alpha1 -0.094813    0.039373  -2.4081 0.016037
## beta1   0.920485    0.041729  22.0586 0.000000
## gamma1  0.218711    0.060802   3.5971 0.000322
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.006649    0.003073   2.1635 0.030502
## omega  -0.423208    0.308230  -1.3730 0.169743
## alpha1 -0.094813    0.051835  -1.8291 0.067382
## beta1   0.920485    0.057270  16.0728 0.000000
## gamma1  0.218711    0.061770   3.5407 0.000399
##
## LogLikelihood : 651.634
##
## Information Criteria
## ------------------------------------
##
## Akaike       -2.5063
## Bayes        -2.4652
## Shibata      -2.5065
## Hannan-Quinn -2.4902
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
```

```
##                           statistic p-value
## Lag[1]                        1.237  0.2661
## Lag[2*(p+q)+(p+q)-1][2]     1.344  0.3989
## Lag[4*(p+q)+(p+q)-1][5]     1.867  0.6501
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                           statistic p-value
## Lag[1]                     0.009633  0.9218
## Lag[2*(p+q)+(p+q)-1][5]  1.087446  0.8396
## Lag[4*(p+q)+(p+q)-1][9]  2.511467  0.8360
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.09128 0.500 2.000  0.7626
## ARCH Lag[5]   1.10479 1.440 1.667  0.7021
## ARCH Lag[7]   2.20197 2.315 1.543  0.6746
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.1719
## Individual Statistics:
## mu     0.21948
## omega  0.61756
## alpha1 0.15868
## beta1  0.61824
## gamma1 0.06386
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           0.1014 0.9192
## Negative Sign Bias  0.2560 0.7980
## Positive Sign Bias  0.1888 0.8503
## Joint Effect        0.3726 0.9458
```

```
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     13.07       0.8350
## 2    30     22.26       0.8094
## 3    40     28.03       0.9040
## 4    50     42.53       0.7314
##
##
## Elapsed time : 0.08021402
```

估计的模型可以写成：

$$
\begin{aligned}
r_t =& 0.0066 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \varepsilon_t \text{ iid } \mathrm{N}(0,1) \\
\ln \sigma_t^2 =& -0.4232 - 0.0948\varepsilon_{t-1} + 0.2187(|\varepsilon_{t-1}| - E|\varepsilon_{t-1}|) + 0.9205 \ln \sigma_{t-1}^2.
\end{aligned}
$$

这种模型是关于 $\{\ln \sigma_t^2\}$ 的一个 AR(1) 模型。输出中参数 alpha1 代表了正负收益率对波动率不对称影响的杠杆效应，在 0.05 水平下这一项显著，说明杠杆效存在。

取 $\varepsilon_{t-1} = \pm 2$ 时，不同正负号的 ε_{t-1} 引起的 σ_t^2 的变化比值为

$$
e^{4 \times 0.0948} = 1.46,
$$

所以这时幅度相等的负的扰动比正的扰动额外增加 46% 的波动率平方。幅度越大，额外增加的波动率也越大。

计算标准化残差 $\tilde{a}_t = a_t/\sigma_t$ 和波动率 σ_t：

```
stdresi <- residuals(mod1ru, standardize=TRUE)
vol <- sigma(mod1ru)
```

标准化残差图：

```
plot(stdresi, format.labels="%Y",
  xlab="Year", ylab="",
  main="IBM 股票收益率模型标准化残差")
```

估计的波动率：

```
plot(vol, format.labels="%Y",
  xlab="Year", ylab="",
  main="IBM 股票收益率模型波动率")
```

标准化残差的 ACF:

```
forecast::Acf(stdresi, lag.max = 36, main="")
```

```
## 或:
## plot(mod1ru, which=10)
```

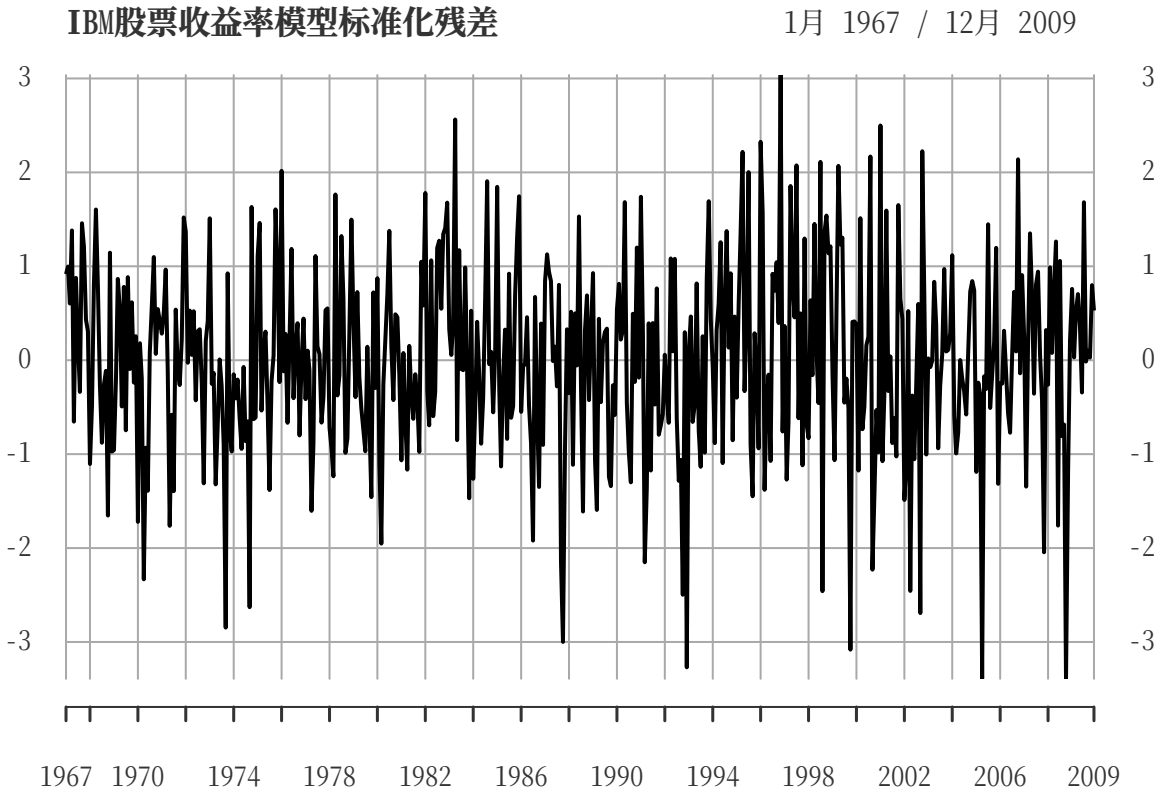


图 19.4: IBM 股票收益率模型标准化残差

图 19.5: IBM 股票收益率模型波动率

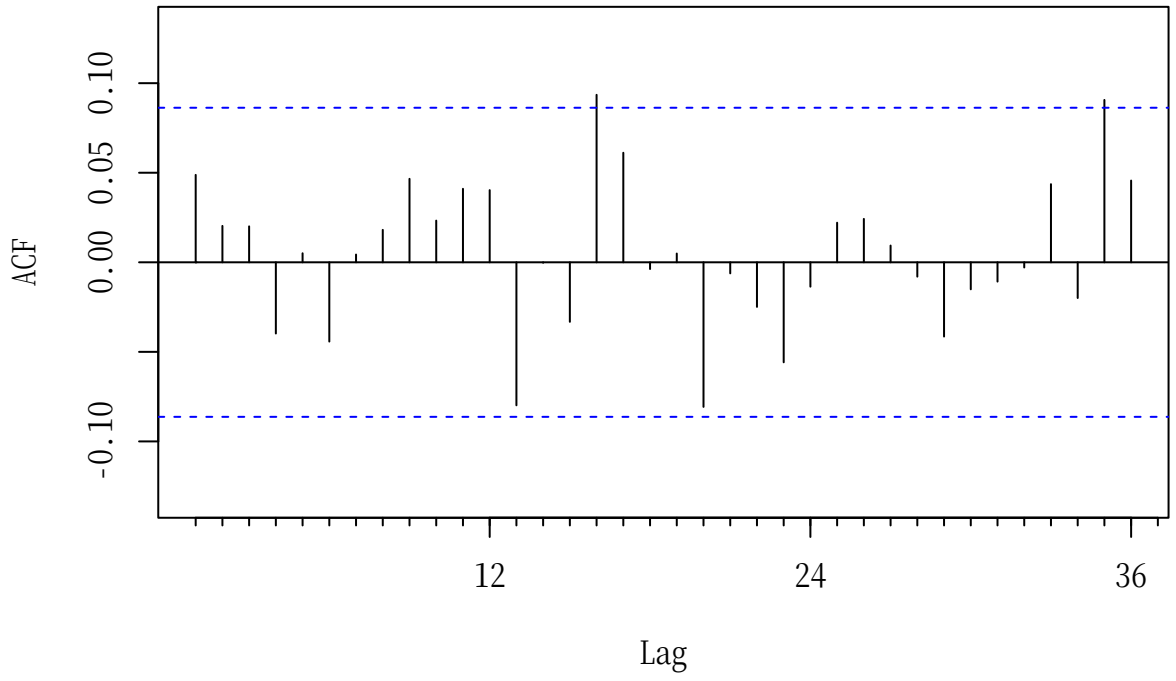


图 19.6: IBM 股票收益率模型标准化残差 ACF

标准化残差的 Ljung-Box 白噪声检验：

```
Box.test(stdresi, lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  stdresi
## X-squared = 6.894, df = 12, p-value = 0.8645
```

标准化残差平方的 ACF:

```
forecast::Acf(stdresi^2, lag.max = 36, main="")
```

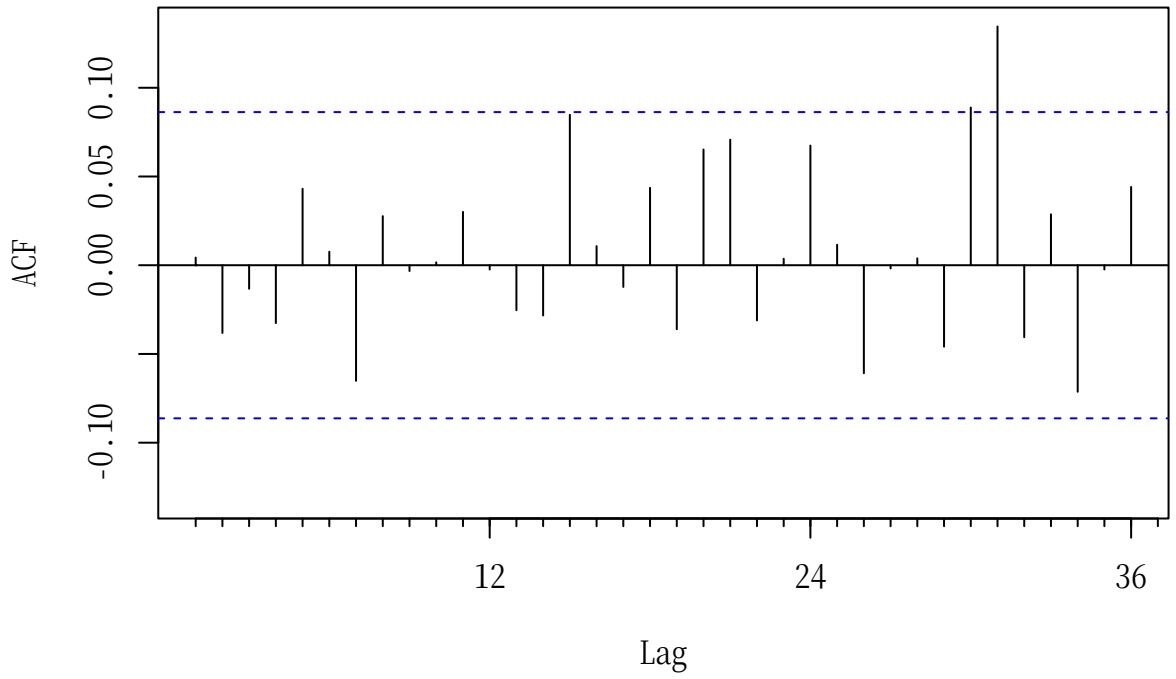


图 19.7: IBM 股票收益率模型标准化残差平方的 ACF

```
## 或:
## plot(mod1ru, which=11)
```

标准化残差平方的 Ljung-Box 白噪声检验：

```
Box.test(stdresi^2, lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
##
```

```
## data:  stdresi^2
## X-squared = 5.537, df = 12, p-value = 0.9376
```

从以上残差诊断看模型比较充分。从 ugarchfit() 的结果看，标准化残差及其残差的白噪声检验通过，分布的拟合优度检验通过，模型参数稳定性检验通过，杠杆效应的检验通过。

19.1.5 预测

以 EGARCH(1,1) 为例说明 EGARCH 模型的超前多步预测。设模型参数已知，ε_t 服从标准正态分布。模型为（见(19.7)、(19.1)和(19.2)）

$$\ln\sigma_t^2 = \alpha_0 + \beta_1 \ln\sigma_{t-1}^2 + \alpha_1\varepsilon_{t-1} + \gamma_1(|\varepsilon_{t-1}| - E|\varepsilon_{t-1}|).$$

其中 $E|\varepsilon_{t-1}| = \sqrt{\frac{2}{\pi}}$。记

$$g(x) = \alpha_1 x + \gamma_1(|x| - \sqrt{\frac{2}{\pi}}),$$

则模型可以写成

$$\ln\sigma_t^2 = \alpha_0 + \beta_1 \ln\sigma_{t-1}^2 + g(\varepsilon_{t-1}),$$

将模型写成指数形式为

$$\sigma_t^2 = e^{\alpha_0}\sigma_{t-1}^{2\beta_1} e^{g(\varepsilon_{t-1})}.$$

以 h 为预测原点，超前一步预测是对 σ_{h+1}^2 的预测，而

$$\sigma_{h+1}^2 = e^{\alpha_0}\sigma_h^{2\beta_1} e^{g(\varepsilon_h)} \in F_h,$$

所以超前一步预测为

$$\hat{\sigma}_h^2(1) = E(\sigma_{h+1}^2|F_h) = \sigma_{h+1}^2. \tag{19.8}$$

对 $t = h+2$，

$$\sigma_{h+2}^2 = e^{\alpha_0}\sigma_{h+1}^{2\beta_1} e^{g(\varepsilon_{h+1})},$$

注意到 $\sigma_{h+1}^2 \in F_h$，ε_{h+1} 与 F_h 独立，所以有

$$\begin{aligned}\hat{\sigma}_h^2(2) =& E(\sigma_{h+2}^2|F_h) = e^{\alpha_0}\sigma_{h+1}^{2\beta_1} E[e^{g(\varepsilon_{h+1})}|F_h] \\ =& e^{\alpha_0}\sigma_{h+1}^{2\beta_1} E[e^{g(\varepsilon_{h+1})}].\end{aligned}$$

只需要计算 $E[e^{g(\varepsilon_{h+1})}]$。设 $\varepsilon \sim \mathrm{N}(0,1)$，则

$$\begin{aligned}Ee^{g(\varepsilon)} =& \int_{-\infty}^{\infty} \exp\left\{\alpha_1 u + \gamma_1|u| - \gamma_1\sqrt{\frac{2}{\pi}}\right\}\varphi(u)\,du \\ =& e^{-\gamma_1\sqrt{\frac{2}{\pi}}}\left\{\int_0^{\infty} \frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}u^2 + (\alpha_1+\gamma_1)u\right]du\right. \\ &\left. + \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}u^2 + (\alpha_1-\gamma_1)u\right]du\right\}\end{aligned}$$

其中

$$-\frac{1}{2}u^2 + (\alpha_1 \pm \gamma_1)u = -\frac{1}{2}[u - (\alpha_1 \pm \gamma_1)]^2 + \frac{1}{2}(\alpha_1 \pm \gamma_1)^2,$$

所以

$$\begin{aligned}
Ee^{g(\varepsilon)} =& e^{-\gamma_1\sqrt{\frac{2}{\pi}}}\left\{e^{\frac{1}{2}(\alpha_1+\gamma_1)^2}\int_0^{\infty}\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}(u-(\alpha_1+\gamma_1))^2\right]du\right.\\
&\left.+e^{\frac{1}{2}(\alpha_1-\gamma_1)^2}\int_{-\infty}^{0}\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}(u-(\alpha_1-\gamma_1))^2\right]du\right\}\\
=& e^{-\gamma_1\sqrt{\frac{2}{\pi}}}\left\{e^{\frac{1}{2}(\alpha_1+\gamma_1)^2}\int_{-(\alpha_1+\gamma_1)}^{\infty}\varphi(v)\,dv+e^{\frac{1}{2}(\alpha_1-\gamma_1)^2}\int_{-\infty}^{-(\alpha_1-\gamma_1)}\varphi(v)\,dv\right\}\\
=& e^{-\gamma_1\sqrt{\frac{2}{\pi}}}\left\{e^{\frac{1}{2}(\alpha_1+\gamma_1)^2}[1-\Phi(-(\alpha_1+\gamma_1))]+e^{\frac{1}{2}(\alpha_1-\gamma_1)^2}\Phi(-(\alpha_1-\gamma_1))\right\}\\
=& e^{-\gamma_1\sqrt{\frac{2}{\pi}}}\left\{e^{\frac{1}{2}(\gamma_1+\alpha_1)^2}\Phi(\gamma_1+\alpha_1)+e^{\frac{1}{2}(\gamma_1-\alpha_1)^2}\Phi(\gamma_1-\alpha_1)\right\}.
\end{aligned}$$

其中 $\varphi(\cdot)$ 和 $\Phi(\cdot)$ 是标准正态分布的密度和分布函数。

于是，超前二步预测为

$$\hat{\sigma}_h^2(2) = e^{\alpha_0}Ee^{g(\varepsilon)}[\hat{\sigma}_h^2(1)]^{\beta_1}. \tag{19.9}$$

超前三步：

$$\sigma_{h+3}^2 = e^{\alpha_0}(\sigma_{h+2}^2)^{\beta_1}e^{g(\varepsilon_{h+2})},$$

注意到 $F_h \subset F_{h+1}$, $\sigma_{h+2}^2 \in F_{h+1}$, ε_{h+2} 与 F_{h+1} 独立，可得

$$\begin{aligned}
\hat{\sigma}_h^2(3) =& e^{\alpha_0}E\left\{(\sigma_{h+2}^2)^{\beta_1}e^{g(\varepsilon_{h+2})}|F_h\right\} && (19.10)\\
=& e^{\alpha_0}E\left\{E\left[(\sigma_{h+2}^2)^{\beta_1}e^{g(\varepsilon_{h+2})}|F_{h+1}\right]|F_h\right\} && (19.11)\\
=& e^{\alpha_0}E\left\{(\sigma_{h+2}^2)^{\beta_1}E\left[e^{g(\varepsilon_{h+2})}|F_{h+1}\right]|F_h\right\} && (19.12)\\
=& e^{\alpha_0}E[e^{g(\varepsilon)}]E\left\{(\sigma_{h+2}^2)^{\beta_1}|F_h\right\} && (19.13)\\
=& e^{\alpha_0}E[e^{g(\varepsilon)}]E\left\{e^{\alpha_0\beta_1}(\sigma_{h+1}^2)^{\beta_1^2}e^{\beta_1 g(\varepsilon_{h+1})}|F_h\right\} && (19.14)\\
=& e^{\alpha_0(1+\beta_1)}E[e^{g(\varepsilon)}]E[e^{\beta_1 g(\varepsilon)}](\hat{\sigma}_h^2(1))^{\beta_1^2}. && (19.15)
\end{aligned}$$

对 $\hat{\sigma}_h^2(j)$ 可类似计算。

19.2 GJR-GARCH 模型

GJR-GARCH 模型是另一个能够反映杠杆效应的波动率模型，参见 (Glosten et al., 1993) 和 (Zakoian, 1994)。或称为 TGARCH。GJR-GARCH(m,s) 模型的形式为

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{m}(\alpha_i+\gamma_i N_{t-i})a_{t-i}^2 + \sum_{j=1}^{s}\beta_j\sigma_{t-j}^2 \tag{19.16}$$

其中 N_{t-i} 是表示 $a_{t-i}<0$ 的示性函数，即

$$N_{t-i} = \begin{cases}1, & a_{t-i}<0\\ 0, & a_{t-i}\geq 0\end{cases}$$

$\alpha_i, \gamma_i, \beta_j$ 是非负参数，满足与 GARCH 模型类似的参数条件。

正的 a_{t-i} 对 σ_t^2 的影响为 $\alpha_i a_{t-i}^2$，负的 a_{t-i} 对 σ_t^2 的影响为 $(\alpha_i+\gamma_i)a_{t-i}^2$，当 $\gamma_i>0$ 时负的 a_{t-i} 影响较大。

模型用 0 作为 a_{t-i} 的门限，还可以用其它实数值作为门限。关于门限模型参见 (Tsay, 2010) 第 4 章。

作为例子，考虑从 1999 年 1 月 4 日到 2010 年 8 月 20 日的欧元对美元汇率的日对数收益率序列。

读入数据：

```
d.useu <- read_table(
  "d-useu9910.txt",
  col_types=cols(.default=col_double())
)
xts.useu <- with(d.useu, xts(rate, make_date(year, mon, day)))
xts.useu.lnrtn <- diff(log(xts.useu))[-1,]*100
```

用 rugarch 包建模：

```
library(rugarch)
spec2 <- ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=TRUE  ),
  variance.model = list(
    model = "gjrGARCH", # GJR-GARCH or TGARCH model
    garchOrder = c(1,1) ) )
mod2ru <- ugarchfit(spec = spec2, data = xts.useu.lnrtn)
show(mod2ru)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.012262    0.010726   1.1432 0.252951
## omega   0.001274    0.000554   2.3013 0.021373
## alpha1  0.022402    0.004277   5.2373 0.000000
## beta1   0.968711    0.001779 544.4570 0.000000
## gamma1  0.012435    0.007027   1.7696 0.076789
##
```

```
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.012262    0.011704    1.0476 0.294800
## omega   0.001274    0.000620    2.0535 0.040021
## alpha1  0.022402    0.004777    4.6891 0.000003
## beta1   0.968711    0.000795 1219.2085 0.000000
## gamma1  0.012435    0.008047    1.5454 0.122258
##
## LogLikelihood : -2731.85
##
## Information Criteria
## ------------------------------------
##
## Akaike       1.8688
## Bayes        1.8790
## Shibata      1.8688
## Hannan-Quinn 1.8725
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2619  0.6088
## Lag[2*(p+q)+(p+q)-1][2]    0.3038  0.7936
## Lag[4*(p+q)+(p+q)-1][5]    3.3494  0.3469
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      4.935 0.02632
## Lag[2*(p+q)+(p+q)-1][5]     7.136 0.04806
## Lag[4*(p+q)+(p+q)-1][9]     9.101 0.07748
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     2.979 0.500 2.000 0.08435
## ARCH Lag[5]     3.239 1.440 1.667 0.25696
## ARCH Lag[7]     4.681 2.315 1.543 0.25882
##
## Nyblom stability test
```

```
## ------------------------------------
## Joint Statistic:  1.4726
## Individual Statistics:
## mu     0.20009
## omega  0.20231
## alpha1 0.06618
## beta1  0.08927
## gamma1 0.08316
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value     prob sig
## Sign Bias            1.663 0.096368   *
## Negative Sign Bias   2.386 0.017101  **
## Positive Sign Bias   1.740 0.082025   *
## Joint Effect        12.898 0.004863 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     59.51    4.628e-06
## 2    30     70.40    2.676e-05
## 3    40     69.44    1.928e-03
## 4    50     81.46    2.454e-03
##
##
## Elapsed time : 0.2142711
```

估计的模型为

$$
\begin{aligned}
r_t =& 0.0123 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \varepsilon_t \text{ i.i.d. } \mathrm{N}(0,1) \\
\sigma_t^2 =& 0.0013 + (0.0224 + 0.0124 N_{t-1}) a_{t-1}^2 + 0.9687 \sigma_{t-1}^2,
\end{aligned}
$$

模型结果中 mu 不显著，在 0.05 水平下 gamma1 也不显著，说明杠杆效应不显著。模型标准化残差的白噪声检验通过，但标准化残差平方的白噪声检验没有通过，所以模型效果不够好。模型参数稳定性检验没有通过。拟合优度检验没有通过。

拟合的波动率 σ_t 序列图形 (单位是百分之一)：

```
plot(sigma(mod2ru),
  format.labels="%Y",
  main=" 欧元汇率日对数收益率波动率 TGARCH 估计 (%)",
```

```
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years")
```

图 19.8: 欧元汇率日对数收益率波动率 TGARCH 估计

波动率的最高值出现在 2008 年末、2009 年初，次贷危机爆发时期。

为了进行模型诊断，计算标准化残差：

```
stdresi2 <- residuals(mod2ru, standardize=TRUE)
```

标准化残差的 ACF:

```
forecast::Acf(stdresi2, lag.max=36, main="")
```

```
## 或:
## plot(mod2ru, which=10)
```

标准化残差平方的 ACF:

```
forecast::Acf(stdresi2^2, lag.max=36, main="")
```

```
## 或:
## plot(mod2ru, which=11)
```

标准化残差的 Ljung-Box 白噪声检验：

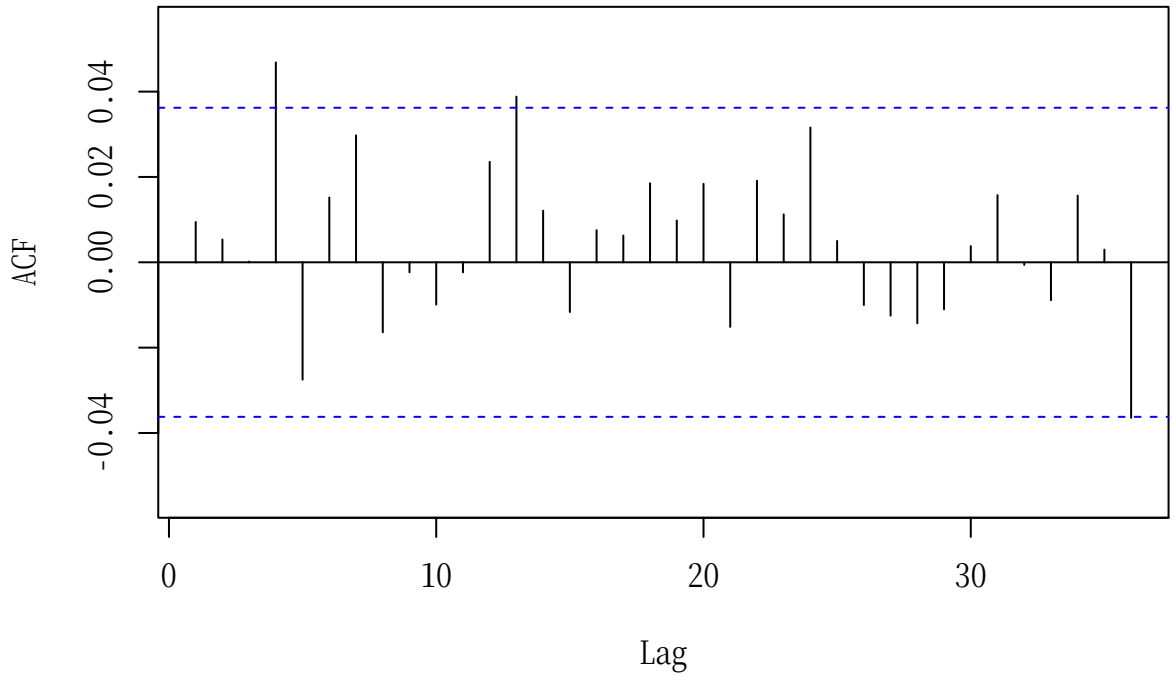


图 19.9: 欧元汇率日对数收益率 TGARCH 标准化残差 ACF

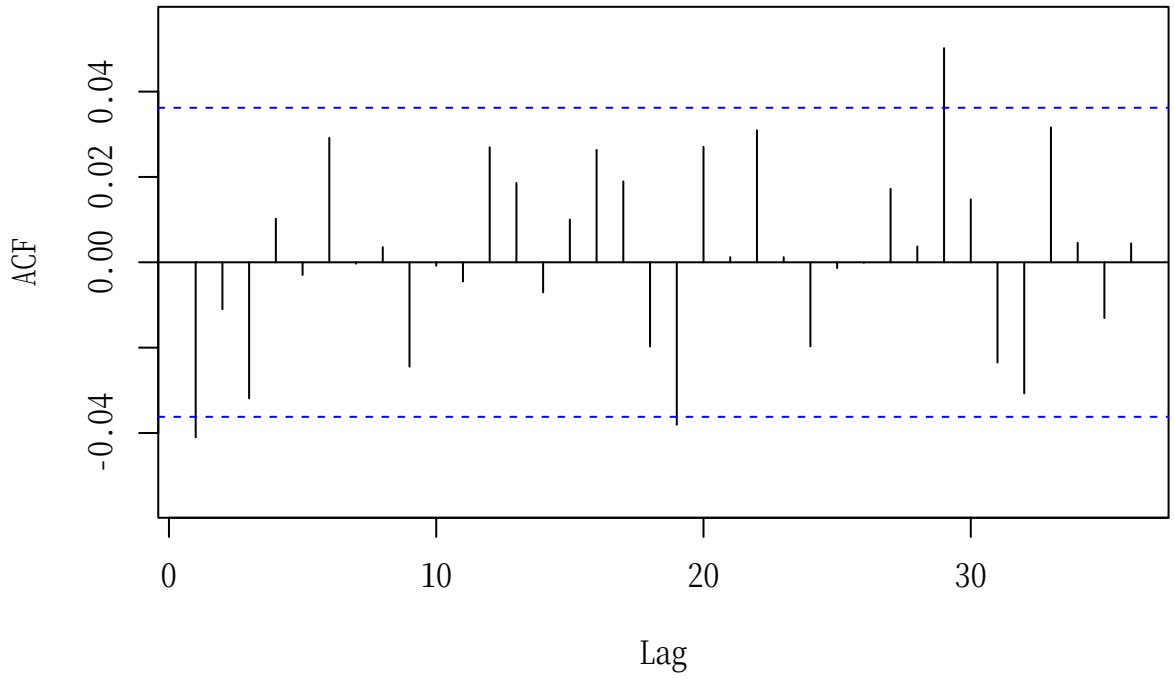


图 19.10: 欧元汇率日对数收益率 TGARCH 标准化残差平方 ACF

```
Box.test(stdresi2, lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  stdresi2
## X-squared = 15.029, df = 12, p-value = 0.2399
```

标准化残差平方的 Ljung-Box 白噪声检验：

```
Box.test(stdresi2^2, lag=12, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  stdresi2^2
## X-squared = 15.094, df = 12, p-value = 0.2363
```

这些残差诊断的结果说明拟合的模型是基本符合的。

作标准化残差的盒形图：

```
boxplot(c(coredata(stdresi2)),
  main="", xlab=" 标准化残差")
```

标准化残差的正态 QQ 图：

```
plot(mod2ru, which=9)
```

从标准化残差的盒形图看有厚尾现象。

可以将 GJR-GARCH 得到的波动率与标准 GARCH 得到的波动率进行比较：

```
library(fGarch, quietly = TRUE)
mod2sg <- garchFit( ~ 1 + garch(1,1),
  data=xts.useu.lnrtn, trace=FALSE)
```

将两个波动率作图，黑色为 GARCH 结果，红色虚线为 GJR-GARCH 结果：

```
plot(cbind(volatility(mod2sg), sigma(mod2ru)),
  main=" 欧元汇率日对数收益率波动率 GARCH 和 NGARCH 估计",
  lty=c(1,2),
  col=c("black", "red"),
  format.labels="%Y",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years")
```

放大显示其中 2008 年以后：

```
plot(cbind(volatility(mod2sg), sigma(mod2ru)),
  subset="2008/",
```

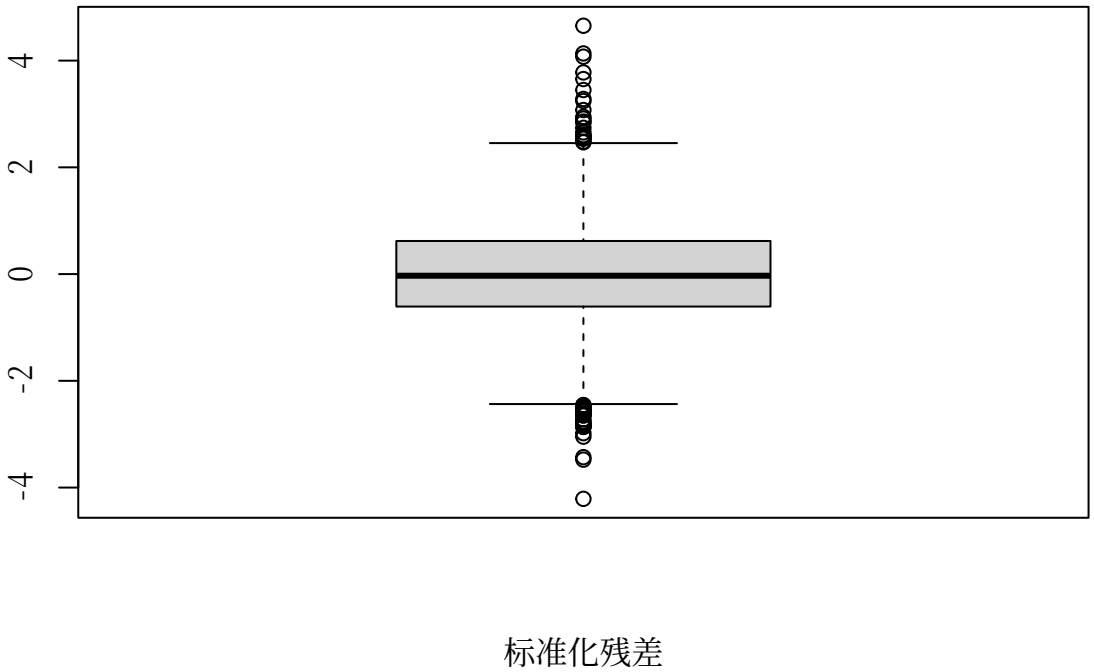


图 19.11: 欧元汇率日对数收益率 TGARCH 标准化残差盒形图

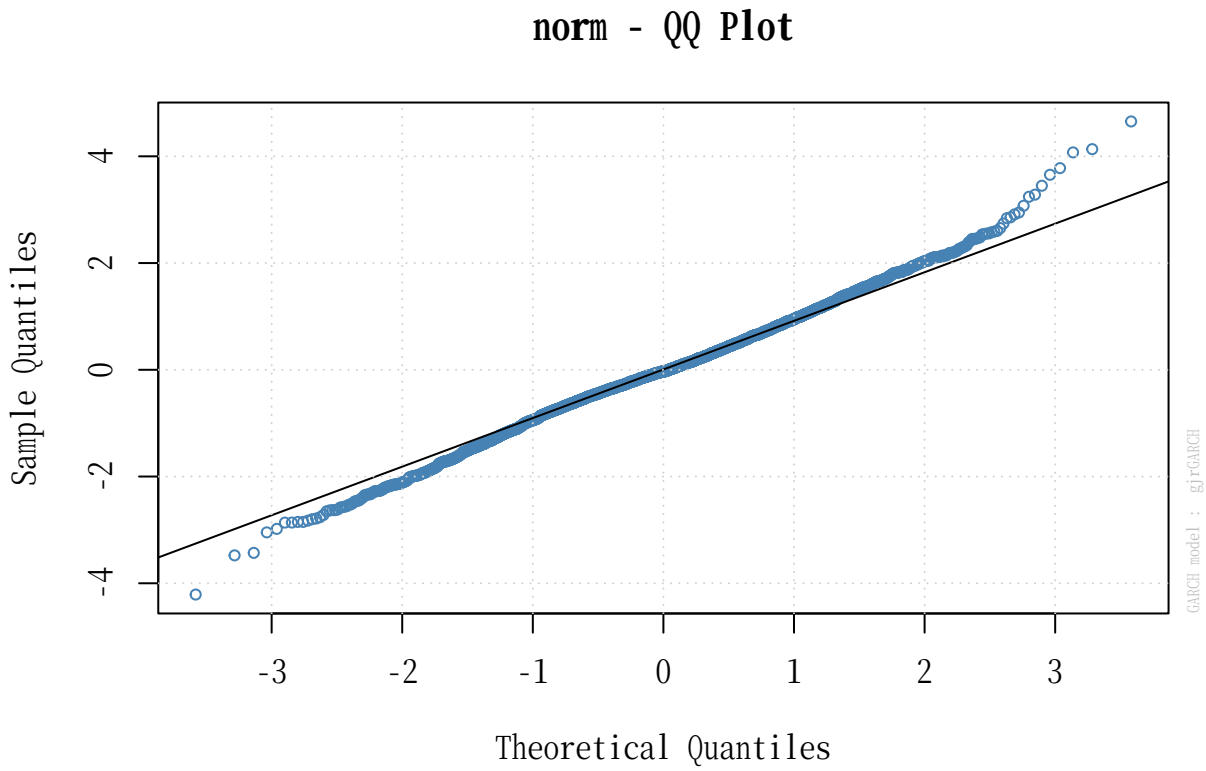


图 19.12: 欧元汇率日对数收益率 TGARCH 标准化残差正态 QQ 图

图 19.13: 欧元汇率 GARCH 和 NGARCH 估计的波动率

```
main=" 欧元汇率日对数收益率波动率 GARCH 和 NGARCH 估计",
lty=c(1,2),
col=c("black", "red"),
format.labels="%Y",
major.ticks="years", minor.ticks=NULL,
grid.ticks.on="years")
```

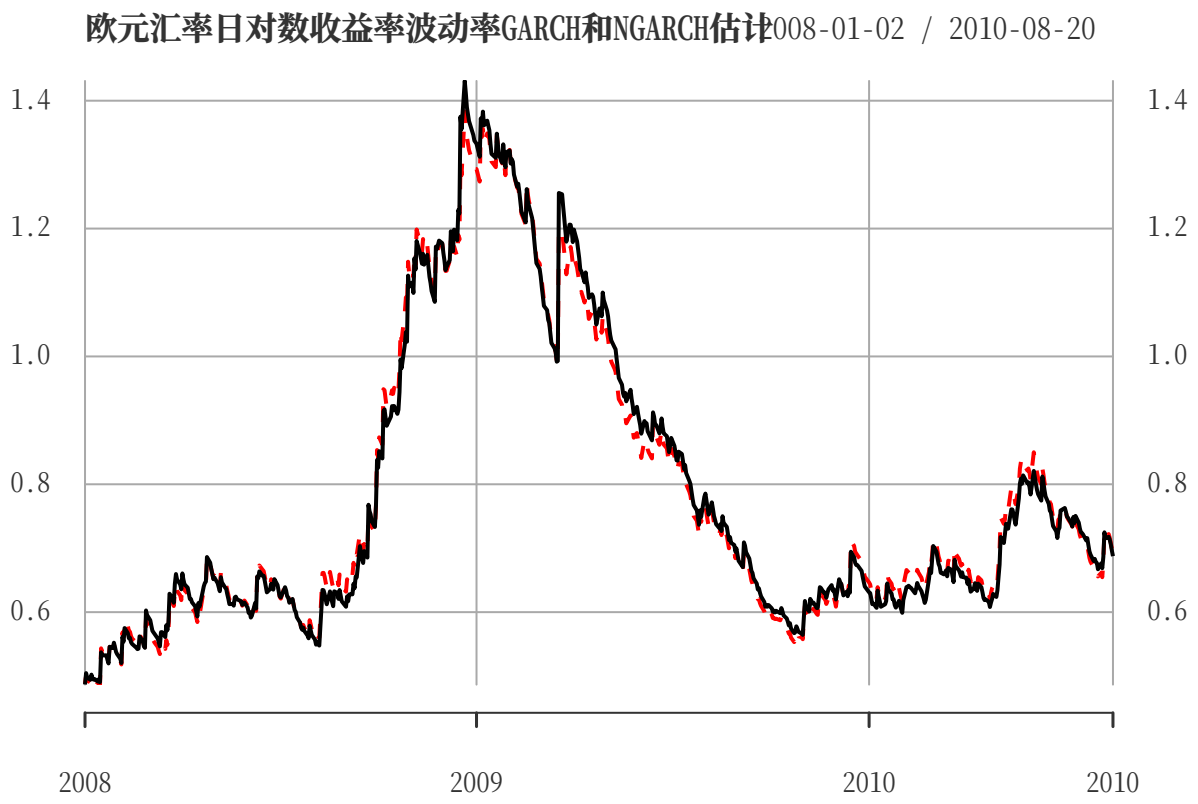


图 19.14: 欧元汇率 GARCH 和 NGARCH 估计的波动率

19.3 APARCH 模型

(Ding et al., 1993) 提出了非对称幂 (asymmetric power) ARCH 模型 (APARCH 模型)，模型形式为

$$r_t = \mu_t + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \varepsilon_t \sim D(0,1) \tag{19.17}$$

$$\sigma_t^\delta = \omega + \sum_{i=1}^{m} \alpha_i (|a_{t-i}| - \gamma_i a_{t-i})^\delta + \sum_{j=1}^{s} \beta_j \sigma_{t-j}^\delta \tag{19.18}$$

其中 μ_t 是条件均值，$D(0,1)$ 表示某个零均值单位方差分布，δ 为正实数，系数 $\omega, \alpha_i, \gamma_i, \beta_j$ 满足某些正则性条件使得波动率为正。

最常用的是最简单的 APARCH(1,1) 模型。这个模型中包含了许多其它模型。

当 $\delta = 2$ 且 $\gamma_j = 0$ 时即普通的 GARCH 模型。

当 $\delta = 2$ 时即 TGARCH 模型（形式略有不同）。

当 $\delta = 1$ 时波动率方程直接使用波动率 σ_t 和新息 a_t 而非其平方。

APARCH 中的幂变换旨在提高拟合程度，但幂次 δ 没有很好解释。

R 的 `fGarch::garchFit()` 函数中可以使用 `aparch(m,s)` 作为模型设定。作为例子，拟合欧元对美元汇率数据：

```
library(fGarch, quietly = TRUE)
modres3 <- garchFit( ~ 1 + aparch(1,1),
  data=xts.useu.lnrtn, trace=FALSE)
summary(modres3)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + aparch(1, 1), data = xts.useu.lnrtn,
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + aparch(1, 1)
## <environment: 0x000001dd6c3b0610>
##  [data = xts.useu.lnrtn]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        mu      omega     alpha1     gamma1      beta1      delta
## 0.0127647  0.0015919  0.0313680  0.1135334  0.9689155  1.6743075
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.0127647   0.0107626    1.186   0.2356
## omega  0.0015919   0.0007226    2.203   0.0276 *
## alpha1 0.0313680   0.0053350    5.880 4.11e-09 ***
## gamma1 0.1135334   0.0711911    1.595   0.1108
## beta1  0.9689155   0.0038405  252.292  < 2e-16 ***
## delta  1.6743075   0.4057124    4.127 3.68e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Log Likelihood:
##  -2731.172    normalized:  -0.9324587
##
## Description:
##  Tue May 30 17:03:58 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  50.20526  1.253331e-11
##  Shapiro-Wilk Test  R    W      0.9956711 1.608388e-07
##  Ljung-Box Test     R    Q(10)  13.37689  0.2033562
##  Ljung-Box Test     R    Q(15)  20.19634  0.1645295
##  Ljung-Box Test     R    Q(20)  22.84736  0.2963514
##  Ljung-Box Test     R^2  Q(10)  13.15611  0.2150739
##  Ljung-Box Test     R^2  Q(15)  16.58008  0.34458
##  Ljung-Box Test     R^2  Q(20)  27.44887  0.1231012
##  LM Arch Test       R    TR^2   14.35739  0.2784708
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 1.869014 1.881269 1.869006 1.873428
```

拟合的白噪声检验说明模型是充分的，但是 δ 的估计为 1.67, 标准误差为 0.41，其 95% 置信区间为 $(0.85, 2.49)$ 包含 2，所以与 $\delta = 2$ 没有显著差异，可以用 TGARCH 模型。γ_1 在 0.05 水平下不显著，可以不需要反映杠杆效应的模型。

拟合的 APARCH 模型为：

$$
\begin{aligned}
r_t =& 0.0128 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \varepsilon_t \sim \mathrm{N}(0,1) \\
\sigma_t^{1.67} =& 0.0016 + 0.0313(|a_{t-1}| - 0.1135 a_{t-1})^{1.67} + 0.9689 \sigma_{t-1}^{1.67}
\end{aligned}
$$

固定 $\delta = 2$ 估计 APARCH(1,1) 模型：

```
modres4 <- garchFit( ~ 1 + aparch(1,1),
  data=xts.useu.lnrtn, delta=2,
  include.delta=FALSE, trace=FALSE)
summary(modres4)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + aparch(1, 1), data = xts.useu.lnrtn,
```

```
##     delta = 2, include.delta = FALSE, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + aparch(1, 1)
## <environment: 0x000001dd6cf1d9a0>
##  [data = xts.useu.lnrtn]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        mu      omega     alpha1     gamma1      beta1
## 0.0122646  0.0012745  0.0282723  0.1100242  0.9687115
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.0122646   0.0107289    1.143   0.2530
## omega  0.0012745   0.0005752    2.216   0.0267 *
## alpha1 0.0282723   0.0038637    7.317 2.53e-13 ***
## gamma1 0.1100242   0.0649051    1.695   0.0900 .
## beta1  0.9687115   0.0039421  245.735  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -2731.85    normalized:  -0.9326902
##
## Description:
##  Tue May 30 17:03:59 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  49.97677  1.405021e-11
##  Shapiro-Wilk Test  R    W      0.9956803 1.655881e-07
##  Ljung-Box Test     R    Q(10)  13.38285  0.2030469
##  Ljung-Box Test     R    Q(15)  20.29833  0.1607845
##  Ljung-Box Test     R    Q(20)  22.87265  0.2950907
##  Ljung-Box Test     R^2  Q(10)  12.89586  0.2295531
##  Ljung-Box Test     R^2  Q(15)  16.55288  0.3462876
```

```
##  Ljung-Box Test      R^2  Q(20)  27.24036  0.128636
##  LM Arch Test        R    TR^2   14.29661  0.2821695
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 1.868795 1.879007 1.868789 1.872472
```

拟合的固定 $\delta = 2$ 的 APARCH 模型为：

$$\begin{aligned} r_t =& 0.0123 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \varepsilon_t \sim \mathrm{N}(0,1) \\ \sigma_t^2 =& 0.00127 + 0.0283(|a_{t-1}| - 0.1100 a_{t-1})^2 + 0.9687\sigma_{t-1}^2 \end{aligned}$$

其中的波动率方程可以借助 N_{t-1} 为 $a_{t-1} < 0$ 的示性函数写成

$$\sigma_t^2 = 0.00127 + (0.0224 + 0.0125 N_{t-1}) a_{t-1}^2 + 0.9687\sigma_{t-1}^2.$$

与直接估计 TGARCH(1,1) 的结果对比：

$$\begin{aligned} r_t =& 0.0123 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \varepsilon_t \text{ i.i.d. } \mathrm{N}(0,1) \\ \sigma_t^2 =& 0.0013 + (0.0224 + 0.0124 N_{t-1}) a_{t-1}^2 + 0.9687\sigma_{t-1}^2. \end{aligned}$$

两个估计结果基本相同。

也可以用 rugarch 包估计 APARCH 模型：

```
library(rugarch)
spec3 <- ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=TRUE  ),
  variance.model = list(
    model = "apARCH", # APARCH model
    garchOrder = c(1,1) ) )
mod3ru <- ugarchfit(spec = spec3,
  data = xts.useu.lnrtn)
show(mod3ru)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : apARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
```

```
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.012516    0.010740    1.1654 0.243858
## omega   0.001683    0.000595    2.8299 0.004656
## alpha1  0.031692    0.000877   36.1495 0.000000
## beta1   0.969388    0.000679 1427.5230 0.000000
## gamma1  0.121817    0.072545    1.6792 0.093115
## delta   1.580101    0.129609   12.1913 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.012516    0.011800    1.0607 0.288841
## omega   0.001683    0.000772    2.1804 0.029229
## alpha1  0.031692    0.002833   11.1859 0.000000
## beta1   0.969388    0.000680 1425.6414 0.000000
## gamma1  0.121817    0.084847    1.4357 0.151084
## delta   1.580101    0.082927   19.0542 0.000000
##
## LogLikelihood : -2731.113
##
## Information Criteria
## ------------------------------------
##
## Akaike       1.8690
## Bayes        1.8812
## Shibata      1.8690
## Hannan-Quinn 1.8734
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2823  0.5952
## Lag[2*(p+q)+(p+q)-1][2]    0.3207  0.7843
## Lag[4*(p+q)+(p+q)-1][5]    3.3970  0.3393
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      5.165 0.02304
## Lag[2*(p+q)+(p+q)-1][5]     7.349 0.04264
```

```
## Lag[4*(p+q)+(p+q)-1][9]     9.341 0.06909
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     2.896 0.500 2.000 0.08882
## ARCH Lag[5]     3.238 1.440 1.667 0.25703
## ARCH Lag[7]     4.723 2.315 1.543 0.25404
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  1.5207
## Individual Statistics:
## mu     0.18789
## omega  0.18056
## alpha1 0.07682
## beta1  0.09516
## gamma1 0.38035
## delta  0.08651
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.49 1.68 2.12
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value     prob sig
## Sign Bias            1.641 0.100814
## Negative Sign Bias   2.398 0.016537  **
## Positive Sign Bias   1.771 0.076666   *
## Joint Effect        12.991 0.004657 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     62.92    1.323e-06
## 2    30     71.22    2.064e-05
## 3    40     73.65    6.628e-04
## 4    50     79.79    3.561e-03
##
##
```

```
## Elapsed time : 0.474442
```

与 fGarch 包的估计结果不完全相同，参数值比较接近。

Chapter 20

随机波动率模型

本章内容来自自 (Tsay, 2013)§4.13 和 §4.14 内容。

20.1 随机波动率模型

前面的波动率方程中 $\sigma_t^2 = \mathrm{Var}(a_t|F_{t-1})$ 都是被 $\sigma_{t-1}, ...$ 和 $a_{t-1}, ...$ 完全决定。另一种方法是假定 σ_t^2 的模型本身有新息，这样的模型称为随机波动率 (Stochastic Volatility, SV) 模型。模型写成

$$a_t = \sigma_t \varepsilon_t, \quad (1 - \alpha_1 B - \cdots - \alpha_m B^m) \ln \sigma_t^2 = \alpha_0 + v_t.$$

其中 σ_t^2 取对数是为了取消系数必须为非负的限制。$\{\varepsilon_t\}$ 独立同标准正态分布，$\{v_t\}$ 独立同 $\mathrm{N}(0, \sigma_v^2)$ 分布，$\{\varepsilon_t\}$ 和 $\{v_t\}$ 相互独立。α_i 为常数，特征多项式 $1 - \alpha_1 z - \cdots - \alpha_m z^m$ 根都在单位圆外。记 $\xi_t = \ln \sigma_t^2$，则 $\{\xi_t\}$ 是一个严平稳 AR(m) 序列。

加入 v_t 新息后，收益率 r_t 的一个新息 a_t 就包含了 ε_t 和 v_t 两个新息，这增加了模型的自由度，但是使得从 r_t 数据估计模型参数变得更加困难，需要使用 Kalman 滤波或者随机模拟方法计算拟似然估计。

当 $m = 1$ 时，有

$$\begin{aligned}
\ln \sigma_t^2 \sim & \mathrm{N}\left(\frac{\alpha_0}{1-\alpha_1}, \frac{\sigma_v^2}{1-\alpha_1^2}\right) = \mathrm{N}(\mu_h, \sigma_h^2), \\
Ea_t^2 = & \exp\left(\mu_h + \frac{1}{2}\sigma_h^2\right), \\
Ea_t^4 = & 3\exp\left(2\mu_h^2 + 2\sigma_h^2\right), \\
\rho(a_t^2, a_{t-i}^2) = & \frac{e^{\sigma_h^2 \alpha_1^i} - 1}{3e^{\sigma_h^2} - 1}.
\end{aligned}$$

SV 模型经常在拟合上有所改善，但是波动率的样本外预测时好时坏。

20.2 长记忆随机波动率模型

对资产收益率的实证分析发现，收益率本身没有长记忆性，但是其平方序列或者绝对值序列的 ACF 往往衰减很慢。前面 GARCH 类模型的建模中 σ_{t-1}^2 的系数很接近于 1，也提示有长记忆。

下面对 1962 年到 2003 年标普 500 指数和 IBM 股票的日对数收益率序列的绝对值作 ACF，可以看到长记忆现象存在。

```
da <- read_table(
  "d-ibmvwewsp5-6203.txt",
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
xts.ibm <- xts(log(1 + da[,-1]), da[["date"]])
ibm <- coredata(xts.ibm)[,"ibm"]
sp5 <- coredata(xts.ibm)[,"sp5"]
```

标普 500 指数日对数收益率绝对值的 ACF：

```
np <- 200; nt <- length(sp5)
tmpa <- acf(abs(sp5), lag.max=np, main="", plot=FALSE)
plot(seq(np), tmpa$acf[2:(np+1)], type="h", xlab="Lag", ylab="acf", ylim=c(-0.05, 0.3))
abline(h=c(2,-2)/sqrt(nt), lty=2, col="blue")
```

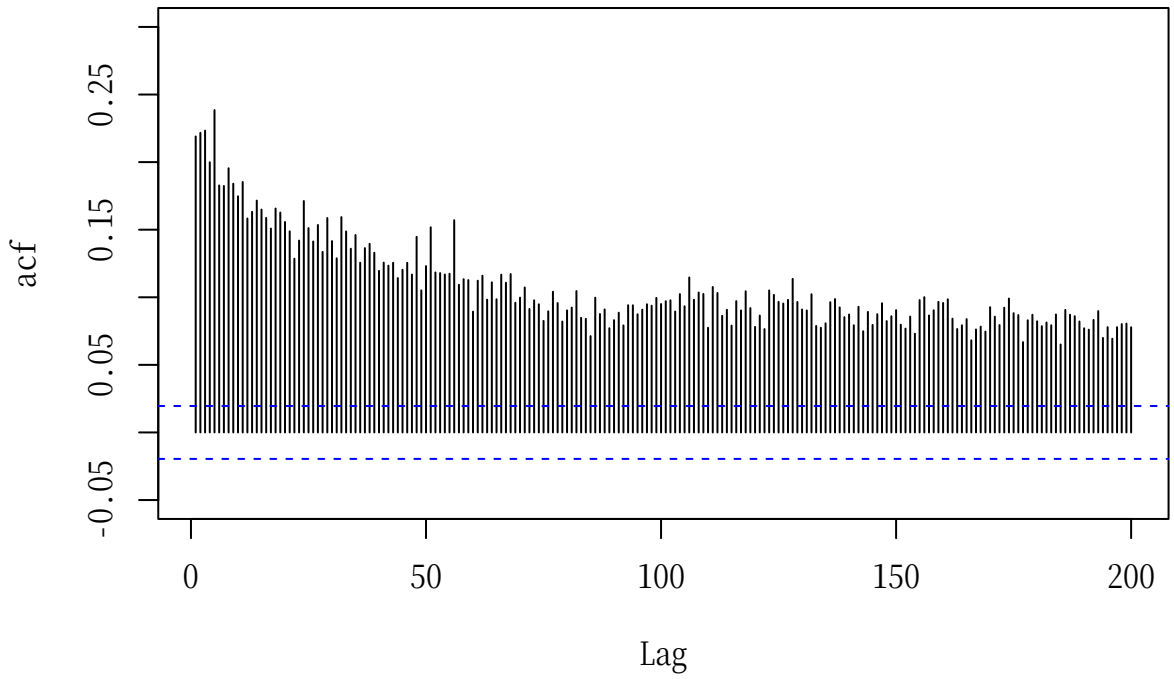


图 20.1: 标普 500 指数日对数收益率绝对值的 ACF

IBM 股票日对数收益率绝对值的 ACF：

```
np <- 200; nt <- length(sp5)
tmpa <- acf(abs(ibm), lag.max=np, main="", plot=FALSE)
plot(seq(np), tmpa$acf[2:(np+1)], type="h", xlab="Lag", ylab="acf", ylim=c(-0.05, 0.3))
abline(h=c(2,-2)/sqrt(nt), lty=2, col="blue")
```

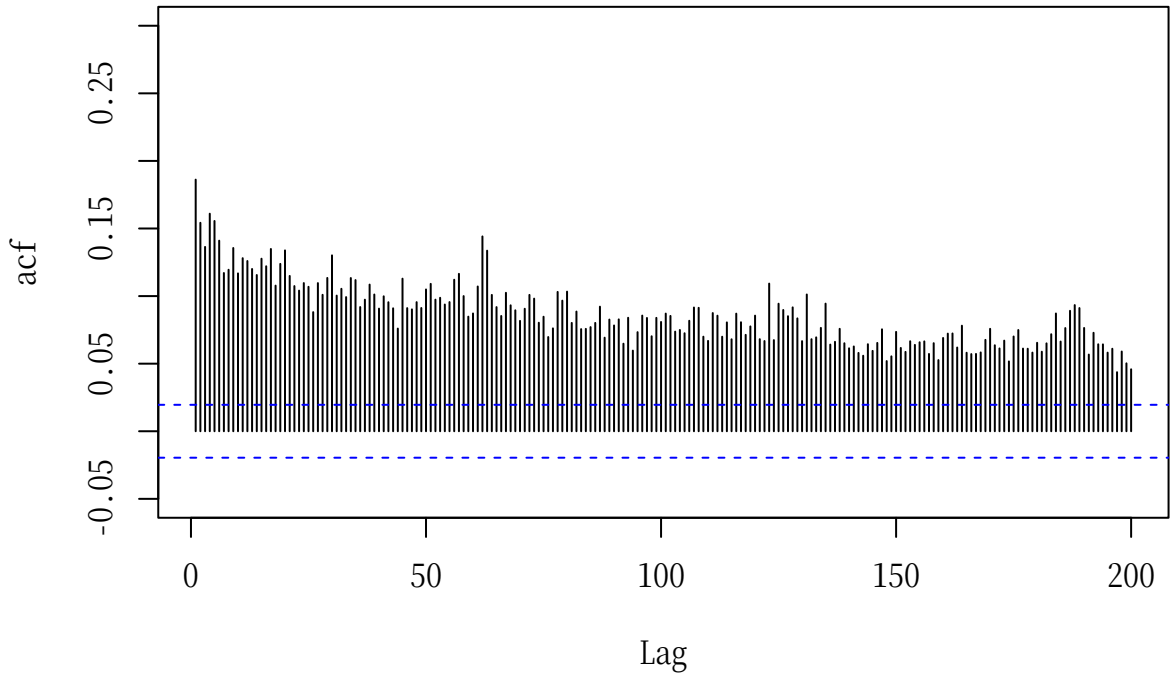


图 20.2: IBM 股票日对数收益率绝对值的 ACF

简单的长记忆随机波动率 (LMSV) 模型可以写成

$$a_t = \sigma_t \varepsilon_t, \quad \sigma_t = \sigma e^{\frac{1}{2}u_t}, \quad (1-B)^d u_t = \eta_t.$$

其中 $\sigma > 0$，$\{\varepsilon_t\}$ 和 $\{\eta_t\}$ 是两个相互独立的独立同分布高斯白噪声列，$\varepsilon_t \sim \mathrm{N}(0,1)$, $\eta_t \sim \mathrm{N}(0, \sigma_\eta^2)$, $0 < d < 0.5$。长记忆来源于分数差分 $(1-B)^d$，这使得 u_t 的 ACF 以负幂速度衰减而非负指数速度衰减。

对 LMSV 有

$$\begin{aligned}\ln a_t^2 =& \ln(\sigma_t^2 \varepsilon_t^2) = \ln \sigma^2 + u_t + \ln \varepsilon_t^2 \\ =& (\ln \sigma^2 + E \ln \varepsilon_t^2) + u_t + (\ln \varepsilon_t^2 - E \ln \varepsilon_t^2) \\ =& \mu + u_t + e_t.\end{aligned}$$

其中 u_t 是一个长记忆的平稳高斯时间序列，e_t 是一个非高斯的独立同分布白噪声列。

LMSV 估计比较复杂，分数参数 d 可以用拟最大似然估计法或者回归方法估计。标普 500 指数成份股日收益率平方的对数序列的 d 估计的中位数是 0.38。同一行业的股票的长记忆成分往往相同。

Chapter 21

其它的波动率计算方法

本章内容来自自 (Tsay, 2013)§4.15 和 §4.16 内容。

21.1 利用日频计算月频波动率

(French et al., 1987) 用高频数据计算低频收益率的波动率，又可参见 (Andersen et al., 2001) 和 (Andersen et al., 2001)。

假设我们对某资产的月波动率感兴趣，有该资产的日收益率数据，设 r_t^m 是该资产第 t 个月的对数收益率，第 t 个月共有 n 个交易日，所有日对数收益率为 $\{r_{t,i}\}_{i=1}^n$，则

$$r_t^m = \sum_{i=1}^n r_{t,i}.$$

设各收益率序列的条件方差存在，记 F_{t-1} 为截止到 $t-1$ 个月为止的信息，则

$$\mathrm{Var}(r_t^m|F_{t-1}) = \sum_{i=1}^n \mathrm{Var}(r_{t,i}|F_{t-1}) + 2\sum_{i<j} \mathrm{Cov}(r_{t,i}, r_{t,j}|F_{t-1}). \tag{21.1}$$

上述公式依赖于 t, i 以及复杂的方差结构。若 $\{r_{t,i}\}$ 是独立同分布零均值白噪声列，则这时 $r_{t,i}$ 与 F_{t-1} 独立，有 $\mathrm{Cov}(r_{t,i}, r_{t,j}|F_{t-1}) = \mathrm{Cov}(r_{t,i}, r_{t,j}) = 0$, $\mathrm{Var}(r_{t,i}|F_{t-1}) = \mathrm{Var}(r_{t,i}) = \mathrm{Var}(r_{t,1})$，所以这时

$$\mathrm{Var}(r_t^m|F_{t-1}) = n\mathrm{Var}(r_{t,1}) \tag{21.2}$$

其中 $\mathrm{Var}(r_{t,1})$ 可以从样本估计为

$$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^n (r_{t,i} - \bar{r}_t)^2, \quad \bar{r}_t = \frac{1}{n}\sum_{i=1}^n r_{t,i}.$$

于是 $\mathrm{Var}(r_t^m|F_{t-1})$ 的估计为

$$\hat{\sigma}_m^2 = \frac{n}{n-1}\sum_{i=1}^n (r_{t,i} - \bar{r}_t)^2. \tag{21.3}$$

若 $\{r_{t,i}\}$ 服从由独立同分布白噪声产生的 MA(1) 模型，则 $r_{t,i}, i = 2, \ldots, n$ 与 F_{t-1} 独立，近似有

$$\mathrm{Var}(r_t^m|F_{t-1}) \approx n\mathrm{Var}(r_{t,2}) + 2(n-1)\mathrm{Cov}(r_{t,2}, r_{t,3}). \tag{21.4}$$

可估计为

$$\hat{\sigma}_m^2 = \frac{n}{n-1}\sum_{i=1}^{n}(r_{t,i}-\bar{r}_t)^2 + 2\sum_{i=1}^{n-1}(r_{t,i}-\bar{r}_t)(r_{t,i+1}-\bar{r}_t). \tag{21.5}$$

这样用高频数据估计低频收益率波动率的方法简单易懂，但问题也不少：

- 日收益率的模型未知，使得 $\mathrm{Var}(r_{t,i}|F_{t-1})$ 和 $\mathrm{Cov}(r_{t,i}, r_{t,j}|F_{t-1})$ 可能有复杂的结构。假设 $\{r_{t,i}\}$ 为独立同分布白噪声或者 MA 可能是不充分的模型。
- 如果用日数据估计月数据的波动率，一个月大约有 21 个交易日，样本量较小，使得方差和协方差估计的精度不高。估计的精度依赖于 $\{r_{t,i}\}$ 的动态结构及其分布，如果 $\{r_{t,i}\}$ 有较高的超额峰度和较高的序列相关性，用(21.3)和(21.5)估计 $\mathrm{Var}(r_t^m|F_{t-1})$ 可能是不相合的，参见 Bai, X., Russell, J. R. 和 Tiao, G. C.(2004) 的未发表论文。

用日价格估计月波动率的 R 函数参见21.3。函数 `vold2m` 不计算第一个月的波动率。这实际上是将第一个月的日数据计算波动率后作为第二个月的波动率。

21.1.1 用日频数据估计标普 500 月对数收益率

用高频方法估计标普 500 的月对数收益率的波动率，时间期间为 1980 年 1 月到 2010 年 8 月，使用日频数据估计。

考虑三种方法的比较：

- 利用日频数据，假定日数据是独立同分布白噪声；
- 利用日频数据，假定日数据是 MA(1)；
- 利用月度数据，应用高斯 GARCH(1,1) 模型估计。

```
da <- read_table(
  "d-sp58010.txt", col_types=cols(.default=col_double()))
xts.sp5d <- xts(
  da[,-(1:3)],
  make_date(da[["Year"]], da[["Mon"]], da[["Day"]]))
```

利用蔡瑞胸教授的 R 函数计算月对数收益率的波动率，假定日对数收益率为独立同分布白噪声列：

```
mod1 <- vold2m(da[,c("Year", "Mon", "Day", "Adjclose")])
v1 <- mod1$volatility
```

结果为包含 `volatility` 和 `ndays` 的列表。

利用蔡瑞胸教授的 R 函数计算月对数收益率的波动率，假定日对数收益率为新息独立同分布的 MA(1) 序列：

```
#mod2 <- vold2m(da[,c("Mon", "Day", "Year", "Adjclose")], ma=1)
mod2 <- vold2m(da[, c("Year", "Mon", "Day", "Adjclose")], ma=1)
v2 <- mod2$volatility
```

读入标普 500 指数月度 OHLC 数据，从 1967 年 1 月到 2010 年 9 月：

```
da2 <- read_table(
  "m-sp56710.txt", col_types=cols(.default=col_double()))
xts.sp5m <- xts(
```

```
  da[,-(1:3)],
  make_date(da[["Year"]], da[["Mon"]], da[["Day"]]))
sp5 <- diff(log(da2[["Adjclose"]]))
```

对月度数据建立 GARCH 模型：

```
library(fGarch, quietly = TRUE)
mod3 <- garchFit( ~ 1 + garch(1,1), data=sp5, trace=FALSE)
summary(mod3)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = sp5, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x000001341fad10a8>
##  [data = sp5]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1       beta1
## 5.3471e-03  9.3264e-05  1.1422e-01  8.4864e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     5.347e-03   1.742e-03    3.069 0.002149 **
## omega  9.326e-05   4.859e-05    1.919 0.054942 .
## alpha1 1.142e-01   3.003e-02    3.804 0.000142 ***
## beta1  8.486e-01   3.186e-02   26.634  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  899.7817    normalized:  1.717141
##
```

```
## Description:
##  Tue May 30 17:04:06 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  172.5215  0
##  Shapiro-Wilk Test  R    W      0.9690782 4.63924e-09
##  Ljung-Box Test     R    Q(10)  11.17329  0.3441774
##  Ljung-Box Test     R    Q(15)  15.45101  0.4194444
##  Ljung-Box Test     R    Q(20)  17.5647   0.6160597
##  Ljung-Box Test     R^2  Q(10)  5.466784  0.857899
##  Ljung-Box Test     R^2  Q(15)  7.031542  0.9567686
##  Ljung-Box Test     R^2  Q(20)  8.200426  0.9904565
##  LM Arch Test       R    TR^2   5.629865  0.9335797
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.419014 -3.386484 -3.419129 -3.406275
```

```
v3 <- window(ts(volatility(mod3), start=c(1967, 2), frequency=12),
             start=c(1980,2), end=c(2010,8))
```

从 GARCH 拟合结果看出了高斯分布假设不成立以外模型是充分的。拟合的模型为

$$
\begin{aligned}
r_t^m =& 0.0053 + a_t, \quad a_t = \sigma_t \varepsilon_t, \quad \varepsilon_t \sim \mathrm{N}(0,1) \\
\sigma_t^2 =& 0.00009326 + 0.1142 a_{t-1}^2 + 0.8486 \sigma_{t-1}^2
\end{aligned}
$$

下面比较三种方法估计的波动率：

```
plot(v1, xlab="Year", ylab="Volatility",
     main="Using Daily Price WN Assumption",
     ylim=c(0, 0.3))
```

```
plot(v2, xlab="Year", ylab="Volatility",
     main="Using Daily Price MA(1) Assumption",
     ylim=c(0, 0.3))
```

```
plot(v3, xlab="Year", ylab="Volatility",
     main="Using Monthly Price GARCH(1,1)",
     ylim=c(0, 0.3))
```

可以看出日频数据估计的波动率厚尾更为严重。将估计的三个波动率序列画在同一坐标系中：

```
plot(c(time(v1)), c(v1), type="l", xlab="Year",
     ylab="Volatility",
     main="Comparing 3 volatility series",
```

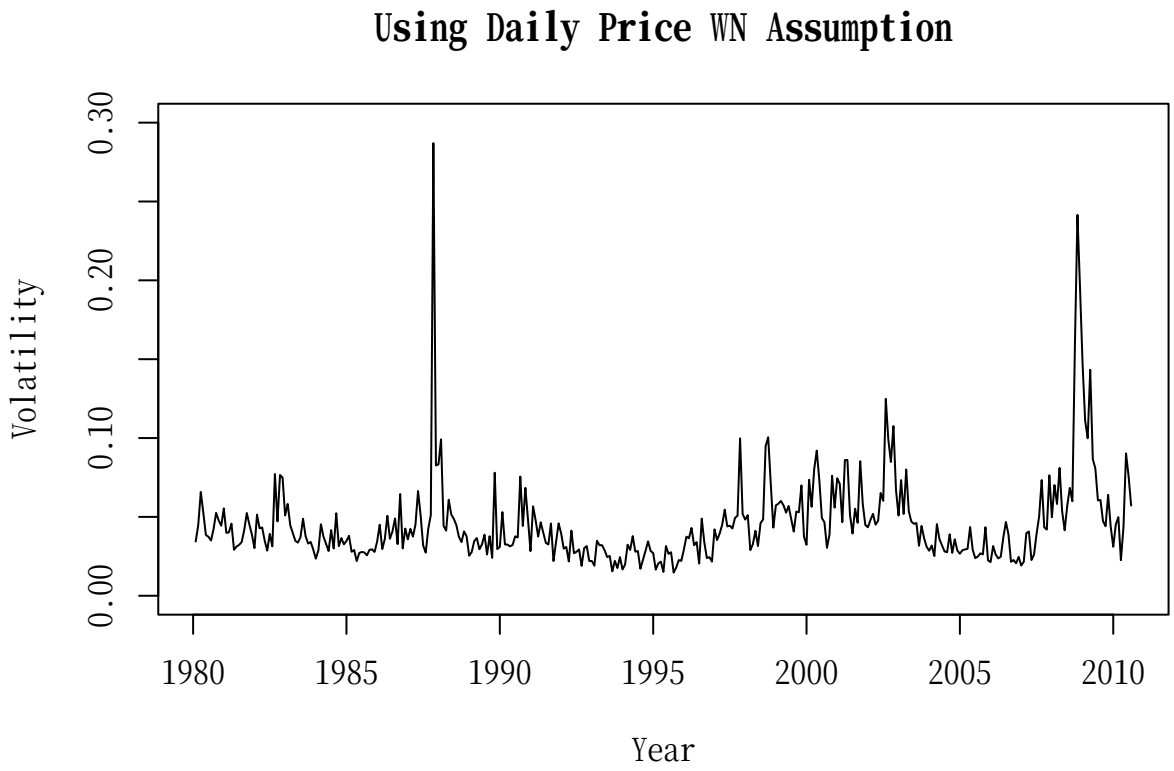


图 21.1: 标普 500 月波动率用假定白噪声日频数据估计

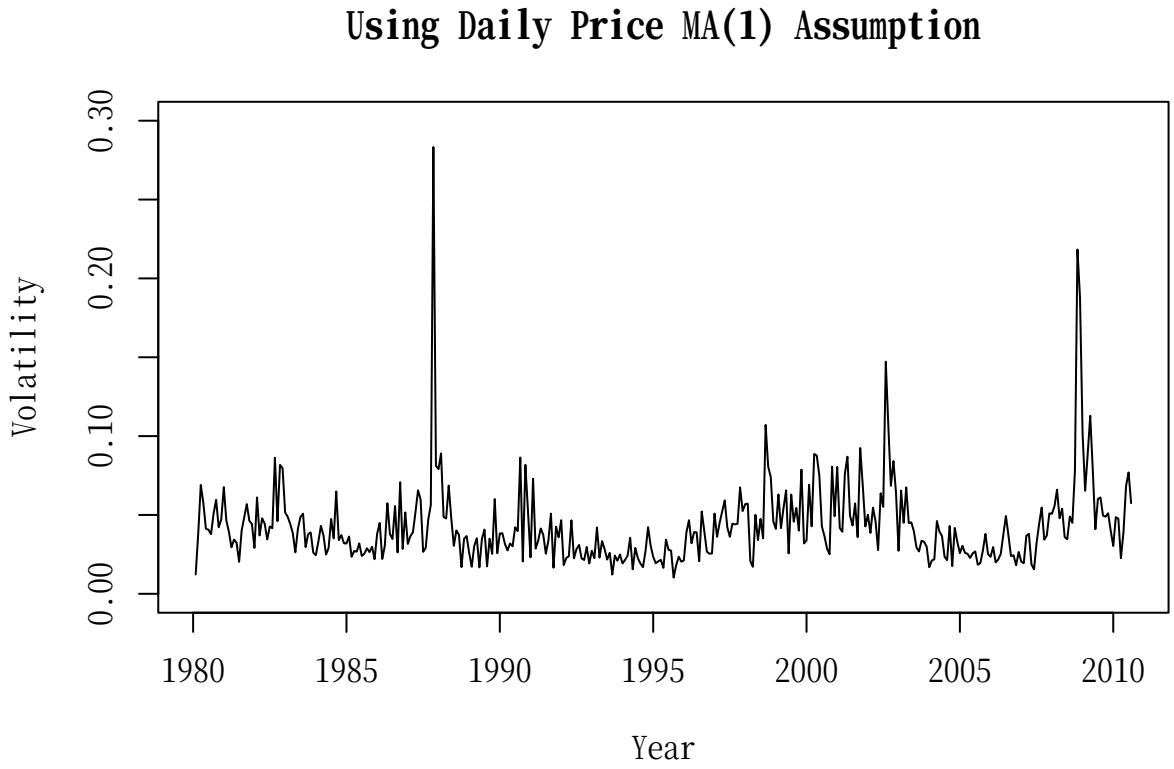


图 21.2: 标普 500 月波动率用假定 MA(1) 日频数据估计

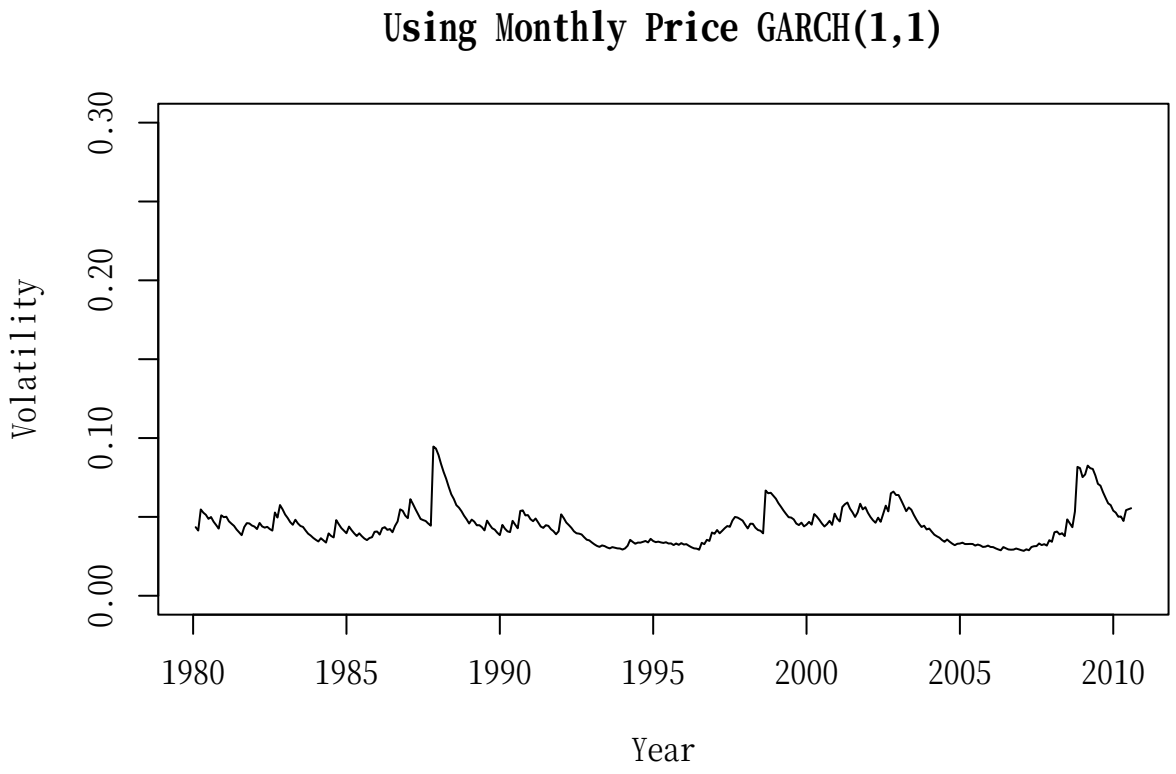


图 21.3: 标普 500 月波动率用 GARCH 模型估计

```
     ylim=c(0, 0.3),
     col="blue")
lines(c(time(v1)), c(v2), col="cyan")
lines(c(time(v1)), c(v3), col="black")
legend("top", lty=1, col=c("blue", "cyan", "black"),
       legend=c(" 日频白噪声假定", " 日频 MA(1) 假定", " 月频 GARCH(1,1) 估计"))
```

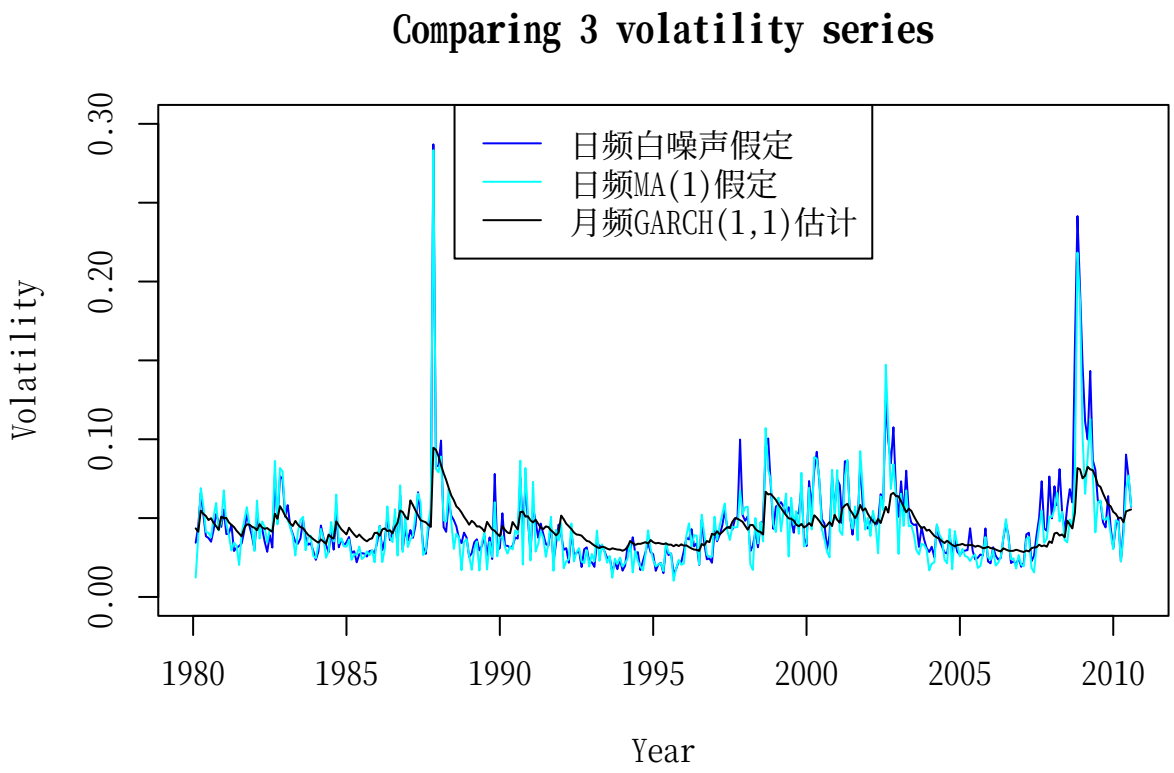


图 21.4: 标普 500 用三种方法估计的月波动率

从图21.4看，日频数据结果仅在很高的时候比月频结果高很多，一般情况下估计结果大小近似；估计的走势基本一致。

比较三种方法得到的月波动率分布密度：

```
den1 <- density(c(v1), from=0.01, to=0.29)
den2 <- density(c(v2), from=0.01, to=0.29)
den3 <- density(c(v3), from=0.01, to=0.29)
plot(den1, xlab="Volatility", ylab="Density",
     main="Comparing 3 volatility densities",
     ylim=c(0, 40),
     col="blue")
lines(den2, col="cyan")
lines(den3, col="black")
legend("topright", lty=1, col=c("blue", "cyan", "black"),
```

```
      legend=c(" 日频白噪声假定", " 日频 MA(1) 假定", " 月频 GARCH(1,1) 估计"))
```

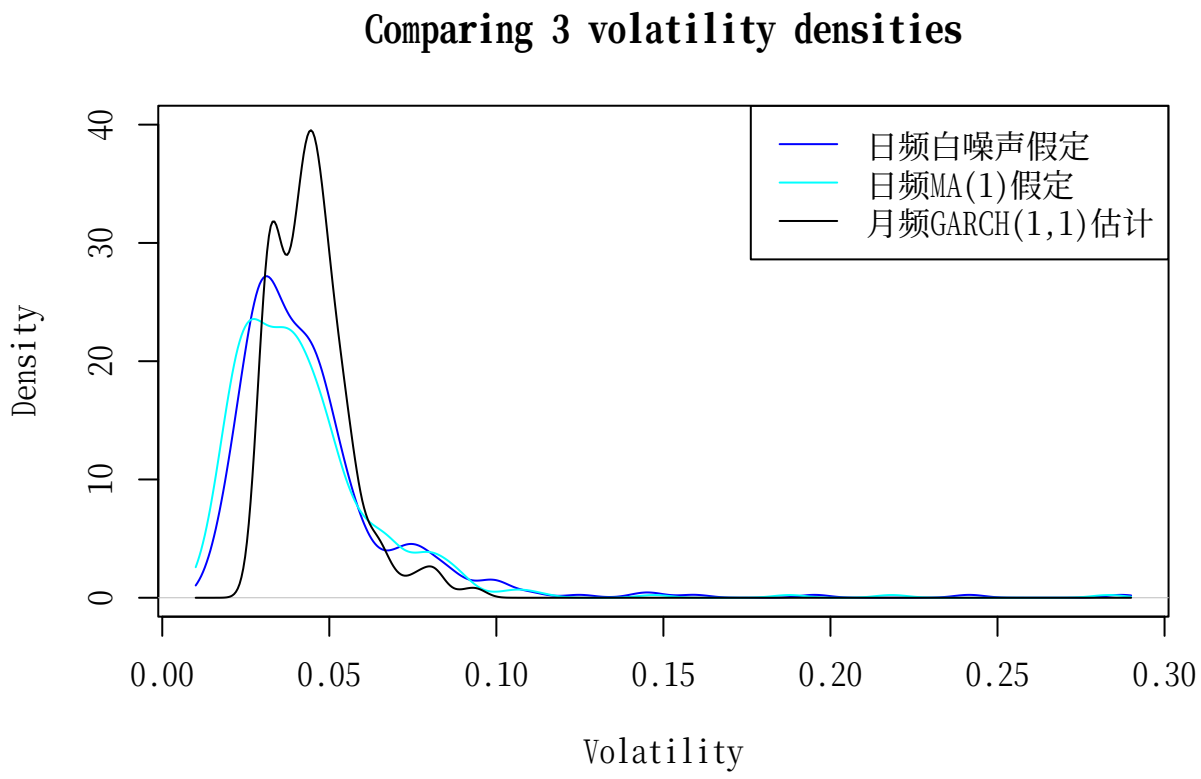


图 21.5: 标普 500 用三种方法估计的月波动率分布密度

从图21.5可以看出日频估计的波动率更为厚尾，月频数据估计的结果分布比较集中。取值除了少数极端值以外，取值范围差别不大。

21.2 使用 OHLC 数据

许多资产都有 OHLC 数据，可以用这样的数据辅助估计波动率。

对于一项资产，定义如下变量：

- O_t: 第 t 个交易日的开盘价的对数值；
- H_t: 第 t 个交易日的最高价的对数值；
- L_t: 第 t 个交易日的最低价的对数值；
- C_t: 第 t 个交易日的收盘价的对数值；
- f：一个自然日中闭市时间长度的比例，是一个 $[0,1]$ 之间的数；
- F_{t-1}: 表示截止到 $t-1$ 个交易日的所有公开信息，但数学上应该是参与建模的所有可观测变量截止到 $t-1$ 交易日的变量的 σ 代数。

当价格为对数价格时，日对数收益率条件方差或波动率为 $\sigma_t^2 = E[(C_t - C_{t-1})^2|F_{t-1}]$。(假定对数收益率均值为 0)。

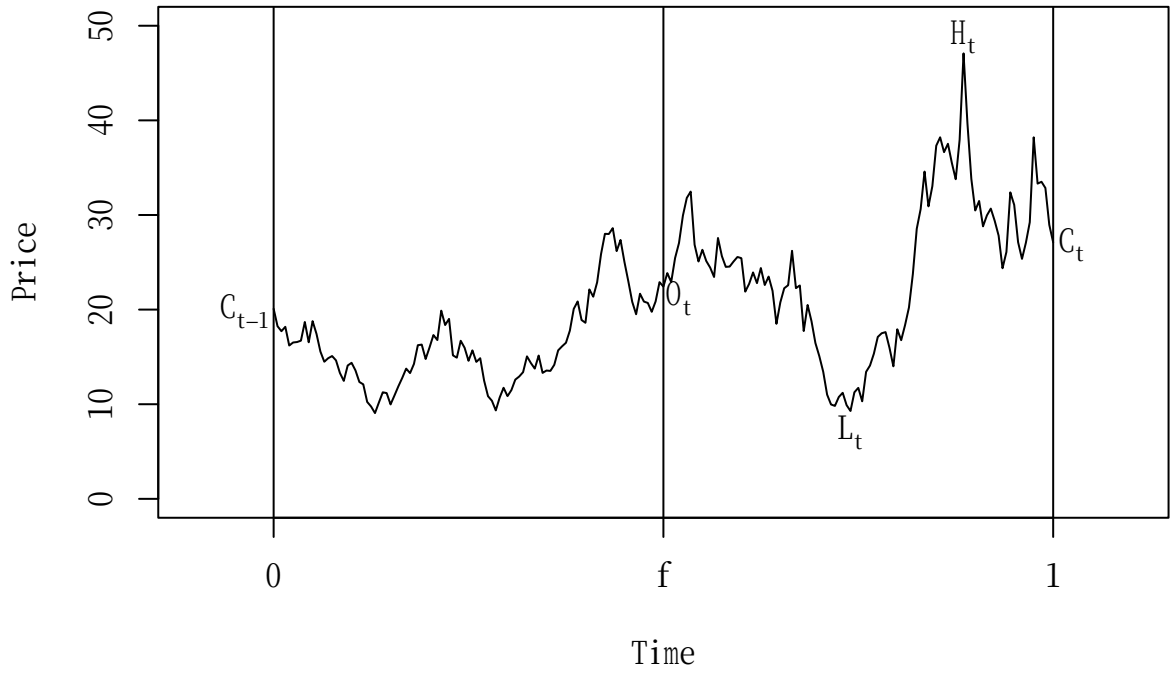


图 21.6: OHLC 数据形成

(Garman & Klass, 1980) 考虑了 σ_t^2 的多种估计，论文假定价格服从一个不带漂移的扩散过程，可比较的估计有

$$\begin{aligned}
\hat{\sigma}_{0t}^2 =& (C_t - C_{t-1})^2, \\
\hat{\sigma}_{1t}^2 =& \frac{(O_t - C_{t-1})^2}{2f} + \frac{(C_t - O_t)^2}{2(1-f)}, \\
\hat{\sigma}_{2t}^2 =& \frac{(H_t - L_t)^2}{4\ln 2} \approx 0.3607(H_t - L_t)^2, \\
\hat{\sigma}_{3t}^2 =& 0.17\frac{(O_t - C_{t-1})^2}{f} + 0.83\frac{(H_t - L_t)^2}{(1-f)4\ln 2}, \\
\hat{\sigma}_{5t}^2 =& 0.5(H_t - L_t)^2 - (2\ln 2 - 1)(C_t - O_t)^2 \approx 0.5(H_t - L_t)^2 - 0.386(C_t - O_t)^2, \\
\hat{\sigma}_{6t}^2 =& 0.15\frac{(O_t - C_{t-1})^2}{f} + 0.88\frac{\hat{\sigma}_{5t}^2}{1-f}.
\end{aligned}$$

其中用到 f 的都需要 $0 < f < 1$。论文还考虑了更复杂的 $\hat{\sigma}_{4t}^2$ 公式，但与 $\hat{\sigma}_{5t}^2$ 接近。$\hat{\sigma}_{2t}^2$ 估计方法是 (Parkinson, 1980) 提出的。

定义波动率估计的效率因子为

$$\text{Eff}(\hat{\sigma}_{it}^2) = \frac{\text{Var}(\hat{\sigma}_{0t}^2)}{\text{Var}(\hat{\sigma}_{it}^2)}.$$

(Garman & Klass, 1980) 发现对于价格服从简单扩散模型的情形，$i = 1, 2, 3, 5, 6$ 时 $\text{Eff}(\hat{\sigma}_{it}^2)$ 分别为 2, 5.2, 6.2, 7.4 和 8.4，即 $\hat{\sigma}_{6t}^2$ 的估计效率最高。

回到对数收益率。定义

- $o_t = O_t - C_{t-1}$ 为标准化对数开盘价；
- $u_t = H_t - O_t$ 为标准化对数最高价；
- $d_t = L_t - O_t$ 为标准化对数最低价；
- $c_t = C_t - O_t$ 为标准化对数收盘价。

设有 n 天数据，波动率在这个期间保持恒定，(Yang & Zhang, 2000) 提出了如下的波动率稳健估计

$$\hat{\sigma}_{yz}^2 = \hat{\sigma}_o^2 + k\hat{\sigma}_c^2 + (1-k)\hat{\sigma}_{rs}^2. \tag{21.6}$$

其中

$$\begin{aligned}
\hat{\sigma}_o^2 =& \frac{1}{n-1}\sum_{t=1}^{n}(o_t - \bar{o})^2, \\
\hat{\sigma}_c^2 =& \frac{1}{n-1}\sum_{t=1}^{n}(c_t - \bar{c})^2, \\
\hat{\sigma}_{rs}^2 =& \frac{1}{n}\sum_{t=1}^{n}\left\{u_t(u_t - c_t) + d_t(d_t - c_t)\right\}, \\
k =& \frac{0.34}{1.34 + (n+1)/(n-1)}.
\end{aligned}$$

估计 $\hat{\sigma}_{rs}^2$ 由 Rogers 和 Satchell(1991) 提出，选择 k 使得 $\hat{\sigma}_{yz}^2$ 的标准误差最小，$\hat{\sigma}_{yz}^2$ 是三种估计的线性组合。

因为需要使用 n 天数据计算，所以实际计算时选定一个窗宽 n，每次向前滚动一天地用窗内的价格数据计算，得到的 $\hat{\sigma}_{yz}^2$ 作为滚动窗内最后一天的波动率平方的估计。

这些滚动计算的算法都有相位对齐问题，一般对齐在滚动窗最后一天，或者最后一天的下一天。这样估计的结果一般会略有滞后，因为最合理的对齐日期是在滚动窗的中间位置。

(Alizadeh et al., 2002) 提出了用第 t 天的变化范围 $H_t - L_t$ 估计波动率的方法。但是实际中股票的价格仅在有交易的时刻被观测到，所以实际的 H_t 和 L_t 可能是未被观测到的，观测的价格变化范围可能低估了实际变化范围，从而低估波动率。对于交易频繁的股票，这种偏差可以忽略；交易不够频繁的股票则需要考虑这种偏差的影响。

21.2.1 用 OHLC 数据估计标普 500 日对数收益率的波动率

标普 500 指数的 OHLC 数据，从 1980-01-03 到 2010-08-31，共 7737 个交易日的数据。估计日对数收益率波动率。

```
chartSeries(
  xts.sp5d, subset="2010-06/2010-08",
  type="bars", theme="white", TA=NULL,
  main="SP500 Daily Price",
  major.ticks="months",
  grid.ticks.on="months")
```

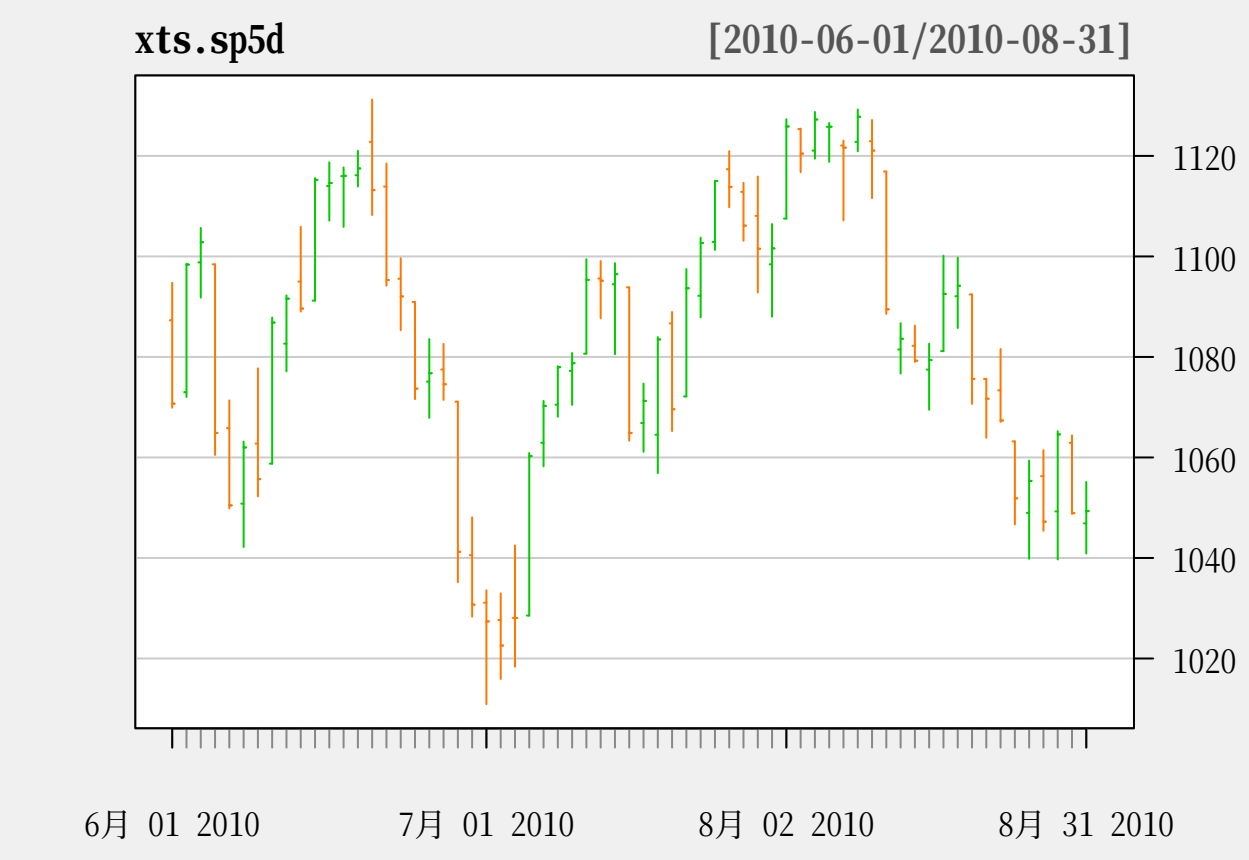


图 21.7: 标普 500 指数序列日数据在 2010 年 6-8 月

图21.7为标普指数日数据在 2010.6 到 8 月的 OHLC 图形。竖线条表示范围，左端的短线表示开盘价，右端的短线表示收盘价。

使用三种方法估计波动率：

- 利用 Yang-Zhang 方法（式(21.6)），取滑动窗口，窗宽 $n = 63$，约三个月；
- 利用 Yang-Zhang 方法但取 $n = 32$，与上一方法比较以查看窗宽选择对估计的影响大小；

- 为日对数收益率序列建立 ARMA-GARCH 模型估计波动率。

(Yang & Zhang, 2000) 式(21.6)估计的 R 函数，利用滑动窗口计算，计算得到的波动率的时刻与窗口最右端对齐：

```
## 输入 x 为 xts 类型的时间序列，bw 为滑动窗口宽度
volatility.ohlc.yz <- function(x, bw=63){
  times <- index(x)
  nobs <- length(times)
  stdop <- log(Op(x)) - log(lag(Cl(x)[,1]))
  stdhi <- log(Hi(x)) - log(Op(x))
  stdlo <- log(Lo(x)) - log(Op(x))
  stdcl <- log(Cl(x)[,1]) - log(Op(x))
  k <- 0.34 / (1.34 + (bw+1)/(bw-1))

  vo <- numeric(nobs); vo[1:bw] <- NA
  vc <- vo
  vrs <- vo
  vyz <- vo

  ## 滑动计算
  for(it in (bw+1):nobs){
    ind <- (it-bw+1):it
    vo[it] <- var(stdop[ind], na.rm=TRUE)
    vc[it] <- var(stdcl[ind], na.rm=TRUE)
    vrs[it] <- 1/bw*sum(stdhi[ind]*(stdhi[ind] - stdcl[ind]) +
                        stdlo[ind]*(stdlo[ind] - stdcl[ind]))
  }
  vyz <- vo + k*vc + (1-k)*vrs

  xts(cbind(Volatility=c(sqrt(vyz))), times)
}
```

利用式(21.6)方法，取滑动窗口，窗口大小 $n = 63$，估计波动率：

```
mod4 <- volatility.ohlc.yz(xts.sp5d, bw=63)
```

估计的波动率序列：

```
plot(mod4, type="l", col="red",
     ylim=c(0, 0.07),
     main="Yang-Zhang(BW=63)",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years")
```

利用式(21.6)方法，取滑动窗口，窗口大小 $n = 32$，估计波动率：

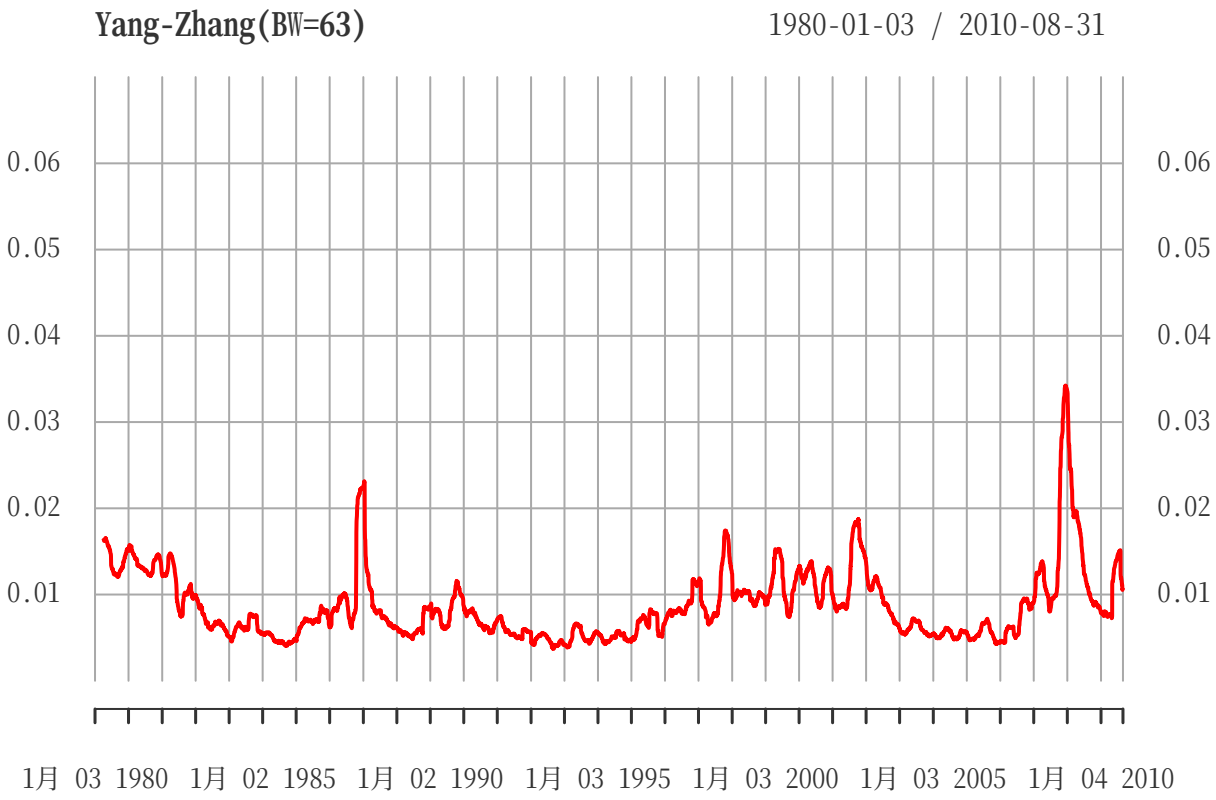


图 21.8: 63 天滑动计算的波动率

```
mod5 <- volatility.ohlc.yz(xts.sp5d, bw=32)
```

估计的波动率序列：

```
plot(mod5, type="l", col="red",
     ylim=c(0, 0.07),
     main="Yang-Zhang(BW=32)",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years")
```

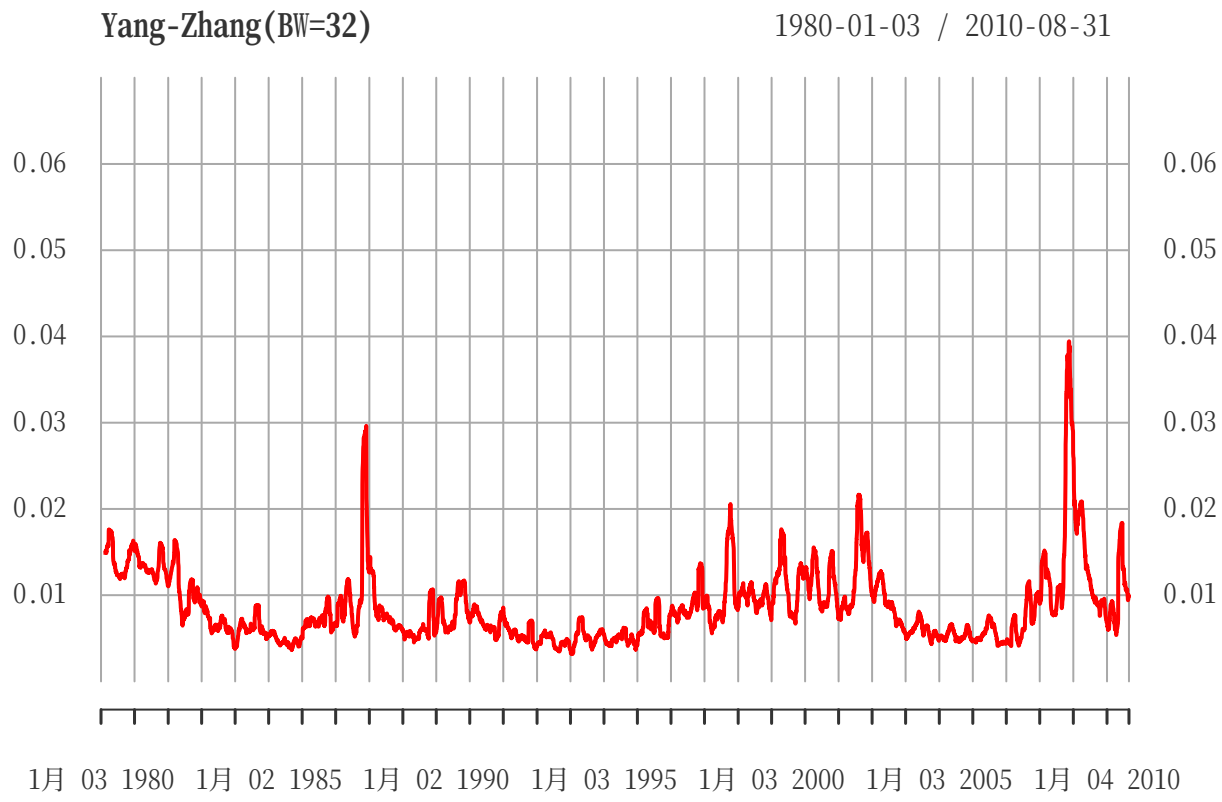


图 21.9: 32 天滑动计算的波动率

下面对日收盘价的对数收益率拟合一个 ARMA-GARCH 模型，并估计波动率。

```
library(fGarch, quietly = TRUE)
mod6 <- garchFit(
  ~ 1 + arma(4,0) + garch(1,1),
  data=diff(log(coredata(xts.sp5d)[,"Close",drop=TRUE])),
  trace=FALSE)
summary(mod6)
```

```
##
## Title:
##  GARCH Modelling
```

```
##
## Call:
##  garchFit(formula = ~1 + arma(4, 0) + garch(1, 1), data = diff(log(coredata(xts.sp5d)[,
##     "Close", drop = TRUE])), trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + arma(4, 0) + garch(1, 1)
## <environment: 0x000001341dcecb48>
##  [data = diff(log(coredata(xts.sp5d)[, "Close", drop = TRUE]))]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##          mu          ar1          ar2          ar3          ar4        omega
##  5.5450e-04   1.4467e-02  -1.1101e-02  -2.2045e-02  -3.3974e-02   1.2481e-06
##      alpha1        beta1
##  7.5577e-02   9.1579e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## mu      5.545e-04   9.545e-05    5.810 6.26e-09 ***
## ar1     1.447e-02   1.218e-02    1.187  0.23510
## ar2    -1.110e-02   1.204e-02   -0.922  0.35655
## ar3    -2.205e-02   1.200e-02   -1.836  0.06629 .
## ar4    -3.397e-02   1.200e-02   -2.830  0.00465 **
## omega   1.248e-06   2.036e-07    6.131 8.74e-10 ***
## alpha1  7.558e-02   5.628e-03   13.428  < 2e-16 ***
## beta1   9.158e-01   6.294e-03  145.511  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  25058.57    normalized:  3.239216
##
## Description:
##  Tue May 30 17:04:12 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
```

```
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  7046.431  0
##  Shapiro-Wilk Test  R    W      NA        NA
##  Ljung-Box Test     R    Q(10)  8.763866  0.5546507
##  Ljung-Box Test     R    Q(15)  20.32413  0.159848
##  Ljung-Box Test     R    Q(20)  23.147    0.2816342
##  Ljung-Box Test     R^2  Q(10)  3.647526  0.9618503
##  Ljung-Box Test     R^2  Q(15)  5.382367  0.9883641
##  Ljung-Box Test     R^2  Q(20)  8.517023  0.9878564
##  LM Arch Test       R    TR^2   4.441792  0.9740826
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -6.476364 -6.469173 -6.476366 -6.473898
```

估计的模型为

$$
\begin{aligned}
r_t =& 0.00055 + 0.0145 r_{t-1} - 0.0111 r_{t-2} - 0.0221 r_{t-3} - 0.0340 r_{t-4} + a_t, \\
a_t =& \sigma_t \varepsilon_t, \quad \varepsilon_t \sim \mathrm{N}(0,1) \\
\sigma_t^2 =& 1.248 \times 10^{-6} + 0.0756 a_{t-1}^2 + 0.9158 \sigma_{t-1}^2
\end{aligned}
$$

除了高斯分布假定外模型是充分的。估计的波动率序列图：

```
plot(
  xts(volatility(mod6), index(xts.sp5d)[-1]),
  type="l", col="red",
  ylim=c(0, 0.07),
  main="ARMA-GARCH Volatility Estimate of BP500",
  major.ticks="years", minor.ticks=NULL,
  grid.ticks.on="years"
)
```

将三种方法得到的波动率画在同一坐标系中比较：

```
plot(index(xts.sp5d[-1]), c(coredata(mod4))[-1],
     type="l", xlab="Year",
     ylab="Volatility",
     main="Comparing 3 volatility series",
     ylim=c(0, 0.07),
     col="blue")
lines(index(xts.sp5d[-1]), c(coredata(mod5))[-1],
      col="cyan")
lines(index(xts.sp5d[-1]), c(volatility(mod6)),
      col="black")
legend("top", lty=1, col=c("blue", "cyan", "black"),
       legend=c("Yang-Zhang BW=63", "Yang-Zhang BW=32", "ARMA-GARCH"))
```

从图21.11看出，三种方法估计的波动率在非极端值时比较接近，ARMA-GARCH 估计的极端值更大，Yang-Zhang

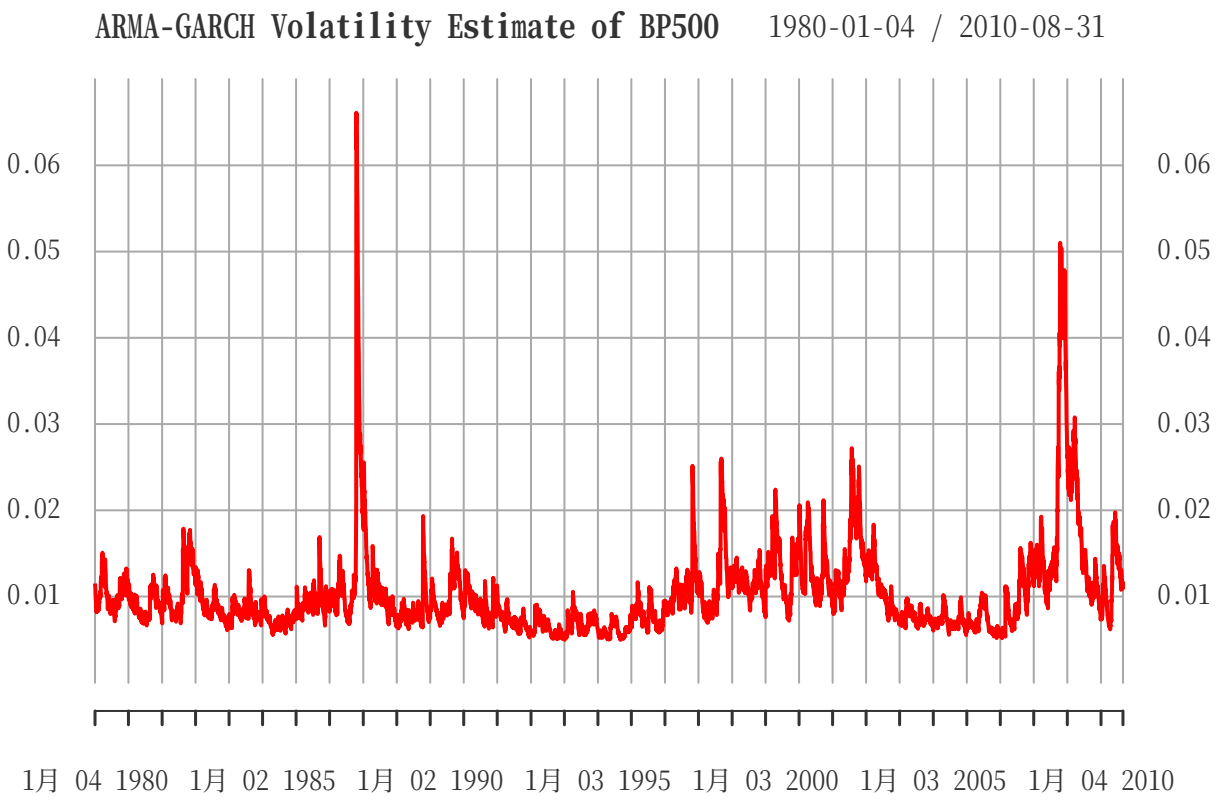


图 21.10: ARMA-GARCH 得到的 SP500 日波动率

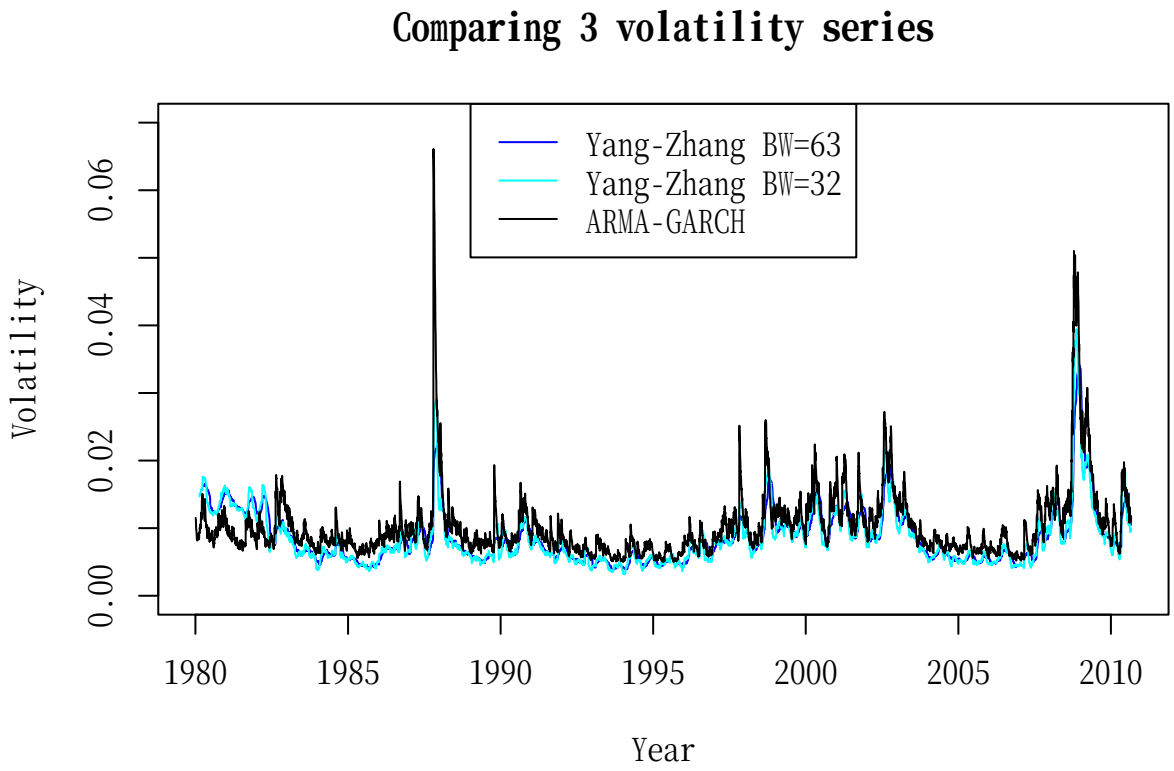


图 21.11: 标普 500 用三种方法估计的日波动率

方法窗口宽度 63 和 32 的结果基本没有差别。

21.3 附录：用日数据估计月波动率的 R 函数

```
"vold2m" <- function(da, ma=0){
  # Compute the monthly volatility from daily prices
  # ma = 0: Assume no serial correlations in the daily returns
  # ma = 1: Assume one-lag of serial correlation in the daily returns
  # da: T-by-4 data matrix in the format [Year, Month, Day, Price]
  #
  if(!is.matrix(da)) da = as.matrix(da)
  T = nrow(da)
  pr = log(da[,4])
  rtn = c(NA, diff(pr))
  ## 不同年月个数
  nmonths <- length(unique(da[-1,1]*100 + da[-1,2]))
  vol = numeric(nmonths-1)
  cnt <- numeric(nmonths-1)

  ## x 保存当前月份的收益率
  x <- numeric(31)
  id <- 0
  im <- 0
  pmon = da[2,2]
  first <- TRUE
  for (t in 2:T){
    if(da[t,2] == pmon){
      id <- id + 1
      x[id] <- rtn[t]
    } else { # new month
      if(first){
        firstmon <- c(da[t, 1:2])
        first <- FALSE
      }
      im <- im + 1
      n <- id
      v1 = n * var(x[1:n])
      v2 = 0
      if(ma==1) v2 = cov1(x[1:n])
      v1 = v1 + v2
      vol[im] <- sqrt(v1)
      cnt[im] <- n
```

```
      id <- 1
      x[1] <- rtn[t]
      pmon = da[t,2]
    }
  } # for t

  vold2m <- list(
    volatility = ts(vol, start=firstmon, frequency=12),
    ndays=cnt)
}

## 计算滞后 1 的中心化叉积 2 倍
"cov1" <- function(x){
  x[] <- x[] - mean(x)
  n <- length(x)
  2 * sum(x[1:(n-1)] * x[2:n])
}
```

Chapter 22

波动率模型的应用

22.1 GARCH 波动率期限结构

下面研究 GARCH 模型导致的波动率期限结构，比如，日对数收益率的波动率与月对数收益率的波动率的关系。以时间 t 为基础，距离 t 时刻 h 期（比如 h 个交易日）的对数收益率为

$$r_{t,h} = \sum_{i=1}^{h} r_{t+i}$$

于是

$$E(r_{t,h}|F_t) = \sum_{i=1}^{h} E(r_{t+i}|F_t)$$

h 期的条件方差，即波动率平方为

$$\mathrm{Var}(r_{t,h}|F_t) = \sum_{i=1}^{h} \mathrm{Var}(r_{t+i}|F_t) + \sum_{1\le i<j\le h} \mathrm{Cov}(r_{t+i}, r_{t+j}|F_t)$$

实证分析和有效市场理论都认为协方差接近零，所以可假定

$$\mathrm{Var}(r_{t,h}|F_t) = \sum_{i=1}^{h} \mathrm{Var}(r_{t+i}|F_t)$$

对于 GARCH 模型，这就是

$$\sigma_{t,h}^2 = \mathrm{Var}(r_{t,h}|F_t) = \sum_{\ell=1}^{h} \sigma_t^2(\ell)$$

其中 $\sigma_{t,h}^2$ 表示以 h 期为单位的基于时刻 t 计算的条件方差，即 h 期的对数收益率的波动率平方，$\sigma_t^2(\ell)$ 是基于时刻 t 的单期对数收益率的波动率的超前 ℓ 步预测。

考虑 GARCH(1,1) 模型的超前 ℓ 步预测问题。模型为：

$$\sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \tag{22.1}$$

其中 $\alpha_0 > 0$, $\alpha_1, \beta_1 \in [0, 1)$, $\alpha_1 + \beta_1 < 1$。已证明

$$\mathrm{Var}(a_t) = \sigma^2 = \frac{\alpha_0}{1 - \alpha_1 - \beta_1}$$

可以将(22.1)改写成

$$(\sigma_t^2 - \sigma^2) = \alpha_1(a_{t-1}^2 - \sigma^2) + \beta_1(\sigma_{t-1}^2 - \sigma^2) \tag{22.2}$$

这个式子可以看成是 a_t^2 的一步预测 $E(a_t^2|F_{t-1})$ 与长期预测 σ^2 的偏离的模型。

波动率的基于 F_t 的超前一步预测为

$$\sigma_t^2(1) = \alpha_0 + \alpha_1 a_t^2 + \beta_1 \sigma_t^2$$

超前 ℓ 步预测为

$$\sigma_t^2(\ell) = \alpha_0 + (\alpha_1 + \beta_1)\sigma_t^2(\ell - 1),\ \ell = 2, 3, ...$$

以 $\sigma^2 = \alpha_0/(1 - \alpha_1 - \beta_1)$ 代入上式可变成

$$\begin{aligned}\sigma_t^2(\ell) - \sigma^2 =& (\alpha_1 + \beta_1)[\sigma_t^2(\ell - 1) - \sigma^2] \\ =& (\alpha_1 + \beta_1)^{\ell-1}[\sigma_t^2(1) - \sigma^2]\end{aligned}$$

当 $\alpha_1 + \beta_1 < 1$ 时令 $\ell \to \infty$，有 $\sigma_t^2(\ell) \to \sigma^2$，即波动率平方的长期预测是均值回转的。均值回转的速度可以用半衰期

$$\ell_m = \ln 0.5 / \ln(\alpha_1 + \beta_1)$$

度量。

对 CAT、CSCO 和 GE 三支股票日对数收益率建模发现，$\alpha_1 + \beta_1$ 很接近于 1（分别是 0.9776, 0.9770, 0.9991），使得波动率持续性很强，半衰期分别为 31、30、770。

有了上面的多期预测公式，GARCH 模型就可以用来从单期对数收益率波动率换算多期对数收益率波动率。

多期波动率一定大于单期波动率，为了不同期限的波动率可比，可将其标准化为年化波动率，公式为

$$\sigma_{t,h,a} = \sqrt{\frac{252}{h}}\sigma_{t,h}$$

其中 $\sigma_{t,h}$ 是基于时刻 t 预测的 h 期波动率，是从第 t 期到第 $t + h$ 期的多期对数收益率的条件方差；$\sigma_{t,h,a}$ 是此多期波动率年化的结果。

下面写一个从 GARCH(1,1) 模型估计计算 h 期波动率，并将其年化的函数。输入为 GARCH(1,1) 模型估计结果，a_t 序列，h 是一到多个多期数值，输出一个年化多期波动率的矩阵，其中第 t 行代表从时间 t 对 $t + h$ 时刻的多期波动率预测，注意：输出的不同 h 在同一时刻 t 代表的不是该时刻的某个收益率的年化波动率，而是用前一期 (相隔 h 日) 的波动率对其进行的预测值。

```
volatility_interval <- function(modres, at, h=1){
  vola <- volatility(modres)
  at <- at - mean(at)
  co <- coef(modres)
  omega <- co["omega"]
  alpha1 <- co["alpha1"]
  beta1 <- co["beta1"]
  ab <- alpha1 + beta1
  nt <- length(vola)
  predi <- numeric(nt)
  resm <- matrix(0.0, nrow=nt, ncol=length(h))
```

```
  hmax <- max(h)
  for(j in seq(along=h)){
    hj <- h[j]
    predi[] <-
      omega + alpha1*at^2 + beta1*vola^2
    resm[,j] <- predi
    if(hj > 1){
      for(jj in 2:hj){
        predi <- omega + ab * predi
        resm[,j] <- resm[,j] + predi
      }
    }
    resm[,j] <- sqrt(252/hj*resm[,j])
  }

  resm
}
```

22.1.1 CAT 股票日对数收益率的波动率期限结构

作为例子，考虑 CAT(卡特彼勒) 股票日对数收益率的期限结构，时间从 2001-01-02 到 2010-12-31。以百分之一单位表示。

```
d.cat <- read_table(
  "d-c2c-0110.txt",
  col_types=cols(.default=col_double())
)
d.sp500 <- read_table(
  "d-sp500-0110.txt",
  col_names = FALSE,
  col_types=cols(.default=col_double())
)
time.cat <- seq(2001, 2010, length.out=nrow(d.cat))
rtn.cat <- log(1 + d.cat[["CAT"]])*100
rtn.csco <- log(1 + d.cat[["CSCO"]])*100
rtn.sp500 <- d.sp500[[1]]*100
```

CAT 股票日对数收益率时间序列图形：

```
plot(time.cat, rtn.cat,
  type="l", xlab="Year", ylab="ln(return)")
```

CAT 股票日对数收益率时间序列 ACF：

