「医学统计学及 SAS 应用」之

R 语言实现

update：2014-1-7

PREFACE

关于这份笔记，有几点需要说明：

0. 客套话。感谢您阅读本文，希望能与您共同进步。

1. 为什么是 R。以前从来没想过用盗版 SPSS 和 SAS 有什么问题，直到到了国外，敏感的版权形势下，R 是最好的选择。

2. 本文所引用数据的版权，归《医学统计学及 SAS 应用》原作者所有，本文只探讨使用 R 实现的可能性。此处还想特别感谢该书的编者之一、上海交通大学医学院统计学教研室的王筱金老师，正是得益于她的用心教学，我才学会基本的统计学知识，也因此才有这份笔记。

3. 关于本文的权利。R 语言的世界里，倡导的精神即是自由与分享，所以您可以自由地阅读、拷贝、使用和修改本文的代码。若您进行转载，请注明出处。

4. 本文的作者学临床医学出身，并非统计专业，对统计知识也是处在一知半解之中，故文中的表述或程序可能存在各种错误和不足，还请读者注意辨别，切勿被作者误导，同时希望您不吝指正。

5. 有任何问题或指正，请随时联系作者邮箱：yefux@med.nagoya-u.ac.jp

YE, FUXIANG

于日本·名古屋

目 录

例 3.19 1
例 3.20 2
例 3.21 2
例 3.22 3
例 3.23 3
例 4.13 4
例 4.14 5
例 4.15 5
例 4.16 5
例 4.17 6
例 4.18 6
例 5.1 7
例 5.2 8
例 5.3 10
例 5.4 12
例 5.5 13
例 5.6 14
例 5.7 15
例 6.1 16
例 6.2 18
例 6.3 19
例 6.4 21
例 6.5 22
例 6.6 23
例 7.1 24
例 7.2 25
例 7.3 27
例 7.4 29
例 8.6（程序 8.1） 30
例 8.7（程序 8.2） 31
例 8.8（程序 8.3） 31
例 8.9（程序 8.4） 32
例 8.10（程序 8.5） 33
例 8.11（程序 8.6） 34
例 8.12（程序 8.7） 35
例 9.1 35
例 9.2 36
例 9.4（程序 9.3） 36
例 9.5（程序 9.4） 37
例 9.6（程序 9.5） 37
例 9.7（程序 9.6） 38
例 9.8（程序 9.7） 38
例 9.9（程序 9.8） 39
例 9.11（程序 9.9） 39
例 9.12（程序 9.10） 40
例 9.13（程序 9.11） 41
例 9.14（程序 9.12） 42
例 9.15（程序 9.13） 43
例 9.16（程序 9.14） 45
例 9.17（程序 9.15） 45
例 9.18（程序 9.16） 46
例 9.19（程序 9.17） 46
例 10.1-2 47
例 10.3 48
例 11.1 49
例 11.2 50
例 11.4（程序 11.3） 51
例 11.5（程序 11.4A） 52
例 11.5（程序 11.4B、C） 53
例 11.6（程序 11.5） 54
例 11.7（程序 11.6A） 55

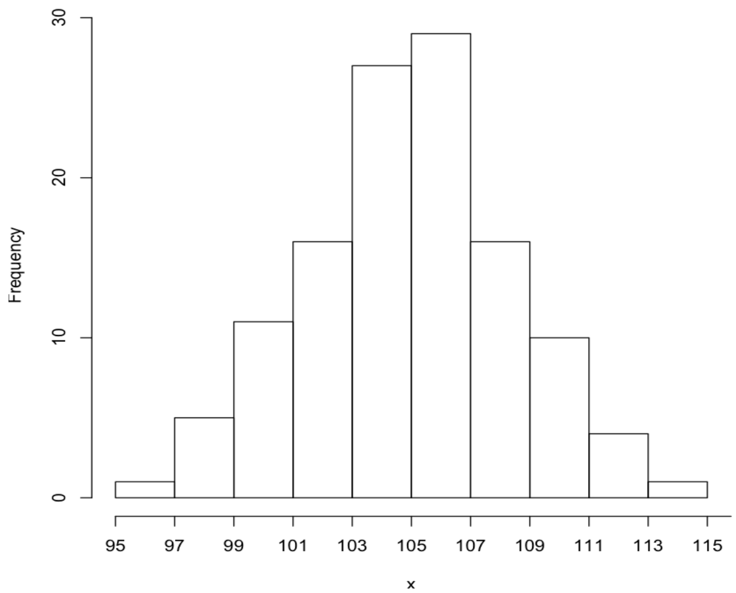
例 11.7（程序 11.6B） 56
例 12.1 57
例 12.3（程序 12.2） 59
例 12.4（程序 12.3） 60
例 12.5（程序 12.4） 62
例 13.2 63
例 13.3 65
例 13.4 66
例 15.3（程序 15_1） 69
例 15.4（程序 15_2） 69
例 15.5（程序 15_3） 70
例 15.6（程序 15_4） 71
例 15.7（程序 15_5） 72
例 15.8（程序 15_6） 73
例 15.9（程序 15_7） 74

例 3.19

：直方图

```
x <- c(108.0, 97.6, 103.4, 101.6, 104.4, 98.5, 110.5, 103.8, 109.7, 109.8,
       104.5, 99.5, 104.0, 103.9, 97.2, 106.3, 106.2, 107.6, 108.3, 97.6,
       102.7, 103.7, 107.6, 103.2, 103.6, 103.3, 102.8, 102.3, 102.2, 103.3,
       101.2, 107.5, 106.3, 109.7, 99.5, 107.4, 103.4, 106.6, 105.7, 107.4,
       103.0, 109.6, 106.4, 107.3, 100.6, 112.3, 100.5, 101.9, 98.8, 99.7,
       104.3, 110.2, 105.3, 95.2, 105.8, 105.2, 106.1, 103.6, 106.6, 105.1,
       105.5, 113.5, 107.7, 106.8, 106.2, 109.8, 99.7, 107.9, 104.8, 103.9,
       106.8, 106.4, 108.3, 106.5, 103.3, 107.7, 106.2, 100.4, 102.6, 102.1,
       110.6, 112.2, 110.2, 103.7, 102.3, 112.1, 105.4, 104.2, 105.7, 104.4,
       102.8, 107.8, 102.5, 102.3, 105.8, 103.7, 103.1, 101.6, 106.5, 100.0,
       103.2, 109.3, 105.8, 106.1, 104.9, 105.9, 105.3, 103.7, 99.6, 106.2,
       102.5, 108.1, 106.1, 108.3, 99.8, 108.3, 104.0, 100.6, 112.6, 103.7)
hist(x, breaks=seq(95, 115, 2), xaxt="n", yaxt="n")
    # breaks=语句用于划分直方图单元格区间
    # seq(95,115,2) 表示构建一组从 95 到 115，公差为 2 的等差数列
    # xaxt="n", yaxt="n" 表示不绘制坐标轴
axis(side= 1, at= seq(95, 117, 2))
axis(side= 2, at= seq(0, 30, 10))
    # 此 2 句代码为 R 语言的低级绘图命令，用于人工控制图像
    # axis() 函数用于生成坐标轴，side=1 为横坐标，side=2 为纵坐标
    # 参数 at 定义了坐标轴的数值，一般用向量表示
```

结果：



例 3.20

：统计描述

```
# 数据见例 3.19
mean(x)
sd(x)
cv <- sd(x)/mean(x)*100
      # 计算变异系数
min(x)
max(x)
```

结果：

```
mean (x)
[1] 104.8858
sd (x)
[1] 3.536349
cv=100*sd(x)/mean(x)
cv
[1] 3.371617
min(x)
[1] 95.2
max(x)
[1] 113.5
```

例 3.21

：分组统计描述

```
x <- c(1, 1.8, 1.4, 2, 1.7, 1.1, 1, 2.2, 1.5, 3, 1.9, 1.2, 2, 2.5, 1.0, 1, 2.7, 1.6, 2, 2.3, 1.3, 2,
       2.8, 0.9, 3, 3.0, 1.1, 1, 2.6, 1.4, 1, 2.4, 1.2, 2, 1.9, 1.3, 3, 2.9, 0.8, 1, 3.2, 1.7, 3,
       3.1, 1.5, 2, 2.6, 1.9, 3, 3.5, 1.6, 3, 3.3, 1.5)
group <- x[c(seq(1, length(x), 3))]
VA <- x[c(seq(2, length(x), 3))]
VB1 <- x[c(seq(3, length(x), 3))]
xframe <- data.frame(group=group, VA=VA, VB1= VB1)
       # 建立 group, VA, VB1 的三列 csv 文件，导入后较为简便
       # 最后一句为建立共有三列的数据框
# 安装 doBy package。更详细说明使用 ?doBy 查询
library(doBy)
summaryBy(VA+ VB1~ group, data= xframe, FUN= c(mean, sd, max, min))
       # 以 group 为分组变量，FUN=c()指出所求参数。
       # 更详细说明使用 ?summaryBy 查询
```

结果：

```
  group  VA.mean  VB1.mean    VA.sd    VB1.sd  VA.max  VB1.max  VA.min  VB1.min
1     1 2.483333  1.466667  0.4750439  0.1751190    3.2     1.7     1.8     1.2
```

```
2    2  2.300000  1.250000  0.4242641  0.3563706    2.8     1.9     1.7     0.9
3    3  2.950000  1.283333  0.5576737  0.3060501    3.5     1.6     1.9     0.8
```