```
forecast::Acf(rtn.cat, lag.max = 30, main="")
```

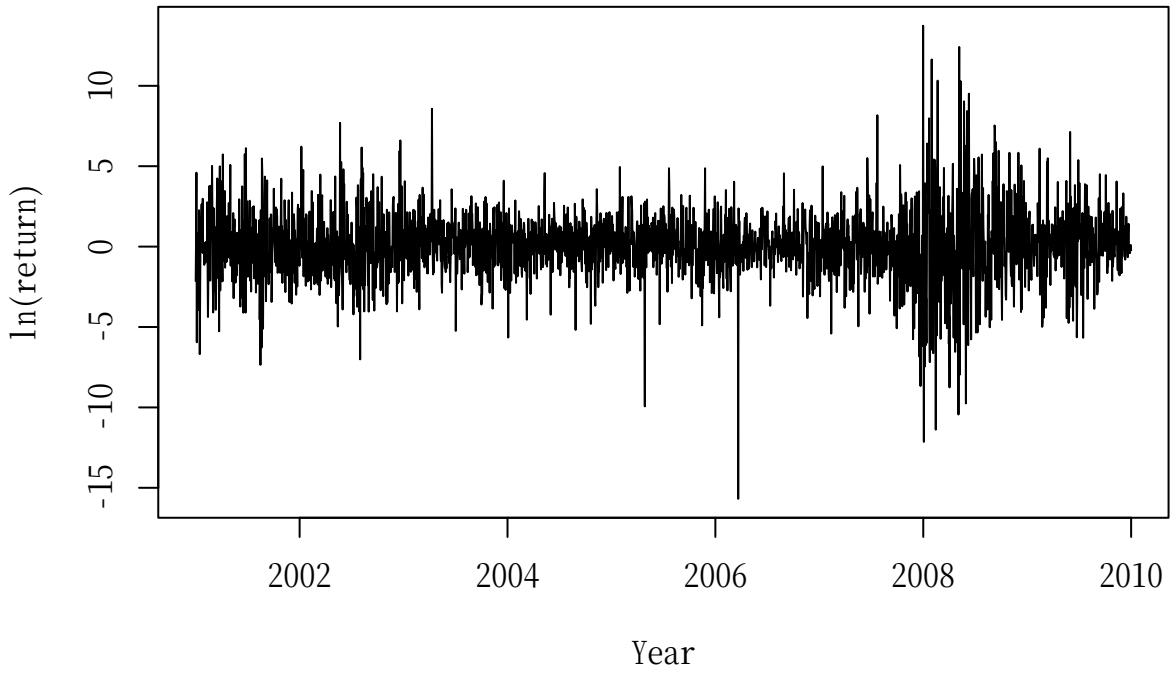


图 22.1: CAT 股票日对数收益率

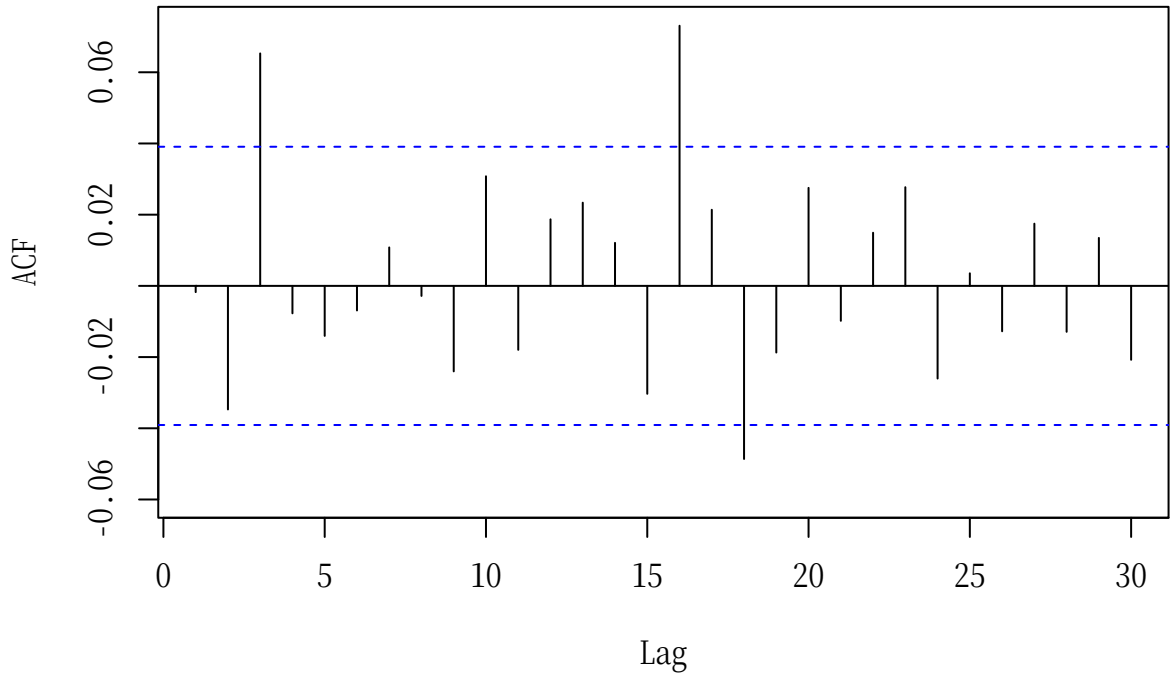


图 22.2: CAT 股票日对数收益率 ACF

建立 GARCH(1,1) 模型：

```
library(fGarch)
mod1.cat <- garchFit(~ 1 + garch(1,1), data=rtn.cat, trace=FALSE)
summary(mod1.cat)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = rtn.cat, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224ac1785e8>
##  [data = rtn.cat]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1     beta1
## 0.103702  0.095669  0.053109  0.924455
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.10370     0.03713    2.793  0.00522 **
## omega    0.09567     0.03385    2.826  0.00471 **
## alpha1   0.05311     0.01134    4.682 2.84e-06 ***
## beta1    0.92446     0.01787   51.739  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -5297.634    normalized:  -2.106415
##
## Description:
##  Tue May 30 17:04:17 2023 by user: Lenovo
##
##
```

```
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  2397.079  0
##  Shapiro-Wilk Test  R    W      0.9720346 0
##  Ljung-Box Test     R    Q(10)  11.99378  0.2854729
##  Ljung-Box Test     R    Q(15)  16.40952  0.355369
##  Ljung-Box Test     R    Q(20)  21.94146  0.3436959
##  Ljung-Box Test     R^2  Q(10)  1.323393  0.9993882
##  Ljung-Box Test     R^2  Q(15)  3.737861  0.9984771
##  Ljung-Box Test     R^2  Q(20)  4.956939  0.9997407
##  LM Arch Test       R    TR^2   2.67451   0.9974382
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 4.216011 4.225283 4.216006 4.219376
```

模型基本充分。

```
plot(time.cat, volatility(mod1.cat), type="l",
     main="CAT 股票日对数收益率波动率 GARCH(1,1) 估计",
     xlab=" 年")
```

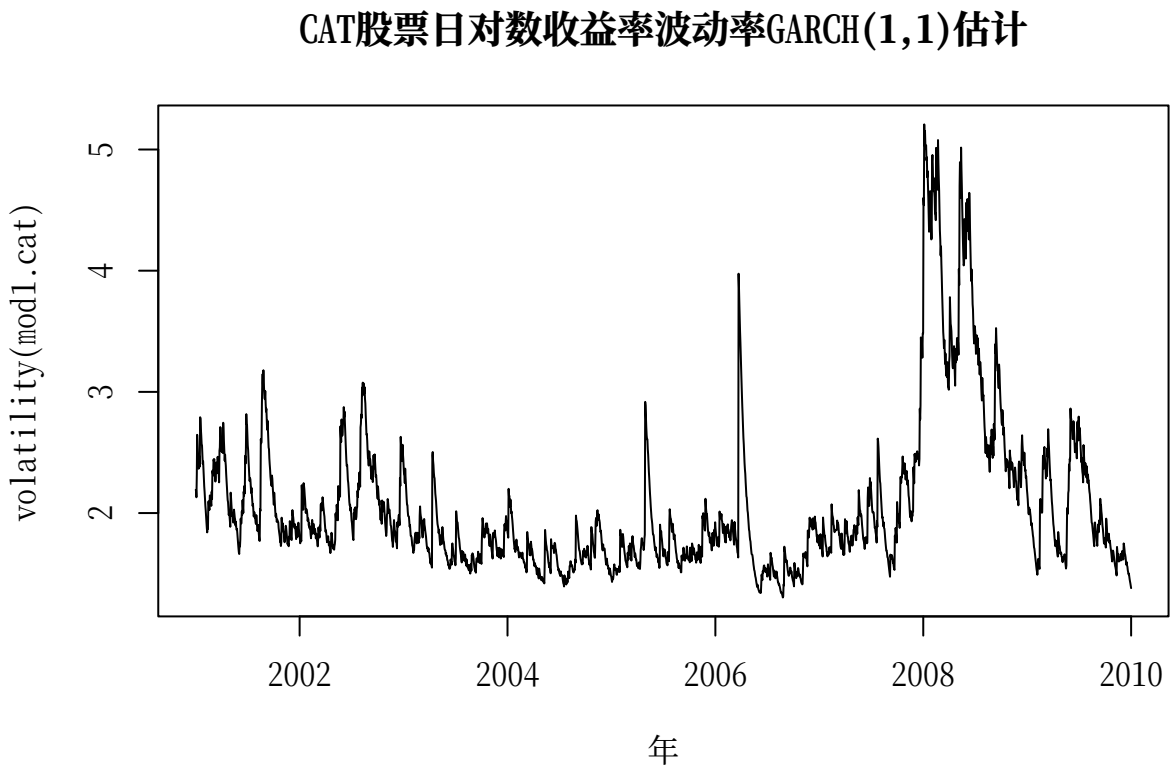


图 22.3: CAT 股票日对数收益率波动率

日波动率分布的盒形图：

```
boxplot(volatility(mod1.cat), ylab=" 日波动率", xlab="")
```

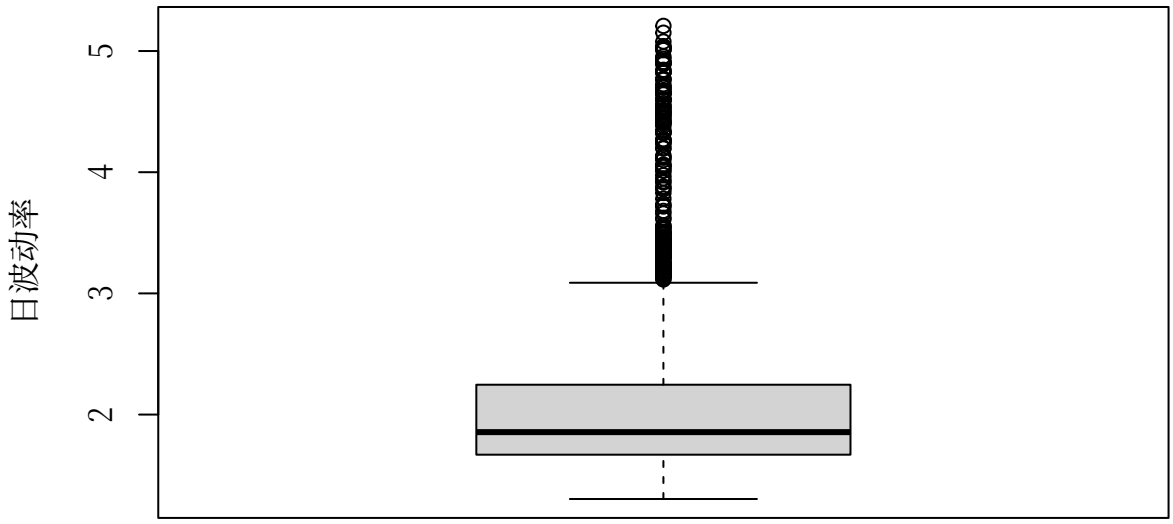


图 22.4: CAT 股票日对数收益率波动率盒形图

利用 GARCH(1,1) 模型结果计算 1 日、5 日、20 日年化波动率并作图：

```
volay.cat <- volatility_interval(
  mod1.cat, at=rtn.cat, h=c(1,5,20)
)
```

```
matplot(
  time.cat, volay.cat, type="l",
  col=c("black", "green", "blue"),
  main="CAT 股票日对数收益率 1、5、20 日年化波动率 GARCH(1,1) 估计",
  xlab=" 年")
```

```
matplot(
  time.cat[2011:2515], volay.cat[2011:2515,], type="l",
  col=c("black", "green", "blue"),
  main="CAT 股票日对数收益率 1、5、20 日年化波动率 GARCH(1,1) 估计",
  xlab=" 年")
```

在局部波动率较高时，20 日波动率低于单日波动率，可能是因为在预测 20 日波动率时发生了均值回转。

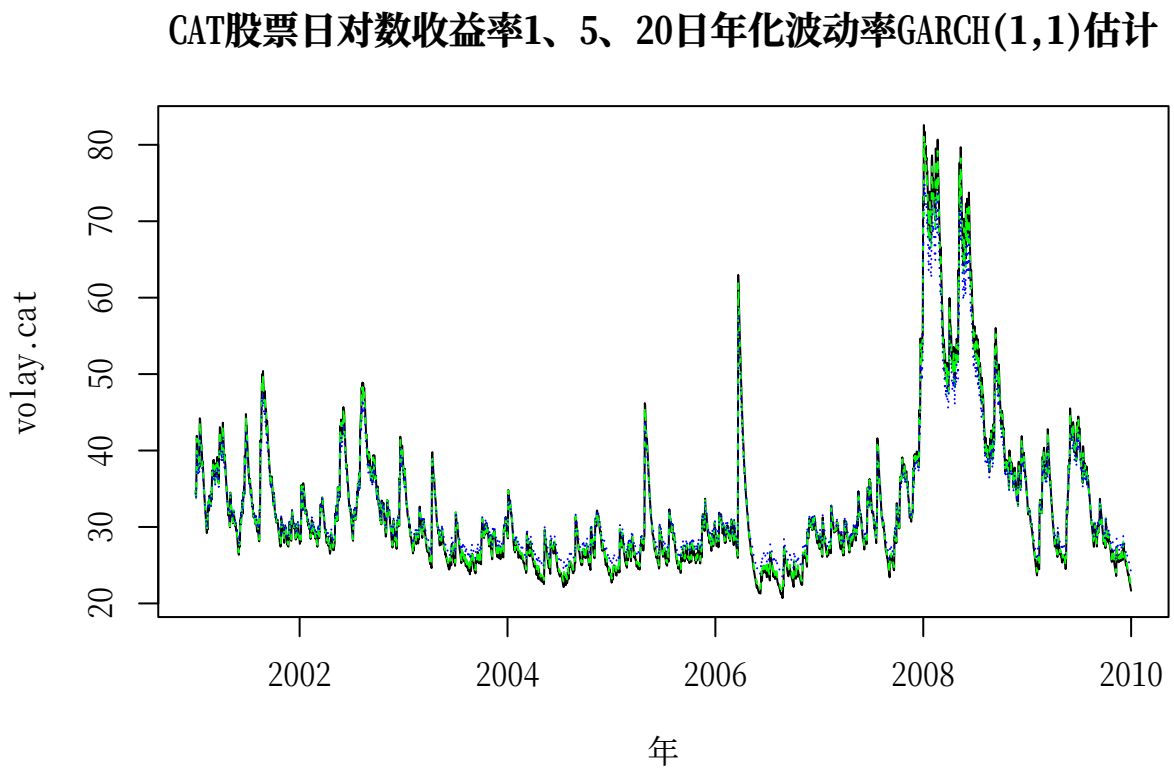


图 22.5: CAT 股票日对数收益率 1、5、20 日年化波动率

图 22.6: CAT 股票日对数收益率 1、5、20 日年化波动率 2008-2009

22.1.2 欧元对美元汇率日对数收益率的波动率期限结构

作为例子，考虑从 1999 年 1 月 4 日到 2010 年 8 月 20 日的欧元对美元汇率的日对数收益率序列。以百分之一为单位。

```
d.useu <- read_table(
  "d-useu9910.txt",
  col_types=cols(.default=col_double())
)
xts.useu <- with(d.useu, xts(rate, make_date(year, mon, day)))
xts.useu.lnrtn <- diff(log(xts.useu))[-1,]*100
eu <- c(coredata(xts.useu.lnrtn))
```

日对数收益率图形（单位百分之一）：

```
plot(xts.useu.lnrtn,
     main=" 欧元汇率日对数收益率",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years")
```

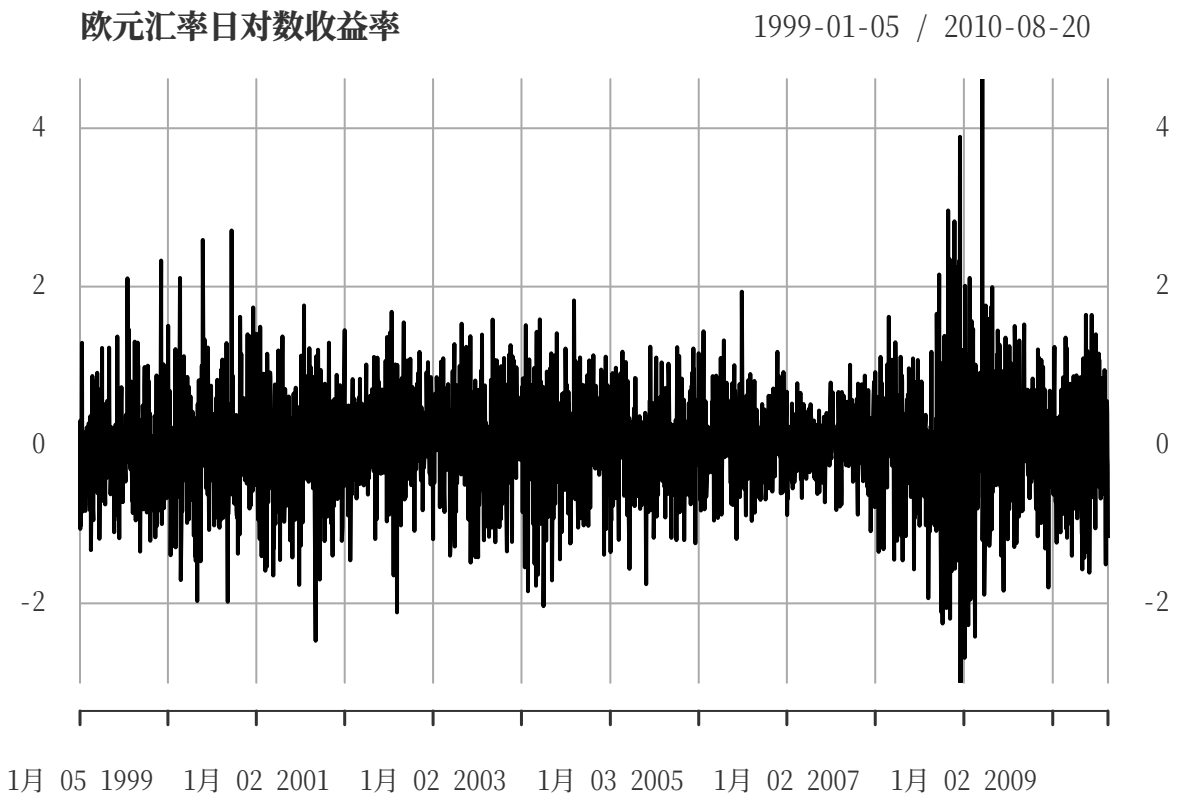
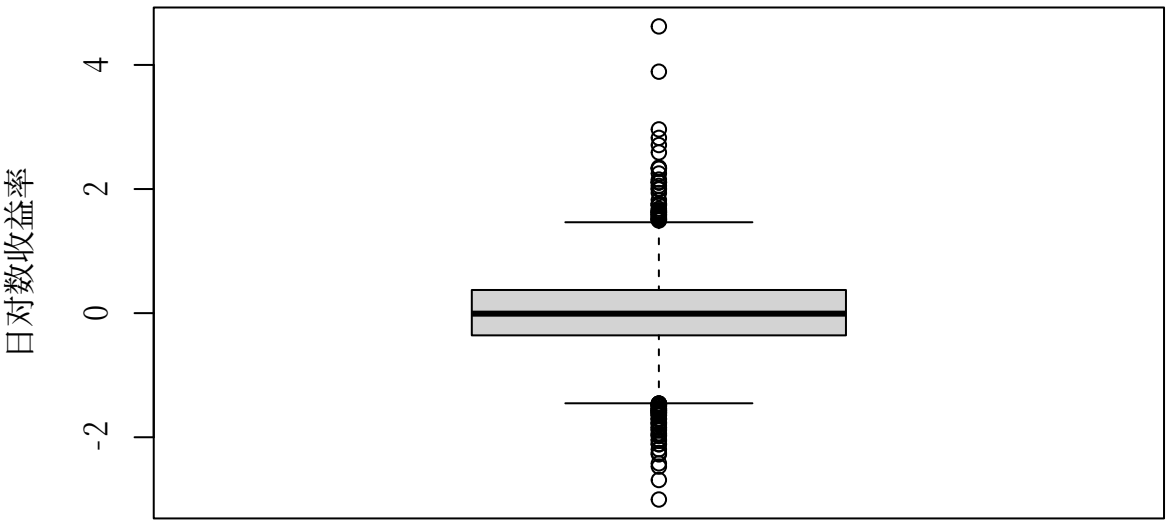


图 22.7: 欧元对美元汇率日对数收益率

日对数收益率分布盒形图：

```
boxplot(eu, ylab=" 日对数收益率", xlab="")
```



日

对数收益率比较集中在 ±2% 之间。

建立均值方程为常数、条件正态分布的 GARCH(1,1) 模型：

```
mod1 <- garchFit(~ 1 + garch(1,1), data=eu, trace=FALSE)
summary(mod1)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = eu, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224b75a5700>
##  [data = eu]
##
## Conditional Distribution:
##  norm
##
```

```
## Coefficient(s):
##        mu      omega     alpha1      beta1
## 0.0148834  0.0011236  0.0292729  0.9685221
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     0.0148834   0.0106235    1.401   0.1612
## omega  0.0011236   0.0005602    2.006   0.0449 *
## alpha1 0.0292729   0.0037996    7.704 1.31e-14 ***
## beta1  0.9685221   0.0038946  248.685  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -2733.418    normalized:  -0.9332256
##
## Description:
##  Tue May 30 17:04:18 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  54.69227  1.329603e-12
##  Shapiro-Wilk Test  R    W      0.9954108 7.170686e-08
##  Ljung-Box Test     R    Q(10)  13.33532  0.205523
##  Ljung-Box Test     R    Q(15)  20.08275  0.1687813
##  Ljung-Box Test     R    Q(20)  22.64627  0.3064951
##  Ljung-Box Test     R^2  Q(10)  13.31502  0.2065881
##  Ljung-Box Test     R^2  Q(15)  17.40896  0.29501
##  Ljung-Box Test     R^2  Q(20)  28.78138  0.09215406
##  LM Arch Test       R    TR^2   14.96898  0.2431374
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 1.869182 1.877352 1.869179 1.872125
```

模型检验基本通过。

日对数收益率波动率的图形：

```
plot(xts(volatility(mod1), index(xts.useu.lnrtn)),
     main=" 欧元汇率日对数收益率波动率 GARCH(1,1) 估计",
```

```
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years")
```

图 22.8: 欧元汇率日对数收益率波动率

日波动率分布的盒形图：

```
boxplot(volatility(mod1), ylab=" 日波动率", xlab="")
```

波动率中位数为 0.6% 左右，主要集中在 0.4% 到 1.0% 之间。

利用 GARCH(1,1) 模型结果计算 1 日、5 日、20 日年化波动率并作图：

```
volay.eu <- volatility_interval(
  mod1, at=eu, h=c(1,5,20)
)
```

```
plot(xts(volay.eu, index(xts.useu.lnrtn)),
     col=c("black", "green", "blue"),
     main=" 欧元汇率 1、5、20 日年化波动率 GARCH(1,1) 估计",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years")
```

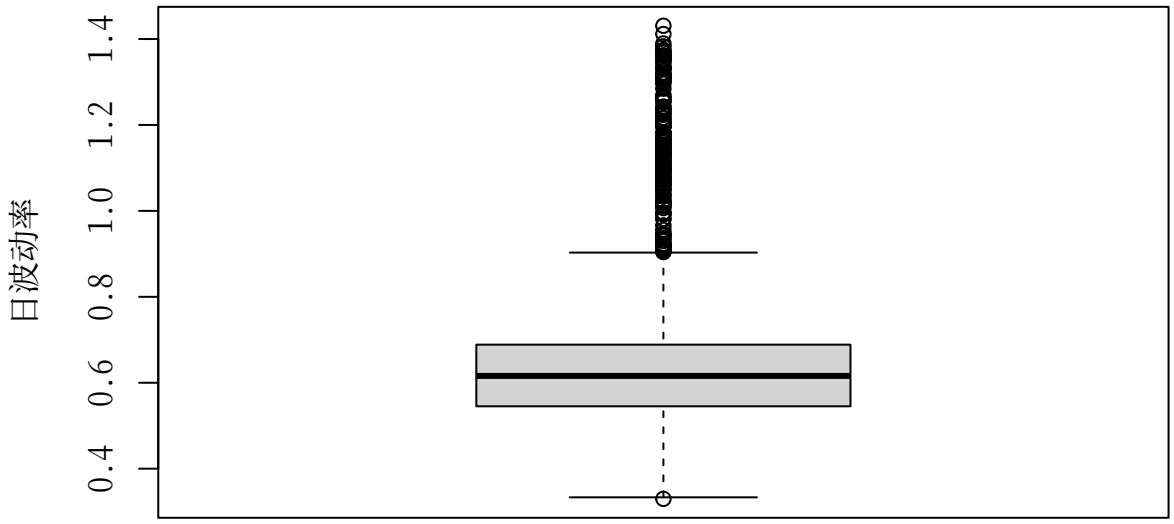


图 22.9: 欧元汇率日对数收益率波动率

图 22.10: 欧元汇率 1、5、20 日年化波动率

```
plot(xts(volay.eu, index(xts.useu.lnrtn)),
     subset="2010",
     col=c("black", "green", "blue"),
     main=" 欧元汇率 1、5、20 日年化波动率 GARCH(1,1) 估计",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years")
```

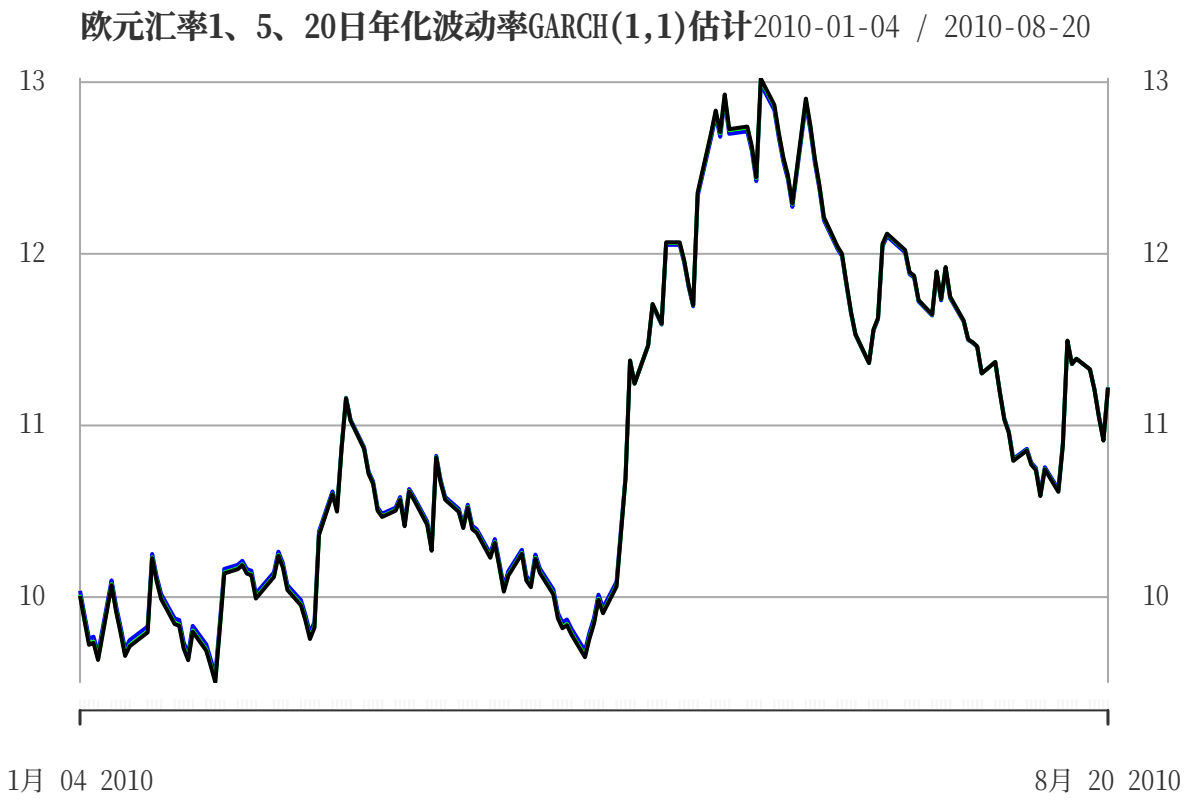


图 22.11: 欧元汇率 1、5、20 日年化波动率预测 2010 年

黑色是日波动率，绿色是 5 日波动率，蓝色是 20 日波动率。不同期的年化波动率差距不大。

22.2 期权定价和对冲

波动率的一个重要应用是对金融衍生产品定价，如著名的 Black-Scholes 期权定价公式，其中设波动率为常数。有多位学者研究了在期权定价中使用 GARCH 模型表示出变化的波动率，基于 GARCH 模型进行期权定价。

单只股票的标准期权定价中假设股票价格服从如下的几何布朗运动：

$$\frac{dP_t}{P_t} = rdt + \sigma dW_t$$

其中 r 是无风险利率，σ 是波动率，W_t 是标准维纳 (Wiener) 过程（连续时随机过程）。根据伊藤 (Ito) 引理，股票

的对数价格 $\ln P_t$ 服从如下随机微分方程

$$d\ln P_t = \left(r - \frac{1}{2}\sigma^2\right)dt + \sigma dW_t$$

实践中假定 r 和 σ 为已知，随机性来自 W_t。离散化形式为

$$P_t = P_{t-1}\exp\left\{r - \frac{1}{2}\sigma^2 + \sigma\varepsilon_t\right\} \tag{22.3}$$

其中 $\{\varepsilon_t\}$ 独立同标准正态分布。

设当前时刻为 0，当前股价为 P_0，期权的执行价格为 K，到期时间为 T，通过产生一个独立同标准正态分布的 $\{\varepsilon_1,\dots,\varepsilon_T\}$，用式(22.3)可以模拟一条价格轨道（实现），重复模拟 N 次得到 N 条价格轨道 $\{P_t^{(i)}, t=1,\dots,T,\ i=1,\dots,N\}$。可以用 $P_T^{(i)}$ 的样本平均值估计最终价格的期望 EP_T。

利用模拟的 N 条轨道，可以计算欧式看涨期权价格为

$$\begin{aligned} C(P_0) =& e^{-rT}E\left[\max(P_T - K,\ 0)\right] \\ \approx& \frac{1}{N}e^{-rT}\sum_{i=1}^{N}\max\left(P_T^{(i)} - K,\ 0\right) \end{aligned}$$

亚式看涨期权使用其轨道的平均值，公式为

$$C(P_0) \approx \frac{1}{N}e^{-rT}\sum_{i=1}^{N}\max\left(\frac{1}{T}\sum_{t=1}^{T}P_t^{(i)} - K,\ 0\right)$$

以上的模拟使用了常数波动率。对于 GARCH(1,1) 模型，在模拟价格轨道时可以同时计算变化的波动率 σ_t，这时波动率方程中的 ε_t 就是式(22.3)中的 ε_t。为了模拟一条轨道，可对 $t=1,2,\dots,T$ 递推计算

$$\begin{aligned} \sigma_t^2 =& \alpha_0 + \alpha_1\sigma_{t-1}^2\varepsilon_{t-1}^2 + \beta_1\sigma_{t-1}^2 \\ P_t =& P_{t-1}\exp\left\{r - \frac{1}{2}\sigma_t^2 + \sigma_t\varepsilon_t\right\} \end{aligned}$$

初值 σ_0^2 可取为模型的无条件方差 $\alpha_0/(1-\alpha_1-\beta_1)$。$\{\varepsilon_t, t=0,1,\dots,T\}$ 用独立同标准正态分布随机数模拟。

利用差分近似，还可以计算相应的 GARCH 期权的 δ 值和 γ 值，如

$$\begin{aligned} \delta =& \frac{1}{2\Delta}\left\{C(P_0+\Delta) - C(P_0-\Delta)\right\} \\ \gamma =& \frac{1}{\Delta^2}\left\{C(P_0+\Delta) - 2C(P_0) + C(P_0-\Delta)\right\} \end{aligned}$$

由于 $C(P_0)$ 是随机模拟得到的，所以其计算存在不稳定性。

上述定价公式是根据完美对冲得到的，在随机波动率的假设下不可能实现。但 (Duan, 1995) 证明存在如下局部的风险中心价格关系：

$$\begin{aligned} r_t =& r - \frac{1}{2}\sigma_t^2 + \lambda\sigma + \sigma_t\varepsilon_t \\ \sigma_t^2 =& \alpha_0 + \alpha_1\sigma_{t-1}^2(\varepsilon_{t-1} - \theta)^2 + \beta_1\sigma_{t-1}^2 \end{aligned}$$

在期权定价中可以用这样的 NGARCH(1,1) 模型模拟价格轨道。

22.3 随时间变化的协方差和贝塔值

22.3.1 随时间变化的协方差

GARCH 模型的另一个应用是对多元资产收益率估计随时间变化的协方差。设有两个资产收益率序列 x_t 和 y_t，因为

$$\begin{aligned}\text{Var}(x_t + y_t) =& \text{Var}(x_t) + \text{Var}(y_t) + 2\text{Cov}(x_t, y_t)\\ \text{Var}(x_t - y_t) =& \text{Var}(x_t) + \text{Var}(y_t) - 2\text{Cov}(x_t, y_t)\end{aligned}$$

所以

$$\text{Cov}(x_t, y_t) = \frac{1}{4}\left[\text{Var}(x_t + y_t) - \text{Var}(x_t - y_t)\right]$$

这样，如果 $\text{Var}(x_t + y_t)$ 和 $\text{Var}(x_t - y_t)$ 是变化的，则 $\text{Cov}(x_t, y_t)$ 也是变化的。

设 $\sigma_{x+y,t}$, $\sigma_{x-y,t}$, $\sigma_{x,t}$, $\sigma_{y,t}$ 分别为 $x_t + y_t$, $x_t - y_t$, x_t 和 y_t 的波动率，则 x_t 和 y_t 之间变化的相关系数为

$$\rho_{t,0} = \frac{\sigma^2_{x+y,t} - \sigma^2_{x-y,t}}{4\sigma_{x,t}\sigma_{y,t}}$$

22.3.2 CAT 与 CSCO 随时间变化的协方差

以 CAT 股票和 CSCO 股票日对数收益率为例，时间是 2010-01-02 到 2010-12-31。分别记 x_t 和 y_t 为 CAT 和 $CSCO$ 股票日对数收益率序列，逐个地对 $x_t + y_t$, $x_t - y_t$, x_t, y_t 建立 GARCH(1,1) 模型：

```
library(fGarch, quietly = TRUE)
mod1.cov <- garchFit(~ 1 + garch(1,1), data=rtn.cat + rtn.csco, trace=FALSE)
summary(mod1.cov)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = rtn.cat + rtn.csco,
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224b36bff10>
##  [data = rtn.cat + rtn.csco]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1     beta1
## 0.142970  0.218821  0.070019  0.915827
##
```

```
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.14297     0.06456    2.215  0.02679 *
## omega    0.21882     0.07328    2.986  0.00283 **
## alpha1   0.07002     0.01464    4.782 1.74e-06 ***
## beta1    0.91583     0.01782   51.404  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -6830.321    normalized:  -2.715833
##
## Description:
##  Tue May 30 17:04:18 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  596.8548  0
##  Shapiro-Wilk Test  R    W      0.9828596 0
##  Ljung-Box Test     R    Q(10)  10.25333  0.4185564
##  Ljung-Box Test     R    Q(15)  19.2436   0.2029067
##  Ljung-Box Test     R    Q(20)  21.87413  0.3473793
##  Ljung-Box Test     R^2  Q(10)  6.559944  0.7662327
##  Ljung-Box Test     R^2  Q(15)  9.018078  0.8765703
##  Ljung-Box Test     R^2  Q(20)  11.85996  0.9208116
##  LM Arch Test       R    TR^2   9.634916  0.6479546
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 5.434848 5.444120 5.434842 5.438213
```

```
mod2.cov <- garchFit(~ 1 + garch(1,1), data=rtn.cat - rtn.csco, trace=FALSE)
summary(mod2.cov)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = rtn.cat - rtn.csco,
```

```
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224ac640cb0>
##  [data = rtn.cat - rtn.csco]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1     beta1
## 0.083118  0.055334  0.017893  0.972639
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.083118    0.046294    1.795   0.0726 .
## omega   0.055334    0.011895    4.652 3.29e-06 ***
## alpha1  0.017893    0.002860    6.257 3.93e-10 ***
## beta1   0.972639    0.004222  230.358  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -5835.676    normalized:  -2.320348
##
## Description:
##  Tue May 30 17:04:19 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  11268.67  0
##  Shapiro-Wilk Test  R    W      0.9233503 0
##  Ljung-Box Test     R    Q(10)  8.862267  0.5452231
##  Ljung-Box Test     R    Q(15)  15.47437  0.4178174
##  Ljung-Box Test     R    Q(20)  17.3576   0.6296459
##  Ljung-Box Test     R^2  Q(10)  1.871285  0.9972348
##  Ljung-Box Test     R^2  Q(15)  4.504232  0.9955568
##  Ljung-Box Test     R^2  Q(20)  5.879926  0.9990502
```

```
##  LM Arch Test       R    TR^2   3.000405  0.9955412
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 4.643877 4.653150 4.643872 4.647242
```

```
mod3.cov <- garchFit(~ 1 + garch(1,1), data=rtn.cat, trace=FALSE)
summary(mod3.cov)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = rtn.cat, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224acef11e8>
##  [data = rtn.cat]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1     beta1
## 0.103702  0.095669  0.053109  0.924455
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.10370     0.03713    2.793  0.00522 **
## omega    0.09567     0.03385    2.826  0.00471 **
## alpha1   0.05311     0.01134    4.682 2.84e-06 ***
## beta1    0.92446     0.01787   51.739  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -5297.634    normalized:  -2.106415
##
## Description:
```

```
##  Tue May 30 17:04:20 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  2397.079  0
##  Shapiro-Wilk Test  R    W      0.9720346 0
##  Ljung-Box Test     R    Q(10)  11.99378  0.2854729
##  Ljung-Box Test     R    Q(15)  16.40952  0.355369
##  Ljung-Box Test     R    Q(20)  21.94146  0.3436959
##  Ljung-Box Test     R^2  Q(10)  1.323393  0.9993882
##  Ljung-Box Test     R^2  Q(15)  3.737861  0.9984771
##  Ljung-Box Test     R^2  Q(20)  4.956939  0.9997407
##  LM Arch Test       R    TR^2   2.67451   0.9974382
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 4.216011 4.225283 4.216006 4.219376
```

```
mod4.cov <- garchFit(~ 1 + garch(1,1), data=rtn.csco, trace=FALSE)
summary(mod4.cov)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = rtn.csco, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224b36d8300>
##  [data = rtn.csco]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1     beta1
## 0.032238  0.155836  0.082301  0.894726
##
## Std. Errors:
##  based on Hessian
##
```

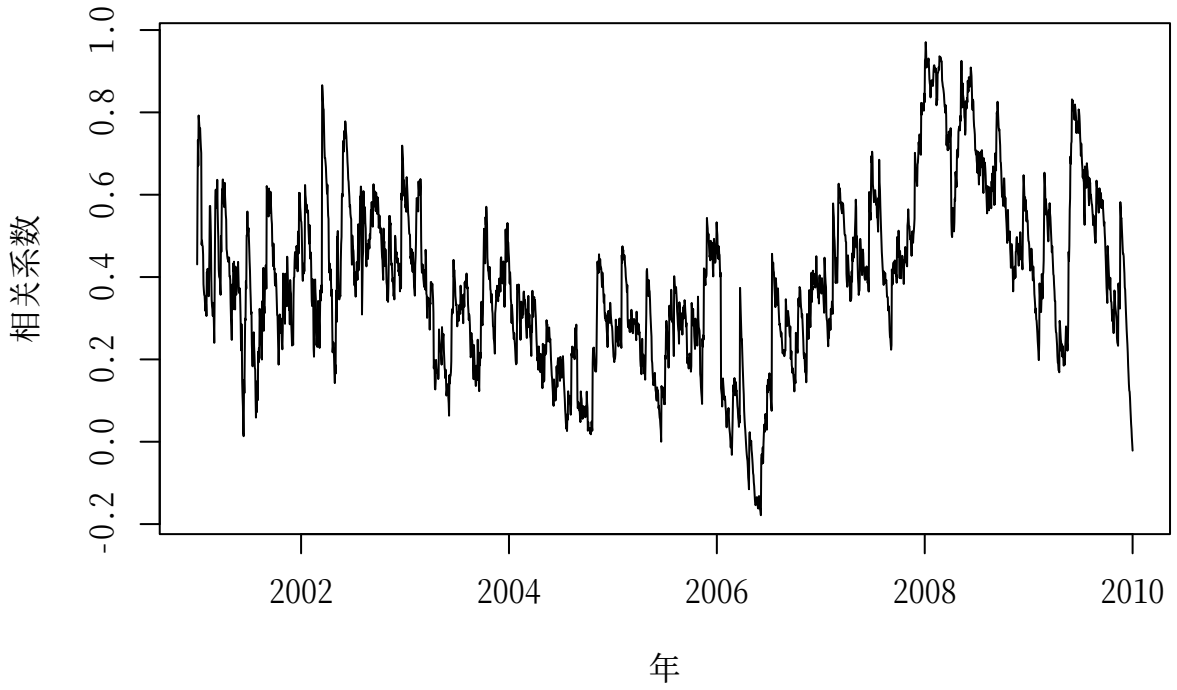
```
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu       0.03224     0.04067    0.793    0.428
## omega    0.15584     0.03941    3.954 7.69e-05 ***
## alpha1   0.08230     0.01784    4.612 3.98e-06 ***
## beta1    0.89473     0.02142   41.767  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -5714.663    normalized:  -2.272232
##
## Description:
##  Tue May 30 17:04:20 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  10943.05  0
##  Shapiro-Wilk Test  R    W      0.9417187 0
##  Ljung-Box Test     R    Q(10)  9.690217  0.468079
##  Ljung-Box Test     R    Q(15)  20.61778  0.1494922
##  Ljung-Box Test     R    Q(20)  22.97434  0.2900564
##  Ljung-Box Test     R^2  Q(10)  1.949596  0.9967121
##  Ljung-Box Test     R^2  Q(15)  2.781043  0.9997496
##  Ljung-Box Test     R^2  Q(20)  4.16767   0.9999349
##  LM Arch Test       R    TR^2   2.935696  0.9959812
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 4.547644 4.556917 4.547639 4.551010
```

变化的协方差 $\mathrm{Cov}(x_t, y_t)$ 和变化的相关系数 $\rho_{t,0}$ 的估计：

```
vcov.cat <- 1/4*(volatility(mod1.cov)^2 - volatility(mod2.cov)^2)
vrho.cat <- vcov.cat / volatility(mod3.cov) / volatility(mod4.cov)
```

变化的相关系数的时间序列图：

```
plot(time.cat, vrho.cat, type="l", xlab=" 年", ylab=" 相关系数")
```



22.3.3 随时间变化的贝塔值

贝塔值的定义来自资产定价模型 (CAPM):

$$r_t = \alpha + \beta r_{m,t} + e_t,\ t = 1, 2, ..., T \tag{22.4}$$

其中 r_t 是所研究的资产的收益率，$r_{m,t}$ 是市场的收益率（如综合股指的收益率）。这里没有使用超额收益率。

这个一元线性回归模型提供了估计下列因子的方法：

- 股票对市场风险的敏感性的因子 β;
- 股票相对于市场的错误定价 α;
- 特定股票回报 e_t。

特定股票回报一般不重要，在资产组合中这一部分经过线性组合后会接近于 0。

β 提供了股票相对于市场如何变化的测量，如果 β 不显著地区别于零，则股票变化与市场变化无关；如果 β 显著地大于 1，则股票对市场变化的反应特别强烈，这样的股票被认为是高风险的。$\beta < 1$ 的股票是低风险的。

α 是 $r_{m,t} = 0$ 时研究的股票的平均收益率，在实践中，如果可能，尽可能选取 α 为正数、β 较小的资产。

对(22.4)的 CAPM 模型，β 的理论值为

$$\beta = \frac{\mathrm{Cov}(r_t, r_{m,t})}{\mathrm{Var}(r_{m,t})}$$

β 的定义依赖于市场指数 $r_{m,t}$ 的选择。

一般相信 β 值是随时间而变的，可以用前面估计变化的协方差的方法估计变化的贝塔值。$\text{Cov}(r_t, r_{m,t})$ 可以通过 $\text{Var}(r_t + r_{m,t})$ 和 $\text{Var}(r_t - r_{m,t})$ 估计。

22.3.4 CAT 股票日对数收益率随时间变化的贝塔值

作为示例，考虑 CAT 股票的日对数收益率 r_t，采用标普 500 指数的日对数收益率作为市场收益率 $r_{m,t}$，时间从 2001-01-02 到 2010-12-31。

用 CAPM 模型计算不随时间变化的贝塔值：

```
lm1.cat <- lm(rtn.cat ~ rtn.sp500)
summary(lm1.cat)
```

```
##
## Call:
## lm(formula = rtn.cat ~ rtn.sp500)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.8901  -0.7715  -0.0025   0.8066   9.9412
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06688    0.03041   2.199   0.0279 *
## rtn.sp500    1.14580    0.02209  51.870   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.525 on 2513 degrees of freedom
## Multiple R-squared:  0.5171, Adjusted R-squared:  0.5169
## F-statistic:  2690 on 1 and 2513 DF,  p-value: < 2.2e-16
```

估计的 CAPM 模型为

$$r_t = 0.0669 + 1.146 r_{t,m} + e_t$$

e_t 的标准差估计为 1.525，回归复相关系数平方为 0.52。贝塔值估计为 $\hat{\beta} = 1.146$。贝塔值的近似 95% 置信区间可以用估计值加减两倍标准误差计算，为 $(1.10, 1.19)$ 位于 1 的右侧，所以可以认为贝塔值显著地超过 1。

估计变化的贝塔值：

```
mod1.beta <- garchFit(~ 1 + garch(1,1), data=rtn.cat + rtn.sp500, trace=FALSE)
summary(mod1.beta)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
```

```
##  garchFit(formula = ~1 + garch(1, 1), data = rtn.cat + rtn.sp500,
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224b408e3a0>
##  [data = rtn.cat + rtn.sp500]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1     beta1
## 0.139285  0.140215  0.065339  0.919071
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.139285    0.050645    2.750 0.005956 **
## omega   0.140215    0.039482    3.551 0.000383 ***
## alpha1  0.065339    0.009868    6.621 3.56e-11 ***
## beta1   0.919071    0.012540   73.292  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -6187.054    normalized:  -2.460061
##
## Description:
##  Tue May 30 17:04:20 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  539.8153  0
##  Shapiro-Wilk Test  R    W      0.9854249 2.198538e-15
##  Ljung-Box Test     R    Q(10)  8.709001  0.5599252
##  Ljung-Box Test     R    Q(15)  15.89029  0.3893801
##  Ljung-Box Test     R    Q(20)  22.16946  0.3313865
##  Ljung-Box Test     R^2  Q(10)  8.114921  0.6176126
##  Ljung-Box Test     R^2  Q(15)  10.04766  0.8167316
```

```
##  Ljung-Box Test     R^2  Q(20)  10.86368  0.9496814
##  LM Arch Test       R    TR^2   9.232218  0.6829784
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 4.923303 4.932576 4.923298 4.926669
```

```
mod2.beta <- garchFit(~ 1 + garch(1,1), data=rtn.cat - rtn.sp500, trace=FALSE)
summary(mod2.beta)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = rtn.cat - rtn.sp500,
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224b8396410>
##  [data = rtn.cat - rtn.sp500]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1     beta1
## 0.085274  0.035620  0.024234  0.960046
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.085274    0.028513    2.991 0.002784 **
## omega   0.035620    0.010593    3.363 0.000772 ***
## alpha1  0.024234    0.005045    4.804 1.56e-06 ***
## beta1   0.960046    0.008561  112.147  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -4554.109    normalized:  -1.810779
```

```
##
## Description:
##  Tue May 30 17:04:21 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  11837.86  0
##  Shapiro-Wilk Test  R    W      0.9329647 0
##  Ljung-Box Test     R    Q(10)  21.67769  0.01683345
##  Ljung-Box Test     R    Q(15)  22.98638  0.08442948
##  Ljung-Box Test     R    Q(20)  27.82026  0.1137229
##  Ljung-Box Test     R^2  Q(10)  1.764148  0.997849
##  Ljung-Box Test     R^2  Q(15)  7.502387  0.9421823
##  Ljung-Box Test     R^2  Q(20)  8.406546  0.9888179
##  LM Arch Test       R    TR^2   6.229631  0.9040666
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 3.624739 3.634011 3.624734 3.628104
```

```
mod3.beta <- garchFit(~ 1 + garch(1,1), data=rtn.sp500, trace=FALSE)
summary(mod3.beta)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~1 + garch(1, 1), data = rtn.sp500, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ 1 + garch(1, 1)
## <environment: 0x00000224b89ee030>
##  [data = rtn.sp500]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       mu     omega    alpha1     beta1
## 0.019327  0.012328  0.078912  0.912590
##
## Std. Errors:
```

```
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.019327    0.017617    1.097    0.273
## omega   0.012328    0.003061    4.028 5.63e-05 ***
## alpha1  0.078912    0.009468    8.335  < 2e-16 ***
## beta1   0.912590    0.009848   92.669  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -3723.516    normalized:  -1.480523
##
## Description:
##  Tue May 30 17:04:21 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  199.5819  0
##  Shapiro-Wilk Test  R    W      0.989017  5.502816e-13
##  Ljung-Box Test     R    Q(10)  15.57277  0.1125331
##  Ljung-Box Test     R    Q(15)  24.90961  0.05117063
##  Ljung-Box Test     R    Q(20)  28.50665  0.09793796
##  Ljung-Box Test     R^2  Q(10)  22.30558  0.01362135
##  Ljung-Box Test     R^2  Q(15)  24.01634  0.06481646
##  Ljung-Box Test     R^2  Q(20)  24.89998  0.2052845
##  LM Arch Test       R    TR^2   26.01127  0.01069455
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 2.964227 2.973500 2.964222 2.967593
```

```
betat.cat <- 1/4*(volatility(mod1.beta)^2 - volatility(mod2.beta)^2) / volatility(mod3.beta)^2
```

作变化的 β 的时间序列图：

```
plot(time.cat, betat.cat,
  type="l", xlab=" 年", ylab=" 贝塔值")
```

这些贝塔值有一个明显的异常值，检查数据发现 CAT 股票在 2006-10-20 有一个 14.52% 跌幅。去掉这个大的值以后作图：

```
plot(time.cat, betat.cat,
  type="l", ylim=c(0.4, 3),
```

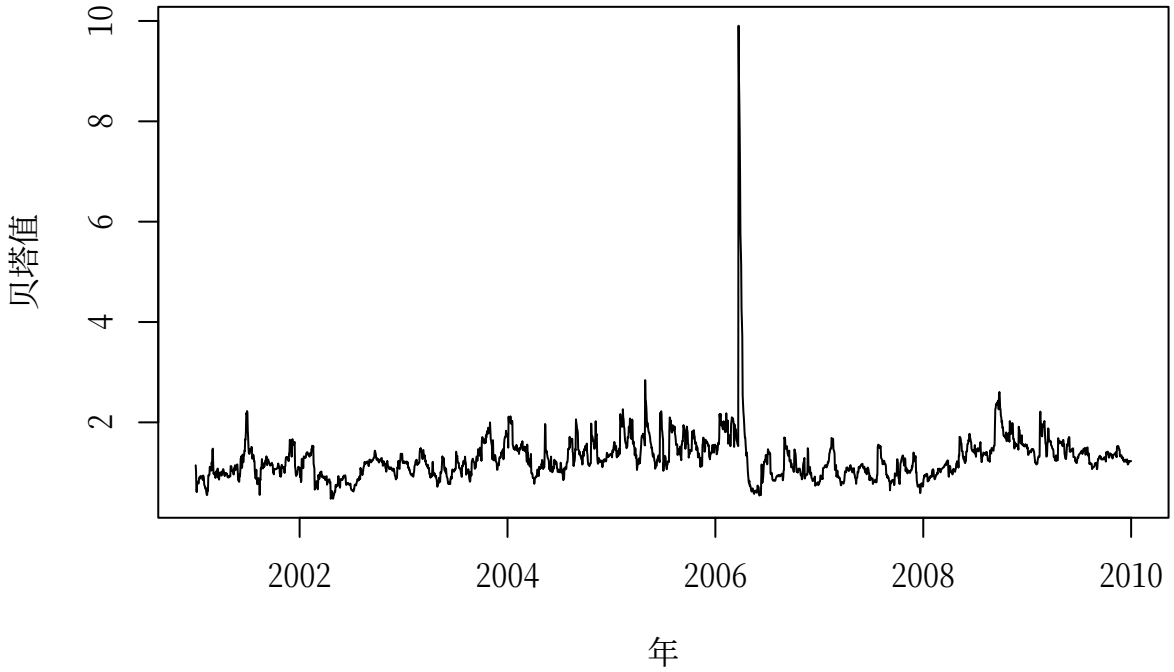


图 22.12: CAT 股票日对数收益率变化的贝塔值

```
  xlab=" 年", ylab=" 贝塔值")
abline(h=mean(betat.cat[betat.cat<4.0]), lty=3)
```

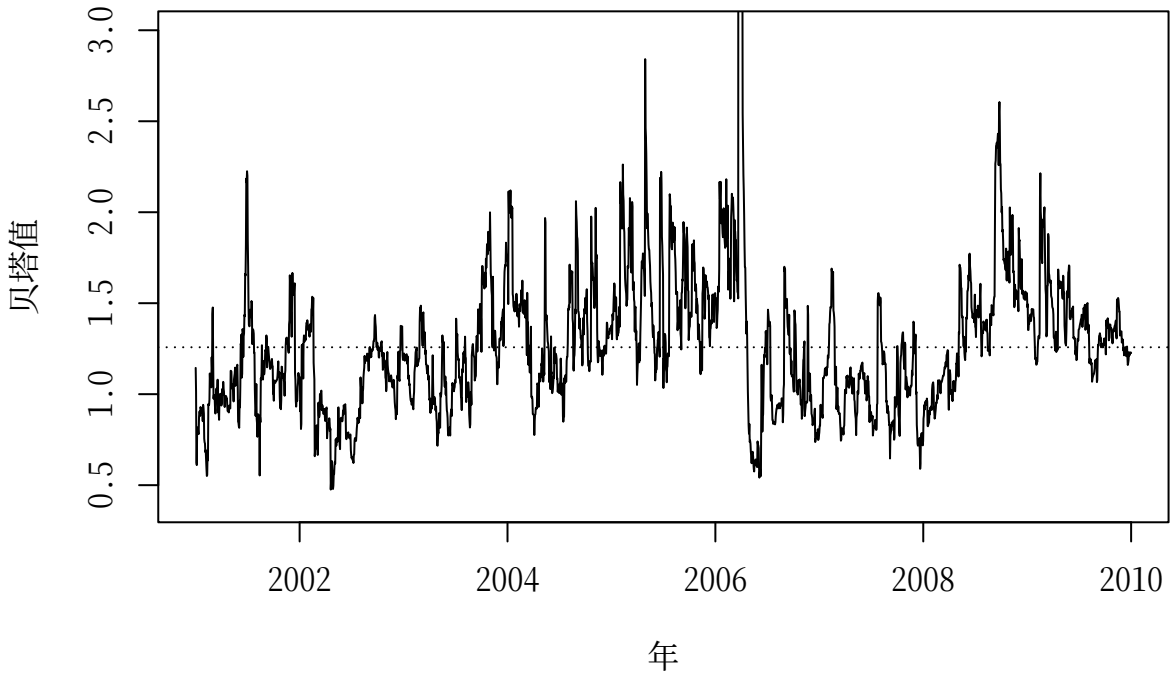


图 22.13: CAT 股票日对数收益率变化的贝塔值

这个序列看起来是平稳的，但不是不相关列。如何检验这样一个序列是否均值为常数？建议：可以将其前后分成若干段，如 4 段，然后引入表示分段的哑变量，建立带有哑变量自变量的 ARMA 模型，检验哑变量是否显著。

22.4 最小方差投资组合

GARCH 模型的另外一个应用是在计算最小方差投资组合的过程当中，计算随时间变化的协方差。

考虑 (Markovitz, 1959) 提出的投资组合中的均值——方差分析，这里集中考虑最小方差组合。假设组合中有 k 个风险资产，用组合的波动率（条件标准差）作为风险度量。设这 k 个资产在 t 时刻的收益率为 $r_t = (r_{1t}, \dots, r_{kt})^T$，设 $\text{Var}(r_t) = V_t$。设 t 时刻组合的权重为 $w_t = (w_{1t}, \dots, w_{kt})^T$，权重是投资组合中每项资产所占的百分比。组合收益率为 $w_t^T r_t$，组合收益率的方差为 $w_t^T V_t w_t$。

最小方差投资组合是选 w_t 为如下约束优化问题的解：

$$\min_{w_t} w_t^T V_t w_t, \text{ s.t.}$$
$$1^T w_t = 1$$

其中 1 表示元素全是 1 的列向量，$1^T w_t = \sum_{i=1}^{k} w_{it}$。

如果允许权重取负数，即可以卖空资产，上述优化问题的解是

$$w_t = \frac{V_t^{-1}1}{1^T V_t^{-1}1}$$

其中分子是协方差阵 V_t 的逆矩阵 V_t^{-1} 的行和组成的向量，分母是标量，使得结果的元素和等于 1。

事实上，定义拉格朗日乘子函数

$$f(w,\lambda) = w^T V w - \lambda(1^T w - 1)$$

令

$$\begin{aligned}\frac{\partial f}{\partial w} =& 2Vw - \lambda 1 = 0\\ \frac{\partial f}{\partial \lambda} =& -(1^T w - 1) = 0\end{aligned}$$

可得

$$w = \frac{\lambda}{2} V^{-1}1$$

代入 $1^T w - 1$ 即可得

$$w = \frac{V^{-1}1}{1^T V^{-1}1}$$

优化问题的最小值，即最小方差投资组合的方差，为

$$w^T V w = \frac{1}{1^T V^{-1}1}$$

22.4.1 分段更新权重的最小方差投资组合

在实践中，权重依赖于用来估计 V_t 的样本。即使使用不随时间变化的权重，也可以将时间分段，每段时间采用一个固定权重。

下面将作者放在多个文件中 2001 年到 2010 的 99 只股票的日简单收益率数据，统一地转换为一个多元 xts 时间序列对象：

```
library(tidyverse, quietly = TRUE)
fpath <- "../refs/IAFD/ch5/ch5data"
da <- read_table2(
  paste(fpath, "d-abt-0110.txt", sep="/"),
  col_types=cols(.default=col_double(),
                 date=col_date(format="%Y%m%d")))
date.0110 <- da$date
files.0110 <- c(
  "d-a2a-0110.txt", "d-b2b-0110.txt", "d-c2c-0110.txt",
  "d-d2e-0110.txt", "d-f2g-0110.txt", "d-h2j-0110.txt",
  "d-k2m-0110.txt", "d-n2p-0110.txt", "d-q2t-0110.txt", "d-u2x-0110.txt"
)
da.d0110 <- tibble(date=date.0110)
for(fn in files.0110){
  cat("===== Reading", fn, "\n")
  da <- read_table2(
```

```
    paste(fpath, fn, sep="/"),
    col_types=cols(.default=col_double()))
  da.d0110 <- cbind(da.d0110, da)
}
xts.d0110 <- xts(da.d0110[,-1], da.d0110$date)
head(xts.d0110)
save(xts.d0110, file="xts-d0110.RData")
```

其中原始文件 d-f2g-0110.txt 有一些格式错误，进行了编辑修改。读入的数据保存成了 R 的 RData 格式。

载入读取的 2001-2010 各个股票的日简单收益率数据：

```
load("xts-d0110.RData")
```

作为示例，仅考虑 5 只股票：波音 (BA)、卡特彼勒 (CAT)、IBM、微软 (MSFT)、宝洁 (PG)。将 2001-2010 的 2515 个交易日，分成三段：1:756(2001-01-02 到 2004-01-07)，757:1512(2004-01-08 到 2007-01-09), 1513:2515(2007-01-10 到 2010-12-31)。

对每一段计算日对数收益率最小方差投资组合的权重，并计算其波动率的函数：

```
## x 为多元 xts 类型的日对数收益率序列,
## 结果返回投资组合权重,
## 各成分股的波动率和组合波动率
portfolio.const <- function(x){
  xm <- coredata(x)
  k <- ncol(xm)
  vmat <- var(xm)
  w <- solve(vmat, rep(1.0, k))
  var.port <- 1 / sum(w)
  w <- w * var.port

  vola <- c(apply(xm, 2, sd, na.rm=TRUE), Portfolio=sqrt(var.port))

  list(weight=w, volatility=vola)
}
```

计算每一段的结果。将简单收益率转换成了对数收益率。

```
stocks5 <- c("BA", "CAT", "IBM", "MSFT", "PG")
per <- rbind(c(1,756), c(757,1512), c(1513,2515))
tab.5w <- tibble(
  " 资产"=c(" 波音", " 卡特彼勒", "IBM", " 微软", " 宝洁"),
  " 交易日 1--756"=0.0, " 交易日 757--1512"=0.0, " 交易日 1513--2515"=0.0
)
tab.5v <- tibble(
  " 资产"=c(" 波音", " 卡特彼勒", "IBM", " 微软", " 宝洁", " 最小方差投资组合"),
  " 交易日 1--756"=0.0, " 交易日 757--1512"=0.0, " 交易日 1513--2515"=0.0
```

```
)
for(j in 1:3){
  res.5s <- portfolio.const(log(1 + xts.d0110[per[j,1]:per[j,2], stocks5]))
  tab.5w[,j+1] <- res.5s$weight*100
  tab.5v[,j+1] <- res.5s$volatility*100
}
```

三个阶段各个股票的权重 (单位为百分之一)：

```
knitr::kable(tab.5w, digits=2)
```

资产	交易日 1–756	交易日 757–1512	交易日 1513–2515
波音	6.73	9.09	2.05
卡特彼勒	14.43	3.77	-8.05
IBM	11.14	28.99	34.34
微软	4.45	15.79	0.42
宝洁	63.24	42.35	71.24

三个阶段各个股票的收益率的波动率和最小方差投资组合的收益率的波动率：

```
knitr::kable(tab.5v, digits=2)
```

资产	交易日 1–756	交易日 757–1512	交易日 1513–2515
波音	2.37	1.35	2.27
卡特彼勒	2.08	1.64	2.61
IBM	2.22	0.99	1.67
微软	2.45	1.11	2.17
宝洁	1.41	0.89	1.38
最小方差投资组合	1.22	0.70	1.30

可以看出投资组合收益率的波动率低于各个成份股的波动率。注意教材 P.204 表 5-2 中 ABT 应改为 “波音”。

22.4.2 逐点更新权重的最小方差投资组合

上述每一段时间采用固定权重的最小方差投资组合是无法实际应用的，因为每一段在计算权重时，要使用本段所有样本来估计协方差阵，但是投资操作需要预先确定权重。

可以选择比较短的时间段，用上一时间段估计的协方差阵估计权重，将估计的权重应用在下一阶段的投资中。

更进一步，可以进行超前一步的协方差阵预测，并解出最小方差投资组合权重，用于下一步的投资。进行超前一步协方差阵预测时，用现有的观测数据建立 GARCH 模型估计条件方差和条件协方差，作为下一步的协方差阵的估计。

这样的方法的缺点是，得到的协方差阵可能不正定。

作者的函数 `GMVP()` 输入一个对数收益率多元时间序列，指定一个起点 h，基于时间 $1:h$ 的数据建立多个 GARCH 模型估计第 h 步的协方差阵 $\hat{V}_h$，用 $\hat{V}_h$ 当作 $h+1$ 时刻的协方差阵解出 $h+1$ 时刻的最小方差投资组合权重，计算该组合的对数收益率；然后基于时间 $1:(h+1)$ 的数据重新建模求解 $h+2$ 时刻的权重，如此递推直到数据末尾。程序如下：

```
"GMVP" <- function(rtn, start=500){
  ## compute the weights and variance of global minimum variance portfolio.
  ## The time period is from (start) to end of the data.
  ## uses cov(x,y) = [var(x+y)-var(x-y)]/4.

  library(fGarch, quietly = TRUE)
  rtn=as.matrix(rtn)

  T=dim(rtn)[1]
  k=dim(rtn)[2]
  wgt = NULL
  mVar=NULL
  VAR = NULL
  ONE=matrix(1,k,1)
  prtn=NULL
  Det=NULL
  for (t in start:T){
    # estimate variances and covariances at time "t".
    COV=matrix(0,k,k)
    for (i in 1:k){
      m1=garchFit(~1+garch(1,1), data=rtn[1:t,i], trace=FALSE)
      COV[i,i]=volatility(m1)[t]^2
      if(i < k){
        for (j in (i+1):k){
          x=rtn[1:t,i]+rtn[1:t,j]
          y=rtn[1:t,i]-rtn[1:t,j]
          m2=garchFit(~1+garch(1,1),data=x,trace=F)
          m3=garchFit(~1+garch(1,1),data=y,trace=F)
          v2=volatility(m2)[t]
          v3=volatility(m3)[t]
          COV[j,i]=(v2^2-v3^2)/4
          COV[i,j]=COV[j,i]
          # end of j-loop
        }
        # end of (if-statement)
      }
      # end of i-loop
    }
    Det=c(Det,det(COV))
    VAR=rbind(VAR,diag(COV))
    Psi=solve(COV, ONE)
    W=sum(Psi)
    Psi=Psi/W
```

```
    wgt=cbind(wgt,Psi)
    mVar=c(mVar,1/W)
    if(t < T){
      prtn=c(prtn,sum(rtn[t+1,]*Psi))
    }

  }

  list(weights=wgt, minVariance=mVar, variances=VAR, returns=prtn, det=Det)
}
```

函数结果为一个列表，weights 是基于 $1:h$ 时刻直到基于 $1:T$ 时刻估计的投资权重，但 t 时刻解出的权重用于 $t+1$ 时刻的投资；minVariance 是 $(h+1):(T+1)$ 时刻对应的最小方差投资组合的最小方差，variances 是 $h:T$ 时刻的各个个股的条件方差估计，returns 是 $(h+1):T$ 时刻的最小方差投资组合日对数收益率。

这个函数需要对 $t=h,h+1,\ldots,T$ 每一步建立多个 GARCH 模型，程序运行速度较慢，可以考虑采用并行计算加快速度。

用上述办法对 ABT、IBM 和 WMT(沃尔玛) 进行逐步最小方差投资组合计算，数据从 2001-01-02 到 2010-12-31 共 2515 个交易日，从 $h=2011$ 位置开始，该时间为 2008-12-31, 资产组合投资从 2009-01-02 到 2010-12-31 日共 504 个投资。

```
res.3p <- GMVP(log(1 + coredata(
  xts.d0110[,c("ABT", "IBM", "WMT")])), start=2011)
save(res.3p, file="garchapp-port-abt-ibm-wmt.RData")
```

GMVP() 的结果 res.3p 为一个列表，除了 returns 之外都有对应于 2011:2515 的 505 个结果，而 returns 则保存了对应于时刻 2012:2515 的 504 个最小方差投资组合的日对数收益率。

投资组合权重的取值范围：

```
range(res.3p$`weights`)
```

```
## [1] -0.5880472  1.0954013
```

最后的 504 天的投资组合的平均日对数收益率：

```
mean(res.3p$returns)
```

```
## [1] 0.0001088877
```

最后 504 天每个个股的平均日对数收益率：

```
T1 <- 2012; T2 <- 2515
colMeans(log(1 + coredata(
  xts.d0110[T1:T2,c("ABT", "IBM", "WMT")])))
```

```
##           ABT           IBM           WMT
## -8.356484e-05  1.180094e-03  1.122753e-05
```

从投资组合的平均收益来看表现并不够好。

最后 504 天投资组合的日对数收益率的样本标准差：

```
sd(res.3p$returns)
```

```
## [1] 0.01026185
```

最后 504 天三个个股的日对数收益率的样本标准差：

```
T1 <- 2012; T2 <- 2515
apply(log(1 + coredata(
  xts.d0110[T1:T2,c("ABT", "IBM", "WMT")])), 2, sd)
```

```
##         ABT         IBM         WMT
## 0.01315420 0.01460831 0.01171992
```

投资组合的对数收益率的样本标准差低于每个个股的样本标准差。

个股的条件方差的分布：

```
summary(res.3p$variances)
```

```
##        V1                   V2                   V3
##  Min.   :7.438e-05   Min.   :6.389e-05   Min.   :5.303e-05
##  1st Qu.:1.007e-04   1st Qu.:1.068e-04   1st Qu.:8.496e-05
##  Median :1.377e-04   Median :1.579e-04   Median :1.055e-04
##  Mean   :1.994e-04   Mean   :2.382e-04   Mean   :1.674e-04
##  3rd Qu.:2.360e-04   3rd Qu.:2.767e-04   3rd Qu.:1.609e-04
##  Max.   :7.029e-04   Max.   :1.369e-03   Max.   :6.714e-04
```

最小方差投资组合的方差的分布：

```
summary(res.3p$minVariance)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## 3.194e-05 4.859e-05 6.913e-05 9.866e-05 1.072e-04 3.486e-04
```

可以看出最小方差投资组合的条件方差要低。用并列盒形图比较：

```
var.3p <- as.data.frame(res.3p$variances)
names(var.3p) <- c("ABT", "IBM", "WMT")
var.3p[["MinVarPortf"]] <- res.3p$minVariance
boxplot(var.3p, main=" 三个个股与最小方差投资组合的日对数收益率条件方差分布")
```

最后的 20 个交易日的个股波动率与最小方差投资组合波动率的列表 (单位：%)：

```
T2 <- length(index(xts.d0110))
T3 <- length(res.3p$minVariance)
vola.3p20 <- as.data.frame(matrix(0.0, 20, 4))
vola.3p20[,1:3] <- sqrt(res.3p$variances[(T3-19):T3,])*100
vola.3p20[,4] <- sqrt(res.3p$minVariance[(T3-19):T3])*100
names(vola.3p20) <- c("ABT", "IBM", "WMT", "MinVarPort")
```

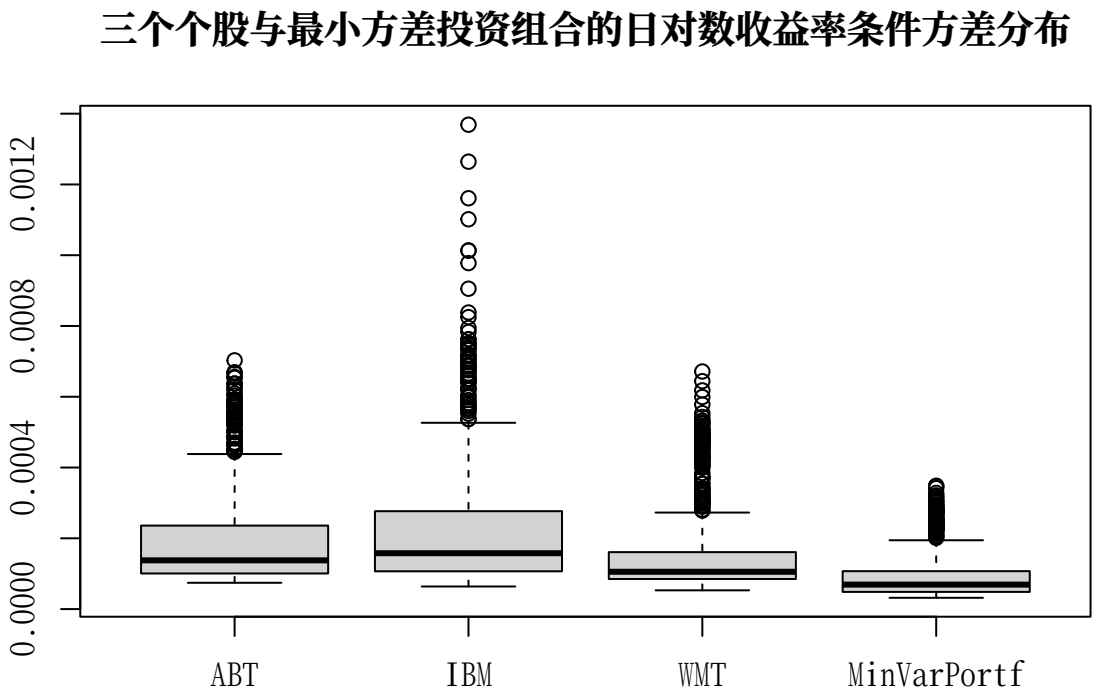


图 22.14: 三个个股与最小方差投资组合的日对数收益率条件方差分布

```
vola.3p20 <- cbind("Date"=index(xts.d0110)[(T2-19):T2], vola.3p20)
knitr::kable(vola.3p20, digits=2)
```

Date	ABT	IBM	WMT	MinVarPort
2010-12-03	0.98	1.25	0.81	0.71
2010-12-06	0.97	1.19	0.80	0.70
2010-12-07	0.98	1.15	0.79	0.70
2010-12-08	0.96	1.12	0.81	0.69
2010-12-09	0.95	1.10	0.80	0.68
2010-12-10	0.95	1.07	0.79	0.66
2010-12-13	0.96	1.03	0.78	0.65
2010-12-14	0.95	1.00	0.77	0.64
2010-12-15	0.96	1.02	0.76	0.65
2010-12-16	0.95	1.01	0.76	0.64
2010-12-17	0.95	0.98	0.76	0.64
2010-12-20	0.94	0.95	0.76	0.63
2010-12-21	0.92	0.92	0.78	0.64
2010-12-22	0.92	0.93	0.77	0.63
2010-12-23	0.91	0.90	0.77	0.62
2010-12-27	0.89	0.87	0.77	0.61
2010-12-28	0.90	0.85	0.76	0.60
2010-12-29	0.89	0.83	0.75	0.59
2010-12-30	0.87	0.82	0.75	0.59
2010-12-31	0.86	0.80	0.74	0.58

22.5 预测

联合使用均值方程和波动率方程建模可以更好地描述数据特征，也可以得到更合理的预测区间。

以石油价格的建模和预测为例。2008 年夏和 2011 年春的石油价格上涨对世界经济有很大的影响。预测原油价格有重大意义，但是石油价格收很多因素和外部扰动的影响，不容易分析。

这里利用 1997-01-03 到 2010-09-24 的美国原油价格的周数据，共 717 个观测值。数据是以估计的进口数量（Freight on board，FOB）为权重的加权离岸价格的周数据，用美元/桶表示。数据可以从 US Energy Information Administration 网站下载。

载入数据：

```
# 输入文件用了空格和制表符分隔，read_table 目前不支持。
d.oil <- read.table(
  "w-petroprice.txt", header=TRUE) |>
  as_tibble()
xts.oil <- xts(
  d.oil[,4:5], make_date(
    d.oil[["Year"]], d.oil[["Mon"]], d.oil[["Day"]])
```

```
)
poil <- d.oil[["US"]]
doil <- diff(poil)
```

价格的时间序列图：

```
plot(xts.oil[,"US"], type="l",
     main=" 美国石油价格周数据",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years",
     col="red")
```

图 22.15: 美国石油价格周数据

可以看出价格序列非平稳，存在明显的趋势，在 2008 年夏季有明显的峰值。

价格差分数据：

```
plot(diff(xts.oil[,"US"]), type="l",
     main=" 美国石油价格周数据的变化量",
     major.ticks="years", minor.ticks=NULL,
     grid.ticks.on="years",
     col="red")
```

可以看出差分的均值基本是平稳的，但存在波动率聚集，用一般的 ARMA 模型无法解释这样的波动率聚集现象。

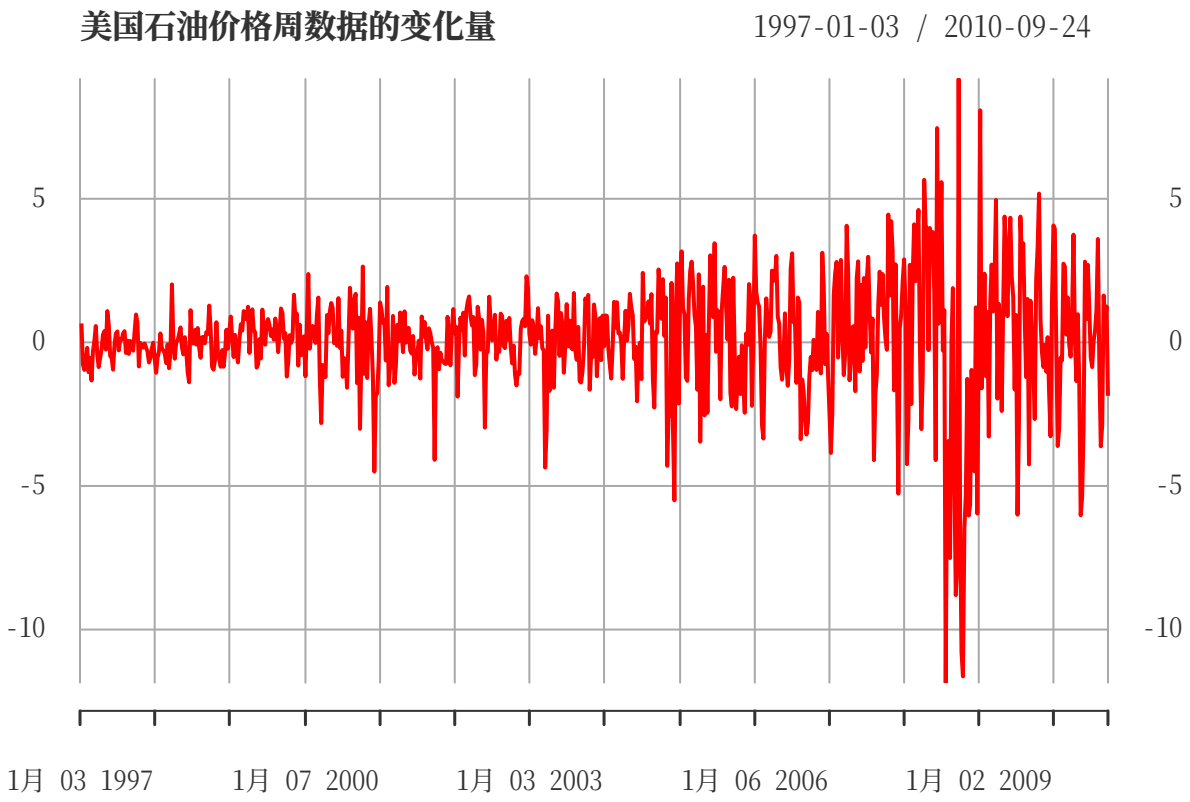


图 22.16: 美国石油价格周数据差分

ARMA-GARCH 模型用 ARMA 模型建立均值方程，用 GARCH 模型描述波动率，可以更好地解释上面的差分序列这样的具有波动率聚集的数据，也能改善预测。

记 C_t 为价格差分序列，对其建立平稳 ARMA 模型。ACF 图：

```
forecast::Acf(doil, lag.max = 30, main="")
```

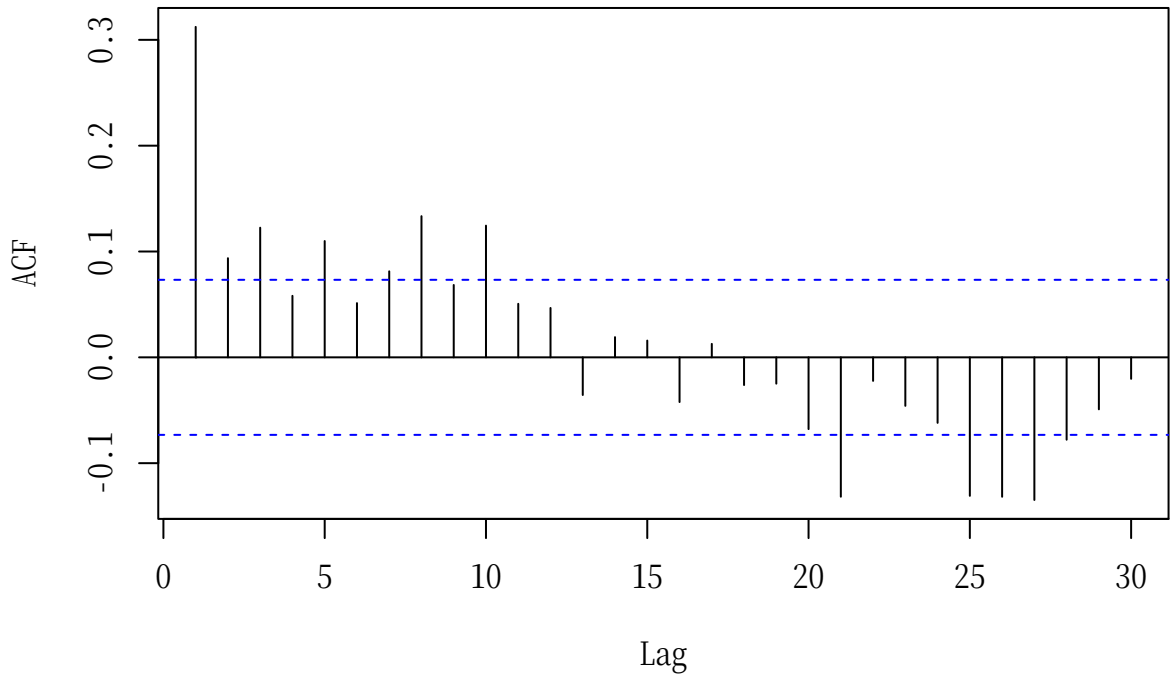


图 22.17: 美国石油价格周数据差分的 ACF

PACF 图：

```
pacf(doil, lag.max = 30, main="")
```

ACF 和 PACF 中可以看出没有单位根，相关性都不强，但是 PACF 中滞后 5、10、20、25 处显著，考虑用周期 5 的季节成分；除此之外 PACF 只有滞后 1、3 显著，可考虑 AR(3)。周期成分的 PACF 值也比较小，考虑低阶的周期成分，用 ARMA$(3,0)(2,0)_5$。

建模：

```
mod.oil1 <- arima(
  doil, order=c(3,0,0),
  seasonal=list(order=c(2,0,0), period=5)
)
mod.oil1
```

```
##
```

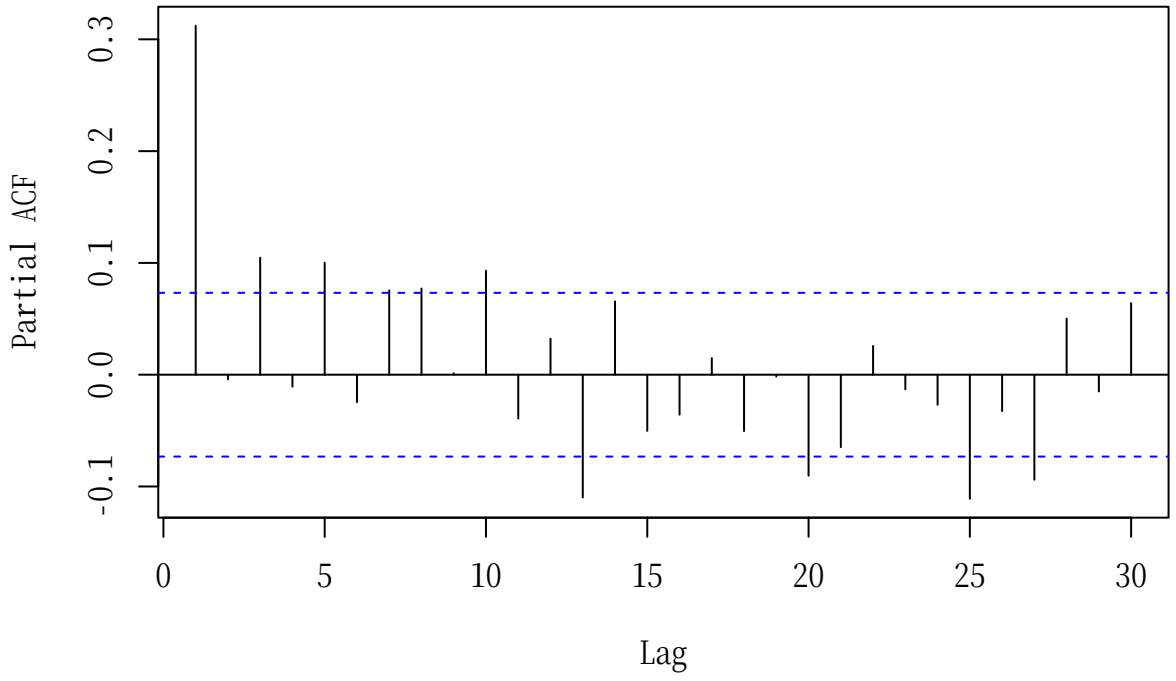


图 22.18: 美国石油价格周数据差分的 PACF

```
## Call:
## arima(x = doil, order = c(3, 0, 0), seasonal = list(order = c(2, 0, 0), period = 5))
##
## Coefficients:
##          ar1      ar2     ar3    sar1    sar2  intercept
##       0.3187  -0.0690  0.1071  0.0814  0.1177     0.0644
## s.e.  0.0372   0.0397  0.0375  0.0377  0.0376     0.1377
##
## sigma^2 estimated as 3.632:  log likelihood = -1477.91,  aic = 2969.82
```

其中的均值 (intercept) 不显著，去掉均值：

```
mod.oil2 <- arima(
  doil, order=c(3,0,0),
  seasonal=list(order=c(2,0,0), period=5),
  include.mean=FALSE
)
mod.oil2
```

```
##
## Call:
## arima(x = doil, order = c(3, 0, 0), seasonal = list(order = c(2, 0, 0), period = 5),
##     include.mean = FALSE)
##
## Coefficients:
##          ar1      ar2     ar3    sar1    sar2
##       0.3191  -0.0689  0.1075  0.0817  0.1181
## s.e.  0.0372   0.0397  0.0375  0.0377  0.0376
##
## sigma^2 estimated as 3.634:  log likelihood = -1478.02,  aic = 2968.04
```

系数只有 AR2 不显著，但也可以保留，模型为

$$(1-0.32B+0.07B^2-0.11B^3)(1-0.08B^5-0.13B^{10})C_t=\varepsilon_t,$$
$$\mathrm{Var}(\varepsilon_t)=3.634$$

模型诊断：

```
tsdiag(mod.oil2)
```

从模型诊断来看，残差存在波动率聚集，白噪声检验都是通过的。

因为 EGARCH 模型的程序不支持季节性的 ARMA，所以先建立一个纯季节模型，对残差再进一步建模。

```
mod.oil3 <- arima(
  doil, order=c(0,0,0),
  seasonal=list(order=c(2,0,0), period=5),
  include.mean=FALSE
```

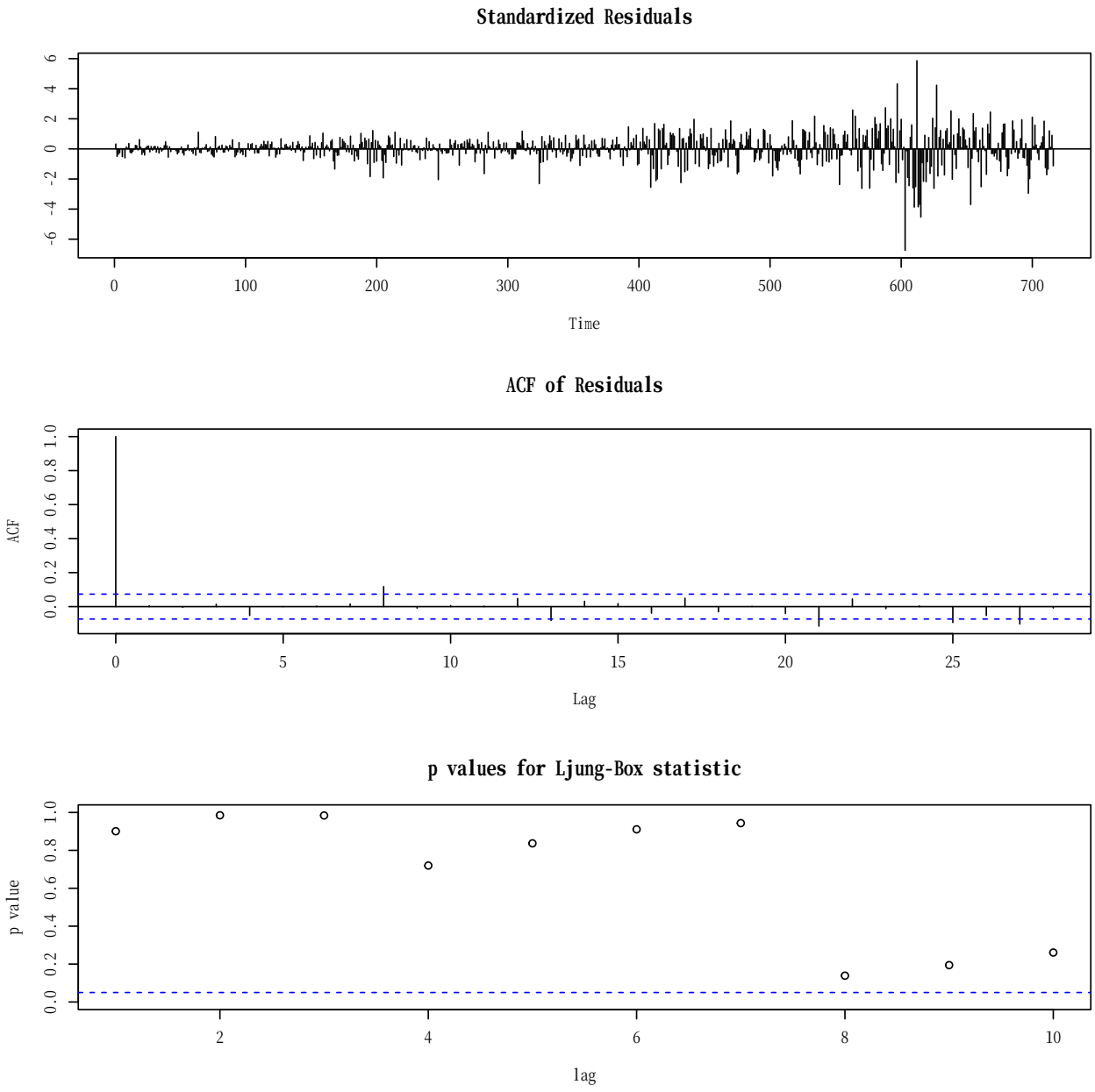


图 22.19: 石油价格模型诊断

```
)
mod.oil3
```

```
##
## Call:
## arima(x = doil, order = c(0, 0, 0), seasonal = list(order = c(2, 0, 0), period = 5),
##     include.mean = FALSE)
##
## Coefficients:
##          sar1    sar2
##       0.0983  0.1152
## s.e.  0.0371  0.0372
##
## sigma^2 estimated as 4.057:  log likelihood = -1517.42,  aic = 3040.84
```

模型为

$$(1-0.0983B^5-0.1152B^{10})C_t = C_t^*,\ t=11,12,\ldots,716$$

用三种不同的方法计算上述纯季节模型的残差：

```
ad1.oil <- residuals(mod.oil3) # 与输入序列等长
ad2.oil <- doil[11:716] - 0.0983*doil[6:711] - 0.1152*doil[1:706]
ad3.oil <- stats::filter(
  doil, c(1, rep(0,4), -0.0983, rep(0,4), -0.1152),
  method="convolution", side=1)
max(abs(ad1.oil[11:716] - ad2.oil))
```

```
## [1] 0.0004208447
```

```
max(abs(ad1.oil[11:716] - ad3.oil[11:716]))
```

```
## [1] 0.0004208447
```

```
adoil <- ad1.oil[11:716]
```

去掉季节项后的序列记为 C_t^*，其 ACF 和 PACF 为:

```
forecast::Acf(adoil, lag.max=30, main="")
```

```
pacf(adoil, lag.max=30, main="")
```

滞后 5、10、15 已经不显著，虽然滞后 20 和 25 显著但在较高的滞后阶。

对 C_t^* 建立 ARMA(3,0) 结合 GARCH(1,1) 的模型。

```
oil.mod4 <- garchFit(
  ~ arma(3,0) + garch(1,1),
  data=adoil,
  include.mean=FALSE, trace=FALSE
```

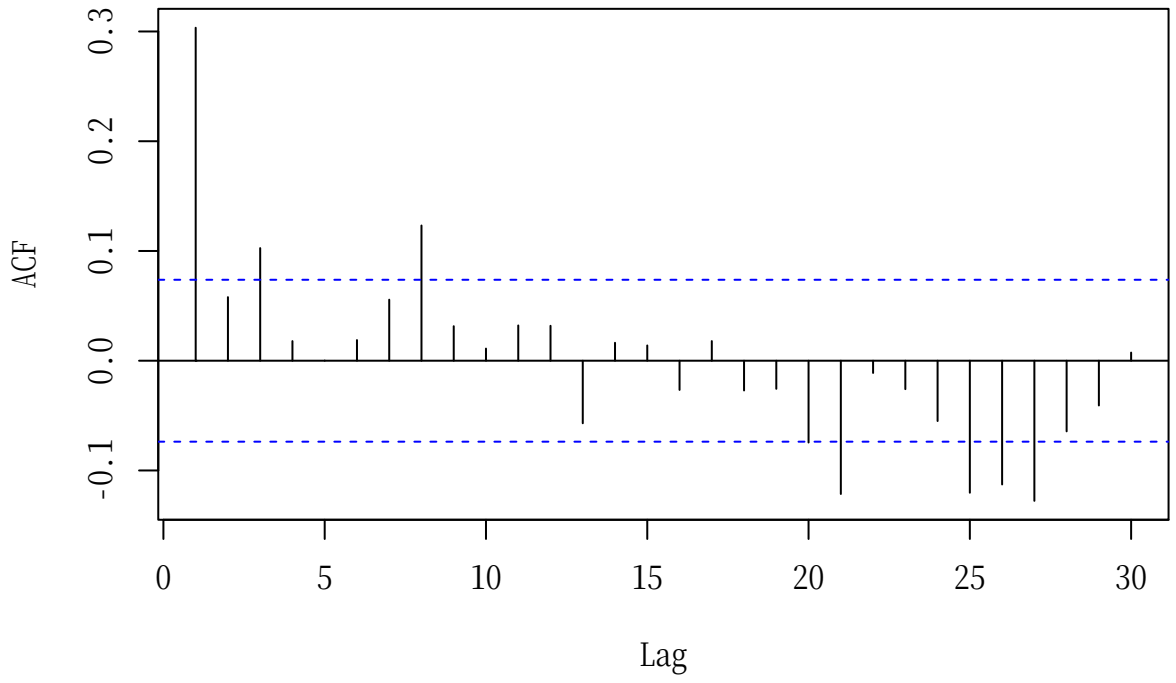


图 22.20: 去掉季节项后的 ACF

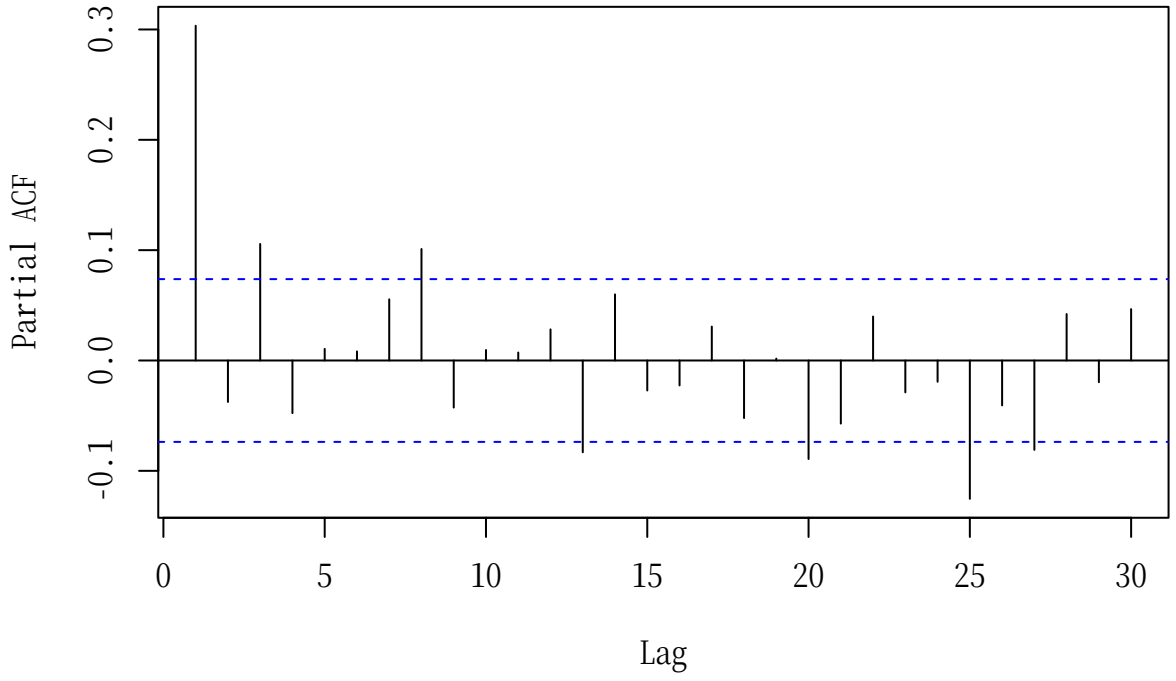


图 22.21: 去掉季节项后的 PACF

```
)
summary(oil.mod4)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(3, 0) + garch(1, 1), data = adoil, include.mean = FALSE,
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(3, 0) + garch(1, 1)
## <environment: 0x00000224b7e53788>
##  [data = adoil]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##       ar1        ar2        ar3      omega     alpha1      beta1
##  0.323494  -0.092111   0.040486   0.016751   0.090404   0.910225
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     0.323494    0.040310    8.025 1.11e-15 ***
## ar2    -0.092111    0.041690   -2.209   0.0271 *
## ar3     0.040486    0.040117    1.009   0.3129
## omega   0.016751    0.008171    2.050   0.0403 *
## alpha1  0.090404    0.015674    5.768 8.04e-09 ***
## beta1   0.910225    0.014837   61.349  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -1262.73    normalized:  -1.78857
##
## Description:
##  Tue May 30 17:04:22 2023 by user: Lenovo
##
```

```
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  174.5855  0
##  Shapiro-Wilk Test  R    W      0.9767892 3.798343e-09
##  Ljung-Box Test     R    Q(10)  15.30529  0.1213217
##  Ljung-Box Test     R    Q(15)  20.05642  0.1697791
##  Ljung-Box Test     R    Q(20)  22.75188  0.3011415
##  Ljung-Box Test     R^2  Q(10)  3.155885  0.9775137
##  Ljung-Box Test     R^2  Q(15)  8.029569  0.9225918
##  Ljung-Box Test     R^2  Q(20)  9.65231   0.9740473
##  LM Arch Test       R    TR^2   3.59896   0.9896355
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 3.594137 3.632887 3.593994 3.609110
```

估计的模型可以写成：

$$
\begin{aligned}
C_t^* =& 0.323 C_{t-1}^* - 0.092 C_{t-2}^* + 0.040 C_{t-3}^* + a_t \\
a_t =& \sigma_t \varepsilon_t,\ \epsilon_t \text{ iid } \mathrm{N}(0,1) \\
\sigma_t^2 =& 0.01675 + 0.0904 a_{t-1}^2 + 0.9102 \sigma_{t-1}^2
\end{aligned}
$$

其中除了 AR3 系数以外都显著。残差白噪声检验都通过了。

```
plot(oil.mod4, which=3)
```

```
plot(oil.mod4, which=13)
```

标准化残差比正态分布厚尾。

为了处理残差的厚尾性，采用标准化 t 分布的 GARCH 模型。

```
oil.mod5 <- garchFit(
  ~ arma(3,0) + garch(1,1),
  cond.dist="std",
  data=adoil,
  include.mean=FALSE, trace=FALSE
)
summary(oil.mod5)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(3, 0) + garch(1, 1), data = adoil, cond.dist = "std",
##      include.mean = FALSE, trace = FALSE)
##
```

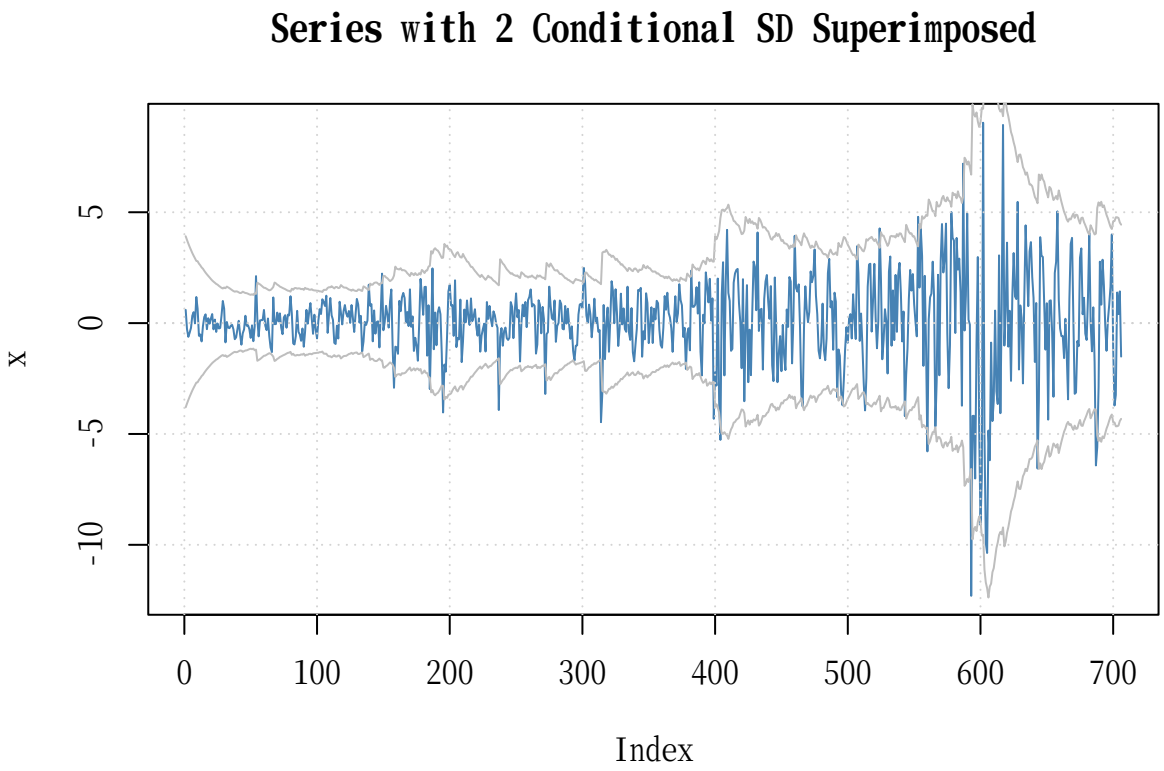


图 22.22: 去除季节项后的序列及两倍波动率

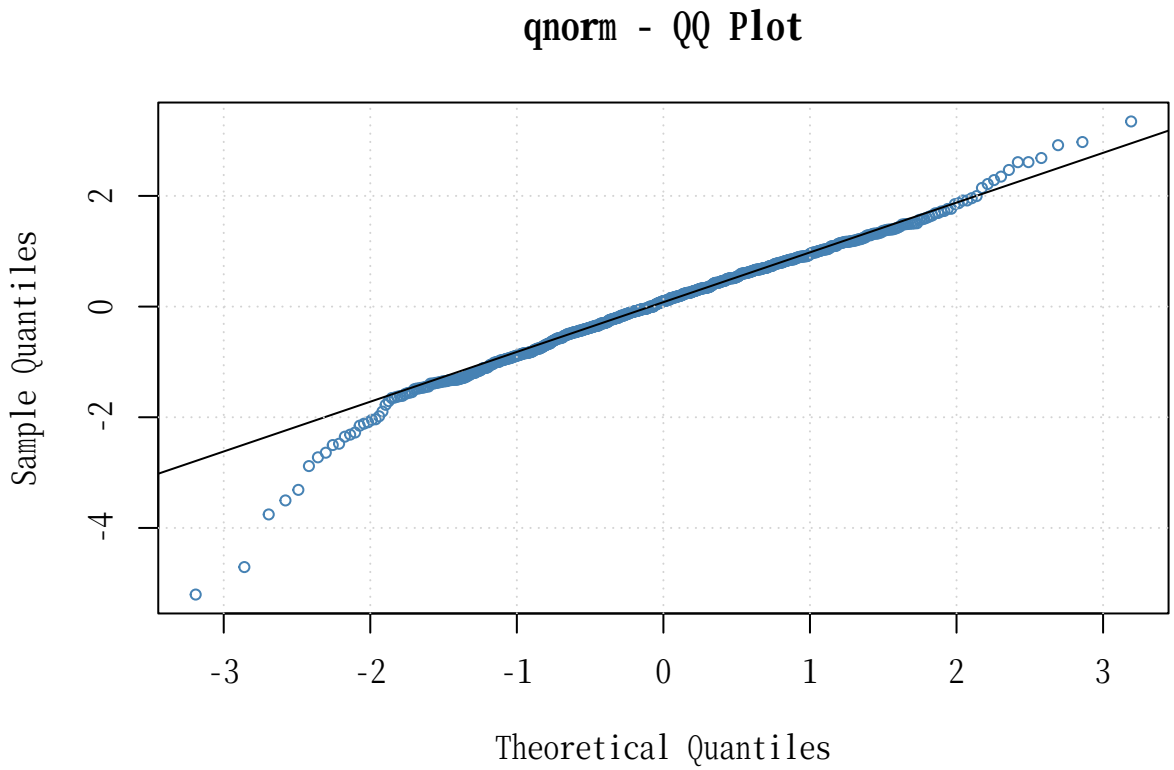


图 22.23: 去除季节项后的 ARMA-GARCH 标准化残差 QQ 图

```
## Mean and Variance Equation:
##  data ~ arma(3, 0) + garch(1, 1)
## <environment: 0x00000224b36a2930>
##  [data = adoil]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##       ar1        ar2        ar3      omega     alpha1      beta1      shape
##  0.325396  -0.065110   0.056019   0.011176   0.119701   0.891778   6.761292
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     0.325396    0.038778    8.391  < 2e-16 ***
## ar2    -0.065110    0.040791   -1.596    0.110
## ar3     0.056019    0.039464    1.420    0.156
## omega   0.011176    0.008849    1.263    0.207
## alpha1  0.119701    0.023879    5.013 5.36e-07 ***
## beta1   0.891778    0.019161   46.541  < 2e-16 ***
## shape   6.761292    1.582183    4.273 1.93e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -1245.39    normalized:  -1.764009
##
## Description:
##  Tue May 30 17:04:23 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  278.6969  0
##  Shapiro-Wilk Test  R    W      0.9715166 1.72838e-10
##  Ljung-Box Test     R    Q(10)  14.96489  0.1333472
##  Ljung-Box Test     R    Q(15)  20.36323  0.158437
##  Ljung-Box Test     R    Q(20)  22.95429  0.2910445
##  Ljung-Box Test     R^2  Q(10)  4.517928  0.9209735
##  Ljung-Box Test     R^2  Q(15)  8.788443  0.8883167
```

```
##  Ljung-Box Test      R^2  Q(20)  10.55469  0.9569638
##  LM Arch Test        R    TR^2   6.019291  0.9151065
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## 3.547847 3.593056 3.547653 3.565316
```

估计的模型可以写成：

$$
\begin{aligned}
C_t^* =& 0.325C_{t-1}^* - 0.065C_{t-2}^* + 0.056C_{t-3}^* + a_t \\
a_t =& \sigma_t \varepsilon_t,\ \epsilon_t \text{ iid } t^*(6.76) \\
\sigma_t^2 =& 0.0112 + 0.1197a_{t-1}^2 + 0.8918\sigma_{t-1}^2
\end{aligned}
$$

系数中 AR2, AR3 不显著，$\alpha_0 = 0.0112$ 不显著。白噪声检验都通过了。

作标准化残差相对于标准化 t 分布的 QQ 图：

```
plot(oil.mod5, which=13)
```

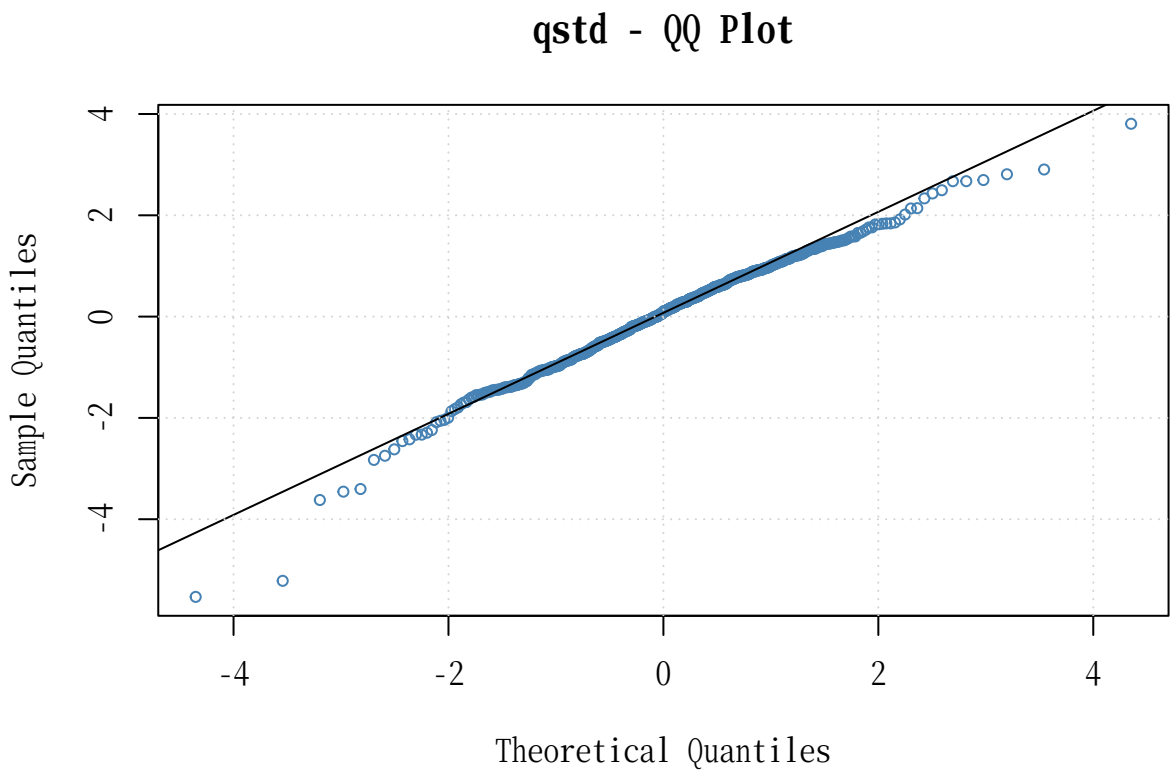


图 22.24: 去除季节项后的 t 分布 ARMA-GARCH 标准化残差 QQ 图

这时厚尾性不太明显，但出现了左偏。

于是，改用有偏的标准化 t 分布作为 GARCH 模型的条件分布：

```r
oil.mod6 <- garchFit(
  ~ arma(1,0) + garch(1,1),
  cond.dist="sstd",
  data=adoil,
  include.mean=FALSE, trace=FALSE
)
summary(oil.mod6)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(1, 0) + garch(1, 1), data = adoil, cond.dist = "sstd",
##      include.mean = FALSE, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 1)
## <environment: 0x00000224b5146890>
##  [data = adoil]
##
## Conditional Distribution:
##  sstd
##
## Coefficient(s):
##      ar1     omega    alpha1     beta1      skew     shape
## 0.294919  0.011334  0.121624  0.892569  0.861831  6.470838
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1     0.294919    0.036755    8.024 1.11e-15 ***
## omega   0.011334    0.009021    1.256    0.209
## alpha1  0.121624    0.024040    5.059 4.21e-07 ***
## beta1   0.892569    0.018457   48.359  < 2e-16 ***
## skew    0.861831    0.047753   18.048  < 2e-16 ***
## shape   6.470838    1.532285    4.223 2.41e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
```

```
##  -1243.852    normalized:  -1.76183
##
## Description:
##  Tue May 30 17:04:23 2023 by user: Lenovo
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  268.3856  0
##  Shapiro-Wilk Test  R    W      0.9721041 2.393857e-10
##  Ljung-Box Test     R    Q(10)  17.27448  0.06850684
##  Ljung-Box Test     R    Q(15)  22.22733  0.1019802
##  Ljung-Box Test     R    Q(20)  24.927    0.204238
##  Ljung-Box Test     R^2  Q(10)  4.370554  0.9290864
##  Ljung-Box Test     R^2  Q(15)  8.949599  0.880138
##  Ljung-Box Test     R^2  Q(20)  10.7288   0.9529546
##  LM Arch Test       R    TR^2   5.740867  0.9285771
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 3.540657 3.579407 3.540514 3.555630
```

估计的模型可以写成：

$$C_t^* = 0.295 C_{t-1}^* + a_t \tag{22.5}$$

$$a_t = \sigma_t \varepsilon_t,\ \epsilon_t \text{ iid } t^*(6.76, 0.862) \tag{22.6}$$

$$\sigma_t^2 = 0.0113 + 0.1216 a_{t-1}^2 + 0.8926 \sigma_{t-1}^2 \tag{22.7}$$

其中参数 $0.8618 < 1$ 表示左偏，且其近似 95% 置信区间为 $(0.766, 0.957)$ 不包含 1，所以可以认为其显著地偏离 1。

标准化残差相对有有偏的标准化 t 分布的 QQ 图：

```
plot(oil.mod6, which=13)
```

这个图形说明有偏 t 分布拟合数据很好。模型残差的白噪声检验也都通过了。

模型的标准化残差的时间序列图：

```
plot(oil.mod6, which=9)
```

C_t^* 序列叠加了两倍波动率的图形：

```
plot(oil.mod6, which=3)
```

从样本内比较来看，最后得到的去除季节项后的 AR(1)-GARCH(1,1) 用有偏 t 分布的模型 (22.7)效果最好。

对于样本外比较，可以用季节调整后的 C_t^* 作为标准进行回测比较。C_t^* 共有 706 个观测，从点 $t = 650$ 开始对最后的 56 个观测逐个地进行超前一步和超前两步预报。采用如下的模型：

- AR(3) 模型;

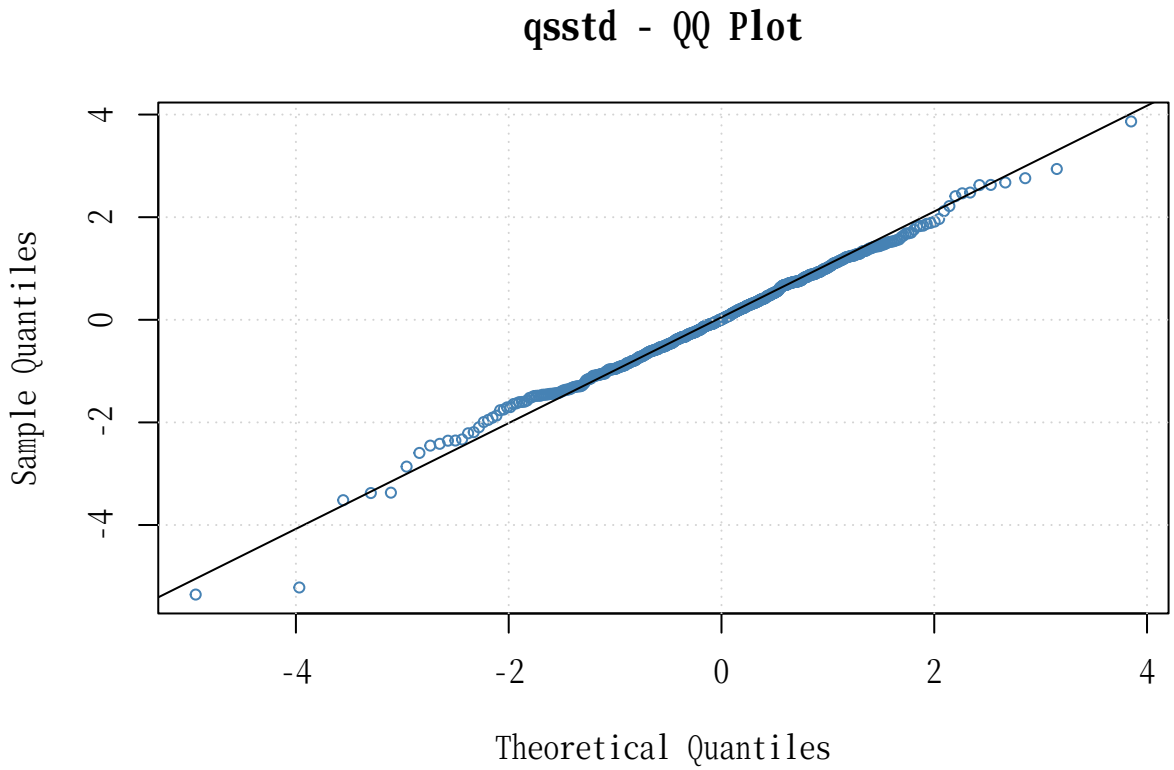


图 22.25: 去除季节项后的有偏 t 分布 ARMA-GARCH 标准化残差 QQ 图

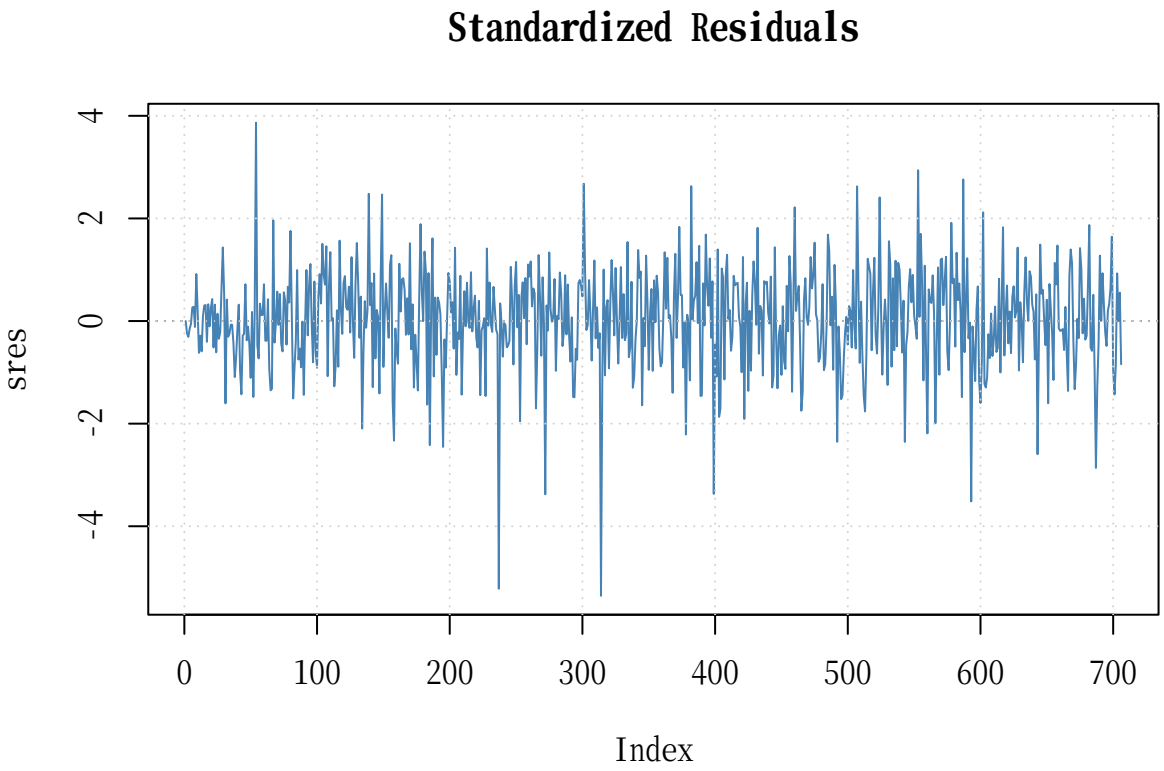


图 22.26: 去除季节项后的有偏 t 分布 ARMA-GARCH 标准化残差

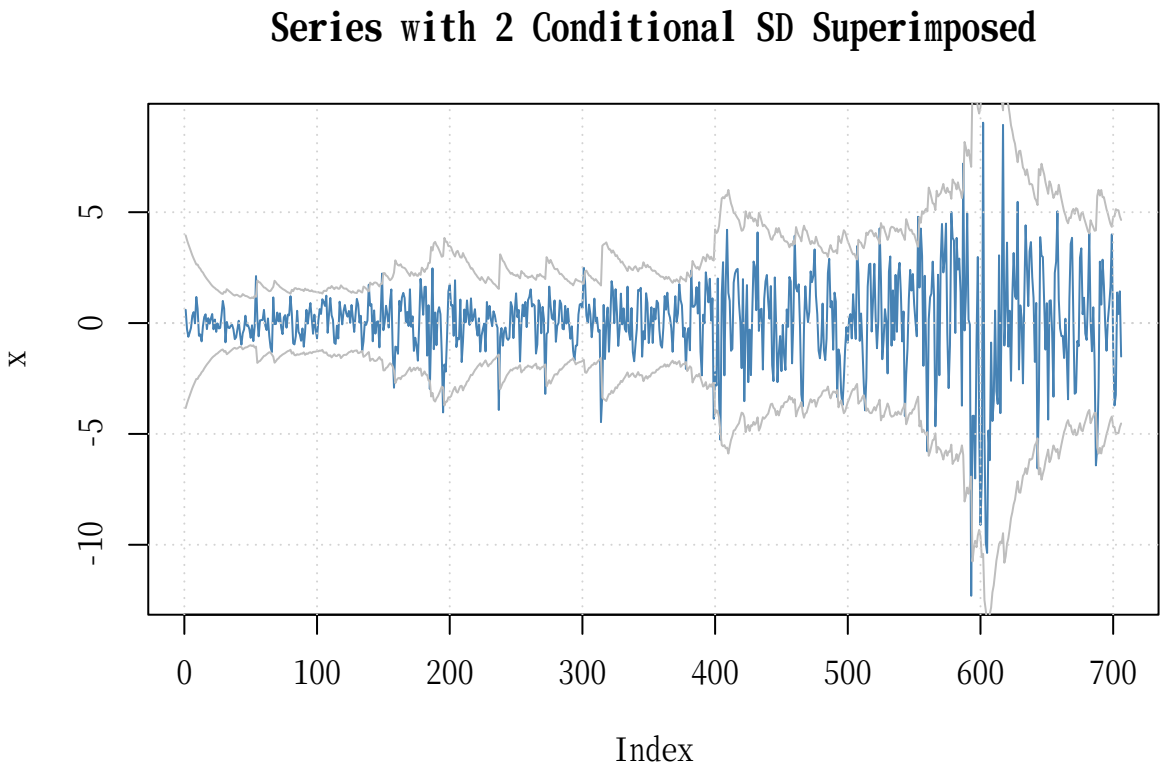


图 22.27: 去除季节项后的序列及有偏 t 分布 ARMA-GARCH 两倍波动率

- 条件分布为有偏 t 分布的 AR(1)–GARCH(1,1) 模型。

AR(3) 的超前一步和超前两步预测均方误差分别为 2.368 和 2.558，AR-GARCH 模型的结果为 2.270 和 2.436，均值的预测也有所改善。当然，AR-GARCH 的预测区间也更精确。

Part IV

多元时间序列模型

Chapter 23

多元时间序列基本概念

经济的全球一体化和信息传播的发展使得各国的金融市场相互关联，一个市场的价格变动可以很快地扩散到另一个市场。持有多个资产的投资者也希望了解多个资产的收益率之间的关系。这些问题属于多元时间序列分析的范畴。

多元时间序列包含多个一元时间序列作为分量，各个一元时间序列的采样时间点相同，所以数据可以用矩阵形式表示，每行为一个时间点，每列为一个一元时间序列。在 R 中可以保存为矩阵、数据框、ts 或者 xts 时间序列对象。设 $r_t = (r_{1t}, \dots, r_{kt})^T$ 表示 k 个资产在时刻 t 的对数收益率。

一元时间序列的某些方法可以推广到多元情形，但是有些问题需要注意。某些情况下需要提出新的模型和方法。

23.1 弱平稳与互相关矩阵

23.1.1 弱平稳列

考虑一个 k 元时间序列 $r_t = (r_{1t}, \dots, r_{kt})^T$。称 r_t 是弱平稳的，如果

$$\begin{cases} Er_t = \mu \text{ 与}t\text{无关} \\ \mathrm{Var}(r_t) = \Gamma_0 \text{ 与}t\text{无关} \\ \mathrm{Cov}(r_t, r_{t-l}) = \Gamma_l,\ l = 0, 1, 2, \dots \text{ 与}t\text{无关} \end{cases}$$

23.1.2 互相关阵

对 k 元弱平稳列 r_t，$\Gamma_0 = \mathrm{Var}(r_t)$ 是 r_t 协方差阵，是一个对称半正定 k 阶方阵，记 $\Gamma_0 = (\Gamma_{ij}(0))_{k\times k}$，则 $\Gamma_{ii}(0)$ 是分量 r_{it} 的方差，$\Gamma_{ij}(0)$ 是分量 r_{it} 与分量 r_{jt} 的协方差。

为了将协方差阵变成相关阵，记

$$D = \mathrm{diag}(\sqrt{\Gamma_{11}(0)}, \dots, \sqrt{\Gamma_{kk}(0)})$$

令

$$\rho_0 = (\rho_{ij}(0))_{k\times k} = D^{-1}\Gamma_0 D^{-1}$$

则 ρ_0 是随机向量 r_t 的相关阵，称为多元时间序列 $\{r_t\}$ 的同步的或者滞后为 0 的互相关阵。其元素

$$\rho_{ij}(0) = \mathrm{corr}(r_{it}, r_{jt}) = \frac{\mathrm{Cov}(r_{it}, r_{jt})}{\sqrt{\mathrm{Var}(r_{it})\mathrm{Var}(r_{jt})}} = \frac{\Gamma_{ij}(0)}{\sqrt{\Gamma_{ii}(0)\Gamma_{jj}(0)}}$$

ρ_0 是一个对角线元素全为 1 的对称半正定阵，且 $|\rho_{ij}(0)| \leq 1$，$\rho_{ij}(0) = \rho_{ji}(0)$。称 $\rho_{ij}(0)$ 为共点或同步相关系数，因为它是两个分量在同一时刻 t 的相关系数。

多元时间序列分析中一个重要概念是引导与滞后关系。为此，用互相关阵来衡量时间序列之间的线性关系的强度。k 元弱平稳列 r_t 的滞后 l 的互协方差阵定义为

$$\Gamma_l = (\Gamma_{ij}(l))_{k\times k} = E[(r_t - \mu)(r_{t-l} - \mu)^T]$$

这是一元时间序列的自协方差函数 γ_l 的推广。Γ_l 仅依赖于滞后 l 而与时刻 t 无关。

k 元弱平稳列 r_t 的滞后 l 的互相关阵 (Cross Correlation Matrix, CCM) 定义为

$$\rho_l = (\rho_{ij}(l))_{k\times k} = D^{-1}\Gamma_l D^{-1}$$

其中

$$\rho_{ij}(l) = \text{corr}(r_{it}, r_{j,t-l}) = \frac{\Gamma_{ij}(l)}{\sqrt{\Gamma_{ii}(0)\Gamma_{jj}(0)}}$$

是 r_{it} 和 $r_{j,t-l}$ 的相关系数。注意 $\text{Var}(r_{j,t-l}) = \Gamma_{jj}(0)$。

对 $l > 0$，如果 $\rho_{ij}(l) \neq 0$，可称先观测到的分量 $r_{j,t-l}$ 对滞后的 $r_{i,t}$ 有先导作用。而 $\rho_{ji}(l)$ 代表的是 $r_{i,t-l}$ 对 $r_{j,t}$ 的先导作用。一般 $\rho_{ij}(l) \neq \rho_{ji}(l)$，所以 $l \neq 0$ 时 ρ_l 一般不是对称阵，Γ_l 一般也不对称。

不同于一元时间序列的自协方差满足 $\gamma_l = \gamma_{-l}$，对 k 元时间序列有

$$\begin{aligned}\Gamma_{ij}(l) =& \text{Cov}(r_{it}, r_{j,t-l}) = \text{Cov}(r_{j,t-l}, r_{i,t}) = \text{Cov}(r_{j,t}, r_{i,t+l}) \\ =& \text{Cov}(r_{j,t}, r_{i,t-(-l)}) = \Gamma_{ji}(-l)\end{aligned}$$

即

$$\Gamma_{-l} = \Gamma_l^T$$

对互相关阵 ρ_l 也有

$$\rho_{-l} = \rho_l^T$$

所以只需要考虑 $\rho_l, l \geq 0$。

23.1.3 时间序列之间的线性相依性的分类

多元弱平稳列的互相关阵 $\{\rho_l, l = 0, 1, ...\}$ 包含如下方面的信息：

- 对角线元素 $\{r_{ii}(l), l = 0, 1, ...\}$ 是一元时间序列 r_{it} 的自相关函数 (ACF);
- $\rho_{ij}(0)$ 是两个分量 r_{it} 与 r_{jt} 的同步线性关系；
- 对 $l > 0$，$\rho_{ij}(l)$ 表示 r_{it} 对另一分量的过去值 $r_{j,t-l}$ 的线性依赖。

如果对 $\forall l > 0$，都有 $\rho_{ij}(l) = 0$, 则 r_{it} 与过去的 $r_{j,t-l}$ 都不相关。

两个分量 r_{it} 和 r_{js} 的线性相依关系可以分类为如下情况：

- 若 $\forall l \geq 0$，$\rho_{ij}(l) = \rho_{ji}(l) = 0$，则两个分量 r_{it} 和 r_{js} 不相关 (对任意 s, t)。
- 若 $\rho_{ij}(0) \neq 0$，则两个分量 r_{it} 和 r_{jt} 具有同步相关。
- 若 $\forall l > 0$, $\rho_{ij}(l) = \rho_{ji}(l) = 0$，则两个分量 r_{it} 和 r_{js} 没有引导与滞后的关系但是可能有同步相关。称这两个序列是分离的。
- 如果 $\forall l > 0$，$\rho_{ij}(l) = 0$，但存在 $v > 0$ 使得 $\rho_{ji}(v) \neq 0$，则 r_{it} 不依赖于过去的 $r_{j,t-l}$，但是 r_{jt} 依赖于过去的 $r_{i,t-v}$，这时从 $r_i(t)$ 到 $r_j(s)$ 有一个单向的引导（领先）关系。
- 如果存在 $l > 0$ 和 $v > 0$ 使得 $\rho_{ij}(l) \neq 0$, $\rho_{ji}(v) \neq 0$, 则 r_{it} 和 r_{js} 之间存在相互的反馈关系，互为引导和滞后。

23.2 样本互相关阵

类似于一元时间序列的自协方差函数估计 $\hat{\gamma}_l$，将互协方差阵 Γ_l 估计为

$$\hat{\Gamma}_l = \frac{1}{T}\sum_{t=l+1}^{T}(r_t - \bar{r})(r_{t-l} - \bar{r})^T$$

令 $\hat{D}$ 为 $\hat{\Gamma}_0$ 的对角线元素的平方根组成的对角阵，估计互相关阵 ρ_l 为

$$\hat{\rho}_l = \hat{D}^{-1}\hat{\Gamma}_l\hat{D}^{-1}$$

对二维时间序列 (x_t, y_t), $t = 1, 2, \ldots, T$，R 的 `ccf(x, y, type="covariance")` 可以计算互协方差函数并作图:

$$\hat{\gamma}_{xy}(l) = \frac{1}{T}\sum_{t=l+1}^{T}(x_t - \bar{x})(y_{t-l} - \bar{y}),\ l = \ldots, -2, -1, 0, 1, 2, 3, \ldots,$$

用 `ccf(x, y)` 可以计算互相关系数函数并作图：

$$\hat{\rho}_{xy}(l) = \frac{\hat{\gamma}_{xy}(l)}{\sqrt{\hat{\gamma}_{xx}(0)\hat{\gamma}_{yy}(0)}},\ l = \ldots, -2, -1, 0, 1, 2, 3, \ldots.$$

对 $l > 0$，$\hat{\rho}_{xy}(l)$ 表示 y 对 x 的领先影响；对 $l < 0$，$\hat{\rho}_{xy}(l)$ 表示 x 对 y 的领先影响。

如果 r_t 是白噪声列，即 ρ_l 对 $l > 0$ 都是零矩阵，则在 Γ_0 正定的条件下，有

$$\mathrm{Var}(\hat{\rho}_{ij}(l)) \approx \frac{1}{T},\ l > 0$$

因为 $\hat{\rho}_l$ 的元素有 i, j, l 三个下标，所以比较难以快速地辨识一个多元时间序列的 CCM 的特点。

为此，(Tiao & Box, 1981) 提出了一种显示简略的互相关阵的方法，对样本相关阵 $\hat{\rho}_l$，显示如下的符号矩阵 s_l：

$$s_{ij}(l) = \begin{cases} + & \text{当}\hat{\rho}_{ij}(l) > \frac{2}{\sqrt{T}} \\ - & \text{当}\hat{\rho}_{ij}(l) < -\frac{2}{\sqrt{T}} \\ . & \text{当}|\hat{\rho}_{ij}(l)| \le \frac{2}{\sqrt{T}} \end{cases}$$

显示为 $+$ 或 $-$ 号的互相关系数在 0.05 渐近水平下显著不等于零。

23.2.1 例 1.1：IBM 和标普

多元时间序列的一些计算使用蔡瑞胸 (R.S. Tsay) 教授的 MTS 扩展包。

考虑 IBM 股票与标普 500 指数从 1926 年 1 月到 2008 年 12 月的月对数收益率，共 996 个观测。单位为百分之一。