例 3.22

：以频数计算均值

```
x <- c(48, 16, 32, 64, 128, 256, 512)
freq <- c(1, 3, 8, 13, 21, 9, 4, 1)
y <- log10(x)
meany <- sum(y*freq)/ sum(freq)
  # 用 2 个向量计算带频数的均值
meanx <- 10^meany
```

结果：

```
meany
[1] 1.695802
meanx
[1] 49.63663
```

例 3.23

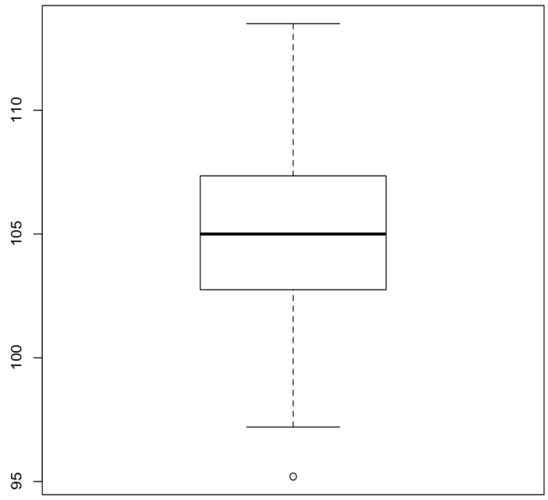
：频数表，正态性检验，茎叶图，箱线图

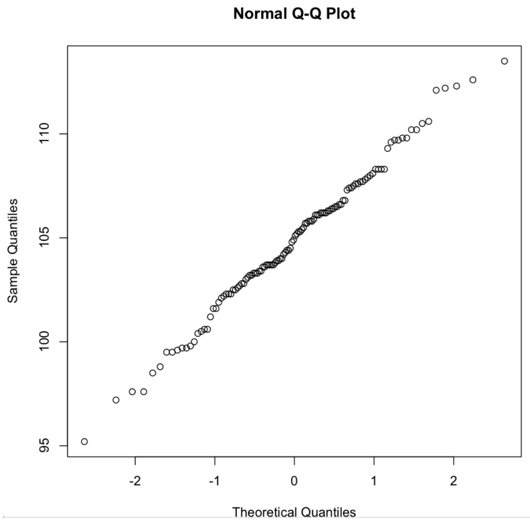
```
# 数据见例 3.19
summary(x)
  # 数据简单描述
table(x)
  # 制作频数表
shapiro.test(x)
     # 正态性检验之 Shapiro-Wilk 检验
stem(x)
  # 画出茎叶图
boxplot(x)
  # 画出箱线图
qqnorm(x)
     # 画出正态概率图（Q-Q 图）
```

结果：

```
summary(x)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  95.2   102.8   105.0   104.9   107.3   113.5
table(x)
95.2  97.2 ....... 113.5
  1     1  ......... 1
```

```
shapiro.test(x)
    Shapiro-Wilk normality test
data:  x
W = 0.9934, p-value = 0.8433
stem(x)
  95 | 2
  96 |
  97 | 266
  98 | 58
  99 | 556778
 100 | 04566
 101 | 2669
 102 | 12333556788
 103 | 01223334466777777899
 104 | 002344589
 105 | 123345778889
 106 | 11122223344556688
 107 | 3445667789
 108 | 013333
 109 | 367788
 110 | 2256
 111 |
 112 | 1236
 113 | 5
```





例 4.13

```
# 数据同例 3.19
t.test(x)
  # 单样本 t 检验的结果中包含 95%置信区间
```

结果：

```
One Sample t-test
data:  x
t = 324.902, df = 119, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
```

```
104.2466 105.5251
sample estimates:
mean of x
 104.8858
```

例 4.14

：单样本 t 检验

```
x <- c(74, 73, 68, 75, 75, 82, 80, 69, 72, 74, 83, 72, 71, 74, 76, 79, 67, 73, 81, 70,
       67, 70, 78, 69, 70, 72, 67, 74, 80, 66)
d <- x-72
t.test(d)
```

结果：

```
t.test(d)
    One Sample t-test
data:  d
t = 1.5585, df = 29, p-value = 0.13
alternative hypothesis: true mean is not equal to 0
```

例 4.15

：配对 t 检验

```
x <- data.frame(zhch=c(3550, 2000, 3000, 3950, 3800, 3750, 3450, 3050, 3350,
     3650), quefa=c(2450, 2400, 1800, 3200, 3250, 2700, 2500, 1750, 2100,
     2550))
t.test(x$zhch, x$quefa, paired= TRUE)
```

结果：

```
Paired t-test
data:  x$zhch and x$quefa
t = 5.5238, df = 9, p-value = 0.0003687
alternative hypothesis: true difference in means is not equal to 0
```

例 4.16

：独立样本（成组）t 检验

```
y <- c(1, 26, 1, 32, 1, 25, 1, 22, 1, 20, 1, 28, 1, 24, 1, 19, 1, 29, 1, 17, 1, 34, 1, 21, 1,
       20, 1, 23, 1, 27, 2, 21, 2, 23, 2, 18, 2, 24, 2, 23, 2, 19, 2, 16, 2, 22, 2, 20, 2, 25,
       2, 23, 2, 17, 2, 15, 2, 26, 2, 22)
```

```
group <- y[c(seq(1, length(y), 2 ) ) ]
x <- y[c(seq(2, length(y), 2 ) ) ]
t.test(x~group)
```

结果:

```
Welch Two Sample t-test
data:  x by group
t = 2.3115, df = 24.653, p-value = 0.02946
alternative hypothesis: true difference in means is not equal to 0
```

例 4.17

：独立样本（成组）t 检验，对数转换

```
x <- c(1, 50, 1, 30, 1, 40, 1, 60, 1, 60, 1, 35, 1, 70, 1, 20, 1, 70, 1, 35, 1, 40, 1, 50, 1,
        25, 2, 40, 2, 30, 2, 25, 2, 10, 2, 25, 2, 30, 2, 35, 2, 15, 2, 20, 2, 40, 2, 15, 2, 30,
        2, 20)
group <- x[c(seq(1, length(x), 2))]
y <- log10(x[c(seq(2, length(x), 2))])
```

结果:

```
Welch Two Sample t-test
data:  y by group
t = 3.5509, df = 23.903, p-value = 0.001631
alternative hypothesis: true difference in means is not equal to 0
```

例 4.18

：两均数等效检验

```
n1 <- 200
n2 <- 200
m1 <- 4.65
m2 <- 4.58
s1 <- 0.47
s2 <- 0.41
delta <- 0.25
ss1 <- s1^2*(n1-1)
ss2 <- s2^2*(n2-1)
sc2 <- (ss1+ss2)/(n1+n2-2)
se <- sqrt(sc2*(1/n1+1/n2))
t <- (delta-abs(m1-m2))/se
p <- (1-pt(t, n1+n2-2))*2
  # pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)为 t 分布的概率函数，表示自
        由度为 df 的变量 X<=q 的概率
```

结果：

```
t
[1] 4.081433
p
[1] 5.410668e-05
```

例 5.1

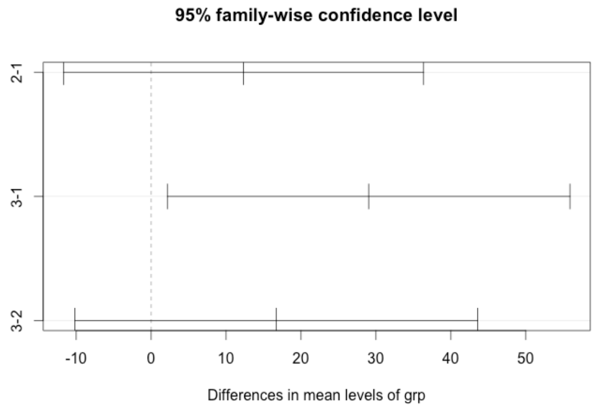
：单因素方差分析

```
x <- c(40, 10, 35, 25, 20, 15, 35, 15, -5, 30, 25, 70, 65, 45, 50, 50, 20, 45, 55, 20, 15,
    80, -10, 105, 75, 10, 60, 45, 60, 30, 60, 30, 100, 85, 20, 55, 45, 30, 77, 105)
group <- c(rep(1, 15), rep(2, 15), rep(3, 10))
    # rep(1, 15)用于生成 15 个 1
grp <- as.factor(group)
  #将 group 转化为 factor 类型，后续 Levene’s test 及方差分析均需使用 factor
     类型

d <- split(x, group)
    # 将 x 以分组变量 group 为准，分割成三组，并以 list 数据结构存储于 d
# 分组做正态性检验
sapply(d, shapiro.test)
    # sapply(x, FUN, ...)可调用清单 x，并对其中包含的向量分别应用函数 FUN

# 方差齐性检验
bartlett.test(x, grp)
    # Bartlett's test 位于 stats 包中，默认加载
library(car)
leveneTest(x, grp)
    # Levene’s test 位于 car 包，需另外加载

# 方差分析
model <- aov(x~grp)
  # 建立方差分析模型，分组变量必须为 factor 类型，否则自由度会出错
summary(model)

# 分组输出均值和标准差
library(doBy)
summaryBy(x~ group, data= data.frame(x= x, group= group), FUN=c(mean, sd))

# 组间比较
compr <- TukeyHSD(model)
compr
plot(compr)
    # Tukey 检验因能给出精确 p 值，较 SNK、Dunnett 等检验为佳
```

```
# plot()能直观的划出三个均数的比较图
```