```
da1 <- read_table(
  "m-ibm3dx2608.txt",
  col_types=cols(.default=col_double(),
  date=col_date(format="%Y%m%d")))
ts.mibm <- ts(100*log(1 + da1[["ibmrtn"]]),
              start=c(1926,1), frequency=12)
ts.msp5 <- ts(100*log(1 + da1[["sprtn"]]),
              start=c(1926,1), frequency=12)
```

IBM 的月对数收益率时间序列图：

```
plot(ts.mibm, xlab="Year", ylab="log(IBM Return)")
```

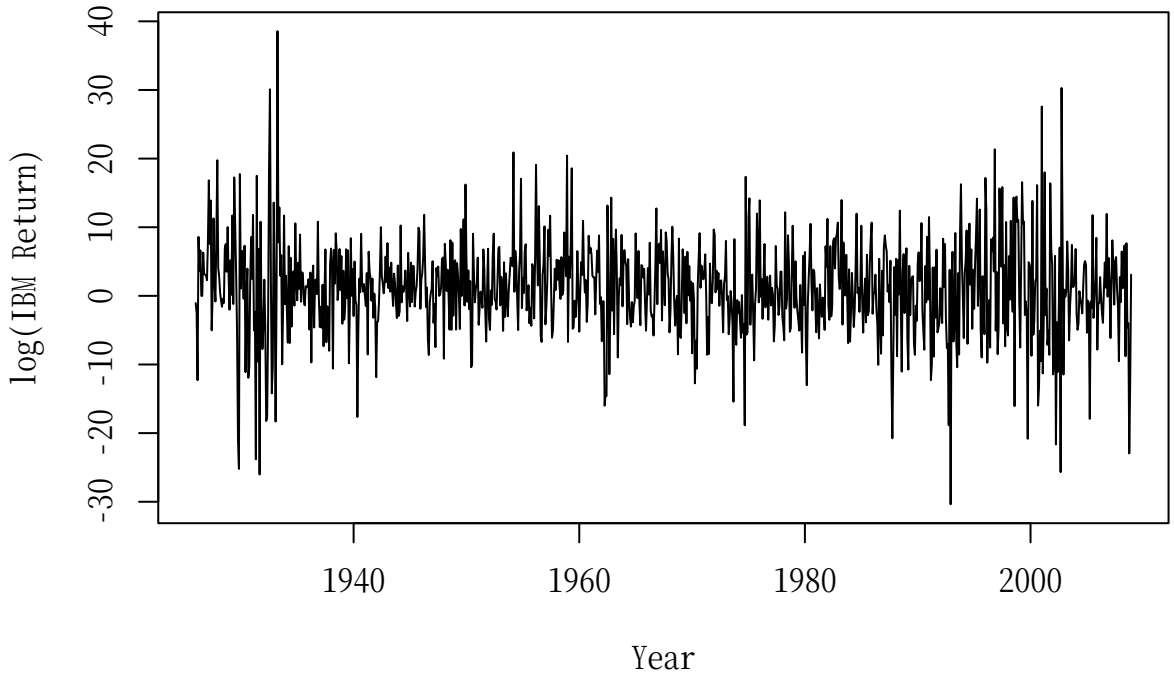


图 23.1: IBM 股票月对数收益率

标普 500 的月对数收益率时间序列图：

```
plot(ts.msp5, xlab="Year", ylab="log(SP500 Return)")
```

两个序列同时刻的散点图：

```
plot(as.vector(ts.msp5), as.vector(ts.mibm),
     xlab=" 标普 500[t]", ylab="IBM[t]")
```

同时刻具有明显的正相关，样本相关系数：

```
cor.test(as.vector(ts.msp5), as.vector(ts.mibm))
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.vector(ts.msp5) and as.vector(ts.mibm)
## t = 26.625, df = 994, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
```

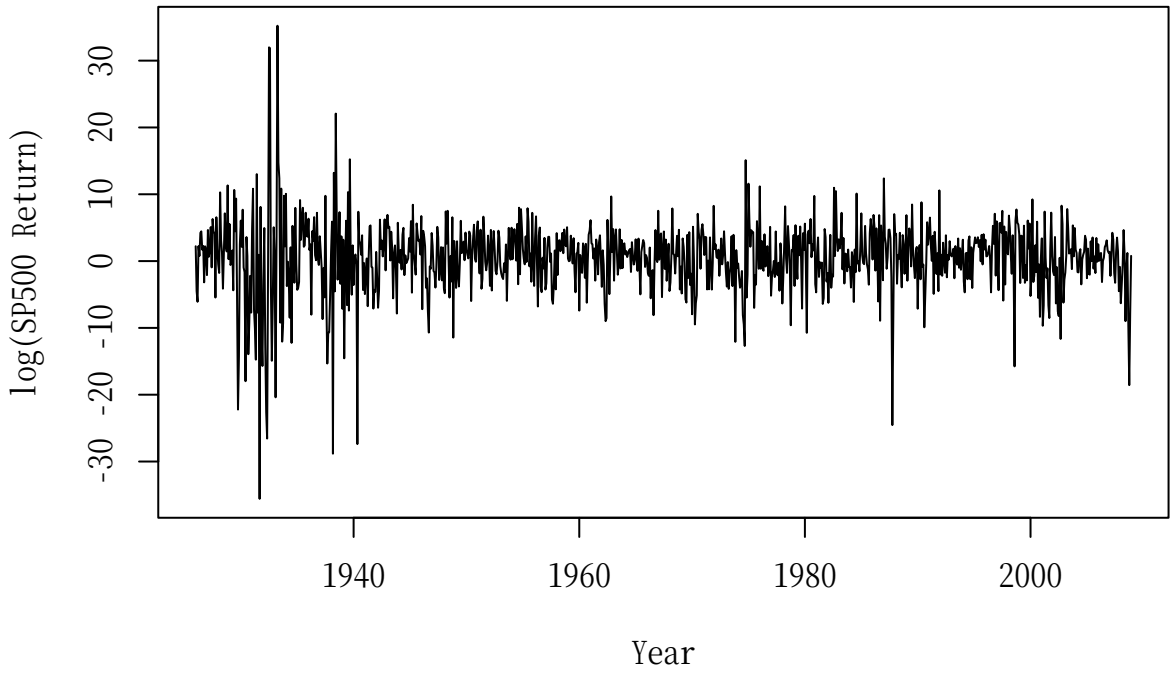


图 23.2: 标普 500 指数月对数收益率

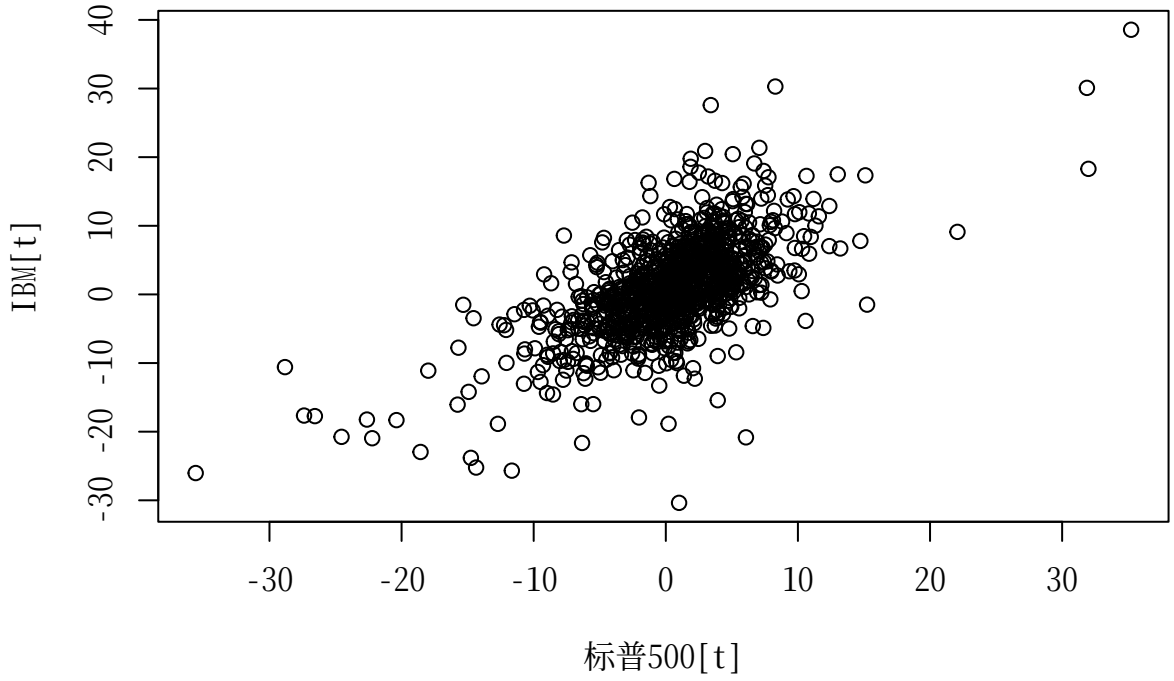


图 23.3: IBM 股票和标普 500 同时刻的月对数收益率

```
##  0.6074247 0.6800598
## sample estimates:
##       cor
## 0.6451977
```

同时刻的相关系数为 0.65，显著。

IBM 对领先一个月的标普 500：

```
plot(as.vector(ts.msp5)[-length(ts.mibm)],
     as.vector(ts.mibm)[-1],
     xlab=" 标普 500[t-1]", ylab="IBM[t]")
```

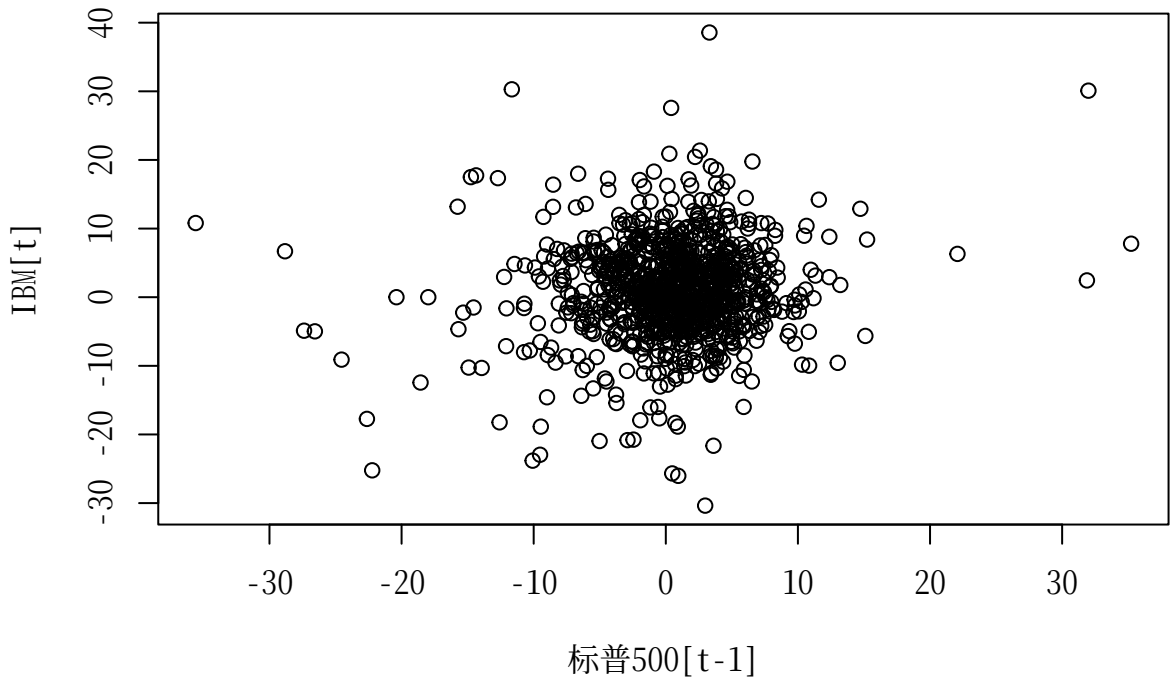


图 23.4: IBM 股票对滞后一步的标普 500 的月对数收益率

```
cor.test(as.vector(ts.msp5)[-length(ts.mibm)],
         as.vector(ts.mibm)[-1])
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.vector(ts.msp5)[-length(ts.mibm)] and as.vector(ts.mibm)[-1]
## t = 3.093, df = 993, p-value = 0.002037
## alternative hypothesis: true correlation is not equal to 0
```

```
## 95 percent confidence interval:
##  0.03575303 0.15886892
## sample estimates:
##        cor
## 0.09768469
```

相关系数很小，但显著不为零。

标普 500 对领先一个月的 IBM：

```
plot(as.vector(ts.mibm)[-length(ts.mibm)],
     as.vector(ts.msp5)[-1],
     xlab="IBM[t-1]", ylab=" 标普 500[t]")
```

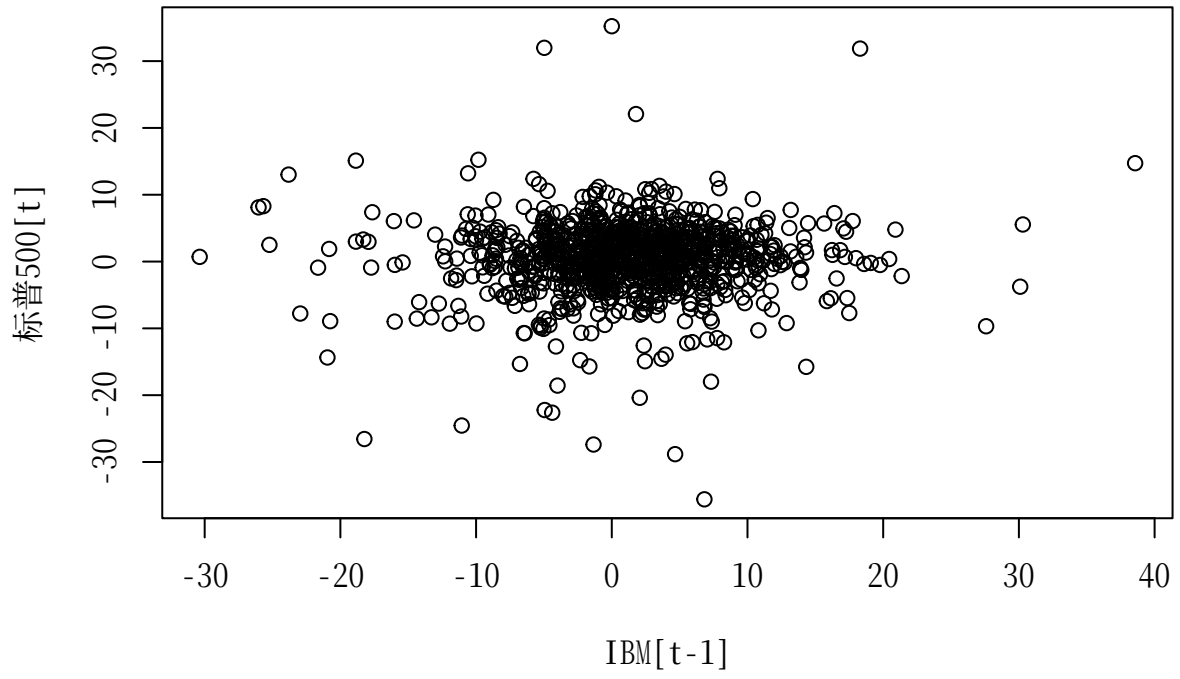


图 23.5: 标普 500 对滞后一步的 IBM 股票的月对数收益率

```
cor.test(as.vector(ts.mibm)[-length(ts.mibm)],
         as.vector(ts.msp5)[-1])
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.vector(ts.mibm)[-length(ts.mibm)] and as.vector(ts.msp5)[-1]
## t = 1.1803, df = 993, p-value = 0.2382
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.02477745  0.09934653
## sample estimates:
##       cor
## 0.0374289
```

相关系数很小，与零无显著差异。

用 MTS 包的 `MTSplot()` 函数绘制时间序列图：

```
MTS::MTSplot(cbind(ts.mibm, ts.msp5))
```

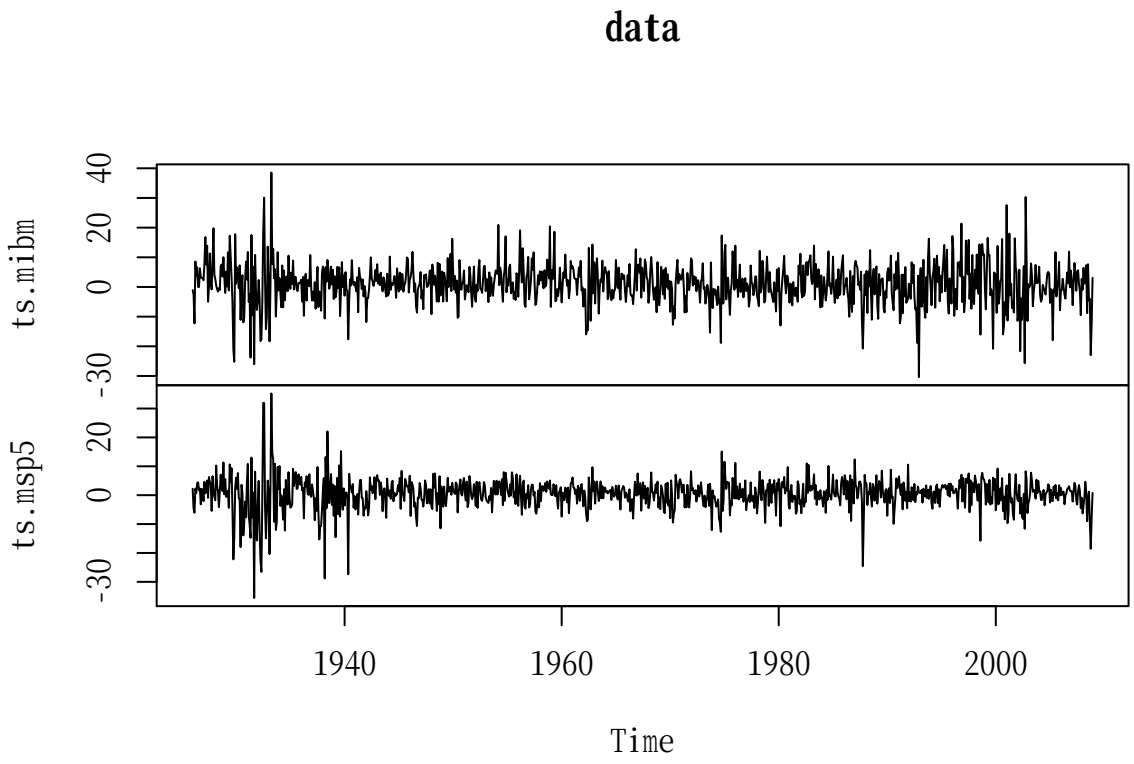


图 23.6: IBM 和标普 500 的对数收益率

计算两个序列的单样本简单样本统计量：

```
fBasics::basicStats(cbind(ts.mibm, ts.msp5))
```

```
##                 ts.mibm    ts.msp5
## nobs         996.000000 996.000000
## NAs            0.000000   0.000000
## Minimum      -30.368274 -35.585100
## Maximum       38.566232  35.222044
## 1. Quartile   -2.800141  -2.032465
```

```
## 3. Quartile    5.039744   3.546559
## Mean           1.089135   0.430068
## Median         1.095871   0.897907
## Sum         1084.778282 428.348045
## SE Mean        0.222860   0.175458
## LCL Mean       0.651806   0.085759
## UCL Mean       1.526464   0.774378
## Variance      49.467724  30.662294
## Stdev          7.033329   5.537354
## Skewness      -0.067766  -0.521287
## Kurtosis       2.621657   7.926907
```

从超额峰度 (Kurtosis) 来看两个序列有重尾分布，而标普的重尾分布更明显。

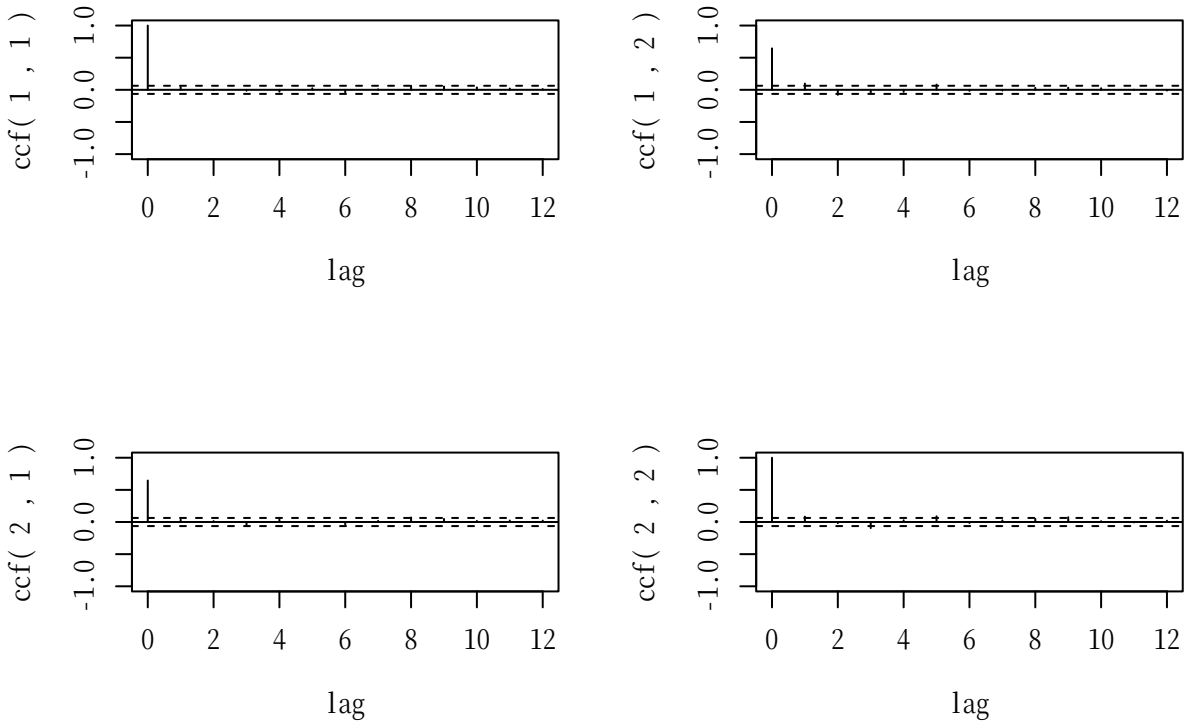
用 MTS 包的 `ccm()` 函数显示简约的互相关阵序列和图形。R.S Tsay 教授的 MTS 包的 `ccm()` 函数的源代码见后面，略作修改。

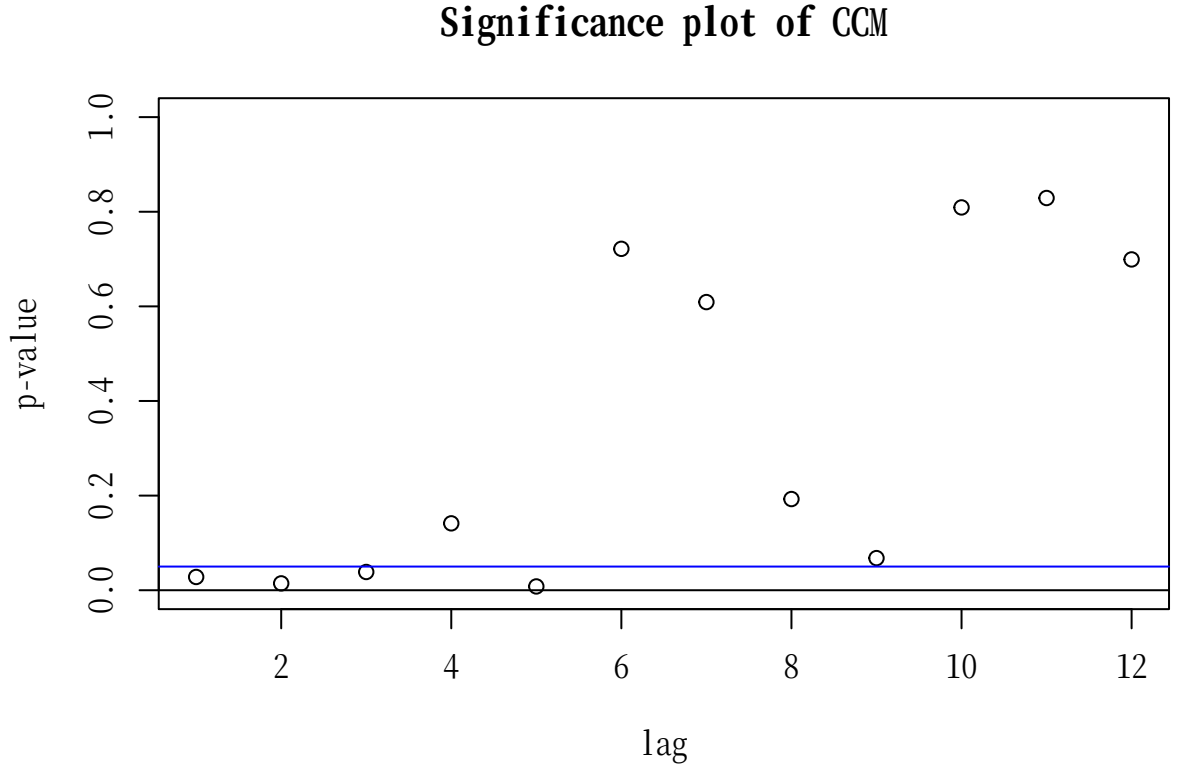
```
ccm(cbind(ts.mibm, ts.msp5), level=TRUE)
```

```
## [1] "Covariance matrix:"
##         ts.mibm ts.msp5
## ts.mibm    49.5    25.1
## ts.msp5    25.1    30.7
## CCM at lag:  0
##       [,1]  [,2]
## [1,] 1.000 0.645
## [2,] 0.645 1.000
## Simplified matrix:
## CCM at lag:  1
## . +
## . +
## Correlations:
##        [,1]   [,2]
## [1,] 0.0443 0.0976
## [2,] 0.0374 0.0836
## CCM at lag:  2
## . -
## . .
## Correlations:
##         [,1]    [,2]
## [1,] 0.00071 -0.0779
## [2,] 0.01505 -0.0178
## CCM at lag:  3
## . .
## . -
```

```
## Correlations:
##         [,1]    [,2]
## [1,] -0.0139 -0.0581
## [2,] -0.0524 -0.0951
## CCM at lag:  4
## . .
## . .
## Correlations:
##         [,1]    [,2]
## [1,] -0.0282 -0.0312
## [2,]  0.0384  0.0269
## CCM at lag:  5
## . +
## . +
## Correlations:
##         [,1]   [,2]
## [1,] 0.02370 0.0812
## [2,] 0.00452 0.0890
## CCM at lag:  6
## . .
## . .
## Correlations:
##         [,1]    [,2]
## [1,] -0.0397 -0.0166
## [2,] -0.0390 -0.0154
## CCM at lag:  7
## . .
## . .
## Correlations:
##         [,1]    [,2]
## [1,] 0.00822 -0.0109
## [2,] 0.01813  0.0266
## CCM at lag:  8
## . .
## + .
## Correlations:
##        [,1]   [,2]
## [1,] 0.0634 0.0301
## [2,] 0.0717 0.0480
## CCM at lag:  9
## . .
## . +
## Correlations:
```

```
##        [,1]   [,2]
## [1,] 0.0515 0.0331
## [2,] 0.0482 0.0749
## CCM at lag:  10
## . .
## . .
## Correlations:
##        [,1]   [,2]
## [1,] 0.0383 0.0256
## [2,] 0.0166 0.0133
## CCM at lag:  11
## . .
## . .
## Correlations:
##        [,1]     [,2]
## [1,] 0.0207  0.00321
## [2,] 0.0237 -0.00659
## CCM at lag:  12
## . .
## . .
## Correlations:
##        [,1]    [,2]
## [1,] 0.0122 -0.0118
## [2,] 0.0225  0.0203
```


ccf(1 , 1)
ccf(1 , 2)
ccf(2 , 1)
ccf(2 , 2)
-1.0 0.0 1.0
0 2 4 6 8 10 12
lag


Significance plot of CCM
p-value
0.0 0.2 0.4 0.6 0.8 1.0
2 4 6 8 10 12
lag

记 IBM 对数收益率为 r_{1t}，记标普 500 对数收益率为 r_{2t}，记 $r_t = (r_{1t}, r_{2t})^T$，则 CCM 为

$$\begin{aligned}\rho_l &= \begin{pmatrix} \rho_{11}(l) & \rho_{12}(l) \\ \rho_{21}(l) & \rho_{22}(l) \end{pmatrix} \\ &= \begin{pmatrix} \text{corr}(x_{1t}, x_{1,t-l}) & \text{corr}(x_{1t}, x_{2,t-l}) \\ \text{corr}(x_{2t}, x_{1,t-l}) & \text{corr}(x_{2t}, x_{2,t-l}) \end{pmatrix}\end{aligned}$$

结果中给出了 r_t 的协方差阵估计，即 $\hat{\Gamma}_0$；给出了 r_t 的相关阵估计，即 $\hat{\rho}_0$；随后给出了 $l = 1, 2, ...$ 的 CCM(即 $\hat{\rho}_l$) 的简化表示。两个序列的同步的相关性较强，相关系数为 0.645；IBM 对数收益率的序列自相关（矩阵的左上角元素）都不显著；标普对数收益率的序列自相关（矩阵的右下角元素）在滞后 1,3,5,9 等位置显著；IBM 受标普过去值的影响（矩阵的右上角元素）在滞后 1,2,5 位置显著；标普受 IBM 过去值的影响（矩阵的左下角元素）都不显著。这符合一般的常识。

在上面的四幅小图中，左上图是 IBM 对数收益率的 ACF，右下图是标普 500 对数收益率的 ACF；右上图是 $\rho_{12}(l) = \text{corr}(x_{1t}, x_{2,t-l}), l = 0, 1, 2, ...$ 的图形，即 t 时刻 IBM 对数收益率与滞后的 $t-l$ 时刻的标普对数收益率之间的相关系数图，因为时间的单向性可以看成是标普对 IBM 股票的领先作用。左下图是标普对数收益率与滞后的 IBM 收益率的相关系数图，可以看成是 IBM 股票对标普的领先作用。虽然文本显示的 $\hat{\rho}_l$ 矩阵有正负号，但是从图形来看除了同步情形以外自相关和互相关都比较弱, 同步时的互相关是显著的 (注意右上图和左下图两个突出值都是 $\ell = 0$ 处的表示同步相关的值，等于 0.645)。标普对 IBM 有很弱的领先作用，而 IBM 对标普没有领先作用。

最后的大图是检验每个 ρ_l 为零矩阵的检验 p 值随 l 变化的图形。结果表明 $l = 1, 2, 3, 5$ 时显著不等于零矩阵，其它各滞后值时不显著。

23.2.2 例 1.2：美国国债

考虑各个期限的美国债券指数的月简单收益率，包括 30 年、20 年、10 年、5 年和 1 年期限，数据来自 CRSP 数据库，从 1942 年 1 月值 1999 年 12 月，共 696 个观测值。读入数据后转换为对数收益率，单位为百分之一。

```
da2 <- read_table(
  "m-bnd.txt",
  col_types=cols(.default=col_double()))
ts.mbnd <- ts(log(1 + da2)*100, start=c(1942,1), frequency=12)
```

序列的基本统计：

```
summary(ts.mbnd)
```

```
##      30yrs             20yrs             10yrs              5yrs
##  Min.   :-8.0440   Min.   :-8.7804   Min.   :-6.9050   Min.   :-5.9771
##  1st Qu.:-0.8146   1st Qu.:-0.7186   1st Qu.:-0.5882   1st Qu.:-0.1133
##  Median : 0.2228   Median : 0.2337   Median : 0.2482   Median : 0.2297
##  Mean   : 0.4024   Mean   : 0.4229   Mean   : 0.4284   Mean   : 0.4489
##  3rd Qu.: 1.5366   3rd Qu.: 1.4728   3rd Qu.: 1.2825   3rd Qu.: 0.9425
##  Max.   :12.4993   Max.   :14.1803   Max.   : 9.5301   Max.   :10.0858
##       1yr
##  Min.   :-1.7360
##  1st Qu.: 0.1049
```

```
##  Median : 0.3404
##  Mean   : 0.4408
##  3rd Qu.: 0.6372
##  Max.   : 5.4545
```

5 个序列的时间序列图：

```
#plot(ts.mbnd, plot.type="multiple", nc=1, yax.flip=TRUE, ylim=c(-10, 15))
plot(as.xts(ts.mbnd), type="l", multi.panel=TRUE, theme="white",
     main=" 美国国债月对数收益率（%）",
     major.ticks="years",
     grid.ticks.on = "auto")
```

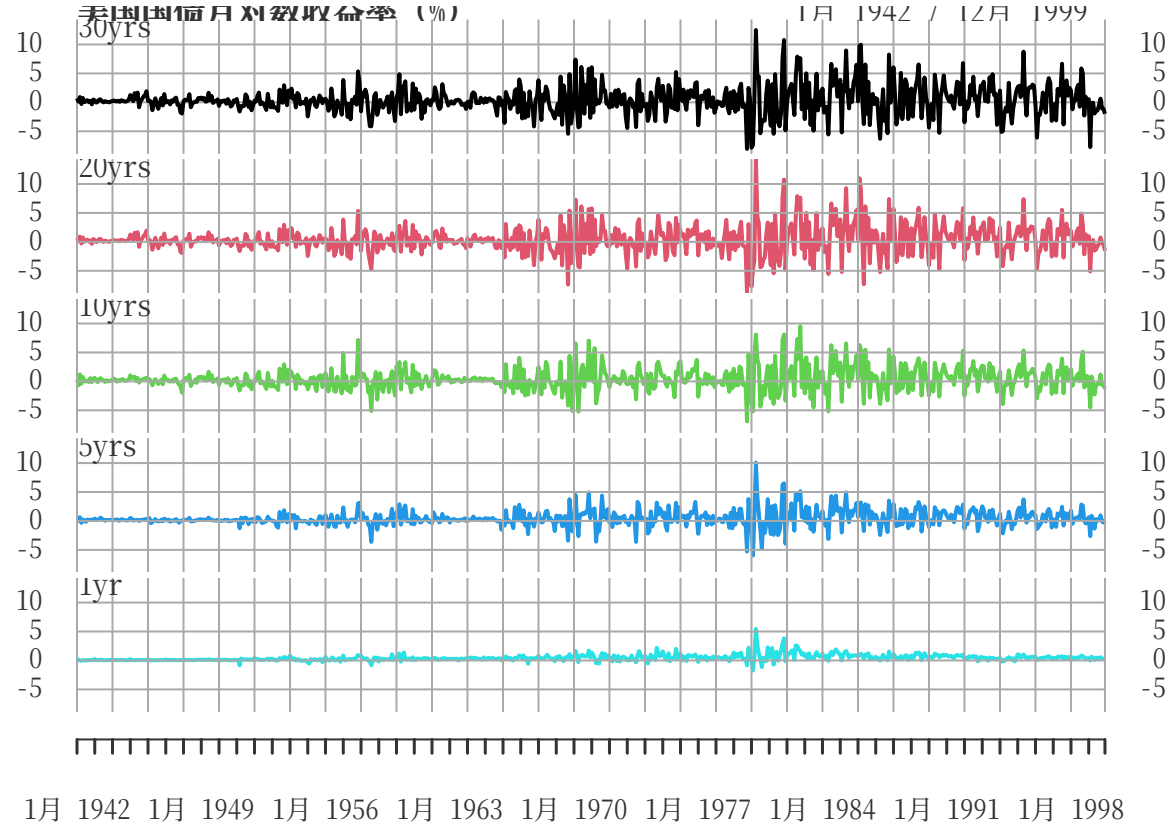


图 23.7: 美国国债月对数收益率（%）

可以看出长期国债的收益率波动较大，短期国债收益率波动较小。

样本均值：

```
colMeans(ts.mbnd)
```

```
##      30yrs      20yrs      10yrs       5yrs        1yr
## 0.4024412 0.4228871 0.4283688 0.4488915 0.4407712
```

折合年利率在 5% 左右。

样本标准差：

```
apply(ts.mbnd, 2, sd)
```

```
##     30yrs     20yrs     10yrs      5yrs       1yr
## 2.5014721 2.3970918 1.9521371 1.3783452 0.5282478
```

标准差折合成年标准差（波动率）为 8.7% 到 1.8% 之间 (月标准差乘以 $\sqrt{12}$)。

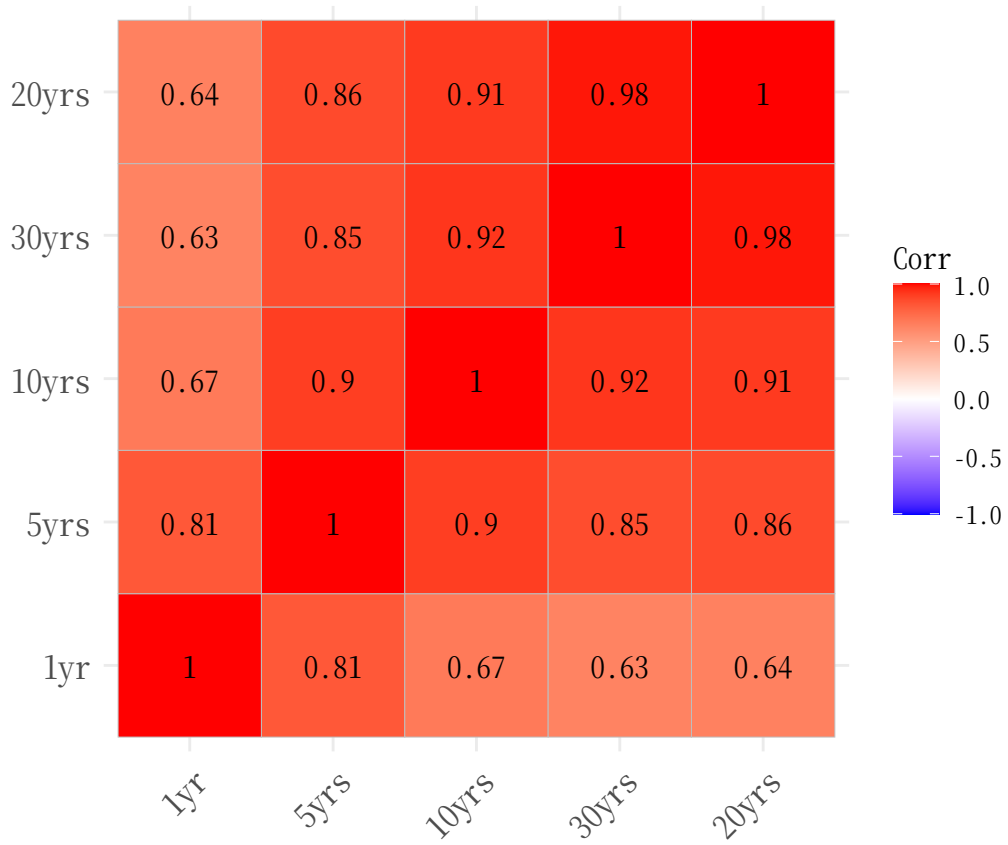
同步相关阵：

```
round(cor(ts.mbnd), 2)
```

```
##       30yrs 20yrs 10yrs 5yrs  1yr
## 30yrs  1.00  0.98  0.92 0.85 0.63
## 20yrs  0.98  1.00  0.91 0.86 0.64
## 10yrs  0.92  0.91  1.00 0.90 0.67
## 5yrs   0.85  0.86  0.90 1.00 0.81
## 1yr    0.63  0.64  0.67 0.81 1.00
```

用 ggcorrplot 作相关系数矩阵图：

```
ggcorrplot::ggcorrplot(cor(ts.mbnd),
  hc.order=TRUE, lab=TRUE)
```



可以看出相关性都很强，而且长期国债之间的相关比短期国债之间的相关要高。

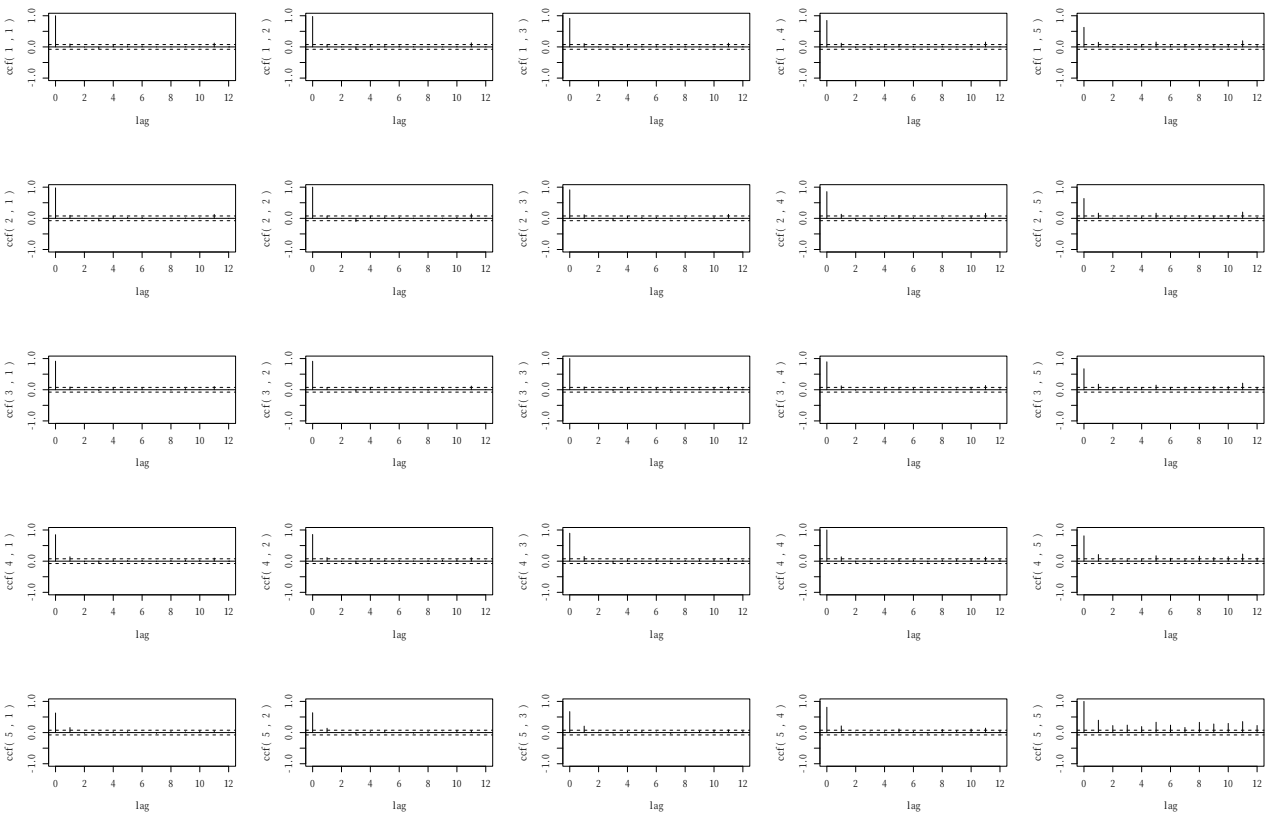
作 CCM 统计表和图形：

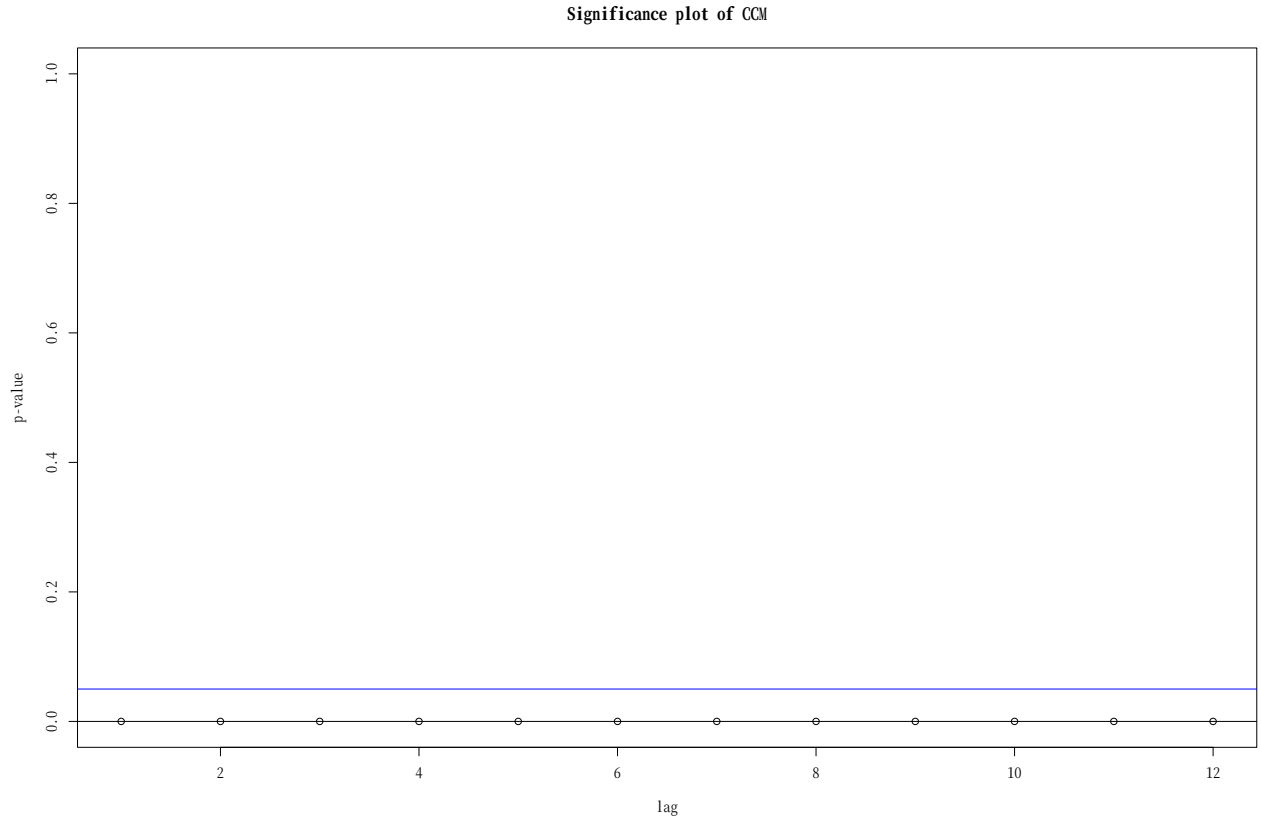
```
ccm(ts.mbnd)
```

```
## [1] "Covariance matrix:"
##       30yrs 20yrs 10yrs  5yrs   1yr
## 30yrs  6.26 5.860 4.476 2.928 0.830
## 20yrs  5.86 5.746 4.279 2.830 0.806
## 10yrs  4.48 4.279 3.811 2.411 0.694
## 5yrs   2.93 2.830 2.411 1.900 0.592
## 1yr    0.83 0.806 0.694 0.592 0.279
## CCM at lag:  0
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] 1.000 0.977 0.917 0.849 0.628
## [2,] 0.977 1.000 0.914 0.857 0.637
## [3,] 0.917 0.914 1.000 0.896 0.673
## [4,] 0.849 0.857 0.896 1.000 0.813
## [5,] 0.628 0.637 0.673 0.813 1.000
## Simplified matrix:
## CCM at lag:  1
## + + + + +
## + . + + +
## + . + + +
## + + + + +
## + + + + +
## CCM at lag:  2
## . . . . .
## . . . . .
## . . . . .
## . . . . .
## . . . . +
## CCM at lag:  3
## . - . . .
## - - . . .
## . . . . .
## . - . . .
## . . . . +
## CCM at lag:  4
## . + . . .
## . . . . .
## . . . . .
## . . . . .
## . . . . +
## CCM at lag:  5
```

```
## . . . + +
## . . . + +
## . . . . +
## . . . . +
## . . . + +
## CCM at lag:  6
## . . . . .
## . . . . .
## . . . . +
## . . . . +
## . . . . +
## CCM at lag:  7
## . . . . .
## . . . . .
## . . . . .
## . . . . .
## . . . . +
## CCM at lag:  8
## . . . . +
## . . . . +
## . . . . +
## . . . . +
## . . . + +
## CCM at lag:  9
## . . . . .
## . . . . .
## . . . . +
## . . . . +
## . . . . +
## CCM at lag:  10
## . . . . .
## . . . . +
## . . . . +
## . . . . +
## . . + + +
## CCM at lag:  11
## + + + + +
## + + + + +
## + + + + +
## + + + + +
## . . + + +
## CCM at lag:  12
## . . . . .
```

```
## . . . . .
## . . . . .
## . . . . +
## . . . . +
```


ccf(1 , 1)
ccf(1 , 2)
ccf(1 , 3)
ccf(1 , 4)
ccf(1 , 5)
ccf(2 , 1)
ccf(2 , 2)
ccf(2 , 3)
ccf(2 , 4)
ccf(2 , 5)
ccf(3 , 1)
ccf(3 , 2)
ccf(3 , 3)
ccf(3 , 4)
ccf(3 , 5)
ccf(4 , 1)
ccf(4 , 2)
ccf(4 , 3)
ccf(4 , 4)
ccf(4 , 5)
ccf(5 , 1)
ccf(5 , 2)
ccf(5 , 3)
ccf(5 , 4)
ccf(5 , 5)
-1.0 0.0 1.0
0 2 4 6 8 10 12
lag



从 CCM 简化矩阵输出看，在滞后 1 时普遍有相互的反馈关系。在滞后 5 时一年期国债对各期限都有领先作用。最后一个 p 值的图是对每一个 $H_0 : \Gamma_l = 0$ 的检验 p 值，横坐标是滞后 l。

23.3 多元混成检验

Hosking(1980,1981), Li 和 McLeod(1981) 已经把一元的 Ljung-Box 白噪声检验推广到了多元的情形。对一个多元序列，检验零假设

$$H_0 : \rho_1 = \cdots = \rho_m = 0$$

对立假设是不全为零矩阵。这可以检验多元时间序列 r_t 为宽白噪声的零假设，即 r_t 为弱平稳列且无序列自相关，可以有同步的分量间相关。

使用检验统计量

$$Q_k(m) = T^2 \sum_{l=1}^{m} \frac{1}{T-l} \mathrm{tr}(\hat{\Gamma}_l^T \hat{\Gamma}_0^{-1} \hat{\Gamma}_l \hat{\Gamma}_0^{-1})$$

其中 k 是分量个数，T 是样本序列长度，tr 表示求矩阵的迹（对角线元素之和）。在若 $\Gamma_l = 0, \forall l > 0$，且 r_t 服从多元正态分布，即 r_t 为独立同多元正态分布，则 $Q_k(m)$ 渐近地服从 $\chi^2(k^2 m)$ 分布，计算右侧 p 值进行检验。事实上，这时 $\hat{\Gamma}_l, l = 1, \ldots, m$ 渐近服从零均值的相互独立的多元正态分布。

利用 Kronecker 积记号 $\otimes$ 可以将 $Q_k(m)$ 用样本互相关阵 $\hat{\rho}_l$ 表示：

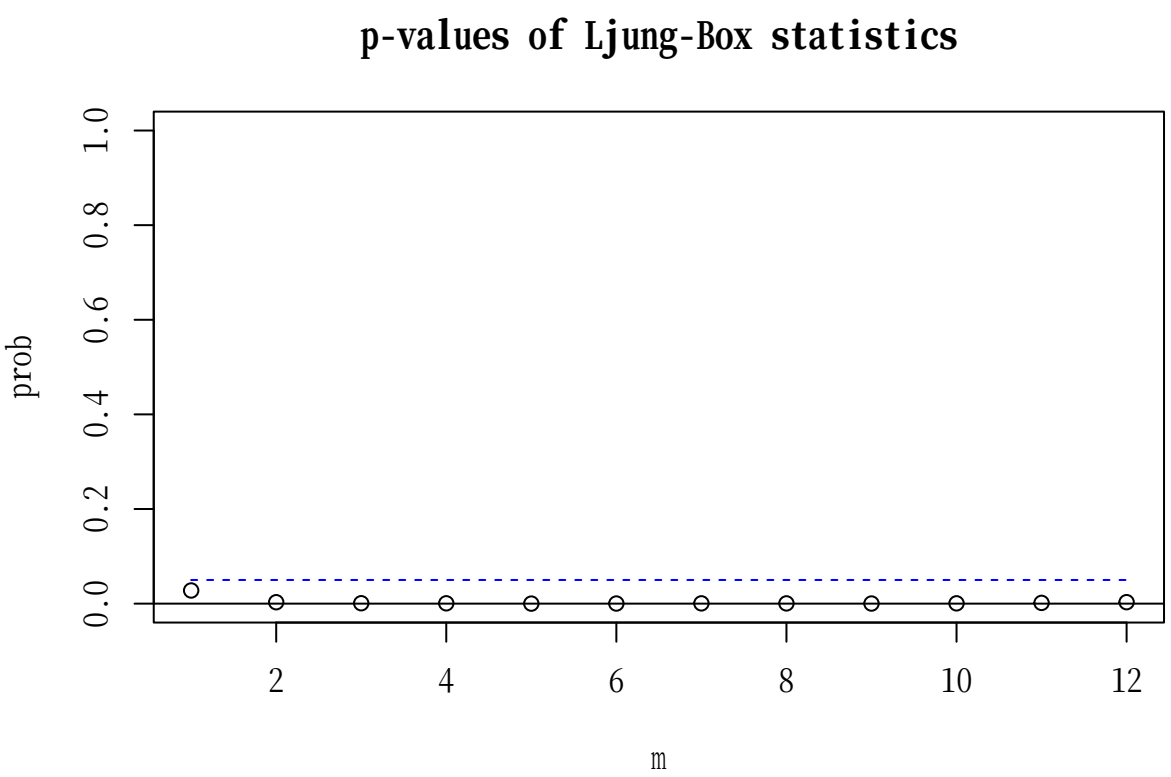
$$Q_k(m) = T^2 \sum_{l=1}^{m} \frac{1}{T-l} b_l^T (\hat{\rho}_0^{-1} \otimes \hat{\rho}_0^{-1}) b_l$$

其中 $b_l = \mathrm{vec}(\hat{\rho}_l^T)$, vec 表示矩阵按列拉直运算。

对 IBM 和标普数据，用 MTS::mq() 计算多元混成检验：

```
MTS::mq(cbind(ts.mibm, ts.msp5), lag=12)
```

```
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
##  [1,]   1.0       10.9     4.0     0.03
##  [2,]   2.0       23.3     8.0     0.00
##  [3,]   3.0       33.5    12.0     0.00
##  [4,]   4.0       40.4    16.0     0.00
##  [5,]   5.0       54.2    20.0     0.00
##  [6,]   6.0       56.3    24.0     0.00
##  [7,]   7.0       59.0    28.0     0.00
##  [8,]   8.0       65.1    32.0     0.00
##  [9,]   9.0       73.8    36.0     0.00
## [10,]  10.0       75.5    40.0     0.00
## [11,]  11.0       77.0    44.0     0.00
## [12,]  12.0       79.2    48.0     0.00
```

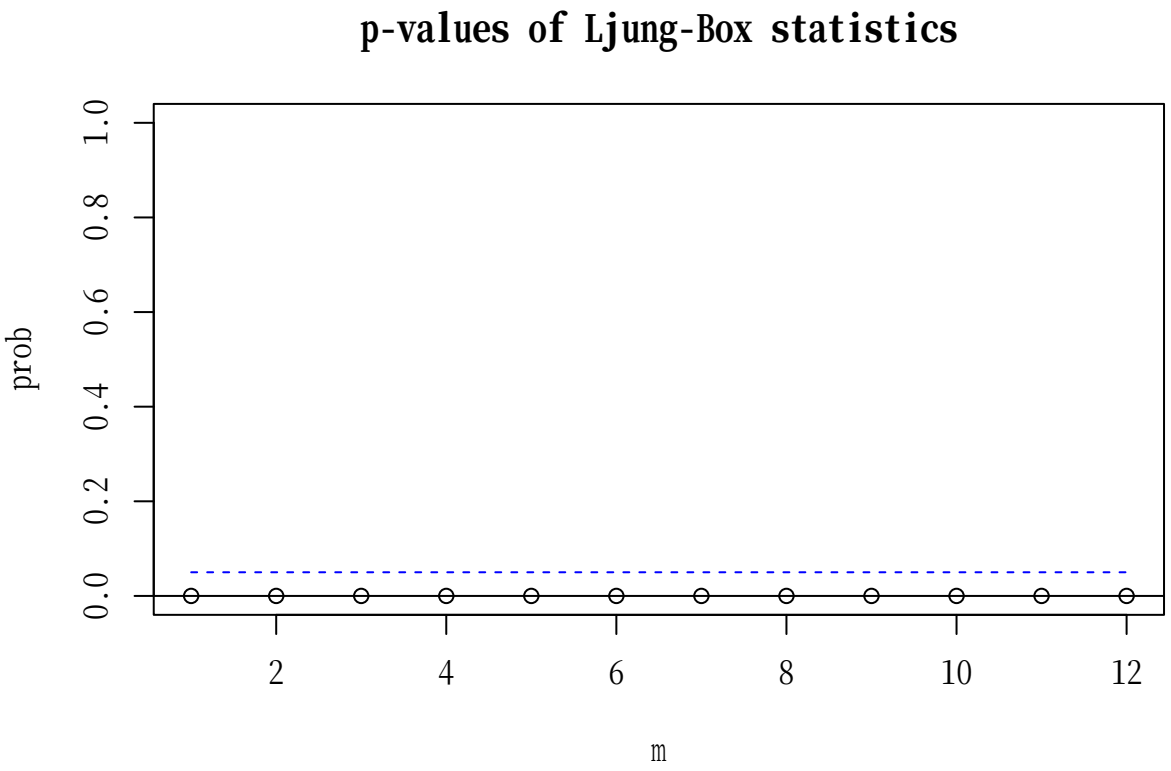


上面的程序对 $m = 1, 2, \ldots, 12$ 分别计算了检验，并做了 p 值随 m 变化的图形。在 0.05 水平下都是显著的。

对国债数据做多元混成检验：

```
MTS::mq(ts.mbnd, lag=12)
```

```
## Ljung-Box Statistics:
##         m       Q(m)     df    p-value
##  [1,]     1        279      25        0
##  [2,]     2        454      50        0
##  [3,]     3        712      75        0
##  [4,]     4        861     100        0
##  [5,]     5       1076     125        0
##  [6,]     6       1268     150        0
##  [7,]     7       1412     175        0
##  [8,]     8       1623     200        0
##  [9,]     9       1835     225        0
## [10,]    10       2009     250        0
## [11,]    11       2213     275        0
## [12,]    12       2366     300        0
```



结果都是显著的。

$Q_k(m)$ 统计量是对 r_t 的前 m 个互相关阵的一个联合检验，如果结果显著，就应该建立多元的均值模型描述序列分量之间的领先–滞后关系。最常用的是向量自回归 (VAR) 模型。

23.4.1 CCM 计算和绘图的函数

23.4 附录：用到的源程序代码和数据

```
"ccm" <- function(x, lags=12, level=FALSE, output=TRUE){
  # Compute and plot the cross-correlation matrices.
  # lags: number of lags used.
  # level: logical unit for printing.
  #
  if(!is.matrix(x)) x = as.matrix(x)
  nT=dim(x)[1]; k=dim(x)[2]
  if(output){
    opar <- par(mfcol=c(k,k))
    on.exit(par(opar))
  }
  if(lags < 1)lags=1

  # remove the sample means
  y = scale(x,center=TRUE,scale=FALSE)

  V1 = cov(y)
  if(output){
    print("Covariance matrix:")
    print(V1,digits=3)
  }

  se = sqrt(diag(V1))
  SD = diag(1/se)
  S0 = SD %*% V1 %*% SD
  ## S0 used later
  ksq = k*k
  wk = matrix(0, ksq, lags+1)
  wk[,1] = c(S0)
  j=0
  if(output){
    cat("CCM at lag: ",j,"\n")
    print(S0, digits=3)
    cat("Simplified matrix:","\n")
  }

  y = y %*% SD
  crit = 2.0/sqrt(nT)
```

```
for (j in 1:lags){
  y1 = y[1:(nT-j),]
  y2 = y[(j+1):nT,]
  Sj = t(y2) %*% y1 / nT
  Smtx = matrix(".", k, k)
  for (ii in 1:k){
    for (jj in 1:k){
      if(Sj[ii,jj] > crit) Smtx[ii,jj] = "+"
      if(Sj[ii,jj] < -crit) Smtx[ii,jj] = "-"
    }
  }
  if(output){
    cat("CCM at lag: ", j, "\n")
    for (ii in 1:k){
      cat(Smtx[ii,],"\n")
    }
    if(level){
      cat("Correlations:","\n")
      print(Sj,digits=3)
    }
  } ## end of if-(output) statement
  wk[, j+1] = c(Sj)
}

if(output){
  iik <- rep(1:k, k)
  jjk <- rep(1:k, each=k)
  tdx=c(0,1:lags)
  jcnt=0
  if(k > 10){
    print("Skip the plots due to high dimension!")
  } else {
    for (j in 1:ksq){
      plot(tdx, wk[j,], type='h',
           xlab='lag',
           ylab=paste('ccf(', iik[j], ",", jjk[j], ")"),
           ylim=c(-1,1))
      abline(h=c(0))
      crit=2/sqrt(nT)
      abline(h=c(crit),lty=2)
      abline(h=c(-crit),lty=2)
      jcnt=jcnt+1
    }
```

```
    }
    ## end of if-(output) statement
  }

  ## The following p-value plot was added on May 16, 2012 by Ruey Tsay.
  ### Obtain a p-value plot of ccm matrix
  r0i=solve(S0)
  R0=kronecker(r0i,r0i)
  pv=rep(0,lags)
  for (i in 1:lags){
    tmp=matrix(wk[,(i+1)],ksq,1)
    tmp1=R0%*%tmp
    ci=crossprod(tmp,tmp1)*nT*nT/(nT-i)
    pv[i]=1-pchisq(ci,ksq)
  }
  if(output){
    par(opar)
    plot(pv,xlab='lag',ylab='p-value',ylim=c(0,1))
    abline(h=c(0))
    abline(h=c(0.05),col="blue")
    title(main="Significance plot of CCM")
  }
  ccm <- list(ccm=wk,pvalue=pv)
}
```

Chapter 24

向量自回归模型

24.1 VAR(1) 模型

多个资产收益率的联合模型中最常用的是向量自回归 (Vector Autoregression, VAR) 模型。称 k 元时间序列 r_t 服从一个一阶向量自回归模型，即 VAR(1)，若有

$$r_t = \phi_0 + \Phi r_{t-1} + a_t \tag{24.1}$$

其中 ϕ_0 是 k 维常数向量，Φ 是 k 阶常数方阵，$\{a_t\}$ 是序列不相关的弱平稳列，$Ea_t = 0$, $\mathrm{Var}(a_t) = \Sigma > 0$(对称正定矩阵)。

文献中经常假定 $\{a_t\}$ 服从多元正态分布 $N_k(0, \Sigma)$，则这时 a_t 是独立同分布随机向量序列。

记 $a_t = (a_{1t}, \dots, a_{kt})^T$，$\phi_0 = (\phi_{10}, \dots, \phi_{k0})^T$, $\Phi = (\phi_{ij})_{k\times k}$。

24.2 模型结构和格兰杰因果性

考虑 $k = 2$ 的情形。模型变成

$$\begin{cases} r_{1t} = \phi_{10} + \phi_{11} r_{1,t-1} + \phi_{12} r_{2,t-1} + a_{1t} \\ r_{2t} = \phi_{20} + \phi_{21} r_{1,t-1} + \phi_{22} r_{2,t-1} + a_{2t} \end{cases} \tag{24.2}$$

如果 $\phi_{12} = \phi_{21} = 0$, 则 r_{1t} 和 r_{2t} 两个序列是分离的：

$$\begin{aligned} r_{1t} =& \phi_{10} + \phi_{11} r_{1,t-1} + a_{1t} \\ r_{2t} =& \phi_{20} + \phi_{22} r_{2,t-1} + a_{2t} \end{aligned}$$

这时两个序列分别服从单独的一元 AR(1) 模型；如果 a_{1t} 和 a_{2t} 不相关，则 $\{r_{1t}\}$ 序列与 $\{r_{2s}\}$ 序列不相关。称这样的分离的序列为**非耦合**的。

如果 $\phi_{12} \neq 0$, $\phi_{21} \neq 0$, 这时两个序列之间有相互反馈关系。

如果 $\phi_{12} = 0$, $\phi_{21} \neq 0$, 模型变成

$$\begin{cases} r_{1t} = \phi_{10} + \phi_{11} r_{1,t-1} + a_{1t} \\ r_{2t} = \phi_{20} + \phi_{21} r_{1,t-1} + \phi_{22} r_{2,t-1} + a_{2t} \end{cases} \tag{24.3}$$

这时 r_{1t} 不受到 r_{2t} 的过去值的影响，但 r_{2t} 受到 $r_{1,t-1}$ 的过去值的影响。可以将 r_{1t} 作为输入变量，r_{2t} 作为输出变量。

在统计学文献中，如果 a_{1t} 和 a_{2t} 也不相关，称式(24.3)的 r_{1t} 和 r_{2t} 之间有传递函数 (transfer function) 关系。传递函数模型是 VARMA 的一种特殊形式，在控制工程中非常有用，可以通过调整 r_{1t} 的值来影响 r_{2t}。在计量经济学文献中，此模型意味着两个序列之间存在格兰杰因果关系，r_{1t} 的过去值影响 r_{2t}，但 r_{2t} 的过去值不影响 r_{1t}。

(C. W. J. Granger, 1969) 提出了因果关系的概念，适用于解释 VAR 模型的结果。考虑一个二元序列 $r_t = (r_{1t}, r_{2t})^T$ 的超前 l 步预测问题。可以分别用 VAR 模型和每个分量的一元模型来进行预测。如果 r_{2t} 的二元预测比它的一元预测更准确，则称 r_{1t} 是 r_{2t} 的格兰格原因，这是因为 r_{1t} 的信息提高了 r_{2t} 的预测精度。这里预测精度用预测的均方误差来度量。当然，r_{2t} 也可以同时是 r_{1t} 的格兰格原因。

令 F_t 表示截止到时刻 t 的可用信息，即 F_t 包含 $\{r_t, r_{t-1}, r_{t-2}, ...\}$。令 $F_{-i,t}$ 表示 F_t 中去掉第 i 个分量的信息的集合。考虑(24.2)的二元 VAR(1) 模型。F_t 包含 $\{r_{1t}, r_{2t}, r_{1,t-1}, r_{2,t-1}, ...\}$, $F_{-1,t}$ 包含 $\{r_{2t}, r_{2,t-1}, ...\}$。考虑基于 F_t 的超前 l 步预测，其预测误差为

$$e_t(l) = (e_{1t}(l), e_{2t}(l))^T = r_{t+l} - E(r_{t+l}|F_t)$$

其中对 $r_{2,t+l}$ 的预测误差为

$$e_{2t}(l) = r_{2,t+l} - E(r_{2,t+l}|F_t)$$

考虑基于 $F_{-1,t}$ 对 $r_{2,t+l}$ 的预测，其预测误差为

$$\tilde{e}_{2t}(l) = r_{2,t+l} - E(r_{2,t+l}|F_{-1,t}) = r_{2,t+l} - E(r_{2,t+l}|r_{2,t}, r_{2,t-1}, ...)$$

如果

$$Ee_{2t}^2(l) = \mathrm{Var}(r_{2,t+l}|F_t) < E\tilde{e}_{2t}^2(l) = \mathrm{Var}(r_{2,t+l}|F_{-1,t})$$

即使用截止到 t 时刻为止的全部信息预测 $r_{2,t+l}$，均方误差比不利用第一个分量序列的均方误差小，则称 r_{1t} 是 r_{2t} 的格兰格原因。

回到二元 VAR(1) 模型(24.2)，如果 $\phi_{12} = 0$ 而 $\phi_{21} \neq 0$，则 $r_{2,t+1}$ 依赖于 $r_{1,t}$，预测 $r_{2,t+1}$ 时需要利用 r_{1t} 的信息，序列 r_{1t} 是序列 r_{2t} 的格兰格原因；$r_{1,t+1}$ 不依赖于 $r_{2,t}$，预测 $r_{1,t+1}$ 就不需要用到 r_{2t}，所以序列 r_{2t} 不是序列 r_{1t} 的格兰格原因。

事实上，对(24.3)，不妨设 $\{a_t\}$ 是独立的多元弱平稳列，这时 $E(a_{t+1}|F_t) = 0$, $E(a_{2,t+1}|F_{-1,t}) = 0$, $\mathrm{Var}(a_{2,t+1}|F_t) = \mathrm{Var}(a_{2,t+1}|F_{-1,t}) = \sigma_{22}$，又

$$\begin{aligned} E(r_{2,t+1}|F_t) =& \phi_{20} + \phi_{21}r_{1,t} + \phi_{22}r_{2,t} \\ \mathrm{Var}(r_{2,t+1}|F_t) =& E[(r_{2,t+1} - E(r_{2,t+1}|F_t))^2|F_t] \\ =& E(a_{2,t+1}^2|F_t) = \sigma_{22} \end{aligned}$$

而

$$\begin{aligned} E(r_{2,t+1}|F_{-1,t}) =& \phi_{20} + \phi_{21}E(r_{1,t}|F_{-1,t}) + \phi_{22}r_{2,t} \\ \mathrm{Var}(r_{2,t+1}|F_{-1,t}) =& E[(r_{2,t+1} - E(r_{2,t+1}|F_{-1,t}))^2|F_{-1,t}] \\ =& E[\left\{\phi_{21}(r_{1,t} - E(r_{1,t}|F_{-1,t})) + a_{2,t+1}\right\}^2|F_{-1,t}] \\ =& \phi_{21}^2\mathrm{Var}(r_{1,t}|F_{-1,t}) + \sigma_{22} > \sigma_{22} \end{aligned}$$

即 r_{1t} 是 r_{2t} 的格兰格原因。

另一方面，

$$\begin{aligned}
E(r_{1,t+1}|F_t) =& \phi_{10} + \phi_{11} r_{1,t} \\
\mathrm{Var}(r_{1,t+1}|F_t) =& E[(r_{1,t+1} - E(r_{1,t+1}|F_t))^2|F_t] \\
=& E(a_{1,t+1}^2|F_t) = \mathrm{Var}(a_{1,t+1}) = \sigma_{11} \\
E(r_{1,t+1}|F_{-2,t}) =& \phi_{10} + \phi_{11} r_{1,t} \\
\mathrm{Var}(r_{1,t+1}|F_{-2,t}) =& E[(r_{1,t+1} - E(r_{1,t+1}|F_{-2,t}))^2|F_{-2,t}] \\
=& E(a_{1,t+1}^2|F_{-2,t}) = \mathrm{Var}(a_{1,t+1}) = \sigma_{11}
\end{aligned}$$

所以 r_{2t} 不是 r_{1t} 的格兰格原因。

如果 $\phi_{21} = 0$ 而 $\phi_{12} \neq 0$，则序列 r_{2t} 是序列 r_{1t} 的格兰格原因，而序列 r_{1t} 不是序列 r_{2t} 的格兰格原因。

如果新息 $a_t = (a_{1t}, a_{2t})^T$ 的协方差阵 Σ 不是对角阵，则 r_{1t} 和 r_{2t} 之间存在同步相关性，这时两个序列存在即期（同期，瞬时）格兰格因果关系，这种即期因果关系是双向的。

注意，这样定义的格兰格因果性依赖于 VAR(1) 模型(24.1)这样的模型形式（称为简化形式），如果改变模型的形式（比如改为下面的结构形式），可能就不会在 ϕ_{12} 和 ϕ_{21} 这样的简单数值上反映出格兰格因果性。

更多分量的 VAR(1) 可以有更灵活的格兰格因果关系。比如三元的模型，如果 Φ 是下三角阵:

$$\begin{pmatrix} \phi_{11} & & \\ \phi_{21} & \phi_{22} & \\ \phi_{31} & \phi_{32} & \phi_{33} \end{pmatrix}$$

则：

- r_{2t} 和 r_{3t} 都不是 r_{1t} 的格兰格原因；
- r_{1t} 是 r_{2t} 的格兰格原因，但 r_{3t} 不是 r_{2t} 的格兰格原因；
- r_{1t} 和 r_{2t} 都是 r_{3t} 的格兰格原因。

如果 Φ 有其他稀疏形式，关系可能多种多样。

24.3 VAR 的简化形式和结构形式

式(24.1)的系数矩阵 Φ 度量了 r_t 的动态相依性，r_{1t} 与 r_{2t} 之间的同步相依性可以通过 a_t 的协方差阵 Σ 的非对角元素 σ_{12} 来反映。如果 $\sigma_{12} = 0$，则两个分量 r_{1t} 和 r_{2t} 之间没有同步线性关系。在计量经济文献中式(24.1)中的 VAR(1) 模型称为**简化形式** (reduced form) 的模型，因为该模型没有清楚地表现出分量序列之间的同步线性相依性。

可以利用矩阵变换对(24.1)进行变换，得到显式地表现同步关系的模型。对 $\Sigma > 0$(正定)，存在 Cholesky 分解 $\Sigma = LGL^T$, 其中 G 是对角线元素都为正数的 k 阶对角方阵，L 为对角线元素都等于 1 的下三角阵，定义

$$b_t = L^{-1} a_t = (b_{1t}, \ldots, b_{kt})^T$$

则

$$\begin{aligned}
Eb_t =& 0 \\
\mathrm{Var}(b_t) =& L^{-1} \mathrm{Var}(a_t) L^{-T} = G
\end{aligned}$$

其中 $L^{-T} = (L^T)^{-1} = (L^{-1})^T$。上式表明 b_t 的各分量不相关。将式(24.1)两边同时左乘 L^{-1}，得

$$\begin{aligned}
L^{-1} r_t =& L^{-1}\phi_0 + L^{-1}\Phi r_{t-1} + L^{-1} a_t && (24.4) \\
=& \phi_0^* + \Phi^* r_{t-1} + b_t && (24.5)
\end{aligned}$$

记 $\phi_0^* = (\phi_{10}^*, \dots, \phi_{k0}^*)^T$, $\Phi^* = (\phi_{ij}^*)_{k\times k}$，设 L^{-1} 的最后一行为 $(w_{k1}, \dots, w_{k,k-1}, 1)$，则模型(24.5)的最后一个方程（第 k 个方程）为

$$r_{kt} + \sum_{i=1}^{k-1} w_{ki} r_{it} = \phi_{k0}^* + \sum_{i=1}^{k} \phi_{ki}^* r_{i,t-1} + b_{kt} \tag{24.6}$$

对 $i < k$，因为 b_{kt} 与 b_{ki} 不相关，所以式(24.6)明确表示了 r_{kt} 对 r_{it} 的同步依赖性。在计量经济文献中式(24.6)称为 r_{kt} 的一个**结构方程** (structural form)。

对 r_t 的任何其它分量 $r_{jt}(j < k)$，可以对 VAR 模型各个方程的次序进行重排，比如，将第 j 个方程与第 k 个方程对换位置，再用前面的方法得到关于 r_{jt} 的结构方程。因此，式(24.1)的简化形式等价于式(24.6)的结构形式。

在时间序列分析中通常使用简化形式，因为

- 简化形式更容易估计；
- 在预测时，同步形式无法使用。

例 2.1

为了说明从简化形式到结构方程的变换，考虑如下的二元 VAR(1) 模型：

$$\begin{pmatrix} r_{1t} \\ r_{2t} \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.4 \end{pmatrix} + \begin{pmatrix} 0.2 & 0.3 \\ -0.6 & 1.1 \end{pmatrix} \begin{pmatrix} r_{1,t-1} \\ r_{2,t-1} \end{pmatrix} + \begin{pmatrix} a_{1t} \\ a_{2t} \end{pmatrix}$$

$$\mathrm{Var}(a_t) = \Sigma = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

对 Σ 作 Cholesky 分解得 $\Sigma = LGL^T$, 其中

$$G = \begin{pmatrix} 2 & 0 \\ 0 & 0.5 \end{pmatrix} \quad L^{-1} = \begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix}$$

在简化形式的 VAR(1) 模型两边同时左乘 L^{-1}，得

$$\begin{pmatrix} 1 & 0 \\ -0.5 & 1 \end{pmatrix} \begin{pmatrix} r_{1t} \\ r_{2t} \end{pmatrix} = \begin{pmatrix} 0.2 \\ 0.3 \end{pmatrix} + \begin{pmatrix} 0.2 & 0.3 \\ -0.7 & 0.95 \end{pmatrix} \begin{pmatrix} r_{1,t-1} \\ r_{2,t-1} \end{pmatrix} + \begin{pmatrix} b_{1t} \\ b_{2t} \end{pmatrix}$$

$$\mathrm{Var}(b_t) = G = \begin{pmatrix} 2 & 0 \\ 0 & 0.5 \end{pmatrix}$$

其中第二个方程为

$$r_{2t} = 0.3 + 0.5 r_{1t} - 0.7 r_{1,t-1} + 0.95 r_{2,t-1} + b_{2t}$$

此方程明确表现了 r_{2t} 对 r_{1t} 的同步依赖（系数 +0.5），方程中也有 r_{2t} 对 $r_{1,t-1}$ 的交叉依赖，对 $r_{2,t-1}$ 的自身依赖。

为了给出 r_{1t} 的结构方程，将简化形式的两个方程对换位置，得（注意 Φ 需要行互换、列互换）

$$\begin{pmatrix} r_{2t} \\ r_{1t} \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.2 \end{pmatrix} + \begin{pmatrix} 1.1 & -0.6 \\ 0.3 & 0.2 \end{pmatrix} \begin{pmatrix} r_{2,t-1} \\ r_{1,t-1} \end{pmatrix} + \begin{pmatrix} a_{2t} \\ a_{1t} \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

此 Σ 的 Cholesky 分解有

$$G = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad L^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

在调换次序的 VAR(1) 模型的两边同时左乘 L^{-1}，得

$$\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} r_{2t} \\ r_{1t} \end{pmatrix} = \begin{pmatrix} 0.4 \\ -0.2 \end{pmatrix} + \begin{pmatrix} 1.1 & -0.6 \\ -0.8 & 0.8 \end{pmatrix}\begin{pmatrix} r_{2,t-1} \\ r_{1,t-1} \end{pmatrix} + \begin{pmatrix} c_{1t} \\ c_{2t} \end{pmatrix}$$

$$\mathrm{Var}(c_t) = G = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

其中第二个方程为

$$r_{1t} = -0.2 + 1.0r_{2t} - 0.8r_{2,t-1} + 0.8r_{1,t-1} + c_{2t}$$

这个方程明确地表现了 r_{1t} 对 r_{2t} 的同步线性依赖。

利用 R 计算 LGL^T 分解的 L^{-1} 和 G 的程序：

```
ldl.decomp <- function(A){
  R <- chol(A)
  D <- diag(diag(R))
  Linv <- solve(t(solve(D, R)))
  G <- D^2

  list(Linv=Linv, G=G)
}
```

例如

```
Sigma <- rbind(c(2,1), c(1,1)); Sigma
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    1    1
```

```
ldl.decomp(Sigma)
```

```
## $Linv
##      [,1] [,2]
## [1,]  1.0    0
## [2,] -0.5    1
##
## $G
##      [,1] [,2]
## [1,]    2  0.0
## [2,]    0  0.5
```

```
Sigma <- rbind(c(1,1), c(1,2)); Sigma
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    1    2
```

```
ldl.decomp(Sigma)
```

```
## $Linv
##      [,1] [,2]
## [1,]    1    0
## [2,]   -1    1
##
## $G
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

24.4 VAR(1) 模型的平稳性条件和矩

设(24.1)式中的 VAR(1) 模型的 r_t 是弱平稳的，在方程两边取期望得

$$Er_t = \phi_0 + \Phi Er_{t-1}$$

由 $Er_t = Er_{t-1}$ 得 $I - \Phi$ 满秩（也叫做非奇异）时

$$\mu = Er_t = (I - \Phi)^{-1}\phi_0$$

在式(24.1)中代入 $\phi_0 = (I - \Phi)\mu$, 则模型(24.1)可以写成

$$r_t - \mu = \Phi(r_{t-1} - \mu) + a_t$$

记中心化的 r_t 为 $\tilde{r}_t = r_t - \mu$，则 VAR(1) 模型(24.1)变成

$$\tilde{r}_t = \Phi\tilde{r}_{t-1} + a_t \tag{24.7}$$

将(24.7)递归得

$$\tilde{r}_t = a_t + \Phi a_{t-1} + \Phi^2 a_{t-2} + \cdots = \sum_{j=0}^{\infty} \Phi^j a_{t-j} \tag{24.8}$$

其中 $\Phi^0 = I_k$。这相当于一元时间序列的 Wold 表示。

由(24.8)可以推出：

- 因为 $\{a_t\}$ 序列不相关，所以 $\mathrm{Cov}(a_t, r_{t-l}) = 0 (l > 0$ 时)。因此可以将 a_t 称为序列 $\{r_t\}$ 在 t 时刻的一个扰动或者新息（innovation）。一般定义新息为 $a_t = r_t - E(r_t|F_{t-1})$ 或 $a_t = r_t - L(r_t|H_{t-1})$，$L(\cdot|\cdot)$ 表示均方误差最小线性预测，H_{t-1} 表示由 $\{r_{t-1}, r_{t-2}, \dots\}$ 张成的 Hilbert 空间。平稳的 VAR(1) 中 a_{it} 是用 $r_{t-1}, r_{t-2}, \dots$ 对 r_{it} 作最优线性预测的预测误差。
- 在(24.8)两边同时右乘 a_t^T 后取期望，得 $\mathrm{Cov}(r_t, a_t) = \Sigma$。
- 对 VAR(1) 模型(24.1)导出的 Wold 表示(24.8)，r_t 按系数矩阵 Φ^j 依赖于过去的新息 a_{t-j}。为了使得(24.8)收敛，需要 $j \to \infty$ 时 $\Phi^j \to 0$，这当且仅当 Φ 的所有特征值的模都小于 1。如果存在模大于或等于 1 的特征值，则 $j \to \infty$ 时 Φ^j 发散或者不收敛到 0 矩阵。仿照一元情形的特征多项式的讨论，因为特征值的模小于 1 当且仅当 $|\lambda I - \Phi|$ 的根的模都小于 1，而 $|\lambda I - \Phi| = \lambda^k |I - \Phi\lambda^{-1}|$，记 $z = \lambda^{-1}$，称关于 z 的多项式 $|P(z)| = |I - \Phi z|$

为模型的 (逆序) **特征多项式**，则式(24.8)收敛的充分必要条件是特征多项式 $|P(z)|$ 的根都在单位圆外。这个条件称为 VAR(1) 模型的**平稳性条件**。当且仅当平稳性条件成立时，模型(24.1)存在满足 a_t 与 $r_{t-1}, r_{t-2}, \ldots$ 不相关条件的平稳解。

- 利用推移算子和特征多项式可以将模型(24.1)写成 $P(B)r_t = \phi_0 + a_t$。
- 利用 Wold 表示(24.8)式还可得

$$\mathrm{Var}(r_t) = \Sigma + \Phi\Sigma\Phi^T + \Phi^2\Sigma(\Phi^2)^T + \cdots = \sum_{j=0}^{\infty} \Phi^j \Sigma (\Phi^j)^T$$

考虑 VAR(1) 模型的互协方差阵和互相关阵。将式(24.7)两边乘以 $\tilde{r}_{t-l}^T$ 后取期望，得互协方差阵

$$\begin{aligned}
\Gamma_l =& E(\tilde{r}_t \tilde{r}_{t-l}^T) = \mathrm{Cov}(r_t, r_{t-l}) && (24.9)\\
=& \Phi E(\tilde{r}_{t-1}\tilde{r}_{t-l}^T) + E(a_t \tilde{r}_{t-l}) && (24.10)\\
=& \Phi\Gamma_{l-1}, \quad (l > 0) && (24.11)
\end{aligned}$$

这个结果是一元时间序列 AR(1) 模型性质的推广。

通过递推可得

$$\Gamma_l = \Phi^l \Gamma_0, \quad l > 0$$

其中 $\Gamma_0 = \mathrm{Var}(r_t)$。

设 D 为 Γ_0 的对角阵，则 VAR(1) 模型的互相关阵为

$$\begin{aligned}
\rho_l =& D^{-\frac{1}{2}} \Gamma_l D^{-\frac{1}{2}} = D^{-\frac{1}{2}} \Phi \Gamma_{l-1} D^{-\frac{1}{2}} \\
=& D^{-\frac{1}{2}} \Phi D^{\frac{1}{2}} D^{-\frac{1}{2}} \Gamma_{l-1} D^{-\frac{1}{2}} \\
=& D^{-\frac{1}{2}} \Phi D^{\frac{1}{2}} \rho_{l-1} = \Upsilon \rho_{l-1}
\end{aligned}$$

其中 $\Upsilon = D^{-\frac{1}{2}} \Phi D^{\frac{1}{2}}$ (Υ 读作 Upsilon)。递推得

$$\rho_l = \Upsilon^l \rho_0, \quad l = 1, 2, \ldots$$

例 2.2

考虑二元 VAR(1) 模型

$$\begin{pmatrix} r_{1t} \\ r_{2t} \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \end{pmatrix} + \begin{pmatrix} 0.2 & 0.3 \\ -0.6 & 1.1 \end{pmatrix} \begin{pmatrix} r_{1,t-1} \\ r_{2,t-1} \end{pmatrix} + \begin{pmatrix} a_{1t} \\ a_{2t} \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 2 \end{pmatrix}$$

计算 Φ 的特征值的绝对值：

```
abs(eigen(rbind(c(0.2, 0.3), c(-0.6, 1.1)))$value)
```

```
## [1] 0.8 0.5
```

特征值的绝对值都小于 1。模型是平稳的。

注意 $\phi_{22} = 1.1 > 1$，但不影响平稳性。多元弱平稳一定也是一元弱平稳的。

24.5 VAR(1) 模型的边缘模型

对 VAR(1) 模型(24.1)的如上的 Wold 表示，同时也将 r_t 的每个分量，表示成了一元的无穷阶 MA，但其中的新息是多元的。利用这种思想，将分量的一元模型可以写成一元 ARMA 模型。

设矩阵 A 是可逆方阵，令 $C=(c_{ij})$，其中 c_{ij} 是 A 删除第 i 行和第 j 列后的子矩阵的行列式乘以 $(-1)^{i+j}$，c_{ij} 称为 A 的 (i,j) 元素的代数余子式，C^T 称为 A 的伴随矩阵，记为 $\mathrm{adj}(A)$，有

$$A^{-1}=\frac{1}{|A|}\mathrm{adj}(A),\quad A\cdot\mathrm{adj}(A)=\mathrm{adj}(A)\cdot A=|A|\cdot I$$

VAR(1) 的 Wold 表示相当于

$$r_t=(I-\Phi B)^{-1}\cdot(\phi_0+a_t)$$

利用伴随矩阵可得

$$|I-\Phi B|r_t=\mathrm{adj}(I-\Phi B)\cdot(\phi_0+a_t)$$

其中 $|I-\Phi z|$ 是 z 的 k 阶多项式；$\mathrm{adj}(I-\Phi z)$ 的每个元素是 z 的 $k-1$ 阶多项式。这样，分量 r_{it} 服从一个 ARMA(k,q) 模型，模型中的 MA 部分涉及到多元的新息，但是可以设法转换成一元的新息。

例 2.3

考虑例 2.2 中的模型。

$$\begin{pmatrix} r_{1t}\\ r_{2t}\end{pmatrix}=\begin{pmatrix}5\\3\end{pmatrix}+\begin{pmatrix}0.2 & 0.3\\ -0.6 & 1.1\end{pmatrix}\begin{pmatrix} r_{1,t-1}\\ r_{2,t-1}\end{pmatrix}+\begin{pmatrix} a_{1t}\\ a_{2t}\end{pmatrix}$$

$$\Sigma=\begin{pmatrix}1 & 0.8\\ 0.8 & 2\end{pmatrix}$$

这时

$$\Phi=\begin{pmatrix}0.2 & 0.3\\ -0.6 & 1.1\end{pmatrix}$$

$$I-\Phi z=\begin{pmatrix}1-0.2z & -0.3z\\ 0.6z & 1-1.1z\end{pmatrix}$$

$$\mathrm{adj}(I-\Phi z)=\begin{pmatrix}1-1.1z & 0.3z\\ -0.6z & 1-0.2z\end{pmatrix}$$

$$|I-\Phi z|=1-1.3z+0.4z^2$$

于是

$$(1-1.3B+0.4B^2)\begin{pmatrix} r_{1t}\\ r_{2t}\end{pmatrix}$$

$$=\begin{pmatrix}1-1.1 & 0.3\\ -0.6 & 1-0.2\end{pmatrix}\begin{pmatrix}5\\3\end{pmatrix}+\begin{pmatrix}1-1.1B & 0.3B\\ -0.6B & 1-0.2B\end{pmatrix}\begin{pmatrix} a_{1t}\\ a_{2t}\end{pmatrix}$$

所以分量 r_{1t} 的一元模型为

$$r_{1t}-1.3r_{1,t-1}+0.4r_{1,t-2}=5.4+a_{1t}+0.3a_{2,t-1}-1.1a_{1,t-1}$$

令 $\eta_t=a_{1t}+0.3a_{2,t-1}-1.1a_{1,t-1}$，易见 $\{\eta_t\}$ 是弱平稳列，且其 ACF 在 $q=1$ 之后截尾（即 $\rho_\eta(j)=0,j>1$）。所以 $\{\eta_t\}$ 是 MA(1) 序列，从而分量 r_{1t} 服从一个一元的 ARMA(2,1) 模型。

一般地，k 元的 VAR(1) 模型的分量服从一元的 ARMA$(k,k-1)$ 模型。由于方程左右可能有公共根，所以模型阶数可能会降低。

24.6 VAR(p) 模型

称 k 元时间序列 r_t 服从一个 VAR(p) 模型，如果

$$r_t = \phi_0 + \Phi_1 r_{t-1} + \cdots + \Phi_p r_{t-p} + a_t \tag{24.12}$$

其中 ϕ_0 和 $\{a_t\}$ 同 VAR(1) 的规定，Φ_ℓ 是 k 阶方阵（$\ell = 1, 2, \ldots, p$），其 (i, j) 元素记为 $\phi_{ij}(\ell)$。利用向后推移算子（滞后算子）B 可以将模型写成

$$(I - \Phi_1 B - \cdots - \Phi_p B^p) r_t = \phi_0 + a_t$$

记

$$P(z) = I - \Phi_1 z - \cdots - \Phi_p z^p \tag{24.13}$$

这是一个从复数 z 到 k 阶方阵 $P(z)$ 的变换，$P(z)$ 的每个元素是关于 z 的阶数不超过 p 的多项式。称一元多项式函数 $\det(P(z))$（或记为 $|P(z)|$）为模型的（逆序）特征多项式。

利用矩阵推移函数 $P(B)$ 可以将模型简记为

$$P(B) r_t = \phi_0 + a_t$$

如果 r_t 弱平稳，则在下式的逆矩阵存在的条件下

$$\mu = E r_t = (I - \Phi_1 - \cdots - \Phi_p)^{-1} \phi_0 = (P(1))^{-1} \phi_0$$

中心化 r_t 得 $\tilde{r}_t = r_t - \mu$，这时模型(24.12)变成

$$\tilde{r}_t = \Phi_1 \tilde{r}_{t-1} + \cdots + \Phi_p \tilde{r}_{t-p} + a_t \tag{24.14}$$

称 VAR(p) 满足平稳性条件，如果特征多项式 $\det(P(z))$ 的根都在单位圆外。当 VAR(p) 满足平稳性条件时，有如下的 Wold 表示的平稳解：

$$r_t = \mu + \sum_{j=0}^{\infty} \Psi_j a_{t-j},$$

其中 $j \to \infty$ 时 Ψ_j 的元素以负指数速度趋于 0，Ψ_j 有如下的递推公式:

$$\begin{aligned} \Psi_0 &= I, \\ \Psi_j &= \sum_{s=0}^{p} \Phi_s \Psi_{j-s}, \end{aligned}$$

对 $s < 0$ 记 $\Psi_s = 0$。

由上述 Wold 表示可知，对平稳解 r_t 有

- $\mathrm{Cov}(r_t, a_t) = \Sigma = \mathrm{Var}(a_t)$;
- $\mathrm{Cov}(r_{t-l}, a_t) = 0 (l > 0)$。

互协方差阵满足如下递推关系

$$\Gamma_l = \Phi_1 \Gamma_{l-1} + \cdots + \Phi_p \Gamma_{l-p},\ l > 0$$

这个性质称为 VAR(p) 的**矩方程**，是一元 AR(p) 模型的 Yule-Walker 方程的多元形式。用 CCM 表示为

$$\rho_l = \Upsilon_1 \rho_{l-1} + \cdots + \Upsilon_p \rho_{l-p},\ l > 0$$

其中 $\Upsilon_j = D^{-\frac{1}{2}}\Phi_j D^{\frac{1}{2}}$，$D$ 是 Σ 的对角阵。

事实上，VAR(p) 模型可以改写成一个 VAR(1) 模型，然后可以用 VAR(1) 模型的结果去研究 VAR(p) 模型的性质。令

$$x_t = \begin{pmatrix} \tilde{r}_t \\ \tilde{r}_{t-1} \\ \vdots \\ \tilde{r}_{t-p+1} \end{pmatrix} \quad b_t = \begin{pmatrix} a_t \\ 0_{k\times 1} \\ \vdots \\ 0_{k\times 1} \end{pmatrix}$$

$$\mathrm{Var}(b_t) = \begin{pmatrix} \Sigma & 0_{k\times k(p-1)} \\ 0_{k(p-1)\times k} & 0_{k(p-1)\times k(p-1)} \end{pmatrix}$$

$$\Phi^* = \begin{pmatrix} \Phi_1 & \Phi_2 & \cdots & \Phi_{p-1} & \Phi_p \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{pmatrix}$$

则

$$x_t = \Phi^* x_{t-1} + b_t \tag{24.15}$$

是关于 kp 元时间序列 x_t 的 VAR(1) 模型。

r_t 弱平稳当且仅当 x_t 弱平稳，其充分必要条件是 Φ^* 的所有特征值模小于 1，计算可知这相当于要求用行列式表示的 z 的多项式 $|P(z)|$ 的根都在单位圆外, $P(z)$ 定义见(24.13)。但是，因为计算 $\det(I - \Phi_1 z - \cdots - \Phi_p z^p)$ 这种用行列式表示的多项式的根还需要求多项式的各个系数，而求 Φ^* 的所有特征值或模最大的特征值则有比较成熟的数值方法，所以为了判断一个 VAR(p) 模型是否满足平稳性条件，可以生成 Φ^* 并判断其模最大的特征值是否模小于 1。

如果 $\det(P(1)) = 0$，或 Φ^* 有等于 1 的特征值，即有单位根，则 r_t 的各个分量中至少有一个是单位根过程，各个分量之间还有可能存在协整关系，可以用 VECM(向量误差修正模型) 建模，参见25。

对 VAR(p) 模型，系数矩阵 $\Phi_l = (\phi_{ij}(l))_{k\times k}$ 的值也表现了 r_t 各个分量之间的先导、滞后关系。比如 $k = 2$ 时，如果所有 $\phi_{12}(l) = 0$，但存在 $l > 0$ 使得 $\phi_{21}(l) \neq 0$，则 r_{1t} 不依赖于 r_{2t} 的过去值，r_{2t} 依赖于 r_{1t} 的过去值。r_{1t} 是 r_{2t} 的格兰杰原因，r_{2t} 不是 r_{1t} 的格兰杰原因。

24.7 估计和定阶

VAR 模型建模也基本遵循定阶、模型估计和模型检验这样的反复尝试过程。一元的 PACF 可以推广到多元情形用以辅助定阶。

考虑如下的阶数递进的 VAR 模型：

$$r_t = \phi_0 + \Phi_1 r_{t-1} + a_t \tag{24.16}$$

$$r_t = \phi_0 + \Phi_1 r_{t-1} + \Phi_2 r_{t-2} + a_t \tag{24.17}$$

$$\vdots \tag{24.18}$$

$$r_t = \phi_0 + \Phi_1 r_{t-1} + \cdots + \Phi_p r_{t-p} + a_t \tag{24.19}$$

模型参数可以对每个方程分别用 OLS（最小二乘）方法估计，在多元统计分析文献中称为多元线性估计，参见 (Johnson & Wichern, 1998)。

对于(24.19)的第 i 个方程，令 $\hat{\Phi}_j^{(i)}$ 是 Φ_j 的最小二乘估计，$\hat{\phi}_0^{(i)}$ 是 ϕ_0 的的最小二乘估计。残差为

$$\hat{a}_t^{(i)} = r_t - \hat{\Phi}_1^{(i)} r_{t-1} - \cdots - \hat{\Phi}_i^{(i)} r_{t-i}$$

当 $i=0$ 时定义 $\hat{a}_t^{(0)} = r_t - \bar{r}$，其中 $\bar{r}$ 是 r_t 的样本均值。

残差的方差阵估计为

$$\hat{\Sigma}_i = \frac{1}{T-(k+1)i-1} \sum_{t=i+1}^{T} \hat{a}_t^{(i)} [\hat{a}_t^{(i)}]^T$$

(参见 (Tsay, 2014) 节 2.5.1 P.34。)

为了定阶，可以对 $l = 1, 2, ...$ 逐一地检验 $H_0 : \Phi_l = 0 \leftrightarrow H_a : \Phi_l \neq 0$。

对 $l=1$，要检验 $H_0 : \Phi_1 = 0 \leftrightarrow H_a : \Phi_1 \neq 0$, 使用检验统计量

$$M(1) = -(T - k - \frac{5}{2}) \ln \frac{|\hat{\Sigma}_1|}{|\hat{\Sigma}_0|}$$

在一些正则性条件下当 H_0 成立时 $M(1)$ 渐近服从 $\chi^2(k^2)$ 分布，参见 (Tiao & Box, 1981)。计算右侧 p 值。

一般地，可以通过(24.19)的第 $l-1$ 个方程和第 l 个方程检验 $H_0 : \Phi_l = 0 \leftrightarrow H_a : \Phi_l \neq 0$。取检验统计量

$$M(l) = -(T - l - kl - \frac{3}{2}) \ln \frac{|\hat{\Sigma}_l|}{|\hat{\Sigma}_{l-1}|}$$

在 H_0 下 $M(l)$ 渐近服从 $\chi^2(k^2)$ 分布，计算右侧 p 值。

另一种定阶方法是利用 AIC 或其它类似的信息准则。设 $\{a_t\}$ 是独立同多元正态分布的多元白噪声，可以用最大似然 (ML) 方法估计模型，对于 VAR 模型，最小二乘估计 $\hat{\phi}_0$ 和 $\hat{\Phi}_j$ 与条件最大似然估计等价，但误差项方差阵 Σ 的 ML 估计为

$$\tilde{\Sigma}_i = \frac{1}{T} \sum_{t=i+1}^{T} \hat{a}_t^{(i)} [\hat{a}_t^{(i)}]^T$$

VAR(i) 模型在正态假定下的 AIC 值定义为

$$\mathrm{AIC}(i) = \ln |\tilde{\Sigma}_i| + \frac{2k^2 i}{T}$$

取最终的阶 p 使得 $\mathrm{AIC}(p) = \min_{0 \le i \le p_0} \mathrm{AIC}(i)$，其中 p_0 是预先规定的可取的最大阶数。

类似的信息准则还有

$$\mathrm{BIC}(i) = \ln |\tilde{\Sigma}_i| + \frac{k^2 i \ln T}{T}$$

$$\mathrm{HQ}(i) = \ln |\tilde{\Sigma}_i| + \frac{k^2 i \ln \ln T}{T}$$

HQ 准则是 Hannan 和 Quinn(1979) 提出的。

这些准则的选择结果不受量纲的影响，对数据乘以常数 λ，结果会使得 $\ln |\tilde{\Sigma}_i|$ 固定地增加 $\ln(\lambda^{2k})$。

24.7.1 英、加、美 GDP 的 VAR 建模

考虑英国、加拿大、美国 GDP 的季度增长率，时间从 1980 年第二季度到 2011 年第二季度。数据经过季节调整，来自圣路易斯联邦储备银行的数据库。GDP 是以本国货币百亿为单位，增长率表示为 GDP 的对数的差分。

读入数据，计算对数增长率:

```
da <- read_table("q-gdp-ukcaus.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   year = col_double(),
##   mon = col_double(),
##   uk = col_double(),
##   ca = col_double(),
##   us = col_double()
## )
```

```
ts.gdp3 <- ts(as.matrix(da[,c("uk", "ca", "us")]),
              start=c(1980,1), frequency=4)
ts.gdp3r <- diff(log(ts.gdp3))*100
```

三个国家的季度 GDP 对数增长率的时间序列图：

```
plot(as.xts(ts.gdp3r), type="l",
     multi.panel=TRUE, theme="white",
     main=" 英国、加拿大、美国 GDP 的季度增长率 (%)",
     major.ticks="years",
     grid.ticks.on = "years")
```

利用 MTS 包的 `VAR()` 函数估计 VAR(1) 模型：

```
library(MTS, quietly = TRUE)
Z <- coredata(as.xts(ts.gdp3r))
m1.gdp3r <- MTS::VAR(Z, 1)
```

```
## Constant term:
## Estimates:  0.1713324 0.1182869 0.2785892
## Std.Error:  0.06790162 0.07193106 0.07877173
## AR coefficient matrix
## AR( 1 )-matrix
##       [,1]  [,2]   [,3]
## [1,] 0.434 0.189 0.0373
## [2,] 0.185 0.245 0.3917
## [3,] 0.322 0.182 0.1674
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0811 0.0827 0.0872
## [2,] 0.0859 0.0877 0.0923
## [3,] 0.0940 0.0960 0.1011
##
## Residuals cov-mtx:
```

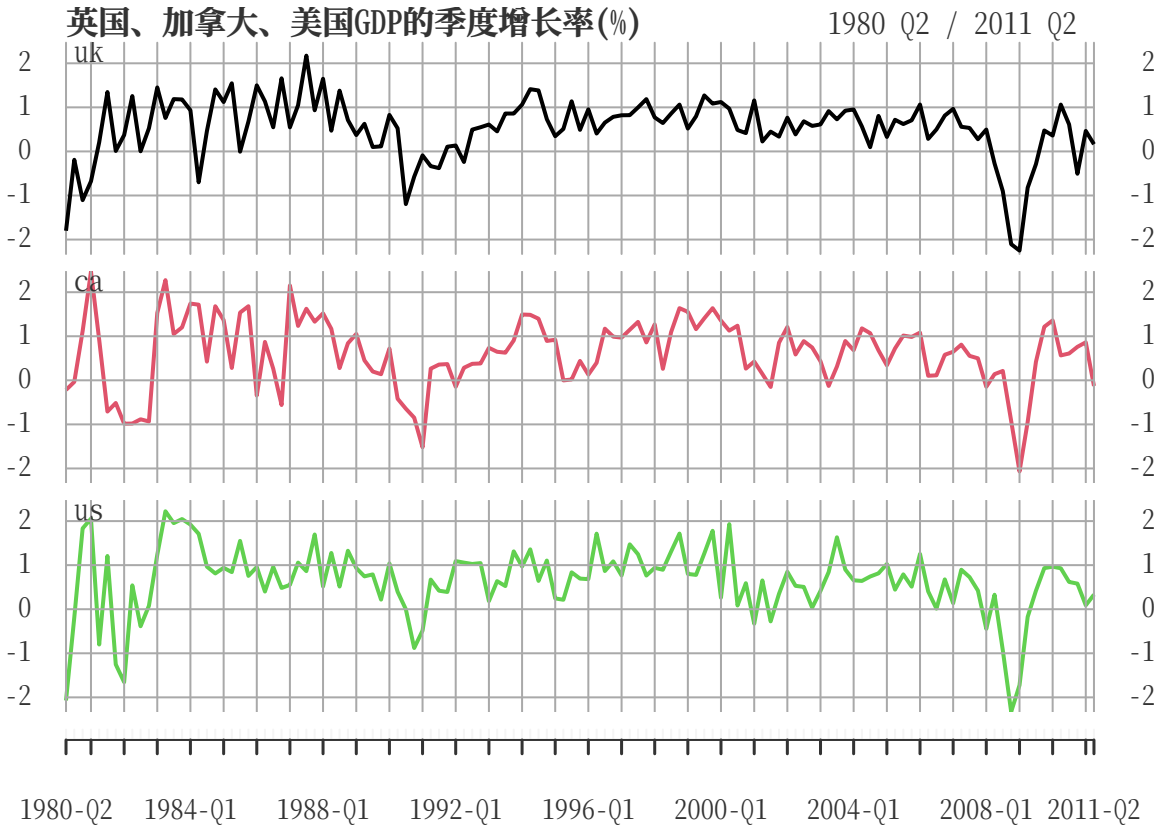


图 24.1: 英国、加拿大、美国 GDP 的季度增长率

```
##             [,1]       [,2]       [,3]
## [1,] 0.28933472 0.01965508 0.06619853
## [2,] 0.01965508 0.32469319 0.16862723
## [3,] 0.06619853 0.16862723 0.38938665
##
## det(SSE) =  0.02721916
## AIC =  -3.459834
## BIC =  -3.256196
## HQ  =  -3.377107
```

估计 VAR(2) 模型：

```
Z <- coredata(as.xts(ts.gdp3r))
m2.gdp3r <- MTS::VAR(Z, 2)
```

```
## Constant term:
## Estimates:  0.1258163 0.1231581 0.2895581
## Std.Error:  0.07266338 0.07382941 0.0816888
## AR coefficient matrix
## AR( 1 )-matrix
##       [,1]  [,2]   [,3]
## [1,] 0.393 0.103 0.0521
## [2,] 0.351 0.338 0.4691
## [3,] 0.491 0.240 0.2356
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0934 0.0984 0.0911
## [2,] 0.0949 0.1000 0.0926
## [3,] 0.1050 0.1106 0.1024
## AR( 2 )-matrix
##         [,1]   [,2]     [,3]
## [1,]  0.0566  0.106  0.01889
## [2,] -0.1914 -0.175 -0.00868
## [3,] -0.3120 -0.131  0.08531
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0924 0.0876 0.0938
## [2,] 0.0939 0.0890 0.0953
## [3,] 0.1038 0.0984 0.1055
##
## Residuals cov-mtx:
##             [,1]       [,2]       [,3]
## [1,] 0.28244420 0.02654091 0.07435286
## [2,] 0.02654091 0.29158166 0.13948786
## [3,] 0.07435286 0.13948786 0.35696571
```

```
##
## det(SSE) =  0.02258974
## AIC =  -3.502259
## BIC =  -3.094982
## HQ  =  -3.336804
```

VAR(1) 的 AIC 为 -3.46，VAR(2) 的 AIC 为 -3.50, VAR(2) 占优。设 r_t 的三个分量分别是英国、加拿大、美国的 GDP 对数增长率（单位为百分之一），得到的模型 VAR(2) 模型可以写成

$$
\begin{aligned}
r_t = &\begin{pmatrix} 0.13 \\ 0.12 \\ 0.29 \end{pmatrix} + \begin{pmatrix} 0.39 & 0.10 & 0.05 \\ 0.35 & 0.34 & 0.47 \\ 0.49 & 0.24 & 0.24 \end{pmatrix} r_{t-1} \\
&+ \begin{pmatrix} 0.06 & 0.11 & 0.02 \\ -0.19 & -0.18 & -0.01 \\ -0.31 & -0.13 & 0.09 \end{pmatrix} r_{t-2} + a_t
\end{aligned}
$$

$$
\mathrm{Var}(a_t) = \begin{pmatrix} 0.28 & 0.03 & 0.07 \\ 0.03 & 0.29 & 0.14 \\ 0.07 & 0.14 & 0.36 \end{pmatrix}
$$

结果中也包括系数的标准误差，可以看出有些系数是不显著的。比如，三个序列的次序为英国、加拿大、美国，$\phi_{12}(l)$, $\phi_{13}(l)(l = 1, 2)$ 都不显著，说明英国的 GDP 季度增长率不受加拿大和美国的过去值的影响，或者粗略地说，加拿大和美国的 GDP 季度增长率不是英国的 GDP 季度增长率的格兰格原因。更严格的分析应该进行假设检验。

利用 MTS 包的 VARorder 函数可以计算 VAR 定阶的 $M(i)$ 统计量检验和各种信息准则：

```
library(MTS, quietly = TRUE)
Z <- coredata(as.xts(ts.gdp3r))
m3.gdp3r <- VARorder(Z/100)
```

```
## selected order: aic =  2
## selected order: bic =  1
## selected order: hq =  1
## Summary table:
##         p      AIC      BIC       HQ     M(p) p-value
##  [1,]   0 -30.9560 -30.9560 -30.9560   0.0000  0.0000
##  [2,]   1 -31.8830 -31.6794 -31.8003 115.1329  0.0000
##  [3,]   2 -31.9643 -31.5570 -31.7988  23.5389  0.0051
##  [4,]   3 -31.9236 -31.3127 -31.6754  10.4864  0.3126
##  [5,]   4 -31.8971 -31.0826 -31.5662  11.5767  0.2382
##  [6,]   5 -31.7818 -30.7636 -31.3682   2.7406  0.9737
##  [7,]   6 -31.7112 -30.4893 -31.2148   6.7822  0.6598
##  [8,]   7 -31.6180 -30.1925 -31.0389   4.5469  0.8719
##  [9,]   8 -31.7570 -30.1279 -31.0952  24.4833  0.0036
## [10,]   9 -31.6897 -29.8569 -30.9451   6.4007  0.6992
## [11,]  10 -31.5994 -29.5630 -30.7721   4.3226  0.8889
## [12,]  11 -31.6036 -29.3636 -30.6936  11.4922  0.2435
```

```
## [13,] 12 -31.6183 -29.1746 -30.6255  11.8168  0.2238
## [14,] 13 -31.6718 -29.0245 -30.5964  14.1266  0.1179
```

从 AIC 比较来看，应该取 $p = 2$。从检验来看，从 $i = 3$ 阶开始 Φ_i 就不显著了，但 Φ_1 和 Φ_2 显著，所以应该取 $p = 2$ 阶。

还可以用 vars 包的 `VAR()` 函数估计。估计 VAR(1)：

```
da1 <- coredata(as.xts(ts.gdp3r))
var1 <- vars::VAR(da1, p=1)
summary(var1)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: uk, ca, us
## Deterministic variables: const
## Sample size: 124
## Log Likelihood: -304.407
## Roots of the characteristic polynomial:
## 0.7091 0.08735 0.05004
## Call:
## vars::VAR(y = da1, p = 1)
##
##
## Estimation results for equation uk:
## ===================================
## uk = uk.l1 + ca.l1 + us.l1 + const
##
##       Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.43435    0.08106   5.358 4.12e-07 ***
## ca.l1  0.18888    0.08275   2.282   0.0242 *
## us.l1  0.03727    0.08716   0.428   0.6697
## const  0.17133    0.06790   2.523   0.0129 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.5468 on 120 degrees of freedom
## Multiple R-Squared: 0.3687,  Adjusted R-squared: 0.3529
## F-statistic: 23.36 on 3 and 120 DF,  p-value: 5.596e-12
##
##
## Estimation results for equation ca:
```

```
## ====================================
## ca = uk.l1 + ca.l1 + us.l1 + const
##
##       Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.18499    0.08587   2.154   0.0332 *
## ca.l1  0.24475    0.08766   2.792   0.0061 **
## us.l1  0.39166    0.09233   4.242 4.38e-05 ***
## const  0.11829    0.07193   1.644   0.1027
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.5792 on 120 degrees of freedom
## Multiple R-Squared: 0.4685,  Adjusted R-squared: 0.4552
## F-statistic: 35.26 on 3 and 120 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation us:
## ====================================
## us = uk.l1 + ca.l1 + us.l1 + const
##
##       Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.32153    0.09404   3.419 0.000859 ***
## ca.l1  0.18196    0.09600   1.895 0.060438 .
## us.l1  0.16740    0.10111   1.656 0.100410
## const  0.27859    0.07877   3.537 0.000577 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.6343 on 120 degrees of freedom
## Multiple R-Squared: 0.3044,  Adjusted R-squared: 0.287
## F-statistic:  17.5 on 3 and 120 DF,  p-value: 1.725e-09
##
##
##
## Covariance matrix of residuals:
##         uk      ca      us
## uk 0.29898 0.02031 0.06841
## ca 0.02031 0.33552 0.17425
## us 0.06841 0.17425 0.40237
##
## Correlation matrix of residuals:
```

```
##         uk      ca     us
## uk 1.00000 0.06413 0.1972
## ca 0.06413 1.00000 0.4742
## us 0.19722 0.47424 1.0000
```

与 MTS::VAR() 的输出格式不同，vars::VAR() 对每个分量单独输出其方程，好处是用我们熟悉的格式显示估计值、标准误差、检验统计量和 p 值。得到的模型与 MTS::VAR() 结果相同，白噪声方差估计略有差别。

用 AIC 自动选择阶，指定 ic="AIC" 和 lag.max=5:

```
da1 <- coredata(as.xts(ts.gdp3r))
var2 <- vars::VAR(da1, ic="AIC", lag.max=5)
summary(var2)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: uk, ca, us
## Deterministic variables: const
## Sample size: 121
## Log Likelihood: -252.647
## Roots of the characteristic polynomial:
## 0.785 0.7516 0.7516 0.7336 0.7336 0.6144 0.6144 0.5679 0.5679 0.5165 0.5122 0.5122
## Call:
## vars::VAR(y = da1, lag.max = 5, ic = "AIC")
##
##
## Estimation results for equation uk:
## ===================================
## uk = uk.l1 + ca.l1 + us.l1 + uk.l2 + ca.l2 + us.l2 + uk.l3 + ca.l3 + us.l3 + uk.l4 + ca.l4 + us.l4 +
##
##         Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.515685   0.095298   5.411  3.8e-07 ***
## ca.l1  0.071863   0.094459   0.761  0.44844
## us.l1  0.063904   0.088668   0.721  0.47264
## uk.l2 -0.050370   0.101228  -0.498  0.61979
## ca.l2  0.160043   0.098620   1.623  0.10754
## us.l2 -0.001984   0.095495  -0.021  0.98346
## uk.l3  0.052363   0.102949   0.509  0.61205
## ca.l3 -0.278830   0.098341  -2.835  0.00547 **
## us.l3  0.141145   0.093728   1.506  0.13501
## uk.l4  0.040130   0.091050   0.441  0.66028
## ca.l4  0.261731   0.086947   3.010  0.00325 **
## us.l4 -0.246485   0.088234  -2.794  0.00617 **
## const  0.147957   0.074786   1.978  0.05043 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.5013 on 108 degrees of freedom
## Multiple R-Squared: 0.478,   Adjusted R-squared:  0.42
## F-statistic: 8.242 on 12 and 108 DF,  p-value: 7.84e-11
##
##
## Estimation results for equation ca:
## ===================================
## ca = uk.l1 + ca.l1 + us.l1 + uk.l2 + ca.l2 + us.l2 + uk.l3 + ca.l3 + us.l3 + uk.l4 + ca.l4 + us.l4 +
##
##       Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.37782    0.10199   3.705 0.000336 ***
## ca.l1  0.31603    0.10109   3.126 0.002276 **
## us.l1  0.40963    0.09489   4.317 3.52e-05 ***
## uk.l2 -0.17399    0.10834  -1.606 0.111193
## ca.l2 -0.25407    0.10554  -2.407 0.017771 *
## us.l2  0.06295    0.10220   0.616 0.539247
## uk.l3  0.09615    0.11018   0.873 0.384757
## ca.l3  0.12035    0.10525   1.143 0.255362
## us.l3  0.01366    0.10031   0.136 0.891925
## uk.l4  0.07470    0.09744   0.767 0.445006
## ca.l4 -0.09031    0.09305  -0.971 0.333959
## us.l4 -0.09781    0.09443  -1.036 0.302622
## const  0.07757    0.08004   0.969 0.334593
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.5365 on 108 degrees of freedom
## Multiple R-Squared: 0.5661,  Adjusted R-squared: 0.5179
## F-statistic: 11.74 on 12 and 108 DF,  p-value: 8.189e-15
##
##
## Estimation results for equation us:
## ===================================
## us = uk.l1 + ca.l1 + us.l1 + uk.l2 + ca.l2 + us.l2 + uk.l3 + ca.l3 + us.l3 + uk.l4 + ca.l4 + us.l4 +
##
##       Estimate Std. Error t value Pr(>|t|)
## uk.l1  0.51905    0.10938   4.745 6.41e-06 ***
## ca.l1  0.17304    0.10841   1.596   0.1134
```

```
## us.l1  0.15039    0.10177   1.478   0.1424
## uk.l2 -0.21784    0.11618  -1.875   0.0635 .
## ca.l2 -0.15931    0.11319  -1.407   0.1622
## us.l2  0.22561    0.10960   2.058   0.0420 *
## uk.l3  0.04783    0.11816   0.405   0.6864
## ca.l3 -0.07856    0.11287  -0.696   0.4879
## us.l3  0.07384    0.10758   0.686   0.4939
## uk.l4  0.15409    0.10450   1.475   0.1433
## ca.l4 -0.15182    0.09979  -1.521   0.1311
## us.l4 -0.05350    0.10127  -0.528   0.5984
## const  0.23868    0.08584   2.781   0.0064 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.5754 on 108 degrees of freedom
## Multiple R-Squared: 0.4531,  Adjusted R-squared: 0.3923
## F-statistic: 7.457 on 12 and 108 DF,  p-value: 7.522e-10
##
##
##
## Covariance matrix of residuals:
##         uk      ca     us
## uk 0.25130 0.04912 0.1001
## ca 0.04912 0.28783 0.1084
## us 0.10009 0.10840 0.3310
##
## Correlation matrix of residuals:
##        uk     ca     us
## uk 1.0000 0.1826 0.3470
## ca 0.1826 1.0000 0.3512
## us 0.3470 0.3512 1.0000
```

结果选择了 4 阶模型，与 `MTS::VARorder()` 结果不同。

vars 包的 `SVAR()` 函数可以用来估计 VAR 的结构形式，但其定义不同于我们教材的定义。

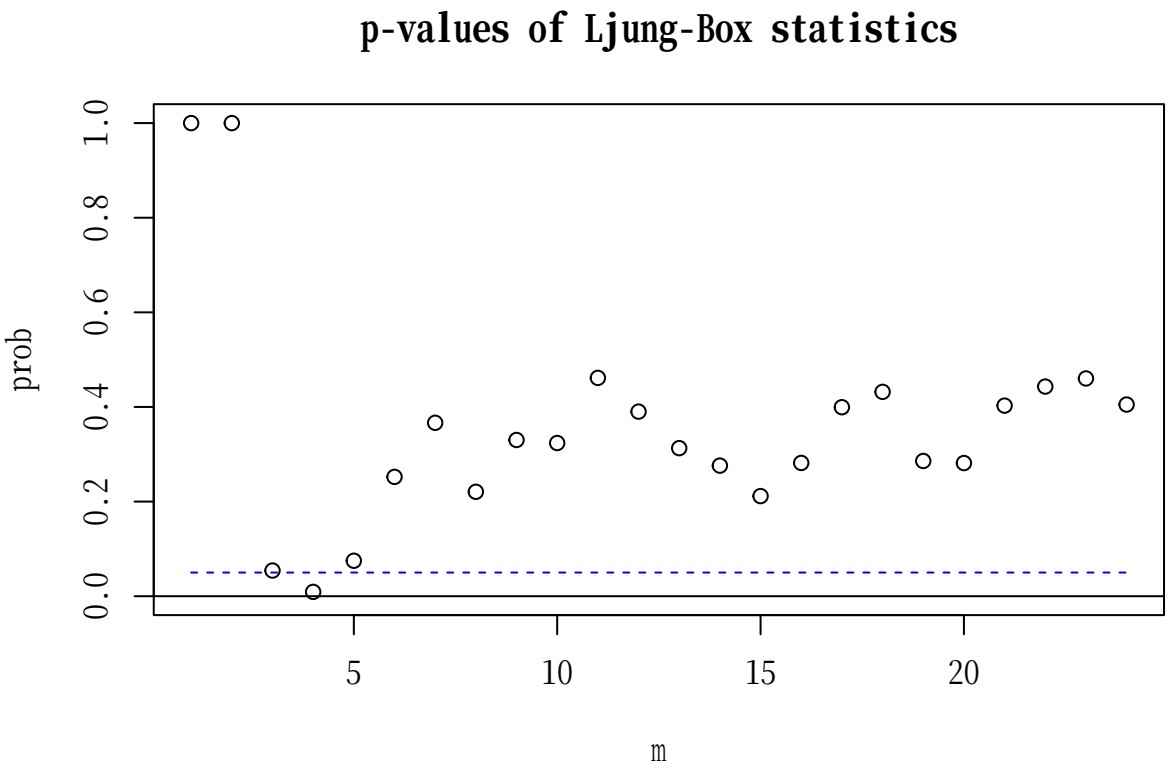
24.8 模型检验

可以计算模型残差，对残差进行多元白噪声检验（多元混成检验）。残差的多元混成检验因为使用了估计的参数，所以统计量的自由度会减少 k^2p，这是系数矩阵 Φ_j, $j = 1, 2, \dots, p$ 中的参数个数。如果系数矩阵中某些参数固定为 0，应按无约束的参数个数计算要扣除的自由度。在 MTS 包的 `mq()` 函数中用 `adj=` 指定需要减少的自由度。

例 3.2 对例 3.1 的三个国家的 GDP 季度增速建模，VAR(2) 模型的残差的多元混成检验程序如下：

```
resi <- m2.gdp3r$residuals
MTS::mq(resi, adj=3^2 * 2)
```

```
## Ljung-Box Statistics:
##           m       Q(m)     df    p-value
##  [1,]   1.000     0.816  -9.000     1.00
##  [2,]   2.000     3.978   0.000     1.00
##  [3,]   3.000    16.665   9.000     0.05
##  [4,]   4.000    35.122  18.000     0.01
##  [5,]   5.000    38.189  27.000     0.07
##  [6,]   6.000    41.239  36.000     0.25
##  [7,]   7.000    47.621  45.000     0.37
##  [8,]   8.000    61.677  54.000     0.22
##  [9,]   9.000    67.366  63.000     0.33
## [10,]  10.000    76.930  72.000     0.32
## [11,]  11.000    81.567  81.000     0.46
## [12,]  12.000    93.112  90.000     0.39
## [13,]  13.000   105.327  99.000     0.31
## [14,]  14.000   116.279 108.000     0.28
## [15,]  15.000   128.974 117.000     0.21
## [16,]  16.000   134.704 126.000     0.28
## [17,]  17.000   138.552 135.000     0.40
## [18,]  18.000   146.256 144.000     0.43
## [19,]  19.000   162.418 153.000     0.29
## [20,]  20.000   171.948 162.000     0.28
## [21,]  21.000   174.913 171.000     0.40
## [22,]  22.000   182.056 180.000     0.44
## [23,]  23.000   190.276 189.000     0.46
## [24,]  24.000   202.141 198.000     0.41
```



检验结果只有在滞后 4 显著，基本可以认为模型是充分的。

对残差还可以进行异方差等检验。

24.9 模型简化

当 VAR 中分量个数 k 较大时，模型有许多参数，系数矩阵中参数个数为 k^2p 个。如果没有先验知识要求参数非零，可以将不显著的参数约束为零再估计。

模型简化没有公认的最优做法。一种办法是计算参数估计值与标准误差的比值，称为 t 比值，将 t 比值绝对值小于 1.645(相当于 0.10 检验水平) 或者小于 1.96（相当于 0.05 检验水平）的系数设置为零。MTS 包的 `refVAR()` 函数输入无约束的 VAR 建模结果，以及 `thres=1.645` 或 `thres=1.96` 这样的 t 比值界限，生成设置部分系数为零的约束估计结果，得到精简的模型。可以通过 AIC 比较完全模型和约束模型。

例 3.3 对三个国家的 GDP 对数增长率的 VAR(2) 模型进行化简。

```
library(MTS, quietly = TRUE)
Z <- coredata(as.xts(ts.gdp3r))
cat("================ Full model:\n")
```

```
## ================ Full model:
```

```
mods1.gdp3r <- VAR(Z, 2)
```

```
## Constant term:
```

```
## Estimates:  0.1258163 0.1231581 0.2895581
## Std.Error:  0.07266338 0.07382941 0.0816888
## AR coefficient matrix
## AR( 1 )-matrix
##       [,1]  [,2]   [,3]
## [1,] 0.393 0.103 0.0521
## [2,] 0.351 0.338 0.4691
## [3,] 0.491 0.240 0.2356
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0934 0.0984 0.0911
## [2,] 0.0949 0.1000 0.0926
## [3,] 0.1050 0.1106 0.1024
## AR( 2 )-matrix
##         [,1]   [,2]     [,3]
## [1,]  0.0566  0.106  0.01889
## [2,] -0.1914 -0.175 -0.00868
## [3,] -0.3120 -0.131  0.08531
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0924 0.0876 0.0938
## [2,] 0.0939 0.0890 0.0953
## [3,] 0.1038 0.0984 0.1055
##
## Residuals cov-mtx:
##             [,1]       [,2]       [,3]
## [1,] 0.28244420 0.02654091 0.07435286
## [2,] 0.02654091 0.29158166 0.13948786
## [3,] 0.07435286 0.13948786 0.35696571
##
## det(SSE) =  0.02258974
## AIC =  -3.502259
## BIC =  -3.094982
## HQ  =  -3.336804
```

```
cat("\n\n================ Restricted model:\n")
```

```
##
##
## ================ Restricted model:
```

```
mods2.gdp3r <- refVAR(mods1.gdp3r, thres=1.96)
```

```
## Constant term:
## Estimates:  0.1628247 0 0.2827525
```

```
## Std.Error:  0.06814101 0 0.07972864
## AR coefficient matrix
## AR( 1 )-matrix
##       [,1]  [,2]  [,3]
## [1,] 0.467 0.207 0.000
## [2,] 0.334 0.270 0.496
## [3,] 0.468 0.225 0.232
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0790 0.0686 0.0000
## [2,] 0.0921 0.0875 0.0913
## [3,] 0.1027 0.0963 0.1023
## AR( 2 )-matrix
##        [,1] [,2] [,3]
## [1,]  0.000    0    0
## [2,] -0.197    0    0
## [3,] -0.301    0    0
## standard error
##        [,1] [,2] [,3]
## [1,] 0.0000    0    0
## [2,] 0.0921    0    0
## [3,] 0.1008    0    0
##
## Residuals cov-mtx:
##            [,1]       [,2]       [,3]
## [1,] 0.29003669 0.01803456 0.07055856
## [2,] 0.01803456 0.30802503 0.14598345
## [3,] 0.07055856 0.14598345 0.36268779
##
## det(SSE) =  0.02494104
## AIC =  -3.531241
## BIC =  -3.304976
## HQ  =  -3.439321
```

无约束的 VAR(2) 的 AIC 值为 -3.50，约束 10 个参数为零的 VAR(2) 的 AIC 值为 -3.53，所以约束模型较优。约

束模型为

$$r_t = \begin{pmatrix} 0.16 \\ 0 \\ 0.29 \end{pmatrix} + \begin{pmatrix} 0.47 & 0.21 & 0 \\ 0.33 & 0.27 & 0.50 \\ 0.47 & 0.23 & 0.23 \end{pmatrix} r_{t-1} + \begin{pmatrix} 0 & 0 & 0 \\ -0.20 & 0 & 0 \\ -0.30 & 0 & 0 \end{pmatrix} r_{t-2} + a_t$$

$$\mathrm{Var}(a_t) = \begin{pmatrix} 0.29 & 0.02 & 0.07 \\ 0.02 & 0.31 & 0.15 \\ 0.07 & 0.15 & 0.36 \end{pmatrix}$$

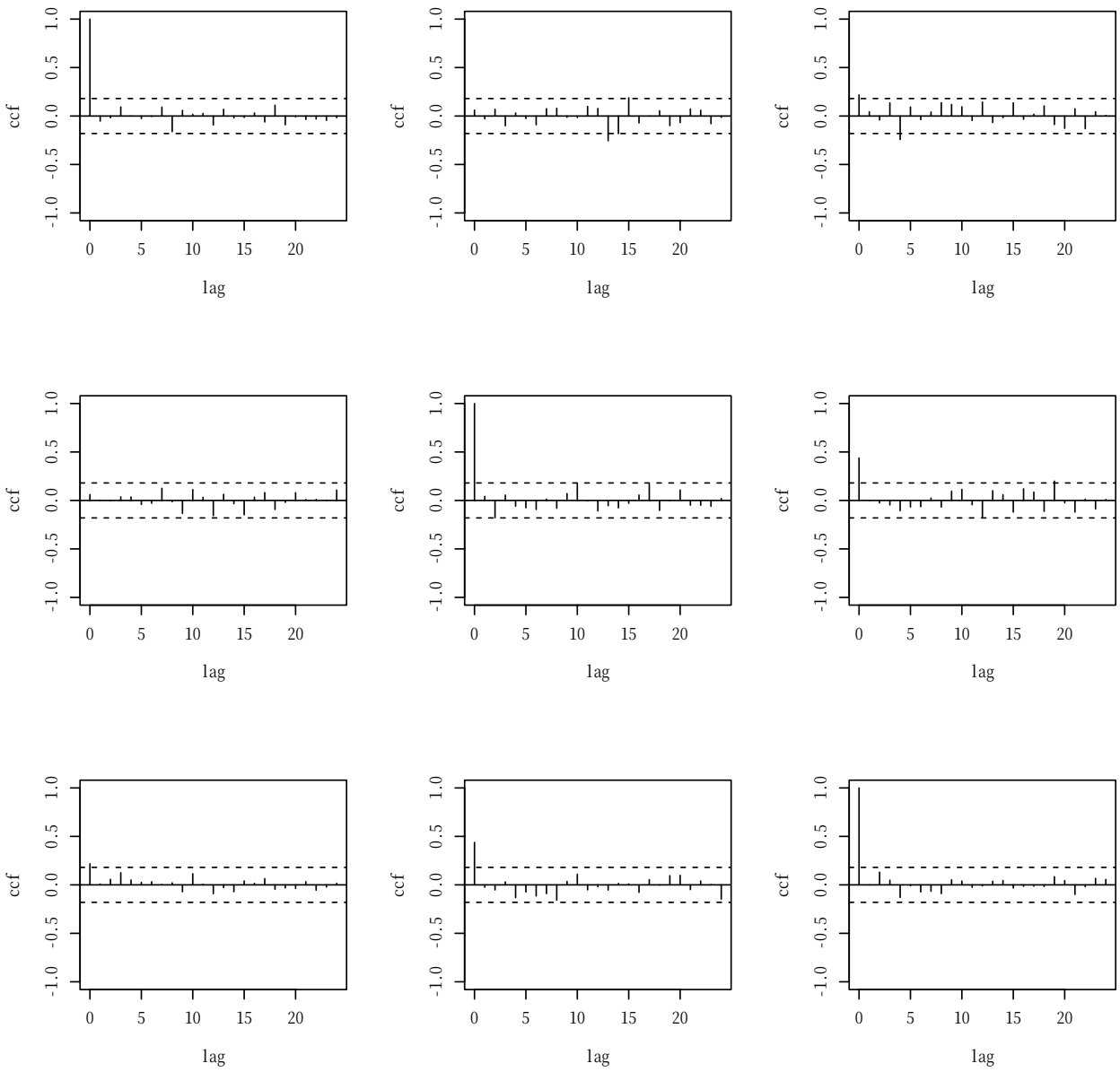
对这个简化的模型的检验，可以提取残差后用 `MTS::mq()` 函数检验，这时自由度缩减个数由原来的 $3^2 \times 2 = 18$ 个，减少到 9 个，因为模型中约束了 9 个参数等于零。MTS 包还提供了一个 `MTSdiag()` 函数，输入模型结果和 `adj=` 自由度缩减个数，作残差的 CCM 估计表、图和残差的多元混成检验：

```
library(MTS, quietly = TRUE)
MTSdiag(mods2.gdp3r, adj=9)
```

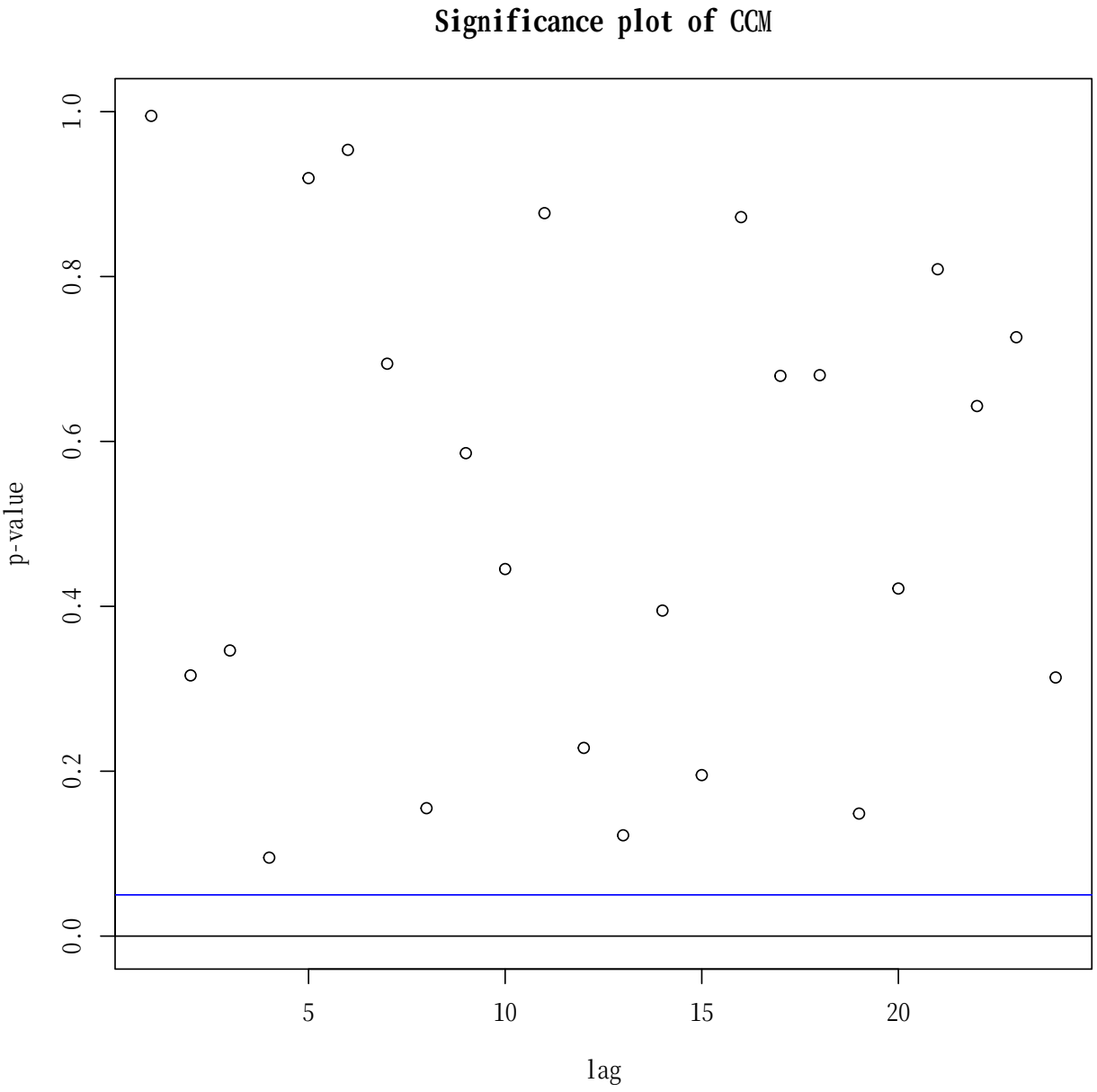
```
## [1] "Covariance matrix:"
##        uk     ca     us
## uk 0.2924 0.0182 0.0711
## ca 0.0182 0.3084 0.1472
## us 0.0711 0.1472 0.3657
## CCM at lag:  0
##        [,1]   [,2]  [,3]
## [1,] 1.0000 0.0605 0.218
## [2,] 0.0605 1.0000 0.438
## [3,] 0.2175 0.4382 1.000
## Simplified matrix:
## CCM at lag:  1
## . . .
## . . .
## . . .
## CCM at lag:  2
## . . .
## . . .
## . . .
## CCM at lag:  3
## . . .
## . . .
## . . .
## CCM at lag:  4
## . . -
## . . .
```

```
## . . .
## CCM at lag:  5
## . . .
## . . .
## . . .
## CCM at lag:  6
## . . .
## . . .
## . . .
## CCM at lag:  7
## . . .
## . . .
## . . .
## CCM at lag:  8
## . . .
## . . .
## . . .
## CCM at lag:  9
## . . .
## . . .
## . . .
## CCM at lag:  10
## . . .
## . . .
## . . .
## CCM at lag:  11
## . . .
## . . .
## . . .
## CCM at lag:  12
## . . .
## . . .
## . . .
## CCM at lag:  13
## . - .
## . . .
## . . .
## CCM at lag:  14
## . - .
## . . .
## . . .
## CCM at lag:  15
## . + .
```

```
## . . .
## . . .
## CCM at lag:  16
## . . .
## . . .
## . . .
## CCM at lag:  17
## . . .
## . . .
## . . .
## CCM at lag:  18
## . . .
## . . .
## . . .
## CCM at lag:  19
## . . .
## . . +
## . . .
## CCM at lag:  20
## . . .
## . . .
## . . .
## CCM at lag:  21
## . . .
## . . .
## . . .
## CCM at lag:  22
## . . .
## . . .
## . . .
## CCM at lag:  23
## . . .
## . . .
## . . .
## CCM at lag:  24
## . . .
## . . .
## . . .
```

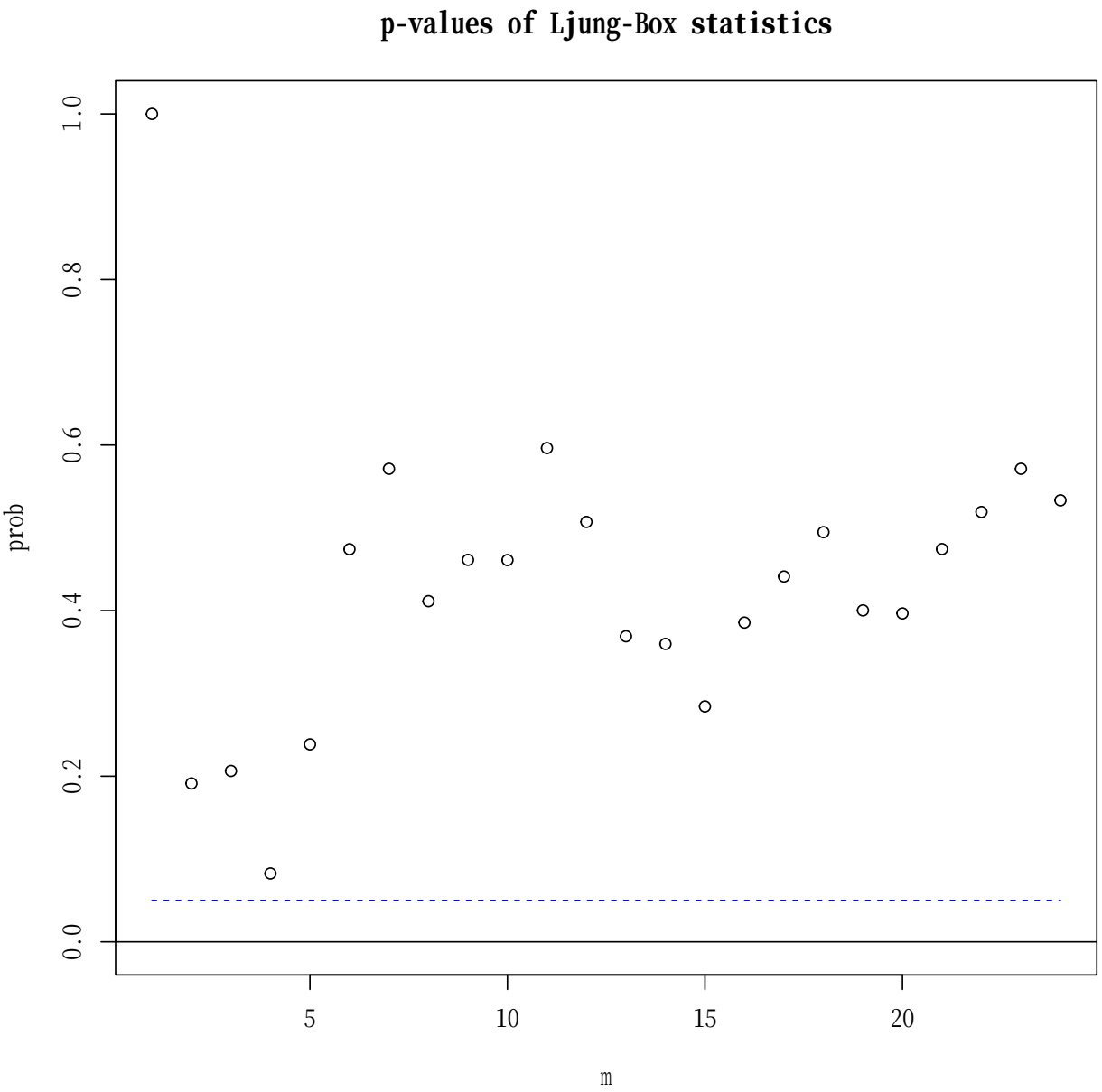



```
## Hit Enter for p-value plot of individual ccm:
```

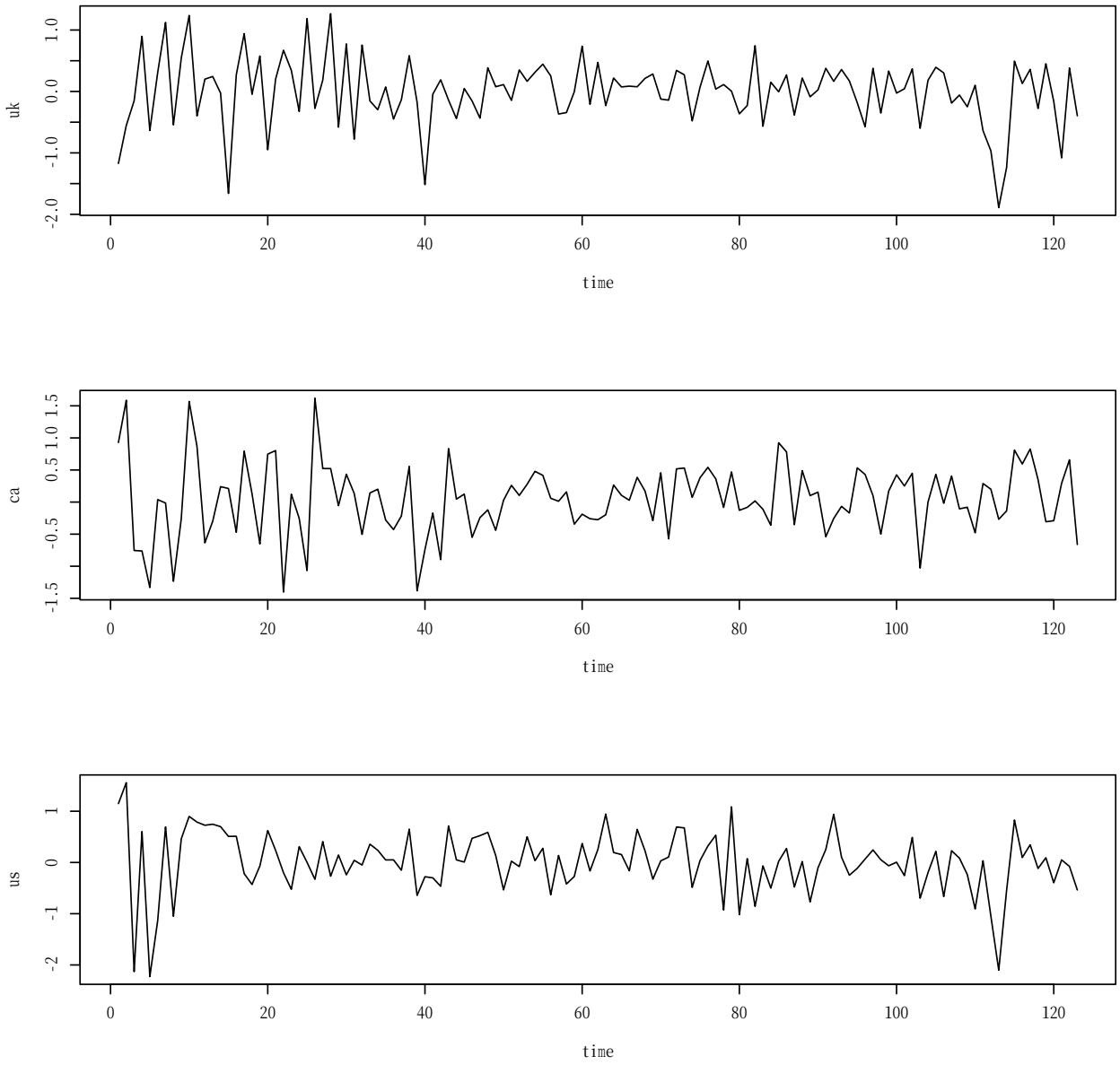



```
## Hit Enter to compute MQ-statistics:
##
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
##  [1,]   1.00      1.78    0.00     1.00
##  [2,]   2.00     12.41    9.00     0.19
##  [3,]   3.00     22.60   18.00     0.21
##  [4,]   4.00     37.71   27.00     0.08
##  [5,]   5.00     41.65   36.00     0.24
##  [6,]   6.00     44.95   45.00     0.47
##  [7,]   7.00     51.50   54.00     0.57
```

```
##  [8,]   8.00     64.87   63.00     0.41
##  [9,]   9.00     72.50   72.00     0.46
## [10,]  10.00     81.58   81.00     0.46
## [11,]  11.00     86.12   90.00     0.60
## [12,]  12.00     98.08   99.00     0.51
## [13,]  13.00    112.31  108.00     0.37
## [14,]  14.00    121.89  117.00     0.36
## [15,]  15.00    134.58  126.00     0.28
## [16,]  16.00    139.16  135.00     0.39
## [17,]  17.00    145.85  144.00     0.44
## [18,]  18.00    152.56  153.00     0.49
## [19,]  19.00    165.91  162.00     0.40
## [20,]  20.00    175.22  171.00     0.40
## [21,]  21.00    180.56  180.00     0.47
## [22,]  22.00    187.40  189.00     0.52
## [23,]  23.00    193.78  198.00     0.57
## [24,]  24.00    204.65  207.00     0.53
```




```
## Hit Enter to obtain residual plots:
```



从残差的 CCM 和多元混成检验结果来看，约束的模型是充分的。

注：蔡教授的多元金融时间序列专著中，残差混成检验时调整的自由度是 k^2p 还是 k^2p+k，没有讲得很清楚。在 §2.7.2（P.52）说明要调整的自由度是 k^2p，但是在 §2.7.3（P.57）的例子中，调整后的自由度为 $9m-12$，这里扣除的 12 个自由度加上约束的 9 个零参数，是 21 个参数，包括了 ϕ_0 部分。

从简化模型可以得到三国的 GDP 季度对数增长率服从如下的模型：

$$
\begin{aligned}
\text{英国:} r_{1t} &= 0.16 + 0.47r_{1,t-1} + 0.21r_{2,t-1} + a_{1t} \\
\text{加拿大:} r_{2t} &= 0.33r_{1,t-1} + 0.27r_{2,t-1} + 0.50r_{3,t-1} - 0.20r_{1,t-2} + a_{2t} \\
\text{美国:} r_{3t} &= 0.28 + 0.47r_{1,t-1} + 0.23r_{2,t-1} + 0.23r_{3,t-1} - 0.30r_{1,t-2} + a_{3t}
\end{aligned}
$$

残差的相关阵为：

$$\hat{\rho}_0 = \begin{pmatrix} 1.00 & 0.06 & 0.22 \\ 0.06 & 1.00 & 0.44 \\ 0.22 & 0.44 & 1.00 \end{pmatrix}$$

残差的相关阵计算程序：

```
cor(mods2.gdp3r$residuals)
```

```
##             [,1]       [,2]      [,3]
## [1,] 1.00000000 0.06054285 0.2175489
## [2,] 0.06054285 1.00000000 0.4382489
## [3,] 0.21754885 0.43824891 1.0000000
```

24.10 基于 VAR 模型的格兰杰因果性检验

如果模型可以简化为某些代表格兰杰因果性的系数等于零，则可以据此进行格兰杰因果性的检验。在二元的 VAR(1) 模型中，如果约束 $\phi_{12}(1) = 0$ 后的模型与无约束模型没有显著差异，则 r_{2t} 不是 r_{1t} 的格兰杰原因。p 阶以及 k 元的情形类似。

为了比较无约束与约束的模型，使用对数似然比检验，得到的统计量在约束参数等于零的零假设下渐近服从卡方分布。

MTS 包中 `GrangerTest()` 函数执行格兰杰因果性检验，默认第一个分量为单向的格兰杰原因，可以用 `locInput=` 序号指定哪一个或者哪几个分量作为原因。用基于 VAR 的方法检验格兰杰因果性，局限是各分量也必须平稳，不支持协整模型。

例 3.4

检验美国的 GDP 季度增长率是不是英国和加拿大的增长率的单向格兰杰原因。

```
library(MTS, quietly = TRUE, warn.conflicts=FALSE)
Z <- coredata(as.xts(ts.gdp3r))
GrangerTest(Z, p=2, locInput=3)
```

```
## Number of targeted zero parameters:  4
## Chi-square test for Granger Causality and p-value:  27.2262 1.789152e-05
```

美国是第 3 个分量，零假设为

$$H_0 : \phi_{31}(1) = \phi_{32}(1) = \phi_{31}(2) = \phi_{32}(2) = 0,$$

即预报 x_{3t} 时不需要用到 $x_{1,t-1}$, $x_{2,t-1}$, $x_{1,t-2}$, $x_{2,t-2}$。结果 p 值小于 0.05，拒绝 H_0，说明美国的 GDP 季度增长率**不是**英国和加拿大的增长率的单向格兰杰原因，即英国和加拿大也是美国的格兰杰原因。

对加拿大进行检验：

```
GrangerTest(Z, p=2, locInput=2)
```

```
## Number of targeted zero parameters:  4
## Chi-square test for Granger Causality and p-value:  48.83871 6.309173e-10
```

检验结果显著，加拿大不是美国、英国的单向格兰格原因，即美国和英国是加拿大的格兰杰原因。

对英国进行检验：

```
GrangerTest(Z, p=2, locInput=1)
```

```
## Number of targeted zero parameters:  4
## Chi-square test for Granger Causality and p-value:  8.948851 0.06239076
## Constant term:
## Estimates:  0.2104448 0.1231581 0.2895581
## Std.Error:  0.06685632 0.07382941 0.0816888
## AR coefficient matrix
## AR( 1 )-matrix
##       [,1]  [,2]  [,3]
## [1,] 0.473 0.000 0.000
## [2,] 0.351 0.338 0.469
## [3,] 0.491 0.240 0.236
## standard error
##        [,1]  [,2]   [,3]
## [1,] 0.0899 0.000 0.0000
## [2,] 0.0949 0.100 0.0926
## [3,] 0.1050 0.111 0.1024
## AR( 2 )-matrix
##        [,1]   [,2]     [,3]
## [1,]  0.151  0.000  0.00000
## [2,] -0.191 -0.175 -0.00868
## [3,] -0.312 -0.131  0.08531
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0859 0.0000 0.0000
## [2,] 0.0939 0.0890 0.0953
## [3,] 0.1038 0.0984 0.1055
##
## Residuals cov-mtx:
##             [,1]       [,2]       [,3]
## [1,] 0.30423344 0.02654091 0.07435286
## [2,] 0.02654091 0.29158166 0.13948786
## [3,] 0.07435286 0.13948786 0.35696571
##
## det(SSE) =  0.02443371
## AIC =  -3.487791
## BIC =  -3.17102
## HQ  =  -3.359104
```

综合以上三个结果，第三个检验说明在 0.05 水平下加拿大和美国不是英国的格兰杰原因，而前两个检验说明英国是美国和加拿大的格兰杰原因，所以可以认为英国是美国和加拿大单向的格兰杰原因。第三个检验结果中还给出了约束系数等于零的模型估计结果。

24.11 预测

若 VAR(p) 模型已知，满足平稳性条件，设 $\{a_t\}$ 是独立的弱平稳时间序列。用 F_t 表示截止到 t 时刻为止的 $r_s, s \le t$ 所包含的信息，则 $E(a_t|F_{t-1}) = 0$。基于 t 时刻的信息进行超前 l 步预测，预测为

$$r_t(l) = E(r_{t+l}|F_t)$$

当 $l = 1$ 时

$$r_t(1) = \phi_0 + \Phi_1 r_t + \cdots + \Phi_p r_{t+1-p}$$

当 $l = 2$ 时

$$\begin{aligned} r_t(2) =& E(r_{t+2}|F_t) \\ =& \phi_0 + \Phi_1 E(r_{t+1}|F_t) + \Phi_2 r_t + \cdots + \Phi_p r_{t+2-p} \end{aligned}$$

若记

$$r_t(l) = \begin{cases} E(r_{t+l}|F_t), & l > 0 \\ r_{t+l}, & l \le 0 \end{cases}$$

则超前 l 步预报可以写成

$$r_t(l) = E(r_{t+l}|F_t) = \phi_0 + \sum_{j=1}^{p} \Phi_j r_t(l-j)$$

可见超前多步预测可以递推地计算。

对于满足平稳性条件的 VAR(p) 模型，可以证明

$$\lim_{l\to\infty} r_t(l) = \mu = Er_t$$

即 VAR(p) 的预测具有均值回归性。均值回归速度由特征多项式 $|P(z)|$ 的最接近单位圆的根的模与 1 的接近程度决定，根越接近单位圆，均值回归越慢。

l 步超前预测误差为

$$e_t(l) = r_{t+l} - r_t(l) = r_{t+l} - E(r_{t+l}|F_t)$$

为了研究预测误差方差阵，最好利用 VAR 的无穷阶 MA 表示。

类似一元情形，VAR(p) 模型可以写成一个无穷阶 MA 形式

$$r_t = \mu + \sum_{l=0}^{\infty} \Psi_l a_{t-l} \tag{24.20}$$

其中 $\Psi_0 = I$, $\mu = [P(1)]^{-1}$（假定 $[P(1)]^{-1}$ 存在），系数矩阵 Ψ_l 可以通过下式使得方程两边 z 的同次项相等求解：

$$(I - \Phi_1 z - \cdots - \Phi_p z^p)(I + \Psi_1 z + \Psi z^2 + \ldots) = I$$

易见

$$e_t(l) = a_{t+l} + \Psi_1 a_{t+l-1} + \cdots + \Psi_{l-1} a_{t+1}$$

从而

$$\begin{aligned} \mathrm{Var}(e_t(1)) =& \Sigma \\ \mathrm{Var}(e_t(l)) =& \Sigma + \sum_{j=1}^{l-1} \Psi_j \Sigma \Psi_j^T \\ =& \mathrm{Var}(e_t(l-1)) + \Psi_{l-1} \Sigma \Psi_{l-1}^T,\ l = 2, 3, \ldots \end{aligned}$$

当 $l \to \infty$ 时

$$\text{Var}(e_t(l)) \to \sum_{j=0}^{\infty} \Psi_j \Sigma \Psi_j^T = \text{Var}(r_t)$$

这一点与均值回归性质吻合。

如果模型是从数据中估计得到的，则点预测仍利用相同的公式，将估计参数代入即可：

$$\begin{aligned} r_t(1) =& \hat{\phi}_0 + \hat{\Phi}_1 r_t + \cdots + \hat{\Phi}_p r_{t+1-p} \\ r_t(l) =& \hat{\phi}_0 + \hat{\Phi}_1 r_t(l-1) + \cdots + \hat{\Phi}_p r_t(l-p) \end{aligned}$$

预测误差的估计还要考虑到参数估计带来的误差，比较复杂，参见 (Tsay, 2014) §2.9.2。

MTS 包的 `VARpred()` 函数可以从 VAR 的建模结果计算点预测值，不考虑参数估计误差的预测标准误差（Standard errors of predictions），考虑参数估计误差的预测标准误差（Root mean squared errors of predictions）。

例 3.5

例 3.1 中建立的三个国家的 GDP 增速的 VAR(2) 模型是基于 1980 年第二季度到 2011 年第二季度的数据，用建立的模型进行超前 1 到 8 步预测。第一个预测对应 2011 年第三季度，最后一个预测对应 2013 年第二季度。

```
library(MTS, quietly = TRUE)
VARpred(m2.gdp3r, 8)
```

```
## orig  125
## Forecasts at origin:  125
##          uk      ca     us
## [1,] 0.3129 0.05166 0.1660
## [2,] 0.2647 0.31687 0.4889
## [3,] 0.3143 0.48231 0.5205
## [4,] 0.3839 0.53053 0.5998
## [5,] 0.4412 0.56978 0.6297
## [6,] 0.4799 0.59478 0.6530
## [7,] 0.5068 0.60967 0.6630
## [8,] 0.5247 0.61689 0.6688
## Standard Errors of predictions:
##        [,1]   [,2]   [,3]
## [1,] 0.5315 0.5400 0.5975
## [2,] 0.5804 0.7165 0.7077
## [3,] 0.6202 0.7672 0.7345
## [4,] 0.6484 0.7785 0.7442
## [5,] 0.6629 0.7824 0.7475
## [6,] 0.6692 0.7838 0.7484
## [7,] 0.6719 0.7842 0.7486
## [8,] 0.6729 0.7843 0.7487
## Root mean square errors of predictions:
##        [,1]   [,2]   [,3]
## [1,] 0.5461 0.5549 0.6140
```

```
## [2,] 0.6001 0.7799 0.7499
## [3,] 0.6365 0.7879 0.7456
## [4,] 0.6601 0.7832 0.7484
## [5,] 0.6689 0.7841 0.7488
## [6,] 0.6719 0.7844 0.7488
## [7,] 0.6730 0.7844 0.7487
## [8,] 0.6734 0.7844 0.7487
```

多步预测会趋近到序列均值，计算序列均值：

```
Z <- coredata(as.xts(ts.gdp3r))
colMeans(Z)
```

```
##        uk        ca        us
## 0.5223092 0.6153672 0.6473996
```

可以看出在超前 8 步时的点预测值很接近于序列的均值。

不管是不考虑参数估计误差的标准误差 (Standard errors of predictions) 还是考虑参数估计误差的标准误差 (Root mean squared errors of predictions)，超前 l 步预报当 $l \to \infty$ 时都应该趋于序列的标准差。计算序列的样本标准差：

```
Z <- coredata(as.xts(ts.gdp3r))
apply(Z, 2, sd)
```

```
##        uk        ca        us
## 0.7086442 0.7851955 0.7872912
```

输出中的标准误差或者根均方误差可以用来计算预测区间。比如，计算美国 GDP 季度对数增长率超前 2 步预测区间，即 2011 年第四季度的预测区间，在 95% 置信度下可计算为 $0.4889 \pm 1.96 \times 0.7077$ 或 $0.4889 \pm 1.96 \times 0.7499$，后一个区间用到的标准误差包含了参数估计误差带来的额外误差估计。

24.12 脉冲响应函数

回忆一元 ARMA 模型的 Wold 表示：

$$X_t = \sum_{j=0}^{\infty} \psi_j \varepsilon_{t-j},$$

其中 $\{\varepsilon_t\}$ 是白噪声列，称为 $\{X_t\}$ 的**新息**，当 $\{\varepsilon_t\}$ 为独立白噪声时有

$$\varepsilon_t = X_t - E(X_t | X_{t-1}, X_{t-2}, \dots).$$

$\{\psi_j\}$ 称为 $\{X_t\}$ 的 **Wold 系数**，又称为 $\{X_t\}$ 的**脉冲响应函数** (IRF, impulse response function)，这里函数自变量为滞后 j。易见

$$\frac{\partial X_t}{\partial \varepsilon_{t-j}} = \psi_j,$$

所以 ψ_j 代表了过去的输入项 ε_{t-j} 的一个单位的变化对 j 期后的 X_t 的影响大小。“脉冲响应” 这个名词是假想取输入 $\{\varepsilon_t\}$ 为 $\varepsilon_0 = 1$，其它的 $\varepsilon_t = 0$ 时，输入序列是一个脉冲，这个脉冲在 $t >= 0$ 时刻的输出为 $X_t = \psi_t$，所以 $\{\psi_j, j = 0, 1, 2, \dots\}$ 序列是脉冲输入对应的输出。

前面给出了 VAR 模型的无穷阶 MA 表示(24.20)：

$$r_t = \mu + \sum_{l=0}^{\infty} \Psi_l a_{t-l}$$

Ψ_l 是过去的信息对 r_t 的影响的系数，称 Ψ_l 的元素为时间序列 r_t 的**脉冲响应函数**的系数。等价地，Ψ_l 的元素也是 a_t 对未来的 r_{t+l} 的线性影响的系数。

以 $k=2$ 的序列 $r_t = (r_{1t}, r_{2t})^T$ 为例，记 $\Psi_l = (\psi_{ij}(l))_{k\times k}$，考虑 r_{1t} 的一个变化对 $r_{2,t+j}$ 的影响。为此，设 $\mu = 0$，设 $a_t = 0 (t \le 0$ 时)，设 $a_0 = (1,0)^T$, $a_t = 0 (t \ne 0$ 时)，这样来考虑 $a_{1,0}$ 的变化引起的后续变化，也可以认为是 $r_{1,0}$ 的一个变化引起的后续变化。这时

$$\begin{aligned} r_0 =& a_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ r_1 =& \Psi_1 a_0 = \begin{pmatrix} \psi_{11}(1) \\ \psi_{21}(1) \end{pmatrix} \\ r_2 =& \Psi_2 a_0 = \begin{pmatrix} \psi_{11}(2) \\ \psi_{21}(2) \end{pmatrix} \\ & \vdots \end{aligned}$$

所以 Ψ_l 的 (i,j) 元可以看成是 r_{jt} 的一个变动对 $r_{i,t+l}$ 造成的影响。

记

$$\underline{\Psi}_n = \sum_{l=0}^{n} \Psi_l$$

$\underline{\Psi}_n$ 表示对 r_t 的单位冲击的**累积响应**，称 $\underline{\Psi}_n$ 的元素为第 n 个短期乘数，全部的响应的累积值

$$\underline{\Psi}_\infty = \sum_{l=0}^{\infty} \Psi_l$$

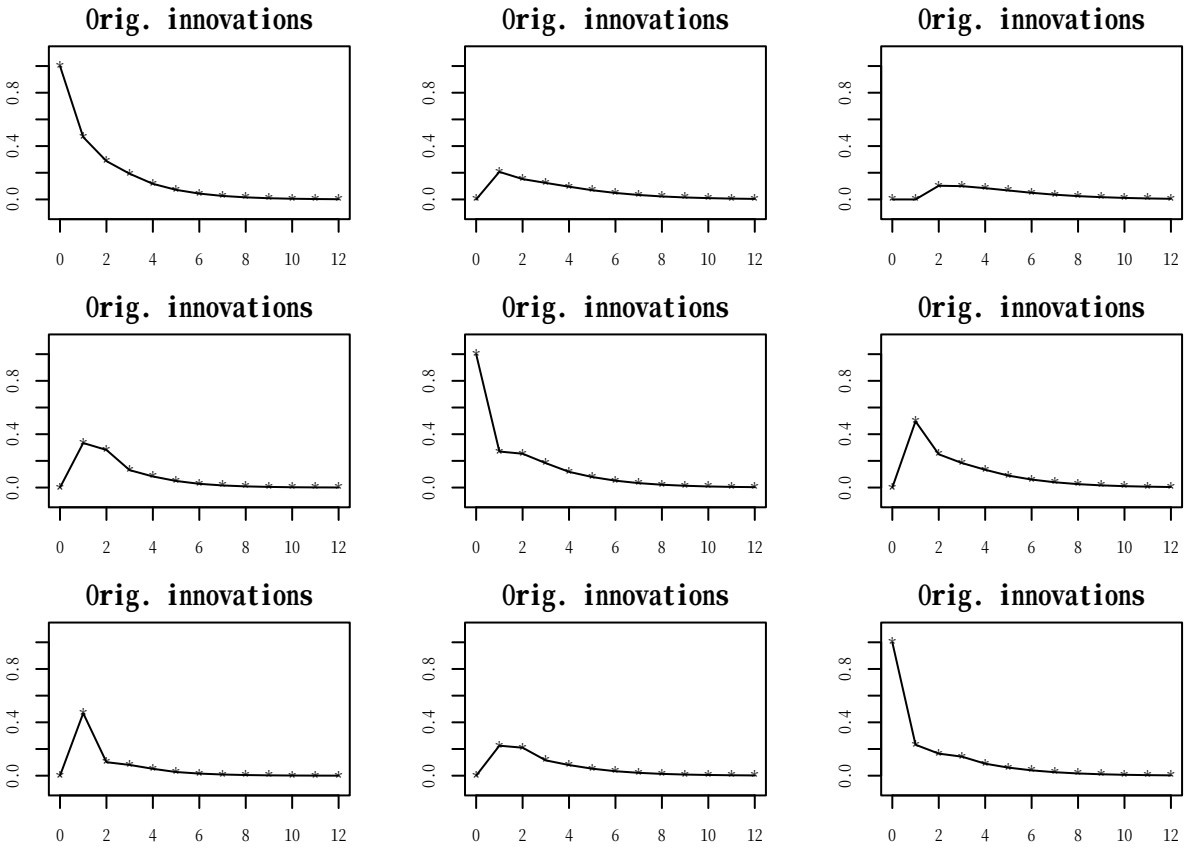
称为总乘数或长期效应。

可以作 $\psi_{ij}(l)$ 沿 $l = 0, 1, 2, \ldots$ 变化的折线图，称为脉冲响应函数图；以及 $\sum_{l=0}^{n} \psi_{ij}(l)$ 沿 $n = 0, 1, 2, \ldots$ 变化的折线图，称为累积响应函数图。

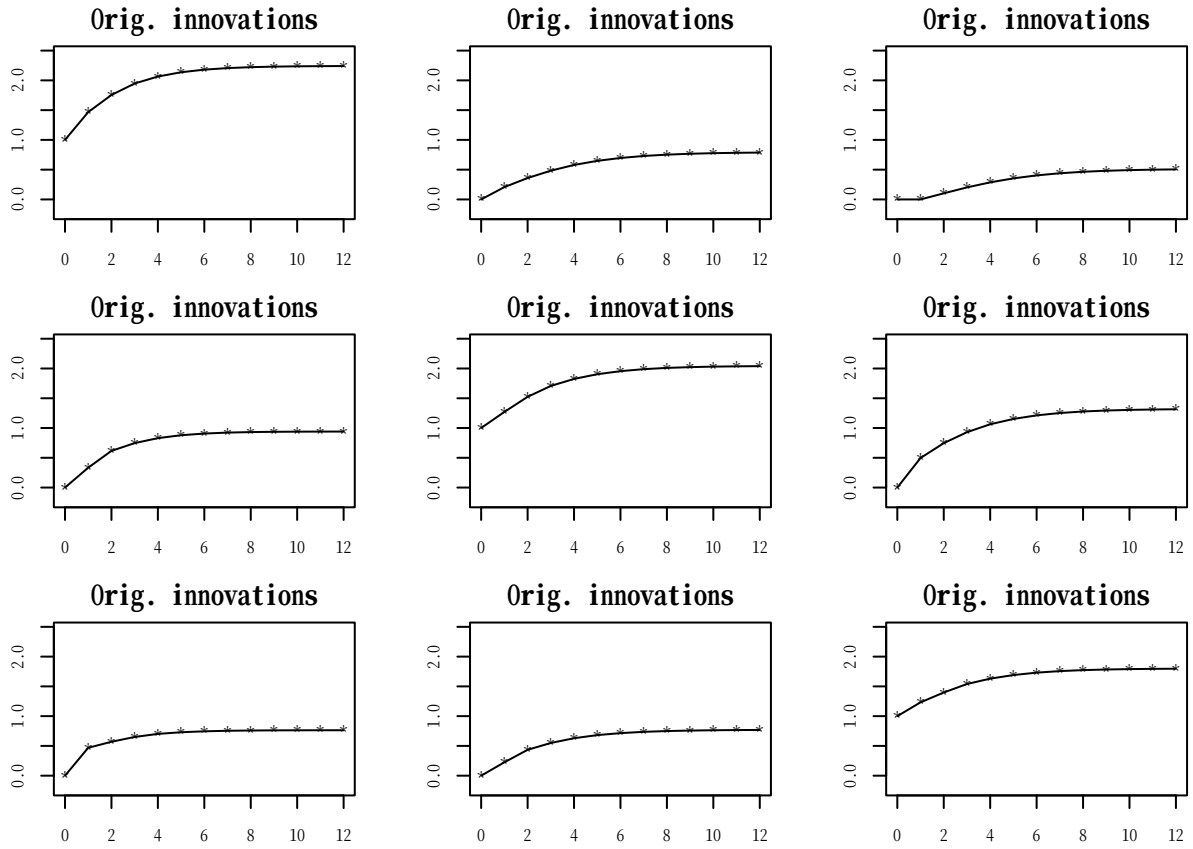
例 3.6

对例 3.2 中建立的简化的 VAR(2) 模型作脉冲响应函数图和累积响应图。三个分量两两的图形共 $3 \times 3 \times 2$ 幅。

```
library(MTS, quietly = TRUE)
VARMAirf(Phi = mods2.gdp3r$Phi,
  Sigma = mods2.gdp3r$Sig,
  orth=FALSE)
```




```
## Press return to continue
```



在脉冲响应的 9 幅图中，右上角的图是 $\psi_{13}(l)$ 的图形，代表美国（分量 3）的数据的一个冲击对英国（分量 1）的延迟的影响。

这样的脉冲响应有一些问题。由于新息 a_t 存在分量间的同步相关性，改变 a_{1t} 的值不可能不同时影响到 a_{2t} 的值。所以原始的脉冲响应 Ψ_l 不够实际。用数学解释，改变 a_{1t} 后对 r_{t+l} 的影响，可以看成是求 $\frac{\partial r_{t+l}}{\partial a_{1t}}$，利用无穷阶 MA 表示(24.20)，因为 $\{a_t\}$ 序列不相关，所以 (以 $k=2$ 为例)

$$\frac{\partial r_{t+l}}{\partial a_{1t}} = \Psi_l \frac{\partial a_t}{\partial a_{1t}} = \Psi_l \begin{pmatrix} 1 \\ \frac{\partial a_{2t}}{\partial a_{1t}} \end{pmatrix}$$

其中 a_{2t} 不是 a_{1t} 的函数，其关系可以用线性回归

$$a_{2t} = \frac{\sigma_{21}}{\sigma_{11}} a_{1t} + e$$

表示，因此 $\frac{\partial a_{2t}}{\partial a_{1t}}$ 可估计为 $\frac{\sigma_{21}}{\sigma_{11}}$，有

$$\frac{\partial r_{t+l}}{\partial a_{1t}} = \Psi_l \begin{pmatrix} 1 \\ \frac{\sigma_{21}}{\sigma_{11}} \end{pmatrix}$$

所以 a_{1t} 对 r_{t+l} 的影响不仅涉及 Ψ_l 的第一列元素的影响，还涉及 Σ 的第一列元素的影响以及 Ψ_l 的第二列元素的影响。

可以利用 Cholesky 分解使得新息正交化（分量间不相关）。对 a_t 的方差阵 Σ 有 Cholesky 分解 $\Sigma = LL^T$，其中 L 是对角线元素都为正数的下三角阵。定义正交化的新息 $b_t = L^{-1}a_t$，则 $\mathrm{Var}(b_t) = I$ 是单位阵，从而 b_t 各分量不相

关。令 $\Psi_l^* = \Psi_l L = (\psi_{ij}^*(l))_{k\times k}$，(24.20)可以用正交化的新息 b_t 表示为

$$\begin{aligned} r_t =& \mu + \sum_{l=0}^{\infty} \Psi_l L \cdot L^{-1} a_{t-l} \\ =& \mu + \sum_{l=0}^{\infty} \Psi_l^* b_{t-l} \end{aligned}$$

因为 b_t 的分量不相关，可得

$$\frac{\partial r_{i,t+l}}{\partial b_{jt}} = \psi_{ij}^*(l)$$

$\psi_{ij}^*(l)$ 是 b_{jt} 对未来的观测 $r_{i,t+l}$ 的影响，称系数矩阵 Ψ_l^* 为 r_t 的带正交新息 b_t 的脉冲响应系数，称沿 $l = 0, 1, 2, ...$ 变化的 $\psi_{ij}^*(l)$ 为 r_t 的带正交新息 b_t 的脉冲响应函数。

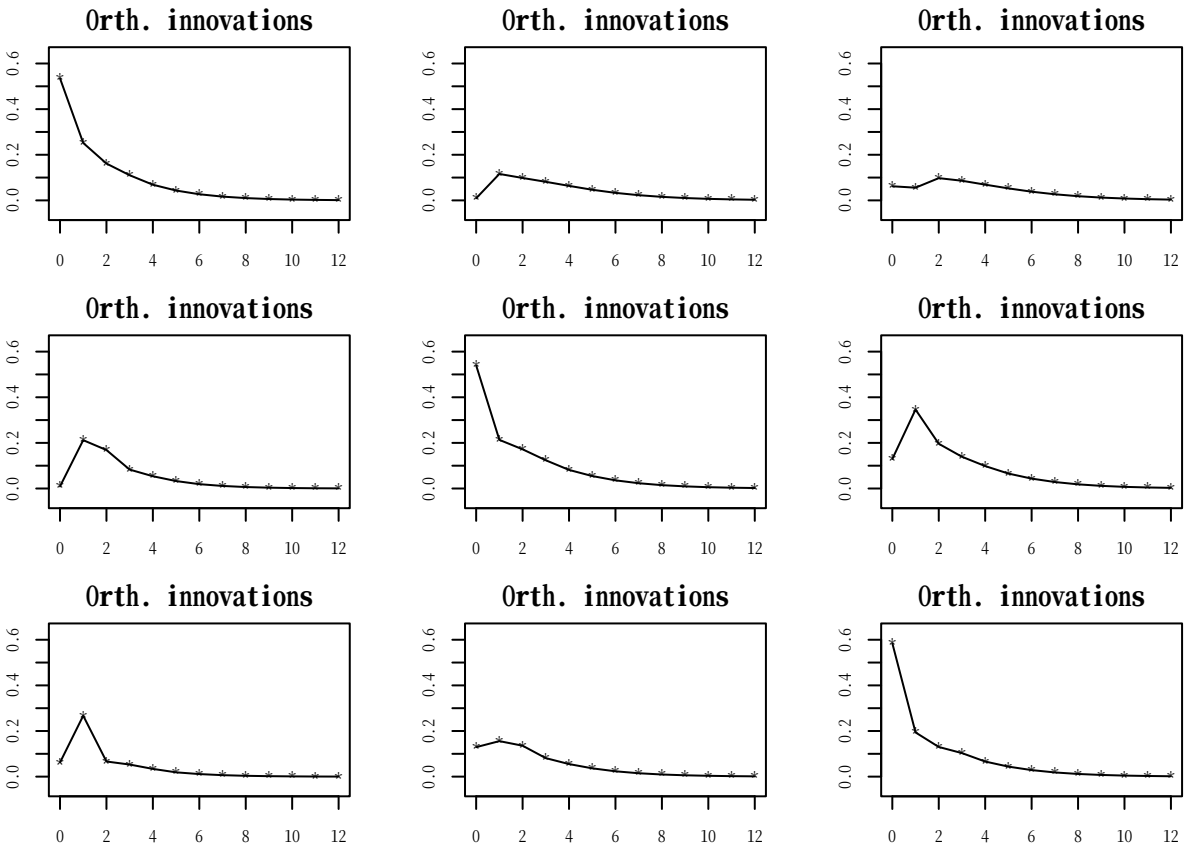
正交化以后，$\psi_{ij}^*(0)$ 是 L^{-1} 的第 (i, j) 元素，可以表示 r_{jt} 的一个冲击对 r_{it} 的即期影响。

这样的三角分解的缺点是结果依赖于 r_t 的分量次序，对脉冲函数的理解与 r_t 的分量次序有关。特别地，$a_{1t} = \sqrt{\sigma_{11}} b_{1t}$，$a_{1t}$ 仅做了方差标准化。

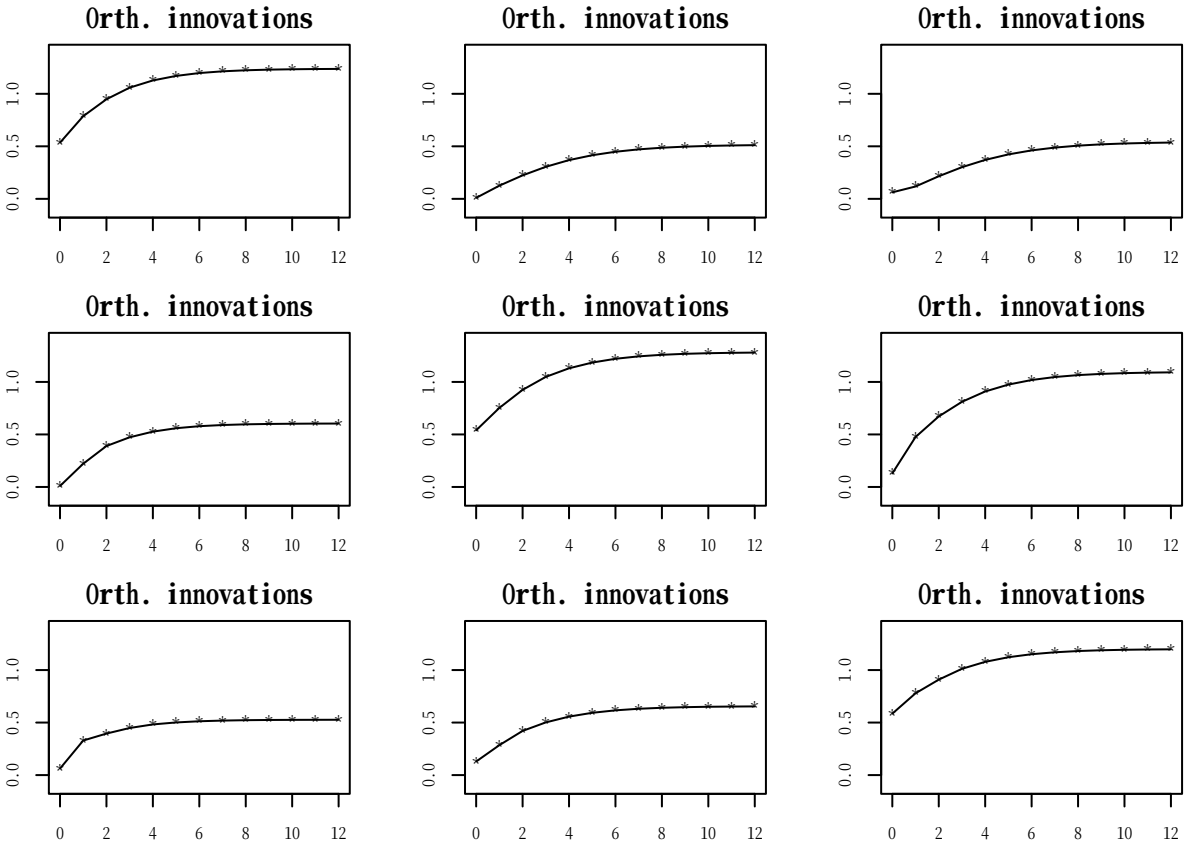
例 3.7

对例 3.2 中建立的简化的 VAR(2) 模型作正交化的脉冲响应函数图和累积响应图。

```
library(MTS, quietly = TRUE)
VARMAirf(Phi = mods2.gdp3r$Phi,
  Sigma = mods2.gdp3r$Sig,
  orth=TRUE)
```



```
## Press return to continue
```



这三个分量的新息的即期相关不大，所以正交化的脉冲响应与原始的表现相近。

24.13 GrangerTest() 函数

MTS::GrangerTest() 函数的用法文档不够明确。我们模拟一个如下的 VAR(1) 模型：

$$r_t = \begin{pmatrix} 100 \\ 200 \\ 300 \end{pmatrix} + \begin{pmatrix} 0.8 & 0 & 0 \\ 0.2 & 0.4 & 0 \\ 0.2 & 0.2 & 0.3 \end{pmatrix} r_{t-1} + \varepsilon_t,$$

其中

$$\varepsilon_t \sim \mathrm{N}\left(0, \mathrm{diag}(1,2,3)\right).$$

这时，r_{2t} 和 r_{3t} 不是 r_{1t} 的格兰杰原因；r_{3t} 不是 r_{2t} 的格兰杰原因。但是，GrangerTest() 仅能指定一个分量，检验是否其它分量都不是此分量的格兰杰原因。

```
library(mvtnorm)
library(MTS)

# 模拟一个一般的 VAR(1) 的函数
sim_VAR1 <- function(
```

```
    n=100, # 样本量
    phi_mat = matrix(0, 2, 2),  # 系数矩阵
    Q = matrix(1, 2, 2),   # 误差方差阵
    mu_vec = c(0,0)) { # 均值向量
  m <- nrow(phi_mat) # 维数
  preheat <- 100 # 为避免初值影响
  nn <- preheat + n
  x <- matrix(0, nn, m)
  eps <- rmvnorm(nn, sigma = Q)
  for(tt in 2:nn){
    x[tt,] <- phi_mat %*% x[tt-1,] + eps[tt,]
  }

  return(x[(preheat+1):nn,])
}

# 模拟前述的 3 维的 VAR(1), 系数矩阵下三角
sim11 <- function(n=500){
  phi_mat <- matrix(c(
    0.8, 0, 0,
    0.2, 0.4, 0,
    0.2, 0.2, 0.3 ),
    ncol=3, nrow=3,
    byrow=TRUE)
  Q <- diag(c(1,2,3))
  mu_vec <- c(100, 200, 300)
  sim_VAR1(n=n, phi_mat=phi_mat, Q=Q, mu_vec=mu_vec)
}

# 执行模拟
set.seed(101)
Z <- sim11(500)


cat("\n\n========== 模型拟合结果: \n")
```

```
##
##
## ========== 模型拟合结果:
```

```
VAR(Z, p=1)
```

```
## Constant term:
## Estimates:  -0.05060994 -0.04315222 0.02430918
```

```
## Std.Error:  0.04523537 0.06626087 0.07976303
## AR coefficient matrix
## AR( 1 )-matrix
##       [,1]   [,2]    [,3]
## [1,] 0.796 0.0239  0.0109
## [2,] 0.250 0.3976 -0.0392
## [3,] 0.191 0.1954  0.3244
## standard error
##        [,1]   [,2]   [,3]
## [1,] 0.0283 0.0273 0.0236
## [2,] 0.0415 0.0401 0.0346
## [3,] 0.0499 0.0482 0.0416
##
## Residuals cov-mtx:
##            [,1]        [,2]        [,3]
## [1,]  0.9827963 -0.10931049  0.10707160
## [2,] -0.1093105  2.10873290 -0.01167435
## [3,]  0.1070716 -0.01167435  3.05569891
##
## det(SSE) =  6.272251
## AIC =  1.872135
## BIC =  1.947998
## HQ  =  1.901904
```

```
cat("\n\n========== locInput=1\n")
```

```
##
##
## ========== locInput=1
```

```
# 检验分量 1 是否不受 2、3 的领先影响:
GrangerTest(Z, p=1, locInput=1)
```

```
## Number of targeted zero parameters:  2
## Chi-square test for Granger Causality and p-value:  1.062548 0.5878557
## Constant term:
## Estimates:  -0.05315053 -0.04315222 0.02430918
## Std.Error:  0.04508747 0.06626087 0.07976303
## AR coefficient matrix
## AR( 1 )-matrix
##       [,1]  [,2]    [,3]
## [1,] 0.805 0.000  0.0000
## [2,] 0.250 0.398 -0.0392
## [3,] 0.191 0.195  0.3244
## standard error
```

```
##         [,1]   [,2]   [,3]
## [1,] 0.0266 0.0000 0.0000
## [2,] 0.0415 0.0401 0.0346
## [3,] 0.0499 0.0482 0.0416
##
## Residuals cov-mtx:
##             [,1]        [,2]        [,3]
## [1,]  0.9849060 -0.10931049  0.10707160
## [2,] -0.1093105  2.10873290 -0.01167435
## [3,]  0.1070716 -0.01167435  3.05569891
##
## det(SSE) =  6.285844
## AIC =  1.8663
## BIC =  1.925305
## HQ  =  1.889453
```

```
# 结果不显著，说明 1 不受到 2、3 的领先影响

cat("\n\n========= locInput=2\n")
```

```
##
##
## ========= locInput=2
```

```
# 检验分量 2 是否不受 1、3 的领先影响：
GrangerTest(Z, p=1, locInput=2)
```

```
## Number of targeted zero parameters:  2
## Chi-square test for Granger Causality and p-value:  36.4841 1.195577e-08
```

```
# 结果显著，分量 2 受到 1、3 的领先影响。

cat("\n\n========= locInput=3\n")
```

```
##
##
## ========= locInput=3
```

```
# 检验分量 3 是否不受 1、2 的领先影响：
res <- GrangerTest(Z, p=1, locInput=3)
```

```
## Number of targeted zero parameters:  2
## Chi-square test for Granger Causality and p-value:  38.98995 3.415379e-09
```

```
# 结果显著，分量 3 受到 2、3 的领先影响。
```

说明：在 GrangerTest() 中，用 locInput= 输入一个分量序号，检验零假设：其它所有分量都不是此分量的格兰杰原因。不能进行其它的格兰杰原因的检验。

Chapter 25

协整分析与向量误差修正模型

25.1 虚假回归问题

线性回归分析是统计学的最常用的模型之一，但是，如果回归的自变量和因变量都是时间序列，回归就不满足回归分析的基本假定：模型误差项独立同分布。

比如，一元线性回归模型

$$y_t = a + bx_t + e_t,\ t = 1, 2, \dots, n,$$

需要假定 $e_1, e_2, \dots, e_t$ 不相关，零均值，方差同为 σ^2，$x_1, x_2, \dots, x_n$ 非随机，这时最小二乘估计是无偏估计。

当 $\lim_{n\to\infty} \frac{1}{n}\sum_{t=1}^{n} x_t$ 极限存在，$\frac{1}{n}\sum_{t=1}^{n}(x_t - \bar{x})^2$ 有正极限时估计相合，见定理25.1。

如果 e_t，x_t，y_t 之中有时间序列，则回归可能不相合，或者估计相合但是回归结果中的标准误差估计和假设检验有错误。下面用一些例子说明。

例 25.1. 考虑一个标准的回归模型。

设 e_t iid $\mathrm{N}(0, 50^2)$，$x_t = t$,

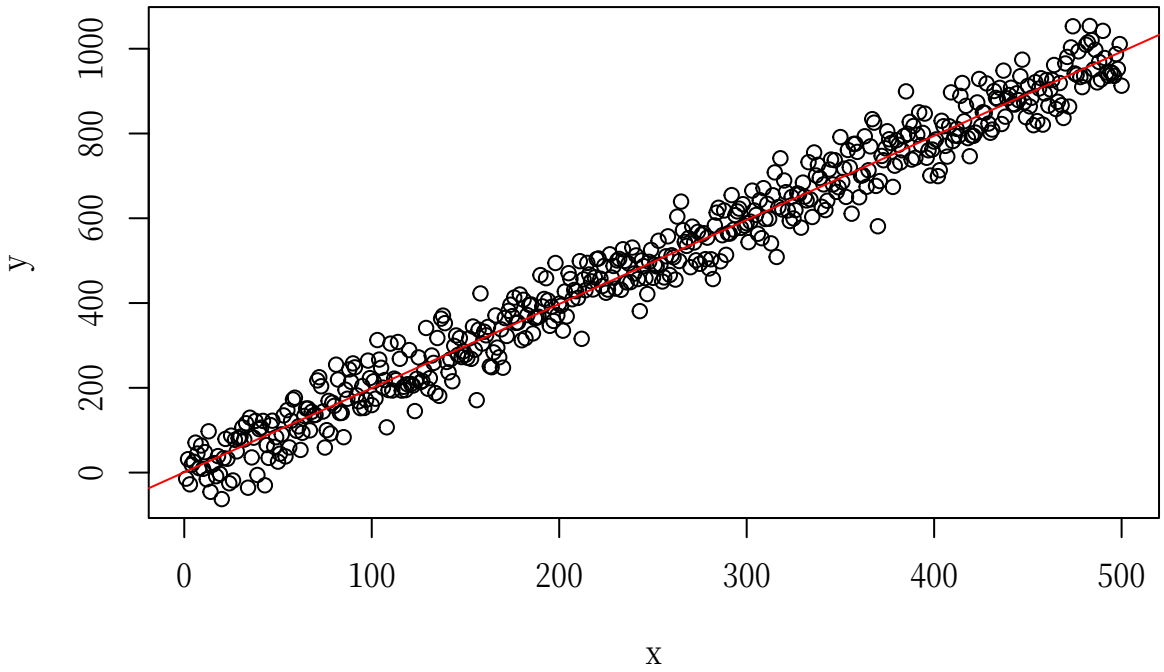
$$y_t = 2x_t + e_t,\ t = 1, 2, \dots, n.$$

这时回归是正常的，用模拟说明：

```
set.seed(101)
n <- 500
spureg01 <- function(n=500){
  x <- seq(n)
  eps <- rnorm(n)*50
  y <- 2 * x + eps
  lmr <- lm(y ~ x)
  summary(lmr) |> print()
  plot(x, y)
  abline(lmr, col="red")
```

```
  invisible(list(x=x, y=y))
}
spureg01(n)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -153.870  -33.591   -0.772   32.220  134.560
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.74231    4.32775   0.172    0.864
## x            1.98451    0.01497 132.572   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 48.31 on 498 degrees of freedom
## Multiple R-squared:  0.9724, Adjusted R-squared:  0.9724
## F-statistic: 1.758e+04 on 1 and 498 DF,  p-value: < 2.2e-16
```



例 25.2. 考虑回归误差项为 AR(1) 序列的情形。

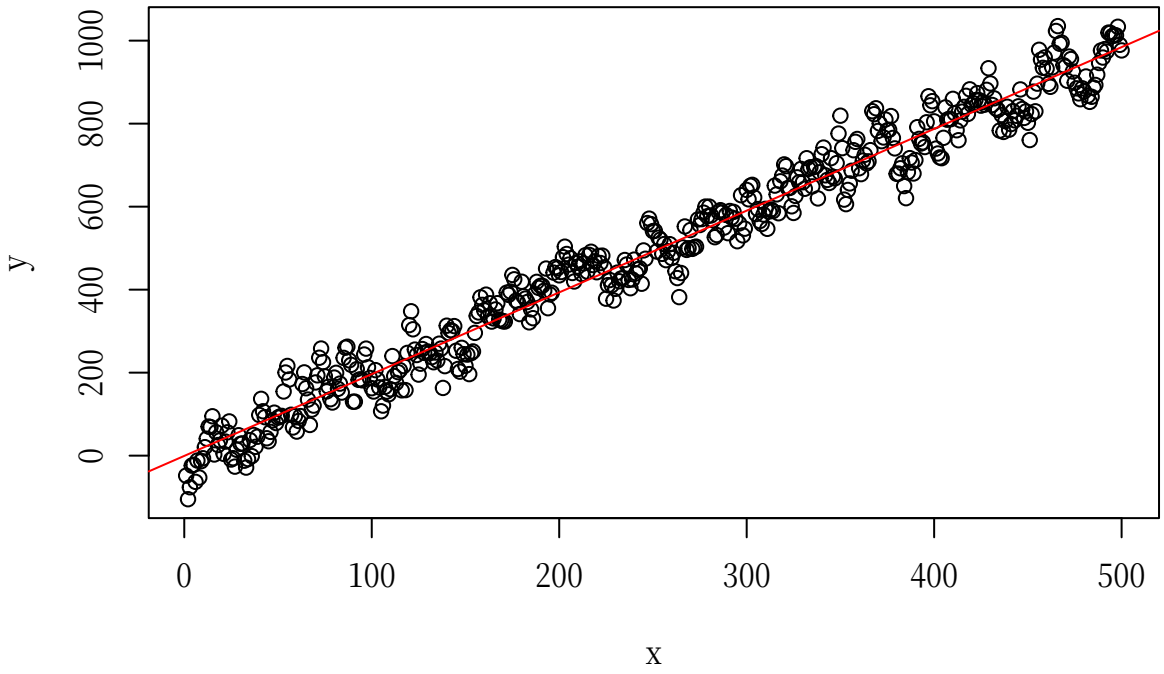
设 $x_t = t$, ε_t iid $\mathrm{N}(0, 36^2)$,

$$\begin{aligned} e_t =& 0.7e_{t-1} + \varepsilon_t, \\ y_t =& 2x_t + e_t,\ t = 1, 2, \ldots, n. \end{aligned}$$

这时回归参数估计相合，无偏，但是估计的标准误差和假设检验结果是错误的：

```
set.seed(101)
spureg02 <- function(n=500){
  x <- seq(n)
  eps <- arima.sim(
    model=list(ar=c(0.7)), n=n)*36
  y <- 2 * x + eps
  lmr <- lm(y ~ x)
  summary(lmr) |> print()
  plot(x, y)
  abline(lmr, col="red")

  invisible(list(x=x, y=y))
}
sim2 <- spureg02(n)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -137.413  -34.282    0.096   35.279  130.342
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.64699    4.36336  -0.148    0.882
## x            1.96993    0.01509 130.524   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 48.71 on 498 degrees of freedom
## Multiple R-squared:  0.9716, Adjusted R-squared:  0.9715
## F-statistic: 1.704e+04 on 1 and 498 DF,  p-value: < 2.2e-16
```



应该用同时估计回归与 ARMA 误差项的方法：

```
arima(sim2$y, order=c(1,0,0), xreg = sim2$x)
```

```
##
## Call:
## arima(x = sim2$y, order = c(1, 0, 0), xreg = sim2$x)
##
## Coefficients:
##          ar1  intercept  sim2$x
##       0.7018    -1.4932  1.9723
## s.e.  0.0317    10.2735  0.0355
##
## sigma^2 estimated as 1197:  log likelihood = -2481.72,  aic = 4971.44
```

直接用普通回归估计的 $\hat{b}$ 的标准误差为 0.015，考虑到误差项序列自相关后的标准误差估计为 0.036。有正的序列自相关时，用普通回归方法估计的斜率项的标准误差容易偏小，使得检验结果更倾向于显著。

例 25.3. 考虑回归误差项为 I(1) 序列的情形。

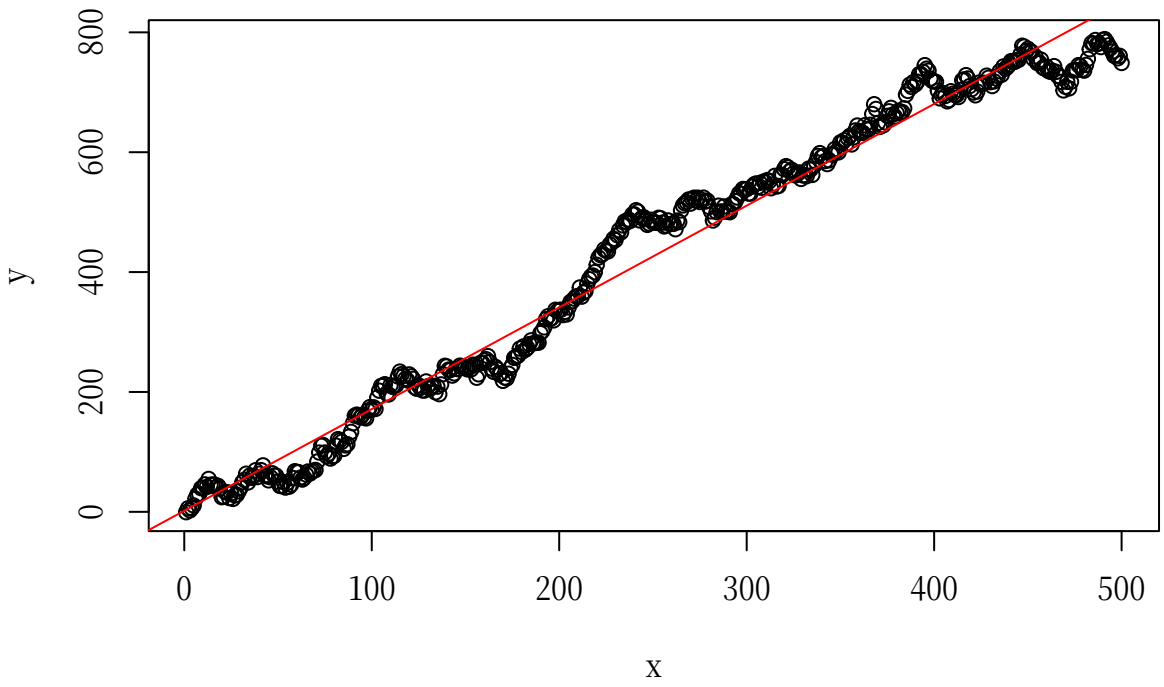
设 $x_t = t$, ε_t iid $\mathrm{N}(0, 8^2)$,

$$\begin{aligned} e_t =& e_{t-1} + \varepsilon_t, \\ y_t =& 2x_t + e_t,\ t = 1, 2, \ldots, n. \end{aligned}$$

这时回归参数估计不相合，但是用普通的回归方法估计，结果会有较大的 R^2 和系数显著性：

```
set.seed(101)
spureg03 <- function(n=500){
  x <- seq(n)
  eps <- cumsum(rnorm(n))*8
  y <- 2 * x + eps
  lmr <- lm(y ~ x)
  summary(lmr) |> print()
  plot(x, y)
  abline(lmr, col="red")

  invisible(list(x=x, y=y))
}
spureg03(n)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -101.312  -21.701    0.489   21.568   92.563
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.78482    3.31790   0.538    0.591
## x            1.69680    0.01148 147.853   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 37.04 on 498 degrees of freedom
## Multiple R-squared:  0.9777, Adjusted R-squared:  0.9777
## F-statistic: 2.186e+04 on 1 and 498 DF,  p-value: < 2.2e-16
```



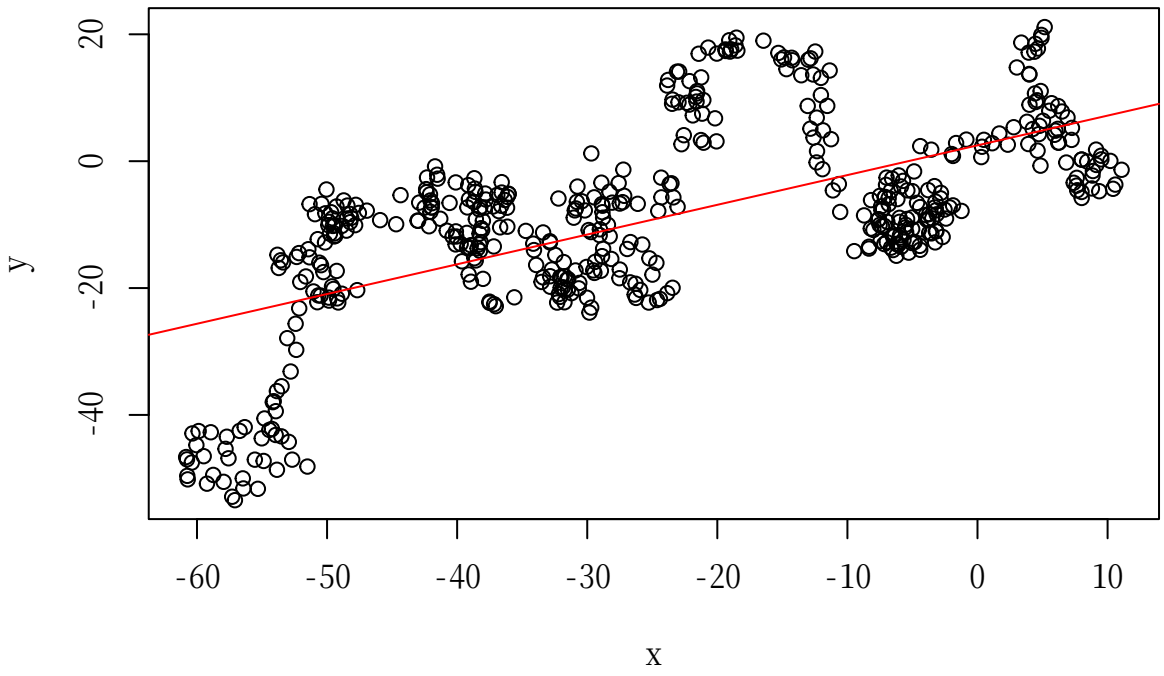
R^2 的值很大，从估计的 $\hat{b}$ 和 SE$(\hat{b})$ 看出，$\hat{b}$ 的估计显著地偏低。这说明估计已经不满足相合性。

例 25.4. 如果 x_t 和 y_t 都是 I(1) 序列但是独立，回归结果也可能很显著。

```
spureg04 <- function(){
  set.seed(110)
  n <- 500
  x <- cumsum(rnorm(n))
  y <- cumsum(rnorm(n))*2
  lmr <- lm(y ~ x)
  summary(lmr) |> print()
  plot(x, y)
  abline(lmr, col="red")

  invisible(list(x=x, y=y, lmr=lmr))
}
sim4 <- spureg04()
```

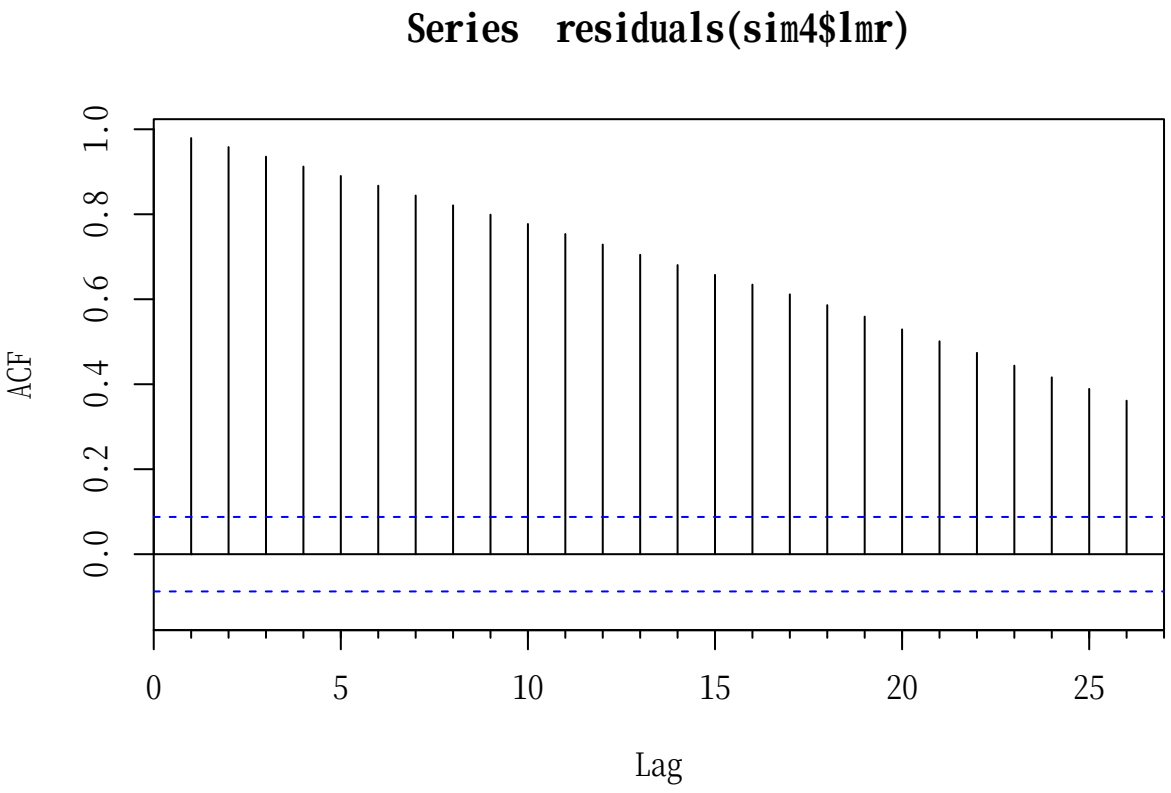
```
##
## Call:
## lm(formula = y ~ x)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -29.1795  -8.6775   0.0375   8.8982  25.6600
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.49701    0.81992   3.045  0.00245 **
## x            0.46871    0.02554  18.350  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.51 on 498 degrees of freedom
## Multiple R-squared:  0.4034, Adjusted R-squared:  0.4022
## F-statistic: 336.7 on 1 and 498 DF,  p-value: < 2.2e-16
```



检验回归残差的序列相关性和单位根：

```
forecast::Acf(residuals(sim4$lmr))
```




```
fUnitRoots::adfTest(residuals(sim4$lmr), lags=1, type="c")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -1.3595
##   P VALUE:
##     0.5525
##
## Description:
##  Tue May 30 17:04:35 2023 by user: Lenovo
```

结果显示回归残差有单位根。

这样的回归称为虚假回归。两者不应该有线性关系的。

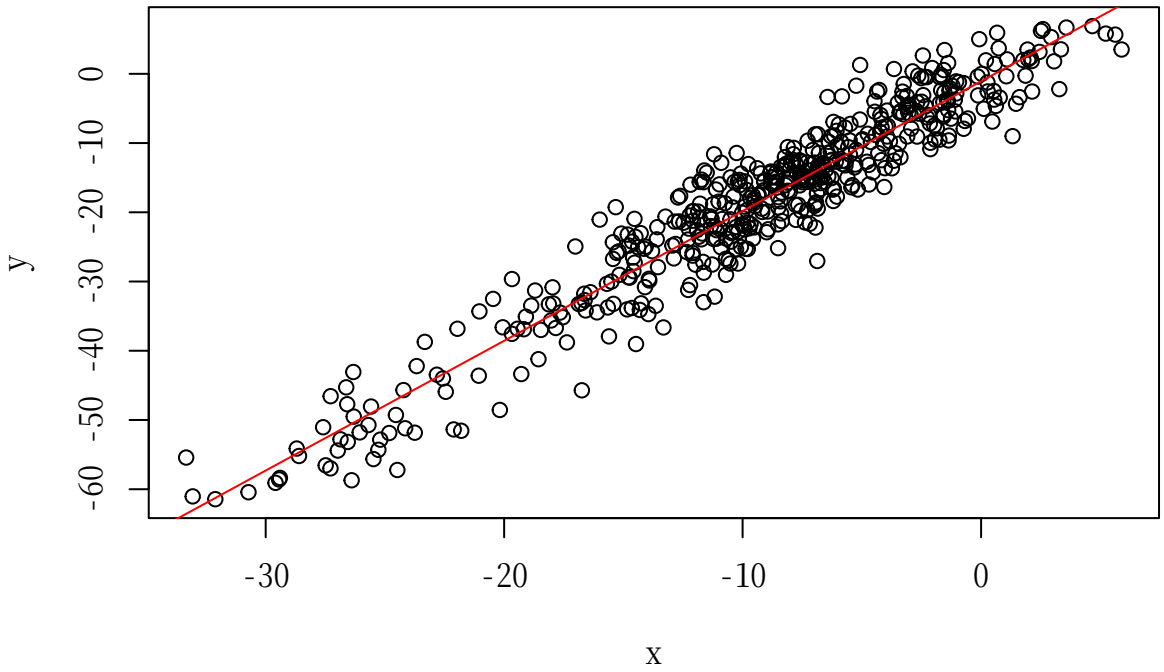
例 25.5. 如果 x_t 和 y_t 都是某个 I(1) 序列的线性变换，则回归结果是有一定意义的，称这两个序列有“协整关系”。

设 ε_t iid $\mathrm{N}(0,1)$, ξ_t iid $\mathrm{N}(0,2^2)$, η_t iid $\mathrm{N}(0,2^2)$,

$$
\begin{aligned}
z_t =& z_{t-1} + \varepsilon_t, \\
x_t =& z_t + \xi_t, \\
y_t =& 2z_t + \eta_t, \ t = 1, 2, \ldots, n.
\end{aligned}
$$

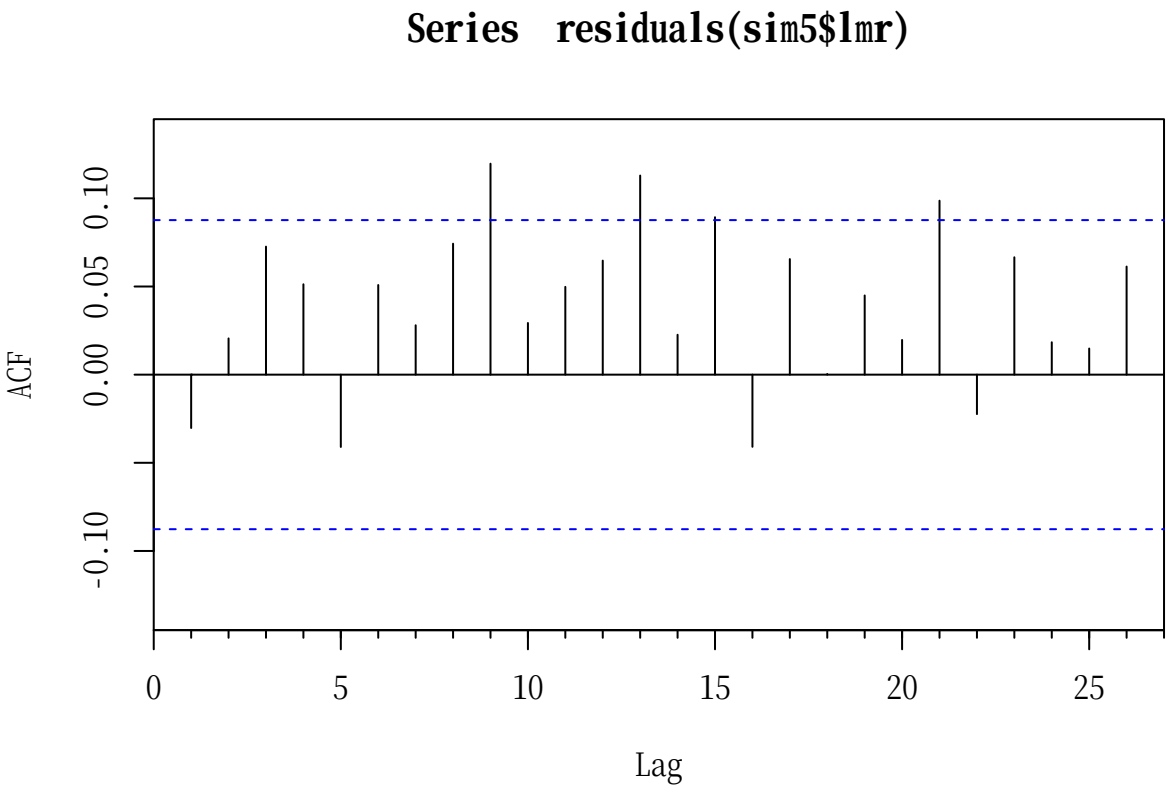
```
set.seed(101)
spureg05 <- function(n=500){
  z <- cumsum(rnorm(n))
  x <- z + rnorm(n)*2
  y <- 2*z + rnorm(n)*2
  lmr <- lm(y ~ x)
  summary(lmr) |> print()
  plot(x, y)
  abline(lmr, col="red")

  invisible(list(x=x, y=y, lmr=lmr))
}
sim5 <- spureg05(n)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2598  -2.6583   0.1401   2.6900  11.8576
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.0736     0.3024   -3.55 0.000421 ***
## x             1.8743     0.0256   73.23  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.181 on 498 degrees of freedom
## Multiple R-squared:  0.915,  Adjusted R-squared:  0.9149
## F-statistic:  5362 on 1 and 498 DF,  p-value: < 2.2e-16
```



这个回归估计的标准误差仍不可信，但是回归残差已经基本平稳：

```
forecast::Acf(residuals(sim5$lmr))
```




```
fUnitRoots::adfTest(residuals(sim5$lmr), lags=1, type="c")
```

```
## Warning in fUnitRoots::adfTest(residuals(sim5$lmr), lags = 1, type = "c"):
## p-value smaller than printed p-value

##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -15.6707
##   P VALUE:
##     0.01
##
## Description:
##  Tue May 30 17:04:35 2023 by user: Lenovo
```

结果显示回归残差没有单位根。

25.2 协整分析概念

对于二元时间序列 $x_t = (x_{1t}, x_{2t})^T$，如果 x_{1t} 和 x_{2t} 都是一元单位根过程，但存在非零线性组合 $\beta = (\beta_1, \beta_2)$ 使得 $z_t = \beta_1 x_{1t} + \beta_2 x_{2t}$ 弱平稳，则称两个分量 x_{1t} 和 x_{2t} 存在**协整**关系（cointegration），$(\beta_1, \beta_2)^T$ 称为 x_t 的**协整向量**。多个分量的多元时间序列可以类似地定义协整关系，多元时可以有多个协整向量。

25.3 Engle 和 Granger 两阶段法

考虑两个分量的多元时间序列 r_t。为了检验协整性，首先要用一元的单位根检验（如 ADF 检验）确认两个分量都是单位根过程，并且差分之后就没有单位根，这样的单位根过程称为 “单整” 的，或 I(1) 序列。

其次，将 x_{1t} 当作因变量，x_{2t} 当作自变量，作一元线性回归，得到残差 e_t 序列，和回归系数 β_1，方程为

$$x_{1t} = \beta_0 + \beta_1 x_{2t} + e_t$$

根据 (R. R. Engle & Granger, 1987) 的研究，回归在协整关系成立时参数估计相合，但是系数的估计非正态，所以用线性最小二乘估计得到的点估计可用，但是结果中的 t 检验和 F 检验结果无效。

为了验证协整关系是否成立，只要对 e_t 序列进行一元的单位根检验，但是因为 e_t 是回归残差，其自由度有变化，所以统计量 p 值的计算需要进行调整，(Phillips & Ouliaris, 1990) 给出了利用回归残差进行协整检验的方法，称为 Phillips-Ouliaris 协整检验。如果经检验 e_t 不存在单位根，则称两个分量是协整的。

这样的检验方法称为 Engle 和 Granger 两阶段法，利用 (Phillips & Ouliaris, 1990) 方法计算这个检验称为 Phillips-Ouliaris 协整检验。但是，这里的两阶段，其实第二阶段指的是在多元情况下需要找出所有的协整向量，这需要利用向量误差修正模型 (VECM)。

R 扩展包 tseries 中 `po.test()` 可以执行基于 EG 两阶段法步骤的 Phillips-Ouliaris 协整检验，零假设是非协整，对立假设是存在协整关系。参见 (Phillips & Ouliaris, 1990)。

R 扩展包 urca 也提供了 Phillips-Ouliaris 协整检验。

例 4.1

考虑 (Pfaff, 2008) 中的一个例子，数据为 urca 包的 Raotbl3，为英国经济的三个季度数据，从 1966 年第 4 季度到 1991 年第 2 季度，都经过季节调整并取了自然对数。lc 是消费，li 是收入，lw 是财富。

```
library(urca, quietly = TRUE)
data(Raotbl3)
ts.Raotbl3 <- ts(Raotbl3[,1:3], start=c(1966,4), frequency=4)
```

数据见附录。

三个序列的时间序列图：

```
library(quantmod, quietly = TRUE)
plot(as.xts(ts.Raotbl3), type="l",
     multi.panel=FALSE, theme="white",
     main=" 英国消费、收入、财富季度数据",
     major.ticks="years",
     grid.ticks.on = "years")
```

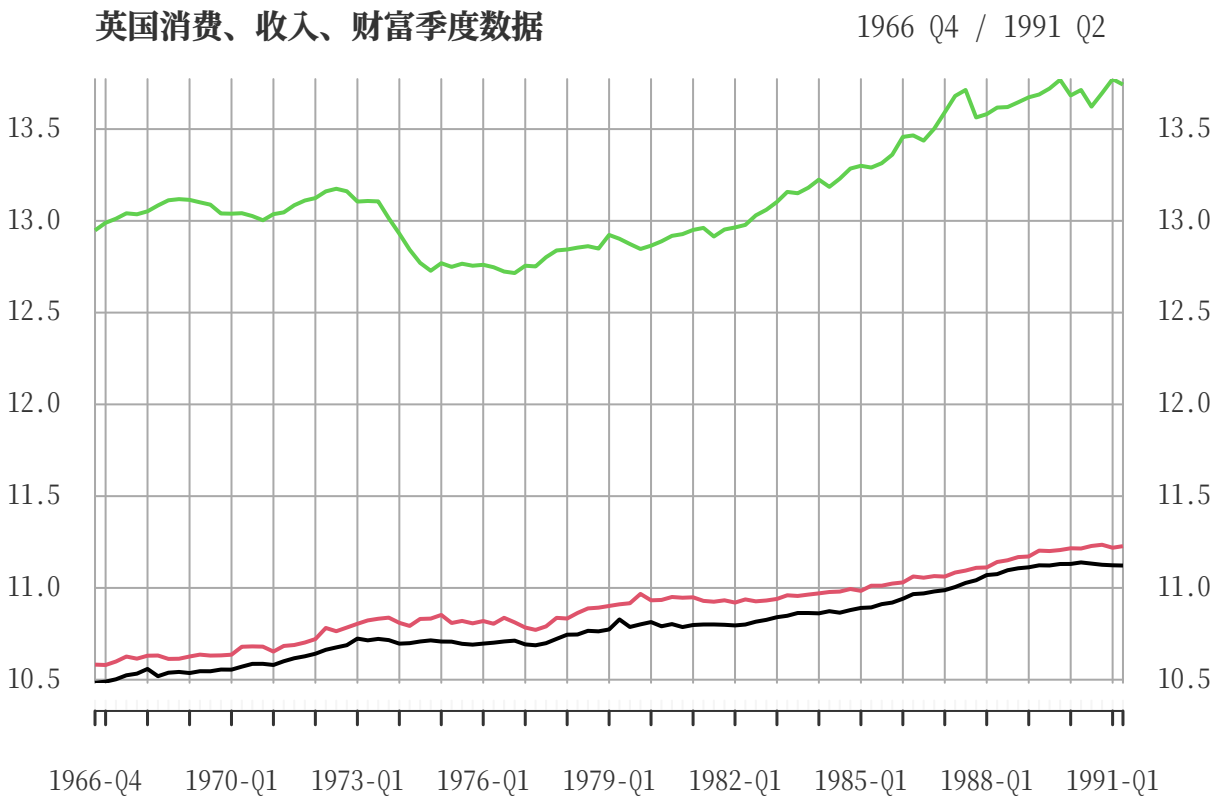


图 25.1: 英国消费、收入、财富季度数据

黑色为消费，红色为收入，绿色为财富。

作 ADF 单位根检验：

```
fUnitRoots::adfTest(Raotbl3$lc, lags=1, type="ct")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -1.378
##   P VALUE:
##     0.8341
##
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

```
fUnitRoots::adfTest(Raotbl3$li, lags=1, type="ct")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -2.0159
##   P VALUE:
##     0.57
##
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

```
fUnitRoots::adfTest(Raotbl3$lw, lags=1, type="ct")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
```

```
##   STATISTIC:
##     Dickey-Fuller: -1.0153
##   P VALUE:
##     0.9318
##
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

结果都不显著，说明每个都有单位根。检验其差分：

```
fUnitRoots::adfTest(diff(Raotbl3$lc), lags=1, type="c")
```

```
## Warning in fUnitRoots::adfTest(diff(Raotbl3$lc), lags = 1, type = "c"): p-value
## smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -6.311
##   P VALUE:
##     0.01
##
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

```
fUnitRoots::adfTest(diff(Raotbl3$li), lags=1, type="c")
```

```
## Warning in fUnitRoots::adfTest(diff(Raotbl3$li), lags = 1, type = "c"): p-value
## smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -7.0434
##   P VALUE:
##     0.01
##
```

```
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

```
fUnitRoots::adfTest(diff(Raotbl3$lw), lags=1, type="c")
```

```
## Warning in fUnitRoots::adfTest(diff(Raotbl3$lw), lags = 1, type = "c"): p-value
## smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -5.9831
##   P VALUE:
##     0.01
##
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

结果都显著，说明差分后没有单位根，三个序列都是一阶差分平稳的（I(1) 序列）。

urca 包的 `ur.df()` 函数可以检验差分的阶数，并可以对非随机趋势的类型选取进行检验。

来检验消费 lc 与收入 li 是否存在协整关系。将 lc 对 li 作一元回归：

```
eglm1 <- lm(lc ~ li, data=Raotbl3); summary(eglm1)
```

```
##
## Call:
## lm(formula = lc ~ li, data = Raotbl3)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.065949 -0.016067  0.001839  0.017844  0.057295
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.18007    0.14399  -1.251    0.214
## li           1.00731    0.01322  76.203   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02379 on 97 degrees of freedom
## Multiple R-squared:  0.9836, Adjusted R-squared:  0.9834
```

```
## F-statistic:  5807 on 1 and 97 DF,  p-value: < 2.2e-16
```

结果中的检验不可信，如果存在协整，因为回归模型是

$$\text{lc} = -0.18 + 1.01\text{li} + e_t$$

所以

$$\text{lc} - 1.01\text{li} = -0.18 + e_t$$

即 $(\text{lc}, \text{li})^T$ 的协整向量为 $(1, -1.01)$。

不考虑自由度调整问题，直接作 ADF 检验，但其 p 值是不准确的，需要使用修改过的临界值。

```
fUnitRoots::adfTest(residuals(eglm1), lags=1, type="c")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -2.6351
##   P VALUE:
##     0.09127
##
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

(Phillips & Ouliaris, 1990) 给出了利用回归残差进行协整检验的方法，包含两种方法，方差比方法和多元迹统计量方法。R 扩展包 urca 的 `ca.po()` 可以用来计算 Phillips-Ouliaris 检验。选项 `demean="constant"` 指定有确定性常数趋势，`demean="trend"` 指定有确定性线性趋势，缺省为 `demean="none"`，没有确定性趋势（无漂移）。选项 `type="Pu"` 指定使用方差比方法，选项 `type="Pz"` 指定使用多元迹方法，多元迹方法对哪个分量作为回归因变量不敏感。

检验消费与收入的协整：

```
library(urca, quietly = TRUE)
summary(ca.po(Raotbl3[-(1:2),c("lc", "li")], type="Pz", demean="constant"))
```

```
##
## ########################################
## # Phillips and Ouliaris Unit Root Test #
## ########################################
##
## Test of type Pz
## detrending of series with constant only
##
```

```
## Response lc :
##
## Call:
## lm(formula = lc ~ zr)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.047772 -0.007401  0.001410  0.008169  0.048541
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01183    0.08952   0.132    0.895
## zrlc         1.01799    0.05960  17.080   <2e-16 ***
## zrli        -0.01831    0.06074  -0.302    0.764
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01393 on 93 degrees of freedom
## Multiple R-squared:  0.9941, Adjusted R-squared:  0.994
## F-statistic:  7844 on 2 and 93 DF,  p-value: < 2.2e-16
##
##
## Response li :
##
## Call:
## lm(formula = li ~ zr)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.036705 -0.011532 -0.000549  0.008860  0.053870
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.09644    0.10642   0.906    0.367
## zrlc         0.33183    0.07085   4.683  9.6e-06 ***
## zrli         0.66298    0.07220   9.182  1.1e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01656 on 93 degrees of freedom
## Multiple R-squared:  0.9914, Adjusted R-squared:  0.9912
## F-statistic:  5334 on 2 and 93 DF,  p-value: < 2.2e-16
##
```

```
##
##
## Value of test-statistic is: 39.7042
##
## Critical values of Pz are:
##                    10pct    5pct    1pct
## critical values 47.5877 55.2202 71.9273
```

零假设是没有存在协整关系，0.05 水平的右侧临界值为 55.2，统计量值为 39.7，没有超过临界值，不拒绝零假设，认为消费与收入不存在协整关系。这也可能是 PO 方法的检验功效不足的问题。

Phillips Ouliaris 协整检验在多元时不能用来确定多个独立的协整向量，这需要利用 VECM 模型。

R 扩展包 urca 提供了多种协整检验，除了 Phillips-Ouliaris 协整检验，还可以进行 Johansen 的迹和最大特征值检验。

R 扩展包 tsDyn 提供了 Johansen 的协整检验，并可以用 AIC 或 BIC 进行秩滞后值选择。

R 扩展包 CommonTrend 可以从协整系统中提取共同的趋势，并绘图。

R 扩展包 cointReg 提供了协整的回归参数估计、推断功能。

R 扩展包 vars 提供了 VAR、VECM、结构 VAR(SVAR) 等模型的估计、检验，因果性检验和协整检验的功能。

这里主要讨论从 VAR 模型的角度定义协整。

25.4 VARMA 模型

仿照一元的 ARMA 模型，VAR 模型可以推广成 VARMA 模型，形如

$$P(B)r_t = Q(B)a_t \tag{25.1}$$

其中

$$\begin{aligned} P(z) =& I - \Phi_1 z - \cdots - \Phi_p z^p \\ Q(z) =& I + \Theta_1 z + \cdots + \Theta_q z^q \end{aligned}$$

是两个矩阵值的多项式，假设 $P(z)$ 和 $Q(z)$ 没有左公因子，否则可以消去公因子化简。

VARMA 存在同一个模型能够表示为不同参数形式的问题，所以尽可能使用 VAR，避免使用 VARMA。

如果需要模拟 VARMA 或者 VAR 模型数据，在 R 中有扩展包 dse 的 `ARMA()` 用来指定 VARMA 的系数，函数 `simulate()` 根据输入的系数生成模拟数据。如

```
library(dse, quietly = TRUE)
# 自回归部分系数矩阵
Apoly <- array(c(
  1,-0.5,0,
  0,-0.4,-0.25,
  0,-0.1,0,
  1,-0.5,0),
  c(3,2,2))
```

```
Apoly[2,,]
# 滑动平均部分系数矩阵，此处有变换
B <- matrix(c(3,0,0,2),ncol=2)*0.01
# 生成模型
var2 <- ARMA(A=Apoly, B=B)
print(var2)
# 模拟数据
varsim <- simulate(
  var2, sampleT=250,
  noise=list(w=matrix(rnorm(500),nrow=250,ncol=2)),
  rng=list(seed=c(123456)))
vardat <- matrix(varsim$output, nrow=250, ncol=2)
```

25.4.1 VARMA 模型协整关系

这里将 VAR 和 VARMA 模型略推广，不要求模型一定满足平稳性条件。如果不然，因为多元弱平稳其分量一定也是一元弱平稳的，就不能容许分量存在单位根。

考虑如下的二元 VARMA(1,1) 模型的例子。

$$\begin{pmatrix} x_{1t} \\ x_{2t} \end{pmatrix} - \begin{pmatrix} 0.5 & -1.0 \\ -0.25 & 0.5 \end{pmatrix} \begin{pmatrix} x_{1,t-1} \\ x_{2,t-1} \end{pmatrix} \tag{25.2}$$

$$= \begin{pmatrix} a_{1t} \\ a_{2t} \end{pmatrix} + \begin{pmatrix} 0.2 & -0.4 \\ -0.1 & 0.2 \end{pmatrix} \begin{pmatrix} a_{1,t-1} \\ a_{2,t-1} \end{pmatrix} \tag{25.3}$$

其中的扰动 a_t 的协方差阵 Σ 为正定阵。

这个模型不是弱平稳的，因为 $|P(z)| = 1 - z$ 有一个单位根。

将模型改写为

$$\begin{pmatrix} 1-0.5B & B \\ 0.25B & 1-0.5B \end{pmatrix} \begin{pmatrix} x_{1t} \\ x_{2t} \end{pmatrix} = \begin{pmatrix} 1-0.2B & 0.4B \\ 0.1B & 1-0.2B \end{pmatrix} \begin{pmatrix} a_{1t} \\ a_{2t} \end{pmatrix}$$

利用伴随矩阵，因为 $A \cdot \mathrm{adj}(A) = |A|I$，模型两边都左乘 $P(B)$ 的伴随矩阵

$$\begin{pmatrix} 1-0.5B & -B \\ -0.25B & 1-0.5B \end{pmatrix}$$

而 $|P(B)| = 1 - B$，模型转化成

$$\begin{pmatrix} 1-B & 0 \\ 0 & 1-B \end{pmatrix} \begin{pmatrix} x_{1t} \\ x_{2t} \end{pmatrix} = \begin{pmatrix} 1-0.7B & -0.6B \\ -0.15B & 1-0.7B \end{pmatrix} \begin{pmatrix} a_{1t} \\ a_{2t} \end{pmatrix}$$

即

$$\begin{aligned} x_{1t} - x_{1,t-1} &= a_{1t} - 0.7a_{1,t-1} - 0.6a_{2,t-1} \\ x_{2t} - x_{2,t-1} &= a_{2t} - 0.7a_{2,t-1} - 0.15a_{1,t-1} \end{aligned}$$

这两个模型左边是一阶差分，右边是 MA(1) 序列，所以 x_{1t} 和 x_{2t} 都是一元单位根过程。这两个分量并不独立，因为模型右边包含共同的扰动成分。

设法对两个分量进行线性变换，使得单位根只出现在一个分量中。令

$$
\begin{aligned}
L &= \begin{pmatrix} 1.0 & -2.0 \\ 0.5 & 1.0 \end{pmatrix} \\
y_t &= \begin{pmatrix} y_{1t} \\ y_{2t} \end{pmatrix} = L \begin{pmatrix} x_{1t} \\ x_{2t} \end{pmatrix} \\
b_t &= La_t
\end{aligned}
$$

原来的 x_t 模型可以写成

$$x_t = \Phi x_{t-1} + a_t + \Theta a_{t-1}$$

两边左乘 L 得到关于 y_t 的模型：

$$
\begin{aligned}
y_t =& Lx_t \\
=& L\Phi x_{t-1} + La_t + L\Theta a_{t-1} \\
=& L\Phi L^{-1} y_{t-1} + b_t + L\Theta L^{-1} b_{t-1}
\end{aligned}
$$

计算其中的系数矩阵，得到 x_t 线性变换得到的 y_t 序列的模型为

$$
\begin{aligned}
&\begin{pmatrix} y_{1t} \\ y_{2t} \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_{1,t-1} \\ y_{2,t-1} \end{pmatrix} \\
=& \begin{pmatrix} b_{1t} \\ b_{2t} \end{pmatrix} + \begin{pmatrix} -0.4 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} b_{1,t-1} \\ b_{2,t-1} \end{pmatrix}
\end{aligned}
$$

即

$$
\begin{aligned}
y_{1t} - y_{1,t-1} =& b_{1t} - 0.4 b_{1,t-1} \\
y_{2t} =& b_{2t}
\end{aligned}
$$

其中

$$
\begin{aligned}
y_{1t} =& x_{1t} - 2x_{2t} \\
y_{2t} =& 0.5x_{1t} + x_{2t}
\end{aligned}
$$

y_{1t} 仍是一元单位根过程，但 y_{2t} 已经是弱平稳列，所以 x_{1t} 和 x_{2t} 之间存在协整关系。

一般地，对于 k 元时间序列 x_t 建立了 VAR 或者 VARMA 模型后，如果分量都是一元单位根过程，但 AR 部分的特征多项式 $|P(z)|$ 的根 1 的重数小于 k，分量间就存在协整关系。如果特征多项式的根 1 的重数为 $h < k$，称 $k-h$ 为**协整因子个数**，这是能找到的线性组合后平稳的线性独立的线性组合系数的个数。这种线性组合称为**协整向量**。在上面的例子中因为 y_{2t} 平稳所以 $(0.5, 1)^T$ 是一个协整向量。

25.5 误差修正模型与协整

25.5.1 误差修正模型

因为在协整系统中，单位根非平稳分量的个数多于单位根的个数（通过线性组合可以使得单位根非平稳的分量减少），所以如果对每个单位根非平稳分量计算差分，虽然使得分量都平稳了，但是会造成过度差分，使得部分分量的 ARMA 模型的 MA 部分有单位根，这样的模型平稳但不可逆，不可逆的模型在估计和预测上比较困难。

(R. R. Engle & Granger, 1987) 讨论了协整系统的误差修正表示，称为向量误差修正模型 (VECM, vector error correction model)。

以(25.3)的协整系统为例。令 $\Delta x_t = x_t - x_{t-1}$，模型可以化为

$$\begin{aligned}\Delta x_t =& x_t - x_{t-1} = \begin{pmatrix} \Delta x_{1t} \\ \Delta x_{2t} \end{pmatrix} \\ =& \begin{pmatrix} -0.5 & -1 \\ -0.25 & -0.5 \end{pmatrix} x_{t-1} + a_t + \begin{pmatrix} -0.2 & 0.4 \\ 0.1 & -0.2 \end{pmatrix} a_{t-1} \\ =& \begin{pmatrix} -1 \\ -0.5 \end{pmatrix} \begin{pmatrix} 0.5 & 1.0 \end{pmatrix} x_{t-1} + a_t + \begin{pmatrix} -0.2 & 0.4 \\ 0.1 & -0.2 \end{pmatrix} a_{t-1}\end{aligned}$$

注意到 $y_{2t} = (0.5, 1)x_{t-1}$ 是平稳的，所以上式两边都是平稳的，MA 部分没有出现单位根，不引入不可逆性。上式描述了平稳序列。这样的表示称为 VECM。

此表示可以推广到一般的 VARMA 模型。对(25.1)的 VARMA 模型，如果其中有 m 个协整因子，$m < k$，单位根的个数大于 m，必存在如下的误差修正形式 (VECM):

$$\Delta x_t = \alpha\beta^T x_{t-1} + \sum_{j=1}^{p-1} \Phi_j^* \Delta x_{t-j} + a_t + \sum_{j=1}^{q} \Theta_j a_{t-j}$$

其中 α 和 β 都是 $k \times m$ 列满秩矩阵，MA 部分没有单位根，m 维时间序列 $y_t = \beta^T x_t$ 是平稳列（没有单位根），β 的每一列都是 x_t 的一个协整系数。Φ_j^* 和 α, β 都依赖于原来的 AR 部分的系数矩阵 Φ_j，关系为：

$$\begin{aligned}\Phi_j^* =& -\sum_{i=j+1}^{p} \Phi_i, \ j = 1, 2, \ldots, p-1 \\ \alpha\beta^T =& \Phi_p + \cdots + \Phi_1 - I = -P(1)\end{aligned}$$

可以对 $-P(1)$ 作奇异值分解得到 α 和 β，α 和 β 并不唯一。当模型参数从样本估计时，不会有严格的单位根，也不能通过矩阵秩得到 $-P(1) = \alpha\beta^T$ 这样的分解，需要进行检验。

25.5.2 VAR 模型的误差修正形式

为简单起见仅考虑 VAR 模型（不要求平稳性条件）。VAR 模型的估计与解释都更容易。

考虑 x_t 的如下的带有非随机线性趋势项的 k 维 VAR(p) 模型

$$x_t = \mu_t + \Phi_1 x_{t-1} + \cdots + \Phi_p x_{t-p} + a_t \tag{25.4}$$

其中 $a_t \sim \mathrm{N}_k(0, \Sigma)$，$\mu_t = \mu_0 + \mu_1 t$。

记

$$P(z) = I - \Phi_1 z - \cdots - \Phi_p z^p$$

当用行列式 $|P(z)|$ 表示 z 的多项式的所有根都在复平面的单位圆外时 $\{x_t\}$ 是弱平稳的（简称平稳的），在讨论单位根和协整问题时记弱平稳列为 I(0) 序列。

为简单起见，设 x_t 的所有分量至多需要一次差分就可以变平稳，即至多为 I(1) 序列。这就是说，如果分量 x_{it} 不平稳，$\Delta x_{it} = x_{it} - x_{i,t-1}$ 就一定是平稳的。这也是实际的大部分序列都能满足的。

上面的 VAR(p) 模型有如下的误差修正形式（VECM）

$$\Delta x_t = \mu_t + \Pi x_{t-1} + \Phi_1^* \Delta x_{t-1} + \cdots + \Phi_{p-1}^* \Delta x_{t-p+1} + a_t \tag{25.5}$$

其中

$$\Phi_j^* = -\sum_{i=j+1}^{p} \Phi_i,\ j = 1, 2, \ldots, p-1$$
$$\Pi = \alpha\beta^T = -P(1)$$

α 和 β 是满秩的 $k \times m$ 矩阵，m 是协整向量个数。容易看出 Φ_j 可以从 VECM 的参数中递推地反解为

$$\Phi_1 = I + \Pi + \Phi_1^*$$
$$\Phi_j = \Phi_j^* - \Phi_{j-1}^*,\ j = 2, \ldots, p,\ \Phi_p^* = 0$$

因为假定每个分量至多 $I(1)$ 所以 VECM 式(25.5)中的 $\{\Delta x_t\}$ 序列是 I(0) 序列。

如果 x_t 有单位根，则 $|P(1)| = 0$，从而 $\Pi = -P(1)$ 不满秩。根据 Π 的秩，将(25.5)的 VECM 分为如下三种情况：

情况 1：$\text{rank}(\Pi) = 0$。这时 $\Pi = 0$，x_t 有 k 个单位根，不存在协整关系，x_t 不是协整的，(25.5)的 VECM 变成了如下的关于 Δx_t 的 VAR$(p-1)$ 模型

$$\Delta x_t = \mu_t + \Phi_1^* \Delta x_{t-1} + \cdots + \Phi_{p-1}^* \Delta x_{t-p+1} + a_t$$

只要关于差分序列建模即可。

情况 2：$\text{rank}(\Pi) = k$ 满秩。这时 $|P(1)| \neq 0$，没有单位根，x_t 本身是平稳列，只要直接关于 x_t 建立平稳的 VAR 模型即可。

情况 3：$0 < m = \text{rank}(\Pi) < k$，这时 x_t 存在协整关系。下面主要考虑情况 3。

Π 可以分解为

$$\Pi_{k\times k} = \alpha\beta^T$$

其中 α 和 β 都是 $k \times m$ 的列满秩矩阵，β 的每一列都是 x_t 的一个协整向量，m 元时间序列 $\omega_t = \beta^T x_t$ 是 I(0) 序列。α 称为载荷矩阵 (loadings matrix)。x_t 有 $k-m$ 个单位根，这些单位根给出了 x_t 的 $k-m$ 个共同趋势，相当于自由变化的趋势；而 m 个协整分量 ω_t，则可以看成是 x_t 的 k 个分量的 m 个线性约束，或长期线性依赖关系。本身就是 I(0) 的分量可以计入这 m 个当中。(25.5)的 VECM 可以写成

$$\Delta x_t = \mu_t + \alpha\beta^T x_{t-1} + \Phi_1^* \Delta x_{t-1} + \cdots + \Phi_{p-1}^* \Delta x_{t-p+1} + a_t$$

记 $m = \text{rank}(\Pi)$。在 $0 < m < k$ 时，得到这 $k-m$ 个公共趋势的方法是先计算 α 的正交补矩阵 $\alpha_\perp$，$\alpha_\perp$ 是 $k \times (k-m)$ 矩阵，$(\alpha, \alpha_\perp)$ 为满秩 k 阶方阵，且 $\alpha_\perp^T \alpha = 0$。令 $y_t = \alpha_\perp^T x_t$, 则 y_t 是 $m-k$ 元的 I(1) 序列，y_t 不再有协整关系，在(25.5)的 VECM 两边同时左乘 $\alpha_\perp^T$ 可以去掉 Πx_{t-1} 项，变成了

$$\Delta y_t = \alpha_\perp^T \mu_t + \alpha_\perp^T \Phi_1^* \Delta x_{t-1} + \cdots + \alpha_\perp^T \Phi_{p-1}^* \Delta x_{t-p+1} + \alpha_\perp^T a_t$$

将 Δy_t 表示成了右侧的 $k-m$ 元平稳列的形式。

容易看出 $\Pi = \alpha\beta^T$ 的分解是不唯一的，对任意 m 阶正交阵 Ω，都有

$$\alpha\beta^T = \alpha\Omega\Omega^T\beta^T = (\alpha\Omega)(\beta\Omega)^T = \alpha_*\beta_*^T$$

其中 α_* 和 β_* 也都是 $k \times m$ 列满秩矩阵。为此增加约束条件，要求

$$\beta = \begin{pmatrix} I_m \\ \beta_1 \end{pmatrix}$$

β_1 为 $(k-m)\times m$ 矩阵。将 x_t 分解为

$$x_t = \begin{pmatrix} u_t \\ v_t \end{pmatrix}$$

u_t 为 m 元，v_t 为 $k-m$ 元，则 $\beta^T x_t = u_t + \beta_1 v_t$。实际中这可能需要将 x_t 的各分量重新排序使得有单位根的分量（至少有 m 个）都排在前面，将没有单位根的分量都排在后面。

为了使得 $\omega_t = \beta^T x_t$ 平稳，还需要其它的参数条件，就像没有协整的 I(0) 的 VAR(1) 需要平稳性条件那样。比如，考虑有一个协整向量的二元 VAR(1) 模型，这里 $k=2$，$m=1$，两个分量都为单位根过程，VECM 为

$$\Delta x_t = \mu_t + \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \begin{pmatrix} 1 & \beta_1 \end{pmatrix} x_{t-1} + a_t$$

两边同时左乘 β^T，注意 $\omega_t = \beta^T x_t$ 则 $\Delta\omega_t = \beta^T \Delta x_t$，所以

$$\Delta\omega_t = \beta^T \mu_t + (\alpha_1 + \alpha_2\beta_1)\omega_{t-1} + \beta^T a_t$$

记 $b_t = \beta^T a_t$，则 $\{b_t\}$ 为白噪声列，将方程左边的 ω_{t-1} 移项到右边，得

$$\omega_t = \beta^T \mu_t + (1 + \alpha_1 + \alpha_2\beta_1)\omega_{t-1} + b_t$$

为使得模型平稳需要 α 和 β 满足

$$|1 + \alpha_1 + \alpha_2\beta_1| < 1.$$

因为协整的条件是 $0 < \text{rank}(\Pi) < k$，所以从数据中估计得到 VECM 模型后对协整的检验就是对 rank(Π) 的检验。(Johansen, 1988), (Johansen, 1995), Reinsel 和 Ahn(1992) 正是这样进行协整检验的。

25.5.3 关于 VECM 中的非随机趋势

为了基于 VECM 模型进行协整检验，需要规定 μ_t 的形式，μ_t 的形式会影响到检验统计量的极限分布。

设 x_t 的所有分量都为 I(1) 序列。文献中已有的 μ_t 的形式包括：

（1）$\mu_t = 0$。这时 x_t 的分量为不带漂移的 I(1) 过程，平稳序列 $\omega_t = \beta^T x_t$ 均值为零。

（2）$\mu_t = \mu_0 = \alpha c_0$，即 μ_t 是有特殊形式的非零常数。VECM 变成

$$\Delta x_t = \alpha(\beta^T x_{t-1} + c_0) + \Phi_1^* \Delta x_{t-1} + \cdots + \Phi_{p-1}^* \Delta x_{t-p+1} + a_t$$

这时 x_t 的分量是不带漂移的 I(1) 过程，但是平稳列 ω_t 的均值为非零的 $-c_0$。

（3）$\mu_t = \mu_0 \neq 0$，即 μ_t 是不带特殊结构的非零常数。这时 x_t 的分量是带有漂移项 μ_0 的 I(1) 序列，且 ω_t 非零均值。

（4）$\mu_t = \mu_0 + \alpha c_1 t$，即 μ_t 是线性项 t 的系数有特殊结构的线性函数，VECM 变成

$$\Delta x_t = \mu_0 + \alpha(\beta^T x_{t-1} + c_1 t) + \Phi_1^* \Delta x_{t-1} + \cdots + \Phi_{p-1}^* \Delta x_{t-p+1} + a_t$$

这时 x_t 的分量是带有漂移项 μ_0 的 I(1) 过程，且 ω_t 有非随机的 $c_1 t$ 线性趋势项。

（5）$\mu_t = \mu_0 + \mu_1 t$，μ_0, μ_1 非零。这是没有约束的线性趋势。这时 x_t 的分量都是带有二次时间趋势的 I(1) 过程，ω_t 是线性趋势项加平稳列。

最后一种情况不常见；第一种情况在经济序列中也不常见。第三种情况在许多对数价格序列中常见。

25.5.4 VECM 的最大似然估计

设 x_t 有观测数据 $\{x_t, t=1,2,\dots,T\}$。记 $\mu_t = \mu d_t$，其中 $d_t = (1,t)^T$。设 Π 的秩为 m，$0 < m < k$。VECM 为

$$\Delta x_t = \mu d_t + \alpha\beta^T x_{t-1} + \Phi_1^* \Delta x_{t-1} + \cdots + \Phi_{p-1}^* \Delta x_{t-p+1} + a_t \tag{25.6}$$

为了求各个参数的最大似然估计，首先求解如下的多对多的线性回归问题

$$\begin{aligned}\Delta x_t &= \gamma_0 d_t + \Omega_1 \Delta x_{t-1} + \cdots + \Omega_{p-1}\Delta x_{t-p+1} + u_t \\ x_{t-1} &= \gamma_1 d_t + \Xi_1 \Delta x_{t-1} + \cdots + \Xi_{p-1}\Delta x_{t-p+1} + v_t\end{aligned}$$

令 $\hat{u}_t$ 和 $\hat{v}_t$ 为最小二乘估计的残差，定义如下的样本协方差阵

$$\begin{aligned}S_{00} &= \frac{1}{T-p}\sum_{t=p+1}^{T} \hat{u}_t \hat{u}_t^T, \\ S_{01} &= \frac{1}{T-p}\sum_{t=p+1}^{T} \hat{u}_t \hat{v}_t^T, \quad S_{10} = S_{01}^T, \\ S_{11} &= \frac{1}{T-p}\sum_{t=p+1}^{T} \hat{v}_t \hat{v}_t^T\end{aligned}$$

然后，计算矩阵 $S_{10}S_{00}^{-1}S_{01}$ 关于 S_{11} 的广义特征值和特征向量，即求 λ 和 ξ 使得

$$S_{10}S_{00}^{-1}S_{01}\xi = \lambda S_{11}\xi$$

设 k 个特征值、特征向量为 $(\hat{\lambda}_i, e_i)$，其中 $\hat{\lambda}_1 > \hat{\lambda}_2 > \cdots > \hat{\lambda}_k$，记 $e = (e_1, e_2, \dots, e_k)$，标准化使得 $e^T S_{11} e = I_k$。

协整向量组成的矩阵 β 的未标准化的最大似然估计为 $\hat{\beta} = (e_1, \dots, e_m)$。由此可以得到 β 的满足可辨识条件和标准化条件的 MLE，将得到的估计记为 $\hat{\beta}_c$，其中下标 c 表示需要满足一定的限制条件。

其它参数的 MLE 可以从下式用最小二乘方法得到：

$$\Delta x_t = \mu d_t + \alpha\hat{\beta}_c^T x_{t-1} + \Phi_1^* \Delta x_{t-1} + \cdots + \Phi_{p-1}^* \Delta x_{t-p+1} + a_t$$

基于 m 个协整向量的似然函数的最大值满足

$$L_{\max}^{-2/T} \propto |S_{00}| \prod_{i=1}^{m}(1-\hat{\lambda}_i)$$

在计算关于 $\text{rank}(\Pi) = m$ 的似然比统计量时需要用到这个似然函数最大值。

α 和 β 的正交补可以用如下公式计算：

$$\begin{aligned}\hat{\alpha}_\perp &= S_{00}^{-1} S_{11}(e_{m+1}, \dots, e_k) \\ \hat{\beta}_\perp &= S_{11}(e_{m+1}, \dots, e_k)\end{aligned}$$

25.5.5 基于 VECM 最大似然估计的 Johansen 协整检验

在给定了 μ_t 形式后，基于 VECM 的最大似然估计对 $\text{rank}(\Pi)$ 进行系列的似然比检验。记 $H(m)$ 表示零假设 $\text{rank}(\Pi) = m$，在序贯地进行检验时可以看成是 $\text{rank}(\Pi) \le m$，有

$$H(0) \subset H(1) \subset \cdots \subset H(k)$$

将(25.6)变成

$$\Delta x_t = \mu d_t + \Pi x_{t-1} \tag{25.7}$$

$$+ \Phi_1^* \Delta x_{t-1} + \cdots + \Phi_{p-1}^* \Delta x_{t-p+1} + a_t, \tag{25.8}$$

$$t = p+1, \ldots, T \tag{25.9}$$

目标是检验 Π 的秩，这等于 Π 的非零奇异值的个数，如果能得到 Π 的相合估计就可以估计 rank(Π)。从(25.9)可以看出，Π 是排除了 μ_t 和 $\Delta x_{t-i}, i = 1, \ldots, p-1$ 的线性影响后，Δx_t 对 x_{t-1} 的多对多的线性回归系数。从上面的 MLE 步骤可知，调整后的 Δx_t 和 x_{t-1} 分别为 $\hat{u}_t$ 和 $\hat{v}_t$，所以

$$\hat{u}_t = \Pi \hat{v}_t + a_t$$

在正态性假定下，可以根据 $\hat{u}_t$ 与 $\hat{v}_t$ 的典型相关分析的方法估计 rank(Π)，$\hat{\lambda}_i$ 是 $\hat{u}_t$ 与 $\hat{v}_t$ 的典型相关系数平方。

Johansen 提出了两种检验方法。首先，考虑假设

$$H_0 : \text{rank}(\Pi) = m \longleftrightarrow H_a : \text{rank}(\Pi) > m$$

(Johansen, 1988) 提出如下的似然比统计量：

$$\text{LR}_{\text{tr}}(m) = -(T-p) \sum_{i=m+1}^{k} \ln(1 - \hat{\lambda}_i)$$

如果 rank(Π) $\leq m$，则 $i > m$ 时 $\hat{\lambda}_i$ 应该很小，因此 $\text{LR}_{\text{tr}}(m)$ 也应该很小，当统计量值超过一个右侧临界值时拒绝零假设。这个检验称为 Johansen 迹协整检验。因为单位根的存在，似然比统计量在零假设下的渐近分布不再服从卡方分布，其临界值需要用模拟方法。从 $H(0)$ 开始到 $H(1)$ 序贯地进行检验，在第一次被接受时，令被接受的 $H(m)$ 的 m 为选定的秩。

(Johansen, 1988) 提出的另一种似然比检验方法针对如下假设

$$H_0 : \text{rank}(\Pi) = m \longleftrightarrow H_a : \text{rank}(\Pi) = m + 1$$

取最大特征值统计量

$$\text{LR}_{\max}(m) = -(T-p) \ln(1 - \hat{\lambda}_{m+1})$$

当统计量值超过一个右侧临界值时拒绝零假设。临界值也需要通过模拟获得。从 $H(0)$ 开始到 $H(1)$ 序贯地进行检验，在第一次被接受时，令被接受的 $H(m)$ 的 m 为选定的秩。

25.5.6 VECM 预测

估计的 VECM 模型可以用于预测。首先可以从模型得到 Δx_t 序列的预测值，然后可以从 Δx_t 反解得到 x_t 的预测值。VECM 预测与 VAR 预测的区别在于 VECM 允许有单位根和协整关系，VAR 预测不允许有单位根。

25.5.7 Johansen 方法进行协整检验的例子

R 扩展包 urca 的 `ca.jo()` 函数可以进行计算 Johansen 的两种检验。参见 (Pfaff, 2008)。

例 4.2 考虑 urca 包的 UKpppuip 数据，此数据用来检验英国经济的 PPP 和 UIP 假设。季度数据，从 1971 年第 1 季度到 1987 年第 2 季度。分量 p1 是英国批发价格指数，p2 是外贸加权批发价格指数，e12 是英镑实际汇率，i1 是三月期国债利率，i2 是三月期欧元利率，dpoil0 是即期国际油价，dpoil1 是上期国际油价。

载入数据：