结果:

```
> sapply(d, shapiro.test)
          1                    2                    3
statistic  0.9778002          0.9834856          0.9400544
p.value   0.9522585          0.9878477          0.5536185
> bartlett.test(x,grp)
  Bartlett test of homogeneity of variances
data:  x and grp
Bartlett's K-squared = 2.4482, df = 2, p-value = 0.294
> leveneTest(x, grp)
Levene's Test for Homogeneity of Variance (center = median)
      Df  F value   Pr(>F)
group  2  1.4002  0.2593
      37
> model= aov(x~grp)
> summary(model)
            Df  Sum Sq  Mean Sq  F value  Pr(>F)
grp          2   5062   2531.2   3.485   0.0411 *
Residuals   37  26877   726.4
> summaryBy(x~ group, data= data.frame(x= x, group= group), FUN=c(mean, sd))
  group   x.mean     x.sd
1     1  31.66667  20.32474
2     2  44.00000  30.36681
3     3  60.70000  30.15534
> compr= TukeyHSD(model)
> compr
  Tukey multiple comparisons of means
    95% family-wise confidence level
Fit: aov(formula = x ~ grp)
$grp
        diff       lwr       upr      p adj
2-1  12.33333  -11.694604  36.36127  0.4302024
3-1  29.03333   2.169283  55.89738  0.0317315
3-2  16.70000  -10.164050  43.56405  0.2944100
> plot(compr)
```



例 5.2

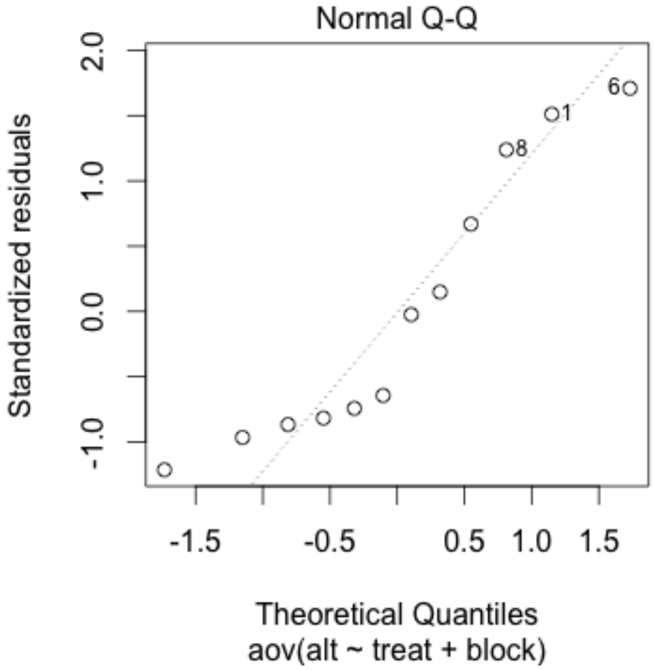
：随机区组设计方差分析

block	treat	alt
1	A	76
1	B	86
1	C	115
2	A	12
2	B	38
2	C	85
3	A	40
3	B	81
3	C	103
4	A	12
4	B	33
4	C	57

```
# 可整理为如上表格，存储为 alt.csv 文档
alt <- read.csv("alt.csv")

# 或
block <- c(rep(1, 3), rep(2, 3), rep(3, 3), rep(4, 3))
treat <- c(rep(c("A", "B", "C"), 4))
alt <- c(76, 86, 115, 12, 38, 85, 40, 81, 103, 12, 33, 57)
alt <- data.frame(block= block, treat= treat, alt= alt)

# 转换为 factor 类型（重要）
alt$block <- as.factor(alt$block)
alt$treat <- as.factor(alt$treat)

# 方差分析、两两比较
model <- aov(alt~ treat+ block, data= alt)
summary(model)
TukeyHSD(model)

# 残差正态性分析、残差图（Q-Q plot）
res <- residuals(model)
  # 从模型中提取残差
shapiro.test(res)
plot(model)
  # 可获得多种图形
```

结果：

```
> summary(model)
          Df  Sum Sq  Mean Sq  F value   Pr(>F)
treat      2   6074   3037.0   33.54   0.000554 ***
block      3   6458   2152.6   23.77   0.000992 ***
Residuals  6    543     90.6
> TukeyHSD(model)
  Tukey multiple comparisons of means
    95% family-wise confidence level
Fit: aov(formula = alt ~ treat + block, data = alt)
$treat
    diff       lwr      upr     p adj
B-A 24.5  3.853961 45.14604  0.0251803
C-A 55.0 34.353961 75.64604  0.0004428
C-B 30.5  9.853961 51.14604  0.0094278
$block
         diff        lwr        upr     p adj
2-1 -47.33333 -74.230267 -20.436399  0.0036152
3-1 -17.66667 -44.563601   9.230267  0.2062995
4-1 -58.33333 -85.230267 -31.436399  0.0011919
3-2  29.66667   2.769733  56.563601  0.0333717
4-2 -11.00000 -37.896934  15.896934  0.5343125
4-3 -40.66667 -67.563601 -13.769733  0.0077913
> shapiro.test(res)
     Shapiro-Wilk normality test
data:  res
W = 0.8828, p-value = 0.09529
```



例 5.3

：拉丁方，方差分析

```
cards <- c("B", 103, "A", 121, "C", 100, "D", 92, "E", 95, "C", 102, "B", 129, "D", 98,
           "E", 124, "A", 115, "D", 118, "C", 133, "E", 103, "A", 109, "B", 90, "E", 99,
```

```
        "D", 122, "A", 99, "B", 84, "C", 100, "A", 102, "E", 139, "B", 103, "C", 104,
        "D", 95)
# 建立一系列空向量
person <- c()
stress <- c()
cloth <- c()
x <- c()
k <- 1
  # k 为观测数指针
for(i in 1:5)
{
 for(j in 1:5)
 {
  person[k]=i
  stress[k]=j
  cloth[k]=cards[(5*(i-1)+j)*2-1]
     x[k]=cards[(5*(i-1)+j)*2]
      # 通过 i、j 的组合读取向量 cards 中的数据
  k=k+1
 }
}

Dat <- data.frame(person=as.factor(person), stress=as.factor(stress),
      cloth=as.factor(cloth), x=as.numeric(x))
  # as.numeric()函数为必须，否则 x 将为 factor（不知何故）
model <- aov(x~ person+ stress+ cloth, data= dat)
summary(model)
write.csv(dat, “5.3.csv”)
  # 输出 csv 文件，表格如下
```

结果：

```
> summary(model)
          Df  Sum Sq  Mean Sq  F value   Pr(>F)
person     4   602.2   150.5    2.014   0.156424
stress     4  3021.4   755.3   10.106   0.000809 ***
cloth      4   307.0    76.7    1.027   0.432611
Residuals 12   896.9    74.7
```

	person	stress	cloth	x
1	1	1	B	103
2	1	2	A	121
3	1	3	C	100
4	1	4	D	92
5	1	5	E	95
6	2	1	C	102
7	2	2	B	129
8	2	3	D	98

9	2	4	E	124
10	2	5	A	115
11	3	1	D	118
12	3	2	C	133
13	3	3	E	103
14	3	4	A	109
15	3	5	B	90
16	4	1	E	99
17	4	2	D	122
18	4	3	A	99
19	4	4	B	84
20	4	5	C	100
21	5	1	A	102
22	5	2	E	139
23	5	3	B	103
24	5	4	C	104
25	5	5	D	95

例 5.4

：析因方差分析

```
a <- c(rep(0, 6), rep(1, 6))
b <- c(rep(c(0, 0, 0, 1, 1, 1), 2))
x <- c(1.0, 0.9, 0.8,
   1.5, 1.4, 1.6,
   1.2, 1.3, 1.1,
   2.3, 2.4, 2.5)
dat <- data.frame(a= as.factor(a), b= as.factor(b), x= as.numeric(x))
model <- aov(x~a*b, data=dat)
   # a*b 等价于 a+b+a:b， a:b 表示 a 和 b 的相互作用
summary(model)
dat
```

结果：

```
> summary(model)
        Df  Sum Sq  Mean Sq  F value   Pr(>F)
a        1   1.08    1.08     108   6.36e-06 ***
b        1   2.43    2.43     243   2.86e-07 ***
a:b      1   0.27    0.27      27   0.000826 ***
```

```
Residuals    8   0.08    0.01
> dat
   a  b   x
1   0  0  1.0
2   0  0  0.9
3   0  0  0.8
4   0  1  1.5
5   0  1  1.4
6   0  1  1.6
7   1  0  1.2
8   1  0  1.3
9   1  0  1.1
10  1  1  2.3
11  1  1  2.4
12  1  1  2.5
```

例 5.5

：析因方差分析

```
cards <- c(1.9, 1.8, 1.6, 1.4, 1.8, 1.7, 1.4, 1.5, 2.3, 2.1, 2.0, 2.6, 2.4, 2.7,
       2.4, 2.6, 2.9, 2.8, 3.4, 3.2, 3.0, 3.1, 3.0, 2.7, 2.1, 2.0, 1.8, 2.2,
       2.3, 2.0, 1.9, 1.7, 2.4, 2.6, 2.7, 2.3, 2.0, 2.3, 2.1, 2.4, 3.6, 3.1,
       3.4, 3.2, 3.1, 3.0, 2.8, 3.2, 1.1, 1.2, 1.0, 1.4, 1.4, 1.0, 1.3, 1.2,
       2.0, 2.1, 1.9, 2.2, 2.4, 2.6, 2.3, 2.2, 2.9, 2.8, 3.0, 3.1, 3.2, 2.9,
       2.8, 2.9)
sort <- c()
temp <- c()
sex <- c()
rate <- c()
h <- 1
for(i in 1:3)
{
 for(j in 1:3)
 {
  for(k in 1:2)
  {
   for(l in 1:4)
   {
     sort[h]=i
     temp[h]=j
     sex[h]=k
     rate[h]=cards[(i-1)*3*2*4+(j-1)*2*4+(k-1)*4+l]
      # (i-1)乘的数应为后续循环的乘积
     h=h+1
   }
  }
 }
}
```

```
dat <- data.frame(sort=as.factor(sort), temp=as.factor(temp),
                  sex=as.factor(sex), rate=as.numeric(rate))

model <- aov(rate~sort*temp*sex, data=dat)
  # sort*temp*sex 等价于 sort+ temp+ sex+ sort:temp+ sort:sex+ temp:sex+
          sort:temp:sex
summary(model)

model1 <- aov(rate~sort+ temp+ sex+ sort:temp+ sort:sex, data=dat)
  # 上步结果后重新建立模型
summary(model1)
```

结果:

```
>model=aov(rate~ sort*temp*sex, data=dat)
> summary(model)
             Df Sum Sq Mean Sq F value   Pr(>F)
sort          2  1.817   0.909  24.475  2.71e-08 ***
temp          2 24.656  12.328 332.024  < 2e-16 ***
sex           1  0.009   0.009   0.239  0.6266
sort:temp     4  1.102   0.275   7.418  7.75e-05 ***
sort:sex      2  0.370   0.185   4.986  0.0103 *
temp:sex      2  0.175   0.088   2.360  0.1041
sort:temp:sex 4  0.221   0.055   1.485  0.2196
Residuals    54  2.005   0.037
> model1=aov(rate~sort+ temp+ sex+ sort:temp+ sort:sex, data=dat)
> summary(model1)
           Df  Sum Sq  Mean Sq  F value   Pr(>F)
sort        2  1.817   0.909    22.711  4.54e-08 ***
temp        2  24.656  12.328   308.091  < 2e-16 ***
sex         1  0.009   0.009    0.222  0.639120
sort:temp   4  1.102   0.275    6.883  0.000125 ***
sort:sex    2  0.370   0.185    4.627  0.013528 *
Residuals  60  2.401   0.040
```

例 5.6

: 正交设计，方差分析

a	b	c	d	y
1	1	1	1	29
1	1	2	2	41
1	2	1	2	51
1	2	2	1	49
2	1	1	2	59
2	1	2	1	55
2	2	1	1	57

2	2	2	2	66

```
# 建立如上文件，存储为 r5.6.csv
dat <- read.csv("r5.6.csv")
model <- aov(y~ a+ b+ c+ d+ a:b+ a:c, data=dat)
summary(model)
```

结果：

```
> summary(model)
          Df  Sum Sq  Mean Sq  F value  Pr(>F)
a          1  561.1   561.1    4489   0.0095 **
b          1  190.1   190.1    1521   0.0163 *
c          1   28.1    28.1     225   0.0424 *
d          1   91.1    91.1     729   0.0236 *
a:b        1   55.1    55.1     441   0.0303 *
a:c        1    3.1     3.1      25   0.1257
Residuals  1    0.1     0.1
```

例 5.7

：平衡不完全区组设计方差分析

block	treat	x
1	1	3.1
1	2	1.3
1	3	2.1
2	1	3.5
2	2	1.7
2	4	1.3
3	1	3.4
3	3	2.3
3	4	1.4
4	2	1.5
4	3	2.5
4	4	1.5

```
# 建立如上文件，存储为 r5.7.csv
dat <- read.csv("r5.7.csv")
dat$block <- as.factor(dat$block)
dat$treat <- as.factor(dat$treat)
```

```
model <- aov(x~ treat+ block, data=dat)
summary(model)
TukeyHSD(model)
```

结果：

```
> model= aov(x~ treat+ block, data=dat)
> summary(model)
          Df Sum Sq Mean Sq  F value   Pr(>F)
treat      3  7.220  2.4067  222.154  9.72e-06 ***
block      3  0.212  0.0708   6.538    0.035 *
Residuals  5  0.054  0.0108
> TukeyHSD(model)
  Tukey multiple comparisons of means
    95% family-wise confidence level
Fit: aov(formula = x ~ treat + block, data = dat)
$treat
        diff        lwr        upr     p adj
2-1 -1.833333 -2.1469156 -1.519751   0 .0000176
3-1 -1.033333 -1.3469156 -0.7197510  0.0002578
4-1 -1.933333 -2.2469156 -1.6197510  0.0000147
3-2  0.800000  0.4864177  1.1135823  0.0008814
4-2 -0.100000 -0.4135823  0.2135823  0.6647867
4-3 -0.900000 -1.2135823 -0.5864177  0.0005028
$block
           diff          lwr       upr     p adj
2-1  0.30000000 -0.013582298 0.6135823  0.0586457
3-1  0.23333333 -0.080248965 0.5469156  0.1334790
4-1  0.31111111 -0.002471187 0.6246934  0.0514606
3-2 -0.06666667 -0.380248965 0.2469156  0.8587605
4-2  0.01111111 -0.302471187 0.3246934  0.9990927
4-3  0.07777778 -0.235804521 0.3913601  0.7988047
```

例 6.1

：线性相关

x1	x2
9.9	7.9
11.2	8.9
9.4	8.5
8.4	9.4
14.8	12
12.4	11.5
13.1	14.5
13.4	12.3
11.2	9.2

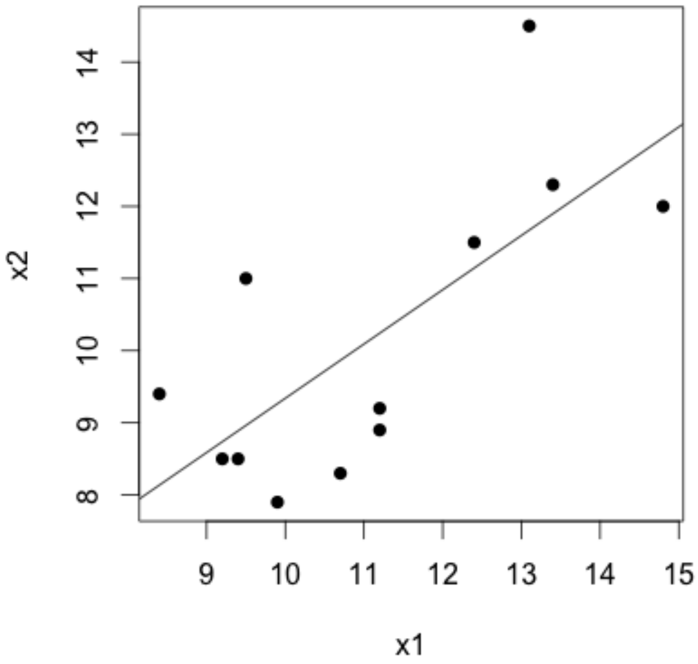
9.5	11
10.7	8.3
9.2	8.5

```
# 建立如上文件，存储为 r6.1.csv
dat <- read.csv("r6.1.csv")
# 绘制散点图
plot(x2~x1, pch=16, data=dat)
# 可用 cor.test()函数求相关系数和 p 值
cor.test(dat$x1, dat$x2, method="pearson")
  # method="pearson"可不写，此为默认

# 或用 lm()函数建立线性回归模型
model <- lm(x2~x1, data=dat)
  # x2 为因变量，x1 为自变量
# 添加回归曲线至散点图
abline(model)
summary(model)

# 验证二元正态分布,即 x1 和残差符合正态分布
res <- residuals(model)
shapiro.test(dat$x1)
shapiro.test(res)
```

结果：

```
> cor.test(dat$x1, dat$x2, method="pearson")
    Pearson's product-moment correlation
data:  dat$x1 and dat$x2
t = 3.2854,  df = 10,  p-value = 0.008214
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2499064 0.9157367
sample estimates:
    cor
0.7204761
> summary(model)
Call:
lm(formula = x2 ~ x1, data = dat)
Residuals:
   Min      1Q  Median      3Q     Max
-1.5658 -1.1169 -0.3129  0.6186  2.8291
Coefficients:
          Estimate  Std. Error   t value  Pr(>|t|)
(Intercept)  1.8182     2.5775    0.705  0.49664
x1          0.7521     0.2289   3.285  0.00821 **
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.495 on 10 degrees of freedom
Multiple R-squared:  0.5191,      Adjusted R-squared:  0.471
F-statistic: 10.79 on 1 and 10 DF,  p-value: 0.008214
> shapiro.test(dat$x1)
```

```
    Shapiro-Wilk normality test
data:  dat$x1
W = 0.9499, p-value = 0.6354
> shapiro.test(res)
    Shapiro-Wilk normality test
data:  res
W = 0.9074, p-value = 0.1974
```