```
library(urca, quietly = TRUE)
data(UKpppuip)
ts.UKpppuip <- ts(UKpppuip, start=c(1971,1), frequency=4)
```

数据表见附录。

时间序列图：

```
library(quantmod, quietly = TRUE)
plot(as.xts(ts.UKpppuip), type="l",
     multi.panel=TRUE, theme="white",
     main=" 英国 PPP-UIP 数据",
     major.ticks="years",
     grid.ticks.on = "years")
```

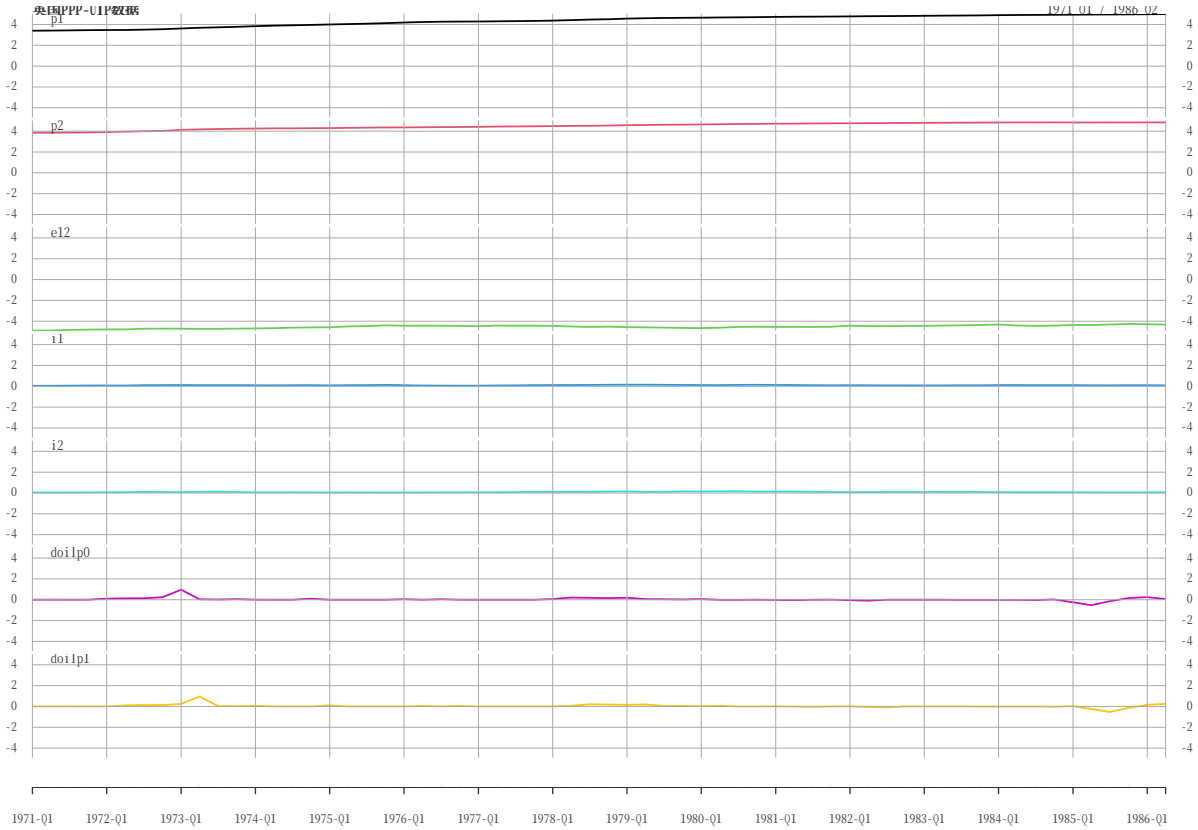


图 25.2: 英国 PPP-UIP 数据

下面对 p1, p2, e12, i1, i2 执行 Johansen 的两种协整检验。ADF 结果表明部分序列有单位根。以油价作为外生变量。选项 `type="trace"` 使用 $\mathrm{LR}_{\mathrm{tr}}$ 统计量，选项 `type="eigen"` 使用 $\mathrm{LR}_{\max}$ 统计量。选项 `ecdet` 指定要考虑的非随机趋势，`"none"` 表示没有非随机趋势，`"constant"` 表示常数趋势，`"trend"` 表示线性趋势。选项 `K` 表示 VAR 模型的阶。选项 `season=4` 表示有与季度数据相关联的哑变量存在，用 `dumvar=` 输入保存了哑变量 (外生变量) 的数据框，其行数必须与用来检验的输入数据行数相同。

```
dat1 <- UKpppuip[,c("p1", "p2", "e12", "i1", "i2")]
dat2 <- UKpppuip[,c("doilp0", "doilp1")]
summary(ca.jo(
  dat1, type="trace", K=2, season=4, dumvar=dat2))
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.40672818 0.28538240 0.25415335 0.10230406 0.08287097
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 4 |  5.19  6.50  8.18 11.65
## r <= 3 | 11.67 15.66 17.95 23.52
## r <= 2 | 29.26 28.71 31.52 37.22
## r <= 1 | 49.42 45.23 48.28 55.43
## r = 0  | 80.75 66.49 70.60 78.87
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##             p1.l2      p2.l2    e12.l2      i1.l2      i2.l2
## p1.l2   1.0000000   1.000000  1.000000  1.0000000  1.0000000
## p2.l2  -0.9086265  -1.143047 -1.272628 -2.4001444 -1.4528820
## e12.l2 -0.9321133  -3.363042  1.113631  1.1221619 -0.4805235
## i1.l2  -3.3746393  35.243576  2.746828 -0.4088865  2.2775510
## i2.l2  -1.8906210 -32.917370 -2.835714  2.9863624  0.7628011
##
## Weights W:
## (This is the loading matrix)
##
##             p1.l2         p2.l2       e12.l2        i1.l2       i2.l2
## p1.d  -0.06816507  0.0011795779 -0.002790218  0.001373599 -0.01333013
## p2.d  -0.01773477  0.0001220008 -0.014159241  0.013178503  0.00755575
## e12.d  0.10065321 -0.0001432122 -0.055628059 -0.035400025 -0.04707585
## i1.d   0.03434737 -0.0041631581 -0.010363374  0.012309982 -0.02394672
## i2.d   0.05766426  0.0082830953  0.004821036  0.026984801 -0.01006765
```

```
summary(ca.jo(
  dat1, type="eigen", K=2, season=4, dumvar=dat2))
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.40672818 0.28538240 0.25415335 0.10230406 0.08287097
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 4 |  5.19  6.50  8.18 11.65
## r <= 3 |  6.48 12.91 14.90 19.19
## r <= 2 | 17.59 18.90 21.07 25.75
## r <= 1 | 20.16 24.78 27.14 32.14
## r = 0  | 31.33 30.84 33.32 38.78
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##             p1.l2      p2.l2    e12.l2      i1.l2      i2.l2
## p1.l2   1.0000000   1.000000  1.000000  1.0000000  1.0000000
## p2.l2  -0.9086265  -1.143047 -1.272628 -2.4001444 -1.4528820
## e12.l2 -0.9321133  -3.363042  1.113631  1.1221619 -0.4805235
## i1.l2  -3.3746393  35.243576  2.746828 -0.4088865  2.2775510
## i2.l2  -1.8906210 -32.917370 -2.835714  2.9863624  0.7628011
##
## Weights W:
## (This is the loading matrix)
##
##             p1.l2         p2.l2       e12.l2        i1.l2       i2.l2
## p1.d  -0.06816507  0.0011795779 -0.002790218  0.001373599 -0.01333013
## p2.d  -0.01773477  0.0001220008 -0.014159241  0.013178503  0.00755575
## e12.d  0.10065321 -0.0001432122 -0.055628059 -0.035400025 -0.04707585
## i1.d   0.03434737 -0.0041631581 -0.010363374  0.012309982 -0.02394672
## i2.d   0.05766426  0.0082830953  0.004821036  0.026984801 -0.01006765
```

Johansen 检验是序贯进行的。从迹方法的检验结果看出，在 0.05 水平下，$H_0: r = 0$（这里 r 就是前面的 m，Π 的秩）统计量为 80.75，右侧临界值是 70.60，拒绝零假设，至少有一个协整关系；$H_0: r = 1$ 的检验统计量为 49.42，右

侧临界值为 48.28，拒绝零假设，认为至少有两个协整关系；$H_0: r = 2$ 的检验统计量为 29.26, 右侧临界值为 31.52，不拒绝零假设，认为恰好有两个协整关系。如果使用最大特征值方法，检验结论则有所不同。

结果中包含了所有的协整变量，前两个协整变量为

$$\left(\begin{array}{c|cc} \text{p1} & 1 & 1 \\ \text{p2} & -0.91 & -1.14 \\ \text{e12} & -0.93 & -3.36 \\ \text{i1} & -3.37 & 35.24 \\ \text{i2} & -1.89 & 32.91 \end{array}\right)$$

人为地计算两个协整变量，检验其平稳性：

```
dat1 <- UKpppuip[,c("p1", "p2", "e12", "i1", "i2")]
f <- function(){
  x1 <- as.matrix(dat1) %*% c(1, -0.91, -0.93, -3.37, -1.89)
  x2 <- as.matrix(dat1) %*% c(1, -1.14, -3.36, -35.24, -32.91)
  print(fUnitRoots::adfTest(x1, lags=1, type="c"))
  print(fUnitRoots::adfTest(x2, lags=1, type="c"))
  invisible()
}
f()
```

```
## Warning in if (class(x) == "timeSeries") x = series(x): 条件的长度大于一，因此只
## 能用其第一元素
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -2.5995
##   P VALUE:
##     0.0994
##
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

```
## Warning in if (class(x) == "timeSeries") x = series(x): 条件的长度大于一，因此只
## 能用其第一元素
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
```

```
## Test Results:
##   PARAMETER:
##     Lag Order: 1
##   STATISTIC:
##     Dickey-Fuller: -3.0532
##   P VALUE:
##     0.03853
##
## Description:
##  Wed Mar 23 16:33:34 2022 by user: Lenovo
```

在 0.05 水平下，第一个协整向量有单位根，第二个协整向量没有单位根。但是，因为估计这些权重消耗了自由度，所以 ADF 检验的 p 值是不可用的。

25.6 附录 A：线性模型估计相合性

考虑线性回归模型

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + e_i,\ i = 1, 2, \ldots$$

其中 $Ee_i = 0$, $\mathrm{Var}(e_i) = \sigma^2$，$e_1, e_2, \ldots$ 为不相关列。对 n 组观测值，写成线性模型

$$Y_n = X_n \beta + E_n,$$

其中 $Y_n = (y_1, \ldots, y_n)^T$, $E_n = (e_1, \ldots, e_n)^T$, $\beta_n = (\beta_1, \ldots, \beta_p)^T$, X_n 为 $n \times p$ 矩阵，第 (i, j) 元素为 x_{ij}。如果 X_n 列满秩，最小二乘估计为

$$\hat{\beta}_n = (X_n^T X_n)^{-1} X_n^T Y_n.$$

定理 25.1 (最小二乘估计的弱相合性)**.** *记 $S_n = X_n^T X_n$, 设 n 充分大时 X_n 列满秩，则 $\hat{\beta}_n$ 是 β 的弱相合估计，当且仅当 $\lim_{n\to\infty} S_n^{-1} = 0$。*

充分性是 $E\hat{\beta}_n = \beta$ 以及 $\mathrm{Var}(\hat{\beta}_n) = \sigma^2 S_n^{-1}$ 的推论。事实上，

$$E\|\hat{\beta}_n - \beta\|^2 = \sigma^2 \mathrm{tr}(S_n^{-1}) \to 0,\ n \to \infty.$$

必要性的证明见陈希孺《数理统计引论》节 5.6。

定理 25.2 (最小二乘估计的强相合性)**.** *若 $e_1, e_2, \ldots$ 独立同 $N(0, \sigma^2)$ 分布，$\lim_{n\to\infty} S_n^{-1} \to 0$，则 $\hat{\beta}_n$ 是 β 的强相合估计。*

25.7 附录 B：用到的数据

25.7.1 英国消费研究数据

```
knitr::kable(Raotbl3)
```

	lc	li	lw	dd682	dd792	dd883
1966.4	10.4831	10.5821	12.9481	NA	NA	NA
1967.1	10.4893	10.5800	12.9895	0	0	0
1967.2	10.5022	10.5990	13.0115	0	0	0
1967.3	10.5240	10.6262	13.0411	0	0	0
1967.4	10.5329	10.6145	13.0357	0	0	0
1968.1	10.5586	10.6307	13.0518	0	0	0
1968.2	10.5190	10.6316	13.0839	1	0	0
1968.3	10.5381	10.6132	13.1120	-1	0	0
1968.4	10.5422	10.6141	13.1183	0	0	0
1969.1	10.5361	10.6263	13.1144	0	0	0
1969.2	10.5462	10.6366	13.1009	0	0	0
1969.3	10.5459	10.6313	13.0882	0	0	0
1969.4	10.5552	10.6324	13.0402	0	0	0
1970.1	10.5548	10.6361	13.0391	0	0	0
1970.2	10.5710	10.6795	13.0417	0	0	0
1970.3	10.5861	10.6813	13.0261	0	0	0
1970.4	10.5864	10.6801	13.0032	0	0	0
1971.1	10.5802	10.6533	13.0364	0	0	0
1971.2	10.6006	10.6839	13.0461	0	0	0
1971.3	10.6168	10.6889	13.0850	0	0	0
1971.4	10.6275	10.7025	13.1107	0	0	0
1972.1	10.6414	10.7212	13.1241	0	0	0
1972.2	10.6629	10.7818	13.1605	0	0	0
1972.3	10.6758	10.7641	13.1748	0	0	0
1972.4	10.6881	10.7841	13.1612	0	0	0
1973.1	10.7240	10.8045	13.1050	0	0	0
1973.2	10.7143	10.8230	13.1082	0	0	0
1973.3	10.7222	10.8319	13.1059	0	0	0
1973.4	10.7156	10.8380	13.0140	0	0	0
1974.1	10.6964	10.8097	12.9301	0	0	0
1974.2	10.6990	10.7928	12.8427	0	0	0
1974.3	10.7081	10.8310	12.7710	0	0	0
1974.4	10.7142	10.8328	12.7281	0	0	0
1975.1	10.7078	10.8527	12.7692	0	0	0
1975.2	10.7073	10.8089	12.7492	0	0	0
1975.3	10.6954	10.8202	12.7664	0	0	0
1975.4	10.6910	10.8069	12.7554	0	0	0
1976.1	10.6967	10.8196	12.7605	0	0	0
1976.2	10.7015	10.8046	12.7471	0	0	0
1976.3	10.7083	10.8372	12.7238	0	0	0
1976.4	10.7127	10.8123	12.7156	0	0	0
1977.1	10.6922	10.7842	12.7555	0	0	0
1977.2	10.6874	10.7713	12.7517	0	0	0
1977.3	10.6989	10.7904	12.8018	0	0	0
1977.4	10.7224	10.8369	12.8388	0	0	0
1978.1	10.7452	10.8333	12.8438	0	0	0

25.7.2 英国 PPP 研究数据

```
knitr::kable(UKpppuip)
```

p1	p2	e12	i1	i2	doilp0	doilp1
3.399837	3.846749	-4.899593	0.0470744	0.0505981	0.0000000	0.0000000
3.412952	3.856261	-4.894152	0.0467882	0.0485997	0.0066867	0.0000000
3.430709	3.864228	-4.832260	0.0598714	0.0554347	0.0000000	0.0066867
3.452861	3.881285	-4.803663	0.0725067	0.0580802	0.0022137	0.0000000
3.465303	3.913175	-4.785582	0.0789036	0.0762200	0.1020269	0.0022137
3.469929	3.953242	-4.785582	0.0714831	0.0829615	0.1327823	0.1020269
3.503215	3.997386	-4.723762	0.1065198	0.1058904	0.1395371	0.1327823
3.542607	4.020950	-4.712609	0.1152019	0.0948555	0.2369103	0.1395371
3.615693	4.130217	-4.719131	0.1160927	0.0877361	0.9655383	0.2369103
3.683799	4.172550	-4.733877	0.1101092	0.1165378	0.0434339	0.9655383
3.722781	4.207048	-4.730211	0.1084956	0.1242510	0.0195273	0.0434339
3.772879	4.236469	-4.713544	0.1066996	0.0977617	0.0553805	0.0195273
3.835640	4.253006	-4.694693	0.0960372	0.0698991	0.0063868	0.0553805
3.885598	4.270279	-4.662791	0.0917584	0.0633503	0.0000000	0.0063868
3.920515	4.271766	-4.619633	0.1017440	0.0719485	0.0000000	0.0000000
3.956147	4.285636	-4.594710	0.1072388	0.0637257	0.0957119	0.0000000
3.998746	4.295829	-4.583076	0.0859024	0.0538251	0.0000000	0.0957119
4.036117	4.320907	-4.497610	0.1038193	0.0587405	0.0000000	0.0000000
4.076345	4.340277	-4.471787	0.1108256	0.0558131	0.0000000	0.0000000
4.125455	4.358491	-4.397771	0.1354046	0.0509782	0.0000000	0.0000000
4.185891	4.362472	-4.431758	0.1035489	0.0511683	0.0491579	0.0000000
4.234323	4.374696	-4.429279	0.0732505	0.0560022	0.0000000	0.0491579
4.263965	4.389198	-4.431758	0.0624113	0.0628809	0.0492205	0.0000000
4.277778	4.401695	-4.453790	0.0565694	0.0685928	0.0000000	0.0492205
4.283250	4.419375	-4.480076	0.0580802	0.0720416	0.0000000	0.0000000
4.302169	4.441176	-4.420557	0.0809346	0.0748291	0.0000000	0.0000000
4.321394	4.454597	-4.429279	0.0884686	0.0859942	0.0000000	0.0000000
4.340256	4.468155	-4.429279	0.1084956	0.1100196	0.0000000	0.0000000
4.370716	4.488001	-4.451366	0.1158256	0.1004786	0.0616222	0.0000000
4.415330	4.505689	-4.505689	0.1166268	0.1013826	0.2077878	0.0616222
4.469429	4.518725	-4.553932	0.1294481	0.1144887	0.1832892	0.2077878
4.502292	4.541496	-4.522783	0.1459169	0.1374985	0.1562188	0.1832892
4.561852	4.575517	-4.572380	0.1541793	0.1579435	0.1885763	0.1562188
4.597851	4.597449	-4.588381	0.1539221	0.1057105	0.0565460	0.1885763
4.623419	4.616333	-4.611395	0.1399358	0.1140427	0.0466827	0.0565460
4.637637	4.629624	-4.646949	0.1301507	0.1578581	0.0270653	0.0466827
4.652340	4.648700	-4.662791	0.1157365	0.1521200	0.0636834	0.0270653
4.671239	4.666700	-4.622706	0.1121673	0.1610128	-0.0059685	0.0636834
4.684259	4.686902	-4.546235	0.1337439	0.1692363	-0.0072731	-0.0059685
4.700753	4.703789	-4.536252	0.1434941	0.1292723	0.0028523	-0.0072731
4.725350	4.718869	-4.552836	0.1280413	0.1400228	-0.0130594	0.0028523
4.740313	4.726432	-4.542918	0.1225716	0.1396750	-0.0345760	-0.0130594
4.750136	4.739622	-4.556120	0.1016537	0.1135072	-0.0029917	-0.0345760
4.759607	4.753597	-4.530662	0.0930349	0.0926703	0.0092772	-0.0029917
4.774913	4.759102	-4.428038	0.1039996	0.0888347	-0.0472742	0.0092772
4.792479	4.769929	-4.474163	0.0933993	0.0898407	-0.0911132	-0.0472742

Chapter 26

格兰格因果性

26.1 介绍

前面已经在 VAR 部分简单介绍了格兰格因果性的概念，以及用 VAR 模型检验格兰格因果性的方法。这里对格兰格因果性的概念与检验方法进行更详细的阐述。这一章的内容主要参考 (Gebhard Kirchgässner, 2013)，(C. W. J. Granger, 1969)。

考虑两个时间序列之间的因果性。这里的因果性指的是时间顺序上的关系，如果 $X_{t-1}, X_{t-2}, ...$ 对 Y_t 有额外的预测作用，而 $Y_{t-1}, Y_{t-2}, ...$ 对 X_t 没有额外的预测作用，则称 $\{X_t\}$ 是 $\{Y_t\}$ 的格兰格原因，而 $\{Y_t\}$ 不是 $\{X_t\}$ 的格兰格原因。如果 $X_{t-1}, X_{t-2}, ...$ 对 Y_t 有额外的预测作用，$Y_{t-1}, Y_{t-2}, ...$ 对 X_t 也有额外的预测作用，则在没有进一步信息的情况下无法确定两个时间序列的因果性关系。

注意这种因果性与采样频率有关系，在日数据或者月度数据中能发现的领先——滞后性质的因果关系，到年度数据可能就以及混杂在以前变成同步的关系了。

(C. W. J. Granger, 1969) 首先提出了时间序列之间因果关系的概念。时间序列因果关系可以是假设没有因果关系，然后检验能否拒绝假设，这是 (C. W. J. Granger, 1969) 的方法；也可以是首先建立包含因果关系的模型，然后检验其中表示因果关系的参数是否不显著，这是利用 VAR 和 VECM 的方法。后一种方法首先提出于 (Sims, 1972)。

在具有格兰杰因果性的情况下，可以用来改善预测。比如，如果 $\{X_t\}$ 是 $\{Y_t\}$ 的格兰格原因，则可以利用 $\{Y_t\}$ 的过去值、$\{X_t\}$ 的过去值去预测 $\{Y_t\}$ 的当前值，会比仅利用 $\{Y_t\}$ 的过去值预测要好，这是认为 $\{X_t\}$ 的过去值中包含了 $\{Y_t\}$ 的过去值中缺少的信息，而这些信息对预测 $\{Y_t\}$ 的当前值是有作用的。(C. W. J. Granger, 1969) 的检验方法就是针对这一点进行检验。

在经济与金融时间序列建模中考虑 X_{t-1} 对 Y_t 影响的另一个原因是，许多时间序列具有较强的正自相关性，比如单位根过程或者特征根接近于单位圆的 ARMA 模型，对于这样的两个时间序列 $\{(X_t, Y_t)\}$，如果以 Y_t 为因变量，X_t 为自变量作线性回归，可能发生虚假的回归，即使两个序列之间独立，但是回归结果可以是显著的。这是因为，Y_t 与 Y_{t-1} 之间的强序列相关性。如果在回归中引入滞后项，这样的虚假回归就可以被消除。参见 (C. W. Granger & Newbold, 1974)。

从因果性的角度讲，只有两个序列的新息（innovation）对另一个序列的影响才是有意义的。参见 (Schwert, 1979)。

26.2 格兰格因果性的定义

设 $\{\xi_t\}$ 为一个时间序列，$\{\eta_t\}$ 为向量时间序列，记

$$\bar{\eta}_t = \{\eta_{t-1}, \eta_{t-2}, \dots\}$$

记 $\text{Pred}(\xi_t|\bar{\eta}_t)$ 为基于 $\eta_{t-1}, \eta_{t-2}, \dots$ 对 ξ_t 作的最小均方误差无偏预报，其解为条件数学期望 $E(\xi_t|\eta_{t-1}, \eta_{t-2}, \dots)$，在一定条件下可以等于 ξ_t 在 $\eta_{t-1}, \eta_{t-2}, \dots$ 张成的线性 Hilbert 空间的投影（比如，(ξ_t, η_t) 为平稳正态多元时间序列），即最优线性预测。直观理解成基于过去的 $\{\eta_{t-1}, \eta_{t-2}, \dots\}$ 的信息对当前的 ξ_t 作的最优预测。(C. W. J. Granger, 1969) 允许最优预测与最优线性预测，实际检验方法用了谱和互谱来分析，因为谱和互谱仅涉及二阶性质，所以 (C. W. J. Granger, 1969) 实际是用最优线性预测定义因果性。

令一步预测误差为

$$\varepsilon(\xi_t|\bar{\eta}_t) = \xi_t - \text{Pred}(\xi_t|\bar{\eta}_t)$$

令一步预测误差方差，或者均方误差，为

$$\sigma^2(\xi_t|\bar{\eta}_t) = \text{Var}(\varepsilon_t(\xi_t|\bar{\eta}_t)) = E\left[\xi_t - \text{Pred}(\xi_t|\bar{\eta}_t)\right]^2$$

考虑两个时间序列 $\{X_t\}$ 和 $\{Y_t\}$，$\{(X_t, Y_t)\}$ 宽平稳或严平稳。

- 如果
$$\sigma^2(Y_t|\bar{Y}_t, \bar{X}_t) < \sigma^2(Y_t|\bar{Y}_t)$$
则称 $\{X_t\}$ 是 $\{Y_t\}$ 的**格兰格原因**，记作 $X_t \Rightarrow Y_t$。这不排除 $\{Y_t\}$ 也可以是 $\{X_t\}$ 的格兰格原因。
- 如果 $X_t \Rightarrow Y_t$，而且 $Y_t \Rightarrow X_t$，则称互相有**反馈**关系，记作 $X_t \Leftrightarrow Y_t$。
- 如果
$$\sigma^2(Y_t|\bar{Y}_t, X_t, \bar{X}_t) < \sigma^2(Y_t|\bar{Y}_t, \bar{X}_t)$$
即除了过去的信息，增加同时刻的 X_t 信息后对 Y_t 预测有改进，则称 $\{X_t\}$ 对 $\{Y_t\}$ 有瞬时因果性。这时 $\{Y_t\}$ 对 $\{X_t\}$ 也有瞬时因果性。
- 如果 $X_t \Rightarrow Y_t$，则存在最小的正整数 m，使得
$$\sigma^2(Y_t|\bar{Y}_t, X_{t-m}, X_{t-m-1}, \dots) < \sigma^2(Y_t|\bar{Y}_t, X_{t-m-1}, X_{t-m-2}, \dots)$$
称 m 为**因果性滞后值** (causality lag)。如果 $m > 1$，这意味着在已有 $Y_{t-1}, Y_{t-2}, \dots$ 和 $X_{t-m}, X_{t-m-1}, \dots$ 的条件下，增加 X_{t-1}, ..., X_{t-m+1} 不能改进对 Y_t 的预测。

例 26.1. 设 $\{\varepsilon_t, \eta_t\}$ 是相互独立的零均值白噪声列，$\text{Var}(\varepsilon_t) = 1$, $\text{Var}(\eta_t) = 1$, 考虑

$$\begin{aligned} Y_t =& X_{t-1} + \varepsilon_t \\ X_t =& \eta_t + 0.5\eta_{t-1} \end{aligned}$$

用 $L(\cdot|\cdot)$ 表示最优线性预测，则

$$\begin{aligned} & L(Y_t|\bar{Y}_t, \bar{X}_t) \\ =& L(X_{t-1}|X_{t-1}, \dots, Y_{t-1}, \dots) + L(\varepsilon_t|\bar{Y}_t, \bar{X}_t) \\ =& X_{t-1} + 0 \\ =& X_{t-1} \\ \sigma^2(Y_t|\bar{Y}_t, \bar{X}_t) =& \text{Var}(\varepsilon_t) = 1 \end{aligned}$$

而

$$Y_t = \eta_{t-1} + 0.5\eta_{t-2} + \varepsilon_t$$

有

$$\gamma_Y(0) = 2.25, \gamma_Y(1) = 0.5, \gamma_Y(k) = 0, k \geq 2$$

所以 $\{Y_t\}$ 是一个 MA(1) 序列，设其方程为

$$Y_t = \zeta_t + b\zeta_{t-1}, \zeta_t \sim \mathrm{WN}(0, \sigma_\zeta^2)$$

可以解出

$$\begin{aligned}
\rho_Y(1) =& \frac{\gamma_Y(1)}{\gamma_Y(0)} = \frac{2}{9} \\
b =& \frac{1 - \sqrt{1 - 4\rho_Y^2(1)}}{2\rho_Y(1)} \approx 0.2344 \\
\sigma_\zeta^2 =& \frac{\gamma_Y(1)}{b} \approx 2.1328
\end{aligned}$$

于是

$$\sigma^2(Y_t|\bar{Y}_t) = \sigma_\zeta^2 = 2.1328$$

所以

$$\sigma^2(Y_t|\bar{Y}_t, \bar{X}_t) = 1 < 2.1328 = \sigma^2(Y_t|\bar{Y}_t)$$

即 X_t 是 Y_t 的格兰格原因。

反之，X_t 是 MA(1) 序列，有

$$\eta_t = \frac{1}{1 + 0.5B} X_t = \sum_{j=0}^{\infty} (-0.5)^j X_{t-j}$$

其中 B 是推移算子（滞后算子）。于是

$$\begin{aligned}
L(X_t|\bar{X}_t) =& L(\eta_t|\bar{X}_t) + 0.5L(\eta_{t-1}|\bar{X}_t) \\
=& 0.5 \sum_{j=0}^{\infty} (-0.5)^j X_{t-1-j} \\
=& -\sum_{j=1}^{\infty} (-0.5)^j X_{t-j} \\
\sigma^2(X_t|\bar{X}_t) =& \mathrm{Var}(X_t - L(X_t|\bar{X}_t)) \\
=& \mathrm{Var}(\eta_t) = 1
\end{aligned}$$

而

$$\begin{aligned}
L(X_t|\bar{X}_t, \bar{Y}_t) =& L(\eta_t|\bar{X}_t, \bar{Y}_t) + 0.5L(\eta_{t-1}|\bar{X}_t, \bar{Y}_t) \\
=& 0 + 0.5L(\sum_{j=0}^{\infty} (-0.5)^j X_{t-1-j}|\bar{X}_t, \bar{Y}_t) \\
=& -\sum_{j=1}^{\infty} (-0.5)^j X_{t-j} \\
=& L(X_t|\bar{X}_t)
\end{aligned}$$

所以 Y_t 不是 X_t 的格兰格原因。

考虑瞬时因果性。

$$\begin{aligned}
L(Y_t|\bar{X}_t, \bar{Y}_t, X_t) =& X_{t-1} + 0(\text{注意}\varepsilon_t\text{与}\{X_s, \forall s\}\text{不相关} \\
=& L(Y_t|\bar{X}_t, \bar{Y}_t)
\end{aligned}$$

所以 X_t 不是 Y_t 的瞬时格兰格原因。

例 26.2. 在例26.1中，如果模型改成

$$\begin{aligned} Y_t =& X_t + \varepsilon_t \\ X_t =& \eta_t + 0.5\eta_{t-1} \end{aligned}$$

有怎样的结果？

这时

$$Y_t = \varepsilon_t + \eta_t + 0.5\eta_{t-1}$$

仍有

$$\gamma_Y(0) = 2.25, \gamma_Y(1) = 0.5, \gamma_Y(k) = 0, k \geq 2$$

所以 Y_t 还服从 MA(1) 模型

$$Y_t = \zeta_t + b\zeta_{t-1}, b \approx 0.2344, \sigma_\zeta^2 \approx 2.1328$$

$$\begin{aligned} L(Y_t|\bar{Y}_t, \bar{X}_t) =& L(X_t|\bar{Y}_t, \bar{X}_t) + 0 \\ =& L(\eta_t|\bar{Y}_t, \bar{X}_t) + 0.5L(\eta_{t-1}|\bar{Y}_t, \bar{X}_t) \\ =& 0 + 0.5L(\sum_{j=0}^{\infty}(-0.5)^j X_{t-1-j}|\bar{Y}_t, \bar{X}_t) \\ =& -\sum_{j=1}^{\infty}(-0.5)^j X_{t-j} \\ =& X_t - \eta_t \\ \sigma^2(Y_t|\bar{Y}_t, \bar{X}_t) =& \mathrm{Var}(\varepsilon_t + \eta_t) = 2 \end{aligned}$$

而

$$\sigma^2(Y_t|\bar{Y}_t) = \sigma_\zeta^2 \approx 2.1328 > \sigma^2(Y_t|\bar{Y}_t, \bar{X}_t) = 2$$

所以 X_t 是 Y_t 的格兰格原因。

反之，

$$\begin{aligned} L(X_t|\bar{X}_t, \bar{Y}_t) =& -\sum_{j=1}^{\infty}(-0.5)^j X_{t-j} \\ =& L(X_t|\bar{X}_t) \end{aligned}$$

所以 Y_t 不是 X_t 的格兰格原因。

考虑瞬时因果性。

$$\begin{aligned} L(Y_t|\bar{X}_t, \bar{Y}_t, X_t) =& X_t + 0(\text{注意}\varepsilon_t\text{与}\{X_s, \forall s\}\text{不相关} \\ =& X_t \\ \sigma^2(Y_t|\bar{X}_t, \bar{Y}_t, X_t) =& \mathrm{Var}(\varepsilon) \\ =& 1 < 2 = \sigma^2(Y_t|\bar{X}_t, \bar{Y}_t) \end{aligned}$$

所以 X_t 是 Y_t 的瞬时格兰格原因。

例 26.3. 设

$$\begin{aligned} X_t =& 0.5X_{t-1} + \varepsilon_t \\ Y_t =& 0.1X_{t-1} + 0.5Y_{t-1} + \eta_t \end{aligned}$$

其中

$$\begin{pmatrix} \varepsilon_t \\ \eta_t \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right)$$

为独立同分布的向量白噪声列，$|\rho| < 1$。

易见

$$\begin{aligned} E(X_t|\bar{X}_t) =& 0.5X_{t-1} \\ \sigma^2(X_t|\bar{X}_t) =& \mathrm{Var}(\varepsilon_t) = 1 \\ E(X_t|\bar{X}_t, \bar{Y}_t) =& 0.5X_{t-1} = E(X_t|\bar{X}_t) \\ \sigma^2(X_t|\bar{X}_t, \bar{Y}_t) =& 1 = \sigma^2(X_t|\bar{X}_t) \end{aligned}$$

所以 Y_t 不是 X_t 的格兰格原因。

反过来，

$$\begin{aligned} E(Y_t|\bar{X}_t, \bar{Y}_t) =& 0.1X_{t-1} + 0.5Y_{t-1} \\ \sigma^2(Y_t|\bar{X}_t, \bar{Y}_t) =& \mathrm{Var}(\eta_t) = 1 \\ E(Y_t|\bar{Y}_t) =& 0.1E(X_{t-1}|\bar{Y}_t) + 0.5Y_{t-1} \\ \sigma^2(Y_t|\bar{Y}_t) =& E\left[Y_t - E(Y_t|\bar{Y}_t)\right]^2 \\ =& E\left[\eta_t + 0.1(X_{t-1} - E(X_{t-1}|\bar{Y}_t))\right]^2 \\ =& 1 + 0.1^2\sigma^2(X_{t-1}|\bar{Y}_t) \\ >& 1 = E(Y_t|\bar{X}_t, \bar{Y}_t) \end{aligned}$$

所以 X_t 是 Y_t 的格兰格原因。

这些概念可以推广到有多个序列的情况。设所有构成互相影响的序列为 $Y_t^D = \{Y_t^{(i)}, i \in D\}$，如 $D = \{1, 2, \dots, K\}$。记 $D(j) = D \backslash \{j\}$，$Y_t^{D(j)} = \{Y_t^{(i)}, i \in D(j)\}$。这时，称 $Y_t^{(j)}$ 是 $Y_t^{(i)}$ 的格兰格原因，如果

$$\sigma^2(Y_t^{(i)}|Y_{t-1}^D, Y_{t-2}^D, \dots) < \sigma^2(Y_t^{(i)}|Y_{t-1}^{D(j)}, Y_{t-2}^{D(j)}, \dots).$$

这样定义的因果性在用到实际问题时，需要假定可能造成因果关系的数据都已经在 $\{Y_t^D\}$ 中，如果存在外面的序列 $\{A_t\}$，使得 A_t 影响了所有 $Y_t^{(i)}$，则仅有 $\{Y_t^D\}$ 数据时得到的因果性可能是虚假的，(C. W. J. Granger, 1969) 第 4 节给出了一个例子。

例 26.4. 设

$$\begin{aligned} Y_t =& A_{t-1} + \xi_t \\ X_t =& A_t + \eta_t \end{aligned}$$

其中 $\{A_t\}$ 是一个宽平稳时间序列，$\{\varepsilon_t\}$, $\{\xi_t\}$ 是两个相互独立的标准白噪声列，且都与 $\{A_t\}$ 独立。

则数据仅有 $Y_t^D = \{X_t, Y_t\}$ 时，X_t 是 Y_t 的格兰格原因。以 $\{A_t\}$ 是独立同分布白噪声列来证明。设 $\{A_t\}$, $\{\xi_t\}$, $\{\eta_t\}$ 相互独立，都是独立同分布白噪声列 WN(0,1)。这时

$$\mathrm{Pred}(Y_t|\bar{Y}_t) = \mathrm{Pred}(A_{t-1}|\bar{Y}_t) = 0,$$

这是因为 ξ_t 与 $\bar{Y}_t$ 独立，A_{t-1} 与 $\bar{Y}_t$ 独立。于是

$$\begin{aligned}\varepsilon(Y_t|\bar{Y}_t) =& Y_t - \text{Pred}(Y_t|\bar{Y}_t) = Y_t = A_{t-1} + \xi_t,\\ \sigma^2(Y_t|\bar{Y}_t) =& \text{Var}[\varepsilon(Y_t|\bar{Y}_t)] = 2;\end{aligned}$$

而

$$\begin{aligned}\text{Pred}(Y_t|\bar{Y}_t, \bar{X}_t) =& \text{Pred}(A_{t-1}|\bar{Y}_t, \bar{X}_t) = \text{Pred}(X_{t-1} - \eta_{t-1}|\bar{Y}_t, \bar{X}_t)\\ =& X_{t-1} - \text{Pred}(\eta_{t-1}|\bar{Y}_t, \bar{X}_t),\\ \varepsilon(Y_t|\bar{Y}_t, \bar{X}_t) =& Y_t - \text{Pred}(Y_t|\bar{Y}_t, \bar{X}_t)\\ =& \xi_t + X_{t-1} - \eta_{t-1} - [X_{t-1} - \text{Pred}(\eta_{t-1}|\bar{Y}_t, \bar{X}_t)]\\ =& \xi_t - [\eta_{t-1} - \text{Pred}(\eta_{t-1}|\bar{Y}_t, \bar{X}_t)],\\ \sigma^2(Y_t|\bar{Y}_t, \bar{X}_t) =& \text{Var}[\varepsilon(Y_t|\bar{Y}_t, \bar{X}_t)]\\ =& 1 + E\{[\eta_{t-1} - \text{Pred}(\eta_{t-1}|\bar{Y}_t, \bar{X}_t)]^2\} < 2,\end{aligned}$$

所以 $\{X_t\}$ 是 $\{Y_t\}$ 的格兰格原因。证明中利用了

$$E\{[\eta_{t-1} - \text{Pred}(\eta_{t-1}|\bar{Y}_t, \bar{X}_t)]^2\} < \text{Var}(\eta_{t-1}) = 1,$$

这是因为

$$\text{Cov}(\eta_{t-1}, X_{t-1}) = \text{Cov}(\eta_{t-1}, A_{t-1} + \eta_{t-1}) = 1 \neq 0,$$

所以上式成立。

另一方面，如果数据为 $Y_t^{D'} = \{A_t, X_t, Y_t\}$，则 A_t 是 Y_t 的格兰格原因，而 X_t 不是 Y_t 的格兰格原因。事实上，

$$\begin{aligned}E(Y_t|\bar{A}_t, \bar{X}_t, \bar{Y}_t) =& E(A_{t-1}|\bar{A}_t, \bar{X}_t, \bar{Y}_t) + E(\xi_t|\bar{A}_t, \bar{X}_t, \bar{Y}_t)\\ =& A_{t-1} + 0\\ =& E(Y_t|\bar{A}_t, \bar{Y}_t) = E(Y_t|\bar{A}_t),\end{aligned}$$

即增加 $\bar{X}_t$ 的信息不能改进预测。所以，当 $Y_t^D = (X_t, Y_t)$ 时得到的因果性是虚假的。

26.3 两变量 VAR 情形

$$\begin{aligned}X_t &= a(B)X_t + b(B)Y_t + \xi_t\\ Y_t &= c(B)X_t + d(B)Y_t + \eta_t\end{aligned}$$

其中 $a(z) = a_1 z + \cdots + a_m z^m$，$b(z), c(z), d(z)$ 类似定义，B 为推移算子。$\{\xi_t\}$, $\{\eta_t\}$ 为零均值白噪声，两者可以有同步相关性。当且仅当 $b(z) \equiv 0$ 时，Y_t 不是 X_t 的格兰格原因。

上述模型应该满足 VAR 的平稳性条件。这时，存在平稳解 (X_t, Y_t) 且 (ξ_t, η_t) 与 $(X_{t-j}, Y_{t-j}), j = 1, 2, ...$ 不相关。

当 ξ_t 和 η_t 有同步相关性时，X_t 与 Y_t 有瞬时因果性。但当 $b(z) \equiv 0$ 时，仍有 Y_t 不是 X_t 的格兰格原因。

如果 ξ_t 和 η_t 没有同步相关性，则 X_t 与 Y_t 没有瞬时因果性。

26.4 两变量 VMA 情形

考虑如下二元时间序列模型：

$$\begin{aligned} X_t &= a(B)\xi_t + b(B)\eta_t \\ Y_t &= c(B)\xi_t + d(B)\eta_t \end{aligned}$$

其中

$$\begin{aligned} a(z) =& 1 + a_1 z + \cdots + a_m z^m \\ b(z) =& b_1 z + \cdots + b_m z^m \\ c(z) =& c_1 z + \cdots + c_m z^m \\ d(z) =& 1 + d_1 z + \cdots + d_m z^m \end{aligned}$$

B 为推移算子。$\{\xi_t\}, \{\eta_t\}$ 为零均值白噪声，可以存在同步相关性。

如果 ξ_t 是 $\{X_t\}$ 的新息且 $b(z) \equiv 0$，则 Y_t 不是 X_t 的格兰格原因。

Y_t 与 X_t 有瞬时因果关系，当且仅当 ξ_t 与 η_t 有同步相关性。

例 26.5. 当 $m = 1$ 时，模型为

$$\begin{aligned} X_t =& \xi_t + a_1\xi_{t-1} + b_0\eta_t + b_1\eta_{t-1} \\ Y_t =& c_0\xi_t + c_1\xi_{t-1} + \eta_t + d_1\eta_{t-1} \end{aligned}$$

其中 $(\xi_t, \eta_t) \sim \mathrm{WN}(0, \mathrm{diag}(\sigma_x^2, \sigma_\eta^2))$。

如果 $b_0 = b_1 = 0$，且设 $0 < |a_1| < 1$，则模型变成

$$\begin{aligned} X_t =& \xi_t + a_1\xi_{t-1} \\ Y_t =& c_0\xi_t + c_1\xi_{t-1} + \eta_t + d_1\eta_{t-1} \end{aligned}$$

这时

$$\xi_t = \sum_{j=0}^{\infty} (-a_1)^j X_{t-j}$$

是 $\{X_t\}$ 的新息列，有

$$\begin{aligned} L(X_t|\bar{X}_t, \bar{Y}_t) =& 0 + a_1 L(\xi_{t-1}|\bar{X}_t, \bar{Y}_t) \\ =& a_1 L(\sum_{j=0}^{\infty} (-a_1)^j X_{t-1-j}|\bar{X}_t, \bar{Y}_t) \\ =& -\sum_{j=1}^{\infty} (-a_1)^j X_{t-j} \\ =& L(X_t|\bar{X}_t) \end{aligned}$$

所以 Y_t 不是 X_t 的格兰格原因。

26.5 单变量模型表示

一个平稳可逆的 VAR(m) 二元时间序列可以有两种表示。简约形式的 VAR 表示为

$$\Phi(B)\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \begin{pmatrix} \Phi_{11}(B) & \Phi_{12}(B) \\ \Phi_{21}(B) & \Phi_{22}(B) \end{pmatrix}\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \begin{pmatrix} \xi_t \\ \eta_t \end{pmatrix}$$

其中 $\Phi(z)=I-A_1z-\cdots-A_mz^m$ 是 2 阶方阵，每个元素是 z 的 m 次多项式，满足 $\det(\Phi(\lambda))$ 的根都在单位圆外的条件。$\Phi_{11}(z)$ 和 $\Phi_{22}(z)$ 的常数项为 1，$\Phi_{12}(z)$ 和 $\Phi_{21}(z)$ 的常数项为 0。B 是推移算子。$(\xi_t,\eta_t)^T$ 是零均值向量白噪声列，协方差阵为 Σ。

当且仅当 $\Phi_{12}(z)\equiv 0$ 时 Y_t 不是 X_t 的格兰格原因。

可以得到

$$\Psi(z)=[\Phi(z)]^{-1}=\begin{pmatrix}\Psi_{11}(z) & \Psi_{12}(z)\\ \Psi_{21}(z) & \Psi_{22}(z)\end{pmatrix}=[\det(\Phi(z))]^{-1}\begin{pmatrix}\Phi_{22}(z) & -\Phi_{12}(z)\\ -\Phi_{21}(z) & \Phi_{11}(z)\end{pmatrix}$$

其中 $\Psi(z)$ 是 2 阶方阵，每个元素是 z 的无穷阶多项式且在根都在单位圆外，$\Psi(z)$ 的 $(1,2)$ 元素与 $(2,1)$ 的常数项等于 0。这时有 MA 表示：

$$\begin{pmatrix}X_t\\ Y_t\end{pmatrix}=\Psi(B)\begin{pmatrix}\xi_t\\ \eta_t\end{pmatrix}$$

当且仅当 $\Psi_{12}(z)\equiv 0$ 时，即 $\Phi_{12}(z)\equiv 0$ 时，Y_t 不是 X_t 的格兰格原因。

(X_t,Y_t) 还可以分别建立一元的 Wold 表示：

$$\begin{pmatrix}X_t\\ Y_t\end{pmatrix}=\begin{pmatrix}W_1(B)u_t\\ W_2(B)v_t\end{pmatrix}=\begin{pmatrix}W_1(B) & 0\\ 0 & W_2(B)\end{pmatrix}\begin{pmatrix}u_t\\ v_t\end{pmatrix}=W(B)\begin{pmatrix}u_t\\ v_t\end{pmatrix}$$

其中 $\{u_t\}$ 是 $\{X_t\}$ 的信息列，其中 $\{v_t\}$ 是 $\{Y_t\}$ 的信息列。利用 MA 形式写成

$$\begin{pmatrix}X_t\\ Y_t\end{pmatrix}=W(B)\{W(B)\}^{-1}\Psi(B)\begin{pmatrix}\xi_t\\ \eta_t\end{pmatrix}=W(B)V(B)\begin{pmatrix}\xi_t\\ \eta_t\end{pmatrix}$$

其中

$$V(z)=\begin{pmatrix}\frac{\Psi_{11}(z)}{W_1(z)} & \frac{\Psi_{12}(z)}{W_1(z)}\\ \frac{\Psi_{21}(z)}{W_2(z)} & \frac{\Psi_{22}(z)}{W_2(z)}\end{pmatrix}$$

这提示

$$\begin{pmatrix}u_t\\ v_t\end{pmatrix}=V(B)\begin{pmatrix}\xi_t\\ \eta_t\end{pmatrix}$$

当 $\Psi_{12}(z)\equiv 0$ 时 Y_t 不是 X_t 的格兰格原因，且这时 ξ_t 和 u_t 都是 $\{X_t\}$ 的新息，所以有 $\xi_t=u_t$，$V_{11}(z)\equiv 1$，对 $k\geq 1$ 有

$$\mathrm{Cov}(u_t,v_{t-k})=E\{\xi_tV_{21}(B)\xi_{t-k}+\xi_tV_{22}(B)\eta_{t-k}\}=0$$

当 ξ_t 与 η_t 没有同步相关时，对 $k=0$ 也有 $\mathrm{Cov}(u_t,v_t)=0$。可以根据这种性质，利用两个序列的新息的互相关系数进行因果性检验。

26.6 因果性检验

为了检验 Y_t 不是 X_t 的格兰格原因，可以检验在对 X_t 预测时加入 $\bar{Y}_t$ 的信息是否改进对 X_t 的预测；可以在 VAR 或者 VAR 模型中检验相应的系数是否为零。

参考：(Gebhard Kirchgässner, 2013)。

26.6.1 直接格兰格方法

原始文献：Sargent, Thomas J. (1976) A Classical Macroeconomic Model for the United States, Journal of Political Economy 84, pp. 207-237.

对两个平稳列 x_t, y_t，直接利用增加一个序列的历史值能否改善对另一个序列的预测来判断是否有格兰格因果关系。使用普通最小二乘法估计如下的回归：

$$y_t = \alpha_0 + \sum_{j=1}^{k_1} \alpha_{11,j} y_{t-j} + \sum_{j=k_0}^{k_2} \alpha_{12,j} x_{t-j} + \xi_{1t} \tag{26.1}$$

其中 $k_0 = 1$。检验零假设：

$$\alpha_{12,1} = \alpha_{12,2} = \ldots \alpha_{11,k_2} = 0$$

采用多个回归系数为零的 F 检验。拒绝零假设时 x_t 是 y_t 的格兰格原因；不拒绝零假设时，x_t 不是 y_t 的格兰格原因。

为检验 y_t 是不是 x_t 的格兰格原因，可以建立类似(26.1)的以 x_t 为因变量的回归方程。

如果两个方向都拒绝零假设，则 x_t 与 y_t 有反馈关系。

为了检验有无瞬时因果性，可以在(26.1)中取 $k_0 = 0$，即在等式右侧加入 x_t 的同步项，并用 t 或者 F 统计量检验

$$H_0 : \alpha_{12,0} = 0$$

拒绝零假设时有瞬时因果性，否则无瞬时因果性。当 $k_1 = k_2$ 时，检验瞬时因果性的结果不依赖于以 x_t 还是以 y_t 为因变量。

这一检验方法的缺点是严重地依赖于滞后项数 k_2 的选择，k_2 越大，能检验到的滞后影响越大，但是由于平均效应的作用检验的功效会下降。可以选取多个 k_2 值进行检验，如果结果一致，则检验结果比较稳定。另一种办法是计算回归的 AIC 或者 BIC 值，这里参数个数计为 $k_1 + k_2 + 1(k_0 = 1)$。

例 26.6. 英、加、美 GDP 季度增长率的因果关系，仅考虑英国和每国。

读入数据，并单独保存各个滞后项。

```
my_lag <- function(x, k){
  c(rep(NA, k), x[1:(length(x)-k)])
}
da <- read.table("q-gdp-ukcaus.txt", header=TRUE)
da2 <- da[,c("uk", "us")]
da3 <- apply(da2, 2, function(x) diff(log(x))*100)
da3 <- as.data.frame(da3)
max_lag <- 8
for(j in 1:max_lag){
  da3 <- cbind(da3, my_lag(da3[,"uk"], k=j))
  da3 <- cbind(da3, my_lag(da3[,"us"], k=j))
}
colnames(da3) <- c(
```

```
  "uk", "us",
  paste(rep(c("uk", "us"), 8), rep(1:8, each=2), sep=""))
ts.gdp2r <- ts(da3, start=c(1980,1), frequency=4)
head(round(da3[,1:10], 2), 12)
```

```
##       uk    us   uk1   us1   uk2   us2   uk3   us3   uk4   us4
## 1  -1.80 -2.07    NA    NA    NA    NA    NA    NA    NA    NA
## 2  -0.19 -0.19 -1.80 -2.07    NA    NA    NA    NA    NA    NA
## 3  -1.11  1.83 -0.19 -0.19 -1.80 -2.07    NA    NA    NA    NA
## 4  -0.68  2.06 -1.11  1.83 -0.19 -0.19 -1.80 -2.07    NA    NA
## 5   0.21 -0.80 -0.68  2.06 -1.11  1.83 -0.19 -0.19 -1.80 -2.07
## 6   1.35  1.21  0.21 -0.80 -0.68  2.06 -1.11  1.83 -0.19 -0.19
## 7   0.01 -1.25  1.35  1.21  0.21 -0.80 -0.68  2.06 -1.11  1.83
## 8   0.37 -1.66  0.01 -1.25  1.35  1.21  0.21 -0.80 -0.68  2.06
## 9   1.26  0.54  0.37 -1.66  0.01 -1.25  1.35  1.21  0.21 -0.80
## 10  0.00 -0.39  1.26  0.54  0.37 -1.66  0.01 -1.25  1.35  1.21
## 11  0.53  0.08  0.00 -0.39  1.26  0.54  0.37 -1.66  0.01 -1.25
## 12  1.45  1.24  0.53  0.08  0.00 -0.39  1.26  0.54  0.37 -1.66
```

建立预报英国的线性回归模型，使用英国和美国的最多 4 个季度的滞后变量：

```
lm1 <- lm(uk ~ uk1 + uk2 + uk3 + uk4 + us1 + us2 + us3 + us4, data=da3)
summary(lm1)
```

```
##
## Call:
## lm(formula = uk ~ uk1 + uk2 + uk3 + uk4 + us1 + us2 + us3 + us4,
##     data = da3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.72521 -0.27778  0.05682  0.32431  1.27704
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.1858767  0.0776542   2.394  0.01834 *
## uk1          0.4320162  0.0954924   4.524 1.52e-05 ***
## uk2         -0.0008427  0.1021392  -0.008  0.99343
## uk3          0.1498499  0.1039568   1.441  0.15224
## uk4         -0.0027531  0.0949600  -0.029  0.97692
## us1          0.1201269  0.0847244   1.418  0.15901
## us2          0.0381883  0.0845357   0.452  0.65233
## us3          0.1363833  0.0829753   1.644  0.10305
## us4         -0.2115821  0.0761625  -2.778  0.00641 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5293 on 112 degrees of freedom
##   (因为不存在，4个观察量被删除了)
## Multiple R-squared:  0.3966, Adjusted R-squared:  0.3535
## F-statistic: 9.203 on 8 and 112 DF,  p-value: 1.136e-09
```

回归有效，R^2 约为 0.40。

去掉美国的滞后项再回归：

```
lm2 <- lm(uk ~ uk1 + uk2 + uk3 + uk4, data=da3)
summary(lm2)
```

```
##
## Call:
## lm(formula = uk ~ uk1 + uk2 + uk3 + uk4, data = da3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.95459 -0.28774  0.07628  0.33604  1.22413
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.22588    0.07255   3.113  0.00233 **
## uk1          0.47320    0.09272   5.103 1.32e-06 ***
## uk2          0.04367    0.10054   0.434  0.66486
## uk3          0.16558    0.10093   1.641  0.10361
## uk4         -0.07051    0.08844  -0.797  0.42692
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5528 on 116 degrees of freedom
##   (因为不存在，4个观察量被删除了)
## Multiple R-squared:  0.3183, Adjusted R-squared:  0.2948
## F-statistic: 13.54 on 4 and 116 DF,  p-value: 4.351e-09
```

回归结果仍有效，R^2 为 0.32。比较这两个嵌套的模型：

```
anova(lm1, lm2)
```

```
## Analysis of Variance Table
##
## Model 1: uk ~ uk1 + uk2 + uk3 + uk4 + us1 + us2 + us3 + us4
## Model 2: uk ~ uk1 + uk2 + uk3 + uk4
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    112 31.374
```

```
## 2    116 35.447 -4   -4.0736 3.6356 0.007985 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

在 0.05 水平下检验结果显著。这说明预报时去掉美国的滞后数据对预报是有损失的，即美国是英国的格兰格原因。

取 8 个滞后项重新做一遍：

```
lm3 <- lm(uk ~ uk1 + uk2 + uk3 + uk4 + uk5 + uk6 + uk7 + uk8
          + us1 + us2 + us3 + us4 + us5 + us6 + us7 + us8, data=da3)
summary(lm3)
```

```
##
## Call:
## lm(formula = uk ~ uk1 + uk2 + uk3 + uk4 + uk5 + uk6 + uk7 + uk8 +
##     us1 + us2 + us3 + us4 + us5 + us6 + us7 + us8, data = da3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44719 -0.27235  0.01485  0.27273  1.05355
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.05038    0.08640   0.583 0.561138
## uk1          0.47535    0.09551   4.977  2.7e-06 ***
## uk2         -0.22172    0.11333  -1.956 0.053204 .
## uk3          0.28998    0.11328   2.560 0.011966 *
## uk4         -0.15414    0.11345  -1.359 0.177294
## uk5          0.18674    0.11190   1.669 0.098286 .
## uk6         -0.10806    0.10047  -1.076 0.284701
## uk7          0.13634    0.09835   1.386 0.168743
## uk8         -0.26884    0.09081  -2.961 0.003835 **
## us1          0.34931    0.09453   3.695 0.000359 ***
## us2         -0.08949    0.09596  -0.933 0.353278
## us3          0.20481    0.09643   2.124 0.036138 *
## us4         -0.28427    0.09173  -3.099 0.002521 **
## us5          0.10928    0.08826   1.238 0.218559
## us6         -0.06517    0.08760  -0.744 0.458694
## us7          0.11906    0.08515   1.398 0.165167
## us8          0.12681    0.07420   1.709 0.090530 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4878 on 100 degrees of freedom
##   (因为不存在，8个观察量被删除了)
```

```
## Multiple R-squared:  0.5325, Adjusted R-squared:  0.4577
## F-statistic:  7.12 on 16 and 100 DF,  p-value: 1.12e-10
```

```
lm4 <- lm(uk ~ uk1 + uk2 + uk3 + uk4 + uk5 + uk6 + uk7 + uk8, data=da3)
summary(lm4)
```

```
##
## Call:
## lm(formula = uk ~ uk1 + uk2 + uk3 + uk4 + uk5 + uk6 + uk7 + uk8,
##     data = da3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.90686 -0.25167  0.04234  0.34808  1.04474
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.21135    0.08226   2.569   0.0116 *
## uk1          0.49539    0.09465   5.234 8.23e-07 ***
## uk2          0.07643    0.10484   0.729   0.4676
## uk3          0.15853    0.10400   1.524   0.1304
## uk4         -0.11202    0.10484  -1.068   0.2877
## uk5          0.02367    0.10593   0.223   0.8236
## uk6          0.02614    0.10293   0.254   0.8000
## uk7          0.12883    0.10135   1.271   0.2064
## uk8         -0.17309    0.08911  -1.942   0.0547 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5435 on 108 degrees of freedom
##   (因为不存在，8个观察量被删除了)
## Multiple R-squared:  0.3732, Adjusted R-squared:  0.3268
## F-statistic: 8.038 on 8 and 108 DF,  p-value: 1.852e-08
```

```
anova(lm3, lm4)
```

```
## Analysis of Variance Table
##
## Model 1: uk ~ uk1 + uk2 + uk3 + uk4 + uk5 + uk6 + uk7 + uk8 + us1 + us2 +
##     us3 + us4 + us5 + us6 + us7 + us8
## Model 2: uk ~ uk1 + uk2 + uk3 + uk4 + uk5 + uk6 + uk7 + uk8
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    100 23.797
## 2    108 31.907 -8   -8.1102 4.2602 0.0001965 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

结果说明美国是英国的格兰格原因。

取滞后 4，检验英国是不是美国的格兰格原因：

```
lm5 <- lm(us ~ uk1 + uk2 + uk3 + uk4
          + us1 + us2 + us3 + us4, data=da3)
summary(lm5)
```

```
##
## Call:
## lm(formula = us ~ uk1 + uk2 + uk3 + uk4 + us1 + us2 + us3 + us4,
##     data = da3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.00435 -0.33896  0.02654  0.39247  1.31429
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.2281998  0.0857538   2.661  0.00893 **
## uk1          0.5335165  0.1054525   5.059 1.66e-06 ***
## uk2         -0.2284360  0.1127926  -2.025  0.04522 *
## uk3          0.0008313  0.1147997   0.007  0.99424
## uk4          0.1621996  0.1048646   1.547  0.12474
## us1          0.2199575  0.0935613   2.351  0.02047 *
## us2          0.2403357  0.0933530   2.574  0.01134 *
## us3         -0.0483437  0.0916298  -0.528  0.59882
## us4         -0.1735589  0.0841065  -2.064  0.04137 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5845 on 112 degrees of freedom
##   (因为不存在，4个观察量被删除了)
## Multiple R-squared:  0.4148, Adjusted R-squared:  0.373
## F-statistic: 9.922 on 8 and 112 DF,  p-value: 2.334e-10
```

```
lm6 <- lm(us ~ us1 + us2 + us3 + us4, data=da3)
summary(lm6)
```

```
##
## Call:
## lm(formula = us ~ us1 + us2 + us3 + us4, data = da3)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -2.35757 -0.30055  0.07061  0.43079  1.51800
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.293791   0.093607   3.139  0.00215 **
## us1          0.341741   0.091681   3.728  0.00030 ***
## us2          0.240559   0.095254   2.525  0.01290 *
## us3          0.004113   0.096357   0.043  0.96603
## us4         -0.052820   0.084229  -0.627  0.53182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6499 on 116 degrees of freedom
##   (因为不存在，4个观察量被删除了)
## Multiple R-squared:  0.2505, Adjusted R-squared:  0.2246
## F-statistic:  9.69 on 4 and 116 DF,  p-value: 8.498e-07
```

```
anova(lm5, lm6)
```

```
## Analysis of Variance Table
##
## Model 1: us ~ uk1 + uk2 + uk3 + uk4 + us1 + us2 + us3 + us4
## Model 2: us ~ us1 + us2 + us3 + us4
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    112 38.259
## 2    116 49.002 -4   -10.742 7.8617 1.272e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

结果表明英国也是美国的格兰格原因，两者是反馈关系。

26.6.2 Haugh-Pierce 方法

原始文献：

- LARRY D. HAUGH (1976), Checking the Independence of Two Covariance Stationary Time Series: A Univariate Residual Cross-Correlation Approach, Journal of the American Statistical Association 71, pp. 378-385
- DAVID A. PIERCE and LARRY D. HAUGH (1977), Causality in Temporal Systems: Characterizations and a Survey, Journal of Econometrics 5, pp. 265-293

Haugh(1976) 和 Pierce and Haugh(1977) 提出了基于一元时间序列建模的方法。对 x_t 和 y_t 分别建立一元 ARMA 模型，计算分别的残差 a_t 和 b_t，用 Ljung-Box 白噪声检验方法进行残差诊断，确认残差为白噪声。然后，计算两个

残差之间的互相关系数 $\hat{\rho}_{ab}(k)$，对 $k>0$，$\rho_{ab}(k)$ 代表了 y_t 对 x_t 的影响。取统计量

$$S = T\sum_{k=k_1}^{k_2}\hat{\rho}_{ab}^2(k)$$

在零假设

$$H_0 : \rho_{ab}(k) = 0, k = k_1, k_1+1, \dots, k_2$$

下 S 近似服从 $\chi^2(k_2-k_1+1)$。

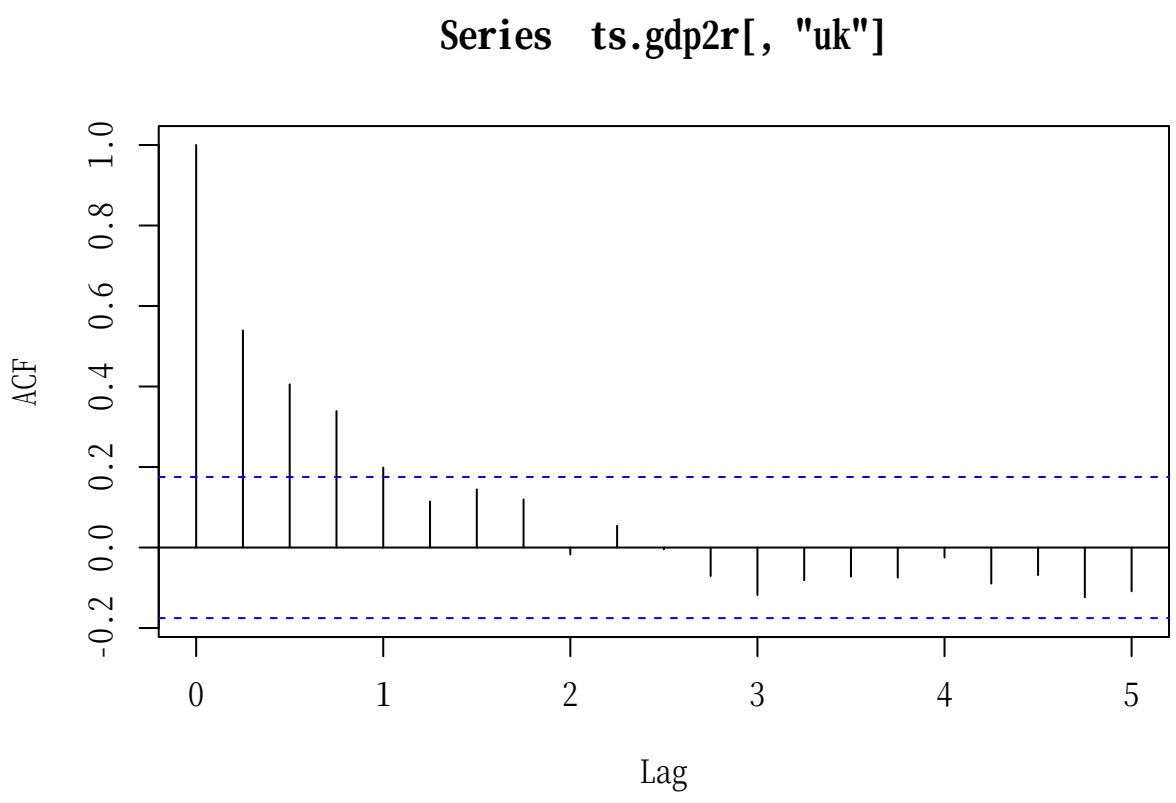
可以先取 $k_1<0$, $k_2>0$，来检验是否存在任何方向的因果性。如果结果显著，再取 $k_1=1$, $k_2\geq k_1$，检验是否有 y_t 对 x_t 序列的因果性；取 $k_2=-1$, $k_1\leq k_2$，检验是否有 x_t 对 y_t 序列的因果性；取 $k_1=k_2=0$，检验有无瞬时因果性。

这种方法比直接格兰格方法功效较低。

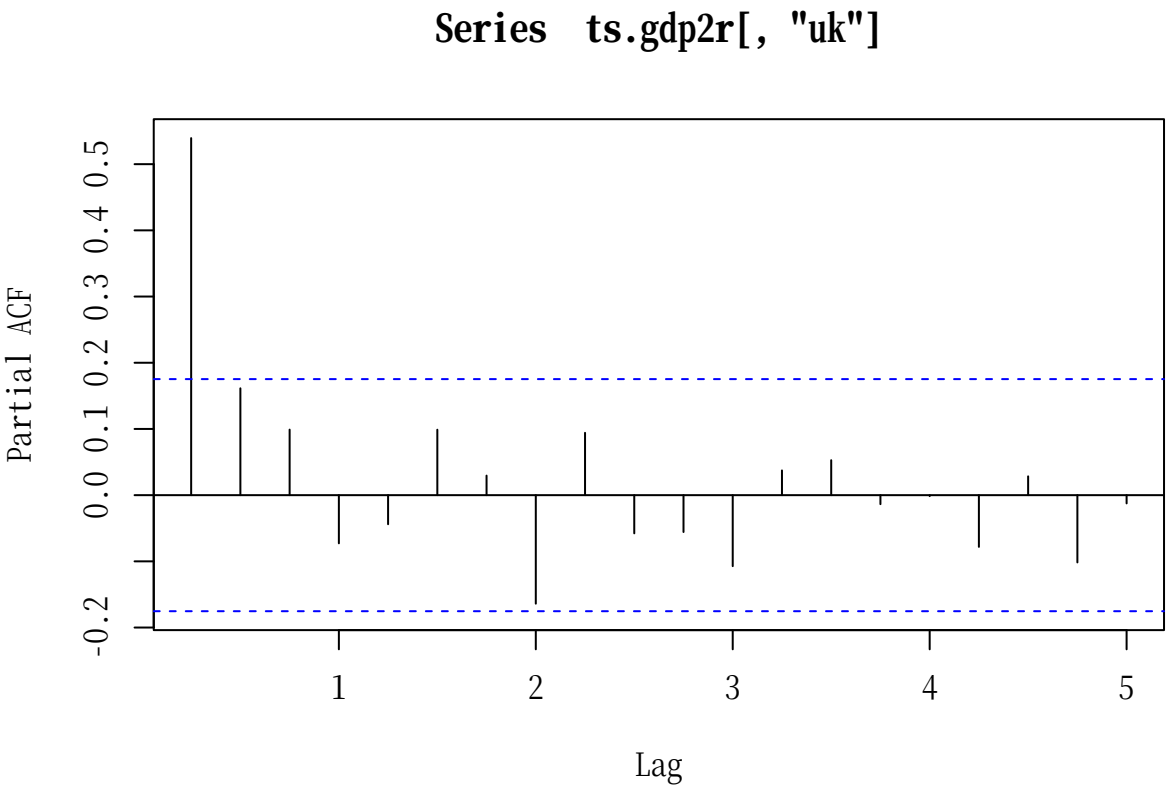
例 26.7. 仍考虑英国和美国的季度 GDP 对数增长率问题。

对英国的数据建立 ARMA 模型：

```
acf(ts.gdp2r[,"uk"])
```




```
pacf(ts.gdp2r[,"uk"])
```



试验模型 1：

```
mod.uk01 <- arima(ts.gdp2r[,"uk"], order=c(1,0,0)); mod.uk01
```

```
##
## Call:
## arima(x = ts.gdp2r[, "uk"], order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.5871     0.4925
## s.e.  0.0766     0.1250
##
## sigma^2 estimated as 0.339:  log likelihood = -109.97,  aic = 225.94
```

试验模型 2：

```
mod.uk02 <- arima(ts.gdp2r[,"uk"], order=c(4,0,0)); mod.uk02
```

```
##
## Call:
## arima(x = ts.gdp2r[, "uk"], order = c(4, 0, 0))
##
## Coefficients:
```

```
##          ar1     ar2     ar3      ar4  intercept
##       0.4897  0.0900  0.1894  -0.0773     0.4672
## s.e.  0.0920  0.1023  0.1048   0.0951     0.1635
##
## sigma^2 estimated as 0.3218:  log likelihood = -106.8,  aic = 225.6
```

试验模型 3：

```
mod.uk03 <- arima(ts.gdp2r[,"uk"], order=c(1,0,1)); mod.uk03
```

```
##
## Call:
## arima(x = ts.gdp2r[, "uk"], order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.7989  -0.3269     0.4645
## s.e.  0.0912   0.1354     0.1691
##
## sigma^2 estimated as 0.3274:  log likelihood = -107.84,  aic = 223.68
```

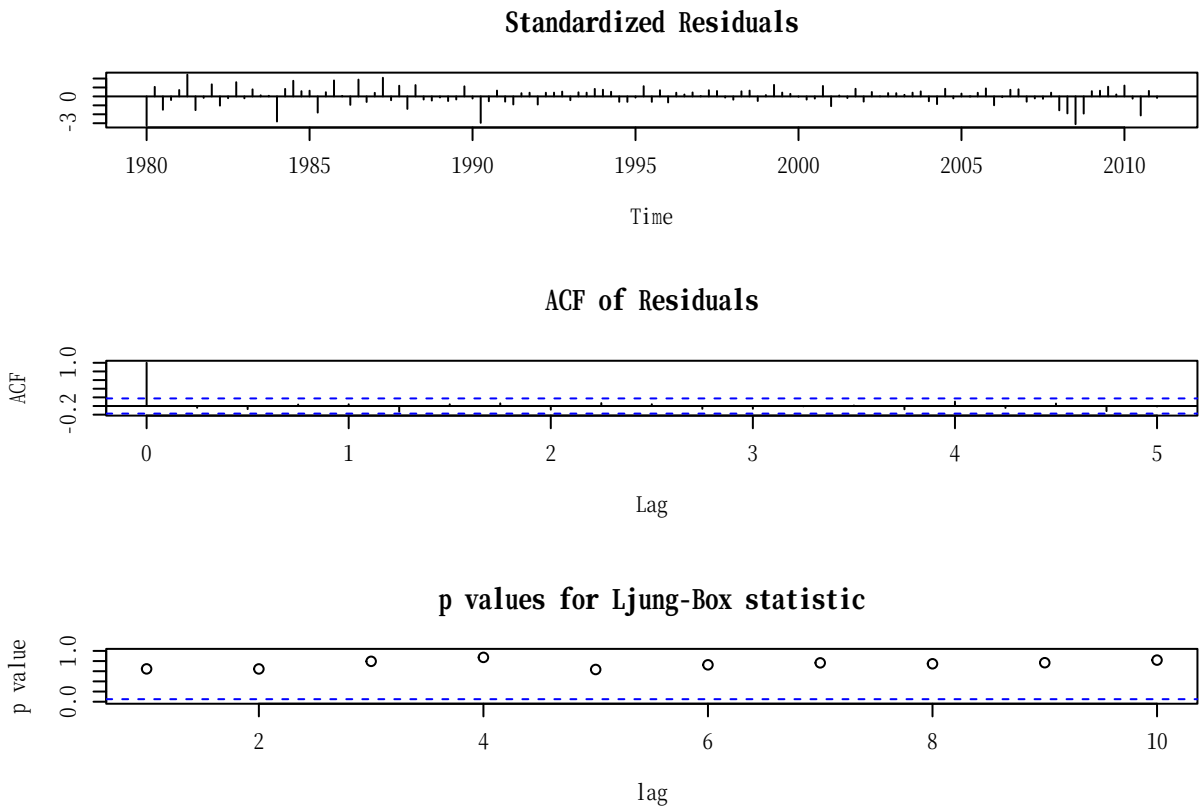
试验模型 4：

```
mod.uk04 <- arima(ts.gdp2r[,"uk"], order=c(1,0,1),
                  seasonal=list(order=c(1,0,1), frequency=4)); mod.uk04
```

```
##
## Call:
## arima(x = ts.gdp2r[, "uk"], order = c(1, 0, 1), seasonal = list(order = c(1,
##     0, 1), frequency = 4))
##
## Coefficients:
##          ar1      ma1    sar1     sma1  intercept
##       0.8585  -0.3906  0.6290  -0.8050     0.5044
## s.e.  0.0791   0.1259  0.1735   0.1228     0.1231
##
## sigma^2 estimated as 0.3144:  log likelihood = -105.5,  aic = 223.01
```

第 4 个模型是一个较好（从 AIC 看）的模型。对其进行残差诊断：

```
tsdiag(mod.uk04)
```



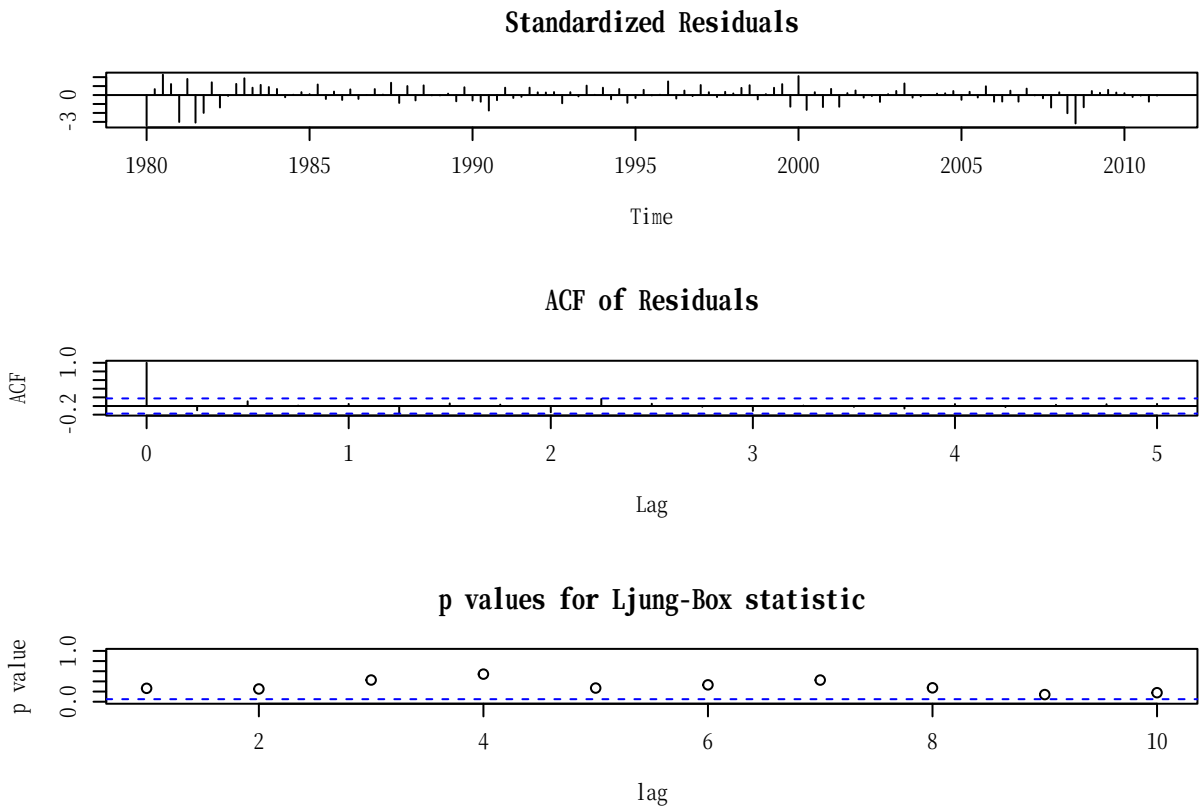
残差诊断通过。

对美国的 GDP 季度对数增长率建模，用类似模型：

```
mod.us01 <- arima(ts.gdp2r[,"us"], order=c(1,0,0)); mod.us01
```

```
##
## Call:
## arima(x = ts.gdp2r[, "us"], order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.4728     0.6259
## s.e.  0.0833     0.1180
##
## sigma^2 estimated as 0.4887:  log likelihood = -132.74,  aic = 271.48
```

```
tsdiag(mod.us01)
```



残差的白噪声检验可以通过。

求两个模型的残差的互相关系数：

```
ccfv <- ccf(mod.uk04$residuals, mod.us01$residuals, plot=FALSE)
ccfv
```

```
##
## Autocorrelations of series 'X', by lag
##
##  -4.25  -4.00  -3.75  -3.50  -3.25  -3.00  -2.75  -2.50  -2.25  -2.00  -1.75
##  0.102 -0.057  0.011 -0.061 -0.067 -0.137 -0.007  0.134 -0.030  0.013  0.074
##  -1.50  -1.25  -1.00  -0.75  -0.50  -0.25   0.00   0.25   0.50   0.75   1.00
##  0.076 -0.101  0.104  0.020 -0.111  0.294  0.278  0.048  0.044  0.239 -0.223
##   1.25   1.50   1.75   2.00   2.25   2.50   2.75   3.00   3.25   3.50   3.75
## -0.036  0.038  0.001  0.023  0.102  0.115 -0.118  0.069 -0.114 -0.046  0.112
##   4.00   4.25
##  0.016  0.033
```

R 的 `ccf(x, y)` 在滞后 k 处计算 $\rho(x_{t+k}, y_k)$，但是上面结果中的显示滞后阶数是除以频率的结果，单位是年，乘以 4 才是实际滞后阶数。

取 $k_1 = -4, k_2 = 4$，计算卡方统计量和 p 值：

```
stat <- ccfv$n.used * sum((ccfv$acf[
  4*ccfv$lag >= -4 - 1E-8 & 4*ccfv$lag <= 4 + 1E-8])^2)
pvalue <- 1 - pchisq(stat, df=9)
c(stat=stat, pvalue=pvalue)
```

```
##         stat       pvalue
## 3.727209e+01 2.353255e-05
```

结果显著，说明存单向、反馈或者瞬时的因果性。

取 $k_1 = 1, k_2 = 4$:

```
stat <- ccfv$n.used * sum((ccfv$acf[
  4*ccfv$lag >= 1 - 1E-8 & 4*ccfv$lag <= 4 + 1E-8])^2)
pvalue <- 1 - pchisq(stat, df=4)
c(stat=stat, pvalue=pvalue)
```

```
##         stat       pvalue
## 13.912630659  0.007579187
```

结果显著，说明美国对英国有因果性影响。

反过来检验，取 $k_1 = -4, k_2 = -1$:

```
stat <- ccfv$n.used * sum((ccfv$acf[
  4*ccfv$lag >= -4 - 1E-8 & 4*ccfv$lag <= -1 + 1E-8])^2)
pvalue <- 1 - pchisq(stat, df=4)
c(stat=stat, pvalue=pvalue)
```

```
##         stat       pvalue
## 13.712136383  0.008272803
```

结果显著，说明英国对美国有因果性影响。

检验有无瞬时因果性：

```
stat <- ccfv$n.used * sum((ccfv$acf[
  4*ccfv$lag >= - 1E-8 & 4*ccfv$lag <= 1E-8])^2)
pvalue <- 1 - pchisq(stat, df=1)
c(stat=stat, pvalue=pvalue)
```

```
##        stat      pvalue
## 9.647319356 0.001896281
```

结果显著，英国和美国有瞬时因果性。

26.6.3 Hsiao 方法

原始文献：

- CHENG HSIAO(1979), Autoregressive Modeling of Canadian Money and Income Data, Journal of the American Statistical Association 74, pp. 553-560.
- ARNOLD ZELLNER(1962), An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias, Journal of the American Statistical Association 57, pp. 348-368.

Hsiao(1979) 对直接格兰格方法进行了改进，给出了滞后项数的选取方法，步骤如下：

1. 对 y_t 定阶估计一元的 AR(k_1^*) 模型。
2. 在对 y_t 的预测中加入 x_t 的滞后项，求出最优的滞后项数 k_2^*。
3. 固定 k_2^* 的值，求 y_t 的滞后项数的最优值 $\bar{k}_1^*$。
4. 比较第 3 步和第 1 步的模型，如果第 3 步 AIC 值优于第 1 步，则认为 x_t 对 y_t 有因果作用，选用第 3 步的模型，否则选用第 1 步的模型。如此得到预报 y_t 的初步的模型。
5. 交换 x_t 和 y_t 的地位重复第 1-4 步，获得预报 x_t 的初步的模型。
6. 将预测 y_t 的模型和预测 x_t 的模型联立估计，允许其随机误差项有相关性，方法采用 Zellner(1962) 提出的表面不相关回归 (SUR) 方法。

注意第 4、5 步就可以得到因果性的结论，第 6 步是为了得到更精确的预测模型。

26.7 多元情形下的因果性检验

在最原始的 (C. W. J. Granger, 1969) 文章中定义因果性时，要求考虑到所有的到时刻 t 为止的信息。在仅有两个序列，考虑其相互的因果性时，如果没有其它的序列对其有影响，因果性检验的结果是可信的；如果有其他序列会有影响，因果性检验的结果可能是虚假的因果性或者虚假的无因果性。

这里考虑多个时间序列的因果性关系的检验问题。

26.7.1 直接格兰格方法

适用于二元时间序列的 Haugh-Pierce 方法用到两个序列之间的互相关函数，如果有三个序列，就无法计算三个序列整体的互相关，所有 Haugh-Pierce 方法无法推广到多于两个分量的情形。但是，仍可以利用格兰格的因果性定义来检验。在预报 y_t 时，如果扣除分量 x_t 的历史值的信息后预报误差比利用所有历史信息的预报误差显著增大，就说明 x_t 是 y_t 的格兰格原因；如果扣除若干个分量后预测误差显著增大，就说明这些分量联合起来是 y_t 的格兰格原因。

例如，三个序列 x_t, y_t, z_t，选取适当滞后项数 k_1, k_2, k_3，用普通最小二乘建立回归方程

$$y_t = \beta_0 + \sum_{k=1}^{k_1} \beta_{1k} y_{t-k} + \sum_{k=1}^{k_2} \beta_{2k} x_{t-k} + \sum_{k=1}^{k_3} \beta_{3k} z_{t-k}$$

则零假设 $H_0 : \beta_{21} = \cdots = \beta_{2k_2} = 0$ 表示 x_t 不是 y_t 的格兰格原因；零假设 $H_0 : \beta_{31} = \cdots = \beta_{2k_3} = 0$ 表示 z_t 不是 y_t 的格兰格原因；零假设 $H_0 : \beta_{21} = \cdots = \beta_{2k_2} = \beta_{31} = \cdots = \beta_{2k_3} = 0$ 表示 (x_t, z_t) 不是 y_t 的格兰格原因；

可以分别建立全集模型、简约模型并用 R 的 `anova()` 函数比较两个模型，如果有显著差异，则说明是格兰格原因，否则不是。

这样的直接格兰格方法可以用 Hsiao 方法进行改进，给出滞后项数的取法和步骤。

26.7.2 缺少第三个变量的影响

设对两个变量 x_t, y_t 做了因果性检验，但是有可能有第三个变量 z_t 对其有影响。会有怎样的影响？

- 仅使用 x_t, y_t 作的瞬时因果检验可能是真实的，也可能是虚假的，只有用完全模型才能确定。
- 如果仅使用 x_t, y_t 作的因果性检验结果是反馈关系，加入 z_t 之后可能变成单向的因果性。
- 如果仅使用 x_t, y_t 作的因果性检验结果是单向关系，则加入 z_t 一般不变。以上三条见 GEBHARD KIRCHGÄSSNER (1981)。
- 如果 x_t, y_t 没有因果关系，但都依赖于 z_t，仅使用作 x_t, y_t 的因果性检验结果可能有虚假的因果性。见 CHRISTOPHER A. SIMS (1977) 文章。

总之，两变量的因果性检验如果结果是单向的因果性，结果相对可信；如果是反馈关系，有可能是虚假的。但是，多数情况下这种由于缺失重要变量或者测量误差引起的虚假反馈因果关系是比较弱的，也有可能会反映成单向；单向因果关系是承认 H_0 得到的，而承认 H_0 具有不可控制的第二类错误概率。所以因果性检验结果要慎重对待。

参考：

- GEBHARD KIRCHGÄSSNER(1981), Einige neuere statistische Verfahren zur Erfassung kausaler Beziehungen zwischen Zeitreihen, Darstellung und Kritik, Vandenhoeck und Ruprecht, Göttingen
- CHRISTOPHER A. SIMS(1977), Exogeneity and Causal Ordering in Macroeconomic Models, in: FEDERAL RESERVE BANK OF MINNEAPOLIS (ed.), New Methods in Business Cycle Research: Proceedings from a Conference, Minneapolis, pp. 23-44.

Part V

状态空间模型

Chapter 27

局部水平模型

状态空间模型是时间序列分析领域中一类强大、灵活、多样的模型，配合卡尔曼滤波技术，可以涵盖 ARIMA 模型、许多非平稳的、带有外生变量的模型，比前面所述的线性时间序列模型更为灵活。R 扩展包 statespacer 实现了许多基于线性高斯状态空间模型的模型，并且可以自定义模型。

参考：

- (Tsay, 2010)
- (Durbin & Koopman, 2012)
- (Beijers, 2020)

作为入门，先介绍一个局部水平模型。这个模型很简单，所以可以用来演示状态空间模型的表示和估计。

27.1 模型

设 $\{y_t, t = 1, 2, \dots, T\}$ 为时间序列，满足如下模型

$$y_t = \mu_t + e_t,\ \{e_t\} \sim \text{iid N}(0, \sigma_e^2),\ t = 1, 2, \dots, n, \tag{27.1}$$

$$\mu_{t+1} = \mu_t + \eta_t,\ \{\eta_t\} \sim \text{iid N}(0, \sigma_\eta^2), \tag{27.2}$$

其中 $\{e_t\}$ 与 $\{\eta_t\}$ 相互独立，初始值 μ_1 为给定值或者是服从正态分布的随机变量，且与 $\{e_t, \eta_t, t > 0\}$ 相互独立。称 $\{\mu_t\}$ 为 $\{y_t\}$ 的水平，模型中 $\{y_t\}$ 可观测而 $\{\mu_t\}$ 不可观测。

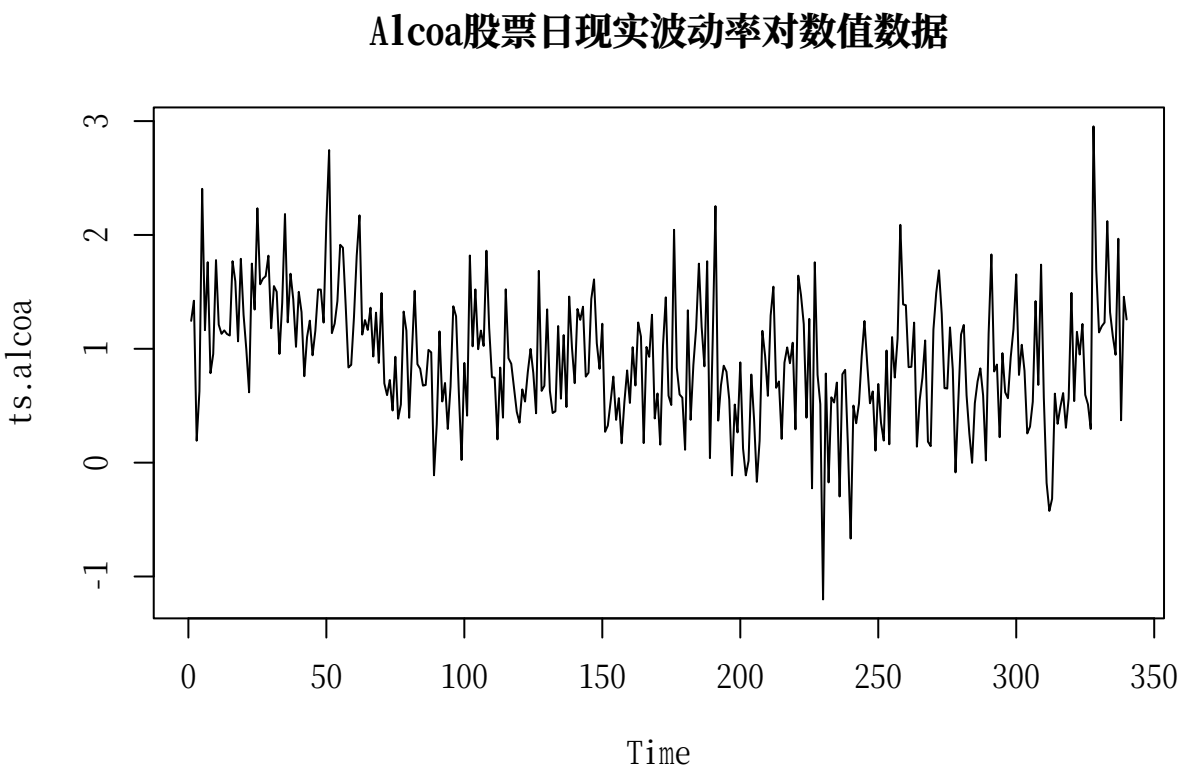
方程(27.1)-(27.2)是线性高斯状态空间模型的一个特例。$\{\mu_t\}$ 代表系统在 t 时刻所处的状态，不可观测，方程(27.2)称为**状态方程**，描述了系统的内在演变规律；$\{y_t\}$ 代表系统在 t 时刻的观测值或输出值，(27.1)称为**观测方程**，$\{e_t\}$ 是观测误差，仅影响到 t 时刻，是一个瞬态的误差或噪声。

这个模型称为局部水平模型，也是“结构时间序列模型”的一个特例。

考虑 Alcoa 股票日现实波动率数据，
时间期间为 2003-01-02 到 2004-05-07，
共 340 个观测。
日现实波动率是用交易日内每隔 10 分钟的对数收益率的平方和计算的。

```
da <- read_table("aa-3rv.txt",
  col_names = FALSE,
  show_col_types = FALSE)
ts.alcoa <- ts(log(da[[2]]))
```

```
plot(ts.alcoa, main="Alcoa 股票日现实波动率对数值数据")
```



可以用局部水平模型获得 $\{\mu_t\}$，作为去噪声的波动率估计。

27.2 局部水平模型与 ARIMA 模型的关系

注意到

$$y_t - y_{t-1} = \eta_{t-1} + e_t - e_{t-1},$$

记 $\xi_t = \eta_{t-1} + e_t - e_{t-1}$，易见 ξ_t 的自相关函数在 1 以后截尾，所以 ξ_t 服从一个 MA(1) 模型。由多元正态分布的性质可知存在 $\{a_t\} \sim \mathrm{N}(0, \sigma_a^2)$ 和 $|\theta| \leq 1$ 使得

$$y_t - y_{t-1} = a_t + \theta a_{t-1},$$

从而 y_t 服从一个高斯的 ARIMA(0,1,1) 模型。

由

$$\begin{aligned}\mathrm{Var}(\xi_t) =& 2\sigma_e^2 + \sigma_\eta^2 = \sigma_a^2(1+\theta^2),\\ \mathrm{Cov}(\xi_t, \xi_{t-1}) =& -\sigma_e^2 = \theta\sigma_a^2,\end{aligned}$$

可以求解出 θ 且使得 $-1 < \theta < 0$:

$$\theta = \frac{b + \sqrt{b^2+4}}{2}, \text{ 其中} b = -2 - \frac{\sigma_\eta^2}{\sigma_e^2} < -2.$$

反过来，如果 y_t 服从一个 ARIMA(0,1,1) 模型且 θ 为负值，则可以将其写成(27.1)-(27.2)的形式。如果 θ 为正值，可以写成 $\sigma_e = 0$ 的形式。

例 27.1. 对 Alcoa 股票日现实波动率对数值数据用 ARIMA 建模。

```
mod <- arima(ts.alcoa,
  order=c(0,1,1))
mod
```

```
##
## Call:
## arima(x = ts.alcoa, order = c(0, 1, 1))
##
## Coefficients:
##           ma1
##       -0.8582
## s.e.   0.0397
##
## sigma^2 estimated as 0.2688:  log likelihood = -258.98,  aic = 521.95
```

得到的 ARIMA(0,1,1) 模型为:

$$(1-B)y_t = (1 - 0.8582B)a_t,\ a_t \sim \text{WN}(0, 0.2688).$$

通过 $\sigma_e^2, \sigma_\eta^2$ 与 θ, σ_a^2 的关系，可以解出

$$\sigma_e^2 = -\theta\sigma_a^2 = 0.2307, \quad \sigma_\eta^2 = \sigma_a^2(1+\theta^2) - 2\sigma_e^2 = 0.0054$$

可见噪声严重超过了信号的扰动。

下面用 R 的 statespacer 扩展包估计局部水平模型。

```
library(statespacer)
ssr1 <- statespacer(
  y = cbind(as.vector(ts.alcoa)),
  local_level_ind = TRUE,
  initial = rep(0.5*log(var(ts.alcoa)), 2),
  verbose = TRUE)
```

```
## Starting the optimisation procedure at: 2022-03-23 16:33:40
```

```
## initial  value 1.022439
## iter  10 value 0.765458
## iter  20 value 0.764395
```

```
## final  value 0.764395
## converged

## Finished the optimisation procedure at: 2022-03-23 16:33:40

## Time difference of 0.00728106498718262 secs
```

上面的程序中输入数据需要是矩阵形式，每列为一个时间序列分量，一元时间序列也需要输入为仅有一列的矩阵。选项 `local_level_ind = TRUE` 表示有局部水平成分。`initial` 给出超参数初值选取，为了保证 σ_e^2 和 σ_η^2 估计为正数，模型中使用了其对数值作为参数，所以初值取了对数。

查看估计的 σ_e^2 和 σ_η^2:

```
c(sigmasqr_e = ssr1$system_matrices$H$H,
  sigmasqr_level = ssr1$system_matrices$Q$level)
```

```
##     sigmasqr_e sigmasqr_level
##    0.230623632    0.005404681
```

与从 ARIMA 模型换算的结果基本相同。statespacer 利用了 (Durbin & Koopman, 2012) 的模型和记号，H 表示观测方程误差项方差阵，Q 表示系统方程误差项方差阵。设 `statespacer()` 的输出为 `ssr`，这是一个列表，其中 `ssr$system_matrices$H$H` 是观测方程的方差阵估计，这里即 σ_e^2，但保存成 1×1 矩阵；`ssr$system_matrices$Q` 是与系统方程方差阵有关的输出，而局部水平的状态方程对应的误差项的方差阵则为 `ssr$system_matrices$Q$level`，这里就是 σ_η^2，保存成 1×1 矩阵。

27.3 滤波、平滑和预报

对状态空间模型如(27.1)-(27.2)，输入数据 $\{y_t, t = 1, 2, \dots, n\}$，如果认为模型中参数（这里是 σ_e^2 和 σ_η^2）已知，经常讨论如下的统计推导问题：

- **滤波**：从 $\{y_1, \dots, y_t\}$ 估计 μ_t；
- **平滑**：从 $\{y_1, \dots, y_n\}$ 估计 $\{\mu_1, \mu_2, \dots, \mu_n\}$；
- **预报**：从 $\{y_1, \dots, y_t\}$ 估计 μ_{t+h} 或 $y_{t+h}(h > 0)$。

当参数已知，记 $y_{1:t} = (y_1, \dots, y_t)$, 易见：

- 滤波解为 $E(\mu_t|y_{1:t})$；
- 平滑解为 $E(\mu_t|y_{1:n})$；
- 预报解为 $E(\mu_{t+h}|y_{1:t})$ 或 $E(y_{t+h}|y_{1:t})$。

设 $\mu_1 \sim \mathrm{N}(a_1, P_1)$，且与扰动序列独立。由正态分布的性质，状态空间模型(27.1)-(27.2)是高斯过程，其条件分布仍为高斯分布，所以 $\mu_t|y_{1:s}$ 和 $y_t|y_{1:s}$ 仍服从高斯分布（多元正态分布），其条件期望为为最小均方误差估计，也是线性无偏估计。

μ_t 在 $y_{1:s}$ 下的条件分布完全由条件期望和条件方差决定。记 $\mu_{t|s} = E(\mu_t|y_{1:s})$, 记 $\Sigma_{t|s} = \mathrm{Var}(\mu_t|y_{1:s})$。记 $y_{t|s} = E(y_t|y_{1:s})$。特别地，记 $a_t = E(\mu_t|y_{1:t-1})$, $P_t = \mathrm{Var}(\mu_t|y_{1:t-1})$。由高斯分布性质，条件方差都是非随机的。

记

$$v_t = y_t - E(y_t|y_{1:t-1}),$$

这是对 y_t 做最优一步预报时的误差，显然 $Ev_t = 0$，令

$$F_t = Ev_t^2 = \mathrm{Var}(v_t),$$

由多元正态分布性质，v_t 与 $y_{1:t-1}$ 独立，所以也有

$$\begin{aligned} F_t =& \mathrm{Var}(v_t) = E(v_t^2) = E(v_t^2|y_{1:t-1}) \\ =& E[(y_t - E(y_t|y_{1:t-1}))^2|y_{1:t-1}] \\ =& \mathrm{Var}(y_t|y_{1:t-1}). \end{aligned}$$

于是由观测方程可得

$$y_{t|t-1} = E(y_t|y_{1:t-1}) = E(\mu_t + e_t|y_{1:t-1}) = \mu_{t|t-1} = a_t, \tag{27.3}$$

$$v_t = y_t - y_{t|t-1} = y_t - a_t, \tag{27.4}$$

$$F_t = \mathrm{Var}(y_t - y_{t|t-1}|y_{1:t-1}) = \mathrm{Var}(\mu_t + e_t - \mu_{t|t-1}|y_{1:t-1}) \tag{27.5}$$

$$= \mathrm{Var}(\mu_t - \mu_{t|t-1}|y_{1:t-1}) + \mathrm{Var}(e_t|y_{1:t-1}) \tag{27.6}$$

$$= \Sigma_{t|t-1} + \sigma_e^2 = P_t + \sigma_e^2. \tag{27.7}$$

由多元正态分布性质可知预报误差 v_t 与 $y_{1:s}(s < t)$ 独立，所以

$$\mathrm{Cov}(v_t, y_s) = 0,\ t > s.$$

从而 $\{v_t, t = 1, 2, \ldots, n\}$ 相互独立。由多元正态性质，

$$\sigma(y_1, \ldots, y_{t-1}, y_t) = \sigma(y_{1:t-1}, v_t).$$

其中 $\sigma(\cdot)$ 表示由其中的随机变量所生成的最小 σ 代数。

对随机向量 X, Y，记 $\mu_X = EX$, $\Sigma_{XX} = \mathrm{Var}(X)$，$\Sigma_{XY} = \mathrm{Cov}(X, Y)$。从多元正态分布性质有如下定理 (Tsay, 2010) P.562：

定理 27.1. 设随机向量 X, Y, Z 的联合分布为多元正态分布，每一个的协方差阵都非退化，$\Sigma_{XZ} = 0$。则有如下性质：

$$\begin{aligned} &(1)\ E(Y|X) = \mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(X - \mu_X); \\ &(2)\ Var(Y|X) = \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}; \\ &(3)\ E(Y|X, Z) = E(Y|X) + E(Y - \mu_Y|Z) \\ &\qquad\qquad = E(Y|X) + \Sigma_{YZ}\Sigma_{ZZ}^{-1}(Z - \mu_Z); \\ &(4)\ Var(Y|X, Z) = Var(Y|X) - \Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{ZY} \\ &\qquad\qquad = \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY} - \Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{ZY}. \end{aligned}$$

另外，$Y - E(Y|X)$ 与 X 独立，所以

$$Var(Y|X) = E\left[(Y - E(Y|X))^2|X\right] = E\left[(Y - E(Y|X))^2\right],$$

等于 X 对 Y 的最优估计的均方误差。

27.4 卡尔曼滤波

卡尔曼滤波是一种递推算法，对 $t=1,2,...$，基于 $\mu_t|y_{1:t-1}$ 的条件分布和新得到的观测值 y_t，求 $\mu_t|y_{1:t}$ 条件分布，这等于 $\mu_t|(y_{1:t-1},v_t)$ 条件分布，只要求条件高斯分布的期望和方差。

由前一节，$\mu_t|y_{1:t-1}\sim \mathrm{N}(a_t,P_t)$, $v_t\sim \mathrm{N}(0,F_t)$ 与 $y_{1:t-1}$ 独立。注意 $\{e_t\}$ 与 $\{\eta_t\}$ 独立所以 $\{e_t\}$ 与 $\{\mu_t\}$ 独立，利用定理27.1，条件期望为

$$\begin{aligned}&\mu_{t:t}=E(\mu_t|y_{1:t})\\=&E(\mu_t|y_{1:t-1},v_t)\\=&E(\mu_t|y_{1:t-1})+E(\mu_t-E\mu_t|v_t)\\=&a_t+\frac{\mathrm{Cov}(\mu_t-E\mu_t,v_t)}{\mathrm{Var}(v_t)}v_t\\=&a_t+\frac{\mathrm{Cov}(\mu_t,v_t)}{F_t}v_t.\end{aligned}$$

其中

$$\begin{aligned}&\mathrm{Cov}(\mu_t,v_t)\\=&E(\mu_t v_t)\quad (\text{注意}Ev_t=0)\\=&E[\mu_t(y_t-a_t)]\\=&E[\mu_t(\mu_t+e_t-a_t)]\\=&E[\mu_t(\mu_t-a_t)]+E[\mu_t e_t]\\=&E[\mu_t(\mu_t-a_t)]+0\\=&E\left\{E\left[(\mu_t-a_t)^2|y_{1:t-1}\right]\right\}\\=&E\{P_t\}=P_t.\end{aligned}$$

于是

$$E(\mu_t|y_{1:t-1},v_t)=a_t+\frac{P_t}{F_t}v_t,$$

记

$$K_t=\frac{P_t}{F_t}=\frac{P_t}{P_t+\sigma_e^2},\tag{27.8}$$

称 K_t 为**卡尔曼增益**。可以将条件期望写成

$$E(\mu_t|y_{1:t-1},v_t)=a_t+K_t v_t,$$

这个公式将 $y_1,\dots,y_{t-1},y_t$ 对 μ_t 的最优预报（滤波）公式分解为两部分，第一部分是 $y_1,\dots,y_{t-1}$ 对 μ_t 的最优预报，第二部分是 y_t 新增的信息 v_t 对 μ_t 的最优预报，v_t 的系数为卡尔曼增益 K_t，最优预报是线性的。

再来求条件方差 $\mathrm{Var}(\mu_t|y_{1:t-1},v_t)$。由定理27.1,

$$\begin{aligned}&\Sigma_{t|t}=\mathrm{Var}(\mu_t|y_{1:t})\\=&\mathrm{Var}(\mu_t|y_{1:t-1},v_t)\\=&\mathrm{Var}(\mu_t|y_{1:t-1})-\frac{[\mathrm{Cov}(\mu_t,v_t)]^2}{\mathrm{Var}(v_t)}\\=&P_t-\frac{P_t^2}{F_t}=P_t(1-\frac{P_t}{F_t})\\=&P_t(1-K_t).\end{aligned}$$

记

$$L_t = 1 - K_t = \frac{\sigma_e^2}{P_t + \sigma_e^2}, \tag{27.9}$$

则有

$$\Sigma_{t|t} = P_t L_t.$$

因此，滤波公式为：

$$\mu_{t|t} = a_t + K_t v_t, \tag{27.10}$$
$$\Sigma_{t|t} = P_t(1 - K_t), \tag{27.11}$$
$$K_t = \frac{P_t}{F_t}. \tag{27.12}$$

有了 $\mu_t|y_{1:t}$ 分布后，可以给出一步预测的条件分布 $\mu_{t+1}|y_{1:t}$，这也只需要计算条件期望和条件方差：

$$a_{t+1} = \mu_{t+1|t} = E(\mu_{t+1}|y_{1:t}) = E(\mu_t + \eta_t|y_{1:t}) \tag{27.13}$$
$$= \mu_{t|t} + 0 = \mu_{t|t}, \tag{27.14}$$
$$P_{t+1} = \Sigma_{t+1|t} = \mathrm{Var}(\mu_{t+1}|y_{1:t}) = \mathrm{Var}(\mu_t|y_{1:t}) + \mathrm{Var}(\eta_t|y_{1:t}) \tag{27.15}$$
$$= \Sigma_{t|t} + \sigma_\eta^2. \tag{27.16}$$

在上式推导中要注意 η_t 是 μ_{t+1} 所对应的状态方程的误差项，η_t 与 $y_{1:t}$ 和 $\mu_1, \dots, \mu_t$ 独立。

(27.10)–(27.16)构成了卡尔曼滤波的一轮操作。在下一轮中，输入了新的观测 y_{t+1} 后，计算 $\mu_{t+1}|y_{1:t+1}$ 的均值 $\mu_{t+1|t+1}$ 和方差 $\Sigma_{t+1|t+1}$，再计算 $\mu_{t+2}|Y_{t+1}$ 的均值 a_{t+2} 和方差 P_{t+2}，……。

各个关键变量的含义汇总：

- $y_{1:t} = \{y_1, y_2, \dots, y_t\}$。
- $a_t = E(\mu_t|y_{1:t-1})$, $P_t = \mathrm{Var}(\mu_t|y_{1:t-1})$, μ_t 一步预报分布为 $\mu_t|y_{1:t-1} \sim \mathrm{N}(a_t, P_t)$。
- $\mu_{t|t} = E(\mu_t|y_{1:t})$, $\Sigma_{t|t} = \mathrm{Var}(\mu_t|y_{1:t})$，$\mu_t$ 的滤波分布为 $\mu_t|y_{1:t} \sim \mathrm{N}(\mu_{t|t}, \Sigma_{t|t})$。
- $v_t = y_t - E(y_t|y_{1:t-1})$ 是 y_t 的一步预报误差，$F_t = E(v_t^2) = \mathrm{Var}(y_t|y_{1:t-1})$，$v_t \sim \mathrm{N}(0, F_t)$ 与 $y_{1:t-1}$ 独立。
- $K_t = \frac{P_t}{F_t}$ 称为 Kalman 增益，是用 $y_{1:t-1}, v_t$ 对 μ_t 作最优线性预测时 v_t 的系数。
- y_t 的一步预报分布为 $y_t|y_{1:t-1} \sim \mathrm{N}(a_t, F_t)$。

设初始状态 μ_1 服从 $\mathrm{N}(\mu_{1|0}, \Sigma_{1|0}) = \mathrm{N}(a_1, P_1)$, 将(27.10)-(27.11) 代入到(27.14)-(27.16)中，得卡尔曼滤波过程如下：

$$\begin{cases} v_t = y_t - a_t, \\ F_t = P_t + \sigma_e^2, \\ K_t = P_t/F_t, \\ a_{t+1} = \mu_{t+1|t} = a_t + K_t v_t, \\ P_{t+1} = \Sigma_{t+1|t} = P_t(1 - K_t) + \sigma_\eta^2, \ t = 1, 2, \dots, n. \end{cases} \tag{27.17}$$

算法中初始分布参数 $a_1 = \mu_{1|0}$ 和 $P_1 = \Sigma_{1|0}$ 的选取有很大影响，后面将专门说明。

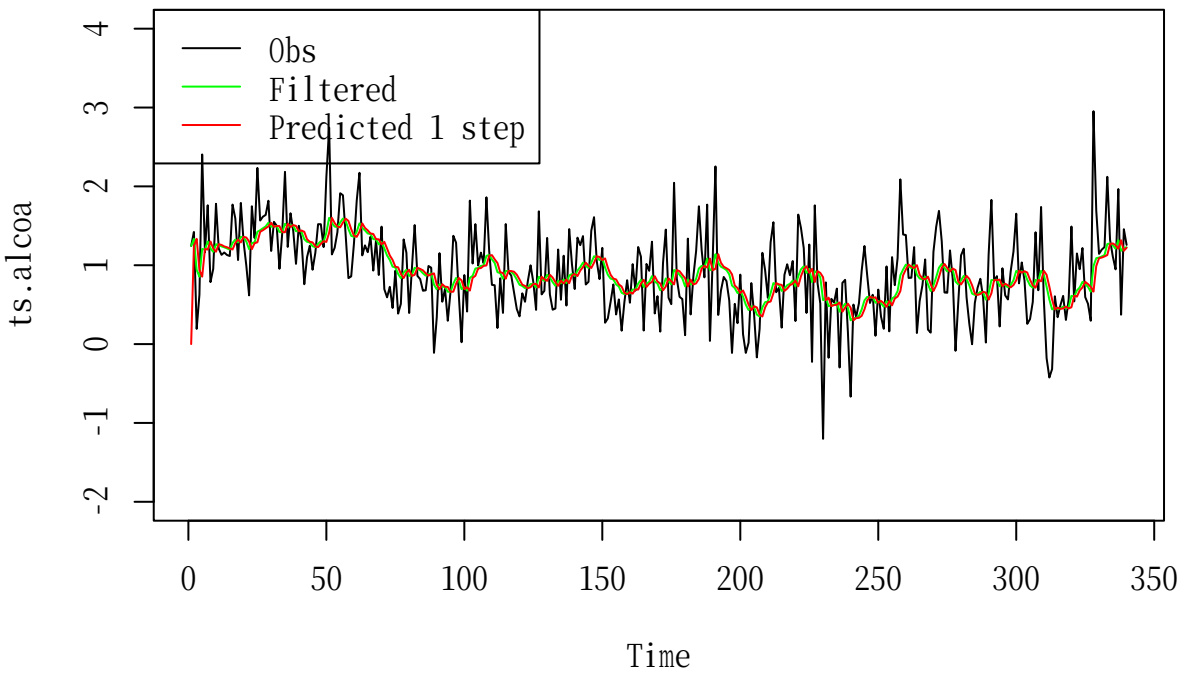
(27.17)的卡尔曼滤波算法在给定参数 σ_e^2 和 σ_η^2 和初始分布参数 a_1, P_1 后可以迭代计算 $(y_1, y_2, \dots, y_n)$ 的联合密度，因此可以用来进行最大似然估计。

如果不假定模型中的 μ_t, y_t 都服从多元正态分布，将滤波问题看成是将 $y_{1:t}$ 看成已知量，用 $y_{1:t}$ 对 μ_t 做最优线性无偏估计 (Minimum Variance Linear Unbiased Estimate, MVLUE，也称为 BLUE 或 BLUP) 的问题，结果将得到完全相同的公式。所以卡尔曼滤波可以看成是递推的最优线性无偏估计算法。

如果将模型中的 μ_t 看成是随机参数，将 y_t 看成是观测值，从先验分布 $\mu_t|y_{1:t-1}$ 到后验分布 $\mu_t|(y_{1:t-1}, y_t)$ 的计算公式也和卡尔曼滤波公式完全相同。参见 (Durbin & Koopman, 2012) 第 2.2 节。

对前面 Alcoa 现实波动率对数值数据，利用估计的模型参数进行卡尔曼滤波计算，将原始序列与滤波结果（$\mu_{t|t}$，图中绿色线），一步预测结果（$y_{t|t-1} = a_t$，图中红色线）同时显示：

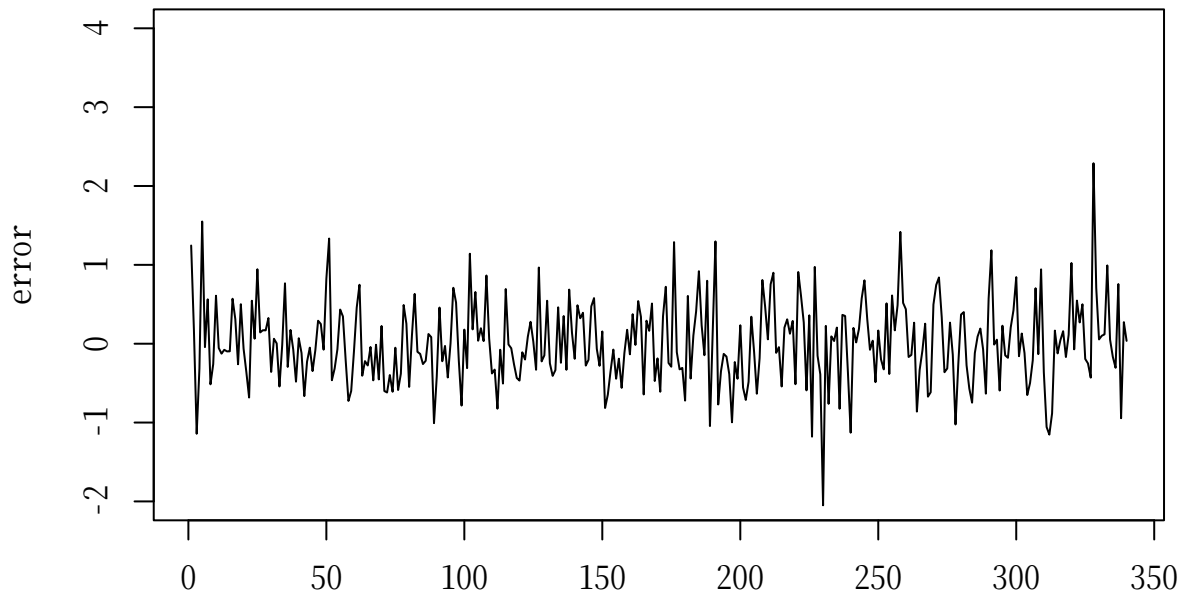
```
plot(ts.alcoa, ylim=c(-2, 4))
lines(as.vector(time(ts.alcoa)),
  ssr1$filtered$level, col="green")
lines(as.vector(time(ts.alcoa)),
  ssr1$predicted$yfit, col="red")
legend("topleft", lty=1, col=c("black", "green", "red"),
  legend=c("Obs", "Filtered", "Predicted 1 step"))
```



滤波值和一步预测值的序列都比较光滑。程序中，拟合结果的 `$filtered` 成分保存滤波结果，`$filtered$level` 为 $\{\mu_t\}$ 的滤波结果 $\{\mu_{t|t}\}$；拟合结果的 `$predicted` 成分保存一步预报结果，`$predicted$yfit` 保存对 y_t 的一步预报 $\{y_{t|t-1}\}$。`$filtered` 中还有 `$filtered$a` 表示状态的滤波，这里即 $\{\mu_{t|t}\}$，`$filtered$P` 表示滤波方差，即 $\{\Sigma_{t|t}\}$，是一个 $1 \times 1 \times n$ 数组，n 为时间序列观测长度。`$predicted` 中还有 `$predicted$v`，即 $\{v_t\}$，`$predicted$Fmat` 即 $\{F_t\}$，`$predicted$a` 即 $\{a_t\}$，`$predicted$P` 即 $\{P_t\}$。

一步预测误差（v_t）的图形：

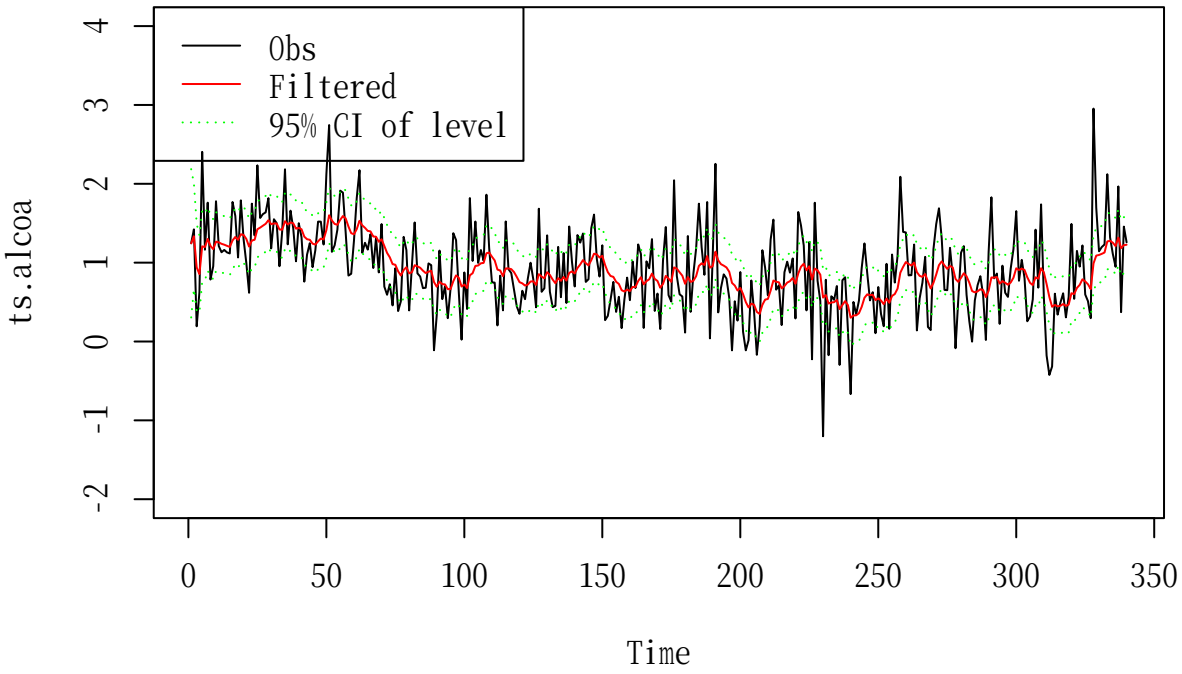
```
plot(as.vector(time(ts.alcoa)),
  ssr1$predicted$v,
  type="l", ylim=c(-2, 4),
  xlab="", ylab="error")
```



一步预测误差较大，这可能是因为用高频数据计算波动率，本身就有较大的观测噪声。

滤波的水平 μ_t 及其 95% 置信区间的图形：