例 6.2

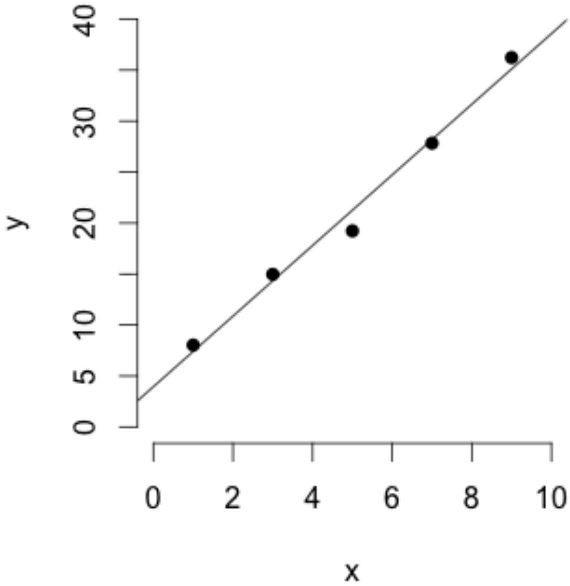
：线性回归

```
x <- c(1, 3, 5 ,7, 9)
y <- c(8.03, 14.97, 19.23, 27.83, 36.23)
model <- lm(y~x)
summary(model)
plot(y~x, pch=16, xaxt="n", yaxt="n", bty="n", xlim=c(0,10), ylim=c(0,40))
    # xaxt="n"表示无坐标; bty="n"表示无边框; xlim=c(0,10)表示坐标轴范围
# 添加坐标轴
axis(side= 1, at= seq(0,10,2))
axis(side= 2, at= seq(0,40,5))
# 添加回归曲线
abline(model)
```

结果：

```
> summary(model)
Call:
lm(formula = y ~ x)
Residuals:
   1    2    3    4    5
```

```
 0.624   0.638  -2.028  -0.354  1.120
Coefficients:
            Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)   3.9430     1.3151   2.998  0.057748 .
x             3.4630     0.2289  15.127  0.000627 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.448 on 3 degrees of freedom
Multiple R-squared:  0.9871,       Adjusted R-squared:  0.9827
F-statistic: 228.8 on 1 and 3 DF,  p-value: 0.0006272
```



例 6.3

：线性回归方程的比较

group	x	y
1	75	84
1	75	93
1	90	98
1	95	106
1	96	103
1	97	110
1	102	113
1	114	126
1	106	115
1	104	115
1	115	122

1	131	135
1	125	129
1	117	125
1	124	130
1	121	129
1	165	178
2	113	92
2	113	92
2	113	102
2	131	112
2	124	102
2	129	128
2	127	122
2	124	115
2	120	108
2	132	126
2	127	115
2	140	121
2	137	121
2	137	122
2	144	133
2	148	130
2	174	157
2	177	155

```
# 建立如上文件，存储为 r6.3.csv
# 此题本质上为协方差分析
dat <- read.csv("r6.3.csv")
dat$group <- as.factor(dat$group)
model <- aov(y~ x*group, data= dat)
  # 建立方差分析模型
summary(model)

model <- aov(y~ x+ group, data= dat)
summary(model)
     # 此项结果的 group 选项为两组的曲线比较，此为 I 型方差分析

# car 包中的 Anova 函数可对 aov 模型求 III 型方差分析
```

```
#模型中资料为不平衡数据（同时有分组和连续资料），应该用 III 型方差分析
library(car)
Anova(model, type=3)
```

结果：

```
> summary(model)
          Df  Sum Sq  Mean Sq  F value   Pr(>F)
x          1   8676    8676  328.989  < 2e-16 ***
group      1   3023    3023  114.627  6.13e-12 ***
x:group    1      2       2    0.089    0.768
Residuals 31    818      26
> summary(model)
          Df  Sum Sq  Mean Sq  F value   Pr(>F)
x          1   8676    8676   338.6   < 2e-16 ***
group      1   3023    3023   118.0   2.89e-12 ***
Residuals 32    820      26
> Anova(model, type=3)
Anova Table (Type III tests)
Response: y
            Sum Sq Df F value    Pr(>F)
(Intercept)   273.7  1  10.684  0.002585 **
x          11683.9  1 456.027 < 2.2e-16 ***
group       3023.0  1 117.987 2.886e-12 ***
Residuals    819.9 32
```

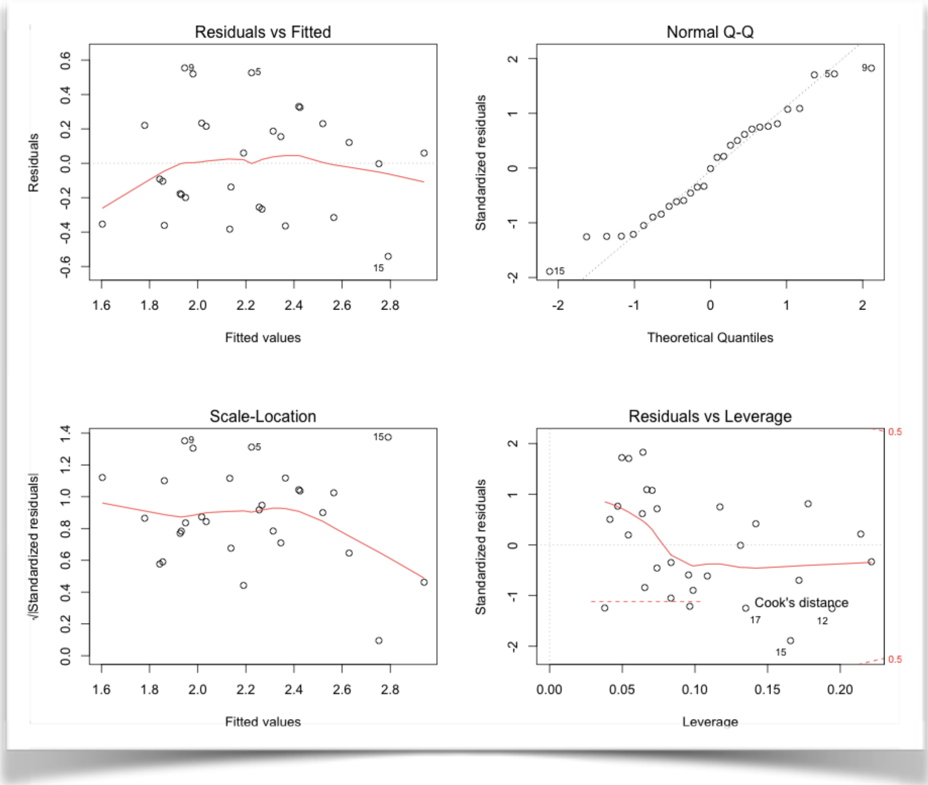
例 6.4

：多元线性回归

```
cards <- c(135.1, 32.0, 1.75, 139.9, 30.4, 2.00, 163.6, 46.2, 2.75, 146.5, 33.5, 2.50,
           156.2, 37.1, 2.75, 156.4, 35.5, 2.00, 167.8, 41.5, 2.75, 149.7, 31.0, 1.50,
           145.0, 33.0, 2.50, 148.5, 37.2, 2.25, 165.5, 49.5, 3.00, 135.0, 27.6, 1.25,
           153.3, 41.0, 2.75, 152.0, 32.0, 1.75, 160.5, 47.2, 2.25, 153.0, 32.0, 1.75,
           147.6, 40.5, 2.00, 157.5, 43.3, 2.25, 155.1, 44.7, 2.75, 160.5, 37.5, 2.00,
           143.0, 31.5, 1.75, 149.4, 33.9, 2.25, 160.8, 40.4, 2.75, 159.0, 38.5, 2.50,
           158.2, 37.5, 2.00, 150.0, 36.0, 1.75, 144.5, 34.7, 2.25, 154.6, 39.5, 2.50,
           156.5, 32.0, 1.75)
x1 <- cards[seq(1, length(cards), 3)]
x2 <- cards[seq(2, length(cards), 3)]
y <- cards[seq(3, length(cards), 3)]
model <- lm(y~x1+x2)
summary(model)
par(mfrow=c(2, 2))
  # 将绘图窗口设置为 2*2 的排列方式
plot(model)
  # 观察残差与预测值散点图(无线性相关)和 Q-Q 图
shapiro.test(residuals(model))
```

结果：

```
> summary(model)
Call:
lm(formula = y ~ x1 + x2)
Residuals:
    Min      1Q   Median      3Q     Max
-0.54117 -0.25524 -0.00266  0.22039  0.55425
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.565664   1.240127  -0.456  0.65208
x1          0.005017   0.010575   0.474  0.63920
x2          0.054061   0.015984   3.382  0.00228 **
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.3137 on 26 degrees of freedom
Multiple R-squared:  0.546,  Adjusted R-squared:  0.511
F-statistic: 15.63 on 2 and 26 DF,  p-value: 3.485e-05
> shapiro.test(residuals(model))
     Shapiro-Wilk normality test
data:  residuals(model)
W = 0.9596, p-value = 0.3205
```



例 6.5

：多元相关

```
cards <- c(135.1, 32.0, 1.75, 139.9, 30.4, 2.00, 163.6, 46.2, 2.75, 146.5, 33.5, 2.50,
           156.2, 37.1, 2.75, 156.4, 35.5, 2.00, 167.8, 41.5, 2.75, 149.7, 31.0, 1.50,
           145.0, 33.0, 2.50, 148.5, 37.2, 2.25, 165.5, 49.5, 3.00, 135.0, 27.6, 1.25,
           153.3, 41.0, 2.75, 152.0, 32.0, 1.75, 160.5, 47.2, 2.25, 153.0, 32.0, 1.75,
```

```
          147.6, 40.5, 2.00, 157.5, 43.3, 2.25, 155.1, 44.7, 2.75, 160.5, 37.5, 2.00,
          143.0, 31.5, 1.75, 149.4, 33.9, 2.25, 160.8, 40.4, 2.75, 159.0, 38.5, 2.50,
          158.2, 37.5, 2.00, 150.0, 36.0, 1.75, 144.5, 34.7, 2.25, 154.6, 39.5, 2.50,
          156.5, 32.0, 1.75)
x1 <- cards[seq(1, length(cards), 3)]
x2 <- cards[seq(2, length(cards), 3)]
y <- cards[seq(3, length(cards), 3)]

# 安装并加载 ppcor 包,内含 pcor.test()偏相关函数
install.packages("ppcor")
library(ppcor)
pcor.test(y, x1, x2)
pcor.test(y, x2, x1)
```

结果：

```
> pcor.test(y, x1, x2)
   estimate   p.value  statistic  n   gp  Method
1 0.09262918  0.6352451  0.4743574  29  1   pearson
> pcor.test(y, x2, x1)
  estimate     p.value    statistic   n  gp   Method
1 0.5527642  0.0007189499  3.382249  29  1   pearson
```

偏相关系数分别为：0.09262918 和 0.5527642

例 6.6

：逐步回归

age	ps	pd	pr	as	ad	sv
33	90	60	25.124	44.673	60	55.86
34	112	70	27.166	43.93	71	51.92
42	116	70	26.785	38.154	82	46
33	110	70	27.728	58.136	59	50.04
33	86	50	20.171	36.114	65	36
39	102	76	28.492	60.058	56	74.07
19	105	70	18.34	35.85	88	96.69
...						

```
# 整理为如上表格，存储为 r6.6.csv
dat <- read.csv("r6.6.csv")
# 建立线性模型
lmmod <- lm(sv~age+ps+pd+as+ad+pr, data=dat)
# 对线性模型进行逐步回归
stpmod <- step(lmmod)
```

```
summary(stpmod)
```

结果：

```
> summary(stpmod)
Call:
lm(formula = sv ~ age + as + ad, data = dat)
Residuals:
    Min      1Q  Median      3Q     Max
-52.704 -16.080  -2.261  15.716  85.465
Coefficients:
            Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  -47.7910    38.5709  -1.239  0.21753
age          -0.8603     0.1986  -4.332  2.90e-05 ***
as           1.8157      0.3595   5.051  1.43e-06 ***
ad           1.0756      0.3405   3.159  0.00196 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 25 on 132 degrees of freedom
Multiple R-squared:  0.215,  Adjusted R-squared:  0.1971
F-statistic: 12.05 on 3 and 132 DF,  p-value: 5.065e-07
```

例 7.1

：完全随机设计协方差分析

```
# 建立数据集
a <- c(631.3, 68.2, 709.4, 77.1, 668.5, 65, 754.1, 85, 629.1, 66.8, 699.5, 70, 727.6,
    81.9, 728.7, 78.8)
b <- c(774.9, 90, 749.1, 82, 689.8, 72.4, 763.2, 87.6, 680.6, 72.1, 744.3, 80.4,
    742.7, 85.9)
c <- c(767.8, 91, 750.7, 83, 780.4, 95, 790.1, 100, 780.5, 102, 760.8, 105, 745.1,
    110, 727.0, 89)
group <- as.factor(c(rep(1, 8), rep(2, 7), rep(3, 8)))
x <- c(a[seq(1, 16, 2)], b[seq(1, 14, 2)], c[seq(1, 16, 2)])
y <- c(a[seq(2, 16, 2)], b[seq(2, 14, 2)], c[seq(2, 16, 2)])

model <- aov(y~group*x)
summary(model)
# x 和 group 无交互作用，去除 group:x 因数
model <- aov(y~group+x)

# 输出 I 型方差分析结果
summary(model)

# car 包中的 Anova 函数可对 aov 模型求 III 型方差分析（type III）
# 模型中资料为不平衡数据（向量长度不同），应该用 III 型方差分析
library(car)
Anova(model, type=3)
```

```r
# install.packages("lsmeans")安装 lsmeans 包，用于求修正均数
library(lsmeans)
lsmeans(model, ~group)
  # ~group 选项说明修正均数以 group 分组输出
```

结果：

```
> model=aov(y~group*x)
> summary(model)
            Df Sum Sq Mean Sq F value   Pr(>F)
group        2 2152.8  1076.4  28.537 3.69e-06 ***
x            1  605.7   605.7  16.059 0.000913 ***
group:x      2   19.8     9.9   0.263 0.771929
Residuals   17  641.2    37.7
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> model=aov(y~group+x)
> summary(model)
            Df Sum Sq Mean Sq F value   Pr(>F)
group        2 2152.8  1076.4   30.94  1.06e-06 ***
x            1  605.7   605.7   17.41  0.000517 ***
Residuals   19  661.1    34.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> Anova(model)
Anova Table (Type III tests)
Response: y
            Sum Sq Df F value    Pr(>F)
(Intercept)  54.75  1  1.5736 0.2249048
group       489.55  2  7.0353 0.0051697 **
x           605.73  1 17.4098 0.0005167 ***
Residuals   661.05 19
> lsmeans(model, ~group)
$`group lsmeans`
 group   lsmean       SE  df lower.CL  upper.CL
     1 79.71513 2.481950  19 74.52035  84.90991
     2 80.76407 2.236124  19 76.08380  85.44433
     3 91.89131 2.403261  19 86.86123  96.92139
```

例 7.2

：随机区组设计，协方差分析

treat	block	x	y
1	1	256.9	27
1	2	271.6	41.7
1	3	210.2	25
1	4	300.1	52
1	5	262.2	14.5
1	6	304.4	48.8

1	7	272.4	48
1	8	248.2	9.5
1	9	242.8	37
1	10	342.9	56.5
1	11	356.9	76
1	12	198.2	9.2
2	1	260.3	32
2	2	271.1	47.1
2	3	214.7	36.7
2	4	300.1	65
2	5	269.7	39
2	6	307.5	37.9
2	7	278.9	51.5
2	8	256.2	26.7
2	9	240.8	41
2	10	340.7	61.3
2	11	356.3	102.1
2	12	199.2	8.1
3	1	544.7	160.3
3	2	481.2	96.1
3	3	418.9	114.6
3	4	556.6	134.8
3	5	394.5	76.3
3	6	426.6	72.8
3	7	416.1	99.4
3	8	549.9	133.7
3	9	580.5	147
3	10	608.3	165.8
3	11	559.6	169.8
3	12	371.9	54.3

```
#整理为如上表格，存储为 r7.2.csv
dat <- read.csv("r7.2.csv")
dat$treat <- as.factor(dat$treat)
dat$block <- as.factor(dat$block)
model <- aov(y~treat+block+x, data=dat)
```

```
# install.packages("car")
library(car)
Anova(model, type=3)
    # 协方差分析应使用 III 型方差分析（非平衡数据）
library(lsmeans)
lsmeans(model, pairwise ~treat)
    # pairwise 表示修正均数的两两比较，若无此选项则不输出两两比较的结果
    # ~treat 表示以 treat 为分组变量
```

结果：

```
> Anova(model)
Anova Table (Type III tests)
Response: y
          Sum Sq Df F value    Pr(>F)
(Intercept) 2030.5  1 19.1614 0.0002635 ***
treat       463.9  2  2.1891 0.1369239
block       3765.3 11  3.2302 0.0100914 *
x           6174.2  1 58.2643 1.733e-07 ***
Residuals   2225.4 21
> lsmeans(model, pairwise ~treat)
$`treat lsmeans`
 treat   lsmean       SE df lower.CL upper.CL
     1 67.42823 4.961606 21 57.11001 77.74646
     2 75.05049 4.859635 21 64.94433 85.15665
     3 59.06294 8.364110 21 41.66882 76.45706
$`treat pairwise differences`
       estimate        SE df  t.ratio p.value
1 - 2 -7.622256  4.204525 21 -1.81287 0.18977
1 - 3  8.365292 12.518182 21  0.66825 0.78418
2 - 3 15.987548 12.397598 21  1.28957 0.41641
   p values are adjusted using the tukey method for 3 means
```

例 7.3

：析因设计协方差分析

```
cards <- c(2.28, 6.33, 2.33, 6.43, 2.32, 6.43, 2.1, 6.23, 2.2, 6.2, 2.32, 4.86,
     2.27, 4.78, 2.3, 4.73, 2.23, 4.68, 2.05, 4.48, 2.15, 3.48, 2.06, 3.3,
     2.23, 3.56, 2.31, 3.63, 2.14, 3.48, 2.28, 6.32, 2.03, 6.1, 2.13, 6.12,
     2.25, 6.32, 2.1, 6.23, 2.24, 4.75, 2.22, 4.68, 2.16, 4.54, 2.35, 4.88,
     2.15, 4.53, 2.28, 3.6, 2.12, 3.47, 2.25, 3.6, 2.2, 3.51, 2.26, 3.58)
# 建立一系列空向量
se <- c()
treat <- c()
x <- c()
y <- c()
h <- 1
# h 为观测数指针
```

```
for(i in 1:2)
{
 for(j in 1:3)
 {
  for(k in 1:5)
  {
   se[h]=i
   treat[h]=j
   x[h]=cards[((3*(i-1)+j-1)*5+k)*2-1]
   y[h]=cards[((3*(i-1)+j-1)*5+k)*2]
   h=h+1
  }
 }
}
dat <- data.frame(se= as.factor(se), treat= as.factor(treat),
                x= as.numeric(x), y= as.numeric(y))
model =aov(y~se+treat+x, data=dat)
# install.packages("car")
library(car)
Anova(model, type=3)
  # 协方差分析应使用 III 型方差分析（非平衡数据）
library(lsmeans)
lsmeans(model, pairwise ~treat)
lsmeans(model, pairwise ~se)
```