```
plot(ts.alcoa, ylim=c(-2, 4))
lines(as.vector(time(ts.alcoa)),
  ssr1$filtered$level, col="red")
lines(as.vector(time(ts.alcoa)),
  ssr1$filtered$level + 1.96*sqrt(ssr1$filtered$P[1,1,]),
  col="green", lty=3)
lines(as.vector(time(ts.alcoa)),
  ssr1$filtered$level - 1.96*sqrt(ssr1$filtered$P[1,1,]),
  col="green", lty=3)
legend("topleft", lty=c(1,1,3), col=c("black", "red", "green"),
  legend=c("Obs", "Filtered", "95% CI of level"))
```



27.5 一步预报误差

一步预报误差序列 $\{v_t\}$ 在滤波算法和参数估计中起到重要作用。从初值 μ_1 的分布 $N(a_1, P_1)$ 出发，可以递推计算 $\{v_t, t = 1, 2, \ldots, n\}$，$v_t$ 是 $y_1, y_2, \ldots, y_t$ 的线性组合，如：

$$\begin{aligned} v_1 =& y_1 - a_1, \\ v_2 =& y_2 - a_2 = y_2 - a_1 - K_1(y_1 - a_1), \\ v_3 =& y_3 - a_3 = y_3 - a_1 - K_2(y_2 - a_1) - K_1(1 - K_2)(y_1 - a_1), \\ & \ldots\ldots \end{aligned}$$

将从 y 到 v 的这些线性变换写成矩阵形式，令 $Y_n = (y_1, \ldots, y_n)^T$, $v = (v_1, \ldots, v_n)^T$，1_T 表示元素都等于 1 的 n 维列向量，则

$$v = K(Y_n - a_1 1_n), \tag{27.18}$$

其中

$$K = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ k_{21} & 1 & 0 & \cdots & 0 \\ k_{31} & k_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n1} & k_{n2} & k_{n3} & \cdots & 1 \end{pmatrix},$$

而 $k_{i,i-1} = -K_{i-1}$, $i = 2, 3, \ldots, n$; 对 $i = 3, 4, \ldots, T$ 和 $j = 1, 2, \ldots, i-2$，有 $k_{ij} = -(1 - K_{i-1})(1 - K_{i-2}) \ldots (1 - K_{j+1})K_j$。

从卡尔曼滤波的公式可以看出，卡尔曼增益 $K_t, t = 1, 2, \dots, n$ 并不依赖于观测值 $\{y_t\}$，只依赖于参数 σ_η^2, σ_e^2 以及初始分布方差 P_1，是非随机的。

$\{v_t\}$ 是相互独立的随机变量序列。事实上，$(y_1, y_2 \dots, y_n)$ 的联合密度函数为

$$p_y(y_1, y_2, \dots, y_n) = p_y(y_1) \prod_{t=2}^{n} p_y(y_t|Y_{t-1}),$$

从(27.17)看出，从 $(y_1, y_2 \dots, y_n)$ 到 $(v_1, v_2 \dots, v_n)$ 的变换的雅科比行列式绝对值等于 1。由多元的连续性随机向量变换的密度公式可知

$$\begin{aligned} p_v(v_1, v_2, \dots, v_n) &= p_y(y_1, y_2, \dots, y_n) = p_y(y_1) \prod_{t=2}^{n} p_y(y_t|Y_{t-1}) \\ &= p_v(v_1) \prod_{t=2}^{n} p_v(v_t) = \prod_{t=1}^{n} p_v(v_t), \end{aligned}$$

其中 $p_y(y_1)$ 表示 y_1 的分布密度，$p_v(v_1)$ 表示 v_1 的分布密度，因为 $y_1 = \mu_1 + e_1 \sim \mathrm{N}(a_1, P_1 + \sigma_e^2)$，而 $v_1 = y_1 - y_{1|0} = y_1 - \mu_{1|0} \sim \mathrm{N}(0, P_1 + \sigma_e^2)$，所以

$$p_y(y_1) = \Phi'(\frac{y_1 - a_1}{\sqrt{P_1 + \sigma_e^2}}) = \Phi'(\frac{v_1}{\sqrt{P_1 + \sigma_e^2}}) = p_v(v_1),$$

对 $t \geq 2$，$y_t|Y_{t-1} \sim \mathrm{N}(a_t, F_t)$，$v_t = y_t - y_{t|t-1} \sim \mathrm{N}(0, F_t)$，$F_t = P_t + \sigma_e^2$，所以

$$p_y(y_t|Y_{t-1}) = \Phi'(\frac{y_t - a_t}{\sqrt{F_t}}) = \Phi'(\frac{v_t}{\sqrt{F_t}}) = p_v(v_t),$$

其中 $\Phi(\cdot)$ 表示标准正态分布函数。

从上述的 $(v_1, v_2, \dots, v_n)$ 的联合密度可知各个分量相互独立，均值为 0，v_t 的方差为 $F_t = P_t + \sigma_e^2$。

从另一个角度看，v 是对 Y_n 的正交化，类似于 Gram-Schimdt 正交化方法，$(v_1, \dots, v_t)$ 与 $(y_1, \dots, y_t)$ 可以互相线性表示。记 $\mathrm{Var}(Y_n) = \Omega$，由(27.17)可知

$$\begin{aligned} \mathrm{Var}(v) &= \mathrm{diag}(F_1, F_2, \dots, F_n) = K\Omega K^T, \\ \Omega &= K^{-1}\, \mathrm{diag}(F_1, F_2, \dots, F_n)\, (K^{-1})^T, \end{aligned}$$

其中 K^{-1} 为下三角矩阵且对角线元素都等于 1，这构成了对 y 的方差阵 Ω 的 Cholesky 分解。

27.6 状态的一步预报误差

令

$$\zeta_t = \mu_t - \mu_{t|t-1} = \mu_t - E(\mu_t|y_{1:t-1}), \tag{27.19}$$

则

$$\mathrm{Var}(\zeta_t|y_{1:t-1}) = E\zeta_t^2 = \mathrm{Var}(\zeta_t) = P_t,$$

由卡尔曼滤波公式得

$$v_t = y_t - \mu_{t|t-1} = \mu_t + e_t - \mu_{t|t-1} = \zeta_t + e_t,$$

下一个时间点

$$\begin{aligned}\zeta_{t+1} =&\mu_{t+1} - \mu_{t+1|t}\\ =&\mu_t + \eta_t - (\mu_{t|t-1} + K_t v_t)\\ =&\mu_t - \mu_{t|t-1} + \eta_t - K_t(\zeta_t + e_t)\\ =&\zeta_t + \eta_t - K_t(\zeta_t + e_t)\\ =&(1 - K_t)\zeta_t + \eta_t - K_t e_t\\ =&L_t\zeta_t + \eta_t - K_t e_t,\end{aligned}$$

于是有

$$v_t = \zeta_t + e_t, \tag{27.20}$$

$$\zeta_{t+1} = L_t\zeta_t + \eta_t - K_t e_t,\ t = 1, 2, \dots, T. \tag{27.21}$$

这构成了以 $\{v_t\}$ 为观测，$\{\zeta_t\}$ 为状态的系数时变的线性高斯状态空间模型。

27.7 状态平滑

模型的滤波，是利用已有观测 $\{y_1, \dots, y_t\}$ 估计 μ_t 得 $\mu_t|y_{1:t}$。在获得所有的观测 $\{y_1, \dots, y_n\}$ 后，可以利用所有的观测来估计 μ_t，得到 $\mu_t|Y_n$ 的分布，这个问题称为平滑问题。

注意：

- 所有的联合分布都是正态的，所以记 $\mu_t|y_{1:n} \sim \mathrm{N}(\mu_{t|n}, \Sigma_{t|n}) = \mathrm{N}(\hat{\mu}_t, V_t)$，称 $\hat{\mu}_t$ 为平滑状态 (smoothed state)，称 V_t 为平滑状态方差。
- $\{v_1, v_2, \dots, v_t\}$ 相互独立，可以与 $\{y_1, y_2, \dots, y_t\}$ 互相线性表示。$Ev_t = 0$, $\mathrm{Var}(v_t) = F_t$。

显然

$$\sigma(y_1, \dots, y_{t-1}, y_t, \dots, y_n) = \sigma(y_1, \dots, y_{t-1}, v_t, \dots, v_n).$$

条件分布 $\mu_t|y_{1:n}$，等于条件分布 $\mu_t|y_{1:t-1}, v_t, \dot{;}v_n$。条件分布为高斯分布，只要求条件期望和条件方差。

注意 $v_t, \dots, v_n$ 与 $y_{1:t-1}$ 独立，由定理27.1，条件期望为

$$\begin{aligned}\hat{\mu}_t = &E(\mu_t|y_{1:n}) = E(\mu_t|y_{1:t-1}, v_t, \dots, v_n)\\ =&E(\mu_t|y_{1:t-1}) + E(\mu_t - E\mu_t|v_t, \dots, v_n)\\ =&a_t + \mathrm{Cov}(\mu_t, (v_t, \dots, v_n))\mathrm{Var}^{-1}((v_t, \dots, v_n))(v_t, \dots, v_n)^T\\ =&a_t + (\mathrm{Cov}(m_t, v_t), \mathrm{Cov}(m_t, v_{t-1}), \dots, \mathrm{Cov}(m_t, v_n)))\\ &\mathrm{diag}(F_t^{-1}, F_{t+1}^{-1}, \dots, F_n^{-1})(v_t, \dots, v_n)^T\\ =&a_t + \sum_{j=t}^{n}\frac{\mathrm{Cov}(m_t, v_j)}{F_j}v_j.\end{aligned}$$

只要求 $\mathrm{Cov}(m_t, v_j)$, $j = t, t+1, \dots, n$。

对 $j \geq t$，注意到 $\zeta_j, e_j, \eta_j, v_j$ 都与 $y_{1:t-1}$ 独立，e_j 与 ζ_j 独立，有

$$\begin{aligned}&\mathrm{Cov}(\mu_t, v_j) = E[(\mu_t v_j)]\\ =&E\{E(\mu_t v_j|y_{1:t-1})\}\\ =&E\{E[(\mu_t - \mu_{t|t-1})v_j|y_{1:t-1}]\}\\ =&E\{E[\zeta_t v_j|y_{1:t-1}]\} = E[\zeta_t v_j],\end{aligned}$$

对 $j=t,t+1,\ldots,n$ 计算 $E[\zeta_t v_j]$：

$$\begin{aligned}
E(\zeta_t v_t) =& E[\zeta_t(\zeta_t+e_t)] = E(\zeta_t^2)+0 = P_t,\\
E(\zeta_t v_{t+1}) =& E[\zeta_t(\zeta_{t+1}+e_{t+1})] = E[\zeta_t\zeta_{t+1}]+0\\
=& E[\zeta_t(L_t\zeta_t+\eta_t-K_te_t)] = E[L_t\zeta_t^2]+0+0\\
=& P_tL_t,\\
E(\zeta_t v_{t+2}) =& E[\zeta_t(\zeta_{t+2}+e_{t+2})] = \cdots = P_tL_tL_{t+1},\\
&\vdots\\
E(\zeta_t v_n) =& P_t\prod_{j=t}^{n-1}L_j.
\end{aligned}$$

从而，$\hat{\mu}_t=\mu_{t|n}$ 公式为

$$\begin{aligned}
\hat{\mu}_t =& \mu_{t|n} = a_t + P_t\frac{v_t}{F_t} + P_tL_t\frac{v_{t+1}}{F_{t+1}} + P_tL_tL_{t+1}\frac{v_{t+2}}{F_{t+2}}\\
&+\cdots+P_tL_tL_{t+1}\ldots L_{n-1}\frac{v_n}{F_n}\\
\equiv& a_t+P_tr_{t-1},
\end{aligned}$$

其中

$$r_{t-1} = \frac{v_t}{F_t}+L_t\frac{v_{t+1}}{F_{t+1}}+L_tL_{t+1}\frac{v_{t+2}}{F_{t+2}}+\cdots+L_tL_{t+1}\ldots L_{n-1}\frac{v_n}{F_n}, \tag{27.22}$$

是新息 $(v_t,v_{t+1},\ldots,v_n)$ 的线性函数，满足如下递推公式：

$$\begin{aligned}
r_{t-1} =& \frac{v_t}{F_t}+L_t\left(\frac{v_{t+1}}{F_{t+1}}+L_{t+1}\frac{v_{t+2}}{F_{t+2}}+\cdots+L_{t+1}\ldots L_{n-1}\frac{v_n}{F_n}\right)\\
=& \frac{v_t}{F_t}+L_tr_t,
\end{aligned}$$

令 $r_n=0$，有反向递推算法

$$r_{t-1}=\frac{v_t}{F_t}+L_tr_t,\ t=n,n-1,\ldots,2,1.$$

所以，为了求得 $\hat{\mu}_t=\mu_{t|n}$，需要先进行卡尔曼滤波求出 a_t, P_t, v_t, F_t, K_t, L_t, 然后令 $r_n=0$，用反向递推计算：

$$r_{t-1} = \frac{v_t}{F_t}+L_tr_t, \tag{27.23}$$

$$\hat{\mu}_t = \mu_{t|n} = a_t+P_tr_{t-1},\ t=n,n-1,\ldots,2,1. \tag{27.24}$$

因为 r_t 是 $v_{t+1},\ldots,v_n$ 的线性组合，由 $v_1,\ldots,v_n$ 的独立性可知 $r_t,r_{t+1},\ldots,r_n$ 与 $v_1,v_2,\ldots,v_t$ 相互独立，r_t 与 $y_{1:t}$ 相互独立。

27.7.1 状态平滑方差计算

记 $v_{t:n}=(v_t,v_{t+1},\ldots,v_n)^T$，则 $y_{1:t-1}$ 和 $v_{t:n}$ 独立，由定理27.1，

$$\begin{aligned}
V_t =& \Sigma_{t|n} = \mathrm{Var}(\mu_t|y_{1:n}) = \mathrm{Var}(\mu_t|y_{1:t-1},v_t,\ldots,v_n)\\
=& \mathrm{Var}(\mu_t|y_{1:t-1},v_{t:n})\\
=& \mathrm{Var}(\mu_t|y_{1:t-1}) - [\mathrm{Cov}(\mu_t,v_{t:n})]^T[\mathrm{Var}(v_{t:n})]^{-1}[\mathrm{Cov}(\mu_t,v_{t:n})]\\
=& P_t-\sum_{j=t}^{n}\frac{[\mathrm{Cov}(\mu_t,v_j)]^2}{F_j},
\end{aligned}$$

其中 $\text{Cov}(\mu_t, v_j) = E(\zeta_t v_j)$ 已经在前面给出公式。所以

$$\begin{aligned} V_t =& \Sigma_{t|n} \\ =& P_t - P_t^2 \frac{1}{F_t} - P_t^2 L_t^2 \frac{1}{F_{t+1}} - \cdots - P_t^2 (\prod_{j=t}^{n-1} L_j^2) \frac{1}{F_n} \\ =& P_t - P_t^2 N_{t-1}, \end{aligned}$$

其中

$$N_{t-1} = \frac{1}{F_t} + L_t^2 \frac{1}{F_{t+1}} + L_t^2 L_{t+1}^2 \frac{1}{F_{t+2}} + \cdots + (\prod_{j=t}^{n-1} L_j^2) \frac{1}{F_n} \tag{27.25}$$

$$= \frac{1}{F_t} + L_t^2 N_t. \tag{27.26}$$

取 $N_n = 0$，N_{t-1} 是 $t-1$ 之后的一步预报误差方差倒数的加权和，并且由 r_{t-1} 中各个 v_t 的独立性恰好有

$$\begin{aligned} \text{Var}(r_{t-1}) =& \frac{1}{F_t} + L_t^2 \frac{1}{F_{t+1}} + L_t^2 L_{t+1}^2 \frac{1}{F_{t+2}} + \cdots + (\prod_{j=t}^{n-1} L_j^2) \frac{1}{F_n} \\ =& N_{t-1}. \end{aligned}$$

取 $N_n = 0$，状态平滑方差 $V_t = \Sigma_{t|n}$ 可以反向递推计算如下：

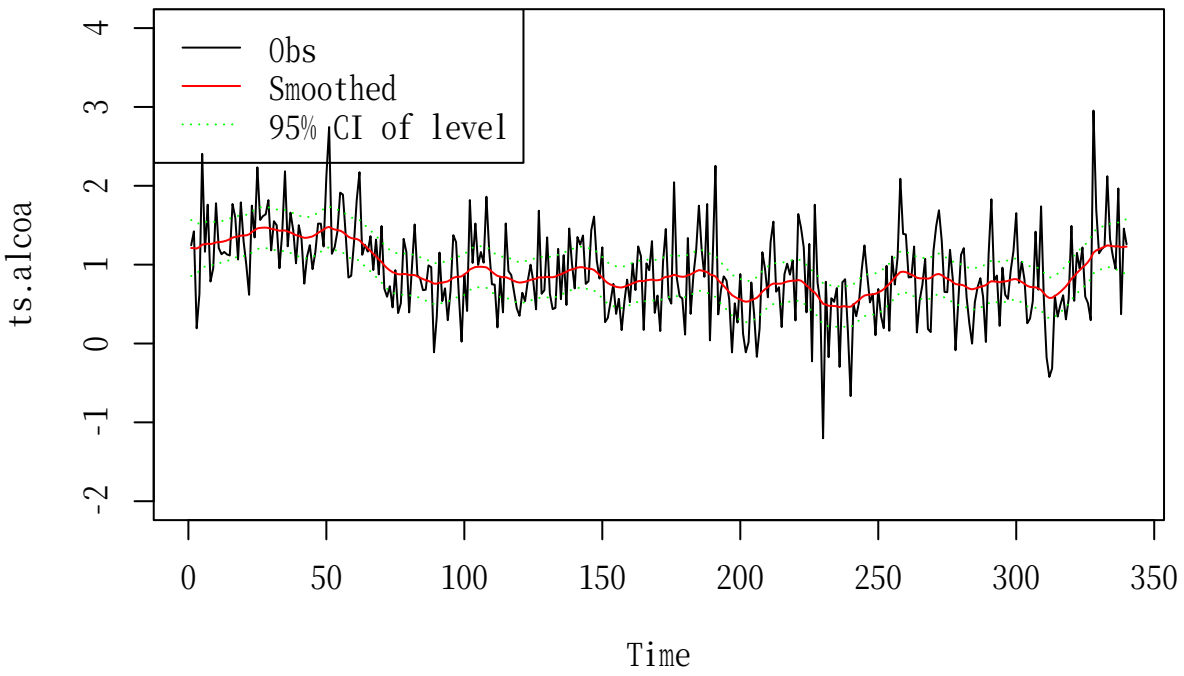
$$N_{t-1} = \frac{1}{F_t} + L_t^2 N_t, \tag{27.27}$$

$$V_t = \Sigma_{t|n} = P_t - P_t^2 N_{t-1}, \ t = n, n-1, \ldots, 2, 1. \tag{27.28}$$

例 27.2. Alcoa 股票日现实波动率对数值数据的平滑。

Alcoa 股票日现实波动率对数值数据以及平滑结果图形：

```
plot(ts.alcoa, ylim=c(-2, 4))
lines(as.vector(time(ts.alcoa)),
  ssr1$smoothed$level, col="red")
lines(as.vector(time(ts.alcoa)),
  ssr1$smoothed$level + 1.96*sqrt(ssr1$smoothed$V[1,1,]),
  col="green", lty=3)
lines(as.vector(time(ts.alcoa)),
  ssr1$smoothed$level - 1.96*sqrt(ssr1$smoothed$V[1,1,]),
  col="green", lty=3)
legend("topleft", lty=c(1,1,3), col=c("black", "red", "green"),
  legend=c("Obs", "Smoothed", "95% CI of level"))
```



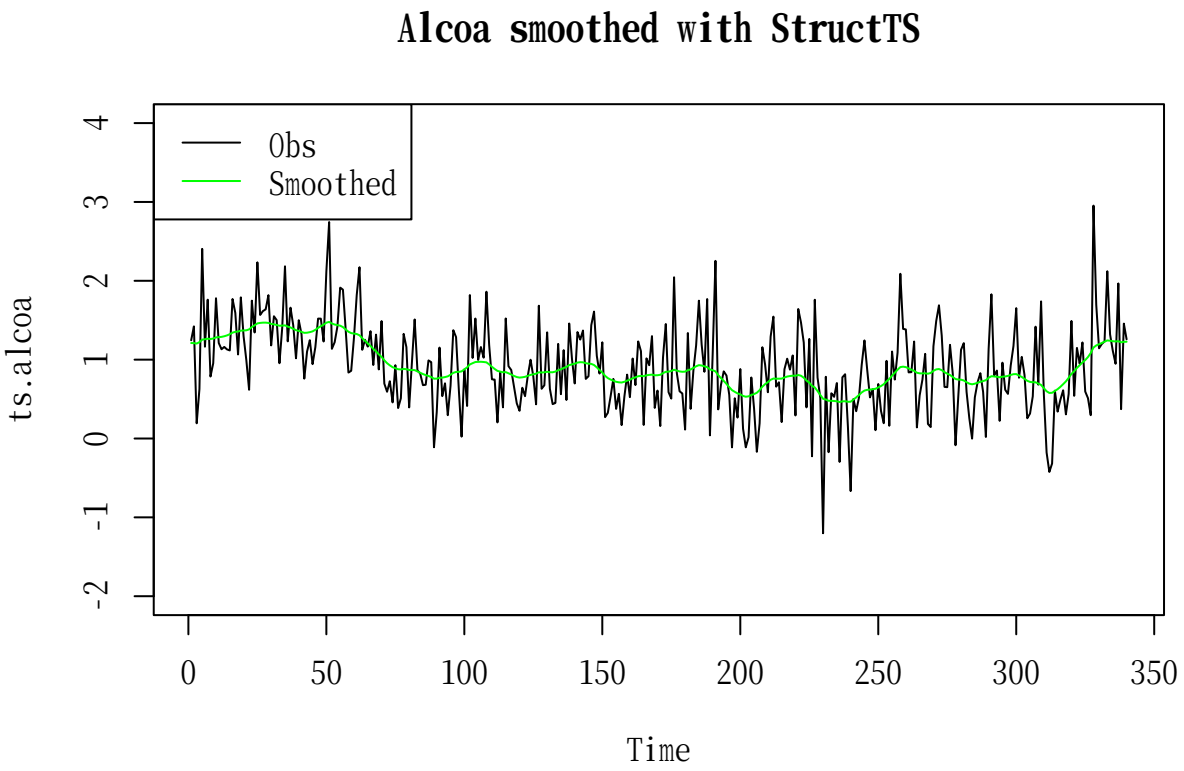
模型结果中 smoothed 成分保存平滑结果，smoothed$level 保存局部水平 μ_t 的平滑结果，smoothed$V 保存平滑方差估计，这里是 $1 \times 1 \times n$ 数组。

实际上，R 的基本的 stats 包提供了 StructTS() 函数，可以直接拟合包括局部水平模型的结构时间序列模型，如：

```
sts.al <- StructTS(ts.alcoa, type="level")
sts.al
```

```
##
## Call:
## StructTS(x = ts.alcoa, type = "level")
##
## Variances:
##    level   epsilon
## 0.005403  0.230652
```

```
plot(ts.alcoa, ylim=c(-2, 4),
  main="Alcoa smoothed with StructTS")
lines(tsSmooth(sts.al), col="green")
legend("topleft", lty=c(1,1),
  col=c("black", "green"),
  legend=c("Obs", "Smoothed"))
```



对 `StructTS()` 的结果，用 `tsSmooth()` 可以提取平滑结果，用 `fitted()` 可以提取滤波结果。可以用 `predict()` 或者 `forecast::forecast()` 进行预报。

27.8 扰动的平滑

有了所有观测值 $\{y_1, y_2, \dots, y_n\}$ 以后，不仅可以利用这些信息估计状态 μ_t，得到平滑状态和平滑方差，还可以估计 e_t 和 η_t 的条件分布，这个问题称为扰动的平滑。

估计 e_t 和 η_t 在 Y_n 下的条件期望和条件方差，可以用来进行模型诊断，查找状态的突变点（对局部水平模型相当于水平的跳跃点或变点），查找观测误差的异常值。

记

$$\hat{e}_t = E(e_t|y_{1:n}), \quad \hat{\eta}_t = E(\eta_t|y_{1:n}),\ t = 1, 2, \dots, n.$$

因为 $e_t = y_t - \mu_t$，在 $y_{1:n}$ 条件下 y_t 已知，所以

$$e_t|y_{1:n} \sim \mathrm{N}(y_t - \mu_{t|n}, \Sigma_{t|n}) = \mathrm{N}(y_t - \hat{\mu}_t, V_t).$$

对 η_t，易见

$$\hat{\eta}_t = E(\mu_{t+1}|y_{1:n}) - E(\mu_t|y_{1:n}) = \mu_{t+1|n} - \mu_{t|n} = \hat{\mu}_{t+1} - \hat{\mu}_t,$$

但 $\eta_t|y_{1:n}$ 的条件方差公式需要用到平滑的协方差。

27.8.1 观测方程扰动的递推平滑公式

从计算效率出发，为了获得 e_t 和 η_t 在 $y_{1:n}$ 下的条件分布，直接从 r_t 和 N_t 计算更简单。关于 $e_t|y_{1:n}$ 的公式为

$$\begin{aligned}E(e_t|y_{1:n}) =&\sigma_e^2\left(F_t^{-1}v_t - K_t r_t\right),\\ \mathrm{Var}(e_t|y_{1:n}) =&\sigma_e^2 - \sigma_e^4\left(\frac{1}{F_t} + K_t^2 N_t\right),\\ &t = n, n-1, \ldots, 2, 1.\end{aligned}$$

证明：由于 $F_t = P_t + \sigma_e^2$, $K_t = \frac{P_t}{F_t}$, $L_t = 1 - K_t = \frac{\sigma_e^2}{F_t}$, 有

$$\begin{aligned}E(e_t|y_{1:n}) =&E(y_t - \mu_t|y_{1:n}) = y_t - \mu_{t|n}\\ =&y_t - a_t - P_t r_{t-1}\\ =&v_t - P_t\left[\frac{v_t}{F_t} + L_t r_t\right]\\ =&\left(1 - \frac{P_t}{F_t}\right)v_t - P_t L_t r_t\\ =&\frac{\sigma_e^2}{F_t}v_t - P_t\frac{\sigma_e^2}{F_t}r_t\\ =&\sigma_e^2\left(\frac{v_t}{F_t} - K_t r_t\right).\end{aligned}$$

记

$$u_t = \frac{v_t}{F_t} - K_t r_t = \sigma_e^{-2}E(e_t|y_{1:n}),$$

称 u_t 为**平滑误差** (smoothing error)。

为了求 $\mathrm{Var}(e_t|y_{1:n})$，注意在联合正态分布下条件分布的方差是非随机的，由公式 $\mathrm{Var(Y)} = E[\mathrm{Var}(Y|X)] + \mathrm{Var}[E(Y|X)]$，以及 v_t 与 r_t 独立，$\mathrm{Var}(r_t) = N_t$，可得

$$\begin{aligned}\mathrm{Var}(e_t|y_{1:n}) =&E[\mathrm{Var}(e_t|y_{1:n})] = \mathrm{Var}(e_t) - \mathrm{Var}[E(e_t|y_{1:n})]\\ =&\sigma_e^2 - \sigma_e^4\left(\frac{1}{F_t} + K_t^2 N_t\right).\end{aligned}$$

27.8.2 状态方程扰动的递推平滑公式

对 $\eta_t|y_{1:n}$ 有

$$\begin{aligned}E(\eta_t|y_{1:n}) =&\sigma_\eta^2 r_t,\\ \mathrm{Var}(\eta_t|y_{1:n}) =&\sigma_\eta^2 - \sigma_\eta^4 N_t,\ t = n, n-1, \ldots, 2, 1.\end{aligned}$$

r_t 与 $\eta_t|y_{1:n}$ 的关系也说明了 r_t 和 N_t 的一个解释，r_t 是平滑的状态扰动项的常数倍，其无条件方差 N_t 是平滑的状态扰动方差的常数倍。

证明：

$$\begin{aligned}E(\eta_t|y_{1:n}) =&E(\mu_{t+1} - \mu_t|y_{1:n}) = \mu_{t+1|n} - \mu_{t|n}\\ =&a_{t+1} + P_{t+1}r_t - a_t - P_t r_{t-1}\\ =&(a_{t+1} - a_t) + \left(P_t\frac{\sigma_e^2}{F_t} + \sigma_\eta^2\right)r_t - P_t\left(\frac{v_t}{F_t} + \frac{\sigma_e^2}{F_t}r_t\right)\\ =&K_t v_t + (K_t\sigma_e^2 + \sigma_\eta^2)r_t - K_t v_t - \sigma_e^2 K_t r_t\\ =&\sigma_\eta^2 r_t.\end{aligned}$$

而

$$\begin{aligned}\mathrm{Var}(\eta_t|y_{1:n}) =&E[\mathrm{Var}(\eta_t|y_{1:n})]\\ =&\mathrm{Var}(\eta_t)-\mathrm{Var}[E(\eta_t|y_{1:n})]\\ =&\sigma_\eta^2-\sigma_\eta^4 N_t.\end{aligned}$$

27.9 缺失值处理和预测

一般的时间序列模型都很难处理出现在时间区间内部的缺失值。状态空间模型的一大优势就是可以比较容易的允许观测有缺失值。在局部水平模型中，设 $\{y_t\}_{t=\ell+1}^{\ell+h}$ 缺失。状态空间模型可以用多种方法解决缺失值问题，这里使用不改变时间步数和模型形式的方法。

对 $t\in\{\ell+1,\dots,\ell+h\}$，有

$$\mu_t=\mu_{t-1}+\eta_{t-1}=\cdots=\mu_{\ell+1}+\sum_{j=\ell+1}^{t-1}\eta_j,$$

这里约定求和下标下限大于下标上限时和为 0。于是

$$\begin{aligned}E(\mu_t|Y_{t-1}) =&E(\mu_t|Y_\ell)=a_{\ell+1},\\ \mathrm{Var}(\mu_t|Y_{t-1}) =&\mathrm{Var}(\mu_t|Y_\ell)=P_{\ell+1}+(t-\ell-1)\sigma_\eta^2,\end{aligned}$$

于是有递推式

$$a_t=\mu_{t|t-1}=\mu_{t-1|t-2}=a_{t-1}, \tag{27.29}$$

$$P_t=\Sigma_{t|t-1}=P_{t-1}+\sigma_\eta^2,\ t=\ell+2,\dots,\ell+h. \tag{27.30}$$

所以，局部水平模型有缺失值时仍可使用(27.17)进行滤波计算，但是对缺失的 y_t，应该相应地取 $v_t=0$, $K_t=0$。这也符合常识，因为观测缺失时就没有新息 v_t 的信息，也没有卡尔曼增益 K_t。

实际上，在已知 $y_1,y_2,\dots,y_n$ 后对 $y_{n+1},\dots,y_{n+h}$ 进行预测时，最优线性无偏估计得到的预测公式，与设 $y_{n+1},\dots,y_{n+h}$ 为缺失值，令 $v_{n+1}=\cdots=v_{n+h}=K_{n+1}=\cdots=K_{n+h}=0$ 后进行滤波计算的计算公式完全相同。所以预测问题可以看成有缺失值时的滤波问题的特例。这时

$$\begin{aligned}E(y_{n+j}|y_{1:n}) =&E(\mu_{n+j}|y_{1:n})=a_{n+1},\\ \mathrm{Var}(y_{n+j}|y_{1:n}) =&\mathrm{Var}(\mu_{n+j}|y_{1:n})+\sigma_e^2=P_n+(j-1)\sigma_\eta^2+\sigma_e^2.\end{aligned}$$

27.10 初值分布参数选取

卡尔曼滤波计算需要假定已知 $\mu_1\sim\mathrm{N}(a_1,P_1)$。实际中 a_1 和 P_1 是未知的。利用滤波公式得

$$\begin{aligned}v_1 =&y_1-a_1,\quad F_1=P_1+\sigma_e^2,\\ a_2 =&a_1+\frac{P_1}{F_1}v_1=a_1+\frac{P_1}{F_1}(y_1-a_1)\\ \to&y_1\quad(P_1\to\infty),\\ P_2 =&P_1\left(1-\frac{P_1}{P_1+\sigma_e^2}\right)+\sigma_\eta^2\\ =&\frac{P_1}{P_1+\sigma_e^2}\sigma_e^2+\sigma_\eta^2\\ \to&\sigma_e^2+\sigma_\eta^2\quad(P_1\to\infty),\end{aligned}$$

因此 $P_1 \to \infty$ 时相当于认为 y_1 是非随机的确定值，而 $\mu_1 \sim \mathrm{N}(y_1, \sigma_e^2)$。这种初始化方法称为扩散 (diffuse) 初始化或者扩散先验。扩散先验相当于对初始状态分布没有任何知识。

对于取扩散先验时的平滑问题，$t = n, n-1, \ldots, 2$ 的反向递推仍完全原样进行。对 $t = 1$，利用 $L_1 = 1 - K_1 = F_1^{-1}\sigma_e^2$, $F_1 = P_1 + \sigma_e^2$，得

$$\begin{aligned}
\hat{\mu}_1 =& \mu_{1|n} = a_1 + P_1 r_0 \\
=& a_1 + P_1 \left[\frac{1}{P_1 + \sigma_e^2} v_1 + \left(1 - \frac{P_1}{P_1 + \sigma_e^2}\right) r_1 \right] \\
=& a_1 + \frac{P_1}{P_1 + \sigma_e^2}(v_1 + \sigma_e^2 r_1) \\
\to& a_1 + v_1 + \sigma_e^2 r_1 = y_1 + \sigma_e^2 r_1 \quad (P_1 \to \infty), \\
V_1 =& \Sigma_{1|n} = P_1 - P_1^2 \left[\frac{1}{P_1 + \sigma_e^2} + \left(1 - \frac{P_1}{P_1 + \sigma_e^2}\right)^2 N_1 \right] \\
=& P_1 \left(1 - \frac{P_1}{P_1 + \sigma_e^2}\right) - \left(1 - \frac{P_1}{P_1 + \sigma_e^2}\right)^2 P_1 N_1 \\
=& \left(\frac{P_1}{P_1 + \sigma_e^2}\right) \sigma_e^2 - \left(\frac{P_1}{P_1 + \sigma_e^2}\right)^2 \sigma_e^4 N_1 \\
\to& \sigma_e^2 - \sigma_e^4 N_1 \quad (P_1 \to \infty).
\end{aligned}$$

另一种想法是将 μ_1 也作为一个未知参数参与到最大似然估计中。

27.11 模型参数估计

滤波和平滑算法都是假定模型参数 σ_e^2 和 σ_η^2 已知的。为了估计参数可以使用最大似然估计法，计算似然函数时可以利用滤波算法进行计算。

$(y_1, y_2, \ldots, y_n)$ 的似然函数为

$$\begin{aligned}
& p_y(y_1, y_2, \ldots, y_n | \sigma_e^2, \sigma_\eta^2) \\
=& p_y(y_1 | \sigma_e^2, \sigma_\eta^2) \prod_{t=2}^{n} p_y(y_t | Y_{t-1}, \sigma_e^2, \sigma_\eta^2) \\
=& p_v(v_1 | \sigma_e^2, \sigma_\eta^2) \prod_{t=2}^{T} p_v(v_t | \sigma_e^2, \sigma_\eta^2),
\end{aligned}$$

其中 $y_1 \sim \mathrm{N}(a_1, P_1)$，$v_t \sim \mathrm{N}(0, F_t)$，设 a_1, P_1 已知，可得对数似然函数为

$$\ln L(\sigma_e^2, \sigma_\eta^2) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\sum_{t=1}^{n}\left(\ln F_t + \frac{v_t^2}{F_t}\right).$$

给定初始值 a_1, P_1 以及一组参数值 $\sigma_e^2, \sigma_\eta^2$ 就可以用卡尔曼滤波算法计算出 v_t, F_t 从而得到对数似然函数值。

具体计算可以用 R 的 statespacer 扩展包，KFAS, bssm 扩展包等，还有许多其它软件也可以进行状态空间模型建模。

Chapter 28

状态空间模型

上一章的局部水平模型是线性高斯状态空间模型的一个简单特例。本章给出状态空间模型，举例说明这种模型能够表示的其它模型，如 ARIMA 模型，结构时间序列模型，时变回归模型，有自相关误差的回归模型，随机波动率模型等，并给出滤波、平滑、预报公式和参数估计方法。

参考：

- (Durbin & Koopman, 2012)
- (Tsay, 2010)
- (Beijers, 2020)

28.1 状态空间模型的公式

28.1.1 线性高斯状态空间模型

状态空间模型有许多不同的表达形式，按照 (Durbin & Koopman, 2012) 的公式，线性高斯模型为：

$$y_t = Z_t\alpha_t + \varepsilon_t,\ \varepsilon_t \sim \mathrm{N}(0, H_t), \tag{28.1}$$

$$\alpha_{t+1} = T_t\alpha_t + R_t\eta_t,\ \eta_t \sim \mathrm{N}(0, Q_t), \tag{28.2}$$

其中 y_t 是 t 时刻的观测值，为 $p\times 1$ 向量；α_t 时 t 时刻系统的状态，是不可观测的 $m\times 1$ 随机向量，第一个方程称为观测方程，第二个方程称为状态方程。$\{\varepsilon_t\}$ 和 $\{\eta_t\}$ 相互独立，都是独立同分布向量白噪声列，ε_t 为 $p\times 1$ 随机向量，η_t 为 $r\times 1$ 随机向量，$r\le m$。设各矩阵 Z_t, T_t, R_t, H_t, Q_t 已知，Z_t 和 T_{t-1} 允许依赖于 $y_1, \dots, y_{t-1}$，初始状态 α_1 服从 $N(a_1, P_1)$，设 a_1, P_1 已知，α_1 与 $\{\varepsilon_t\}$ 和 $\{\eta_t\}$ 独立。当参数未知时，设 ψ 为未知参数，矩阵 Z_t, T_t, R_t, H_t, Q_t 可以依赖于未知参数 ψ。

模型中的 R_t 常常是单位阵，$r=m$，有些教材的模型就没有 R_t 这一项。包含 R_t 的好处是，R_t 常常是单位阵 I_m 的某些列组成的一个 $m\times r$ 矩阵，称为选择矩阵，这允许某些状态分量对应的方程误差为 0，同时 η_t 的方差阵 Q_t 还可以是满秩的 $r\times r$ 正定阵，如果没有 R_t 矩阵 Q_t 就可能不满秩。如果 R_t 是一般的 $m\times r$ 矩阵，关于状态空间模型的大部分结论仍成立。

28.1.2 推广的状态空间模型

可以将线性高斯的状态空间模型，推广到状态方程仍为线性高斯形式，而观测方程的分布为非高斯分布，或者观测方程中观测变量与状态变量的关系非线性，更进一步可以推广到状态方程的关系也非线性，分布为非高斯分布。

较一般的非线性、非高斯状态空间模型形式为：

$$\begin{aligned} y_t &\sim f_t(\alpha_t;\beta), \\ \alpha_{t+1} &\sim g_t(\alpha_t;\theta), \end{aligned}$$

其中 $f_t(\cdot)$, $g(\cdot)$ 是密度函数或概率质量函数，β 和 θ 是超参数。这样的模型一般需要用 MCMC、序贯重要抽样等随机模拟方法进行滤波、平滑和估计。

28.1.3 MARSS 包的模型

(Holmes et al., 2021) 中的状态空间模型与 (Durbin & Koopman, 2012) 的模型形式相近，但更加灵活。有配套的扩展包 MARSS，MARSS 是多元自回归状态空间模型的缩写，实际上就是线性高斯状态空间模型。

基本的模型公式为：

$$\begin{aligned} x_t &= Bx_{t-1} + u + w_t, \quad w_t \sim \mathrm{N}(0, Q), \\ y_t &= Zx_t + a + v_t, \qquad v_t \sim \mathrm{N}(0, R), \\ x_0 &\sim \mathrm{N}(\pi, \Lambda). \end{aligned}$$

这与 (Durbin & Koopman, 2012) 的公式基本相同，只是记号不同，并且多出了 a 和 u。矩阵 B, Z, u, a 都允许时变。更复杂的模型还可以在两个方程中增加关于外生变量影响的部分。MARSS 扩展包的参数化方法和估计方法与其它状态空间模型扩展包有比较大的区别。

包含外生变量的回归部分并且各个矩阵允许时变的模型可以写成

$$\begin{aligned} x_t &= B_t x_{t-1} + u_t + C_t c_t + w_t, \quad w_t \sim \mathrm{N}(0, Q_t), \\ y_t &= Z_t x_t + a_t + D_t d_t + v_t, \qquad v_t \sim \mathrm{N}(0, R_t), \\ x_0 &\sim \mathrm{N}(\pi, \Lambda). \end{aligned}$$

其中 c_t 是系统方程中的 p 维外生变量数据，可以输入为一个 $p \times T$ 矩阵；C_t 是相应的回归载荷矩阵，可以包含未知量，如果是非时变的，只要输入为 $m \times p$ 矩阵；如果是时变的，则需要输入为 $m \times p \times T$ 的三维数组，用最后一个下标表示时间 t。d_t 则是观测方程中的 q 维外生变量数据，D_t 是相应的载荷矩阵。

这样的 MARSS 模型不仅可以表示回归模型，也可以表示有内生状态变量 x_t 的带有外生变量（回归自变量）的时间序列模型。

28.1.4 动态线性模型

(West & Harrison, 1997) 基于贝叶斯理论框架提出了高斯的动态线性模型，其结构和结果与线性高斯时间序列模型卡尔曼滤波基本相同。

用 θ_t 表示 t 时刻的状态，y_t 是 t 时刻的因变量，D_{t-1} 表示 t 时刻之前的已知信息，则 $p(\theta_t|D_{t-1})$ 是 t 时刻的先验分布，相当于状态的一步预报分布，$p(\theta_t|D_t)$ 是 t 的后验分布，相当于滤波分布，有

$$p(y_t, \theta_t|D_{t-1}) = p(y_t|\theta_t, D_{t-1})p(\theta_t|D_{t-1}).$$

可以从上述联合分布得到一步预报分布 $p(y_t|D_{t-1})$。

多元形式的高斯动态线性模型为：

$$Y_t|\theta_t \sim \mathrm{N}[F_t^T\theta_t, V_t], \tag{28.3}$$
$$\theta_t|\theta_{t-1} \sim \mathrm{N}[G_t\theta_{t-1}, W_t]. \tag{28.4}$$

或写成：

$$\begin{aligned} Y_t &= F_t^T\theta_t + \nu_t, \quad & \nu_t &\sim \mathrm{N}(0, V_t), \\ \theta_t &= G_t\theta_{t-1} + \omega_t, & \omega_t &\sim \mathrm{N}(0, W_t), \\ & & \theta_0 &\sim \mathrm{N}(m_0, C_0). \end{aligned}$$

与 (Durbin & Koopman, 2012) 的模型记号对比，上面的 θ_t 相当于 α_t，F_t^T 相当于 Z_t，V_t 相当于 H_t，G_t 相当于 T_{t-1}，W_t 相当于 $R_{t-1} = I$ 时的 Q_{t-1}。((28.3))-((28.4)) 中的分布也可以依赖于 D_{t-1}。模型中的四个矩阵 (F_t, G_t, V_t, W_t) 为非随机的超参数。

对于常见的线性时间序列模型，(F_t, G_t, V_t, W_t) 一般不依赖于 t。

例 28.1 (局部水平模型)**.** 局部水平模型用 DLM 表示，是单个参数的一元时间序列模型。公式为：

$$\begin{aligned} Y_t &= F_t\mu_t + \nu_t, \quad & \nu_t &\sim \mathrm{N}(0, V_t), \\ \mu_t &= \lambda\mu_{t-1} + \omega_t, & \omega_t &\sim \mathrm{N}(0, W_t), \\ & & (\mu_0|D_0) &\sim \mathrm{N}(m_0, C_0). \end{aligned}$$

其中 F_t, λ 已知，V_t 和 W_t 可以是未知的。当 $F_t = 1, \lambda = 1$ 则为一般的局部水平模型，在 (West & Harrison, 1997) 中称为一阶多项式模型：

$$\begin{aligned} Y_t &= \mu_t + \nu_t, \quad & \nu_t &\sim \mathrm{N}(0, V_t), \\ \mu_t &= \mu_{t-1} + \omega_t, & \omega_t &\sim \mathrm{N}(0, W_t), \\ & & (\mu_0|D_0) &\sim \mathrm{N}(m_0, C_0). \end{aligned}$$

其中 W_t 应该接近于 0，μ_t 在短期内变化很小。

例 28.2 (时变系数回归模型)**.** 对时间序列使用线性回归，当不同时期的关系有缓慢变化时不能自动调整关系，可以推广为系数缓慢变化的模型。模型公式：

$$\begin{aligned} y_t &= \alpha_t + \beta_t X_t + \nu_t, \quad & \nu_t &\sim \mathrm{N}(0, V), \\ \alpha_t &= \alpha_{t-1} + \xi_t, & \xi_t &\sim \mathrm{N}(0, W^{(\alpha)}), \\ \beta_t &= \beta_{t-1} + \eta_t, & \eta_t &\sim \mathrm{N}(0, W^{(\beta)}). \end{aligned}$$

其中 $W^{(\alpha)}$ 和 $W^{(\beta)}$ 都很接近于 0。

28.2 有关 R 扩展包

R 软件中有许多扩展包支持进行状态空间模型建模。

- statespacer：支持线性高斯状态时间序列建模。
- KFAS：支持线性高斯状态时间序列建模，也支持指数分布族的非高斯情况。
- dlm：使用 (West & Harrison, 1997) 的模型，动态线性模型。支持线性高斯时间序列模型，可使用最大似然估计，支持时变模型。

- dynr：支持带有机制切换的离散时间或者连续时间的模型建模。
- dse: 线性高斯的 ARMA、VAR 和状态空间模型，所用的方法不太符合 R 的使用习惯。
- bssm: 非线性、非高斯状态空间模型的贝叶斯推断。
- MARSS: 多元自回归状态空间模型。参数可以时变，观测值可以包含缺失值。
- MSwM: 马尔可夫机制切换的一元自回归模型，支持线性和广义线性模型。

28.2.1 statespacer

statespacer 包支持线性高斯状态空间模型建模，文档比较详细，使用 (Durbin & Koopman, 2012) 的模型记号和初始化思想。对常用的结构时间序列模型、ARMA 等提供了更简单的设定功能。对各矩阵时变的情况支持不足。

`statespacer()` 是主要的建模函数。对于常见模型，可以使用选项直接指定模型。需要输入超参数的初始值，这需要熟悉模型如何用状态空间表示。结果都存成了深入嵌套的列表，所以要熟悉 (Durbin & Koopman, 2012) 的模型记号:

$$\begin{aligned} y_t =& Z_t\alpha_t + \varepsilon_t, & \varepsilon_t &\sim \mathrm{N}(0, H_t), \\ \alpha_{t+1} =& T_t\alpha_t + R_t\eta_t, & \eta_t &\sim \mathrm{N}(0, Q_t), \\ & & \alpha_1 &\sim \mathrm{N}(a_1, P_1). \end{aligned}$$

设 sr 为保存了 `statespacer()` 结果的列表，访问其中的成分如下：

- `system_matrices`: 系统矩阵
  - `H`, `Z`, `T`, `R`, `Q` 等。
- `predicted`: 一步预测分布
  - `yfit`: y_t 的一步预测
  - `v`: y_t 的一步预测的误差
  - `Fmat`: y_t 的一步预测的误差方差阵
  - `a`: α_t 的一步预测
  - `P`: α_t 的一步预测的误差方差阵
  - ……
- `filtered`: α_t 的滤波分布均值 `a` 和方差阵 `P`。
- `smoothed`: 平滑结果
  - `a`: α_t 的平滑分布的均值
  - `V`: α_t 的平滑分布的方差阵
  - ……

28.2.2 KFAS

KFAS 支持线性高斯状态时间序列建模，也支持指数分布族的非高斯情况。对于线性高斯状态时间序列仍使用 (Durbin & Koopman, 2012) 的模型记号。

用 `SSModel()` 指定要估计的模型并输入数据，但不进行估计，支持常见模型的简单设定。用 `fitSSM()` 进行超参数估计。用 `KFS()` 进行一步预报、滤波与平滑计算。

28.2.3 dlm

dlm(动态线性模型) 使用 (West & Harrison, 1997) 的模型，基于贝叶斯理论框架，支持线性高斯时间序列模型，也可使用最大似然估计，支持时变模型。

用 `dlm()` 函数指定超参数已知的状态空间模型，常用模型可以用专有函数指定。为了进行超参数估计，需要用户自己编写一个输入超参数值，输出状态空间模型对象的函数。用 `dlmMLE()` 进行超参数的最大似然估计。估计后，将结果中的超参数估计映射到一个状态空间模型对象，用 `dlmFilter()` 提取滤波结果，用 `dlmSmooth()` 提取平滑结果。

28.2.4 MARSS

MARSS 扩展包有详细的数百页的用户手册。

MARSS 包的基本函数是 `MARSS()`，用法为

```
fit <- MARSS(y, model=list(...))
```

其中 `y` 是要建模的时间序列，如果是一维的也可以是普通 R 向量，更一般地可以输入为一个 $n \times T$ 矩阵，其中 n 是观测值 y_t 的维数，T 是时间序列观测时间点个数，每一列对应于一个时间点。观测值允许有缺失值。

`model` 用一个列表输入 `B`, `U`, `Q` 等成分，变量名就用(28.1.3)中的记号，但 π 用 `x0` 表示。如果还指定了 `V0`，则表示 x_0 先验分布指定为均值 `x0`, 方差阵 `V0`。

`MARSS()` 返回一个 `marseMLE` 类型的对象，可以用各种信息提取函数提取或者进一步分析：

- `print(MLEobj)` 显示主要结果。`summary(MLEobj)` 显示结果更少一些。
- `coef(MLEobj)` 提取参数估计。用 `tidy::broom(MLEobj)` 提取成数据框格式。
- `residuals(MLEobj)` 提取观测或者状态的预测、滤波或平滑的残差，返回数据框形式。
- `tsSmooth(MLEobj)` 提取预测、滤波或平滑结果，用 `type` 参数选择，默认为平滑，返回数据框形式。加选项 `interval = "confidence"` 可以同时输出平滑的预测区间。`fitted(MLEobj)` 默认为一步预测，并且在预测、滤波、平滑时不会对噪声部分进行估计，所以要拟合观测因变量值应该使用此函数。支持对缺失数据的估计。在预测时可以用 `n.ahead` 指定步数作多步预测。
- `logLik(MLEobj)` 返回对数似然函数值。
- `AIC(MLEobj)` 返回 AIC 值,`AICc(MLEobj)` 是对小样本情形进行修正的一个变种。`MLEobj <- MARSSaic(MLEobj)` 则添加更多的 AIC 类判别准则值。
- `MARSSkf(MLEobj)` 进行滤波、平滑，结果中：`xtt1` 是状态的一步预报期望；`xtt` 是状态的滤波期望；`xtT` 是状态的平滑期望。`Vtt1`, `Vtt`, `VtT` 分别是状态的一步预报、滤波、平滑的方差阵估计，等等，见 MARSS 用户手册 §3.3 和 §5.10。
- `MARSSparamCIs()` 计算参数置信区间,默认使用海色阵计算,用 `method = "parametric"` 指定使用参数 bootstrap 方法，用 `method = "innovation"` 指定使用新息重抽样方法。对于方差阵参数，不应使用海色阵方法。bootstrap 方法计算时间很长，应仅在算例充足时使用。
- `MARSSboot(MLEobj)` 用 bootstrap 方法进行置信区间、偏差估计等，可以用参数方法，或者新息重抽样方法，当观测值包含缺失值时仅支持新息重抽样方法。
- 还有一些进一步计算的函数，见 MARSS 文档中用户手册的 §2.4。

例 28.3. 设 x_t 是一元 AR(1) 序列，y_t 是 x_t 加上观测噪声。模型公式为

$$\begin{aligned} x_t &= bx_{t-1} + w_t, \quad & w_t &\sim \mathrm{N}(0, q), \\ y_t &= x_t + v_t, & v_t &\sim \mathrm{N}(0, r), \\ x_0 &= \mu. \end{aligned}$$

其中 r, q, b 是未知参数。生成模拟数据并用 `MARSS()` 滤波。

解答：注意公式中所有项都是矩阵，列向量也看成矩阵，用 `matrix("q")` 这样的办法表示有未知参数 `q` 的矩阵。

```
library(MARSS)
```

```
## Warning: package 'MARSS' was built under R version 4.2.3
```

```
set.seed(101)

f_marss01 <- function(
    n = 100, q = 0.1, r = 0.1, b = 0.7){
  x <- arima.sim(
    model=list(ar=b), n=n, sd = sqrt(q))
  y <- x + rnorm(n, 0, sqrt(r))
  y
}
mod.list <- list(
  B = matrix("b"),
  U = matrix(0),
  Q = matrix("q"),
  Z = matrix(1),
  A = matrix(0),
  R = matrix("r"),
  x0 = matrix("mu"),
  tinitx = 0 )

fit <- MARSS(f_marss01(), model = mod.list)
```

```
## Success! abstol and log-log tests passed at 51 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 51 iterations.
## Log-likelihood: -74.36424
## AIC: 156.7285   AICc: 157.1495
##
```

```
##       Estimate
## R.r     0.0888
## B.b     0.6272
## Q.q     0.1478
## x0.mu  -1.1265
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

估计结果比较接近于真实值。

用 MARSS::glance() 以数据框形式查看拟合概况：

```
MARSS::glance(fit)
```

```
##    coef.det     sigma df    logLik      AIC     AICc convergence errors
## 1 0.2379051 0.2628362  4 -74.36424 156.7285 157.1495           0      0
```

用 MARSS::tidy() 提取估计结果：

```
MARSS::tidy(fit)
```

```
##    term    estimate  std.error    conf.low   conf.up
## 1   R.r  0.08882909 0.06462791 -0.03783928 0.2154975
## 2   B.b  0.62718937 0.15488220  0.32362584 0.9307529
## 3   Q.q  0.14775143 0.08279321 -0.01452027 0.3100231
## 4 x0.mu -1.12646009 0.82153615 -2.73664134 0.4837212
```

用 summary() 查看基本估计结果：

```
summary(fit)
```

```
##
## m: 1 state process(es) named X.Y1
## n: 1 observation time series named Y1
##
##    term    estimate
## 1   R.r  0.08882909
## 2   B.b  0.62718937
## 3   Q.q  0.14775143
## 4 x0.mu -1.12646009
```

使用 MARSS 的关键是将问题写成(28.1.3)的格式，然后在 MARSS() 中用 model 指定一个表示模型形式的列表。上面的例子中所有矩阵都是 1×1 的，对于未知参数可以用一个变量名字符串表示。如果要指定矩阵，而且矩阵元素有些位置，有些已知，则需要使用列表矩阵（元素为长度 1 的列表的矩阵，元素可以取数值，也可以取字符串）。比如，

设矩阵 B 为

$$B = \begin{pmatrix} 1 & \lambda_1 \\ \lambda_2 & 0 \end{pmatrix},$$

则可以定义为：

```
B <- matrix(list(), 2, 2)
B[1,1] <- 1
B[2,2] <- 0
B[1,2] <- "lam1"
B[2,1] <- "lam2"
B
```

```
##      [,1]   [,2]
## [1,] 1      "lam1"
## [2,] "lam2" 0
```

也可以简写为：

```
B <- matrix(list(
  1, "lam1",
  "lam2", 0),
  nrow=2, byrow=TRUE); B
```

```
##      [,1]   [,2]
## [1,] 1      "lam1"
## [2,] "lam2" 0
```

u 和 a 的设定：u 是 $m \times 1$ 列向量 (m 是 x_t 的维数)，在模型设定中用 `U` 表示。a 是 $n \times 1$ 列向量 (n 是 y_t 的维数)，在模型设定中用 `A` 表示。

- 如果都是 0 值，可以写成`"zero"`。
- 若包含未知参数和已知值，可以用列表元素矩阵表示。
- 如果都是未知参数，只要用字符型矩阵表示。
- 如果设置为`"unequal"` 或者`"unconstrained"`，则表示所有元素都是未知参数。
- 如果设置为`"equal"`，则表示所有元素都相等且未知。
- 对于 a，还可以设置 `A = "scaling"`，这适用于同一个状态向量分量在多个观测值分量上有输出结果时进行调整。

三个方差阵 Q, R, Λ 的设置：Λ 表示为 `V0`。Λ 的设置比较微妙，设置错误可能给出表面上没有问题但实际错误的估计结果。都可以用列表矩阵表示。也有一些简写方式：

- `"diagonal and unequal"`: 对角阵，元素未知且互异；
- `"diagonal and equal"`: 对角阵，元素未知但相等；
- `"unconstrained"`: 元素未知，仅要求对称；
- `"equalvarcov"`: 方差相等，协方差相等；
- `"identity"`: 已知，为单位阵；
- `"zero"`: 已知，所有元素都等于 0。

矩阵 B 是状态转移矩阵，可以输入为普通数值矩阵、字符型矩阵、列表矩阵，也可以使用 Q 那样的简写。

因为 MARSS 使用 EM 算法或者 BFGS 算法进行最大似然估计,这些算法都可能陷入到局部最优中。所以,用 MARSS 进行实际建模时，应该选择不同的初值，对每组初值仅用 EM 执行十几次迭代 (用 `control = list(maxit=20)`)，选择对数似然值最大的初值。可以随机地产生一批初值。见 MARSS 用户手册 §6.2。

28.3 一元结构时间序列模型

下面给出多种时间序列模型的状态空间模型表示和 R 程序应用。

前一章的局部水平模型是一元结构时间序列模型的特例。在局部水平模型中增加一个斜率 ν_t 项，变成

$$\begin{aligned} y_t =& \mu_t + e_t, \\ \mu_{t+1} =& \mu_t + \nu_t + \xi_t, \\ \nu_{t+1} =& \nu_t + \zeta_t, \end{aligned}$$

写成状态空间模型，为

$$\begin{aligned} y_t =& (1\ 0) \begin{pmatrix} \mu_t \\ \nu_t \end{pmatrix} + e_t, \\ \begin{pmatrix} \mu_{t+1} \\ \nu_{t+1} \end{pmatrix} =& \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_t \\ \nu_t \end{pmatrix} + \begin{pmatrix} \xi_t \\ \zeta_t \end{pmatrix}. \end{aligned}$$

再增加一个季节项 γ_t，季节项有多种模型可选，比如 $\gamma_t = -\sum_{j=1}^{s-1} \gamma_{t-j} + \omega_t$，或者同一季度的随机游动，或者三角多项式形式的表达式。这里用

$$\begin{aligned} y_t =& \mu_t + \gamma_t + e_t, \\ \mu_{t+1} =& \mu_t + \nu_t + \xi_t, \\ \nu_{t+1} =& \nu_t + \zeta_t, \\ \gamma_{t+1} =& -\sum_{j=1}^{s-1} \gamma_{t+1-j} + \omega_t \end{aligned}$$

写成 SSM 形式，

$$\begin{aligned} \alpha_t =& (\mu_t, \nu_t, \gamma_t, \dots, \gamma_{t-s+2})^T, \\ y_t =& (1, 0, 1, 0, \dots, 0)\alpha_t + e_t, \\ \alpha_{t+1} =& T_t \alpha_t + R_t \eta_t. \end{aligned}$$

例如，当 $s=4$（季度数据）时，

$$\begin{aligned}\alpha_t =&(\mu_t,\nu_t,\gamma_t,\gamma_{t-1},\gamma_{t-2})^T,\\ y_t =&(1,0,1,0,0)\alpha_t+e_t,\\ \begin{pmatrix}\mu_{t+1}\\ \nu_{t+1}\\ \gamma_{t+1}\\ \gamma_t\\ \gamma_{t-1}\end{pmatrix} =&\begin{pmatrix}1&1&0&0&0\\ 0&1&0&0&0\\ 0&0&-1&-1&-1\\ 0&0&1&0&0\\ 0&0&0&1&0\end{pmatrix}\begin{pmatrix}\mu_t\\ \nu_t\\ \gamma_t\\ \gamma_{t-1}\\ \gamma_{t-2}\end{pmatrix}\\ &+\begin{pmatrix}1&0&0\\ 0&1&0\\ 0&0&1\\ 0&0&0\\ 0&0&0\end{pmatrix}\begin{pmatrix}\xi_t\\ \zeta_t\\ \omega_t,\end{pmatrix},\quad \begin{pmatrix}\xi_t\\ \zeta_t\\ \omega_t,\end{pmatrix}\sim \mathrm{N}\left(0,\mathrm{diag}(\sigma_\xi^2,\sigma_\zeta^2,\sigma_\omega^2)\right).\end{aligned}$$

模型中可以加入商业周期项 c_t，可以增加若干个回归自变量，可以增加干预变量如：$w_t=I_{\{t\geq\tau\}}$。

28.3.1 Alcoa 现实波动率的局部水平模型

例 28.4. 考虑 Alcoa 股票日现实波动率数据的局部水平模型，时间期间为 2003-01-02 到 2004-05-07，共 340 个观测。

读入数据：

```
da <- readr::read_table("aa-3rv.txt",
  col_names=FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double()
## )
```

```
ts.alcoa <- ts(log(da[[2]]))
```

使用局部水平模型，状态空间模型为：

$$\begin{aligned}y_t =&\mu_t+e_t,\\ \mu_{t+1} =&\mu_t+\eta_t.\end{aligned}$$

28.3.1.1 用 statespacer

使用 statespacer 包进行估计，这个包直接支持对结构时间序列模型进行简化的模型设定：

```
library(statespacer)
ssr1 <- statespacer(
```

```
  y = cbind(as.vector(ts.alcoa)),
  local_level_ind = TRUE,
  initial = rep(0.5*log(var(ts.alcoa)), 2),
  verbose = TRUE)
```

```
## Starting the optimisation procedure at: 2023-05-30 17:04:42

## initial  value 1.022439
## iter  10 value 0.765458
## iter  20 value 0.764395
## final  value 0.764395
## converged

## Finished the optimisation procedure at: 2023-05-30 17:04:43

## Time difference of 0.0232410430908203 secs
```

```
c("Var_obs"=ssr1$system_matrices$H$H,
  "Var_level"=ssr1$system_matrices$Q$level)
```

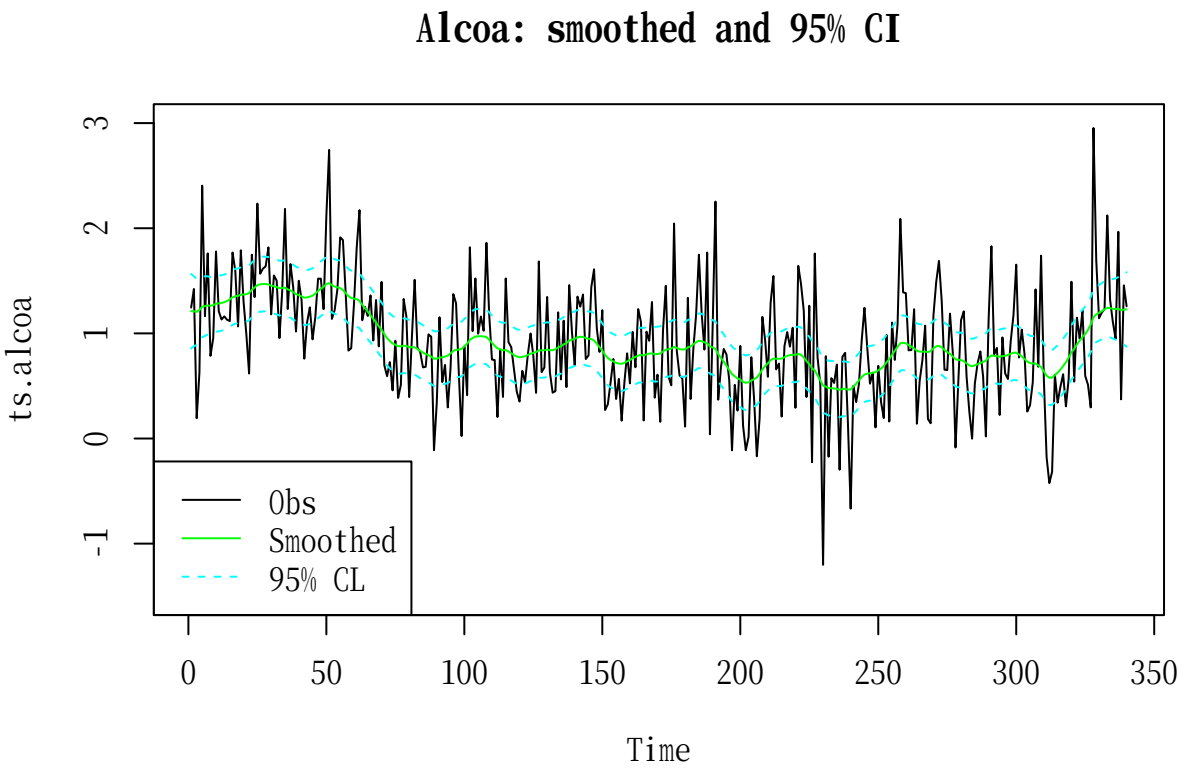
```
##     Var_obs   Var_level
## 0.230623632 0.005404681
```

超参数估计为

$$\sigma_e^2 = 0.2306,\ \sigma_\eta^2 = 0.005405.$$

做原始序列与平滑得到的趋势（水平值）的时间序列图，包括 95% 预测区间：

```
smsd <- sqrt(ssr1$smoothed$V[1,1,])
plot(ts.alcoa, ylim=c(-1.5, 3),
  main="Alcoa: smoothed and 95% CI")
lines(as.vector(time(ts.alcoa)),
  ssr1$smoothed$level, col="green")
lines(as.vector(time(ts.alcoa)),
  ssr1$smoothed$level - 1.96*smsd,
  lty=2, col="cyan")
lines(as.vector(time(ts.alcoa)),
  ssr1$smoothed$level + 1.96*smsd,
  lty=2, col="cyan")
legend("bottomleft", lty=c(1,1,2),
  col=c("black", "green", "cyan"),
  legend=c("Obs", "Smoothed", "95% CL"))
```



28.3.1.2 用 KFAS

下面改用 KFAS 包进行估计。这个包也支持对结构时间序列模型进行简化的模型设定。

```
library(KFAS)
## 模型设定
kfas01a <- SSModel(
  ts.alcoa ~ SSMtrend(
    1, Q=list(matrix(NA))),
  H=matrix(NA))
## 超参数估计
kfas01b <- fitSSM(
  kfas01a,
  rep(log(var(ts.alcoa)), 2),
  method="BFGS")
c("Var_obs"=c(kfas01b$model$H),
  "Var_level"=c(kfas01b$model$Q))
```

```
##     Var_obs   Var_level
## 0.230651733 0.005404091
```

```
## 用估计的超参数进行滤波平滑
kfas01c <- KFS(kfas01b$model)
```

超参数估计结果与 statespacer 基本一致。计算两个包的平滑结果的最大差距：

```
c(" 水平值平滑差距"=max(abs(
  c(ssr1$smoothed$level) - c(kfas01c$muhat))),
  " 水平值平滑方差差距"=max(abs(
    c(ssr1$smoothed$V) - c(kfas01c$V_mu))))
```

```
##      水平值平滑差距 水平值平滑方差差距
##        3.475598e-05        9.103878e-07
```

可见 statespacer 和 KFAS 两个包的平滑结果是一致的。

28.3.1.3 用 dlm

dlm 包也可以实现这个模型。在超参数 σ_e^2 和 σ_η^2 已知时，可以指定模型如下：

```
library(dlm)
dlm(
  FF = 1,  # $Z_t$ 矩阵
  V = 0.8, # $\sigma_e^2$
  GG = 1,  # $T_t$ 矩阵
  W = 0.1, # $\sigma_{\eta}^2$
  m0 = 0,  # 与 $a_1$ 作用相同，为 $\mu_0$ 期望
  C0 = 100 # 与 $P_1$ 作用相同，为 $\mu_0$ 方差
)
```

```
## $FF
##      [,1]
## [1,]    1
##
## $V
##      [,1]
## [1,]  0.8
##
## $GG
##      [,1]
## [1,]    1
##
## $W
##      [,1]
## [1,]  0.1
##
## $m0
## [1] 0
##
## $C0
##      [,1]
```

```
## [1,]  100
```

也可以用 dlmModPoly() 指定这样的局部水平模型：

```
dlmModPoly(
  order = 1,  # 局部水平模型
  dV = 0.8, # $\sigma_e^2$
  dW = 0.1, # $\sigma_{\eta}^2$
  C0 = 100 # 与 $P_1$ 作用相同，为 $\mu_0$ 方差
)
```

```
## $FF
##      [,1]
## [1,]    1
##
## $V
##      [,1]
## [1,]  0.8
##
## $GG
##      [,1]
## [1,]    1
##
## $W
##      [,1]
## [1,]  0.1
##
## $m0
## [1] 0
##
## $C0
##      [,1]
## [1,]  100
```

为了对超参数进行最大似然估计，需要自定义一个给定超参数后输出 dlm 模型的函数：

```
buildFun <- function(para){
  dlmModPoly(
    order = 1,
    dV = exp(para[[1]]),
    dW = exp(para[[2]])
  )
}
```

注意模型中并没有输入数据。用 dlmMLE() 函数输入数据后对超参数进行最大似然估计：

```
fit_dlm <- dlmMLE(
  ts.alcoa,        # 输入的观测值序列
  parm = c(0, 0), # 超参数对数值的初值
  build = buildFun
)
fit_dlm$conv
```

```
## [1] 0
```

获得估计的超参数对应的 dlm 模型：

```
dlm_al <- buildFun(fit_dlm$par)
```

估计的超参数值：

```
c("Var_obs" = dlm::V(dlm_al),
  "Var_level" = dlm::W(dlm_al))
```

```
##    Var_obs  Var_level
## 0.23065283 0.00540346
```

估计结果与 statespacer 和 KFAS 结果基本相同。为了获得滤波结果，可以调用 `dlmFilter(dlm_al)`，平滑结果使用 `dlmSmooth(dlm_al)`，从结果中可以很容易地提取滤波估计与平滑估计，但是条件分布方差阵估计使用了奇异值分解表示，提取时要使用 `dlmSvd2var()` 函数转换为方差阵。

28.3.1.4 用 MARSS

局部水平模型写成 MARSS 需要的形式为：

$$
\begin{aligned}
x_t &= 1 \cdot x_{t-1} + 0 + w_t, \quad & w_t &\sim \mathrm{N}(0, q), \\
y_t &= 1 \cdot x_t + 0 + v_t, & r_t &\sim \mathrm{N}(0, r), \\
x_0 &= \mu.
\end{aligned}
$$

估计：

```
library(MARSS)
fit_marss_al <- MARSS(ts.alcoa, model=list(
  B = "identity",
  U = "zero",
  Q = matrix("q"),
  Z = "identity",
  A = "zero",
  R = matrix("r"),
  x0 = matrix("mu")) )
```

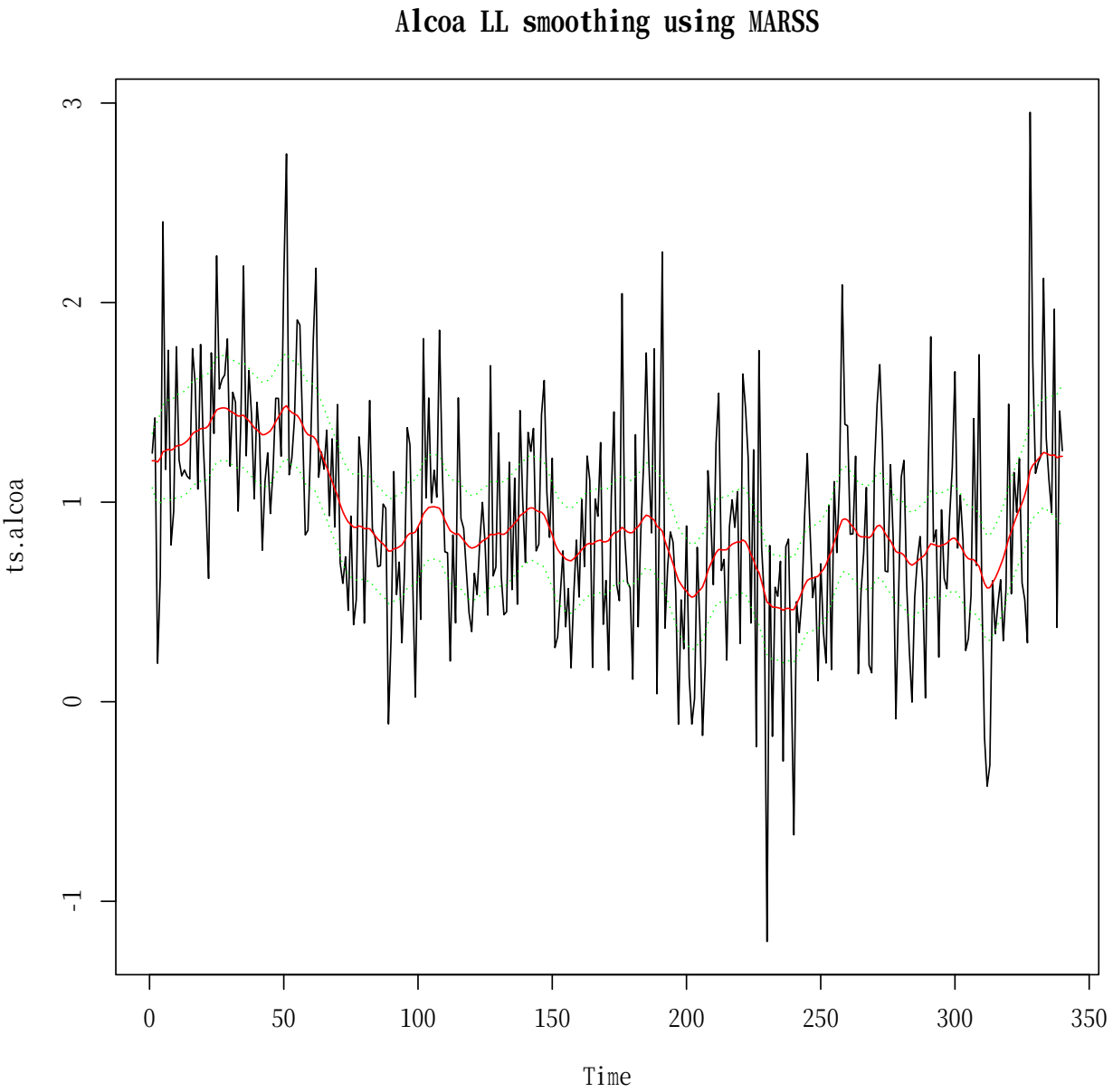
```
## Success! abstol and log-log tests passed at 132 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
```

```
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 132 iterations.
## Log-likelihood: -258.2761
## AIC: 522.5521   AICc: 522.6235
##
##       Estimate
## R.r    0.22958
## Q.q    0.00567
## x0.mu  1.20767
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

两个方差的估计与 statespacer 结果相近，不完全相同。

状态的平滑结果用 `tsSmooth(object, type="xtT")` 提取，均值为`.estimate`，标准误差为`.se`。如：

```
plot(ts.alcoa, main="Alcoa LL smoothing using MARSS")
res_sm <- tsSmooth(fit_marss_al, type="xtT")
lines(res_sm$.estimate,
  col = "red")
lines(res_sm$.estimate - 1.96*res_sm$.se,
  col = "green", lty=3)
lines(res_sm$.estimate + 1.96*res_sm$.se,
  col = "green", lty=3)
```

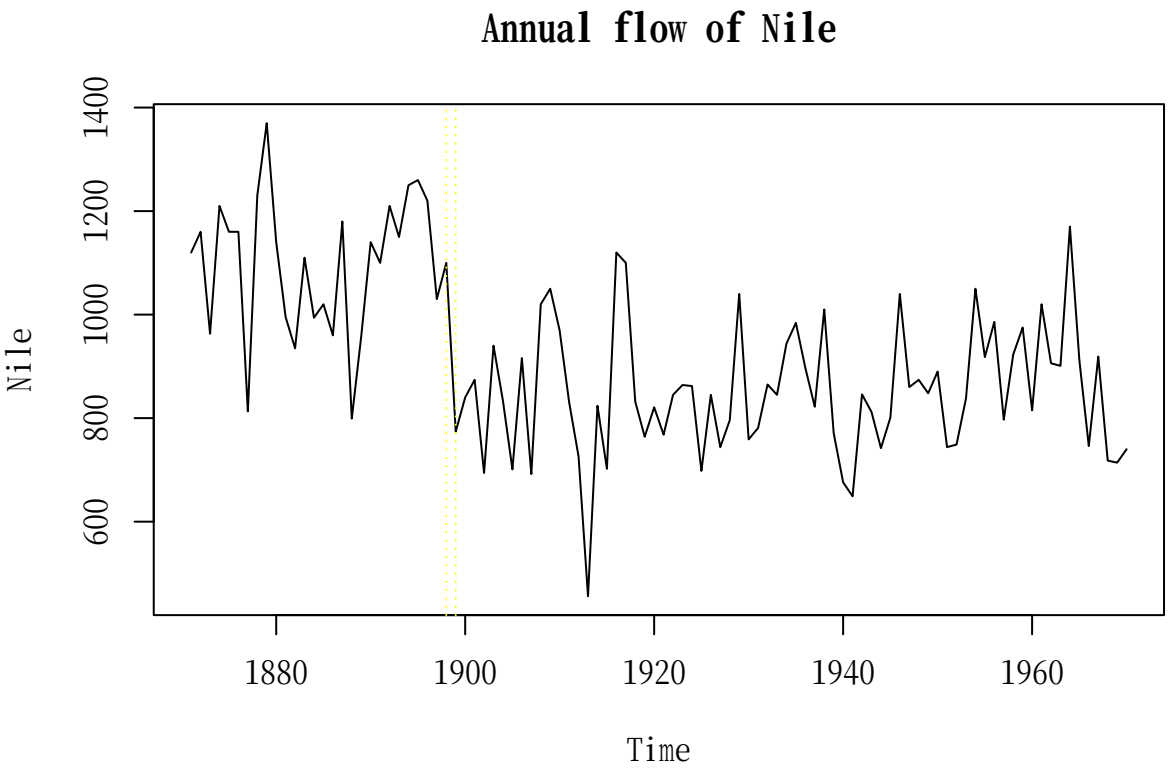


28.3.2 尼罗河流量带变点的局部水平模型

例 28.5. 考虑尼罗河 1871-1970 年的年流量数据，单位是 10^8 立方米，在 1898-1899 年有明显的变点。用局部水平模型建模并考虑变点的拟合。

基本 R 的 datasets 包中 Nile 变量为长度 100 的时间序列：

```
plot(Nile, main="Annual flow of Nile")
abline(v = c(1898, 1899), lty=3, col="yellow")
```



28.3.2.1 用 DLM 包

使用 dlm 包，输入超参数建立 dlm 模型的函数：

```
library(dlm)
buildFun <- function(para){
  mod <- dlmModPoly(
    order = 1,
    dV = exp(para[[1]]) # $\sigma_e^2$, 非时变
  )
  mod$JW <- matrix(1)   # 示性函数，表示 $\sigma_{\eta_t}^2$ 时变
  # mod$X: 用来保存时变的成分
  mod$X <- cbind(rep(exp(para[[2]]), length(Nile)))
  # 给 t=1898 年的观测的 $\eta_t$ 方差放大
  jj <- which(time(Nile) == 1899)[1]
  mod$X[jj,1] <- mod$X[jj,1] * (1 + exp(para[[3]]))

  mod
}
```

在上面的模型中，状态方差的扰动项 η_t 的方差是时变的，但仅在 1899 年与其它年份不同。为了容许模型中的矩阵时变，在 dlm 中用 JW 作为示性函数，表示状态方程的扰动项方差阵哪些元素是时变的，然后将时变的内容保存在 X 矩阵中，矩阵的行数等于观测长度 n。用第三个超参数表示 1899 年的状态方程扰动项方差放大倍数（对数尺度）。

三个超参数的最大似然估计：

```
fit_nile <- dlmMLE(
  Nile,
  parm = c(0, 0, 0),
  build = buildFun)
fit_nile$conv
```

```
## [1] 0
```

获取估计的模型：

```
dlm_nile <- buildFun(fit_nile$par)
```

σ_e^2 估计：

```
V(dlm_nile)
```

```
##          [,1]
## [1,] 16300.33
```

除去变点以外的所有观测的 η_t 方差估计，以及 1899 年的 η_t 方差估计：

```
dlm_nile$X[c(1, which(time(Nile) == 1899)[1]), 1]
```

```
## [1] 2.792224e-02 6.048379e+04
```

可以看出，非变点处的 η_t 的标准差，与观测 y_t 的取值范围相比，基本上等于 0。

计算滤波和平滑估计：

```
filt_nile <- dlmFilter(Nile, dlm_nile)
sm_nile <- dlmSmooth(filt_nile)
```

提取平滑估计：

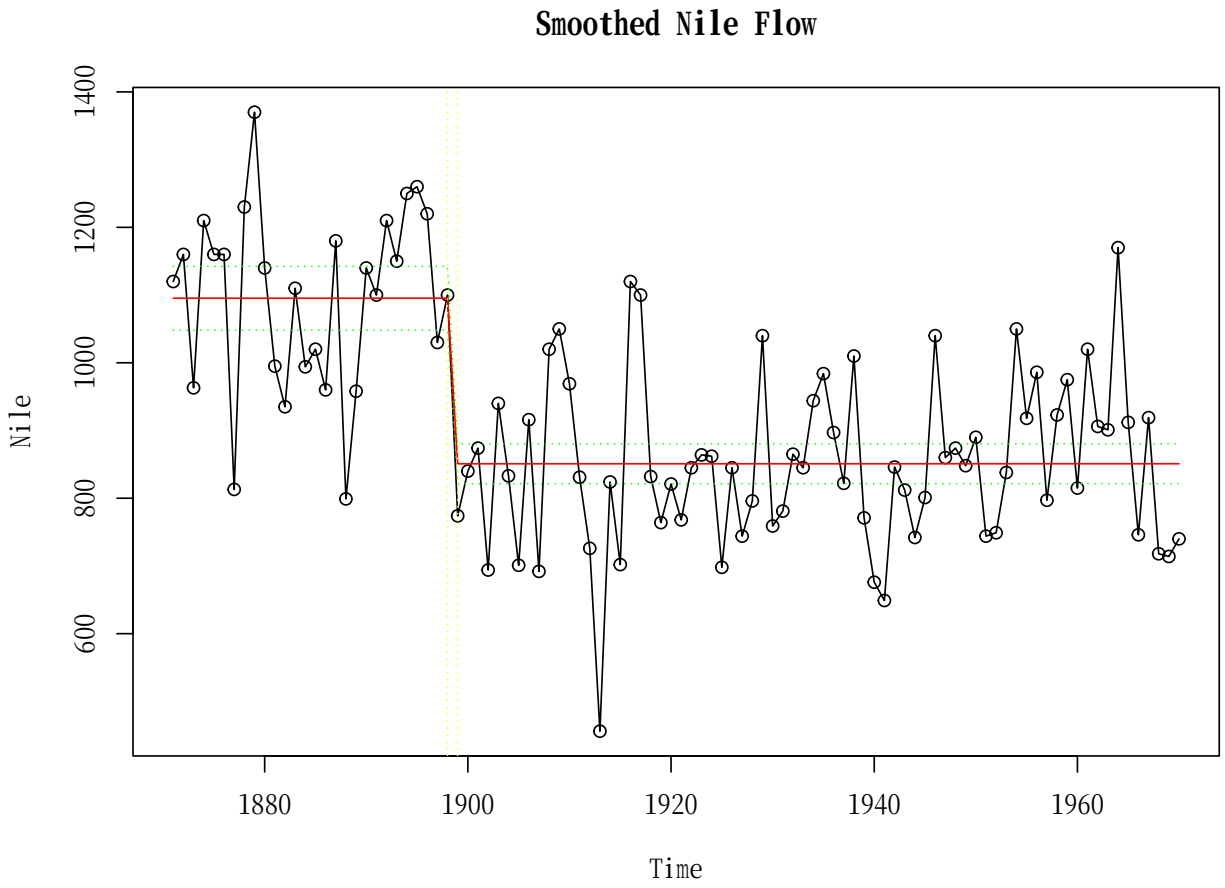
```
sme_nile <- dropFirst(sm_nile$s)
```

提取平滑分布方差：

```
smv_nile <- dlmSvd2var(sm_nile$U.S, sm_nile$D.S) |>
  unlist() |>
  dropFirst()
```

作原始序列图，添加平滑结果曲线（红色），平滑区间（绿色）：

```
plot(Nile, type="o", main="Smoothed Nile Flow")
lines(sme_nile, col="red")
lines(sme_nile - 1.96 * sqrt(smv_nile), lty=3, col="green")
lines(sme_nile + 1.96 * sqrt(smv_nile), lty=3, col="green")
abline(v=c(1898, 1899), lty=3, col="yellow")
```



这个模型中的 η_t 的方差是时变的，水平在 1898 到 1999 之间有一个变点，在变化之前的 η_t 的方差基本恒定，水平也基本恒定，变化之后也是如此。

28.3.2.2 用 MARSS 包

为了演示 MARSS 的用法，先考虑一个恒定水平模型：

$$y_t = a + \nu_t,\ \nu_t \sim \mathrm{N}(0, R).$$

将其写成 MARSS 需要的形式：

$$\begin{aligned} x_t &= 1 \times x_{t-1} + 0 + w_t,\ w_t \sim \mathrm{N}(0, 0), \\ y_t &= 1 \times x_t + 0 + \nu_t,\ \nu_t \sim \mathrm{N}(0, R), \\ x_0 &= a. \end{aligned}$$

因为 $x_0 = a$，$w_t = a$，所以 $x_t \equiv a$，不需要估计变化的 x_t。程序可以写成

```
library(MARSS)
fit1_mn <- MARSS(Nile, model = list(
  B = "identity",
  U = "zero",
  Q = "zero",
```

```
  Z = "identity",
  A = "zero",
  R = matrix("r"),
  x0 = matrix("a")) )
```

```
## Success! algorithm run for 15 iterations. abstol and log-log tests passed.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Algorithm ran 15 (=minit) iterations and convergence was reached.
## Log-likelihood: -654.5157
## AIC: 1313.031   AICc: 1313.155
##
##      Estimate
## R.r     28352
## x0.a      919
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

然后，考虑趋势为非随机直线的模型。即

$$y_t = a + ut + \nu_t.$$

写成 MARSS 标准形式：

$$\begin{aligned} x_t =& 1 \times x_{t-1} + u + w_t,\ w_t \sim \mathrm{N}(0, 0), \\ y_t =& 1 \times x_t + 0 + \nu_t,\ \nu_t \sim \mathrm{N}(0, R), \\ x_0 =& a. \end{aligned}$$

程序为

```
fit2_mn <- MARSS(Nile, model = list(
  B = "identity",
  U = matrix("u"),
  Q = "zero",
  Z = "identity",
  A = "zero",
  R = matrix("r"),
  x0 = matrix("a")) )
```

```
## Success! abstol and log-log tests passed at 18 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
```

```
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 18 iterations.
## Log-likelihood: -642.3159
## AIC: 1290.632   AICc: 1290.882
##
##      Estimate
## R.r  22213.60
## U.u     -2.69
## x0.a  1054.94
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

拟合不考虑变点的局部水平模型。MARSS 模型公式为

$$
\begin{aligned}
x_t =& 1 \times x_{t-1} + 0 + w_t,\ w_t \sim \mathrm{N}(0, Q),\\
y_t =& 1 \times x_t + 0 + \nu_t,\ \nu_t \sim \mathrm{N}(0, R),\\
x_0 =& x_0.
\end{aligned}
$$

为了得到更精确的结果，可以在用 EM 估计后，用估计结果作为初值，使用 BFGS 算法估计：

```
fit3_mn_em <- MARSS(Nile, model = list(
  B = "identity",
  U = "zero",
  Q = matrix("q"),
  Z = "identity",
  A = "zero",
  R = matrix("r"),
  x0 = matrix("x0")) )
```

```
## Success! abstol and log-log tests passed at 74 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 74 iterations.
## Log-likelihood: -637.7631
## AIC: 1281.526   AICc: 1281.776
##
##       Estimate
```

```
## R.r      15066
## Q.q       1425
## x0.x0     1112
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

```
fit3_mn <- MARSS(Nile, model = list(
  B = "identity",
  U = "zero",
  Q = matrix("q"),
  Z = "identity",
  A = "zero",
  R = matrix("r"),
  x0 = matrix("x0")),
  inits = fit3_mn_em$par,
  method = "BFGS")
```

```
## Success! Converged in 10 iterations.
## Function MARSSkfas used for likelihood calculation.
##
## MARSS fit is
## Estimation method: BFGS
## Estimation converged in 10 iterations.
## Log-likelihood: -637.7451
## AIC: 1281.49   AICc: 1281.74
##
##       Estimate
## R.r      15337
## Q.q       1218
## x0.x0     1112
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

下面考虑随机漂移项中带有一个非时变的漂移项的模型。这只要添加 U 为待估参数：

```
fit4_mn <- MARSS(Nile, model = list(
  B = "identity",
  U = matrix("u"),
  Q = matrix("q"),
  Z = "identity",
  A = "zero",
```

```
  R = matrix("r"),
  x0 = matrix("mu")) )
```

```
## Success! abstol and log-log tests passed at 94 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 94 iterations.
## Log-likelihood: -637.3027
## AIC: 1282.605   AICc: 1283.026
##
##       Estimate
## R.r   15585.28
## U.u      -3.25
## Q.q    1088.99
## x0.mu  1124.04
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

按 AIC 比较，增加漂移项并没有改善。

带有随机漂移项的模型，即具有随机斜率项的模型：

$$
\begin{aligned}
y_t =& (1,0)\begin{pmatrix} x_t \\ u_t \end{pmatrix} + \nu_t,\ \nu_t \sim \mathrm{N}(0,R), \\
\begin{pmatrix} x_t \\ u_t \end{pmatrix} =& \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x_{t-1} \\ u_{t-1} \end{pmatrix} + \begin{pmatrix} w_t \\ z_t \end{pmatrix},\ \begin{pmatrix} w_t \\ z_t \end{pmatrix} \sim \mathrm{N}(0,\mathrm{diag}(Q,P)), \\
\begin{pmatrix} x_0 \\ u_0 \end{pmatrix} =& \begin{pmatrix} x_0 \\ u_0 \end{pmatrix}.
\end{aligned}
$$

程序为（为了算法收敛先指定初值用 EM 粗略估计，然后用 BFGS 估计）：

```
mod5 <- list(
  B = rbind(c(1,1),
            c(0,1)),
  U = "zero",
  Q = ldiag(c("q", "p")),
  Z = rbind(c(1, 0)),
  A = "zero",
  R = matrix("r"),
  x0 = cbind(c("x0", "u0")))
```

```
fit5_mn_tmp <- MARSS(Nile, model = mod5,
  inits = list(x0 = cbind(c(1000, -4))),
  control = list(maxit = 20),
  silent = TRUE)
fit5_mn <- MARSS(Nile, model = mod5,
  inits = fit5_mn_tmp,
  method = "BFGS")
```

```
## Success! Converged in 86 iterations.
## Function MARSSkfas used for likelihood calculation.
##
## MARSS fit is
## Estimation method: BFGS
## Estimation converged in 86 iterations.
## Log-likelihood: -637.278
## AIC: 1284.556   AICc: 1285.194
##
##        Estimate
## R.r    1.61e+04
## Q.q    8.43e+02
## Q.p    3.69e-07
## x0.x0  1.12e+03
## x0.u0 -3.11e+00
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

估计结果中随机斜率项的方差近似为 0，所以斜率项实际是常数，从 AIC 看模型没有改善。

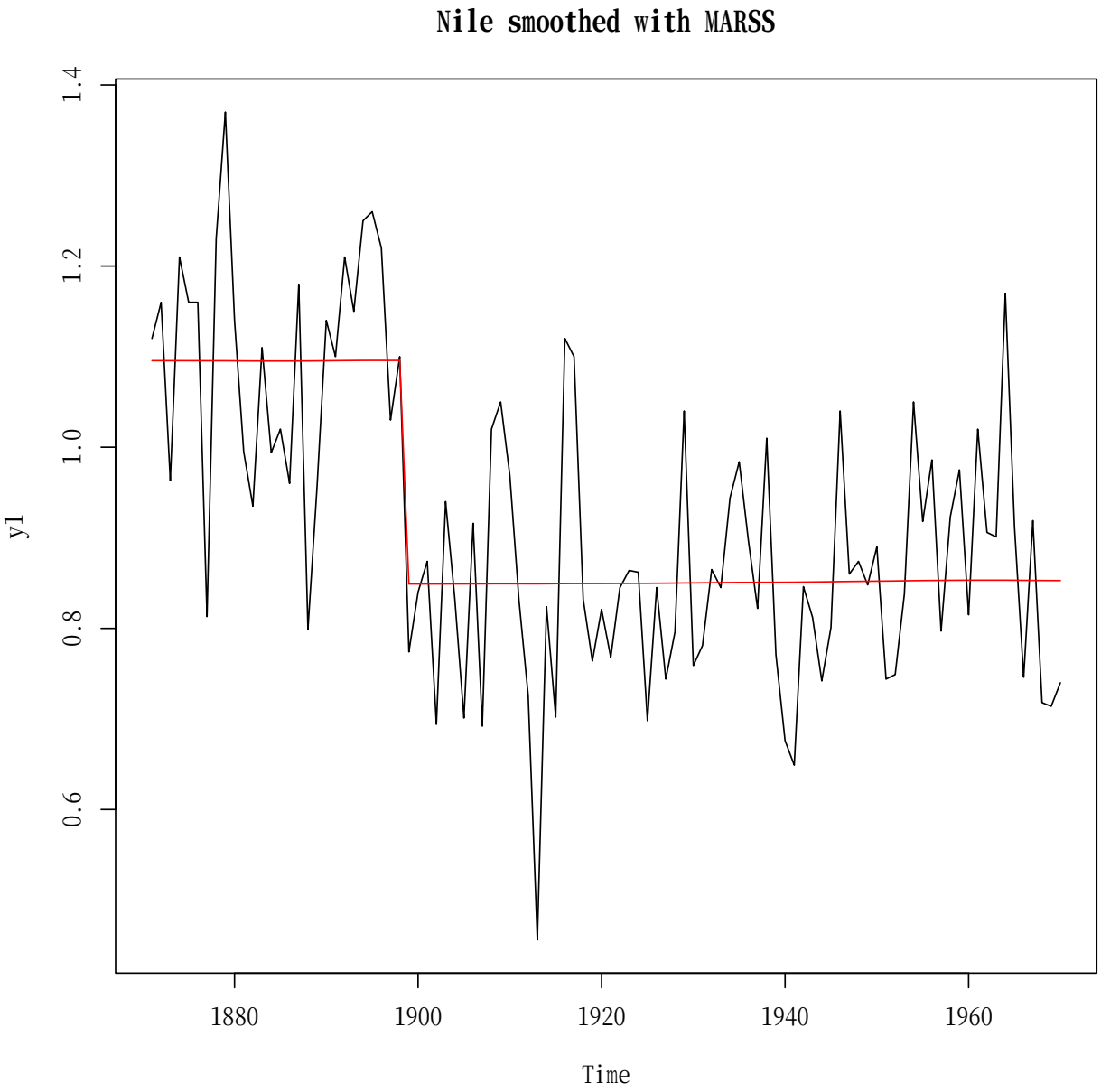
为了拟合 1898-1899 年的变点，考虑 1899 年的系统方程方差很大的模型。为了模型中的矩阵时变，需要使用三维数组，将时间 t 作为第三维。如：

```
jj <- which(time(Nile) == 1899)[1]
Q1 <- array("q", dim=c(1,1, length(Nile)))
Q1[1,1,jj] <- "q2"
y1 <- Nile / 1000
mod_list <-  list(
  B = "identity",
  U = "zero",
  Q = Q1,
  Z = "identity",
  A = "zero",
  R = matrix("r"),
  x0 = matrix("mu"))
```

```
fit6 <- MARSS(y1,
  model = mod_list,
  control = list(maxit = 1000))
```

```
## Warning! Abstol convergence only. Maxit (=1000) reached before log-log convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## WARNING: Abstol convergence only no log-log convergence.
##  maxit (=1000) reached before log-log convergence.
##  The likelihood and params might not be at the ML values.
##  Try setting control$maxit higher.
## Log-likelihood: 61.58321
## AIC: -115.1664   AICc: -114.7454
##
##       Estimate
## R.r   1.61e-02
## Q.q   3.58e-06
## Q.q2  6.13e-02
## x0.mu 1.10e+00
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
##
## Convergence warnings
##  Warning: the  Q.q  parameter value has not converged.
##  Type MARSSinfo("convergence") for more info on this warning.
```

```
sm <- tsSmooth(fit6, type="xtT")
plot(y1, main="Nile smoothed with MARSS")
lines(as.vector(time(Nile)), sm$.estimate, col="red")
```



从 AIC 值比较，这个模型比前面的模型要好得多。系统方程在变点以外的误差项方差基本等于零。

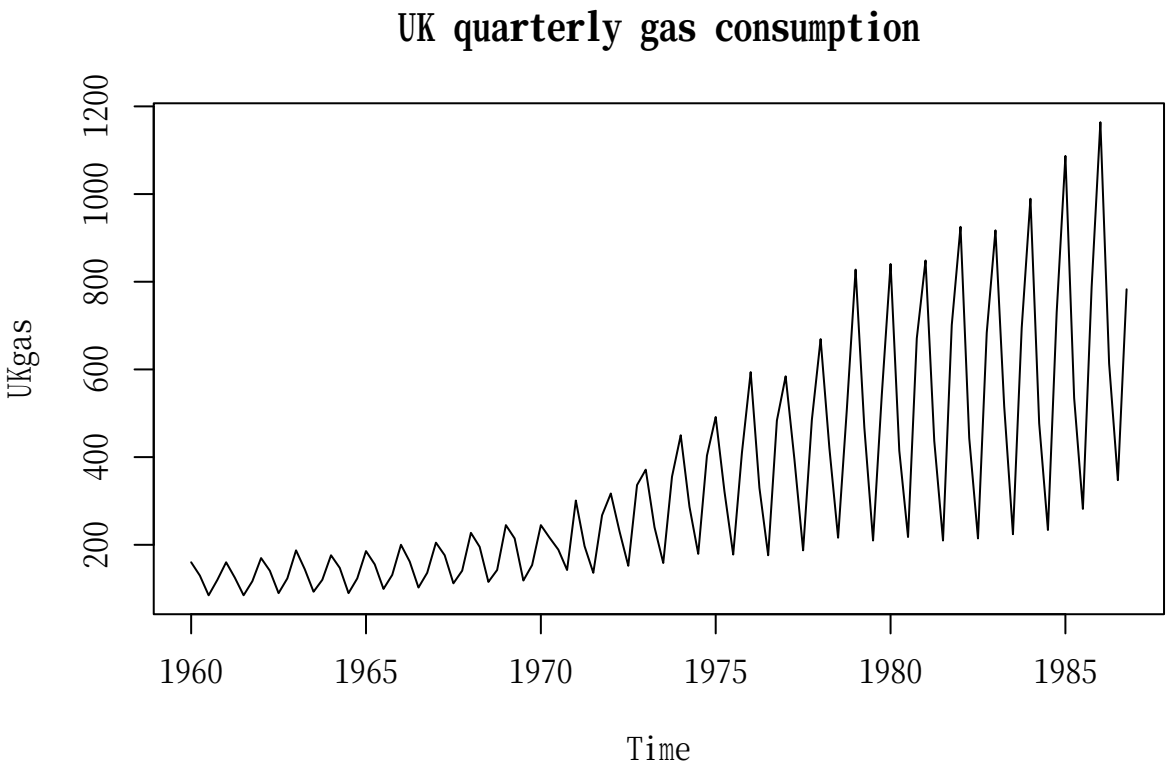
MARSS 包与 DLM 包得到的变点处的系统方程方差不同，这是因为只有一个变点，其方差估计本身就变化很大。

MARSS 默认使用 EM 算法估计超参数，这种算法常常比较慢。一种方法是先用 EM 算法获得初步估计，然后使用 BFGS 等算法加速计算。

28.3.3 英国燃气消耗量结构时间序列建模

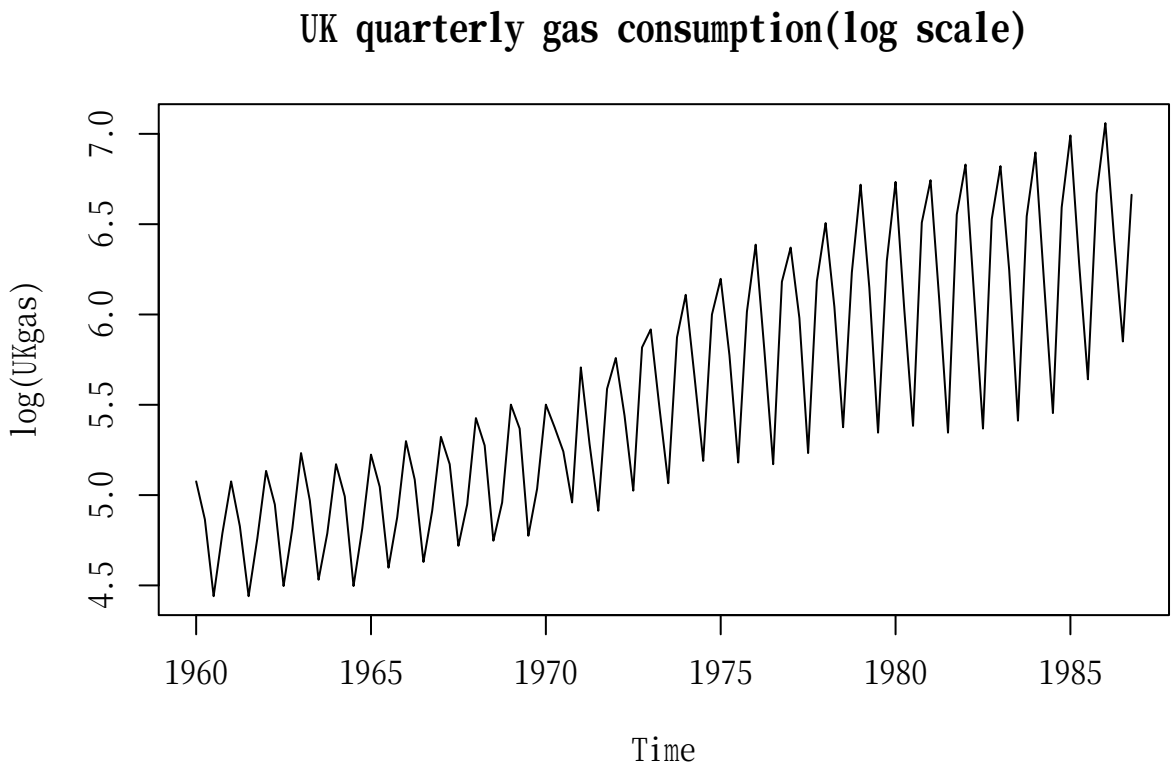
基本 R 的 datasets 包的 UKgas 变量保存了英国 1960 年到 1986 年的季度燃气消耗时间序列。使用结构时间序列建模，需要有趋势部分和季节部分。

```
plot(UKgas, main="UK quarterly gas consumption")
```



因为趋势与季节部分呈现乘性的组合，改用对数值：

```
plot(log(UKgas), main="UK quarterly gas consumption(log scale)")
```



28.3.3.1 用 dlm 包

dlm 包支持结构时间序列模型的简单设定。

指定模型：

```
library(dlm)
lGas <- log(UKgas)
dlmGas <- dlmModPoly() + dlmModSeas(4)
```

函数 dlmModPoly() 会指定一个有局部水平和水平的增量（时变斜率）的模型，dlmModSeas(4) 指定季节模型，用加号表示这两部分的状态分别建模，在观测方程中将两部分结果相加。

拟合模型：

```
buildFun <- function(para){
  dlmGas <- dlmModPoly() + dlmModSeas(4)
  # 仅指定了时变斜率和季节项的方差,
  # 没有指定局部水平的方差
  diag(W(dlmGas))[2:3] <- exp(para[1:2])
  V(dlmGas) <- exp(para[3])

  dlmGas
}
```

```
fit_gas <- dlmMLE(
  lGas,
  parm = log(c(10^-5, 10^-3, 10^-3)),
  build = buildFun )
fit_gas$conv
```

```
## [1] 0
```

估计得到的模型：

```
dlmGas <- buildFun(fit_gas$par)
```

观测方程方差：

```
drop(V(dlmGas))
```

```
## [1] 0.001822492
```

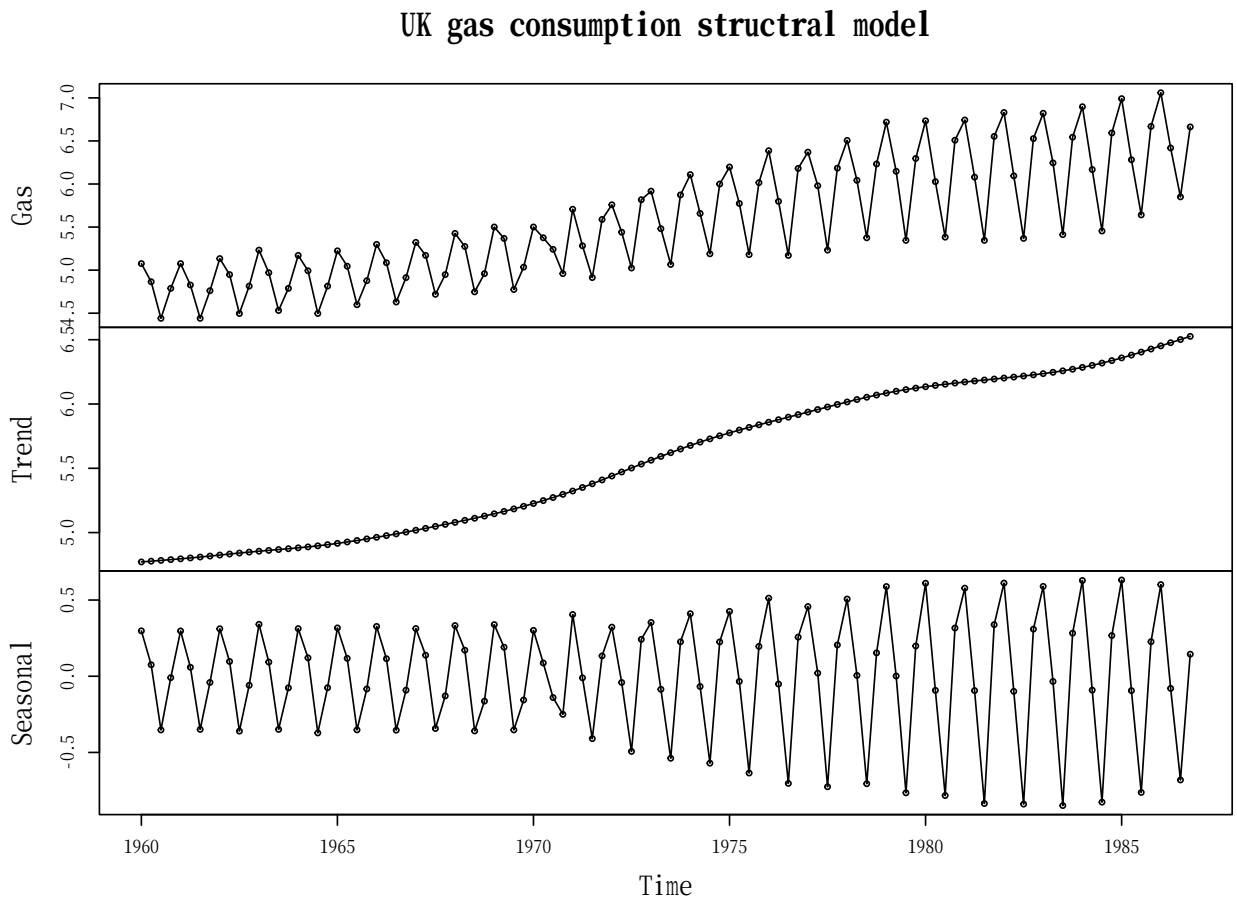
状态方程方差：

```
diag(W(dlmGas))[2:3]
```

```
## [1] 7.901263e-06 3.308590e-03
```

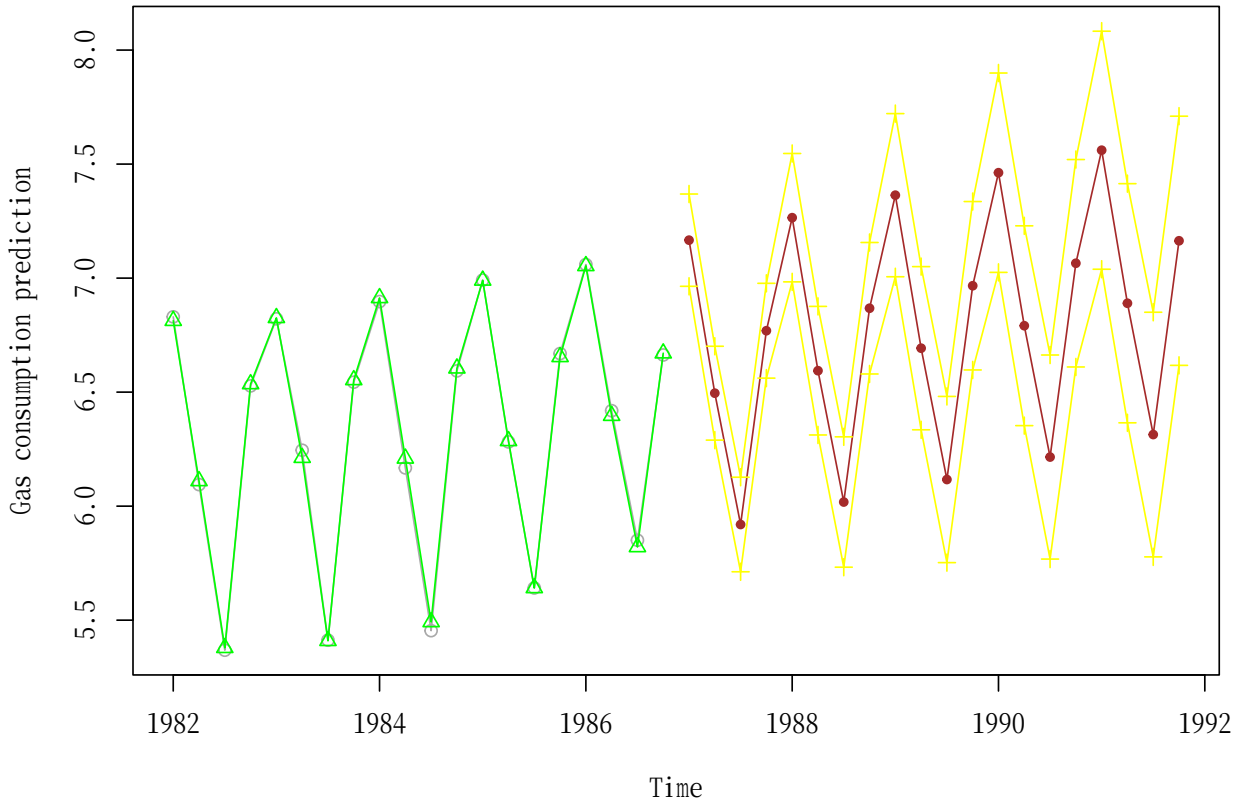
获取平滑得到的水平、季节项：

```
sm_gas <- dlmSmooth(lGas, mod=dlmGas)
da_tmp <- cbind(
  lGas,
  dropFirst(sm_gas$s[, c(1,3)]))
colnames(da_tmp) <- c("Gas", "Trend", "Seasonal")
plot(da_tmp, type="o",
  main = "UK gas consumption structral model")
```



预报分布的均值和方差可以用 `dlmForecast()` 获得。下面的预测区间是 90% 置信度的。

```
filt_gas <- dlmFilter(lGas, mod = dlmGas)
fore_gas <- dlmForecast(filt_gas, nAhead = 20)
# 预测
gas_pred20 <- fore_gas$f[,1]
gas_se20 <- sqrt(unlist(fore_gas$Q))
da_tmp <- ts.union(
  window(lGas, start=c(1982,1)),
  window(sm_gas$s[,1] + sm_gas$s[,3], start=c(1982,1)),
  gas_pred20,
  gas_pred20 - 1.96 * gas_se20,
  gas_pred20 + 1.96 * gas_se20)
plot(da_tmp, plot.type = "single",
  type = "o",
  pch = c(1, 2, 20, 3, 3),
  col = c("darkgrey", "green",
    "brown", "yellow", "yellow"),
  ylab = "Gas consumption prediction"
  )
```



绿色三角形部分是平滑结果，其中黑色圆圈是真实值；棕色部分是多步预测结果，黄色是 95% 预测区间。

使用 dlm 发现，如果在 `dlmMLE()` 中使用 `parm=rep(0, 3)`，则将原始数据作自然对数变换的结果比较好，状态方程扰动方差和观测方程扰动方差很小，但改为常用对数变换时，状态方程扰动项方差会增大一两个数量级，使得上图中的预测区间会大得不可接受。如果在 `dlmMLE()` 中人为取 `parm` 为较合适的初值，就可以得到与自然对数变换相同的结果，这说明 dlm 计算最大似然估计可能不太可靠，很可能落入局部极值点。`dlmMLE()` 的结果会依赖于参数的初始值 `parm` 的选择。

28.3.3.2 用 MARSS

将季度数据的结构时间序列模型写成：

$$
\begin{aligned}
x_t = \begin{pmatrix} \mu_t \\ \mu_t^b \\ \gamma_t \\ \gamma_{t-1} \\ \gamma_{t-2} \end{pmatrix} &= \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} x_{-1} + w_t, \\
w_t &\sim \mathrm{N}(0, \mathrm{diag}(q_\mu, q_{\mu^b}, q_\gamma, 0)) \\
y_t &= (1,0,1,0,0)x_t + \nu_t, \quad \nu_t \sim \mathrm{N}(0, R), \\
x_0 &\sim \mathrm{N}(\mu_0, V_0).
\end{aligned}
$$

制作这样的模型设定用列表：

```
build_sts4 <- function(y){
  B1 <- matrix(c(
    1, 1, 0, 0, 0,
    0, 1, 0, 0, 0,
    0, 0, -1,-1,-1,
    0, 0, 1, 0, 0,
    0, 0, 0, 1, 0 ),
    nrow = 5, byrow=TRUE)
  Q1 <- ldiag(list("qmu", "qmub", "qgam", 0, 0))

  Z1 <- rbind(c(1, 0, 1, 0, 0))
  R1 <- matrix("r")
  A1 <- matrix(0)

  x0 <- cbind(c(y[1], rep(0, 4)))
  vy <- var(y)/100
  ## 扩散先验
  #V0 <- matrix(vy*1E-6, 5, 5) + diag(1E-10, 5)
  V0 <- matrix(vy*1E6, 5, 5) + diag(1E-10, 5)

  list(
    B = B1,
    Q = Q1,
    Z = Z1,
    R = R1,
    U = "zero",
    A = "zero",
    x0 = x0, V0 = V0
    )
}
```

这里 V0 参数是 x_0 的分布方差，这里指定了一个发散先验，即很大的方差。

建模：

```
library(MARSS)
gas_marss <- MARSS(lGas,
  model = build_sts4(lGas),
  method="BFGS")
```

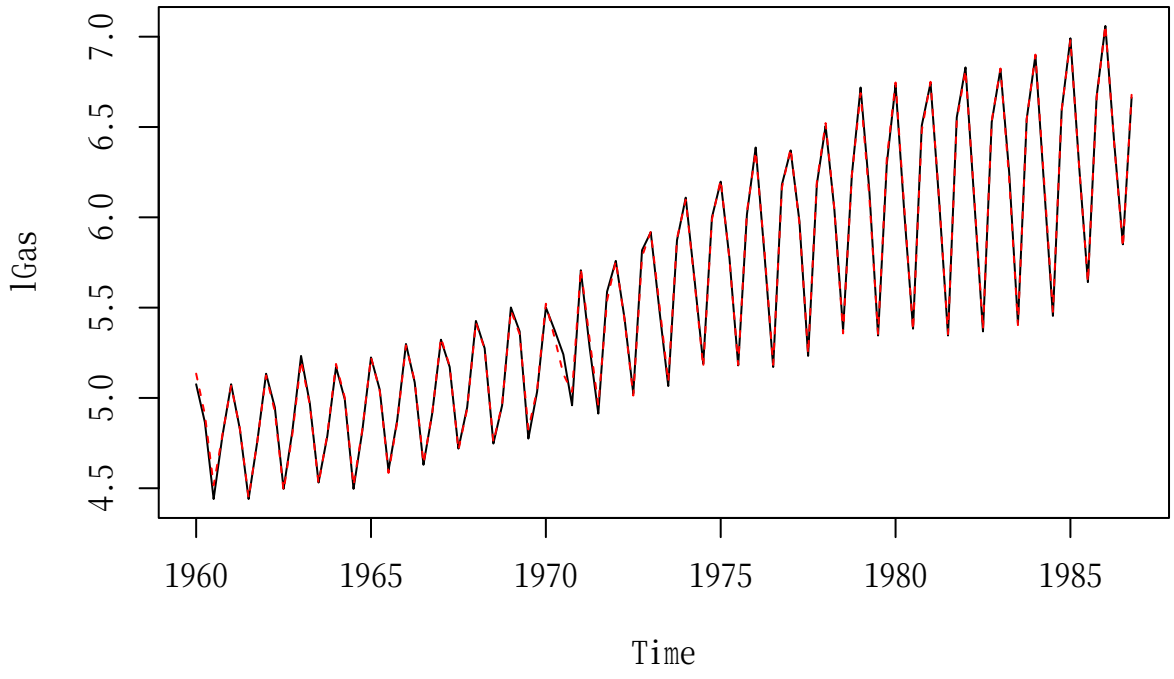
```
## Success! Converged in 28 iterations.
## Function MARSSkfas used for likelihood calculation.
##
## MARSS fit is
```

```
## Estimation method: BFGS
## Estimation converged in 28 iterations.
## Log-likelihood: 63.92348
## AIC: -119.847   AICc: -119.4586
##
##         Estimate
## R.r     1.93e-03
## Q.qmu   2.44e-08
## Q.qmub 9.26e-05
## Q.qgam 3.79e-03
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

带有季节成分的模型使用 MARSS 默认的 EM 算法往往比较困难，所以这里用了 BFGS 算法。

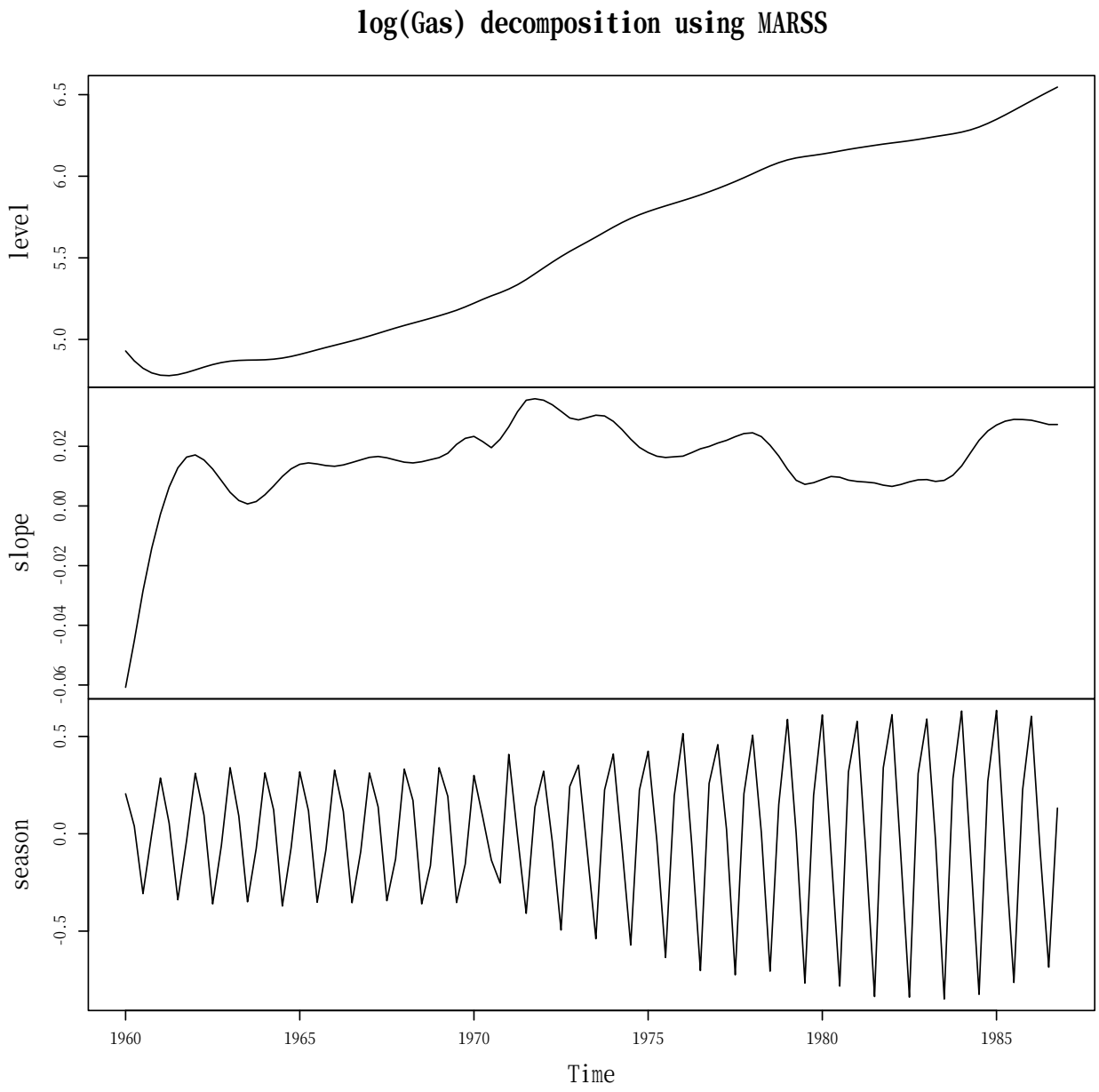
平滑结果：

```
sm <- fitted(gas_marss, type="ytT")
plot(lGas)
lines(as.vector(time(lGas)), sm$.fitted, col="red", lty=2)
```