结果：

```
Anova Table (Type III tests)
Response: y
            Sum Sq Df  F value    Pr(>F)
(Intercept)  0.631  1  287.5993 3.179e-15 ***
se           0.000  1    0.0422    0.8388
treat       38.124  2 8681.4568 < 2.2e-16 ***
x            0.315  1  143.3767 7.525e-12 ***
Residuals    0.055 25
> lsmeans(model, pairwise ~treat)
$`treat lsmeans`
 treat   lsmean         SE df lower.CL upper.CL
     1 6.280783 0.01484044 25 6.250219 6.311348
     2 4.669086 0.01493053 25 4.638336 4.699836
     3 3.533131 0.01485253 25 3.502542 3.563720
$`treat pairwise differences`
      estimate         SE df   t.ratio p.value
1 - 2 1.611697 0.02112226 25  76.30326       0
1 - 3 2.747652 0.02095664 25 131.11129       0
2 - 3 1.135955 0.02114773 25  53.71520       0
   p values are adjusted using the tukey method for 3 means
> lsmeans(model, pairwise ~se)
$`se lsmeans`
 se   lsmean         SE df lower.CL upper.CL
  1 4.829434 0.01213093 25 4.804450 4.854418
  2 4.825899 0.01213093 25 4.800915 4.850883
$`se pairwise differences`
         estimate         SE df t.ratio p.value
1 - 2 0.003535067 0.01720105 25 0.20551 0.83884
```

p values are adjusted using the tukey method for 2 means

例 7.4

：两个协变量，协方差分析

sex	x1	x2	y
1	54	3	2446.2
1	50.5	2.25	1928.4
1	51	2.5	2094.5
1	56.5	3.5	2506.7
1	52	3	2121
1	76	9.5	3845.9
1	80	9	4380.8
1	74	9.5	4314.2
1	80	9	4078.4
1	76	8	4134.5
1	96	13.5	5830.2
1	97	14	6013.6
1	99	16	6410.6
1	92	11	5283.3
1	94	15	6101.6
2	54	3	2117.3
2	53	2.25	2200.2
2	51.5	2.5	1906.2
2	51	3	1850.3
2	51	3	1632.5
2	77	7.5	3934
2	77	10	4180.4
2	77	9.5	4246.1
2	74	9	3358.8
2	73	7.5	3809.7
2	91	12	5358.4
2	91	13	5601.7
2	94	15	6074.9

2	92	12	5299.4
2	91	12.5	5291.5

```
# 整理为以上表格,存储为 r7.4.csv
dat <- read.csv("r7.4.csv")
dat$sex <- as.factor(dat$sex)
  # 必须，否则求校正均数时会出错
model <- aov(y~sex+x1+x2, data=dat)
library(car)
Anova(model, type=3)
  # III 型方差分析结果
library(lsmeans)
lsmeans(model, pairwise ~ sex)
```

结果：

```
> Anova(model)
Anova Table (Type III tests)
Response: y
             Sum Sq Df F value   Pr(>F)
(Intercept)  207432  1  5.0622 0.033141 *
sex          139769  1  3.4109 0.076178 .
x1           938154  1 22.8947 5.93e-05 ***
x2           368955  1  9.0040 0.005876 **
Residuals   1065400 26
> lsmeans(model, pairwise ~ sex)
$`sex lsmeans`
 sex  lsmean       SE df lower.CL upper.CL
   1 4013.458 52.32694 26 3905.898 4121.017
   2 3876.629 52.32694 26 3769.069 3984.189
$`sex pairwise differences`
      estimate       SE df t.ratio p.value
1 - 2 136.8286 74.08676 26 1.84687 0.07618
    p values are adjusted using the tukey method for 2 means
```

例 8.6（程序 8.1）

：四格表卡方检验

```
dat <- matrix(c(63, 47, 16, 7), nrow=2)
chi <- chisq.test(dat)
  # 默认 correct=T
chi$expected
  # 打印理论频数。因 n>=40，且所有格子理论频数>=5，故无需连续性校正
chisq.test(dat, correct=F)
  # 计算无连续校正的卡方检验
```

结果：

```
> chi$expected
        [,1]      [,2]
[1,] 65.33835  13.661654
[2,] 44.66165  9.338346
> chisq.test(dat, correct=F)
     Pearson's Chi-squared test
data:  dat
X-squared = 1.1919, df = 1, p-value = 0.275
```

例 8.7（程序 8.2）

：K * 2 表的卡方检验

```
dat <- matrix(c(63, 47, 65, 16, 7, 3), nrow=3)
  # nrow=3 表示 3 行
chi <- chisq.test(dat)
  # 非四格表，默认 correct=F
chi
chi$expected
  # 打印理论频数
fisher.test(dat)
  # Fisher 确切概率法检验
```

结果：

```
> chi
     Pearson's Chi-squared test
data:  dat
X-squared = 8.1431, df = 2, p-value = 0.01705
> chi$expected
        [,1]      [,2]
[1,] 68.78109  10.218905
[2,] 47.01493  6.985075
[3,] 59.20398  8.796020
> fisher.test(dat)
     Fisher's Exact Test for Count Data
data:  dat
p-value = 0.01241
alternative hypothesis: two.sided
```

例 8.8（程序 8.3）

：R*C，行列皆无须

```
dat <- matrix(c(112, 200, 362, 150, 112, 219,
    205, 135, 310, 40, 73, 69), nrow=3)
chi <- chisq.test(dat)
chi
```

```
chi$expected
  # 打印理论频数
```

结果：

```
> chi
    Pearson's Chi-squared test
data:  dat
X-squared = 71.5186, df = 6, p-value = 1.995e-13
> chi$expected
        [,1]     [,2]     [,3]     [,4]
[1,] 171.9768 122.7313 165.8530 46.43885
[2,] 176.3865 125.8782 170.1057 47.62959
[3,] 325.6366 232.3905 314.0413 87.93156
```

例 8.9（程序 8.4）

：R*C，列有序

```
# 行有序或列有序的列联表一般使用 Ridit 分析较为方便。用于对比分组变量
  （无序）是否随着等级变量（有序）的增加而不同。
# 与 SAS 类似，行、列或皆有序的联列表均可以使用 Cochran-Mantel-Haenszel
  test 进行统计。vcdExtra 包的 CMHtest()函数可给出全部结果。
# 另外，coin 包下的 cmh_test()和 lbl_test()函数也可分别用于独立资料和有序
  资料的分析。详见帮助文档。

dat <- matrix(c(13, 21, 10, 7, 6, 4), nrow=2)
dimnames(dat) <- list(c("test", "control"), c("no","better","good"))
# 给矩阵的行列命名,或用以下两行代码
# rownames(dat) <- c("test", "control")
# colnames(dat) <- c("no","better","good")
# install.packages("Ridit")
library(Ridit)
ridit(dat, 1)
  # dat 为 martix 类型列联表，1 表示分组信息在行（若 2 则表示在列，即为行
      有序）
chisq.test(dat)
     # 输出卡方检验的结果以供对比

# install.packages("vcdExtra")
library(vcdExtra)
CMHtest(dat)
  # 结果中，cor 为行列皆有序；cmeans 为列有序；rmeans 为行有序；general
      为无序（与 Pearson Chi-square 结果相近）
```

结果：

```
> ridit(dat, 1)
Ridit Analysis:
```

```
Group   Label       Mean Ridit
-----   -----       ----------
1   test        0.5551
2   control     0.4501
Reference: Total of all groups
chi-squared = 2.4752, df = 1, p-value = 0.1157
> chisq.test(dat)
    Pearson's Chi-squared test
data:  dat
X-squared = 2.6707, df = 2, p-value = 0.2631
> CMHtest(dat)
Cochran-Mantel-Haenszel Statistics
             AltHypothesis  Chisq Df    Prob
cor       Nonzero correlation 2.2194  1 0.13629
cmeans  Col mean scores differ 2.2194  1 0.13629
rmeans  Row mean scores differ 2.6269  2 0.26889
general    General association 2.6269  2 0.26889
```

例 8.10（程序 8.5）

：R*C，行有序

```
# 行有序或列有序的列联表一般使用 Ridit 分析较为方便。用于对比分组变量（无序）是否随着等级变量（有序）的增加而不同。
# 与 SAS 类似，行、列或皆有序的联列表均可以使用 Cochran-Mantel-Haenszel test 进行统计。vcdExtra 包的 CMHtest()函数可给出全部结果。
# 另外，coin 包下的 cmh_test()和 lbl_test()函数也可分别用于独立资料和有序资料的分析。详见帮助文档。

dat <- matrix(c(59, 169, 196, 25, 29, 9), nrow=3)
dimnames(dat) <- list(drug=c("low", "mid", "high"), toxicity=c("yes","no"))
# 给矩阵的行列命名,drug 和 toxicity 为行列名称
# install.packages("Ridit")
library(Ridit)
ridit(dat, 2)
  # dat 为 martix 类型列联表，2 表示分组信息在列（若 1 则表示在行，即为列有序）
chisq.test(dat)
    # 输出卡方检验的结果以供对比

# install.packages("vcdExtra")
library(vcdExtra)
CMHtest(dat)
  # 结果中，cor 为行列皆有序；cmeans 为列有序；rmeans 为行有序；general 为无序（与 Pearson Chi-square 结果相近）
```