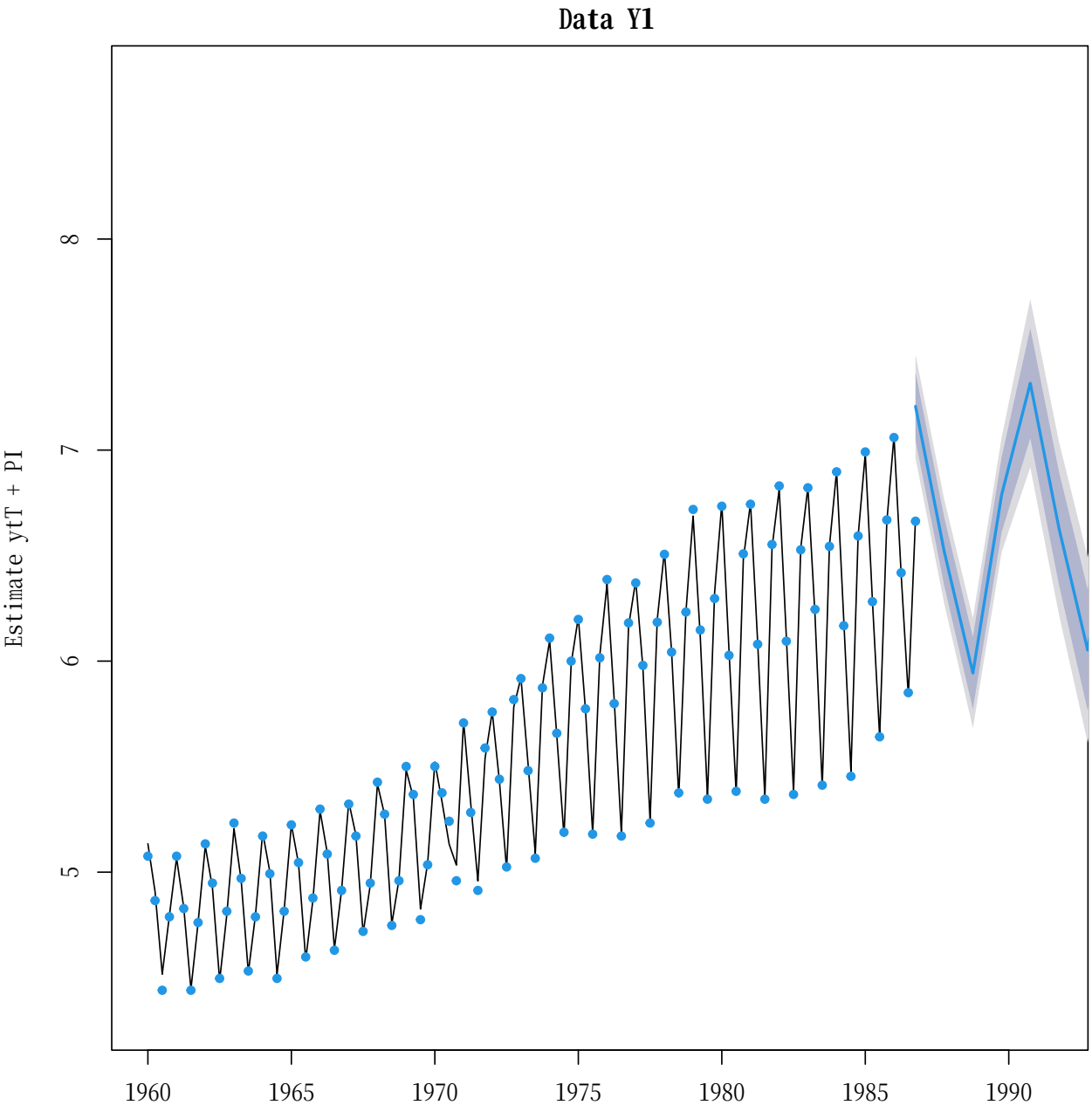
拟合误差很小，这也可以从估计的观测误差方差 (`R.r`) 远小于水平的误差方差 (`Q.qmu`) 解释。各个分解成分情况：

```
sm2 <- tsSmooth(gas_marss, type="xtT")
sm2t <- sm2 |>
  pivot_wider(
    id_cols = "t",
    names_from = ".rownames",
    values_from = ".estimate"  ) |>
  rename(level = X1, slope = X2, season = X3)
plot(ts(sm2t[,2:4], start=start(lGas), frequency=4),
  main="log(Gas) decomposition using MARSS")
```



预测：

```
library(forecast)
gas_fore <- forecast(gas_marss, h=20, interval="prediction")
plot(gas_fore, showgap=FALSE)
```



预测图形中深浅不同的阴影分别是 80% 和 95% 预测区间。所谓预测区间，就是考虑了观测误差的影响。

可以用 `residuals()` 提取估计残差。

28.4 ARMA 和 ARIMA 模型

考虑 ARMA(p,q) 模型，取 $r=\max(p,q+1)$，则模型可以写成

$$y_t=\sum_{j=1}^{r}\phi_j y_{t-j}+\zeta_t+\sum_{j=1}^{r-1}\theta_j\zeta_{t-j},$$

其中某些系数可以为 0。将此模型写成状态空间模型形式，有多种不同形式，这里给出一种形式比较复杂但计算方便的表达形式：

$$\begin{aligned}\alpha_t=&\begin{pmatrix}y_t & \\ \phi_2 y_{t-1}+\cdots+\phi_r y_{t-(r-1)} & +\theta_1\zeta_t+\cdots+\theta_{r-1}\zeta_{t-(r-2)}\\ \phi_3 y_{t-1}+\cdots+\phi_r y_{t-(r-2)} & +\theta_2\zeta_t+\cdots+\theta_{r-1}\zeta_{t-(r-3)}\\ \vdots & \vdots\\ \phi_r y_{t-1} & +\theta_{r-1}\zeta_t\end{pmatrix}\\ y_t=&(1,0,0,\ldots,0)\alpha_t\\ \alpha_{t+1}=&\begin{pmatrix}\phi_1 & 1 & 0 & \cdots & 0\\ \phi_2 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & & \ddots & \vdots\\ \phi_{r-1} & 0 & 0 & \cdots & 1\\ \phi_r & 0 & 0 & \cdots & 0\end{pmatrix}\alpha_t+\begin{pmatrix}1\\ \theta_1\\ \theta_2\\ \vdots\\ \theta_{r-1}\end{pmatrix}\zeta_{t+1}.\end{aligned}$$

对一个 ARIMA(2,1,1)，模型

$$\begin{aligned}y_t^*=&y_t-y_{t-1},\\ y_t^*=&\phi_1 y_{t-1}^*+\phi_2 y_{t-2}^*+\zeta_t+\theta_1\zeta_{t-1},\end{aligned}$$

可以写成

$$\begin{aligned}\alpha_t=&\begin{pmatrix}y_{t-1}\\ y_t^*\\ \phi_2 y_{t-1}^*+\theta_1\zeta_t\end{pmatrix}\\ y_t=&(1,1,0)\alpha_t\\ \alpha_{t+1}=&\begin{pmatrix}1 & 1 & 0\\ 0 & \phi_1 & 1\\ 0 & \phi_2 & 0\end{pmatrix}\alpha_t+\begin{pmatrix}0\\ 1\\ \theta_1\end{pmatrix}\zeta_{t+1}\end{aligned}$$

观测方程没有观测误差。

更一般的 ARIMA 以及带有乘性季节部分的 ARIMA 都可以类似表示。将 ARIMA 模型表示成状态空间模型以后，好处是状态空间模型的一系列工具都可以用在 ARIMA 的模型推断中，比如 SSM 的精确最大似然估计和初始化方法都可以使用。现在的估计 ARIMA 模型的软件中许多都是利用状态空间模型形式进行估计和推断。

反过来，许多状态空间模型的观测值也服从 ARIMA 模型，比如局部趋势模型，带有斜率项的局部趋势模型等，加入了季节项的结构时间序列模型等。但是，结构时间序列模型转换成 ARIMA 模型会丢失一些有可解释含义的信息。

状态空间模型的优点：

- 状态空间模型更灵活，能够将随着时间增加的已知的机制变化添加到模型中，而 ARIMA 则很难修改；
- 容易处理缺失值；
- 很容易增加额外的解释变量，有回归自变量时允许回归系数为时变系数，很容易进行日历调整；

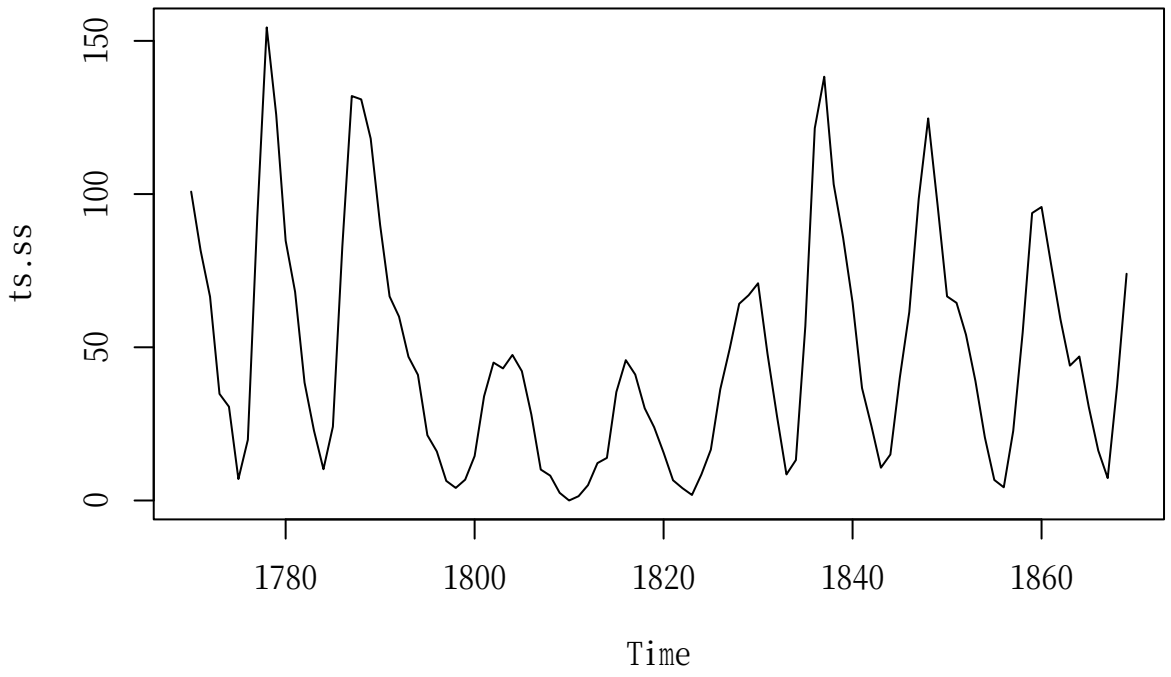
- 预测不需要单独的理论；
- 不要求平稳或者差分后平稳。

另一方面，ARIMA 模型则无法将趋势成分、季节项成分提取出来，要求差分后平稳，而经济和金融中许多数据是差分后也不平稳的。

例 28.6. 考虑太阳黑子数的 ARMA 建模。数据为 1700-1988 的年数据。

数据：

```
library(datasets)
ts.ss <- window(sunspot.year, start = 1770, end = 1869)
plot(ts.ss)
```



可以看出，数据有周期性。

用 R 的 arima 函数建立 AR(3) 模型：

```
ss1 <- arima(
  ts.ss, order=c(3,0,0))
ss1
```

```
##
## Call:
## arima(x = ts.ss, order = c(3, 0, 0))
##
```

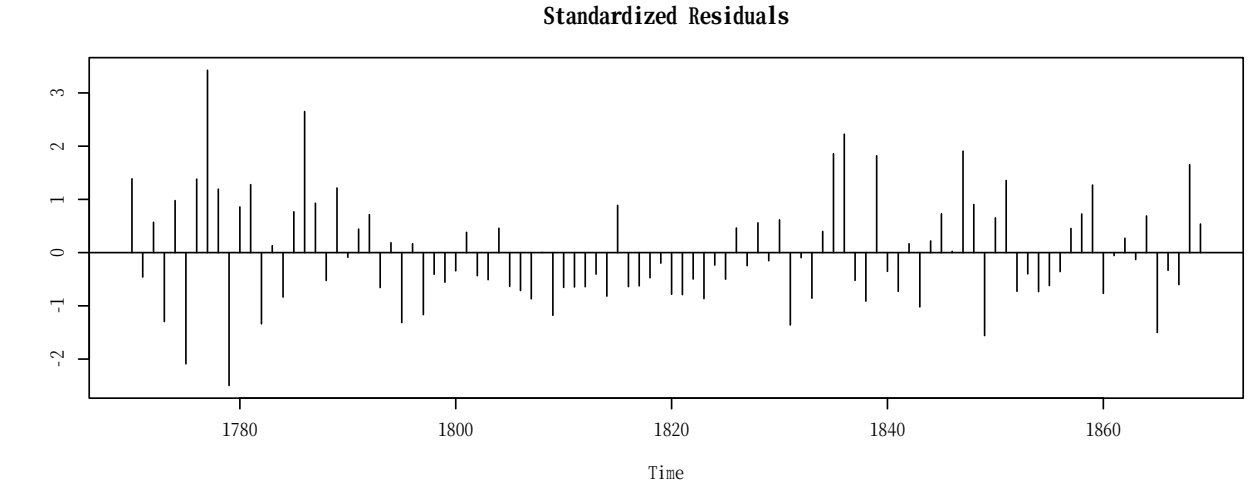
```
## Coefficients:
##          ar1      ar2     ar3  intercept
##       1.5471  -0.9915  0.2004    48.5119
## s.e.  0.0983   0.1545  0.0990     6.0560
##
## sigma^2 estimated as 220.2:  log likelihood = -412.94,  aic = 835.88
```

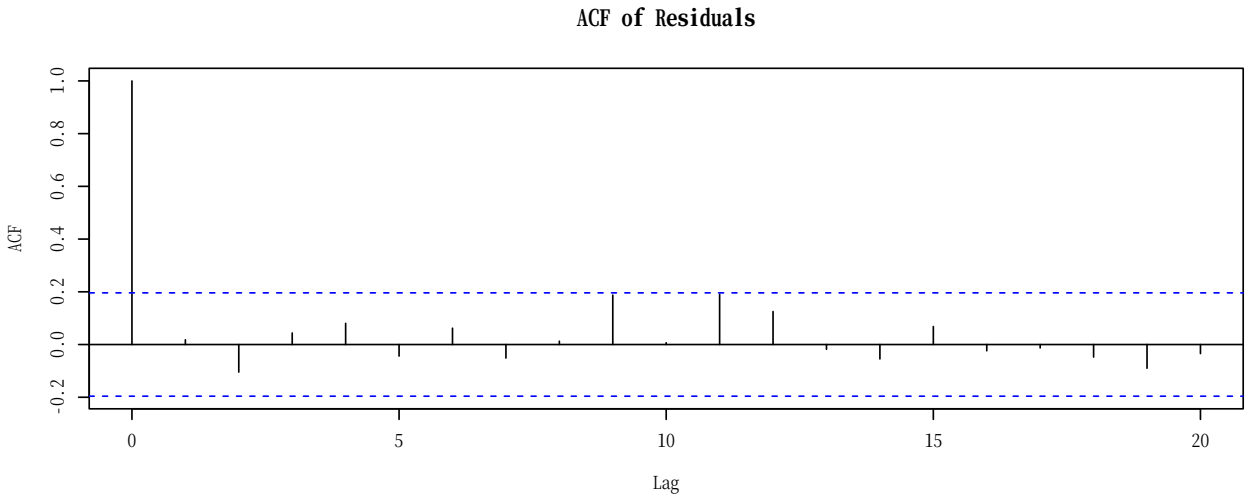
估计的模型为

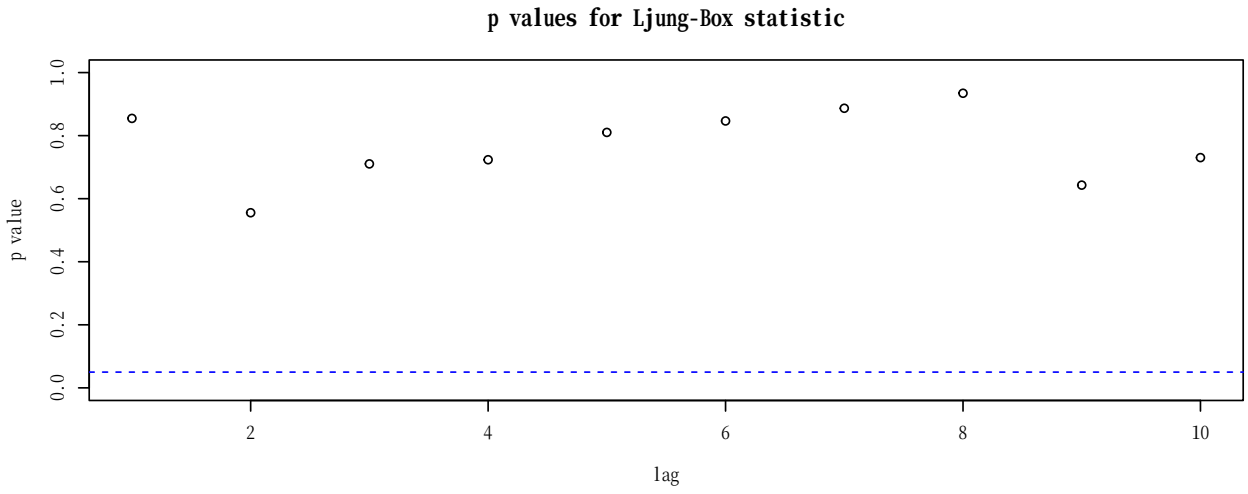
$$(1 - 1.5471B + 0.9915B^2 - 0.2004B^3)(y_t - 48.5119) = e_t,\ e_t \sim \text{WN}(0, 220.2).$$

模型诊断：

```
tsdiag(ss1)
```







从模型诊断看是合适的。

估计的 AR(3) 中特征多项式的根和复根对应的周期：

```
rt1 <- polyroot(c(1, -coef(ss1)[1:3])); rt1
```

```
## [1] 1.045502+0.808659i 1.045502-0.808659i 2.855749-0.000000i
```

```
2*pi/Arg(rt1[1])
```

```
## [1] 9.543839
```

求出的周期 9.5 年左右，与实际观察到的大约 11 年周期不完全相同。

用 statespacer 包将 AR(3) 估计成状态空间模型，此包直接支持恢复 ARIMA 参数：

```
library(datasets)
library(statespacer)
mat.suns <- matrix(window(
  sunspot.year, start = 1770, end = 1869))
fit <- statespacer(
  y = mat.suns,
  H_format = matrix(0),
  # H 是观测方程的误差方差阵
  local_level_ind = TRUE,
  arima_list = list(c(3,0,0)),
  format_level = matrix(0),
  # format_level 是结构时间序列模型中水平值 $\mu_t$ 的方程的误差方差阵
  initial = c(0.5*log(var(mat.suns)), 0, 0, 0),
  verbose=TRUE)
```

```
## Starting the optimisation procedure at: 2023-05-30 17:04:44
```

```
## initial  value 5.022561
## iter  10 value 4.114363
## final  value 4.111163
## converged
```

```
## Finished the optimisation procedure at: 2023-05-30 17:04:44
```

```
## Time difference of 0.07333588600015869 secs
```

构造 ARMA 的结果：

```
arma_coeff <- rbind(
   fit$system_matrices$AR$ARIMA1,
   fit$standard_errors$AR$ARIMA1
)
arma_coeff <- cbind(
   arma_coeff,
   c(fit$smoothed$level[1],
     sqrt(fit$system_matrices$Z_padded$level %*%
          fit$smoothed$V[,,1] %*%
          t(fit$system_matrices$Z_padded$level))
```

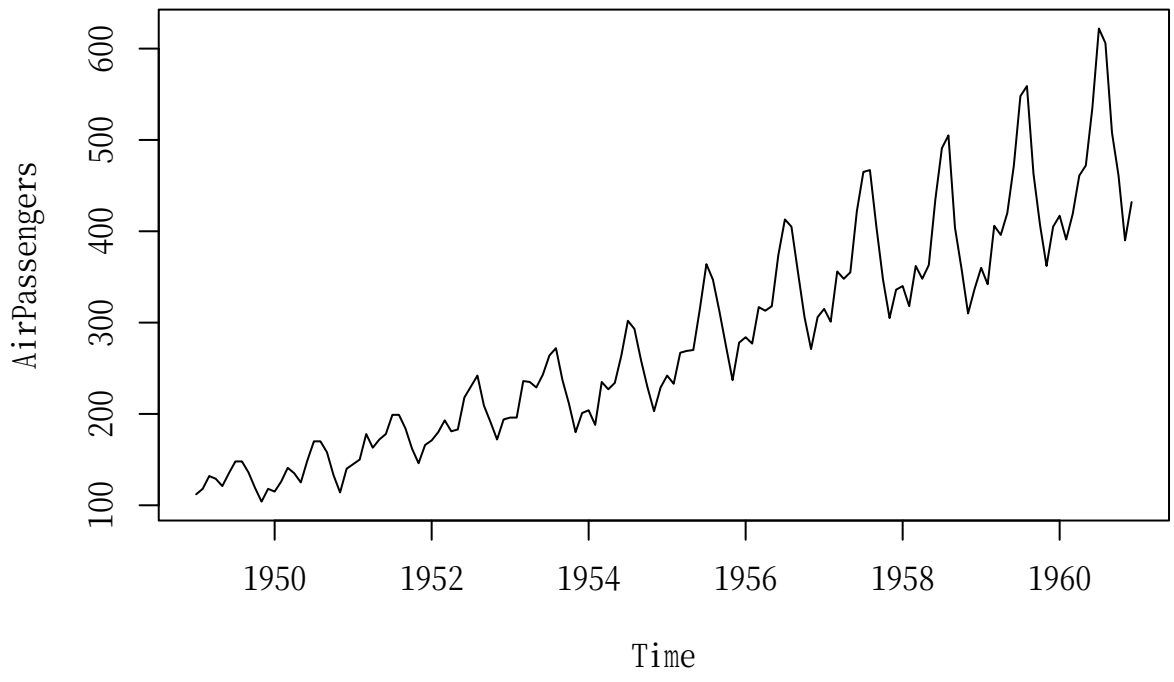
```
   )
)
rownames(arma_coeff) <- c("coefficient", "std_error")
colnames(arma_coeff) <- c("ar1", "ar2", "ar3", "intercept")
arma_coeff
```

```
##                    ar1       ar2       ar3 intercept
## coefficient 1.55976415 -1.005462 0.2129622 48.605905
## std_error   0.09962468  0.155982 0.1003591  6.358039
```

与 arima() 函数结果相近但不完全相同。

例 28.7. 考虑航空乘客数的建模，1949-1961 年的月度数据，用 ARIMA$(0,1,1)(0,1,1)_{12}$ 模型。

```
plot(AirPassengers)
```



将数据取自然对数，用 arima() 建模：

```
ap01 <- arima(
  log(AirPassengers),
  order=c(0,1,1),
  seasonal=list(order=c(0,1,1), frequency=12))
ap01
```

```
##
```

```
## Call:
## arima(x = log(AirPassengers), order = c(0, 1, 1), seasonal = list(order = c(0,
##     1, 1), frequency = 12))
##
## Coefficients:
##           ma1     sma1
##       -0.4018  -0.5569
## s.e.   0.0896   0.0731
##
## sigma^2 estimated as 0.001348:  log likelihood = 244.7,  aic = -483.4
```

模型为：

$$(1-B)(1-B^{12})\ln y_t = (1-0.4018B)(1-0.5569B^{12})e_t,\ e_t \sim \mathrm{WN}(0, 0.001348).$$

利用 statespacer 计算：

```
mat.ap <- matrix(log(AirPassengers))

# 模型设定列表
sarima_list <- list(list(
  s = c(12, 1), ar = c(0, 0), i = c(1, 1), ma = c(1, 1) ))

# 拟合模型
fit <- statespacer(
  y = mat.ap,
  H_format = matrix(0),
  sarima_list = sarima_list,
  initial = c(0.5*log(var(diff(mat.ap))), 0, 0),
  verbose = TRUE)
```

```
## Starting the optimisation procedure at: 2023-05-30 17:04:45
```

```
## initial  value -1.034434
## final  value -1.616321
## converged
```

```
## Finished the optimisation procedure at: 2023-05-30 17:04:45
```

```
## Time difference of 0.451583862304688 secs
```

提取估计的参数：

```
arma_coeff <- rbind(
   c(fit$system_matrices$SMA$SARIMA1$S1,
     fit$system_matrices$SMA$SARIMA1$S12),
   c(fit$standard_errors$SMA$SARIMA1$S1,
     fit$standard_errors$SMA$SARIMA1$S12) )
rownames(arma_coeff) <- c("coefficient", "std_error")
```

```
colnames(arma_coeff) <- c("ma1 s = 1", "ma1 s = 12")
arma_coeff
```

```
##                ma1 s = 1  ma1 s = 12
## coefficient -0.40188859 -0.55694248
## std_error    0.08963614  0.07309788
```

```
goodness_fit <- rbind(
   fit$system_matrices$Q$SARIMA1,
   fit$diagnostics$loglik,
   fit$diagnostics$AIC
)
rownames(goodness_fit) <- c("Variance", "Loglikelihood", "AIC")
goodness_fit
```

```
##                          [,1]
## Variance         0.001347882
## Loglikelihood 232.750284785
## AIC            -3.010420622
```

结果与 `arima()` 结果相同。

28.5 回归模型

28.5.1 常系数回归模型

考虑回归模型

$$y_t = X_t\beta + e_t,\ e_t \sim \text{ iid } \mathrm{N}(0, H_t),\ t = 1, 2, \ldots, n,$$

其中 X_t 为 $1 \times k$ 非随机的已知值，β 是 $k \times 1$ 未知的回归系数向量。可以写成状态空间模型：

$$\begin{aligned} \alpha_t &= \beta, \\ \alpha_{t+1} &= I\alpha_t, \\ y_t &= X_t\alpha_t + e_t. \end{aligned}$$

状态方程没有误差。

这个回归问题的加权最小二乘解为

$$\hat{\beta} = (\sum_{i=1}^{n} X_t^T H_t^{-1} X_t)^{-1} \sum_{i=1}^{n} X_t^T H_t^{-1} y_t.$$

写成状态空间模型后，用滤波算法对 $\alpha_t = \beta$ 进行估计，等价于对 β 进行递推改进估计，相当于递推的最小二乘估计。

MARSS 包的模型直接支持回归，其中的 $C_t c_t$ 是系统方程中的回归，c_t 是回归自变量数据；$D_t d_t$ 是观测方程中的回归，d_t 是回归自变量数据。

对于常系数回归模型，因为回归系数不是时变的，这种模型可以用 MARSS 包的回归功能表示。仅进行回归的 MARSS

模型可以写成

$$
\begin{aligned}
x_0 =& 0, \\
x_t =& 1 \times x_{t-1} + w_t,\ w_t \sim \mathrm{N}(0, 0) \\
y_t =& 1 \times x_t + \beta d_t + v_t,\ v_t \sim \mathrm{N}(0, Q).
\end{aligned}
$$

其中 d_t 是自变量在 t 时刻的观测值，如果有 p 个自变量，则 c_t 为 $p \times 1$ 向量，β 为 $1 \times p$ 向量。

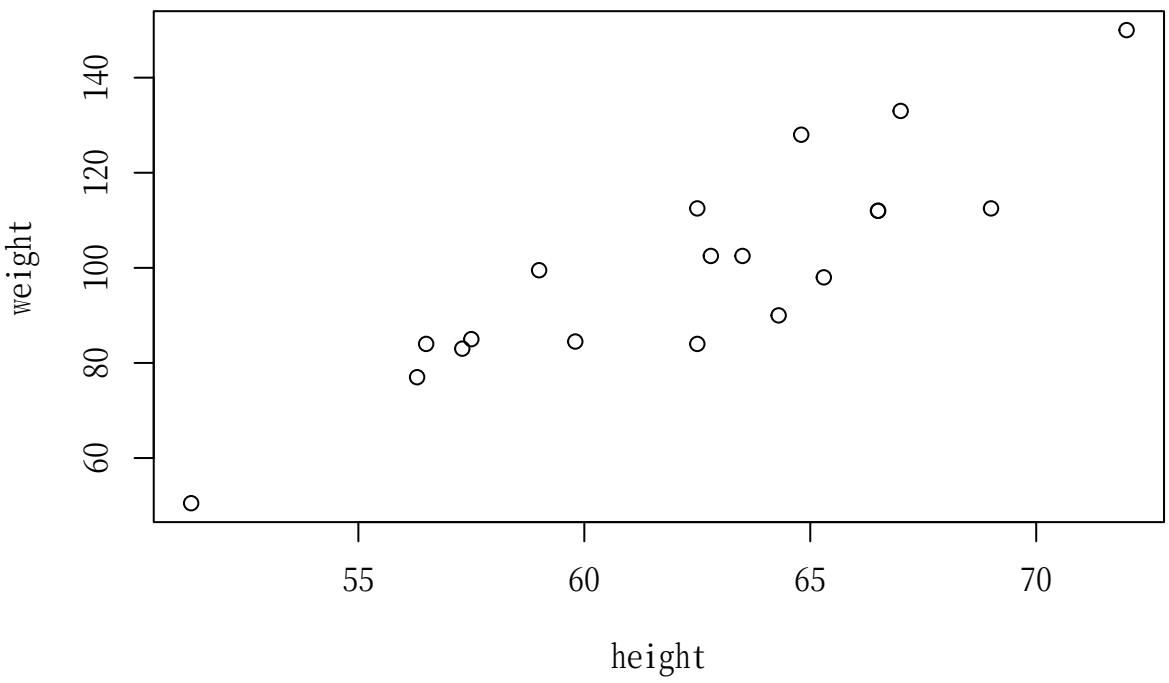
28.5.2 体重对身高的一元回归示例

考虑体重对身高的回归：

```
d_class <- readr::read_csv("class.csv",
  show_col_types = FALSE)
knitr::kable(d_class)
```

name	sex	age	height	weight
Alice	F	13	56.5	84.0
Becka	F	13	65.3	98.0
Gail	F	14	64.3	90.0
Karen	F	12	56.3	77.0
Kathy	F	12	59.8	84.5
Mary	F	15	66.5	112.0
Sandy	F	11	51.3	50.5
Sharon	F	15	62.5	112.5
Tammy	F	14	62.8	102.5
Alfred	M	14	69.0	112.5
Duke	M	14	63.5	102.5
Guido	M	15	67.0	133.0
James	M	12	57.3	83.0
Jeffrey	M	13	62.5	84.0
John	M	12	59.0	99.5
Philip	M	16	72.0	150.0
Robert	M	12	64.8	128.0
Thomas	M	11	57.5	85.0
William	M	15	66.5	112.0

```
with(d_class, plot(height, weight))
```




```
lm(weight ~ height, data=d_class) |>
  summary()
```

```
##
## Call:
## lm(formula = weight ~ height, data = d_class)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.6807  -6.0642   0.5115   9.2846  18.3698
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -143.0269    32.2746  -4.432 0.000366 ***
## height         3.8990     0.5161   7.555 7.89e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.23 on 17 degrees of freedom
## Multiple R-squared:  0.7705, Adjusted R-squared:  0.757
## F-statistic: 57.08 on 1 and 17 DF,  p-value: 7.887e-07
```

用 MARSS 求解：

```
library(MARSS)
fit_class <- MARSS(
  rbind(d_class$weight),
  model = list(
    B = "identity",
    U = "zero",
    Z = "identity",
    A = matrix("a"),
    x0 = "zero",
    Q = "zero",
    R = matrix("R"),
    D = matrix("beta"),
    d = rbind(d_class$height)))
```

```
## Success! algorithm run for 15 iterations. abstol and log-log tests passed.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Algorithm ran 15 (=minit) iterations and convergence was reached.
## Log-likelihood: -71.85003
## AIC: 149.7001   AICc: 151.3001
##
##        Estimate
## A.a      -143.0
## R.R       112.8
## D.beta      3.9
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

与 lm() 估计结果基本相同。

不使用 MARSS 的回归功能，而是将回归写成普通的状态空间形式：

```
library(MARSS)
# 将输入矩阵的每行作为输出的三维数组的一个子矩阵
# 第三维代表时间 $t$
row2d3 <- function(A){
  A <- t(A)
  dim(A) <- c(1, dim(A))
  A
```

```
}
fit_class2 <- MARSS(
  rbind(d_class$weight), model = list(
  B = "identity",
  U = "zero",
  Q = "zero",
  Z = row2d3(cbind(1, d_class$height)),
  A = "zero",
  x0 = cbind(c("beta0", "beta1")),
  R = matrix("R")),
  inits = list(x0=cbind(
    c(mean(d_class$weight),0))))
```

```
## Success! algorithm run for 15 iterations. abstol and log-log tests passed.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Algorithm ran 15 (=minit) iterations and convergence was reached.
## Log-likelihood: -71.85003
## AIC: 149.7001   AICc: 151.3001
##
##          Estimate
## R.R         112.8
## x0.beta0   -143.0
## x0.beta1      3.9
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

28.5.3 变系数回归模型

考虑变系数的回归模型

$$y_t = X_t\beta_t + e_t,\ e_t \sim \text{ iid } \mathrm{N}(0, H_t),\ t = 1, 2, \ldots, n,$$

其中的随时间而变化的回归系数可以用如下的随机游动建模：

$$\beta_{t+1} = \beta_t + \eta_t,$$

这样，以 β_t 为状态向量，就可以将变系数回归模型写成简单的状态空间形式：

$$\begin{aligned} y_t &= X_t\beta_t + e_t, \\ \beta_{t+1} &= I\beta_t + I\eta_t. \end{aligned}$$

这样的模型可以用标准的状态空间模型工具进行估计和推断。

变系数回归模型不能用 MARSS 包的回归功能表示，如果使用 MARSS 包，也需要将模型写成上面的形式，将时变的回归系数作为状态向量的一部分。

28.5.4 大鳞大马哈鱼生存例子

例子来自 MARSS 手册。研究大鳞大马哈鱼生存状况受到海洋上升洋流强度的影响。海洋上升洋流带来丰富营养，使得近海浮游生物繁盛，有利于大鳞大马哈鱼幼鱼生长。用变系数回归模型：

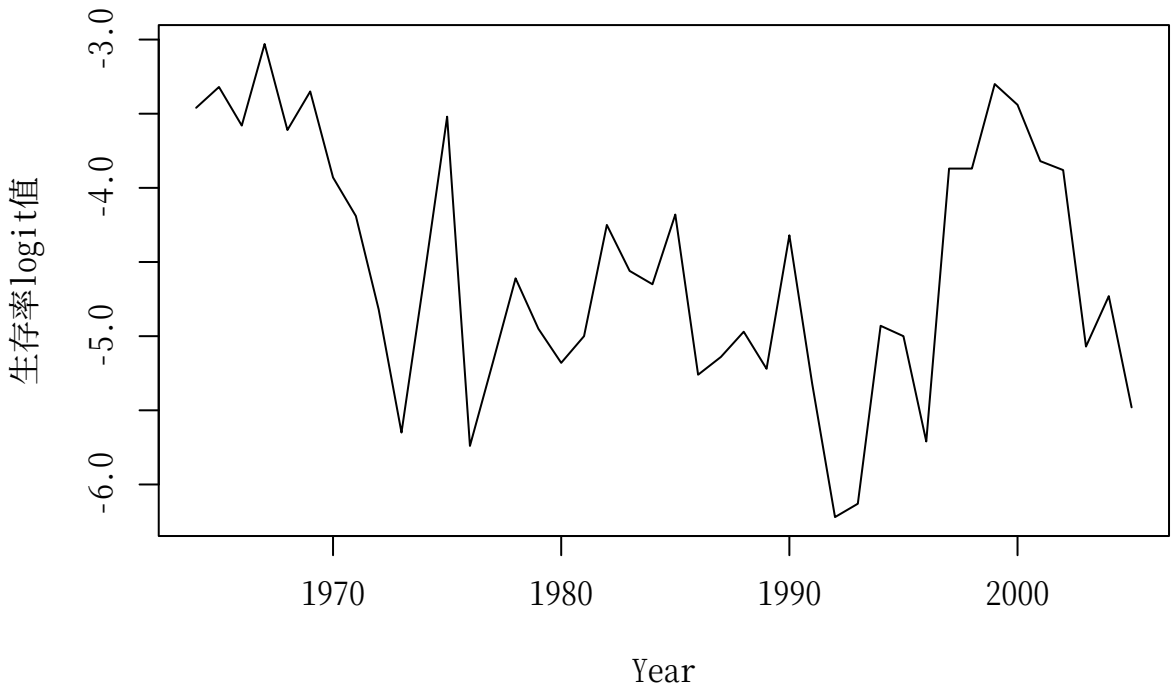
$$\text{logit}(\text{生存率})_t = \alpha_t + \beta_t \times \text{洋流强度}_t + \nu_t,\ \nu_t \sim \text{N}(0, r).$$

回归系数模型为

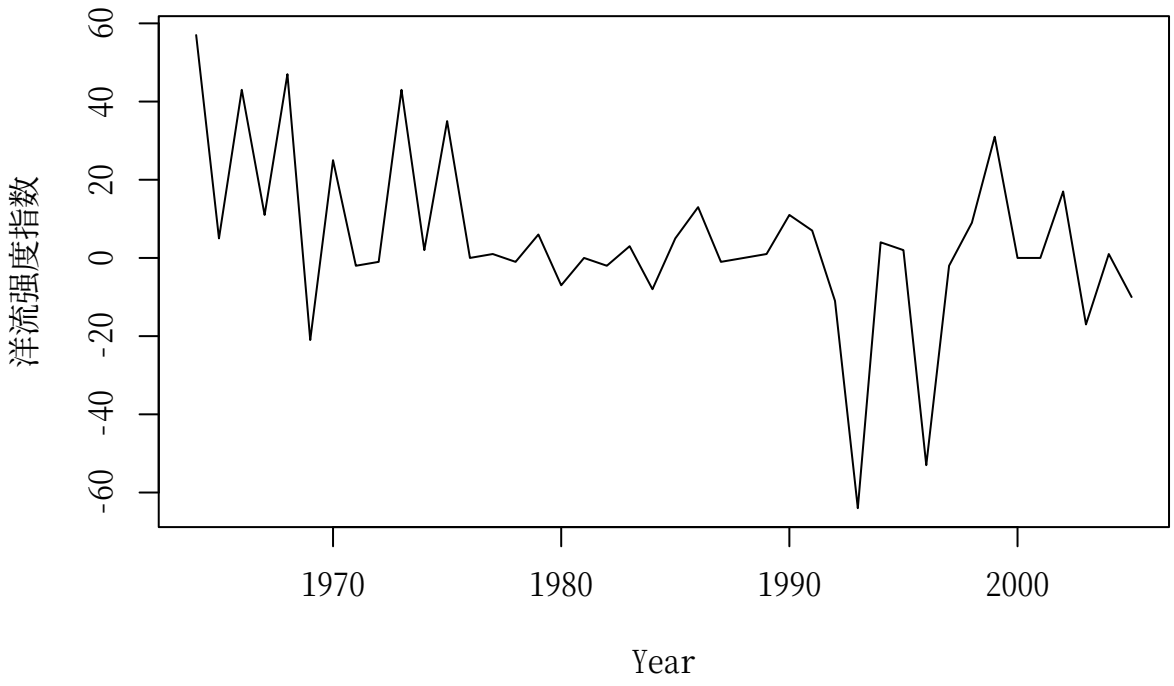
$$\begin{pmatrix}\alpha_t \\ \beta_t\end{pmatrix} = \begin{pmatrix}1 & 0 \\ 0 & 1\end{pmatrix}\begin{pmatrix}\alpha_{t-1} \\ \beta_{t-1}\end{pmatrix} + w_t,\ w_t \sim \text{N}(0, \text{diag}(q_1, q_2)).$$

28.5.4.1 数据

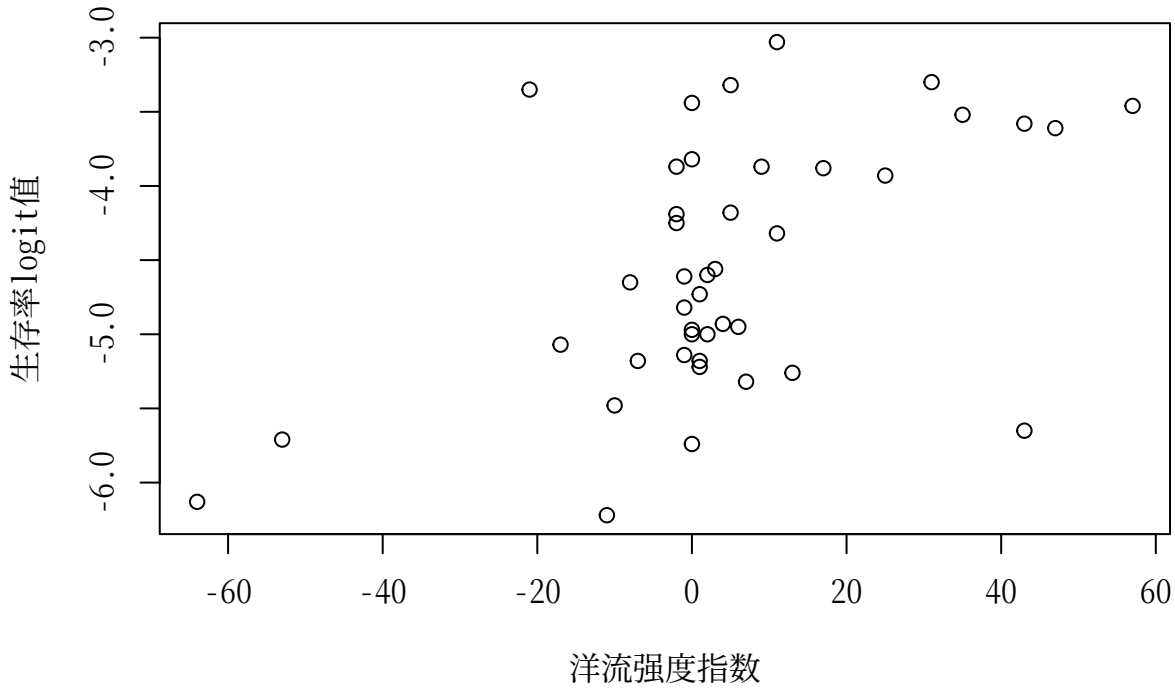
```
library(MARSS)
data(SalmonSurvCUI)
with(SalmonSurvCUI, plot(
  year, logit.s, type="l",
  xlab="Year", ylab=" 生存率 logit 值"))
```



```
with(SalmonSurvCUI, plot(
  year, CUI.apr, type="l",
  xlab="Year", ylab=" 洋流强度指数"))
```




```
with(SalmonSurvCUI, plot(
  CUI.apr, logit.s, type="p",
  xlab=" 洋流强度指数", ylab=" 生存率 logit 值"))
```



28.5.4.2 变系数回归建模

```
# 将输入矩阵的每行作为输出的三维数组的一个子矩阵
# 第三维代表时间 $t$
# 适用于一元因变量的多元回归自变量转换为 Z 矩阵
row2d3 <- function(A){
  A <- t(A)
  dim(A) <- c(1, dim(A))
  A
}

dat <- rbind(SalmonSurvCUI$logit.s)     # 观测变量，回归因变量
CUI.z <- zscore(SalmonSurvCUI$CUI.apr) # 回归自变量，标准化
sal_mod <- list(
  B = "identity",
  U = "zero",
  Q = "diagonal and unequal",
  Z = row2d3(cbind(1, CUI.z)),
  A = "zero",
  R = matrix("r") )
sal_fit1 <- MARSS(
```

```
  dat, model = sal_mod,
  inits = list(x0 = cbind(c(0,0))) )
```

```
## Success! abstol and log-log tests passed at 115 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 115 iterations.
## Log-likelihood: -40.03813
## AIC: 90.07627   AICc: 91.74293
##
##           Estimate
## R.r        0.15708
## Q.(X1,X1)  0.11264
## Q.(X2,X2)  0.00564
## x0.X1     -3.34023
## x0.X2     -0.05388
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

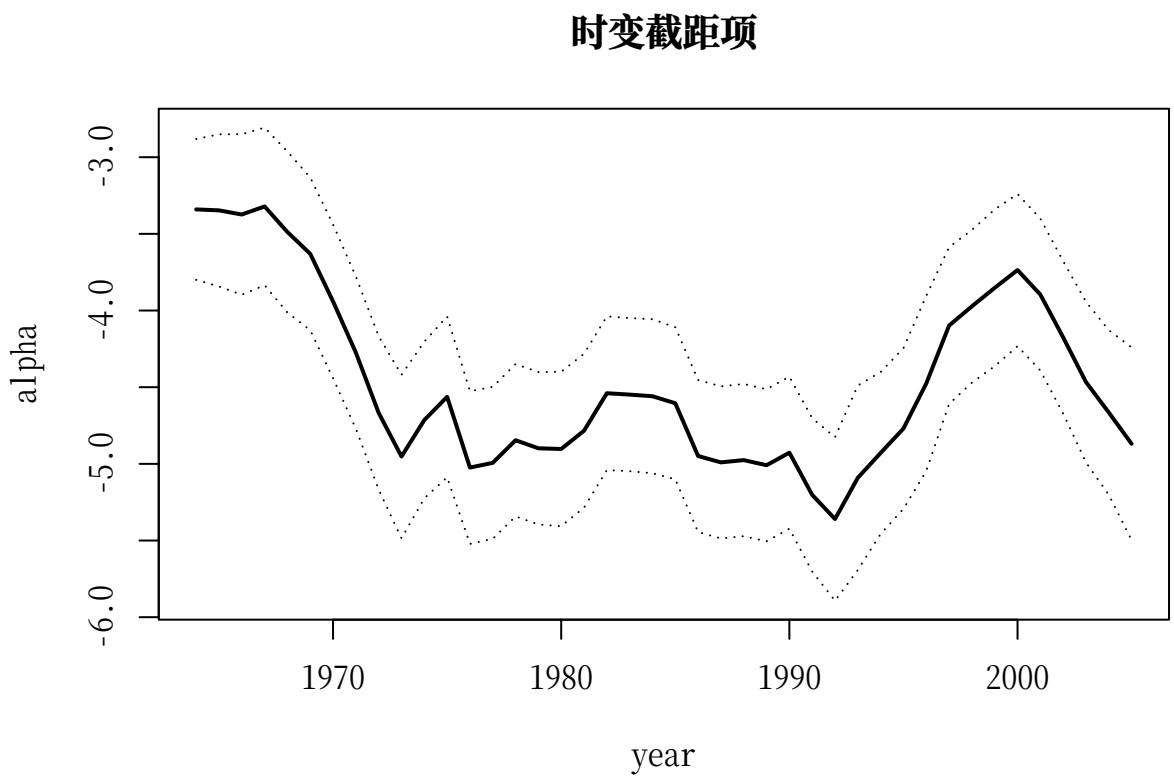
估计结果为：

$$
\begin{aligned}
\text{logit}(\text{生存率})_t =& \alpha_t + \beta_t \times \text{洋流强度}_t + \nu_t,\ \nu_t \sim \text{N}(0, 0.16);\\
\alpha_t =& \alpha_{t-1} + w_{1t},\ w_{1t} \sim \text{N}(0, 0.11),\\
\beta_t =& \beta_{t-1} + w_{2t},\ w_{2t} \sim \text{N}(0, 0.0056).
\end{aligned}
$$

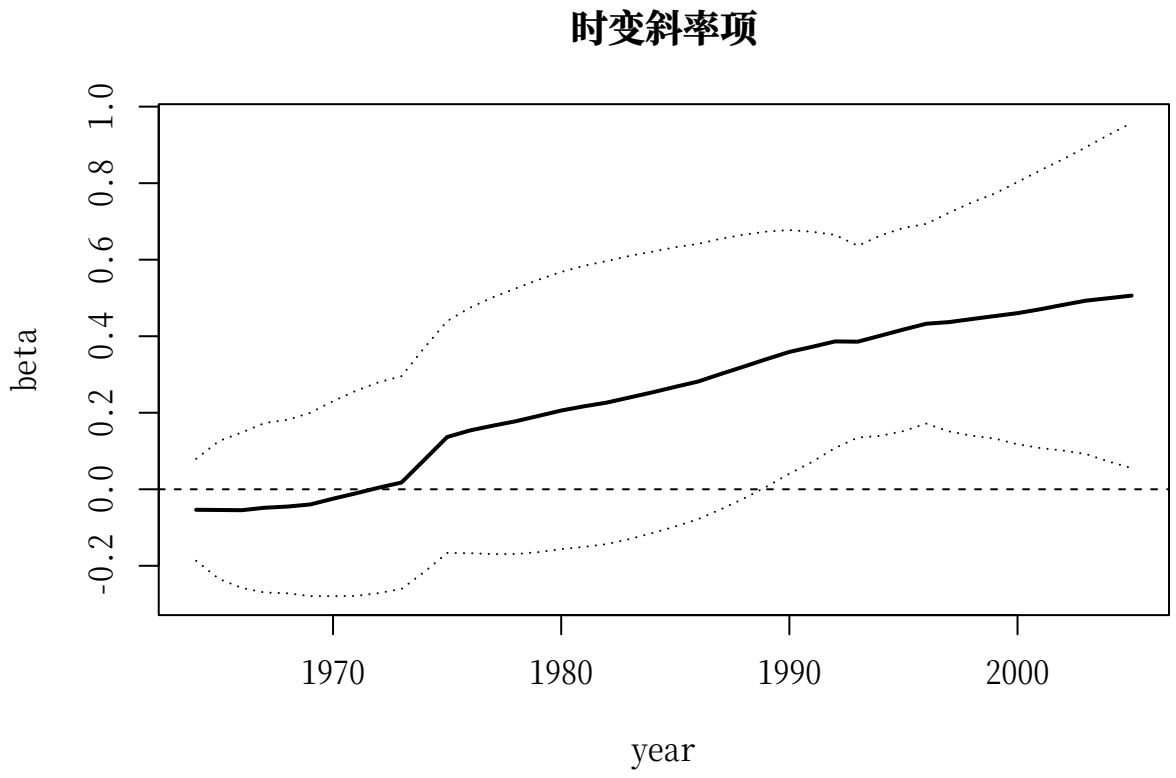
提取时变的回归系数：

```
sal_sm <- tsSmooth(sal_fit1, type="xtT") |>
  pivot_wider(
    id_cols = c("t"),
    names_from = ".rownames",
    values_from = c(".estimate", ".se") ) |>
  mutate(t = t-1 + 1964) |>
  setNames(c("year", "alpha", "beta", "se_alpha", "se_beta"))
with(sal_sm, {
  ran <- range(c(alpha - 2*se_alpha, alpha + 2*se_alpha))
  plot(year, alpha, type="l", lwd=2, ylim = ran,
       main=" 时变截距项")
```

```
  lines(year, alpha - 2*se_alpha, lty=3)
  lines(year, alpha + 2*se_alpha, lty=3)
  })
```




```
with(sal_sm, {
  ran <- range(c(beta - 2*se_beta, beta + 2*se_beta))
  plot(year, beta, type="l", lwd=2, ylim = ran,
       main=" 时变斜率项")
  lines(year, beta - 2*se_beta, lty=3)
  lines(year, beta + 2*se_beta, lty=3)
  abline(h=0, lty=2)
  })
```



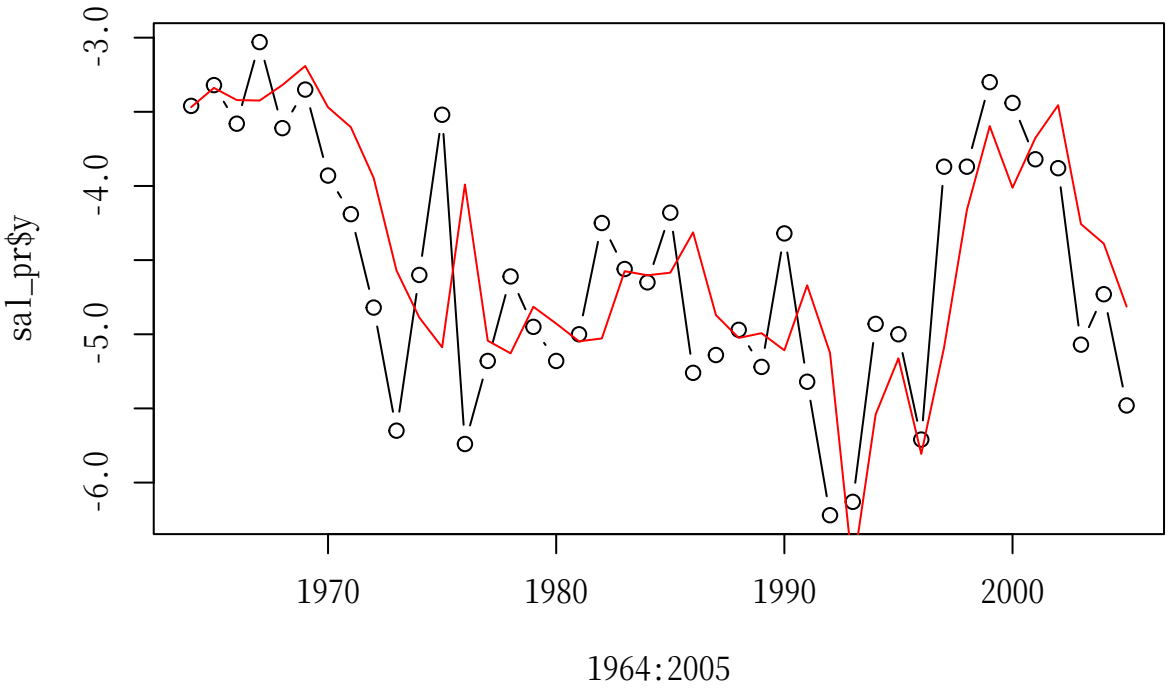
28.5.4.3 预测

应该基于超参数的最大似然估计和 $1:(t-1)$ 的观测值来预测时变的回归系数：

```
sal_pr_co <- tsSmooth(sal_fit1, type = "xtt1")
```

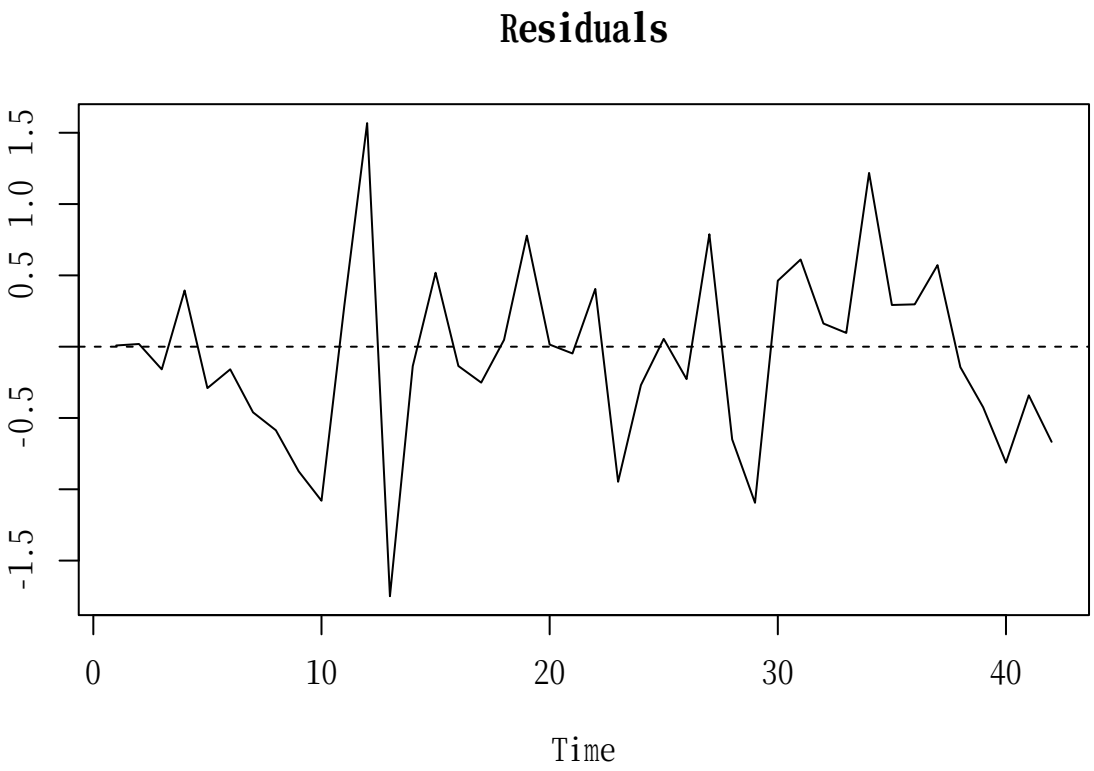
因变量的预测值，利用 $1:(t-1)$ 的信息预测时变回归系数，用 t 时刻回归系数预测生存率：

```
sal_pr <- predict(sal_fit1, type="ytt1")$pred
plot(1964:2005, sal_pr$y, type="b")
lines(1964:2005, sal_pr$estimate, col="red")
```



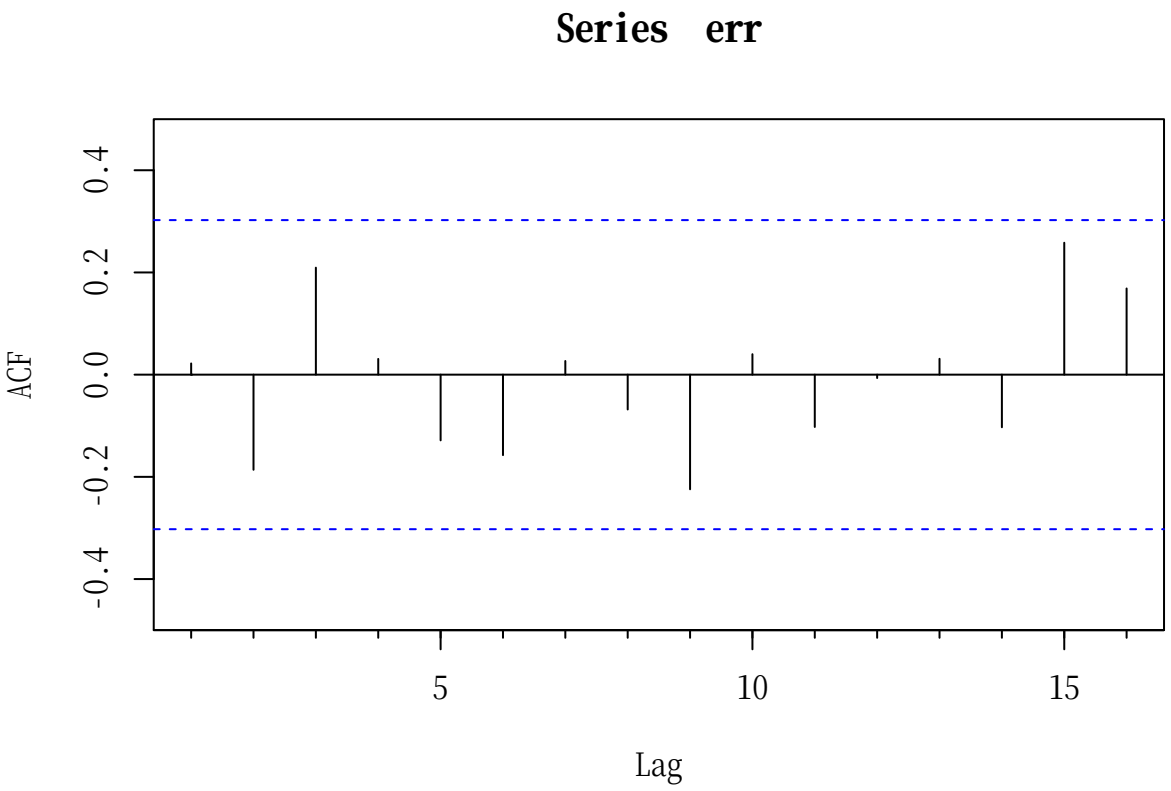
残差诊断：

```
err <- with(sal_pr, y - estimate)
ts.plot(err, main="Residuals", ylab="")
abline(h=0, lty=2)
```



残差的 ACF:

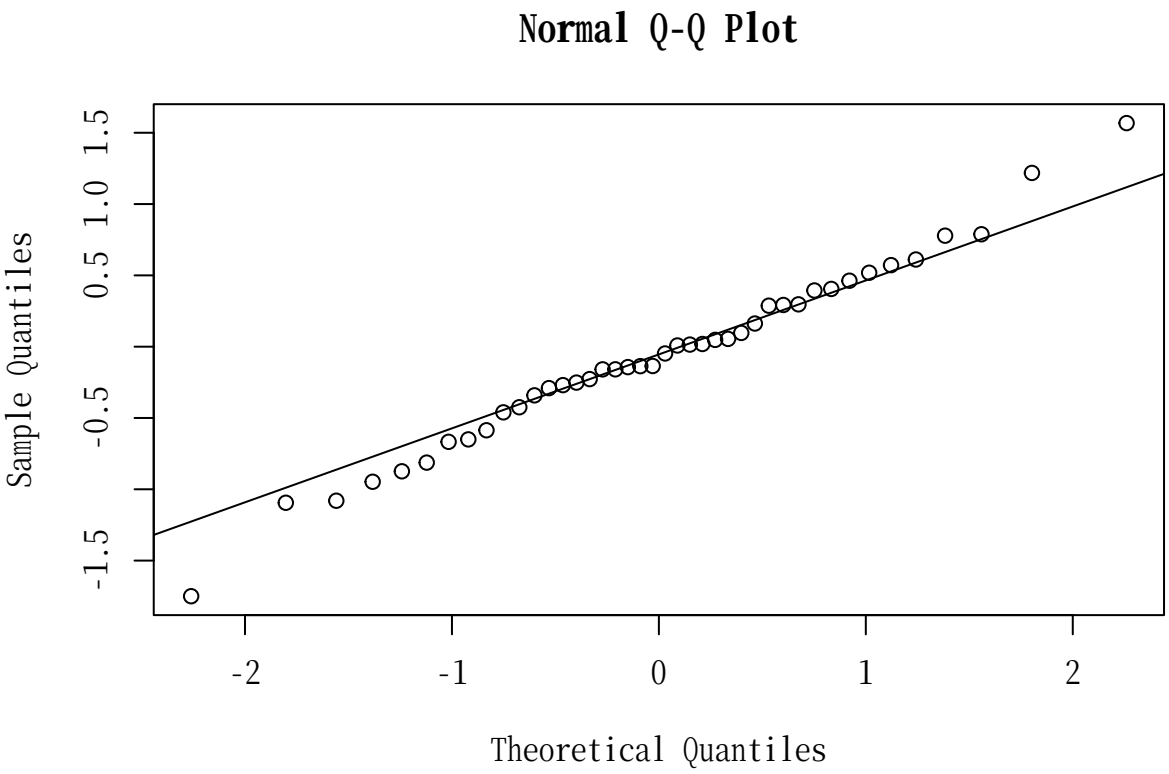
```
forecast::Acf(err)
```



残差没有序列自相关。

残差的正态 QQ 图：

```
qqnorm(err); qqline(err)
```



不偏离正态。

28.5.5 时变系数 CAPM 模型

28.5.5.1 模型

考虑时变系数的资产定价模型 (CAPM)：

$$\begin{aligned} r_t =& \beta_{0t} + \beta_{1t} r_{M,t} + e_t,\ e_t \sim \text{ iid } \mathrm{N}(0, \sigma_e^2), \\ \beta_{0,t+1} =& \beta_{0,t} + u_t,\ u_t \sim \text{ iid } \mathrm{N}(0, \sigma_u^2), \\ \beta_{1,t+1} =& \beta_{1,t} + v_t,\ v_t \sim \text{ iid } \mathrm{N}(0, \sigma_v^2), \end{aligned}$$

其中 $\{e_t\}, \{u_t\}, \{v_t\}$ 相互独立，r_t 是某金融资产的超额收益率，$r_{M,t}$ 是市场的超额收益率。可以写成状态空间模型

$$\begin{aligned} r_t =& (1, r_{M,t}) \begin{pmatrix} \beta_{0t} \\ \beta_{1t} \end{pmatrix} + e_t, \\ \begin{pmatrix} \beta_{0,t+1} \\ \beta_{1,t+1} \end{pmatrix} =& \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \beta_{0,t} \\ \beta_{1,t} \end{pmatrix} + \begin{pmatrix} u_t \\ v_t \end{pmatrix}. \end{aligned}$$

令状态变量 $\alpha_t = (\beta_{0t}, \beta_{1t})^T$, 观测变量为 r_t，且

$$\begin{aligned} Z_t =& (1, r_{M,t}), \quad H_t = \sigma_e^2, \\ T_t =& I_2, \quad R_t = I, \quad \eta_t = (u_t, v_t)^T, \\ Q_t =& \begin{pmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{pmatrix}, \end{aligned}$$

则时变系数 CAPM 对应于状态空间模型

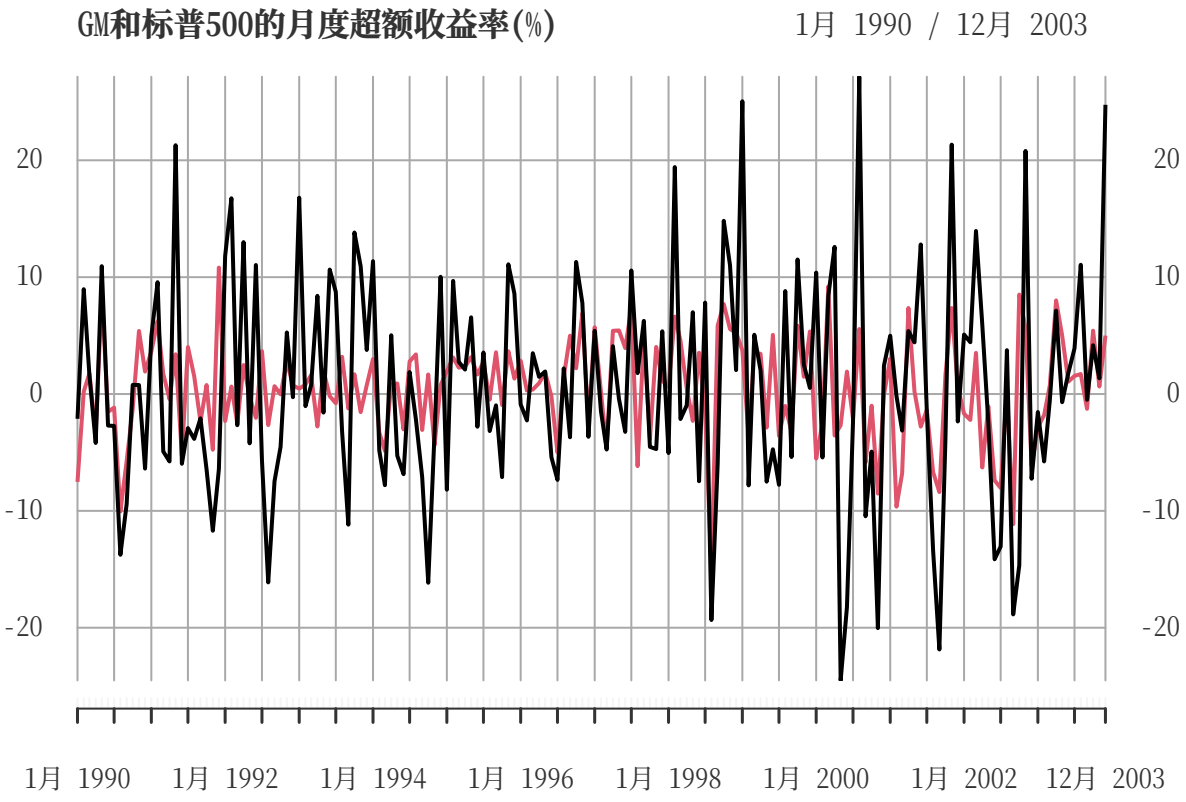
$$\begin{aligned} r_t =& Z_t\alpha_t + e_t,\ e_t \sim \mathrm{N}(0, H_t), \\ \alpha_{t+1} =& T_t\alpha_t + R_t\eta_t,\ \eta_t \sim \mathrm{N}(0, Q_t). \end{aligned}$$

28.5.5.2 通用动力的例子

例 28.8. 考虑通用动力 (GM) 股票的月度超额收益率 r_t 的 CAPM 模型，时间为 1990 年 1 月到 2003 年 12 月，以标普 500 超额收益率为市场超额收益率 $r_{M,t}$。

数据（收益率单位：百分之一）：

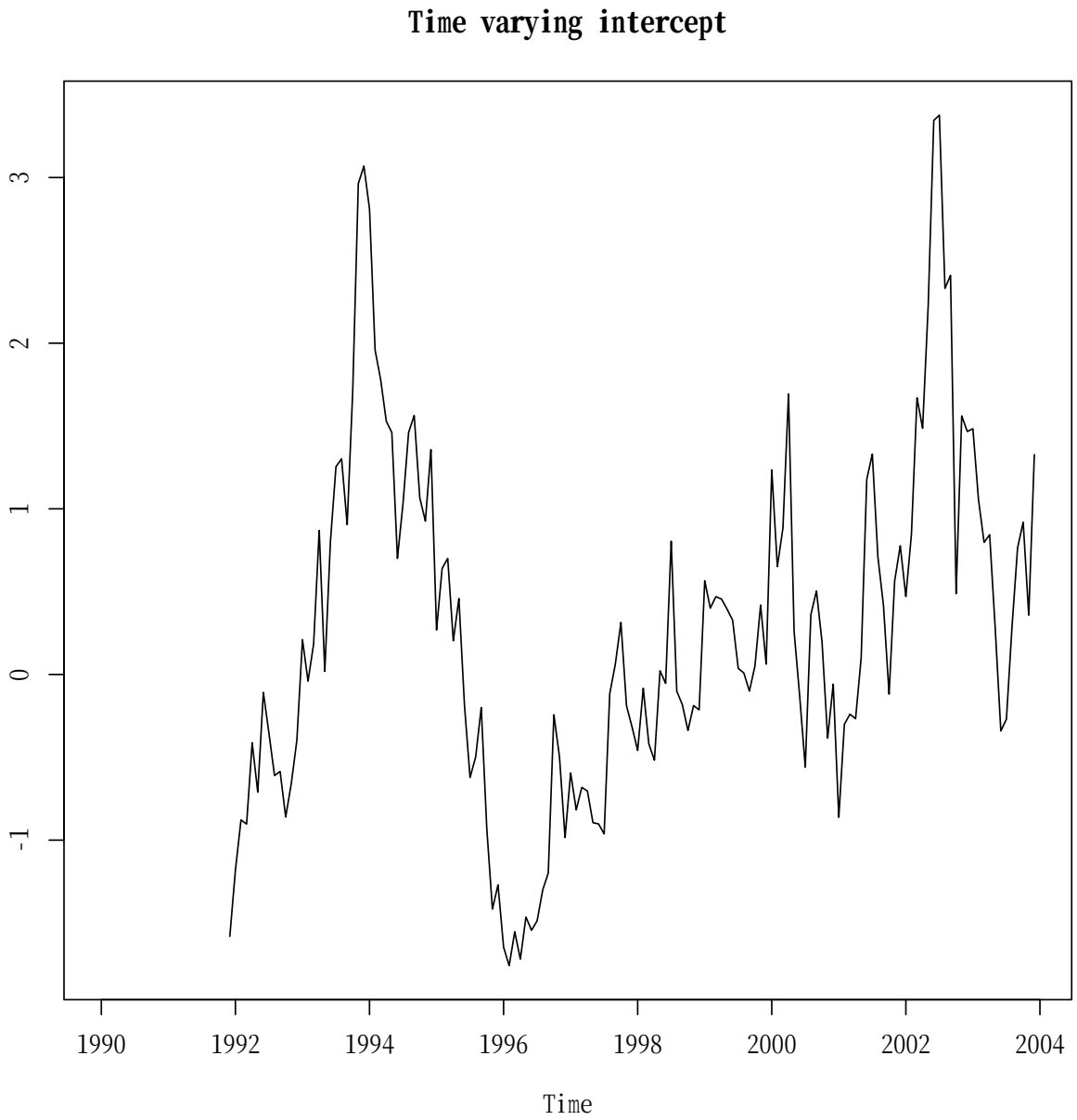
```
da_gm <- readr::read_table(
  "m-fac9003.txt",
  col_types = cols(.default=col_double()))[,c("GM", "SP5")]
nda <- nrow(da_gm)
ts.gm <- ts(as.matrix(da_gm),
  start=c(1990, 1), frequency=12)
plot(as.xts(ts.gm),
  main="GM 和标普 500 的月度超额收益率 (%)")
```



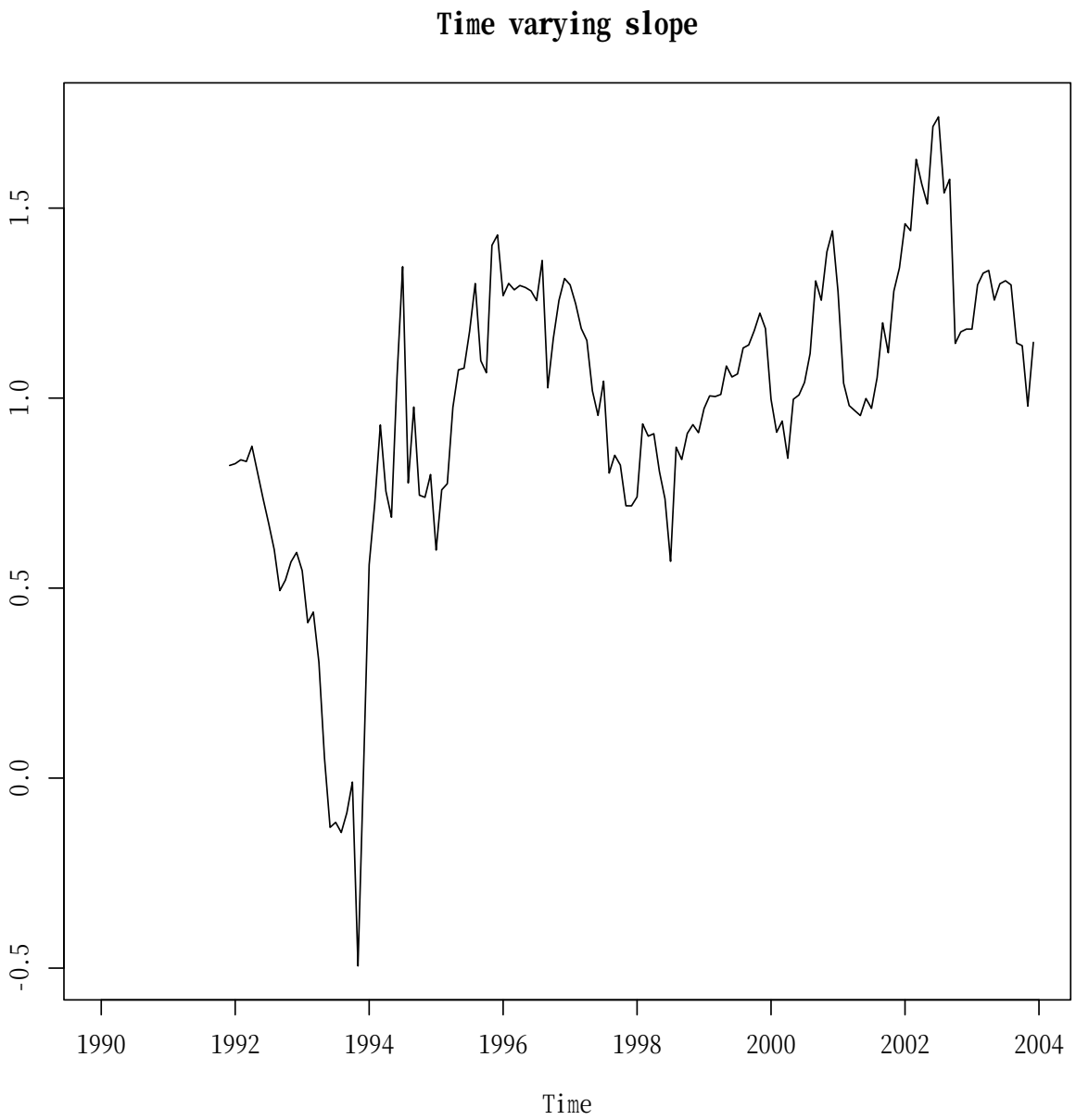
28.5.5.3 滚动计算回归的简单做法

尝试滚动地进行线性回归，取固定的拟合时间窗口为 24 个月，获得动态的系数估计：

```
wh <- 24 # 滚动窗口
n <- nrow(da_gm)
betamat <- matrix(NA, n, 2)
for(t in wh:n){
  tind <- (t-wh+1):t
  betamat[t,] <-
    lm(GM ~ SP5, data=da_gm[tind,]) |>
    coef()
}
plot(ts(
  betamat[,1],
  start=c(1990, 1),
  frequency=12),
  main="Time varying intercept",
  ylab="")
```




```
plot(ts(
  betamat[,2],
  start=c(1990, 1),
  frequency=12),
  main="Time varying slope",
  ylab="")
```



这个结果可以作为基准。

28.5.5.4 使用 KFAS 包拟合时变系数回归模型

statespacer 包没有提供时变系数回归模型，而 Z_t 依赖于 t 时 statespacer 包没有提供自定义方法。

使用 KFAS 包。这个包提供了 `SSMregression()` 函数，可以指定带有随机游动斜率的回归模型，但不支持随机截距项，尝试使用自定义的对应于截距项的自变量会使得滤波平滑失败。

我们使用 `SScustom()` 直接指定各个矩阵。KFAS 对时变的 Z_t 等模型矩阵，允许直接输入一个三维数组 `Z`，`Z[,,t]` 的值为 Z_t。

Z_t, T_t, H_t, Q_t 等矩阵如果输入为三维矩阵，就是时变的；如果输入为二维矩阵，就是非时变的，也可以输入为标量，就相当于非时变的 1×1 矩阵。

```
library(KFAS)

# 将每一行是一个 $Z_t$ 值的矩阵，
# 转换成 $1 \times m \times n$ 的数组，
# $m$ 是状态空间维数，$n$ 是样本量
dim2to3 <- function(M){
  M <- t(M)
  dim(M) <- c(1, dim(M))
  M
}

# Zt: 用三维数组表示的 $Z_t, t=1,2,\dots,n$,
# 第三下标代表 $t$,
# Zt[,,t] 是 $1 \times m$ 矩阵 $Z_t$
Zt <- dim2to3(cbind(1, da_gm[["SP5"]]))

mod_gm <- SSModel(
  da_gm$GM ~ SSMcustom(
    Z = Zt, # 观测方程的载荷阵
    T = diag(2), # 状态方程的转移矩阵
    # 状态的初始值
    a1 = c(0, 1),
    # Q 是状态方程扰动项方差阵
    Q = diag(NA, 2)),
  # H 是观测方程误差方差阵
  H = NA)
fit_gm <- fitSSM(
  mod_gm,
  inits = c(0,0,0),
  method="BFGS") #"Nelder-Mead")
fit_gm$optim.out$value
```

```
## [1] 589.0349
```

估计的随机截距和随机斜率的扰动标准差：

```
sqrt(diag(fit_gm$model$Q[,,1]))
```

```
## [1] 0.0043620203 0.0006551979
```

这两个标准差接近于 0，说明估计的时变截距项和时变斜率项近似等于常数值。

估计的回归误差项标准差：

```
sqrt(fit_gm$model$H[,,1])
```

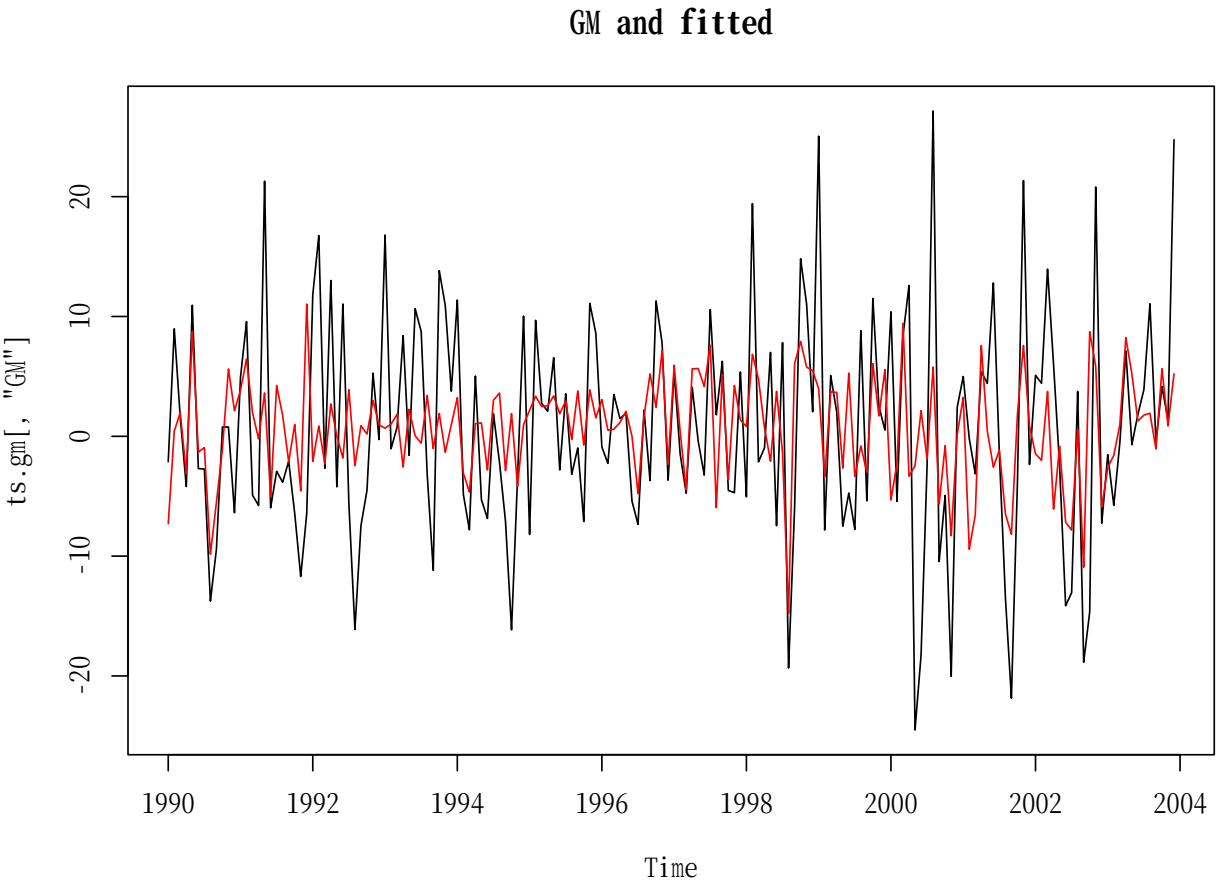
```
## [1] 8.108269
```

计算平滑结果：

```
res_gm <- KFS(fit_gm$model)
```

GM 原始值与拟合值（使用平滑）的曲线：

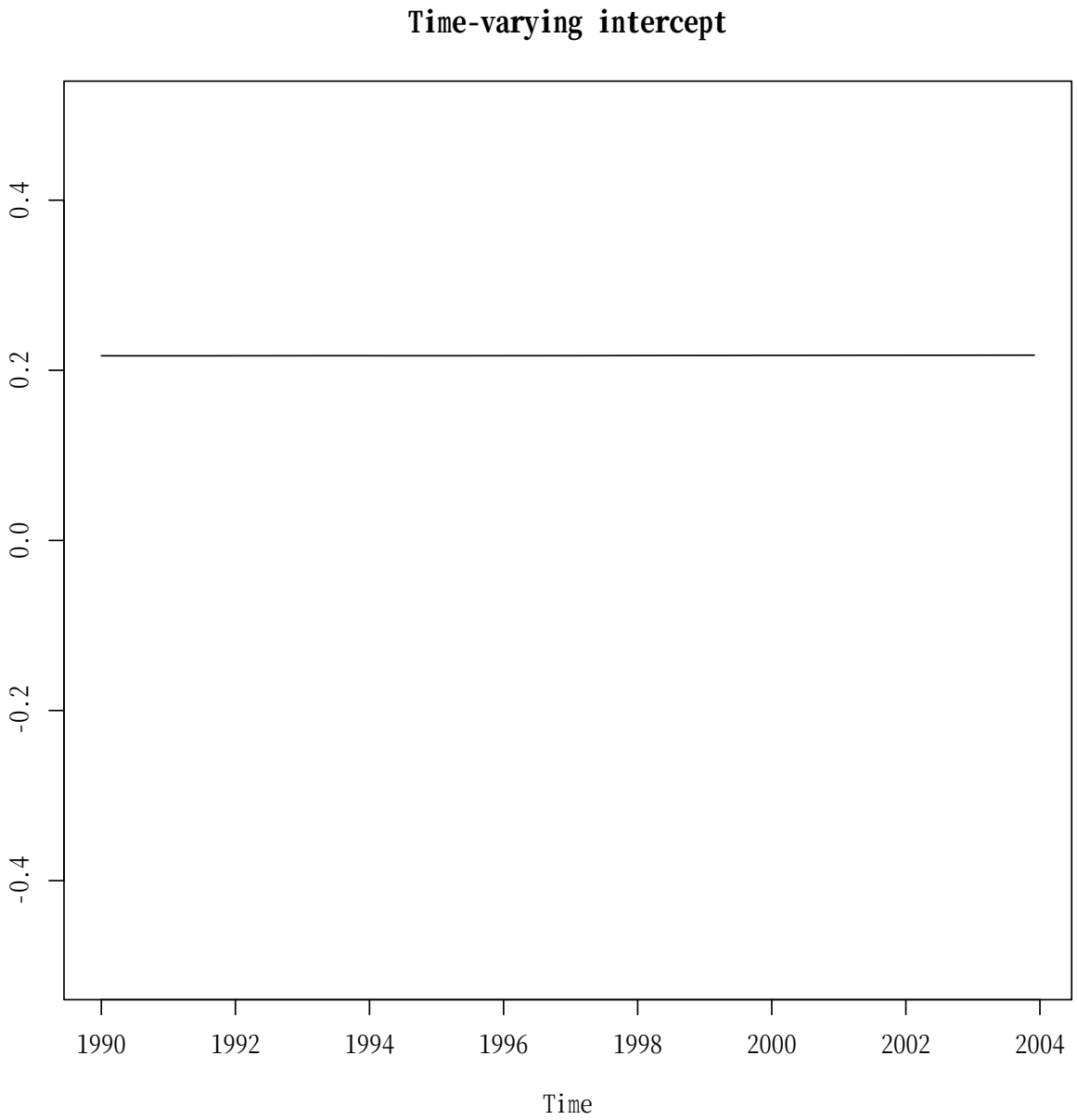
```
plot(ts.gm[,"GM"], main="GM and fitted")
lines(as.vector(time(ts.gm)),
  c(fitted(res_gm, filtered=FALSE)),
  col="red")
```



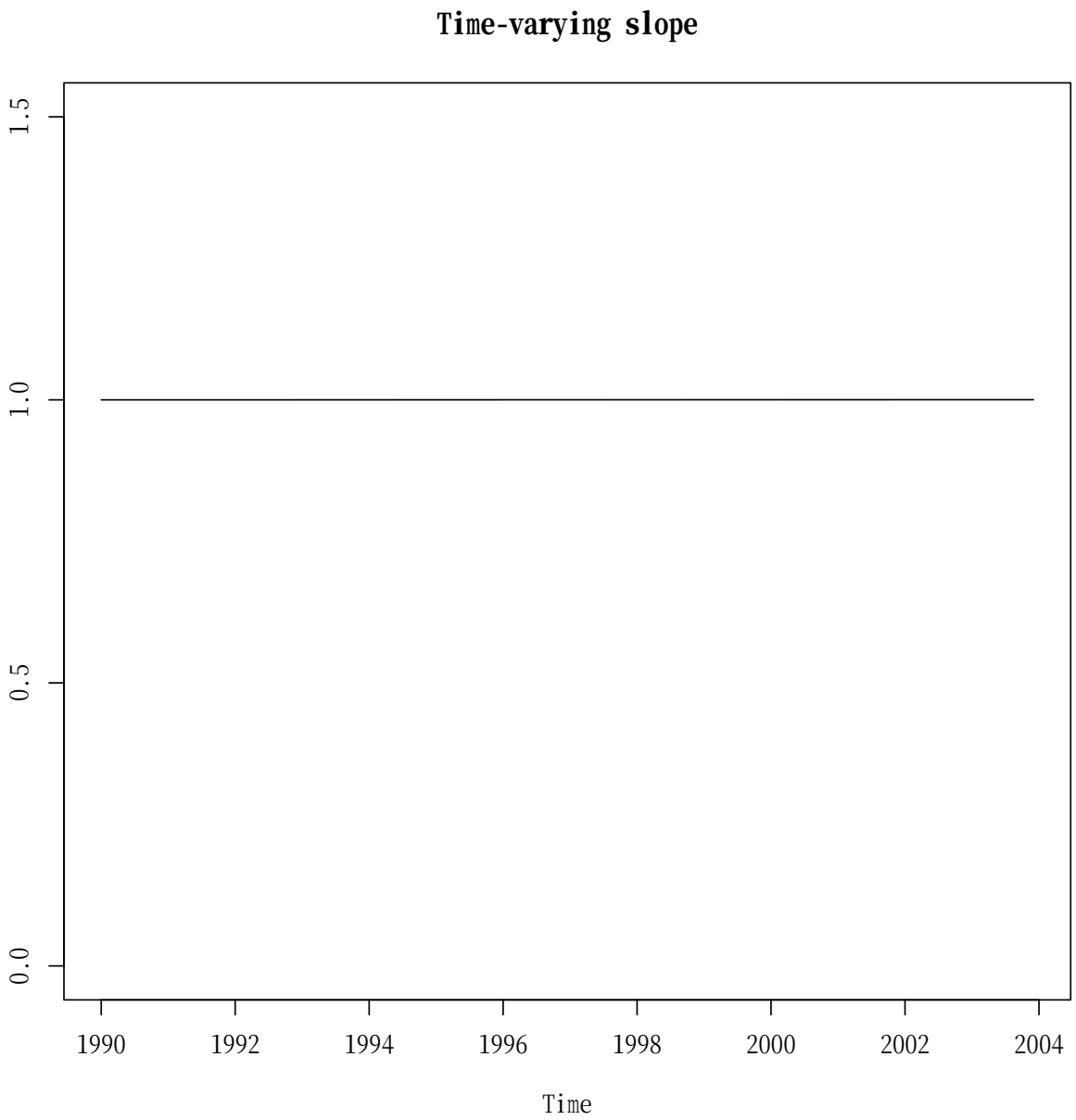
如果前面的程序中没有指定 `a1 = c(0, 1)`，即初始的随机截距、随机斜率，则拟合上图在左端的拟合效果很差。

估计的时变截距项和时变斜率项：

```
states <- as.matrix(coef(res_gm))
states[,2] <- states[,1] + states[,2]
states <- states[,2:3]
plot(ts(states[,1], start=c(1990, 1), frequency=12),
  main = "Time-varying intercept",
  ylab = "",
  ylim = c(-0.5, 0.5))
```




```
plot(ts(states[,2], start=c(1990, 1), frequency=12),
  main = "Time-varying slope",
  ylab="",
  ylim = c(0, 1.5))
```



可以看出，时变的截距项 β_{0t} 和时变斜率项 β_{1t} 基本上取常数值，与滚动回归得到的结果不一致，没有起到时变的效果。

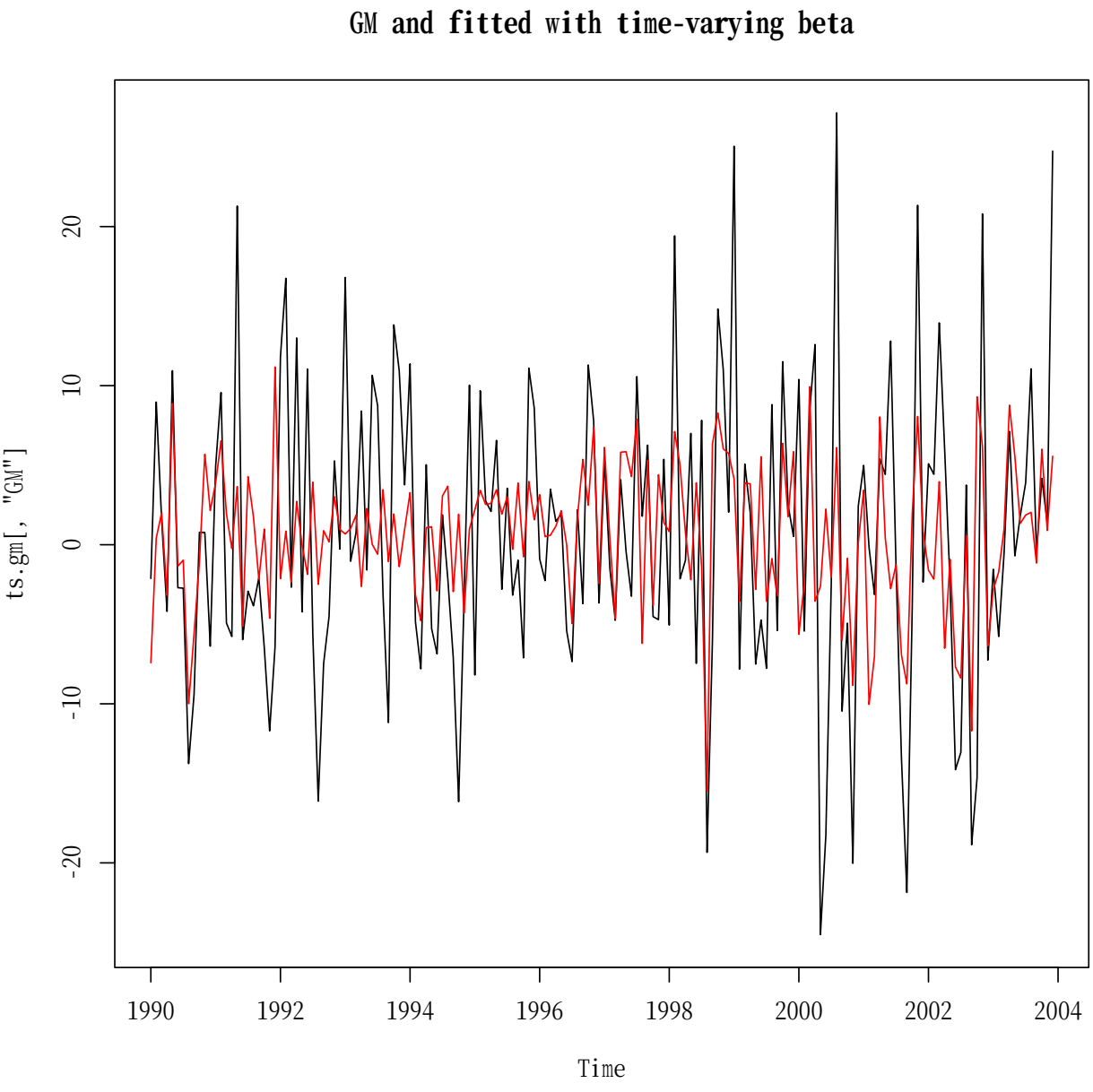
使用 KFAS 的 `SSMregression()`，仅使用随机斜率项：

```
mod_gm <- SSModel(
  da_gm$GM ~ SSMregression(
    ~ da_gm$SP5,
    # Q 是状态方程扰动项方差阵
    Q = NA),
  # H 是观测方程误差方差阵
  H = NA)
fit_gm <- fitSSM(
```

```
  mod_gm,
  inits = c(0, 0),
  method="BFGS")
res_gm <- KFS(fit_gm$model)
```

拟合结果：

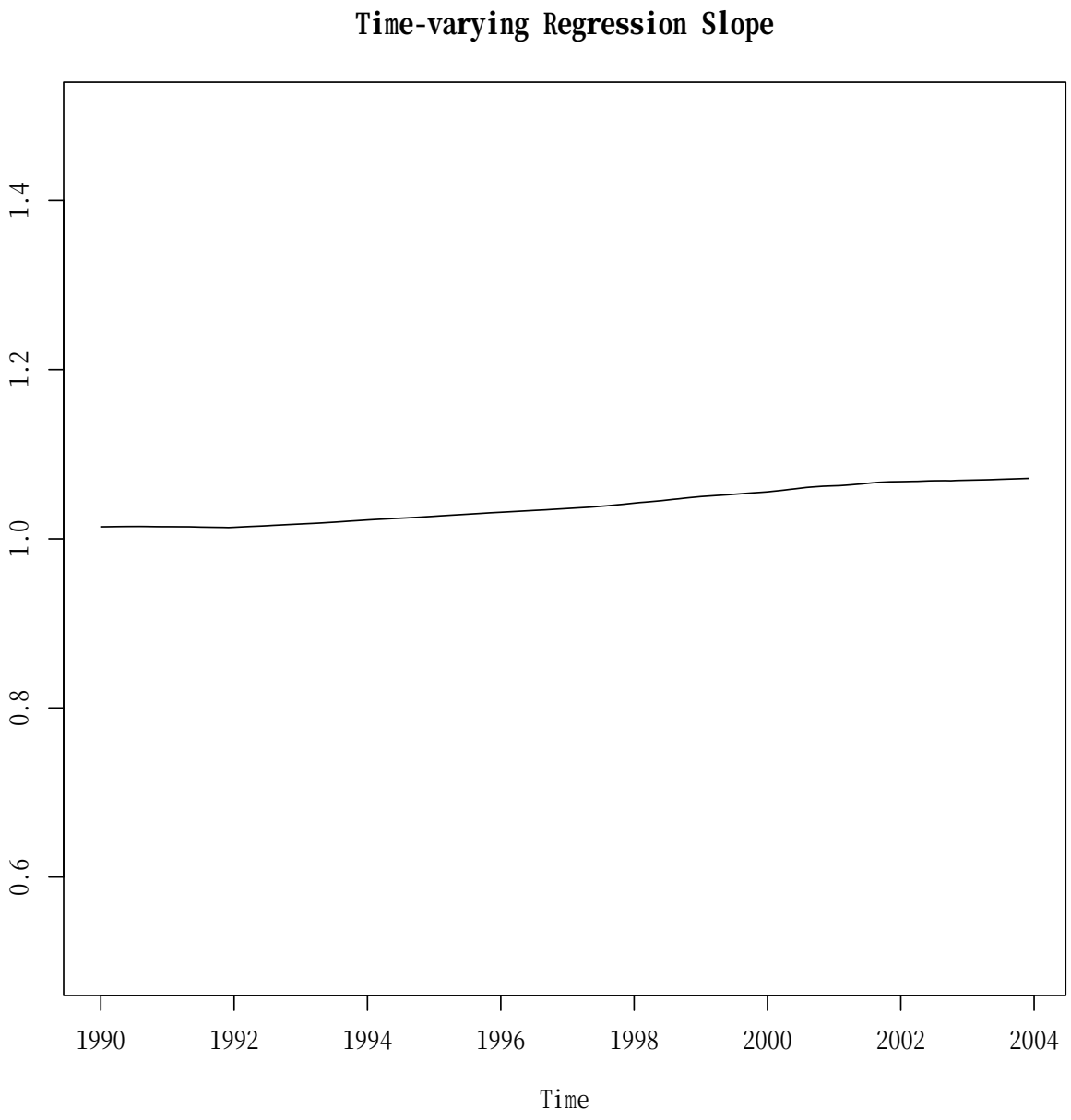
```
plot(ts.gm[,"GM"], main="GM and fitted with time-varying beta")
lines(as.vector(time(ts.gm)),
  c(fitted(res_gm, filtered=FALSE)),
  col="red")
```



这个拟合结果比较合理。

随机斜率项估计：

```
states <- as.matrix(coef(res_gm))
plot(ts(states[,2], start=c(1990, 1), frequency=12),
  main="Time-varying Regression Slope",
  ylab="",
  ylim = c(0.5, 1.5))
```



这个结果与滚动窗口回归计算的斜率项还是不一致，但是变化范围比较小，有一定合理性。

28.5.5.5 使用 dlm 包拟合时变系数回归模型

dlm 包也支持时变系数回归，可以用 dlm 尝试估计时变截距和斜率。

输入超参数的建模函数：

```
library(dlm)
buildFun <- function(para){
  dlmModReg(
    cbind(da_gm[["SP5"]]), # 自变量值
    dV = exp(para[1]),  # 回归误差方差
    dW= exp(para[2:3])) # 状态方程误差方差
}
```

超参数最大似然估计：

```
fit_dlm <- dlmMLE(
  da_gm[["GM"]],
  parm = c(0,-2,-2),
  build = buildFun)
fit_dlm$conv
```

```
## [1] 0
```

估计的数值算法收敛。提取最优参数的模型：

```
dlm_gm <- buildFun(fit_dlm$par)
```

回归误差项标准差：

```
V(dlm_gm) |> sqrt()
```

```
##          [,1]
## [1,] 8.130086
```

随机截距项和斜率项的标准差：

```
W(dlm_gm) |> diag() |> sqrt()
```

```
## [1] 9.177050e-04 1.271935e-06
```

估计得到的状态方程扰动项标准差很小，比 KFAS 模型得到的结果小得多，使得截距和斜率基本等于常数。

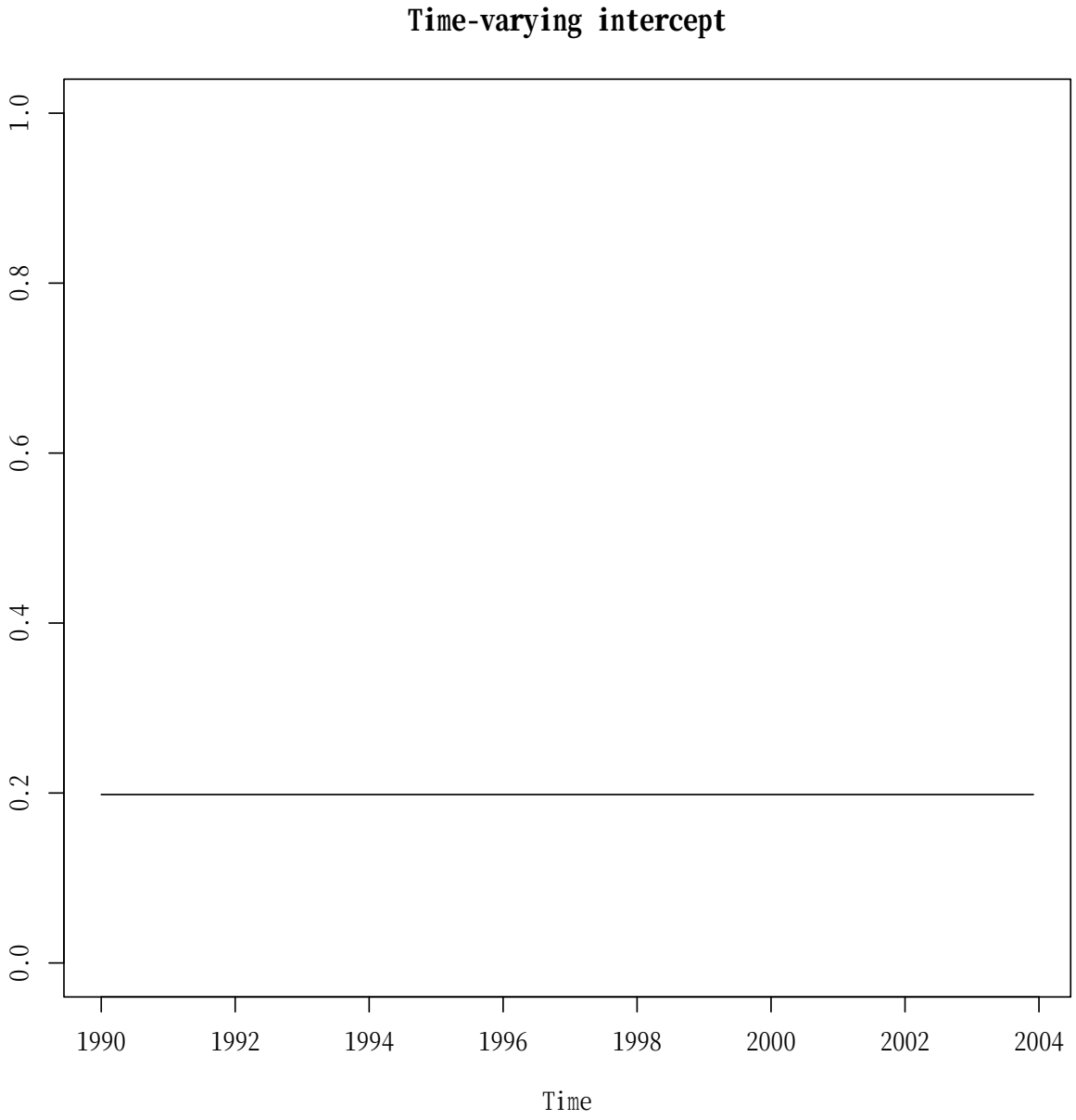
利用最大似然估计得到的超参数进行滤波和平滑：

```
filt_gm <- dlmFilter(da_gm[["GM"]], dlm_gm)
smth_gm <- dlmSmooth(filt_gm)
```

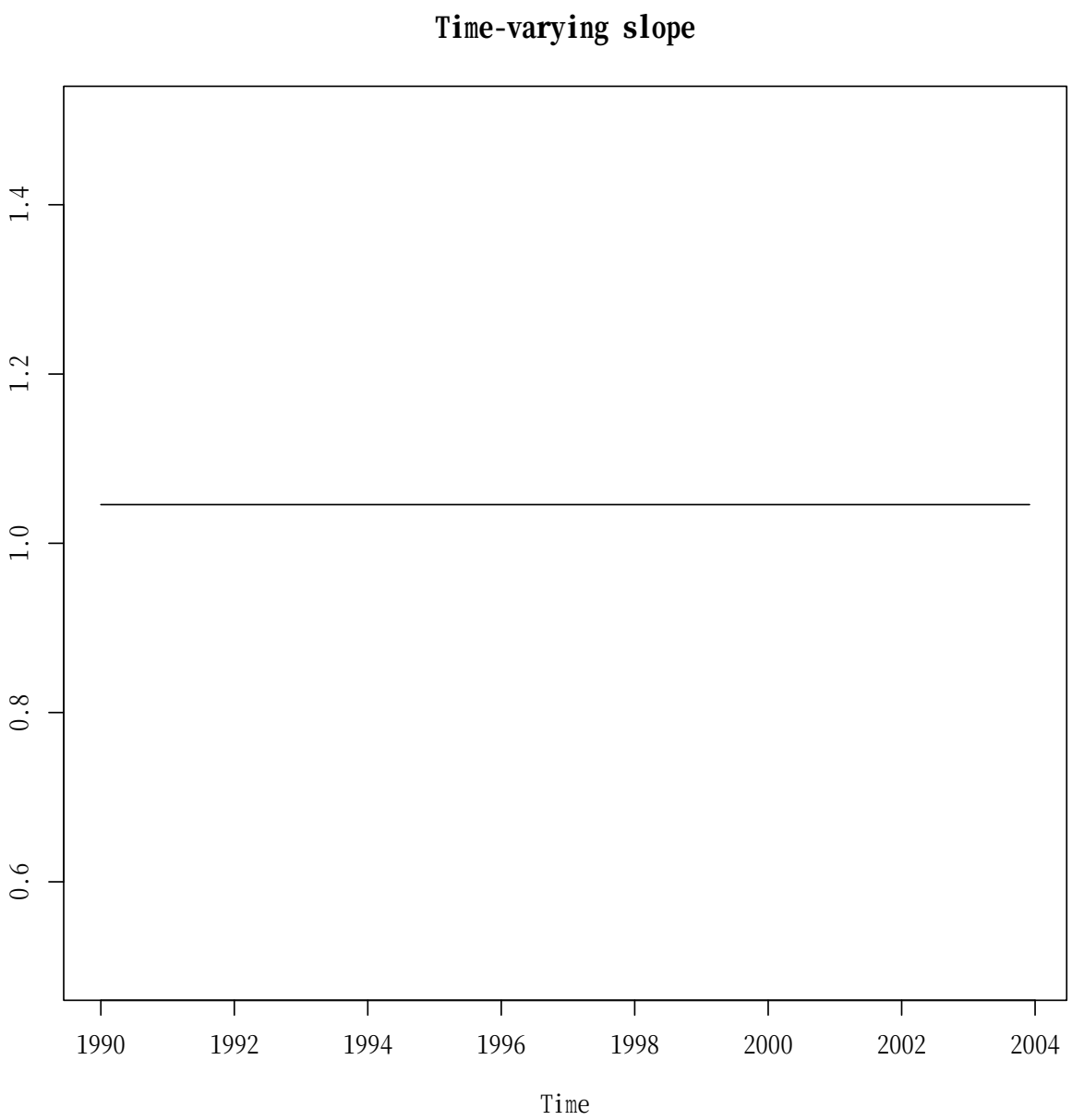
平滑得到的时变截距项和时变斜率项：

```
plot(ts(
  smth_gm$s[-1,1],
  start=c(1990, 1),
  frequency=12),
  ylim=c(0,1),
  main="Time-varying intercept",
```

```
  ylab="")
```




```
plot(ts(
  smth_gm$s[-1,2],
  start=c(1990, 1),
  frequency=12),
  ylim=c(0.5, 1.5),
  main="Time-varying slope",
  ylab="")
```



与 KFAS 的结果不一致。另外，实际上截距和斜率的变化范围很小，基本上是常数。

28.5.5.6 使用 MARSS 包拟合时变系数回归模型

```
library(MARSS)

mod_capm <- list(
    B = "identity",
    U = "zero",
    Q = "diagonal and unequal",
    Z = row2d3(cbind(1, da_gm$SP5)),
    A = "zero",
```

```
    R = matrix("r"),
    x0 = cbind(c("b00", "b10"))  )
fit_capm_em <- MARSS(
  rbind(da_gm$GM),
  model = mod_capm,
  control = list(maxit=100),
  inits = list(x0 = cbind(c(0,0))) )
```

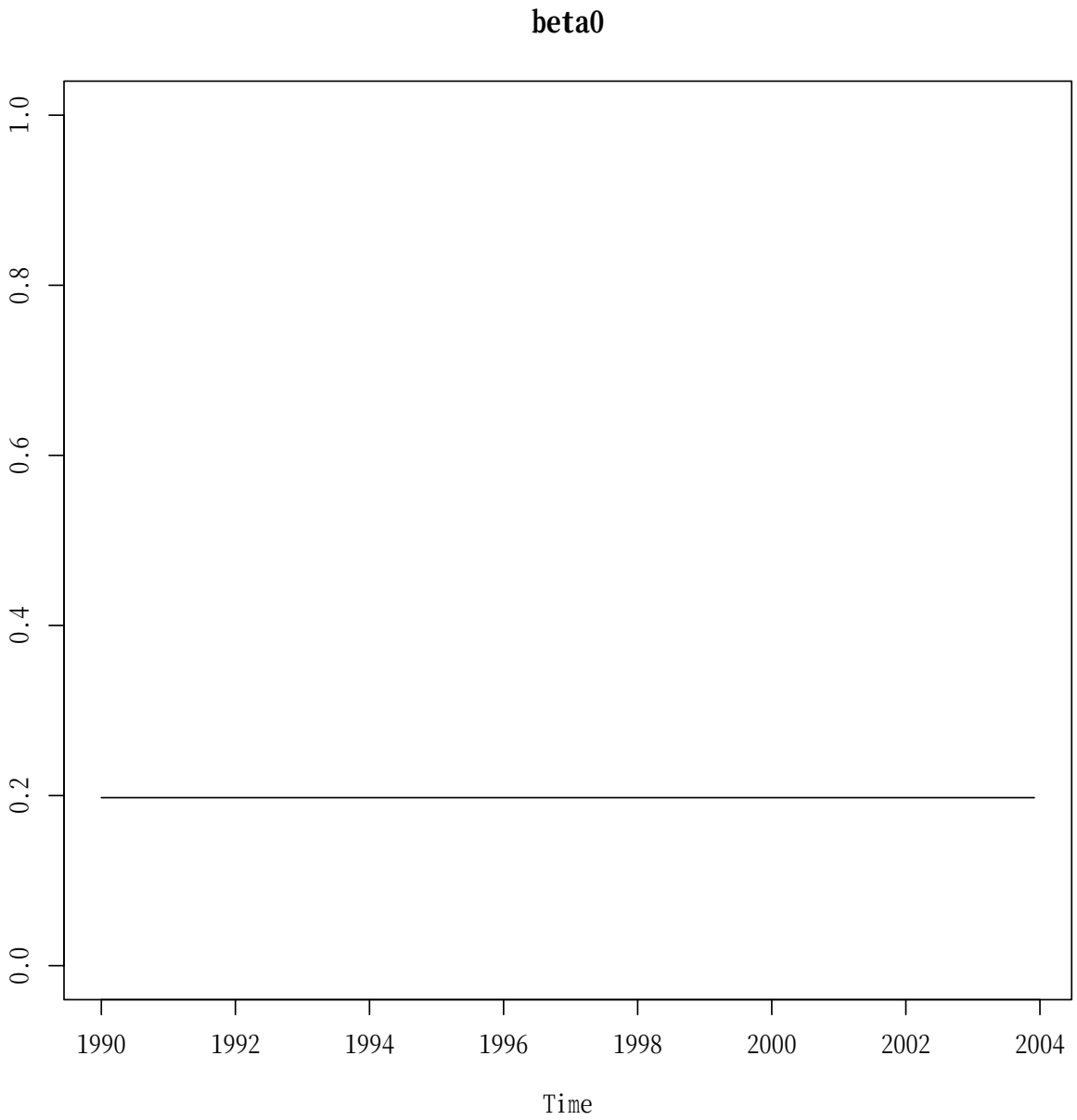
```
## Warning! Reached maxit before parameters converged. Maxit was 100.
##  neither abstol nor log-log convergence tests were passed.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## WARNING: maxit reached at  100  iter before convergence.
##  Neither abstol nor log-log convergence test were passed.
##  The likelihood and params are not at the MLE values.
##  Try setting control$maxit higher.
## Log-likelihood: -596.9282
## AIC: 1203.856   AICc: 1204.227
##
##           Estimate
## R.r        65.7133
## Q.(X1,X1)   0.1995
## Q.(X2,X2)   0.0254
## x0.b00      0.2843
## x0.b10      0.9198
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
##
## Convergence warnings
##  Warning: the  Q.(X1,X1)  parameter value has not converged.
##  Warning: the  Q.(X2,X2)  parameter value has not converged.
##  Warning: the  x0.b00  parameter value has not converged.
##  Warning: the  logLik  parameter value has not converged.
##  Type MARSSinfo("convergence") for more info on this warning.
```

```
fit_capm <- MARSS(
  rbind(da_gm$GM),
  model = mod_capm,
  inits = fit_capm_em$par,
  method = "BFGS")
```

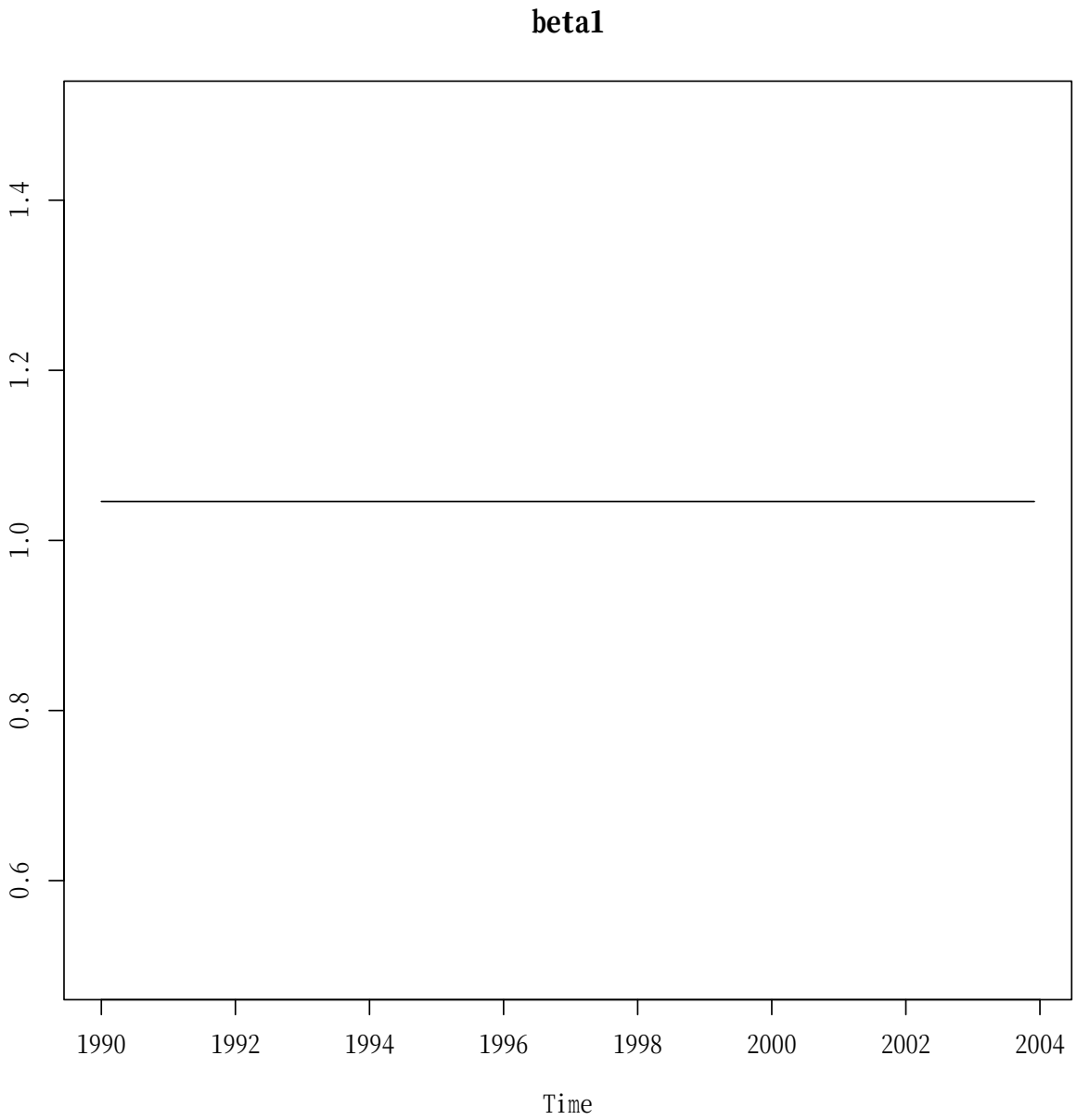
```
## Success! Converged in 11 iterations.
## Function MARSSkfas used for likelihood calculation.
##
## MARSS fit is
## Estimation method: BFGS
## Estimation converged in 11 iterations.
## Log-likelihood: -589.4323
## AIC: 1188.865   AICc: 1189.235
##
##           Estimate
## R.r       6.53e+01
## Q.(X1,X1) 1.07e-10
## Q.(X2,X2) 3.01e-13
## x0.b00    1.98e-01
## x0.b10    1.05e+00
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

提取 β_{0t} 和 β_{1t} 的估计序列：

```
extract_tvc <- function(mfit){
  res <- tsSmooth(mfit, type="xtT")
  sm <- ts(
    cbind(res[res$.rownames=="X1", ".estimate"],
          res[res$.rownames=="X2", ".estimate"]),
    start=c(1990, 1), frequency=12)
  sm
}
sm <- extract_tvc(fit_capm)
plot(sm[,1], ylim=c(0,1), main="beta0", ylab="")
```



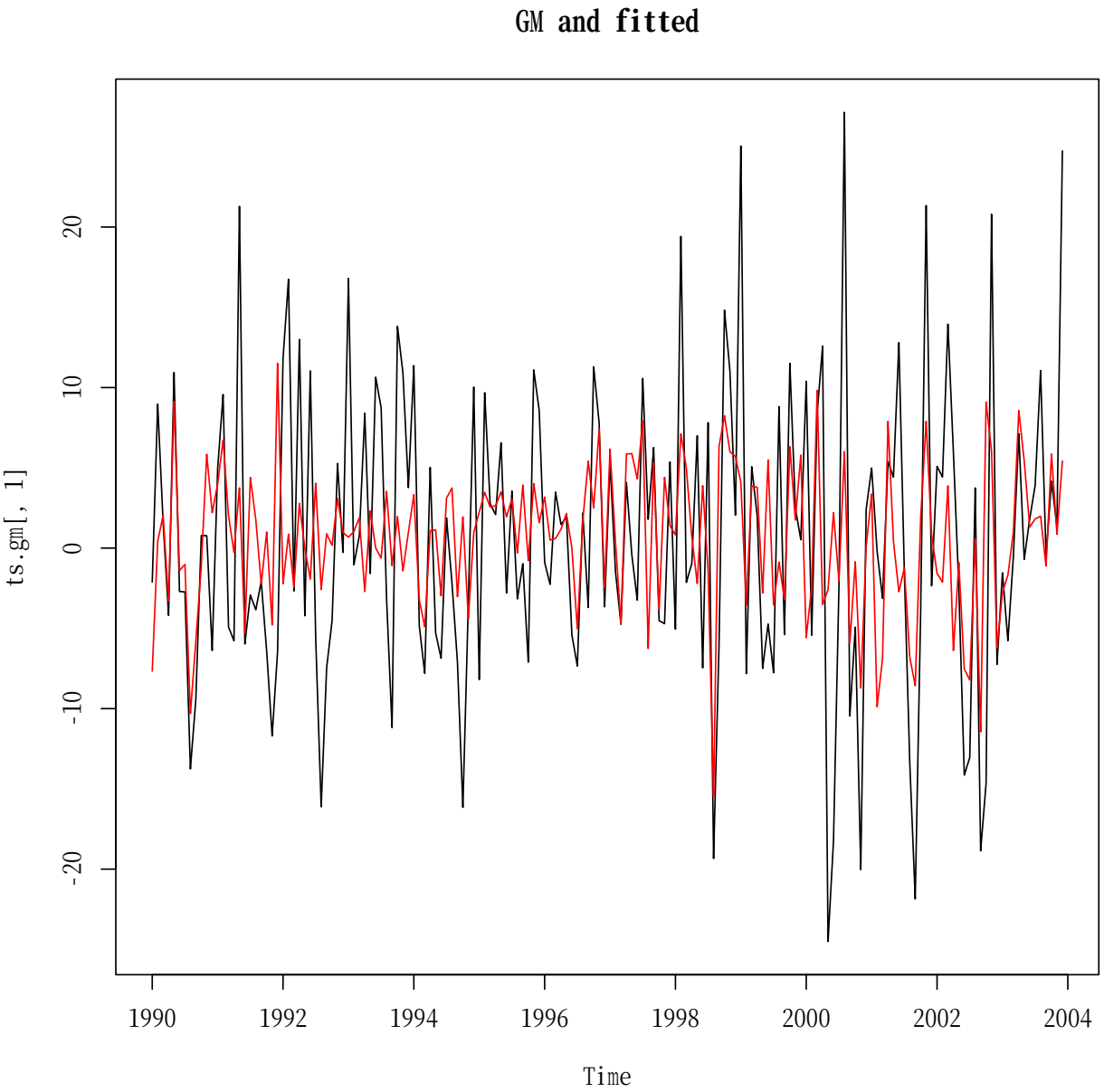

```
plot(sm[,2], ylim=c(0.5,1.5), main="beta1", ylab="")
```



结果与 KFAS, dlm 包结果类似，都相当于非时变的情形。

序列拟合结果：

```
res_capm_y <- fitted(fit_capm, type="ytT")
plot(ts.gm[,1], main="GM and fitted")
lines(as.vector(time(ts.gm)), res_capm_y$.fitted, col="red")
```



28.5.5.7 模型改进

前面的状态空间模型中时变的回归系数用了随机游动，注意随机游动是非平稳时间序列模型。计算结果都接近于常系数结果。

下面用平稳的 VAR(1) 模型作为回归系数模型。设

$$
\begin{aligned}
r_t =& \beta_{0t} + \beta_{1t} r_{M,t} + e_t,\ e_t \sim \text{ iid } \mathrm{N}(0, \sigma_e^2),\\
\beta_{0,t+1} =& b_{00} + b_{01}\beta_{0,t} + u_t,\ u_t \sim \text{ iid } \mathrm{N}(0, \sigma_u^2),\\
\beta_{1,t+1} =& b_{10} + b_{11}\beta_{1,t} + v_t,\ v_t \sim \text{ iid } \mathrm{N}(0, \sigma_v^2).
\end{aligned}
$$

```
mod_capm2 <- list(
    B = "diagonal and unequal",
    U = "unconstrained",
    Q = "diagonal and unequal",
    Z = row2d3(cbind(1, da_gm$SP5)),
    A = "zero",
    R = matrix("r"),
    x0 = cbind(c(mean(da_gm$GM), 0)),
    V0 = diag(1E6, 2))
fit_capm_em2 <- MARSS(
  rbind(da_gm$GM),
  model = mod_capm2,
  control = list(maxit=100))
```

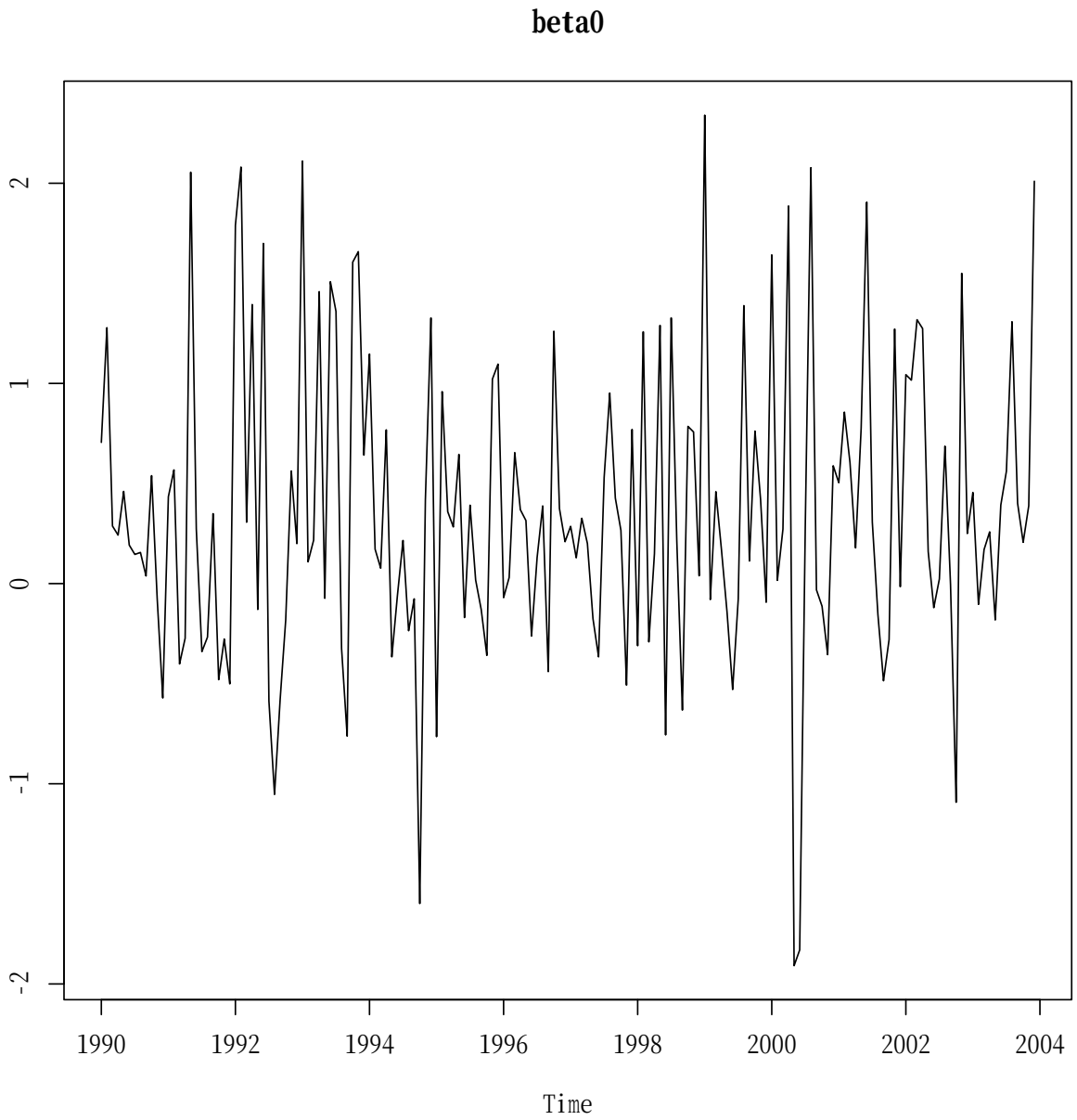
```
## Warning! Reached maxit before parameters converged. Maxit was 100.
##  neither abstol nor log-log convergence tests were passed.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## WARNING: maxit reached at  100  iter before convergence.
##  Neither abstol nor log-log convergence test were passed.
##  The likelihood and params are not at the MLE values.
##  Try setting control$maxit higher.
## Log-likelihood: -588.658
## AIC: 1191.316   AICc: 1192.016
##
##           Estimate
## R.r       47.40402
## B.(X1,X1)  0.01852
## B.(X2,X2) -0.00013
## U.X1       0.32866
## U.X2       1.03591
## Q.(X1,X1)  5.90385
## Q.(X2,X2)  0.70689
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
##
## Convergence warnings
##  Warning: the  logLik  parameter value has not converged.
##  Type MARSSinfo("convergence") for more info on this warning.
```

```
fit_capm2 <- MARSS(
  rbind(da_gm$GM),
  model = mod_capm2,
  inits = fit_capm_em2$par,
  method = "BFGS")
```

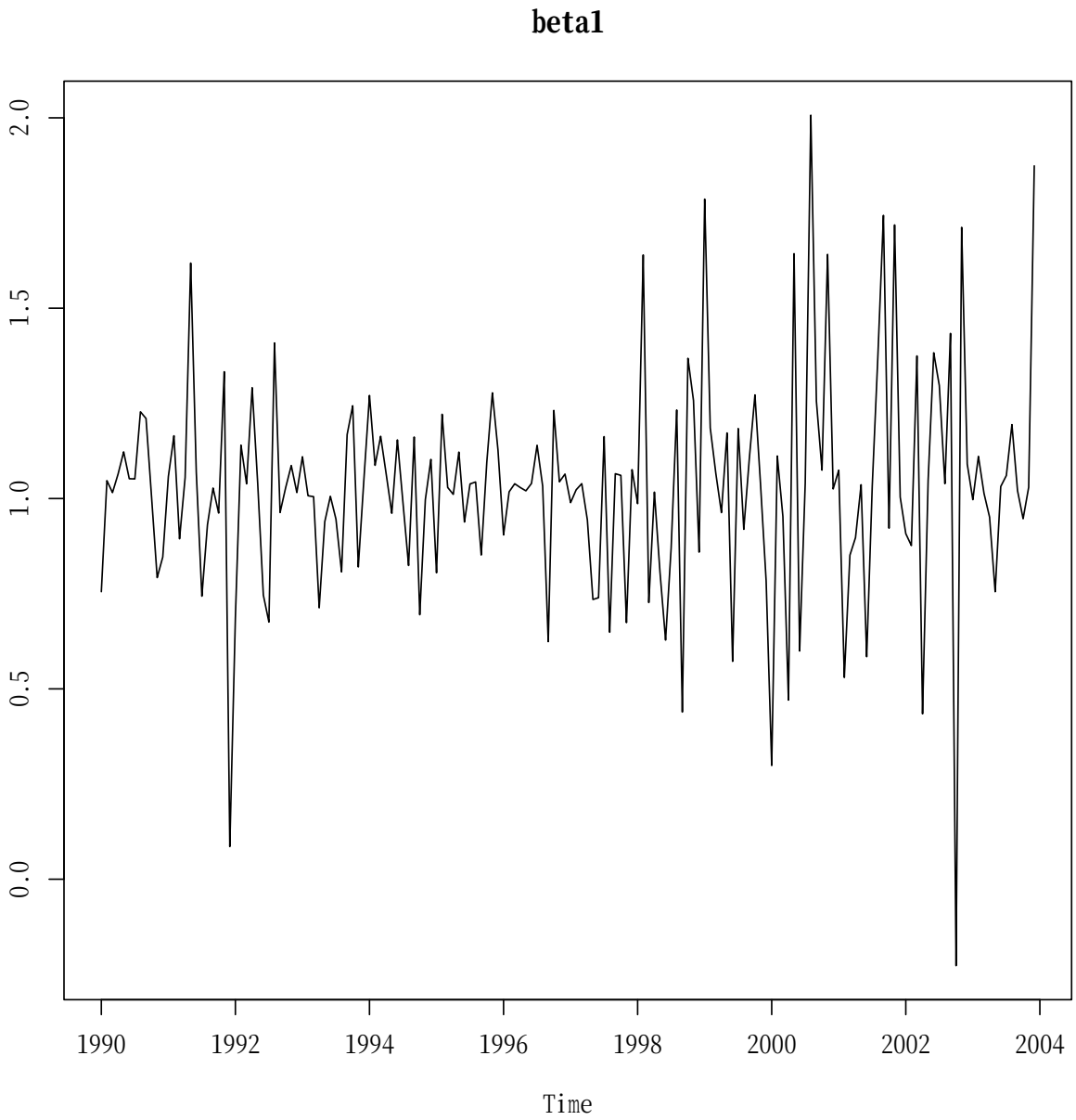
```
## Success! Converged in 11 iterations.
## Function MARSSkfas used for likelihood calculation.
##
## MARSS fit is
## Estimation method: BFGS
## Estimation converged in 11 iterations.
## Log-likelihood: -587.9871
## AIC: 1189.974   AICc: 1190.674
##
##            Estimate
## R.r        4.83e+01
## B.(X1,X1) -2.78e-05
## B.(X2,X2) -9.40e-06
## U.X1       3.51e-01
## U.X2       1.03e+00
## Q.(X1,X1)  5.99e+00
## Q.(X2,X2)  6.12e-01
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

用了扩散先验。但是，平稳的 VAR(1) 模型需要其初始值分布服从平稳分布，否则会影响系数矩阵 B 的估计。结果的时变回归系数基本是白噪声。改进效果不好。时变回归系数估计结果图形：

```
sm <- extract_tvc(fit_capm2)
plot(sm[,1], main="beta0", ylab="")
```




```
plot(sm[,2], main="beta1", ylab="")
```



将初始时间点设为 $t = 1$：

```
mod_capm3 <- list(
    B = "diagonal and unequal",
    U = "unconstrained",
    Q = "diagonal and unequal",
    Z = row2d3(cbind(1, da_gm$SP5)),
    A = "zero",
    R = matrix("r"),
    tinitx = 1
    )
fit_capm_em3 <- MARSS(
```

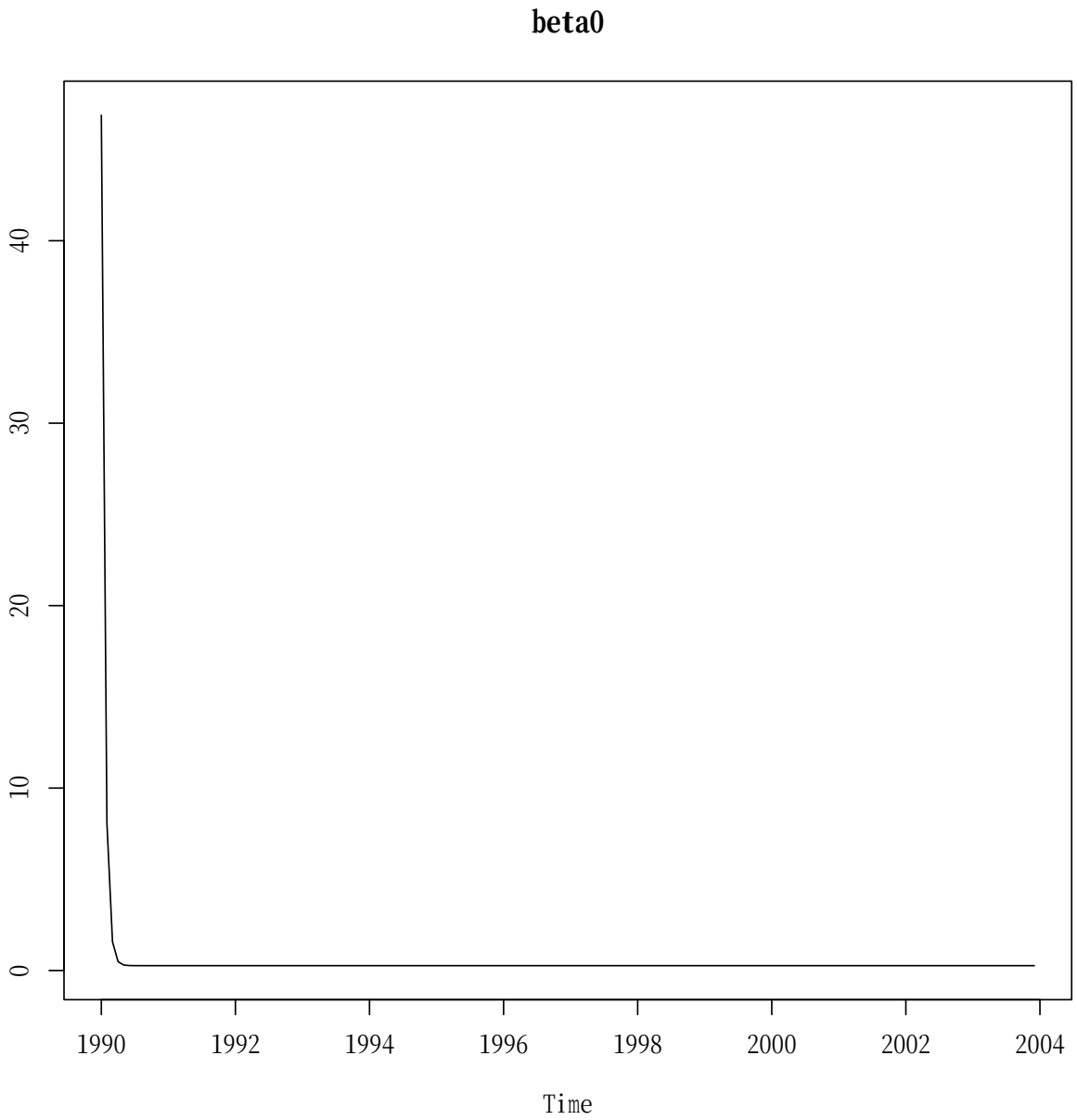
```
  rbind(da_gm$GM),
  model = mod_capm3,
  inits = list(x0 = cbind(c(0,0))),
  control = list(maxit=100))
```

```
## Warning! Reached maxit before parameters converged. Maxit was 100.
##  neither abstol nor log-log convergence tests were passed.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## WARNING: maxit reached at  100  iter before convergence.
##  Neither abstol nor log-log convergence test were passed.
##  The likelihood and params are not at the MLE values.
##  Try setting control$maxit higher.
## Log-likelihood: -586.2302
## AIC: 1190.46   AICc: 1191.6
##
##           Estimate
## R.r         44.938
## B.(X1,X1)    0.159
## B.(X2,X2)   -0.314
## U.X1         0.235
## U.X2         1.329
## Q.(X1,X1)    5.189
## Q.(X2,X2)    0.778
## x0.X1       46.751
## x0.X2        6.486
## Initial states (x0) defined at t=1
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
##
## Convergence warnings
##  Warning: the  x0.X1  parameter value has not converged.
##  Warning: the  x0.X2  parameter value has not converged.
##  Type MARSSinfo("convergence") for more info on this warning.
```

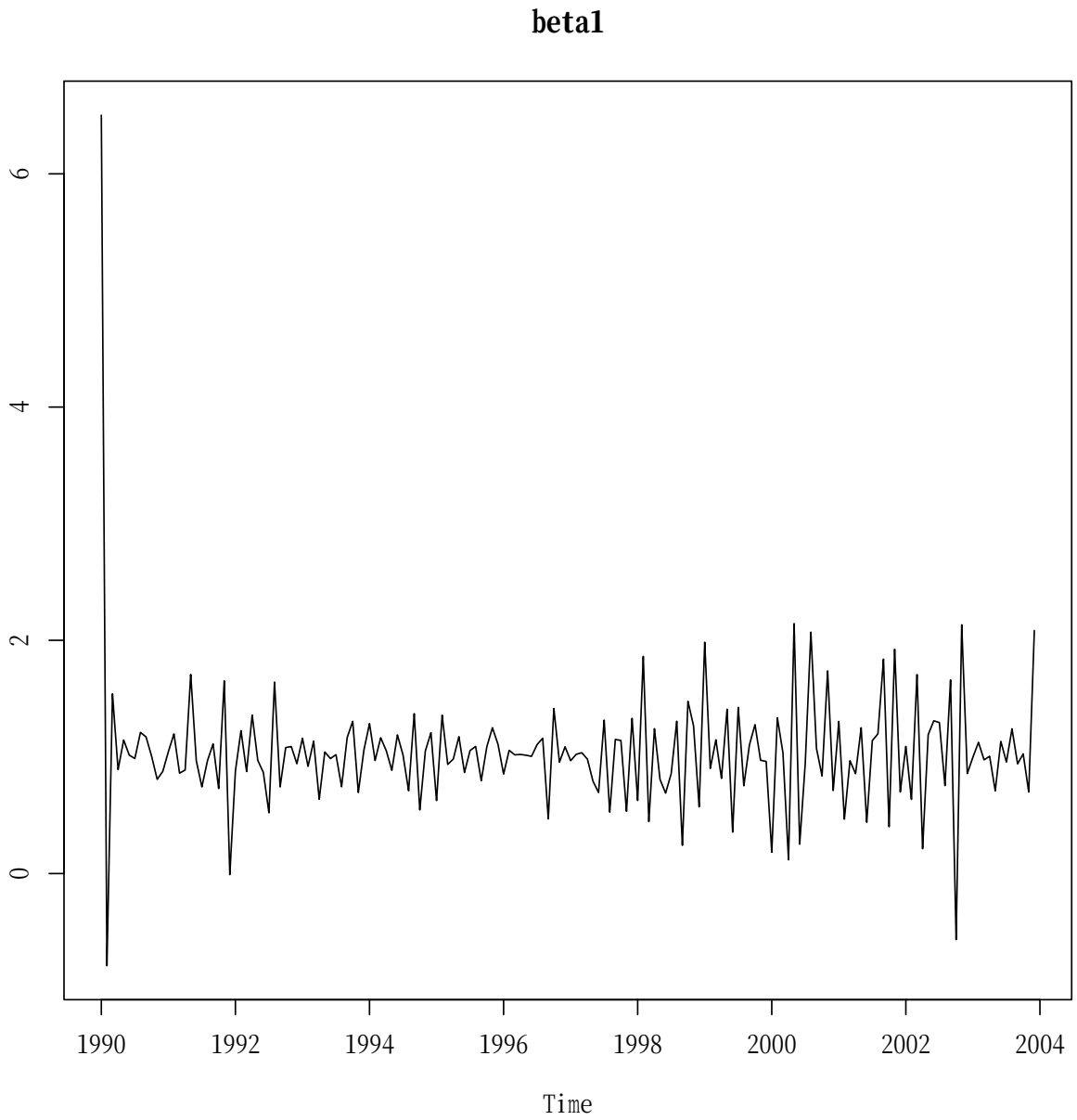
```
fit_capm3 <- MARSS(
  rbind(da_gm$GM),
  model = mod_capm3,
  inits = fit_capm_em3$par,
  method = "BFGS")
```

```
## Success! Converged in 17 iterations.
## Function MARSSkfas used for likelihood calculation.
##
## MARSS fit is
## Estimation method: BFGS
## Estimation converged in 17 iterations.
## Log-likelihood: -586.164
## AIC: 1190.328   AICc: 1191.467
##
##             Estimate
## R.r         5.10e+01
## B.(X1,X1)   1.68e-01
## B.(X2,X2) -3.35e-01
## U.X1        2.21e-01
## U.X2        1.36e+00
## Q.(X1,X1)   1.83e-06
## Q.(X2,X2)   6.96e-01
## x0.X1       4.69e+01
## x0.X2       6.50e+00
## Initial states (x0) defined at t=1
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

```
sm <- extract_tvc(fit_capm3)
plot(sm[,1], main="beta0", ylab="")
```




```
plot(sm[,2], main="beta1", ylab="")
```



初值估计还是有较大问题。

28.5.6 带有 ARMA 误差的回归模型

设

$$y_t = X_t\beta + \xi_t,$$

ξ_t 服从 ARMA 模型，可以写成状态空间形式，状态方程为

$$\alpha_{t+1} = T\alpha_t + R\eta_t,$$

则 y_t 可以写成状态空间模型形式，状态变量为 α_t^*，模型为

$$\begin{aligned}\alpha_t^* =& \begin{pmatrix}\beta\\ \alpha_t\end{pmatrix},\\ y_t =& (X_t, 1, 0, \dots, 0)\alpha_t^*,\\ \alpha_{t+1}^* =& \begin{pmatrix}I_k & 0\\ 0 & T\end{pmatrix}\alpha_t^* + \begin{pmatrix}0\\ R\end{pmatrix}\eta_t,\end{aligned}$$

可以用状态空间模型的工具进行估计和推断。

MARSS 包直接支持协变量回归，所以可以将时变误差放在系统方程中，将协变量回归放在观测方程中，观测方程中还可以带有本身的独立观测误差：

$$\begin{aligned}x_t =& bx_{t-1} + w_t,\ w_t \sim \mathrm{N}(0, q),\\ y_t =& x_t + \alpha + (\beta_1, \beta_2, \dots)(d_{1t}, d_{2t}, \dots)^T + v_t,\ v_t \sim \mathrm{N}(0, r).\end{aligned}$$

28.5.7 动态 Nelson-Siegel 模型

考虑如下的关于利率期限结构的动态 Nelson-Siegel 模型：

$$\begin{aligned}y_t(\tau) =& \beta_{1t} + \beta_{2t}\left(\frac{1-e^{-\lambda\tau}}{\lambda\tau}\right) + \beta_{3t}\left(\frac{1-e^{-\lambda\tau}}{\lambda\tau} - e^{-\lambda\tau}\right) + e_t(\tau),\\ &\tau = 3, 6, 12, 24, 36, 60, 84, 120,\ t = 1, 2, \dots, n.\end{aligned}$$

其中 λ 是未知参数，τ 是贷款期限，$y_t(\tau)$ 是 t 时刻期限为 τ 的贷款利息。设各 $e_t(\tau)$ 相互独立，服从 $\mathrm{N}(0, \sigma_e^2)$；$\beta_{1t}$, β_{2t}, β_{3t} 是时变的回归系数，设其服从一阶向量自回归模型：

$$\beta_{t+1} = (I - \Phi)\mu + \Phi\beta_t + \eta_t,$$

其中 $\beta_t = (\beta_{1t}, \beta_{2t}, \beta_{3t})^T$, μ 为 β_t 的均值，Φ 为 3×3 的回归系数矩阵，设 $\eta_t \sim \mathrm{N}(0, \Sigma_\eta)$, $\beta_1 \sim \mathrm{N}(\mu, P_\beta)$，初始分布中的 P_β 也满足平稳性条件

$$P_\beta - \Phi P_\beta \Phi^T = \Sigma_\eta.$$

因为对每个 t 有 8 个不同的 τ 对应的观测值 $y_t(\tau)$，将这 8 个观测值写成一个 8×1 观测值向量 y_t。对包含期望值 μ 的 VAR(1) 模型，可转换成如下的状态方程：

$$\begin{aligned}\alpha_t =& \begin{pmatrix}\beta_t\\ \mu\end{pmatrix},\\ \begin{pmatrix}\beta_{t+1}\\ \mu\end{pmatrix} =& \begin{pmatrix}\Phi & I-\Phi\\ 0 & I\end{pmatrix}\begin{pmatrix}\beta_t\\ \mu\end{pmatrix} + \begin{pmatrix}I\\ 0\end{pmatrix}\eta_t,\ \eta_t \sim \mathrm{N}(0, \Sigma_\eta).\end{aligned}$$

令 $\tau = (3, 6, 12, 24, 36, 60, 84, 120)^T$, 观测方程可以写成

$$y_t = \begin{pmatrix}Z^{(1)} & 0_{8\times 3}\end{pmatrix}\alpha_t + e_t,\ e_t \sim \mathrm{N}(0, \sigma_e^2 I_8),$$

其中 $Z^{(1)}$ 是 8×3 矩阵，第一列元素都等于 1，第二列元素为各 $\frac{1-e^{-\lambda\tau}}{\lambda\tau}$ 值，第三列元素为各 $\frac{1-e^{-\lambda\tau}}{\lambda\tau} - e^{-\lambda\tau}$ 值。

这样将 DNS(动态 Nelson-Siegel) 模型写成了状态空间形式，可以用 statespacer 包进行估计和平滑。如果不用状态空间模型方法，可以先对每个 t 的 8 个观测回归得到 3 个系数在 t 时刻的估计值，然后再对估计的 β_{jt} 建模，这样的做法不能充分利用模型结构。

28.5.7.1 用 statespacer 建模

statespacer 包对于 ARIMA、结构时间序列模型都有一些方便的模型参数规定方式和输出方式，对于更一般的模型就需要用户指定需要各个矩阵和初始值。

示例所用的数据是 YieldCurve 扩展包中的 `FedYieldCurve` 数据集。取 1984-12-31 到 2000-12-01 的数据子集。数据是 192×8 矩阵，有 192 天 (对应 t)，8 个期限（对应 τ）。

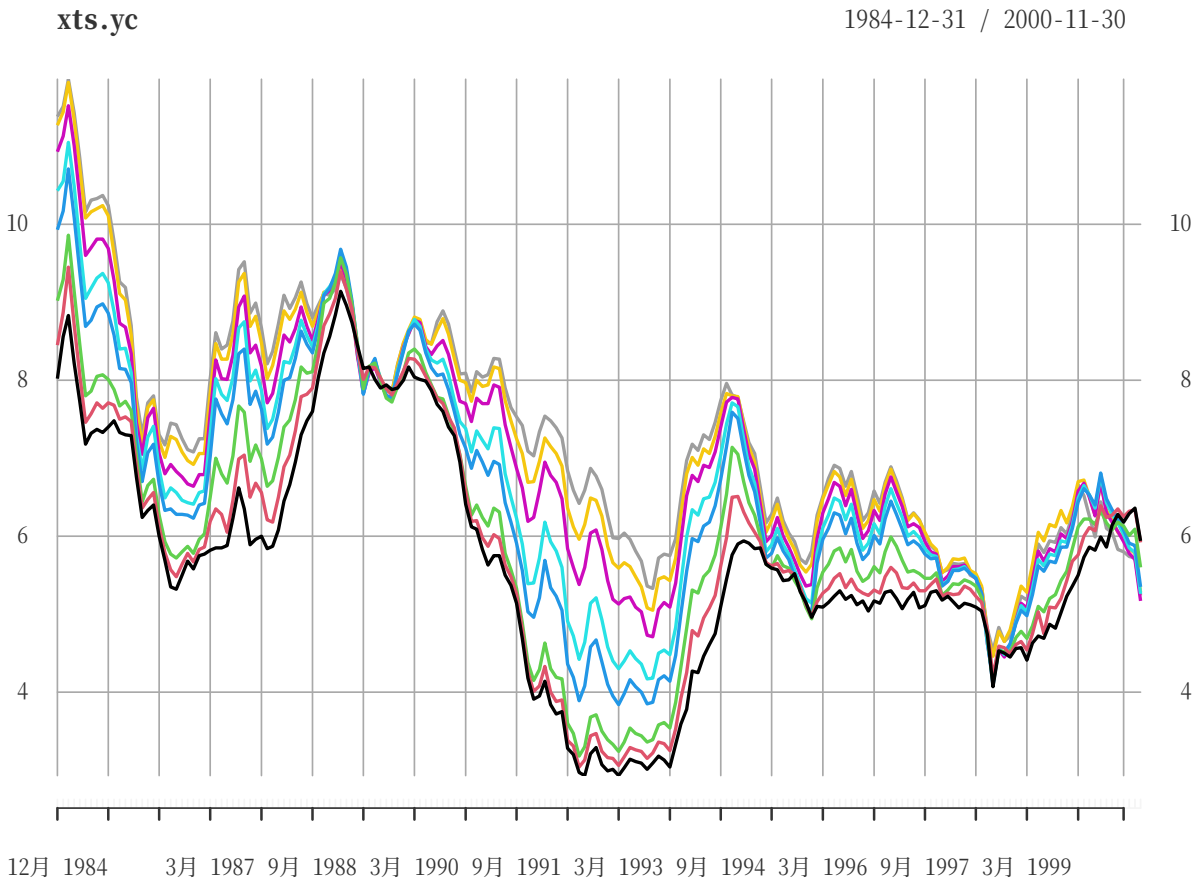
```
library(statespacer)
library(YieldCurve, quietly=TRUE, warn.conflicts=FALSE)
```

```
## Warning: package 'YieldCurve' was built under R version 4.2.3
```

```
data(FedYieldCurve)
str(FedYieldCurve)
```

```
## An 'xts' object on 1981-12-31/2012-11-30 containing:
##   Data: num [1:372, 1:8] 12.9 14.3 13.3 13.3 12.7 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:8] "R_3M" "R_6M" "R_1Y" "R_2Y" ...
##   Indexed by objects of class: [Date] TZ:
##   xts Attributes:
##  NULL
```

```
xts.yc <- FedYieldCurve["1984-12-31/2000-12-01"]
date.yc <- index(xts.yc)
y.yc <- coredata(xts.yc)
plot(xts.yc, multi.panel=FALSE)
```



需要自己指定模型的各个矩阵和初值。

```
## 保存自定义设置用的列表
spec <- list()
## 自己规定 H 矩阵（观测误差的方差阵）
spec$H_spec <- TRUE
## 状态向量维数，3 个 beta，3 个常数 beta 均值
spec$state_num <- 6

## 待估参数个数 20 个，包括：
##   1 个 - $\lambda$
##   1 个 - $\sigma_e^2$
##   6 个 - $\Sigma_{\boldsymbol\eta}\ 3 \times 3$ 对称阵
##   9 个 - $\Phi_{3\times 3}$ 矩阵
##   3 个 - $\boldsymbol\mu$
spec$param_num <- 20

## 状态方程中 $R$ 矩阵不起作用，方差结构直接编写在矩阵 $Q$ 中
spec$R <- diag(1, 6, 6) # $I_6$
```

```
## 初始状态 $\boldsymbol\alpha_1$ 的方差，元素都取为 0，因为平稳
spec$P_inf <- matrix(0, 6, 6)

## 指定状态向量中不输出到观测的分量，
## 当压缩观测向量为标量时可提高计算效率
spec$state_only <- 4:6
```

模型要估计的参数（超参数）为 λ, σ_e^2, 对称的 $(\Sigma_\eta)_{3\times3}$, $\Phi_{3\times3}$, $\mu_{3\times1}$。

对于必须取正值的参数，将其迭代计算的参数取自然对数后除以 2。

对于要求正定的 Σ_η，将其做 Cholesky 分解 LDL^T，其中 D 是对角元素为正值的对角阵，L 是对角元素都等于 1 的下三角阵，用 D 和 L 的 6 个元素来作为待估参数。statespacer 提供了 Cholesky 函数将这样编码的参数转换成正定矩阵。

对向量自回归系数矩阵 Φ，要求其满足平稳性条件，statespacer 包的 `CoeffARMA()` 函数用来改造输入的系数矩阵使其满足条件。

因为模型的各个重要矩阵、初始值等都需要编程计算而不是常数值，所以将自定义设定的 `sys_mat_fun` 元素设置为一个自定义函数：

```
spec$sys_mat_fun <- function(param) {
  ## 输入 20 个参数，具体意义见前面注释和说明

  # 8 个期限，即 $\tau$ 值
  maturity <- c(3, 6, 12, 24, 36, 60, 84, 120)

  # 从取了对数的参数值恢复 $\lamabda$ 的值
  lambda <- exp(2 * param[1])

  # \sigma_e^2 的值
  sigma2 <- exp(2 * param[2])

  # H 是观测方程的误差向量的协方差阵参数。
  # 注意每个 $t$ 对应的观测值是 $8\times 1$ 向量。
  H <- sigma2 * diag(1, 8, 8)

  # 观测方程的 Z 矩阵，一个 $8 \times 6$ 矩阵，
  # 前三列对应于 $\beta_{1t}$, $\beta_{2t}$, $\beta_{3t}$
  # 后三列对应于 $\beta$ 的三个均值，系数为 0
  lambda_maturity <- lambda * maturity
  ze <- exp(-lambda_maturity)
  Z <- matrix(1, 8, 3)
  Z[, 2] <- (1 - ze) / lambda_maturity
  Z[, 3] <- Z[, 2] - ze
```

```
# 状态方程的误差方差阵（$6 \times 6$）的左上角 $3 \times 3$ 部分
# 使用 6 个参数
Q <- Cholesky(
  param = param[3:8],
  decompositions = FALSE,
  format = matrix(1, 3, 3))

# 从输入的 9 个矩阵元素生成满足平稳性条件的向量自回归系数矩阵 $\Phi$
Tmat <- CoeffARMA(
  A = array(param[9:17], dim = c(3, 3, 1)),
  variance = Q,
  ar = 1, ma = 0)$ar[,,1]

# 生成初始状态 $\boldsymbol\alpha_1$ 的方差阵，确保其满足平稳性条件
T_kronecker <- kronecker(Tmat, Tmat)
Tinv <- solve(
  diag(1, dim(T_kronecker)[1],
       dim(T_kronecker)[2]) - T_kronecker)
vecQ <- matrix(Q)
vecPstar <- Tinv %*% vecQ
P_star <- matrix(vecPstar, dim(Tmat)[1], dim(Tmat)[2])


## 添加对应于常数均值的部分
# 在观测方程的矩阵 Z 中增加对应于常数均值的 3 列
Z <- cbind(Z, matrix(0, 8, 3))
# 在状态方程的误差方差阵中增加对应于常数均值的部分，都是 0
Q <- BlockMatrix(Q, matrix(0, 3, 3))
# 在 6 维的状态方程中，设置转移矩阵 $T$
Tmat <- cbind(Tmat, diag(1, 3, 3) - Tmat)
Tmat <- rbind(Tmat, cbind(matrix(0, 3, 3), diag(1, 3, 3)))
# 初始状态的方差阵中对应于常数均值部分方差和协方差为 0
P_star <- BlockMatrix(P_star, matrix(0, 3, 3))

# $\beta$ 的均值是最后的 3 个待估参数，
# 也作为 $t=1$ 时 $\beta$ 的初始分布均值
a1 <- matrix(param[18:20], 6, 1)

# 函数返回模型所需的所有系统矩阵
return(list(
  H = H,    # 观测误差随机向量的方差阵
  Z = Z,    # 观测方程的观测矩阵
  Tmat = Tmat, # 状态方程的状态转移矩阵
```

```
    Q = Q,   # 状态方程的误差随机向量的方差阵
    a1 = a1, # 初始状态的均值
    P_star = P_star)) # 初始状态的方差阵
}
```

为了从估计的参数（超参数）提取某些成分，可以提供一个 transform_fun 函数：

```
spec$transform_fun <- function(param) {
  lambda <- exp(2 * param[1])
  sigma2 <- exp(2 * param[2])
  means <- param[18:20]
  return(c(lambda, sigma2, means))
}
```

下面给 20 个参数设置算法迭代初值：

```
initial <- c(
  -1, # $0.5 \ln\lambda$
  -2, # $0.5 \ln\sigma_e^2$
  # $\Sigma_{\boldsymbol\eta}$ 的 LDL 分解中 D 的对角元素的 0.5 倍对数值
  -1, -1, -1,
  # $\Sigma_{\boldsymbol\eta}$ 的 LDL 分解中 L 的元素值
  0, 0, 0,
  # VAR(1) 模型系数矩阵
  1, 0, 0,
  0, 1, 0,
  0, 0, 1,
  # 三个 beta 的均值
  0, 0, 0)
```

进行模型拟合，statespacer() 函数需要调用 optim 函数进行最大似然估计计算，可能需要消耗一定的时间：

```
fit <- statespacer(
  y = y.yc,
  self_spec_list = spec,
  collapse = TRUE,
  initial = initial,
  method = "BFGS",
  control = list(maxit = 200),
  verbose = TRUE)
```

```
## Starting the optimisation procedure at: 2023-05-30 17:05:05

## initial  value 18.089728
## iter  10 value -6.160531
## iter  20 value -6.760946
## iter  30 value -6.953583
```

```
## iter  40 value -6.990606
## iter  50 value -6.996315
## iter  60 value -6.999004
## iter  70 value -7.002370
## iter  80 value -7.004278
## iter  90 value -7.005021
## iter 100 value -7.005174
## iter 110 value -7.005240
## iter 120 value -7.005375
## iter 130 value -7.005400
## iter 140 value -7.005430
## iter 150 value -7.005440
## iter 160 value -7.005462
## final  value -7.005470
## converged
```

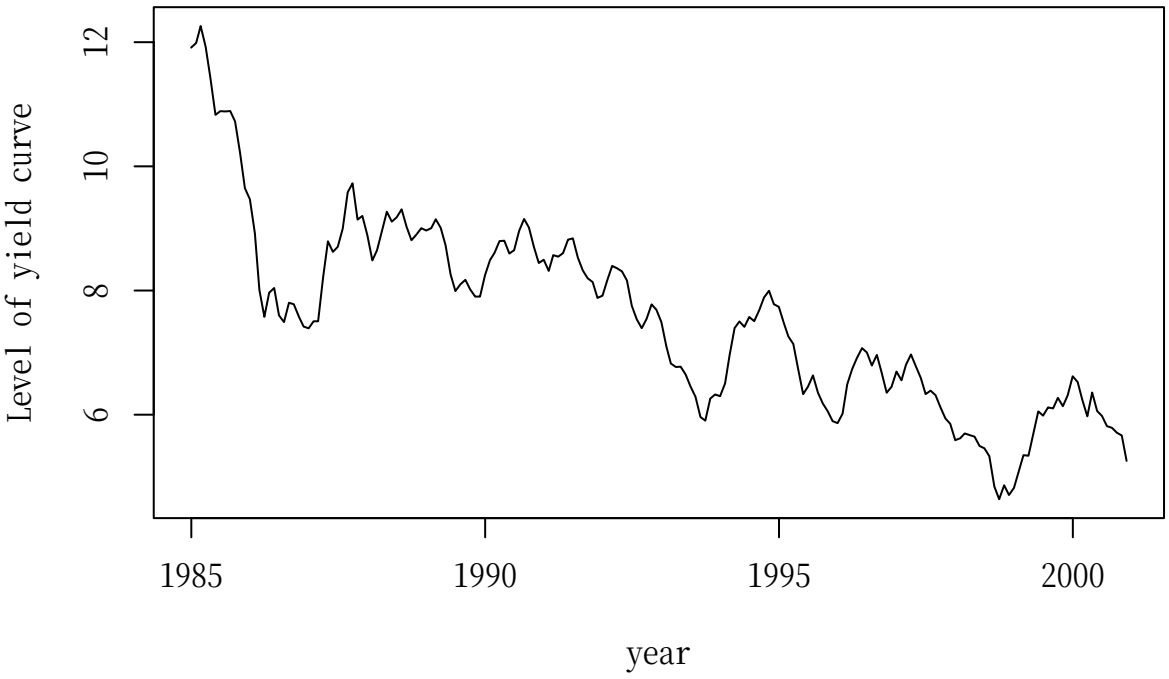
```
## Finished the optimisation procedure at: 2023-05-30 17:05:18
```

```
## Time difference of 12.8279469013214 secs
```

算法有可能在中间因矩阵不可逆而中断，这时可以修改初值。模型估计一次需要大约半分钟。将计算系统矩阵的函数改写为 C++ 版本可能会加快速度。

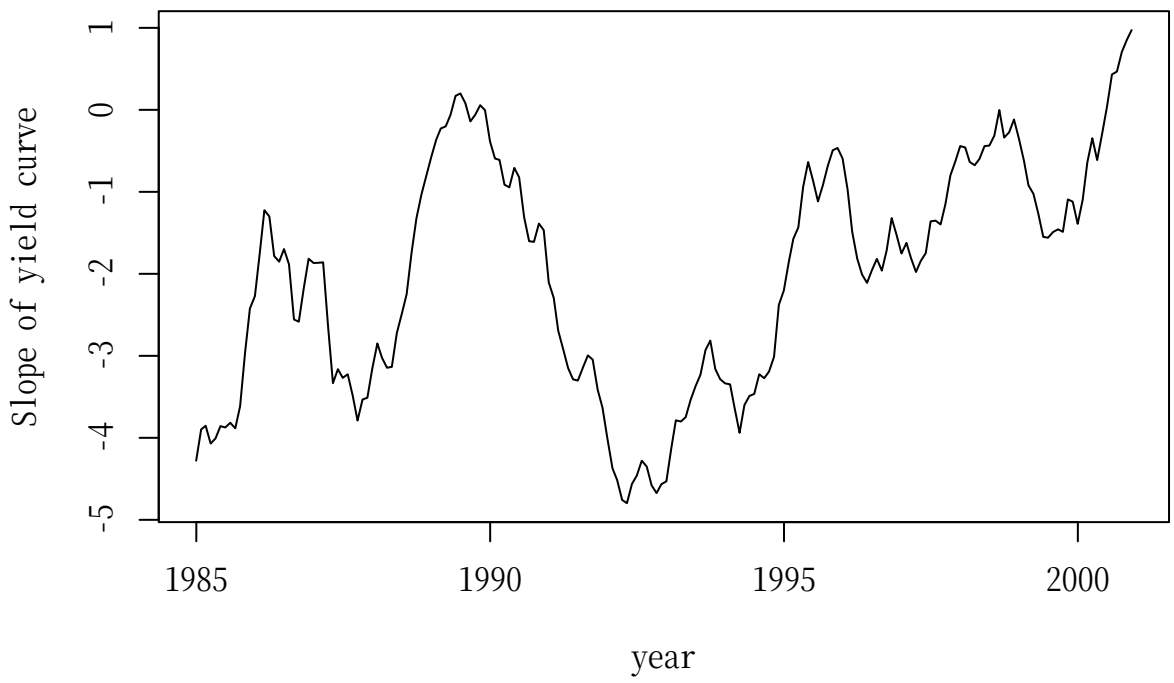
平滑得到的 β_{1t} 的时间序列图形：

```
plot(date.yc, fit$smoothed$a[, 1], type = 'l',
  xlab = "year", ylab = "Level of yield curve")
```



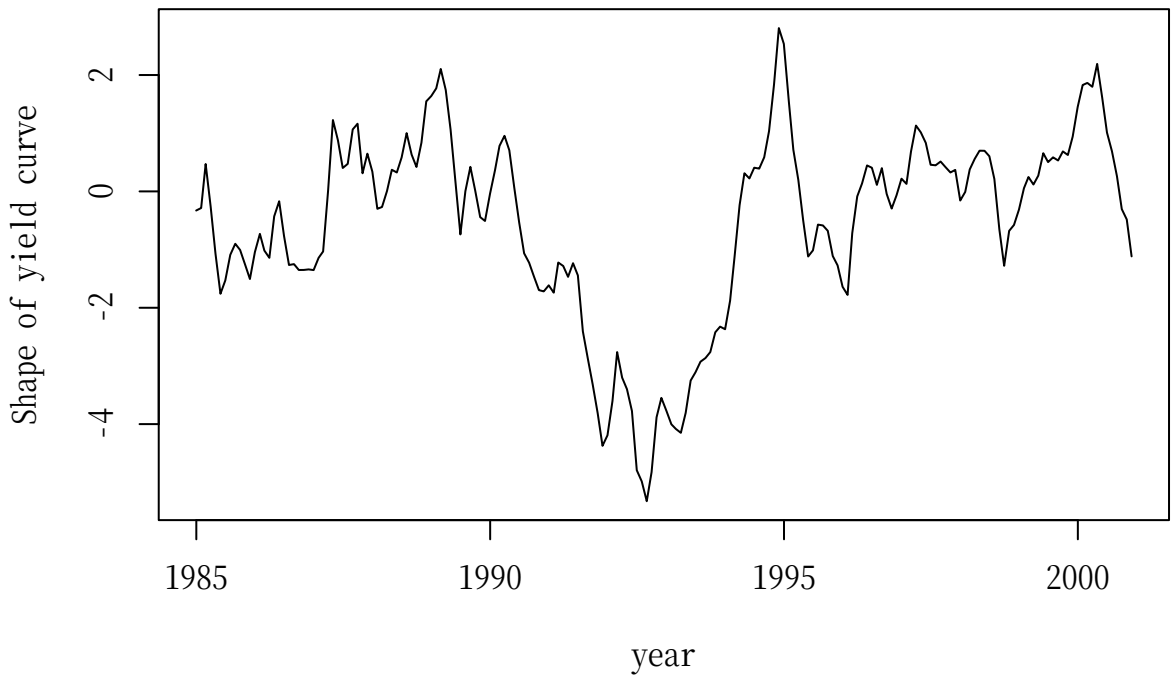
平滑得到的 β_{2t} 的时间序列图形：

```
plot(date.yc, fit$smoothed$a[, 2], type = 'l',
  xlab = "year", ylab = "Slope of yield curve")
```



平滑得到的 β_{3t} 的时间序列图形：

```
plot(date.yc, fit$smoothed$a[, 3], type = 'l',
  xlab = "year", ylab = "Shape of yield curve")
```



获取估计的参数：

```
parameters <- data.frame(
  Parameter = c("lambda", "sigma2", "mu1", "mu2", "mu3"),
  Value = fit$system_matrices$self_spec,
  SE = fit$standard_errors$self_spec
)
knitr::kable(parameters, digits=4)
```

Parameter	Value	SE
lambda	0.0789	0.0017
sigma2	0.0035	0.0002
mu1	8.2277	2.0759
mu2	-2.2748	1.4364
mu3	-0.4256	0.7191

向量自回归系数矩阵：

```
fit$system_matrices$T$self_spec[1:3, 1:3]
```

```
##              [,1]        [,2]        [,3]
## [1,]  0.98488006 -0.01776100 0.005741521
## [2,] -0.01980538  0.94678326 0.062497259
## [3,] -0.03354339 -0.04024349 0.964429517
```

状态方程误差方差阵：

```
fit$system_matrices$Q$self_spec[1:3, 1:3]
```

```
##              [,1]        [,2]        [,3]
## [1,]  0.06723244 -0.05232934  0.06856474
## [2,] -0.05232934  0.07118337 -0.04047250
## [3,]  0.06856474 -0.04047250  0.24097759
```

28.5.8 四地点海豹数目调查建模

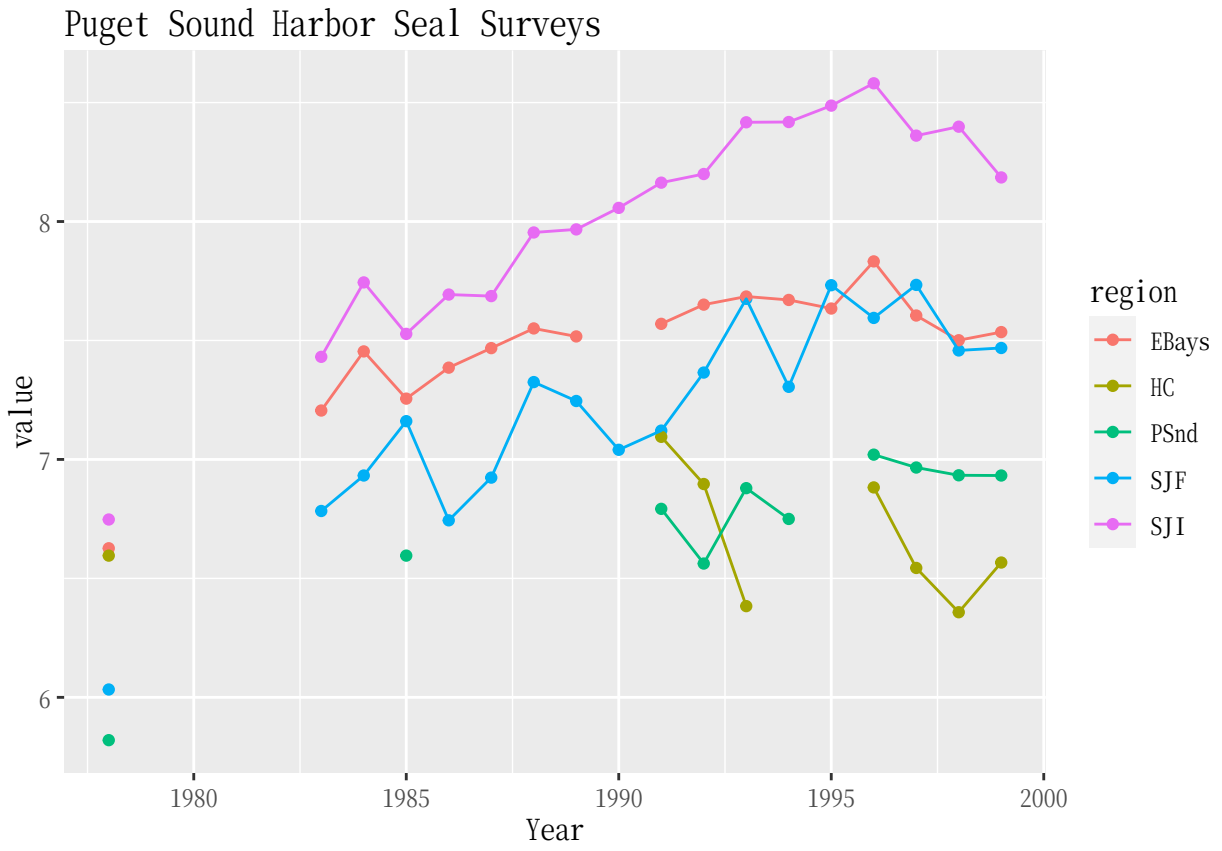
本例子来自 (Holmes et al., 2021) 第 7 章。

28.5.8.1 问题介绍

MARSS 包的例子数据集中，有一个数据集是美国西海岸 5 个采样点历年调查的海豹数目（对数值）。调查年份是 1978-1999，不同采样点的数据缺失情况不同。

```
library(MARSS)
library(tidyverse)
pdat <- as.data.frame(MARSS::harborSealWA) |>
  pivot_longer(
    all_of(c("SJF", "SJI", "EBays", "PSnd", "HC")),
    names_to = "region",
    values_to = "value"  )
ggplot(pdat, aes(
  x = Year, y = value, col = region)) +
  geom_point() +
  geom_line() +
  labs(title = "Puget Sound Harbor Seal Surveys")
```

```
## Warning: Removed 39 rows containing missing values (geom_point).
```



其中 HC 采样点比较独立，而且受到的外界影响与其它采样点区别较大，所以建模时去掉这个采样点。

下面对数据进行整理，最后保留 4 个采样点的逐年次序并转换为每个采样点占数据集一行的格式，这是 MARSS 需要的格式。

```
dat <- as.data.frame(MARSS::harborSealWA)
years <- dat[["Year"]]
d.seal <- dat |>
  select(-Year, -HC) |>
  t()
colnames(d.seal) <- paste0(years)
```

28.5.8.2 单一总体的模型

假设所有的采样点的海豹来自同一个总体，其中的个体可以自由移动。设总体数据以几何速率增长。这时，设总体个数的对数值为 x_t，则

$$x_t = x_{t-1} + u + w_t,\ w_t \sim \mathrm{N}(0, q).$$

其中 u 是年对数增长率。

设四个采样点调查到的海豹数分别占总体数目的一个比例，我们不期望能估计这个比例本身，但是四个采样点观测到

的海豹数的相对比例是可估的。所以设 y_1, y_2, y_3, y_4 是四个采样点观测到的海豹数目，则观测方程为

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}_t = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}_t \tag{28.5}$$

注意数据是对数尺度的，以 y_1 为标准，总体数目 x_t 也是以 y_1 为一个相对标准的，并不会估计出总体的实际数目；然后，对数尺度上，y_2 相对于 y_1 增加了 a_2，y_3 相对于 y_1 增加了 a_3，y_4 相对于 y_1 增加了 a_4。设观测误差阵 R 是对角阵，对角元素不相等。

使用 MARSS 包的规定，模型指定如下：

```
seal.mod.0 <- list(
  B = matrix(1),    # 状态转移矩阵
  U = matrix("u"), # 状态漂移向量
  Q = matrix("q"), # 状态方程误差方差阵
  Z = matrix(1, 4, 1), # 观测方程矩阵
  A = "scaling",    # 观测方程各分量间的相对差值
  R = "diagonal and unequal", # 观测误差方差阵
  x0 = matrix("mu"), # 初值，未知常数
  tinitx = 0) # 模型从 t=0 开始
```

拟合模型：

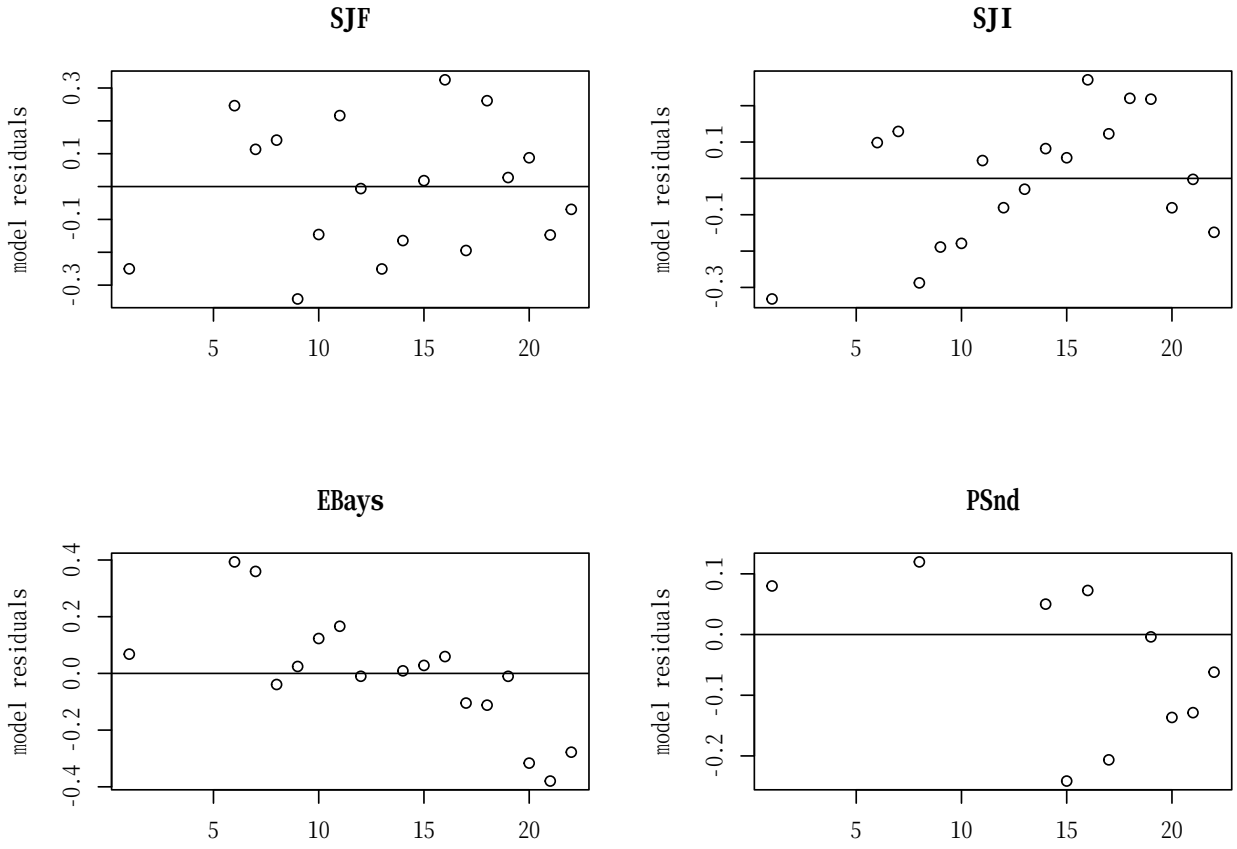
```
seal.fit.0 <- MARSS(d.seal, model = seal.mod.0)
```

```
## Success! abstol and log-log tests passed at 32 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 32 iterations.
## Log-likelihood: 21.62931
## AIC: -23.25863   AICc: -19.02786
##
##                  Estimate
## A.SJI             0.79583
## A.EBays           0.27528
## A.PSnd           -0.54335
## R.(SJF,SJF)       0.02883
## R.(SJI,SJI)       0.03063
## R.(EBays,EBays)   0.01661
## R.(PSnd,PSnd)     0.01168
## U.u               0.05537
```

```
## Q.q              0.00642
## x0.mu            6.22810
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

可以通过拟合残差进行模型诊断。这里的残差是指向前一步预测结果的残差，即 $y_t - \hat{y}_t$，$\hat{y}_t$ 是利用 $1:(t-1)$ 的数据计算的预测值。

```
seal.resi <- function(seal.fit){
  opar <- par(mfrow = c(2, 2))
  on.exit(par(opar))
  resids <- MARSSresiduals(seal.fit, type = "tt1")
  for (i in 1:4) {
    plot(
      resids$model.residuals[i, ],
      ylab = "model residuals",
      xlab = "")
    abline(h = 0)
    title(rownames(d.seal)[i])
  }
}
seal.resi(seal.fit.0)
```



残差应该是没有规律的，但是这个模型的四个调查点的残差的都有一定规律。这说明四个调查点都来自同一个总体的假设不合理。

28.5.8.3 四个独立总体的模型

假设四个采样点的观测分别来自四个总体，且四个总体之间独立，但增长率相同，系统方程方差相等，即：

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{t-1} + \begin{bmatrix} u \\ u \\ u \\ u \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}_t$$

$$\text{其中} \mathbf{w}_t \sim \text{MVN}\left(0, \begin{bmatrix} q & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & q & 0 \\ 0 & 0 & 0 & q \end{bmatrix}\right) \tag{28.6}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_0 = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix}_t$$

这时观测方程不再需要一个位移项 a，变成

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_t + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}_t \tag{28.7}$$

仍设观测误差方差阵是对角阵，方差不相等。

设定 MARSS 模型：

```
seal.mod.1 <- list(
  B = "identity",  # 系统方程矩阵
  U = "equal",     # 系统方程位移项
  Q = "diagonal and equal", # 系统误差方差阵
  Z = "identity",  # 观测方程矩阵
  A = "zero",      # 观测方程位移项
  R = "diagonal and unequal", # 观测误差方差阵
  x0 = "unequal",  # 状态初值
  tinitx = 0)      # 系统从 t=0 计算
```

拟合模型：

```
seal.fit.1 <- MARSS(d.seal, model = seal.mod.1)
```

```
## Success! abstol and log-log tests passed at 24 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 24 iterations.
## Log-likelihood: 23.21635
## AIC: -26.43271   AICc: -22.20194
##
##                 Estimate
## R.(SJF,SJF)      0.02912
## R.(SJI,SJI)      0.00638
## R.(EBays,EBays)  0.00365
## R.(PSnd,PSnd)    0.00974
## U.1              0.05837
## Q.diag           0.00966
## x0.X.SJF         6.13673
## x0.X.SJI         6.73763
## x0.X.EBays       6.58977
## x0.X.PSnd        5.80029
```

```
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

28.5.8.4 四个相关总体的模型

在上面的模型中，假定四个总体的误差项是相关的，设 Q 矩阵形式为

$$\mathbf{Q} = \begin{bmatrix} q & c & c & c \\ c & q & c & c \\ c & c & q & c \\ c & c & c & q \end{bmatrix}.$$

设定模型：

```
seal.mod.2 <- seal.mod.1
seal.mod.2$Q <- "equalvarcov"
```

估计：

```
seal.fit.2 <- MARSS(d.seal, model = seal.mod.2)
```

```
## Success! abstol and log-log tests passed at 31 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 31 iterations.
## Log-likelihood: 34.09079
## AIC: -46.18158   AICc: -41.00511
##
##                 Estimate
## R.(SJF,SJF)      0.02439
## R.(SJI,SJI)      0.00282
## R.(EBays,EBays)  0.00186
## R.(PSnd,PSnd)    0.00797
## U.1              0.05736
## Q.diag           0.01306
## Q.offdiag        0.01049
## x0.X.SJF         6.05525
## x0.X.SJI         6.70630
## x0.X.EBays       6.56580
## x0.X.PSnd        5.75608
## Initial states (x0) defined at t=0
```

```
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

比较三个模型的 AICc：

```
c(seal.fit.0$AICc, seal.fit.1$AICc, seal.fit.2$AICc)
```

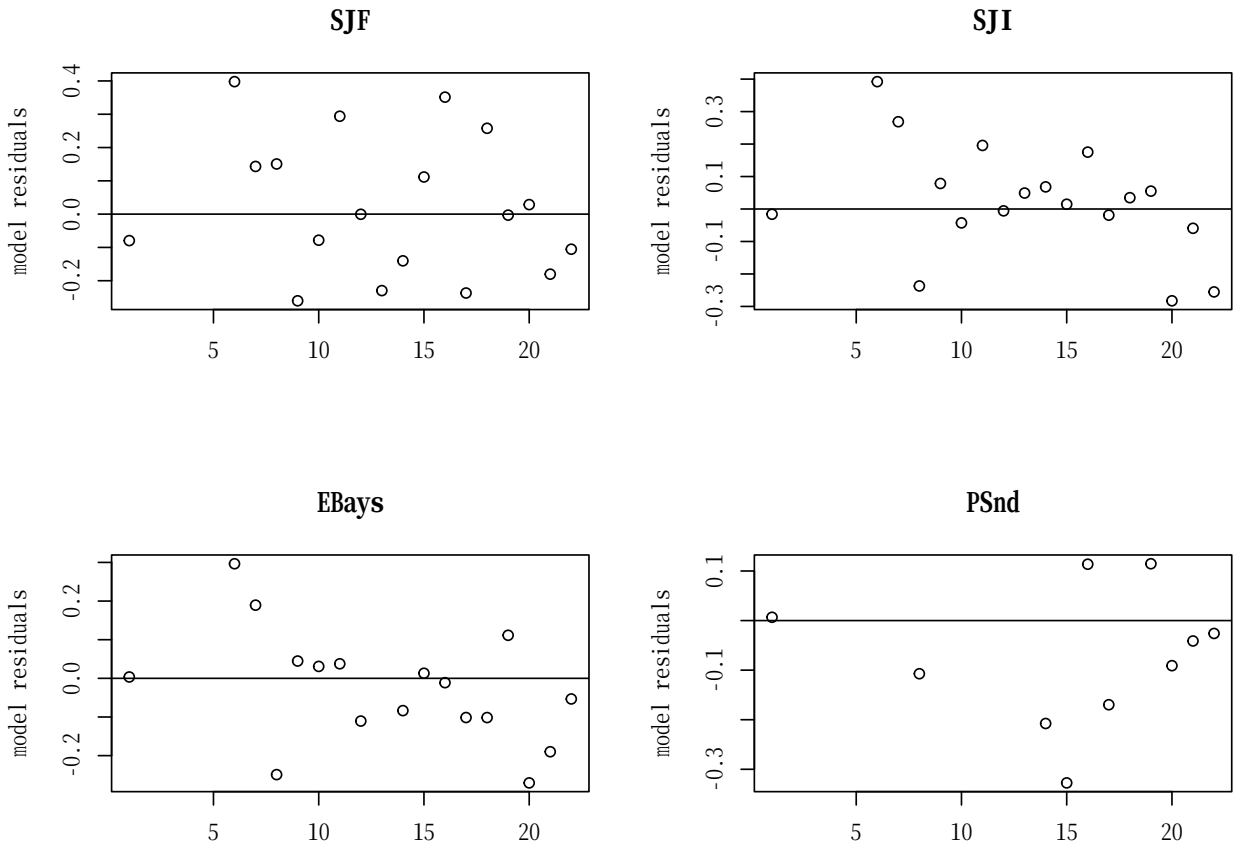
```
## [1] -19.02786 -22.20194 -41.00511
```

可见四个总体且总体之间相关的模型最优。

对最新的这个模型进行残差诊断：

```
seal.resi(seal.fit.2)
```

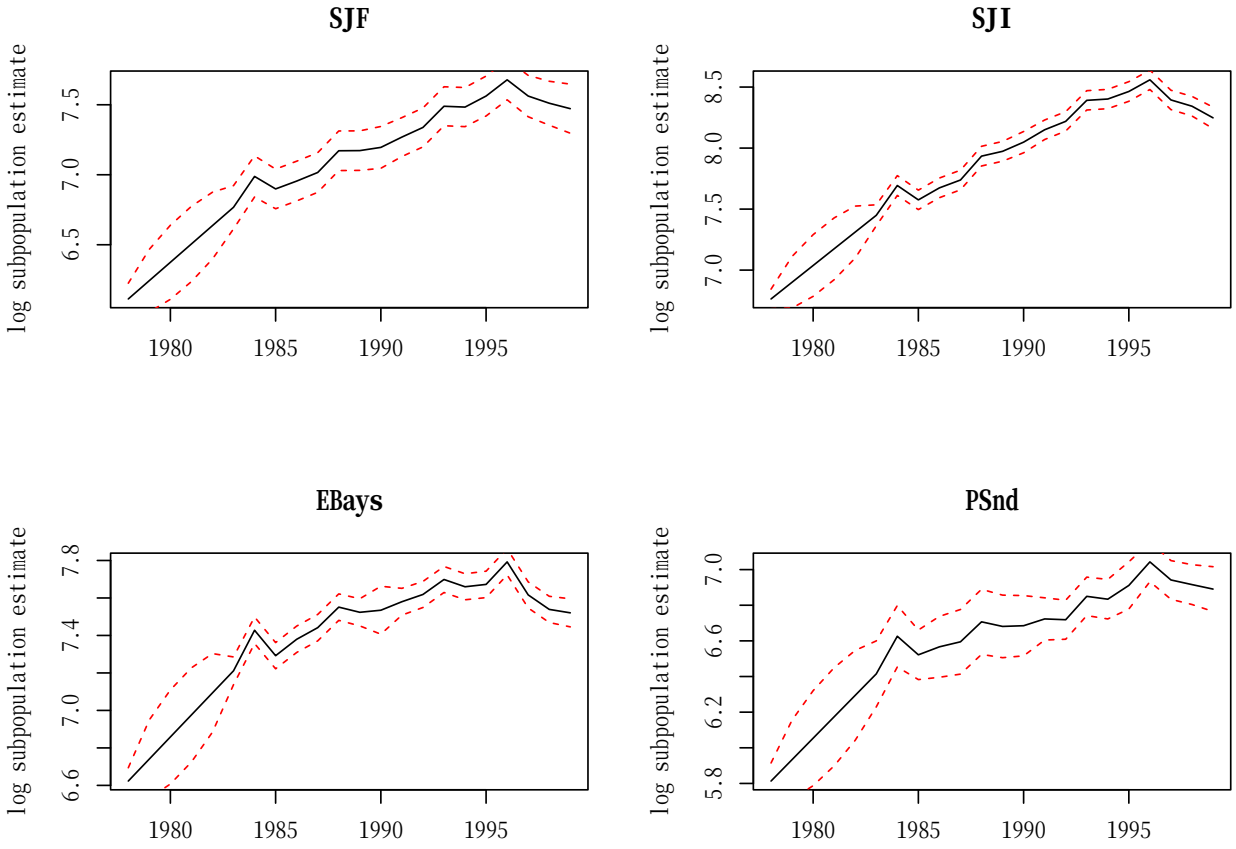
```
## MARSSresiduals.tt1 reported warnings. See msg element of returned residuals object.
```



下面显示四个总体的对数数量（差一个常数位移值）估计值，是平滑结果：

```
seal.sta <- function(seal.fit){
  opar <- par(mfrow = c(2, 2))
  on.exit(par(opar))
  for (i in 1:4) {
```

```
    plot(years,
      seal.fit$states[i, ],
      ylab = "log subpopulation estimate",
      xlab = "", type="l")
    lines(
      years,
      seal.fit$states[i, ]
      - 1.96 * seal.fit$states.se[i, ],
      type = "l",
      lwd = 1, lty = 2, col = "red")
    lines(
      years,
      seal.fit$states[i, ]
      + 1.96 * seal.fit$states.se[i, ],
      type = "l",
      lwd = 1, lty = 2, col = "red")
    title(rownames(d.seal)[i])
  }
}
seal.sta(seal.fit.2)
```



28.5.9 十一地点海豹数目调查建模

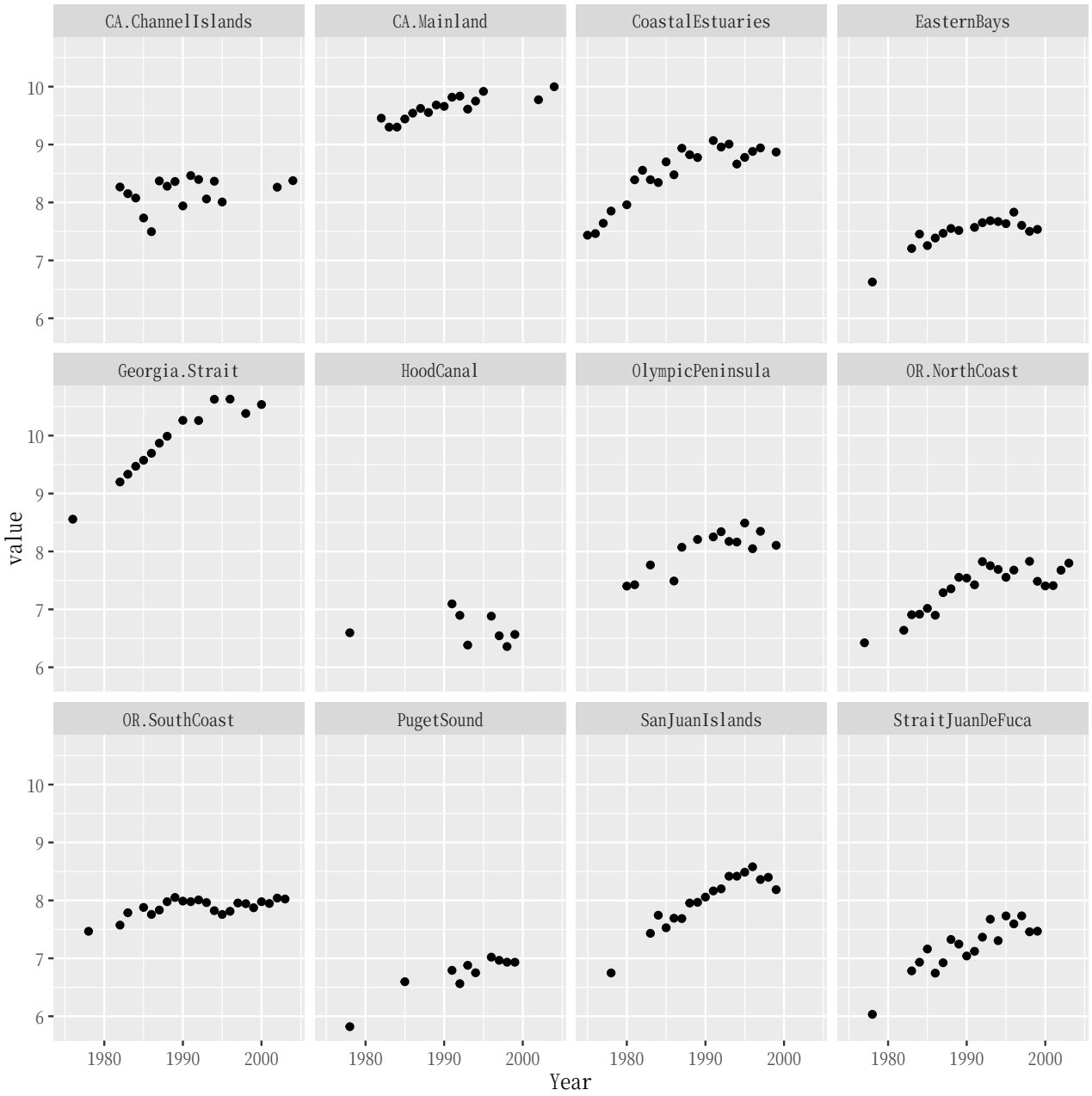
本例子来自 (Holmes et al., 2021) 第 7 章。

28.5.9.1 问题介绍

MARSS 包的 harborSeal 数据集包含了在美国西海岸的 12 个采样点逐年调查的海豹数据，采样年度为 1975-2004，不同地点有不同的缺失年。

```
library(MARSS)
library(tidyverse)
pdat <- as.data.frame(MARSS::harborSeal) |>
  pivot_longer(
    -Year,
    names_to = "region",
    values_to = "value"  )
ggplot(pdat, aes(
  x = Year, y = value)) +
  geom_point() +
  facet_wrap(~region)
```

```
## Warning: Removed 163 rows containing missing values (geom_point).
```



这 12 个采样点中 HoolCanal 的规律与其它采样点差别很大，建模中去掉。

```
dat <- MARSS::harborSeal
years <- dat[, "Year"]
good <- !(colnames(dat) %in% c("Year", "HoodCanal"))
sealData <- t(dat[, good])
colnames(sealData) <- years
```

28.5.9.2 状态方程的不同设置

对海豹总体个数，子总体组成，子总体的误差方差、相关性有不同的模型假设，应在这些模型之间进行比较选择。

- H1: 三个子总体，按采样点的组织结构划分；
- H2: 按内陆采样点和海岸采样点分成两个子总体；
- H3: 按北部和南部分为两个子总体；
- H4: 分为海岸北部，海岸南部，SJF 和佐治亚海峡，Puget Sound 地区共 4 个子总体；
- H5: 仅一个总体；
- H6: 每个调查点单独分属一个子总体。

这时，不同模型对应的观测方程矩阵 Z 的形式分别为：

	$H1$	$H2$	$H4$	$H5$
	pnw ps ca	coast pc	nc is ps sc	pan
Coastal Estuaries	$\begin{bmatrix}1&0&0\end{bmatrix}$	$\begin{bmatrix}1&0\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
Olympic Peninsula	$\begin{bmatrix}1&0&0\end{bmatrix}$	$\begin{bmatrix}1&0\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
Str. Juan de Fuca	$\begin{bmatrix}0&1&0\end{bmatrix}$	$\begin{bmatrix}0&1\end{bmatrix}$	$\begin{bmatrix}0&1&0&0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
San Juan Islands	$\begin{bmatrix}0&1&0\end{bmatrix}$	$\begin{bmatrix}0&1\end{bmatrix}$	$\begin{bmatrix}0&1&0&0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
Eastern Bays	$\begin{bmatrix}0&1&0\end{bmatrix}$	$\begin{bmatrix}0&1\end{bmatrix}$	$\begin{bmatrix}0&0&1&0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
Puget Sound	$\begin{bmatrix}0&1&0\end{bmatrix}$	$\begin{bmatrix}0&1\end{bmatrix}$	$\begin{bmatrix}0&0&1&0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
CA Mainland	$\begin{bmatrix}0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
CA Channel Islands	$\begin{bmatrix}0&0&1\end{bmatrix}$	$\begin{bmatrix}1&0\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
OR North Coast	$\begin{bmatrix}1&0&0\end{bmatrix}$	$\begin{bmatrix}1&0\end{bmatrix}$	$\begin{bmatrix}1&0&0&0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
OR South Coast	$\begin{bmatrix}1&0&0\end{bmatrix}$	$\begin{bmatrix}1&0\end{bmatrix}$	$\begin{bmatrix}0&0&0&1\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$
Georgia Strait	$\begin{bmatrix}0&1&0\end{bmatrix}$	$\begin{bmatrix}0&1\end{bmatrix}$	$\begin{bmatrix}0&1&0&0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$

H6 的 Z 是一个单位阵。

可以先生成一个基本的 MARSS 设置列表，在此基础上修改。

```
hseal.mod <- list(
  B = "identity", # 这对应于各子总体为随机游动
  U = "unequal",  # 系统方程位移项
  Q = "equalvarcov", # 系统方程误差方差阵
  Z = "placeholder", # 观测方程矩阵，待定
  A = "scaling",     # 观测方程各子总体之间的相对差值
  R = "diagonal and equal", # 观测误差方差阵
  x0 = "unequal",    # 状态向量初值
  tinitx = 0         # 系统从 t=0 开始
)
```

下面将六种模型假设的每一个的 Z 矩阵都保存起来：

```
Z.models <- list(
  H1 = factor(c(
    "pnw", "pnw", rep("ps", 4),
    "ca", "ca", "pnw", "pnw", "ps")),
  H2 = factor(c(
    rep("coast", 2), rep("ps", 4),
```

```
    rep("coast", 4), "ps")),
  H3 = factor(c(
    rep("N", 6), "S", "S",
    "N", "S", "N")),
  H4 = factor(c(
    "nc", "nc", "is", "is",
    "ps", "ps", "sc", "sc",
    "nc", "sc", "is")),
  H5 = factor(rep("pan", 11)),
  H6 = factor(1:11) )
names(Z.models) <-  c(
  "stock", "coast+PS", "N+S",
  "NC+strait+PS+SC", "panmictic", "site")
```

28.5.9.3 拟合 6 个模型

这 6 个模型的系统方程误差方差阵都设置成了方差相等、协方差相等的形式。还应比较其它的设置。

下面的程序批量拟合 6 个模型，将结果保存到数据框中：

```
out.tab <- NULL
fits <- list()
for (i in seq_along(Z.models)) {
    hseal.mod$Z <- Z.models[[i]]
    fit <- MARSS::MARSS(
      sealData, model = hseal.mod,
      silent = TRUE,
      control = list(maxit = 1000))
    out <- data.frame(
      H = names(Z.models)[i],
      logLik = fit$logLik,
      AICc = fit$AICc,
      num.param = fit$num.params,
      m = length(unique(Z.models[[i]])),
      num.iter = fit$numIter,
      converged = !fit$convergence)
    out.tab <- rbind(out.tab, out)
    fits <- c(fits, list(fit))
}
```

拟合汇总表格：

```
knitr::kable(out.tab, digits=2)
```

H	logLik	AICc	num.param	m	num.iter	converged
stock	54.97	-72.37	17	3	347	TRUE
coast+PS	52.89	-70.62	16	2	294	TRUE
N+S	58.80	-82.43	16	2	70	TRUE
NC+strait+PS+SC	79.71	-119.40	18	4	433	TRUE
panmictic	39.07	-47.73	14	1	82	TRUE
site	84.86	-111.74	25	11	657	TRUE

如果按 AICc 指标比较，最好的是 H4，分成 4 个子总体。

可以按 AICc 排序，并计算 AICc 差值（即减去最小值），然后用公式 $e^{-\frac{1}{2}\delta}$ 计算 “相对似然”，其中 δ 是 AICc 差值，然后可以计算 AICc 权重，即将相对似然值归一化为总和等于 1。

```
out.tab2 <- out.tab |>
  select(H, AICc) |>
  arrange(AICc) |>
  mutate(
    delta.AICc = AICc - min(AICc),
    rel.like = exp(-0.5*delta.AICc),
    AIC.weight = rel.like / sum(rel.like))
knitr::kable(out.tab2, digits = 3)
```

H	AICc	delta.AICc	rel.like	AIC.weight
NC+strait+PS+SC	-119.395	0.000	1.000	0.979
site	-111.743	7.652	0.022	0.021
N+S	-82.428	36.967	0.000	0.000
stock	-72.371	47.024	0.000	0.000
coast+PS	-70.618	48.777	0.000	0.000
panmictic	-47.730	71.665	0.000	0.000

28.5.10 就业人数回归模型

R 的 longley 数据框包含了与就业有关的宏观经济数据。待完成。

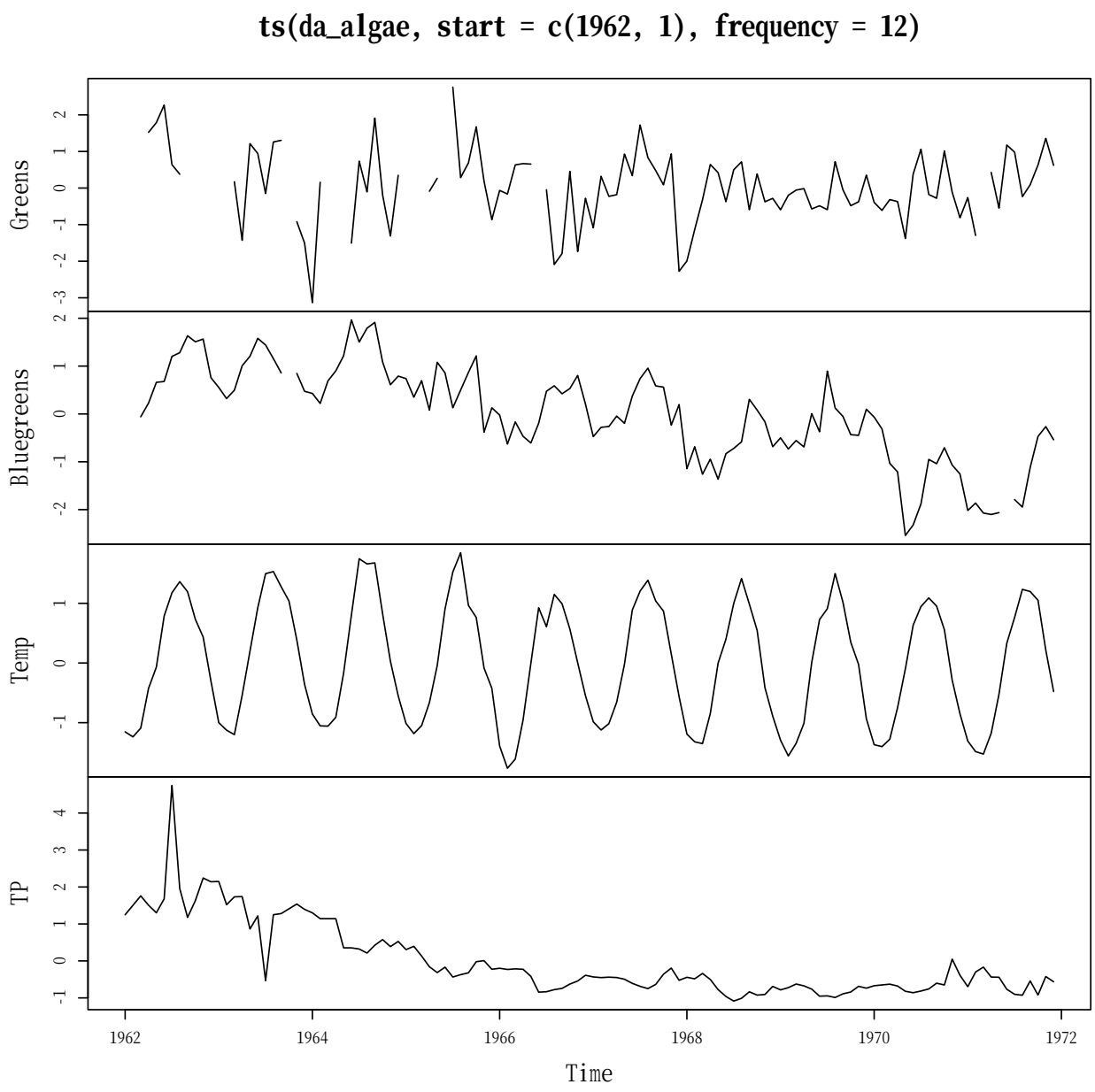
28.5.11 蓝藻和蓝绿藻数据

此例来自 MARSS 手册。用来演示协变量回归。

数据是美国华盛顿湖浮游生物数据，取其中的绿藻 (变量 Greens)、蓝绿藻 (变量 Bluegreens) 以及温度 (变量 Temp)、磷含量 (变量 TP) 数据在 1965-1974 这 10 年的数据。绿藻、蓝绿藻含量数据中有缺失值。用温度、磷含量作为协变量（自变量）。

```
library(MARSS)
sc_fun <- \(da) scale(
  da,
  center=apply(da, 2, \(x) mean(x, na.rm=TRUE)),
```

```
  scale=apply(da, 2, \(x) sd(x, na.rm=TRUE)))

# 蓝藻、蓝绿藻，二维的时间序列数据，
# 每列代表一个时间点，这是 MARSS 要求
dat <- lakeWAplanktonTrans |>
  as.data.frame() |>
  dplyr::filter(Year >= 1965 & Year < 1975) |>
  select(Greens, Bluegreens) |>
  sc_fun() |>
  t()

## 协变量，温度和磷含量
covariates <- lakeWAplanktonTrans |>
  as.data.frame() |>
  dplyr::filter(Year >= 1965 & Year < 1975) |>
  select(Temp, TP) |>
  scale() |>
  t()

da_algae <- cbind(t(dat), t(covariates)) |>
  as.data.frame()

plot(ts(da_algae,
  start=c(1962,1), frequency=12))
```



28.5.11.1 常系数多元线性回归模型

不考虑时间序列性质，仅以蓝藻、蓝绿藻含量为因变量，以温度、磷含量为自变量，作多元回归。这时系统方程的初值、误差方差阵都取为 0 矩阵，使得系统变量 x_t 恒等于 0。

```
alg_mod1 <- list(
  B = "identity",
  U = "zero",
  Q = "zero",
  Z = "identity",
  A = "zero",
  # 在观测方程中的协变量矩阵：
```

```
  d = covariates,
  # 观测方程中的协变量回归系数矩阵:
  D = "unconstrained",
  R = "diagonal and equal",
  x0 = "zero")
alg_fit1 <- MARSS(
  dat, model = alg_mod1)
```

```
## Success! algorithm run for 15 iterations. abstol and log-log tests passed.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Algorithm ran 15 (=minit) iterations and convergence was reached.
## Log-likelihood: -276.4287
## AIC: 562.8573   AICc: 563.1351
##
##                     Estimate
## R.diag                 0.706
## D.(Greens,Temp)        0.367
## D.(Bluegreens,Temp)    0.392
## D.(Greens,TP)          0.058
## D.(Bluegreens,TP)      0.535
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

结果模型为 (因为所有变量都标准化了，所以没有截距项)

$$\begin{pmatrix} \text{Greens}_t \\ \text{BlueGreens}_t \end{pmatrix} = \begin{pmatrix} 0.367 & 0.058 \\ 0.392 & 0.535 \end{pmatrix} \begin{pmatrix} \text{Temp}_t \\ \text{TP}_t \end{pmatrix} + v_t,$$

其中 v_t 的方差阵为对角阵且对角元素都等于 0.706。

直接做多元回归的结果：

```
lm(Greens ~ -1 + Temp + TP, data=da_algae) |>
  summary()
```

```
##
## Call:
## lm(formula = Greens ~ -1 + Temp + TP, data = da_algae)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -2.8951 -0.5329  0.1322  0.5200  2.2228
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## Temp  0.36740    0.09149   4.016 0.000113 ***
## TP    0.05801    0.09756   0.595 0.553378
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9325 on 103 degrees of freedom
##   (15 observations deleted due to missingness)
## Multiple R-squared:  0.1387, Adjusted R-squared:  0.122
## F-statistic: 8.294 on 2 and 103 DF,  p-value: 0.0004572
```

```
lm(Bluegreens ~ -1 + Temp + TP, data=da_algae) |>
  summary()
```

```
##
## Call:
## lm(formula = Bluegreens ~ -1 + Temp + TP, data = da_algae)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.11028 -0.38226  0.06841  0.56525  1.46161
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## Temp  0.39193    0.07096   5.523 2.10e-07 ***
## TP    0.53456    0.07162   7.464 1.74e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7647 on 115 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.4202, Adjusted R-squared:  0.4101
## F-statistic: 41.68 on 2 and 115 DF,  p-value: 2.44e-14
```

28.5.11.2 带有序列自相关误差项的回归

用系统方程表示回归模型中的误差项的相关性，使用两个一元零均值 AR(1) 模型，但无法检查参数估计结果是否平稳。

$$
\begin{aligned}
x_t =& Bx_{t-1} + w_t,\ w_t \sim \mathrm{N}(0, Q), \\
y_t =& D_t d_t + x_t.
\end{aligned}
$$

其中 d_t 是协变量，D_t 是回归系数矩阵，B 是对角阵，对角元素为 AR(1) 模型的自回归系数，观测方程没有观测误差。

```
alg_mod2 <- list(
  B = "diagonal and unequal",
  U = "zero",
  Q = "diagonal and unequal",
  Z = "identity",
  A = "zero",
  # 在观测方程中的协变量矩阵:
  d = covariates,
  # 观测方程中的协变量回归系数矩阵:
  D = "unconstrained",
  R = "diagonal and equal",
  x0 = "zero")
alg_fit2 <- MARSS(
  dat, model = alg_mod2)
```

```
## Success! abstol and log-log tests passed at 72 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 72 iterations.
## Log-likelihood: -209.6348
## AIC: 437.2695   AICc: 438.1186
##
##                               Estimate
## R.diag                          0.0449
## B.(X.Greens,X.Greens)           0.2442
## B.(X.Bluegreens,X.Bluegreens)   0.9183
## Q.(X.Greens,X.Greens)           0.7620
## Q.(X.Bluegreens,X.Bluegreens)   0.1234
## D.(Greens,Temp)                 0.3632
## D.(Bluegreens,Temp)             0.2542
## D.(Greens,TP)                   0.0349
## D.(Bluegreens,TP)               0.0608
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

模型为：

$$\begin{aligned}
\text{Greens}_t &= 0.3632\text{Temp}_t + 0.0349\text{TP}_t + x_{1t},\\
x_{1t} &= 0.2442x_{1,t-1} + w_{1t},\ w_{1t} \sim \text{N}(0, 0.7620);\\
\text{Bluegreens}_t &= 0.2542\text{Temp}_t + 0.0608\text{TP}_t + x_{2t},\\
x_{2t} &= 0.9183x_{2,t-1} + w_{2t},\ w_{2t} \sim \text{N}(0, 0.1234).
\end{aligned}$$

实际操作上，还可以取 B 为单位阵，但这使得回归误差项为单位根过程，会破坏估计的稳定性。实际计算时效果也不好，得到的 Q 矩阵会很小，使得序列相关误差项变成可忽略的小值。

28.5.11.3 将主要模型表示在系统方程中

协变量的作用可以表示在观测方程中，也可以表示在系统方程中，令观测方程没有误差项。

$$x_t = x_{t-1} + Cc_t + w_t,\ w_t \sim \text{N}(0, Q).$$

而 $y_t = x_t$。因变量的模型包括一个随机游动和协变量部分。

```
alg_mod3 <- list(
  B = "identity",
  U = "zero",
  # 在系统方程中的协变量矩阵:
  c = covariates,
  # 系统方程中的协变量回归系数矩阵:
  C = "unconstrained",
  Q = "equalvarcov",
  Z = "identity",
  A = "zero",
  R = "zero")
alg_fit3 <- MARSS(
  dat, model = alg_mod3)
```

```
## Success! algorithm run for 15 iterations. abstol and log-log tests passed.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Algorithm ran 15 (=minit) iterations and convergence was reached.
## Log-likelihood: -285.0732
## AIC: 586.1465   AICc: 586.8225
##
##                      Estimate
## Q.diag                 0.7269
## Q.offdiag             -0.0210
## x0.X.Greens           -0.5189
```

```
## x0.X.Bluegreens        -0.2431
## C.(X.Greens,Temp)      -0.0434
## C.(X.Bluegreens,Temp)   0.0988
## C.(X.Greens,TP)        -0.0589
## C.(X.Bluegreens,TP)     0.0104
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

结果很不合理，且对数似然值变差了许多，这是因为藻类含量用随机游动是不合理的模型。

改用 VAR(1) 模型，但限制系数阵为对角阵：

```
alg_mod3b <- alg_mod3
alg_mod3b$B <- "diagonal and unequal"
alg_fit3b <- MARSS(
  dat, model = alg_mod3b)
```

```
## Success! algorithm run for 15 iterations. abstol and log-log tests passed.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Algorithm ran 15 (=minit) iterations and convergence was reached.
## Log-likelihood: -236.6106
## AIC: 493.2211   AICc: 494.2638
##
##                              Estimate
## B.(X.Greens,X.Greens)          0.1981
## B.(X.Bluegreens,X.Bluegreens)  0.7672
## Q.diag                         0.4899
## Q.offdiag                     -0.0221
## x0.X.Greens                   -1.2915
## x0.X.Bluegreens               -0.4179
## C.(X.Greens,Temp)              0.2844
## C.(X.Bluegreens,Temp)          0.1655
## C.(X.Greens,TP)                0.0332
## C.(X.Bluegreens,TP)            0.1340
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

这个模型结果比较合理，对数似然函数值也比较高，但是初值 x_0 的估计为负值不合理；要求估计从 $t=1$ 开始会出错，这是 EM 算法的固有设置，使用 BFGS 则会对 Q 矩阵有其它要求。

28.5.11.4 系统方程和观测方程都有误差项的模型

在前面的模型中对观测方程增加误差项。

$$
\begin{aligned}
x_t =& Bx_{t-1} + Cc_t + w_t,\ w_t \sim \mathrm{N}(0, Q).\\
y_t =& x_t + v_t,\ v_t \sim \mathrm{N}(0, R).
\end{aligned}
$$

```
alg_mod4 <- list(
  B = "diagonal and unequal",
  U = "zero",
  # 在系统方程中的协变量矩阵:
  c = covariates,
  # 系统方程中的协变量回归系数矩阵:
  C = "unconstrained",
  Q = "diagonal and unequal",
  Z = "identity",
  A = "zero",
  R = "diagonal and unequal")
alg_fit4_em <- MARSS(
  dat, model = alg_mod4,
  control = list(maxit=20))
```

```
## Warning! Reached maxit before parameters converged. Maxit was 20.
##  neither abstol nor log-log convergence tests were passed.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## WARNING: maxit reached at  20  iter before convergence.
##  Neither abstol nor log-log convergence test were passed.
##  The likelihood and params are not at the MLE values.
##  Try setting control$maxit higher.
## Log-likelihood: -208.4015
## AIC: 440.803   AICc: 442.2958
##
##                               Estimate
## R.(Greens,Greens)               0.4674
## R.(Bluegreens,Bluegreens)       0.0413
## B.(X.Greens,X.Greens)           0.3176
## B.(X.Bluegreens,X.Bluegreens)   0.8254
## Q.(X.Greens,X.Greens)           0.3489
## Q.(X.Bluegreens,X.Bluegreens)   0.1246
```

```
## x0.X.Greens                    -0.8270
## x0.X.Bluegreens                -0.3351
## C.(X.Greens,Temp)               0.2262
## C.(X.Bluegreens,Temp)           0.1506
## C.(X.Greens,TP)                 0.0279
## C.(X.Bluegreens,TP)             0.1049
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
##
## Convergence warnings
##  Warning: the  x0.X.Greens  parameter value has not converged.
##  Warning: the  logLik  parameter value has not converged.
##  Type MARSSinfo("convergence") for more info on this warning.
```

```
alg_fit4 <- MARSS(
  dat, model = alg_mod4,
  inits = alg_fit4_em$par,
  method = "BFGS")
```

```
## Success! Converged in 30 iterations.
## Function MARSSkfas used for likelihood calculation.
##
## MARSS fit is
## Estimation method: BFGS
## Estimation converged in 30 iterations.
## Log-likelihood: -207.2771
## AIC: 438.5542   AICc: 440.047
##
##                                Estimate
## R.(Greens,Greens)              3.67e-05
## R.(Bluegreens,Bluegreens)      3.23e-02
## B.(X.Greens,X.Greens)          1.88e-01
## B.(X.Bluegreens,X.Bluegreens)  8.17e-01
## Q.(X.Greens,X.Greens)          8.25e-01
## Q.(X.Bluegreens,X.Bluegreens)  1.36e-01
## x0.X.Greens                   -1.40e+00
## x0.X.Bluegreens               -3.52e-01
## C.(X.Greens,Temp)              2.89e-01
## C.(X.Bluegreens,Temp)          1.53e-01
## C.(X.Greens,TP)                3.43e-02
## C.(X.Bluegreens,TP)            1.09e-01
## Initial states (x0) defined at t=0
```

```
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

观测误差方差较小，没有起太大作用。

28.5.12 Seatbelts 数据建模

R 的 datasets 包的 Seatbelts 数据是一个多元时间序列，为月度数据，保存了英国 1969 年到 1984 年与道路交通事故有关的数据。分量含义：

- DriversKilled: 小汽车交通事故死亡人数。
- drivers: 小汽车交通事故死亡以及重伤人数。
- front: 前排驾乘人员死亡和重伤人数。
- rear: 后排乘客死亡和重伤人数。
- kms: 已行驶里程。
- PetroPrice: 汽油价格。
- VanKilled: 小型厢式载货汽车司机死亡人数。
- law: 强制安全带法律是否已实施的 0-1 变量，从 1983-01-31 起实行。

28.5.12.1 数据预处理

因为取正值的变量建模时比较受限制，所以将这些变量取对数值。

```
library(statespacer)
library(datasets)
da <- as.xts(Seatbelts) |> coredata()
for(v in c("drivers", "front", "rear", "PetrolPrice", "kms"))
  da[, v] <- log(da[,v])
```

28.5.12.2 确定性水平和季节项、有回归自变量的一元模型

以小汽车交通事故死亡、重伤人数（对数值）为因变量。以汽油价格（对数值）和是否实行了安全带法令为自变量。模型为

$$
\begin{aligned}
y_t =& \mu_t + \gamma_t + \beta_1 x_{1t} + \beta_2 x_{2t} + e_t, \\
\mu_{t+1} =& \mu_t, \\
\gamma_{t+1} =& -\sum_{j=1}^{11} \gamma_{t+1-j}.
\end{aligned}
$$

```
y <- cbind(da[,"drivers"])
ssmr1 <- statespacer(
  y = y, # 因变量
  local_level_ind = TRUE, # 是否有局部水平项
  # 设置局部水平项误差方差为零，从而非随机：
  format_level = matrix(0),
  BSM_vec = 12, # 季节项的周期
  # 设置季节项误差项方差为零，从而非随机：
```

```
  format_BSM_list = list(matrix(0)),
  # 添加的外生自变量，每个因变量分量需要输入一个自变量矩阵，
  # 所以即使是一元因变量也需要用列表
  addvar_list = list(as.matrix(
    da[, c("PetrolPrice", "law")])),
  method = "BFGS",
  # 唯一的方差参数 $\sigma_e^2$ 的初值，
  # 算法中使用其对数值的 2 倍以确保取正值
  initial = 0.5 * log(var(c(y))),
  verbose = TRUE)
```

```
## Starting the optimisation procedure at: 2023-05-30 17:04:46
```

```
## initial  value -0.443442
## final  value -0.735372
## converged
```

```
## Finished the optimisation procedure at: 2023-05-30 17:04:46
```

```
## Time difference of 0.058128833770752 secs
```

观测误差方差 σ_e^2 估计：

```
ssmr1$system_matrices$H$H[1,1]
```

```
## [1] 0.007402481
```

非随机的局部水平 μ 的平滑估计：

```
ssmr1$smoothed$level[1,1]
```

```
## [1] 6.401571
```

第一个自变量，即汽油价格对数值，的回归系数：

```
ssmr1$smoothed$addvar_coeff[1, 1]
```

```
## [1] -0.4521301
```

第二个自变量，即强制安全带法令是否实行，的回归系数，和相应标准误差估计，t 统计量值：

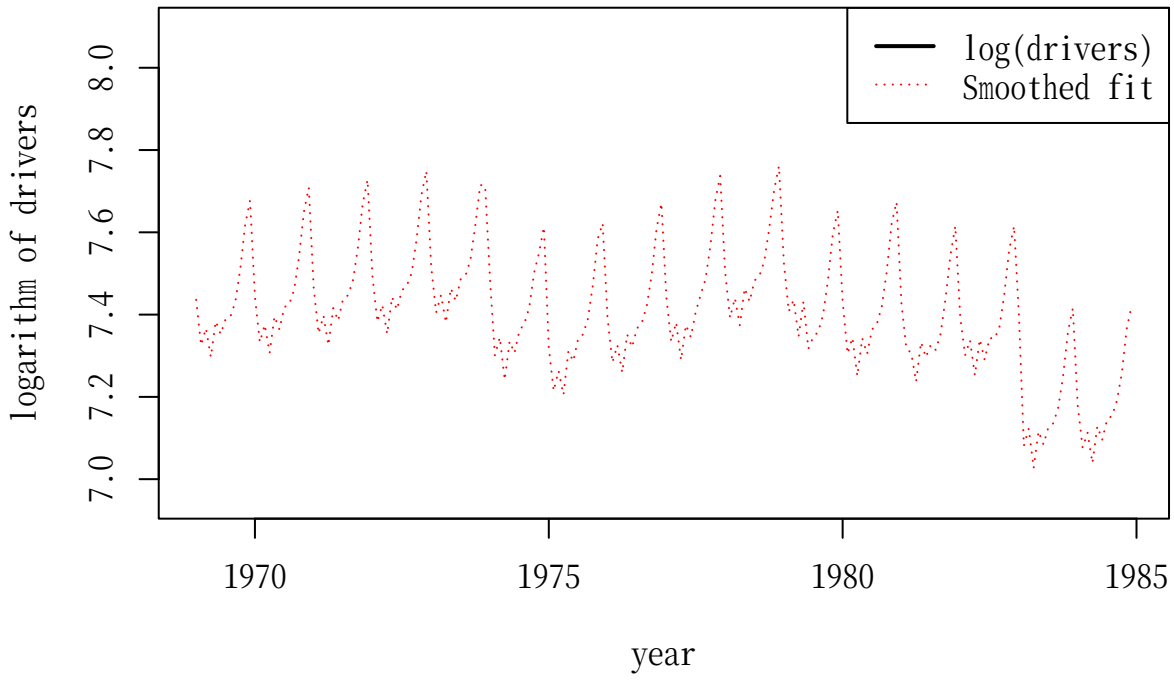
```
c(coef = ssmr1$smoothed$addvar_coeff[1, 2],
  se = ssmr1$smoothed$addvar_coeff_se[1, 2],
  tstat = ssmr1$smoothed$addvar_coeff[1, 2] /
    ssmr1$smoothed$addvar_coeff_se[1, 2])
```

```
##        coef          se       tstat
## -0.19713947  0.02072792 -9.51081922
```

由此计算的法令影响的 t 统计量低于 -2，在 0.05 水平下显著，认为法令生效后水平伤亡水平显著降低。

下面将原始数据与平滑拟合结果（包括局部水平、季节项与自变量作用）作图：

```
plot(Seatbelts[, c("drivers")], type = "l", ylim = c(6.95, 8.1),
     lwd=2,
     xlab = "year", ylab = "logarithm of drivers")
lines(seq(tsp(Seatbelts)[1], tsp(Seatbelts)[2], 1/tsp(Seatbelts)[3]),
      ssmr1$smoothed$level
      + ssmr1$smoothed$BSM12
      + ssmr1$smoothed$addvar,
      type = 'l', col = "red",
      lty=3)
legend("topright",
  c("log(drivers)", "Smoothed fit"),
  lty = c(1,3), lwd=c(2, 1), col = c("black", "red"))
```



28.5.12.3 随机水平和季节项、有回归自变量的一元模型

模型为

$$\begin{aligned} y_t =& \mu_t + \gamma_t + \beta_1 x_{1t} + \beta_2 x_{2t} + e_t, \\ \mu_{t+1} =& \mu_t + \eta_t, \\ \gamma_{t+1} =& -\sum_{j=1}^{11} \gamma_{t+1-j} + \zeta_t. \end{aligned}$$

和确定性的 μ, γ_t 相比，程序只需将零方差改为非零方差：

```
y <- cbind(da[,"drivers"])
ssmr2 <- statespacer(
  y = y, # 因变量
  local_level_ind = TRUE, # 是否有局部水平项
  # 设置局部水平项误差方差为非零，从而随机：
  format_level = matrix(1),
  BSM_vec = 12, # 季节项的周期
  # 设置季节项误差项方差为零，从而随机：
  format_BSM_list = list(matrix(1)),
  # 添加的外生自变量，每个因变量分量需要输入一个自变量矩阵，
  # 所以即使是一元因变量也需要用列表
  addvar_list = list(as.matrix(da[, c("PetrolPrice", "law")])),
  method = "BFGS",
  # 唯一的方差参数 $\sigma_e^2$ 的初值，
  # 算法中使用其对数值的 2 倍以确保取正值
  initial = log(var(as.vector(y))),
  verbose = TRUE)
```

```
## Warning: Number of initial parameters is less than the required amount of
## parameters (3), recycling the initial parameters the required amount of times.
```

```
## Starting the optimisation procedure at: 2023-05-30 17:04:46
```

```
## initial  value -0.172423
## iter  10 value -0.893962
## iter  20 value -0.915493
## iter  30 value -0.915515
## final  value -0.915517
## converged
```

```
## Finished the optimisation procedure at: 2023-05-30 17:04:47
```

```
## Time difference of 0.732856035232544 secs
```

程序中还修改了原来的迭代初值，原来的迭代初值导致迭代过程出错。初值需要三个方差参数对应的初值，这里只给了一个初值，就重复利用。

估计结果：

```
## 观测误差方差估计
ssmr2$system_matrices$H$H[1,1]
```

```
## [1] 0.003786183
```

```
## 局部水平的系统方程的误差方差估计
ssmr2$system_matrices$Q$level[1,1]
```

```
## [1] 0.0002676875
```

```
## 季节项的系统方程的误差方差估计
ssmr2$system_matrices$Q$BSM12[1,1]
```

```
## [1] 1.161438e-06
```

```
## 关于汽油价格对数值这一自变量的回归系数估计
ssmr2$smoothed$addvar_coeff[1, 1]
```

```
## [1] -0.2913944
```

第二个自变量，即强制安全带法令是否实行，的回归系数，和相应标准误差估计，t 统计量值：

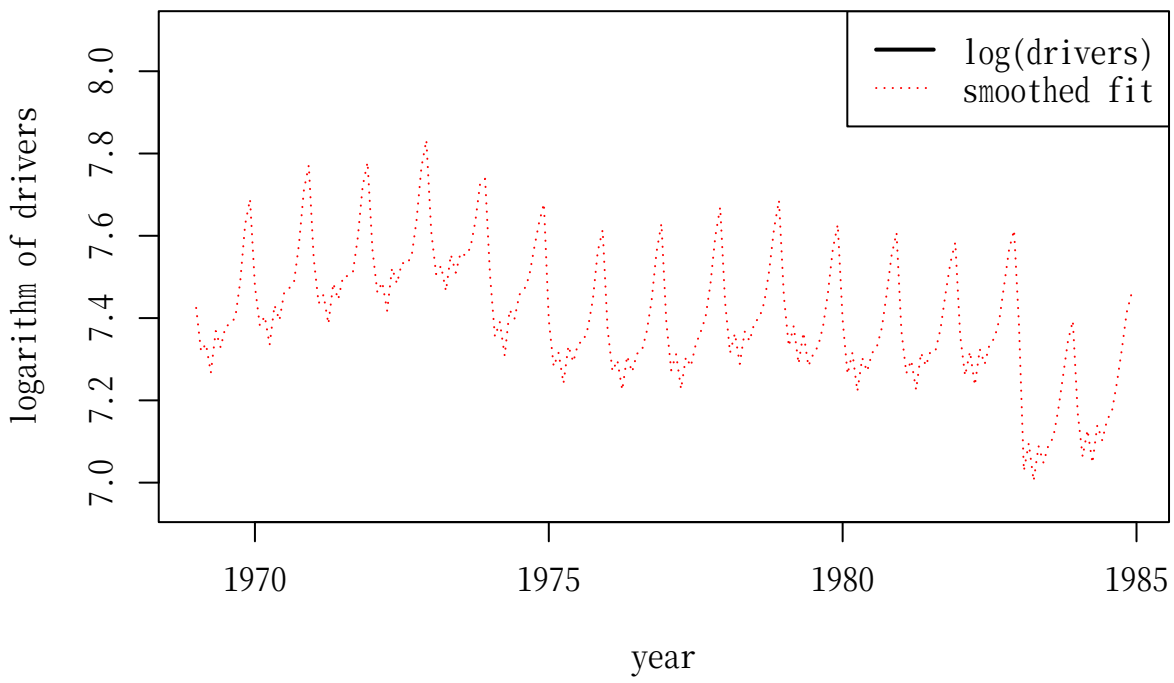
```
c(coef = ssmr2$smoothed$addvar_coeff[1, 2],
  se = ssmr2$smoothed$addvar_coeff_se[1, 2],
  tstat = ssmr2$smoothed$addvar_coeff[1, 2] /
    ssmr2$smoothed$addvar_coeff_se[1, 2])
```

```
##        coef          se       tstat
## -0.23773735  0.04631717 -5.13281209
```

结果表明法令生效后伤亡水平显著降低。

原始数据与拟合结果（拟合结果是平滑结果，利用了平滑的局部水平与自变量线性组合）：

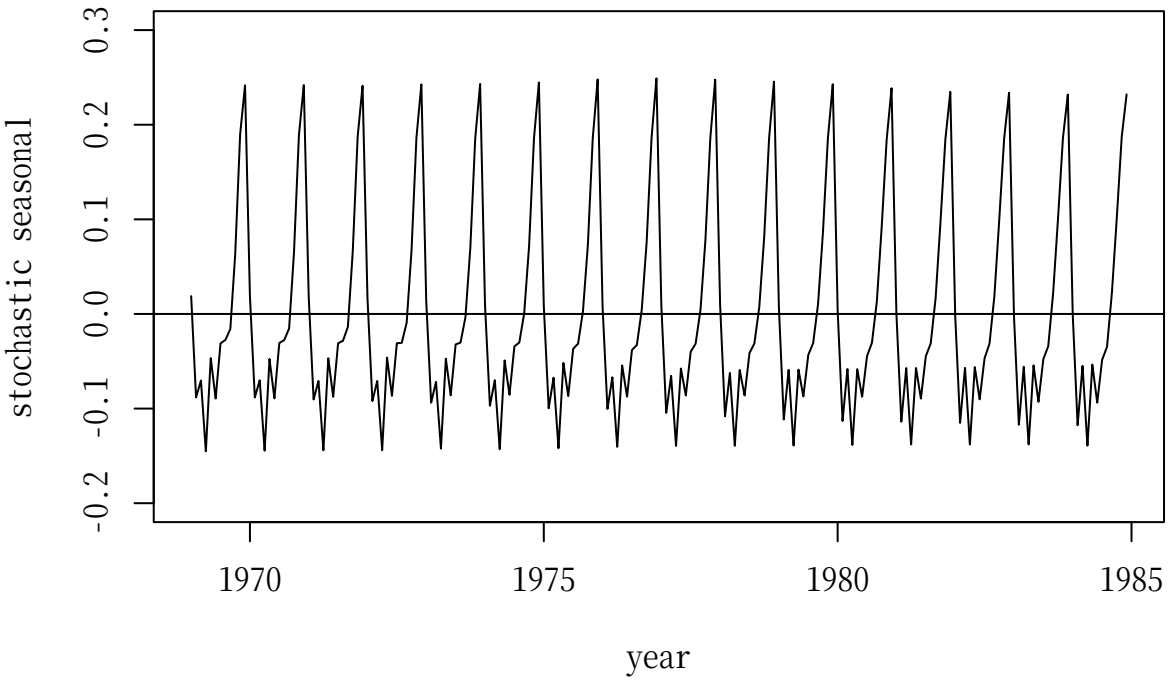
```
plot(Seatbelts[, c("drivers")], type = "l",
     ylim = c(6.95, 8.1),
     lwd=2,
     xlab = "year", ylab = "logarithm of drivers")
lines(seq(tsp(Seatbelts)[1], tsp(Seatbelts)[2], 1/tsp(Seatbelts)[3]),
      ssmr2$smoothed$level
      + ssmr2$smoothed$BSM12
      + ssmr2$smoothed$addvar,
      lwd=1, lty=3,
      type = 'l', col = "red")
legend("topright",
  c("log(drivers)", "smoothed fit"),
  lty = c(1,3), lwd=c(2, 1), col = c("black", "red"))
```



与非随机的水平和季节项的拟合结果比较，随机的水平和季节项使得拟合结果更贴近原始数据。

实际上，从季节项的系统方程的误差方差 10^{-6} 级别可以看出季节项的随机性很小，下面作平滑估计的季节项图形：

```
plot(seq(tsp(Seatbelts)[1], tsp(Seatbelts)[2], 1/tsp(Seatbelts)[3]),
     ssmr2$smoothed$BSM12,
     type = "l", ylim = c(-0.2, 0.3),
     xlab = "year", ylab = "stochastic seasonal")
abline(h = 0)
```



我们将随机的季节项改为非随机：

```
y <- cbind(da[,"drivers"])
ssmr3 <- statespacer(
  y = y, # 因变量
  local_level_ind = TRUE, # 是否有局部水平项
  # 设置局部水平项误差方差为非零，从而随机：
  format_level = matrix(1),
  BSM_vec = 12, # 季节项的周期
  # 设置季节项误差项方差为零，从而非随机：
  format_BSM_list = list(matrix(0)),
  # 添加的外生自变量，每个因变量分量需要输入一个自变量矩阵，
  # 所以即使是一元因变量也需要用列表
  addvar_list = list(as.matrix(
    da[, c("PetrolPrice", "law")])),
  method = "BFGS",
  # 唯一的方差参数 $\sigma_e^2$ 的初值，
  # 算法中使用其对数值的 2 倍以确保取正值
  initial = log(var(c(y))),
  verbose = TRUE)
```

```
## Warning: Number of initial parameters is less than the required amount of
## parameters (2), recycling the initial parameters the required amount of times.
```

```
## Starting the optimisation procedure at: 2023-05-30 17:04:48

## initial  value -0.577129
## iter  10 value -0.912856
## final  value -0.912859
## converged

## Finished the optimisation procedure at: 2023-05-30 17:04:48

## Time difference of 0.21986198425293 secs
```

估计结果：

```
## 观测误差方差估计
ssmr3$system_matrices$H$H[1,1]
```

```
## [1] 0.004033516
```

```
## 局部水平的系统方程的误差方差估计
ssmr3$system_matrices$Q$level[1,1]
```

```
## [1] 0.0002681651
```

```
## 关于汽油价格对数值这一自变量的回归系数估计
ssmr3$smoothed$addvar_coeff[1, 1]
```

```
## [1] -0.2767301
```

第二个自变量，即强制安全带法令是否实行，的回归系数，和相应标准误差估计，t 统计量值：

```
c(coef = ssmr3$smoothed$addvar_coeff[1, 2],
  se = ssmr3$smoothed$addvar_coeff_se[1, 2],
  tstat = ssmr3$smoothed$addvar_coeff[1, 2] /
    ssmr3$smoothed$addvar_coeff_se[1, 2])
```

```
##       coef         se      tstat
## -0.2375904  0.0464483 -5.1151578
```

结果表明法令生效后伤亡水平显著降低。

比较三个模型的 AIC：

```
c(ssmr1$diagnostics$AIC,
  ssmr2$diagnostics$AIC,
  ssmr3$diagnostics$AIC)
```

```
## [1] -1.314494 -1.653950 -1.659052
```

第三个模型的 AIC 最小。

28.5.12.4 前后座驾乘人员同时建模的二元模型

考虑前座驾乘人员和后座乘客死亡、重伤人数对数的二元模型，都以汽油价格对数值、行驶里程对数值和安全带法令为自变量。

```
y <- da[,c("front", "rear")]
Xmat <- da[,c("PetrolPrice", "kms", "law")]
ssmr4 <- statespacer(y = y, # 输入数据是两列矩阵，二元时间序列
  # 观测方程误差项方差阵格式，这里是无限制的二阶方差阵
  H_format = matrix(1,2,2),
  local_level_ind = TRUE,
  # 局部水平的系统方程的误差项方差设置，无限制：
  format_level = matrix(1, 2, 2),
  BSM_vec = 12,
  # 季节项的系统方程的误差项方差设置，设置为零，表示非随机季节项
  format_BSM_list = list(matrix(0, 2, 2)),
  # 每个因变量所需的自变量矩阵：
  addvar_list = list(Xmat, Xmat),
  method = "BFGS",
  initial = 0.5 * log(diag(var(y))),
  verbose = TRUE)
```

```
## Warning: Number of initial parameters is less than the required amount of
## parameters (6), recycling the initial parameters the required amount of times.

## Starting the optimisation procedure at: 2023-05-30 17:04:48

## initial  value 0.389588
## iter  10 value -1.583525
## iter  20 value -1.676129
## iter  30 value -1.676530
## final  value -1.676537
## converged

## Finished the optimisation procedure at: 2023-05-30 17:05:00

## Time difference of 11.3847918510437 secs
```

观测方程误差方差阵估计：

```
ssmr4$system_matrices$H$H
```

```
##             [,1]        [,2]
## [1,] 0.005402166 0.004449533
## [2,] 0.004449533 0.008566858
```

局部水平的系统方程的误差方差阵估计：

```
ssmr4$system_matrices$Q$level
```

```
##              [,1]         [,2]
## [1,] 0.0002556392 0.0002247045
## [2,] 0.0002247045 0.0002319561
```

6 个回归系数及相应的 t 统计量：

```
data.frame(
  x1 = c("front PetrolPrice", "front kms", "front law",
    "rear PetrolPrice", "rear kms", "rear law"),
  x2 = ssmr4$smoothed$addvar_coeff[1,],
  x3 = ssmr4$smoothed$addvar_coeff[1,]
  / ssmr4$smoothed$addvar_coeff_se[1,]) |>
  knitr::kable(
    col.names=c("", "Coef", "t stat"),
    digits=4  )
```

	Coef	t stat
front PetrolPrice	-0.3076	-2.8986
front kms	0.1518	1.1678
front law	-0.3370	-6.8466
rear PetrolPrice	-0.0858	-0.7639
rear kms	0.5504	3.7927
rear law	0.0009	0.0171

与常识一致的是，后排乘客的伤亡数并不受安全带法令的影响。汽油价格上升对前排驾乘人员伤亡数有抑制作用。行驶里程数对前排驾乘人员伤亡数没有显著影响，但对后排乘客伤亡有正向的影响。

程序中还可以限制某个方差阵的秩，比如取 `format_level` 为

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

可以限制水平的系统方差的误差方差阵（2×2 方差阵）为秩等于 1。

附录 A

测试

这一附录存在说明本讲义仍处于草稿状态！

$$abc$$

符号：§

Bibliography

Alizadeh, S., Brandt, M., & Diebold, F. X. (2002). Range-based estimation of stochastic volatility models. *J. Finance*, *57*, 1047–1092.

Andersen, T., Bollerslev, T., Diebold, F. X., & Ebens, H. (2001). The distribution of realized stock return volatility. *J. Financ. Econ.*, *96*, 42–55.

Beijers, D. (2020). *statespacer: State space modelling in r* [R package version 0.4.0]. https://dylanb95.github.io/statespacer/

Bollerslev, T. (1986). Generalized autoregressive conditional heterskedastici. *Journal of Econom.*, *31*, 307–327.

Box, G. (1976). Science and statistics. *Journal of American Statistician Association*, 33526–536.

Box, G., & Pierce, D. (1970). Distribution of residual autocorelations in autoregressive-integrated moving average time series models. *J. of American Stat. Assoc.*, *65*, 1509–1526.

Brockwell, P. J., & Davis, R. A. (1991). *Time series: Theory and methods* (2nd). Springer-Verlag.

Christoffersen, P. F. (2003). *Elements of financial risk management*. Elsevier Science (USA).

Cochrane, D., & Orcutt, G. H. (1949). Application of least squares regression to relationships containing auto-correlated error terms. *Journal of the American Statistical Association*, *44*, 32–61.

Cryer, J. D., & Chan, K.-S. (2008). *Time series analysis with applications in r* (2nd Ed.). Springer.

Ding, Z., Granger, C. W. J., & Engle, R. F. (1993). A long memory property of stock returns and a new model. *Journal of Empirical Finance*, *1*, 83–106.

Duan, J. (1995). The garch option pricing model. *Mathematical Finance*, *5*, 13–32.

Durbin, J., & Koopman, S. J. (2012). *Time series analysis by state space methods* (2nd). Oxford University Press.

Durbin, J., & Watson, G. S. (1950). Testing for serial correlation in least squares regression, i. *Biometrika*, *37*, 409–428.

Durbin, J., & Watson, G. S. (1951). Testing for serial correlation in least squares regression, ii. *Biometrika*, *38*, 159–178.

Engle, R. R., & Granger, C. W. J. (1987). Cointegration and error correction representation, estimation and testing. *Econometrica*, *55*, 251–276.

Engle, R. (1982). Autoregressive conditional heteroscedasticity with estmates of the varinance of united kingdom inflation. *Econometrica*, *50*, 987–1007.

Fernandez, C., & Steel, M. F. J. (1998). On bayesian modeling of fat tails and skewness. *Journal of the American Statistical Association*, *93*(441), 359–371.

French, K. R., Schwert, G. W., & Stambaugh, R. F. (1987). Expected stock returns and volatility. *J. Financ. Econ.*, *19*, 3–29.

Garman, M. B., & Klass, M. J. (1980). On the estimation of security price volatilites from historical data. *J. Buss.*, *53*, 67–78.

Gebhard Kirchgässner, U. H., Jürgen Wolters. (2013). *Introduction to modern time series analysis*. Springer.

Geweke, J., & Porter-Hudak, S. (1983). The estimation and application of long memory time series models. *Journal of Time Series Analysis*, *4*(4), 221–238.

Glosten, L. R., Jagannathan, R., & Runkle, D. E. (1993). On the relation between the expected value and the volatility of nominal excess return on stocks. *J. Finance*, *48*, 1779–1801.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, (37), 424–438.

Granger, C. W., & Newbold, P. (1974). Spurious regressions in econometrics. *Journal of Econometrics*, *2*, 111–120.

Holmes, E. E., Scheuerell, M. D., & Ward, E. J. (2021). *Applied time series analysis for fisheries and environmental sciences*. https://atsa-es.github.io/atsa-labs/

Johansen, S. (1988). Analysis of cointegration vectors. *Journal of Economic Dynamics and Control*, *12*, 231–254.

Johansen, S. (1995). *Likelihood based inference in cointegrated vector error correction models*. Oxford University Press, Oxford, UK.

Johnson, R. A., & Wichern, D. W. (1998). *Applied multivariate statistical analysis* (4th Edition). Prentice Hall.

Ljung, G., & Box, G. (1978). On a measure of lack of fit in time series models. *Biometrika*, *66*, 67–72.

Lutkepohl, H., & Kratzig, M. (2004). *Applied time series econometrics*. Cambridge University Press.

Markovitz, H. (1959). *Portfolio selection: Efficient diversification of investments*. John Wiley & Sons, New York.

Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: A new approach. *Econometrica*, *59*, 347–370.

Parkinson, M. (1980). The extreme value method for estimating the variance of the rate of return. *J. Buss.*, *53*, 61–65.

Pfaff, B. (2008). *Analysis of integrated and cointegrated time series with r* (2nd ed.). Springer.

Phillips, P. C. B., & Ouliaris, S. (1990). Asymptotic properties of residual based tests for cointegration. *Econometrica*, *58*, 165–193.

Schwert, G. W. (1979). Tests of causality: The message in the innovations. In K. Brunner & A. Meltzer (Eds.), *Three aspects of policy and policymaking: Knowledge, data, and institutions, carnegie-rochester conference series on public policy, band 10* (pp. 55–96). North-Holland, Amsterdam.

Shumway, R. H., & Stoffer, D. S. (2017). *Time series analysis and its applications: With r examples* (4th). Springer International Publishing. https://doi.org/10.1007/978-3-319-52452-8

Shumway, R. H., & Stoffer, D. S. (2019). *Time series: A data analysis approach using r*. Taylor; Francis Group, LLC.

Sims, C. A. (1972). Money, income, and causality. *American Economic Review*, *62*, 540–552.

Tiao, G. C., & Box, G. E. P. (1981). Modeling multiple time series with applications. *Journal of the Americal Statistical Association*, *76*, 802–816.

Tiao, G., & Tsay, R. (1994). Some advances in non-linear and adaptive modeling in time series. *Journal of Forecasting*, 13109–131.

Tsay, R. S. (2010). *Analysis of financial time series* (3rd Ed.). John Wiley & Sons, Inc.

Tsay, R. S. (2013). 金融数据分析导论：基于 *r* 语言. 机械工业出版社.

Tsay, R. S. (2014). *Multivariate time series analysis with r and financial applications*. John Wiley & Sons, Inc.

West, M., & Harrison, J. (1997). *Bayesian forecasting and dynamic models*. Springer-Verlag. https://doi.org/10.1007/b98971

Yang, D., & Zhang, Q. (2000). Drift-independent volatility estimation based on high low open and close prices. *J. Buss.*, *73*, 477–491.

Zakoian, J. M. (1994). Threshold heteroscedastic models. *J Econ Dyn Control*, *18*, 931–955.

何书元. (2003). 应用时间序列分析. 北京大学出版社.

吴喜之 & 刘苗. (2018). 应用时间序列分析——*r* 软件陪同 (第二版). 机械工业出版社.

武东 & 汤银才. (2007). 稳定分布及其在金融中的应用. 应用概率统计.

谢衷洁. (1990). 时间序列分析. 北京大学出版社.