结果：

```
> ridit(dat, 2)
```

```
Ridit Analysis:
Group   Label       Mean Ridit
-----       -----  ----------
1   yes   0.5267
2   no    0.32
Reference: Total of all groups
chi-squared = 32.9167, df = 1, p-value = 9.619e-09
> chisq.test(dat)
    Pearson's Chi-squared test
data:  dat
X-squared = 34.9217, df = 2, p-value = 2.611e-08
> CMHtest(dat)
Cochran-Mantel-Haenszel Statistics for drug by toxicity
             AltHypothesis  Chisq Df      Prob
cor      Nonzero correlation 34.284  1 4.7621e-09
cmeans  Col mean scores differ 34.850  2 2.7066e-08
rmeans  Row mean scores differ 34.284  1 4.7621e-09
general    General association 34.850  2 2.7066e-08
```

例 8.11（程序 8.6）

：R*C，行列皆有序

```
# 行有序或列有序的列联表一般使用 Ridit 分析较为方便。用于对比分组变量（无序）是否随着等级变量（有序）的增加而不同。
# 与 SAS 类似，行、列或皆有序的联列表均可以使用 Cochran-Mantel-Haenszel test 进行统计。vcdExtra 包的 CMHtest()函数可给出全部结果。
# 另外，coin 包下的 cmh_test()和 lbl_test()函数也可分别用于独立资料和有序资料的分析。详见帮助文档。

dat <- matrix(c(4, 9, 39, 147, 11, 37, 22, 94, 143, 317, 182, 139, 411, 1183,
                355,160), nrow= 4)
dimnames(dat) <- list(age=c("5~", "11~", "21~", "41~"), vision=c("<=0.6",
                "0.7~0.9", "1.0~1.2", "1.5"))
# 给矩阵的行列命名

# install.packages("vcdExtra")
library(vcdExtra)
CMHtest(dat)
  # 结果中，cor 为行列皆有序；cmeans 为列有序；rmeans 为行有序；general 为无序（与 Pearson Chi-square 结果相近）
```

结果：

```
> CMHtest(dat)
Cochran-Mantel-Haenszel Statistics for age by vision
             AltHypothesis  Chisq Df      Prob
cor      Nonzero correlation 593.28  1 4.8491e-131
cmeans  Col mean scores differ 783.82  3 1.3995e-169
rmeans  Row mean scores differ 627.51  3 1.0924e-135
general    General association 858.69  9 4.8929e-179
```

```
> chisq.test(dat)
    Pearson's Chi-squared test
data:  dat
X-squared = 858.9587, df = 9, p-value < 2.2e-16
```

例 8.12（程序 8.7）

：方表，一致性检验

```
dat <- matrix(c(160, 5, 26, 48), nrow=2)
dimnames(dat) <- list(fluo=c("+", "-"), regular=c("+", "-"))
# 给矩阵的行列命名

# 输出卡方检验的结果以供对比
chi <- chisq.test(dat)
# 查看方表的理论频数
chi$expected

# 计算 Kappa 值,Kappa()函数位于 vcd 包中
library(vcd)
Kappa(dat)

# McNemar 检验
mcnemar.test(dat, correct=F)
  # 因所有格理论频数均大于 5,故无序连续性校正(correct=F)
```

结果：

```
> chi$expected
   regular
fluo       +       -
  + 128.41004  57.58996
  -  36.58996  16.41004
> Kappa(dat)
         value    ASE     z
Unweighted 0.6708 0.05515 12.16
Weighted   0.6708 0.06596 10.17
> mcnemar.test(dat, correct=F)
    McNemar's Chi-squared test
data:  dat
McNemar's chi-squared = 14.2258, df = 1, p-value = 0.0001621
```

例 9.1

：符号秩和检验

```
a <- c(1, 1, 1, 0, 1, 1, 1, 1, 1)
```

```
b <- c(0, 0, 0, 1, 0, 0, 0, 0, 0)
wilcox.test(a, b, paired=T)
  # paired=T 时为 Wilcoxon 符号秩和检验，即比较差值与 0 的差别
```

结果：

```
> wilcox.test(a,b,paired=T)
    Wilcoxon signed rank test with
    continuity correction
data:  a and b
V = 40, p-value = 0.02341
alternative hypothesis: true location shift is not equal to 0
```

例 9.2

：Wilcoxon 符号秩和检验，配对样本

```
a <- c(336, 258, 371, 291, 386, 300, 364, 285, 377, 298, 292, 303, 288, 312, 304,
    260, 333, 339, 302, 290)
before <- a[seq(1, length(a), 2)]
after <- a[seq(2, length(a), 2)]
# 差值正态性检验
shapiro.test(before-after)
# Wilcoxon 符号秩和检验
wilcox.test(before, after, paired=T)
```

结果：

```
> shapiro.test(before-after)
   Shapiro-Wilk normality test
data:  before - after
W = 0.8212, p-value = 0.0262
> wilcox.test(before,after,paired=T)
   Wilcoxon signed rank test with continuity
   correction
data:  before and after
V = 48, p-value = 0.04136
alternative hypothesis: true location shift is not equal to 0
```

例 9.4（程序 9.3）

：单样本资料，符号秩和检验

```
x <- c(1.40, 2.34, 2.36, 2.34, 1.42, 1.87, 2.42, 2.33, 2.56, 2.54)
shapiro.test(x-1.44)
wilcox.test(x-1.44)
```

结果：

```
> shapiro.test(x-1.44)
    Shapiro-Wilk normality test
data:  x - 1.44
W = 0.7786, p-value = 0.007958
> wilcox.test(x-1.44)
    Wilcoxon signed rank test with continuity correction
data:  x - 1.44
V = 52, p-value = 0.01437
alternative hypothesis: true location is not equal to 0
```

例 9.5（程序 9.4）

：两独立样本秩和检验

```
x <- c(0.32, 0.47, 0.57, 2.21, 0.64, 3.08, 0.67, 2.13)
y <- c(0.26, 0.24, 0.59, 0.37, 0.58, 0.21, 0.33, 0.42, 0.67, 0.45)
shapiro.test(x)
shapiro.test(y)
wilcox.test(x, y)
```

结果：

```
> shapiro.test(x)
    Shapiro-Wilk normality test
data:  x
W = 0.8098, p-value = 0.03643
> shapiro.test(y)
    Shapiro-Wilk normality test
data:  y
W = 0.9378, p-value = 0.5284
> wilcox.test(x,y)
    Wilcoxon rank sum test with continuity correction
data:  x and y
W = 65.5, p-value = 0.02625
alternative hypothesis: true location shift is not equal to 0
```

例 9.6（程序 9.5）

：两独立样本秩和检验

```
height <- c(1:20)
group <- c(1, 1, 2, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1, 2, 2, 1, 2, 2)
wilcox.test(height~ group)
```

结果：

```
> wilcox.test(height~group)
    Wilcoxon rank sum test
data:  height by group
W = 35, p-value = 0.2799
```

```
alternative hypothesis: true location shift is not equal to 0
```

例 9.7（程序 9.6）

：两独立样本秩和检验

```
x <- c(0.004, 0.05, 0.015, 0.4, 0.005, 0.004, 0.025, 0.004, 0.01, 0.05)
y <- c(0.025, 0.05, 0.005, 0.035, 0.1, 0.01, 0.004, 0.015, 0.02, 0.004)
wilcox.test(x, y)
```

结果：

```
> wilcox.test(x, y)
    Wilcoxon rank sum test with continuity correction
data:  x and y
W = 47, p-value = 0.8485
alternative hypothesis: true location shift is not equal to 0
```

例 9.8（程序 9.7）

：等级资料，计数资料，Wilcoxon 检验

```
# 与例 8.9、8.10 类似，可用 Ridit 分析或 CMH test，Wilcoxon 检验并不是最佳。
# Wilcoxon 检验的方法见 9.11
dat <- matrix(c(4, 15, 32, 13, 13, 26, 21, 8), nrow=4)
dimnames(dat) <- list(effect=c("no", "yes", "obvious", "control"),
                      type=c("simple", "gasp"))

# install.packages("Ridit")
library(Ridit)
ridit(dat, 2)
# dat 为 martix 类型列联表，2 表示分组信息在列

chisq.test(dat)
# 输出卡方检验的结果以供对比

# install.packages("vcdExtra")
library(vcdExtra)
CMHtest(dat)
```

结果：

```
> ridit(dat, 2)
Ridit Analysis:
Group   Label      Mean Ridit
-----   -----  ----------
```

```
1   simple     0.5777
2   gasp       0.4269
Reference: Total of all groups
chi-squared = 9.9219, df = 1, p-value = 0.001633
> chisq.test(dat)
    Pearson's Chi-squared test
data:  dat
X-squared = 11.0784, df = 3, p-value = 0.01131
> CMHtest(dat)
Cochran-Mantel-Haenszel Statistics for effect by type
             AltHypothesis   Chisq Df     Prob
cor       Nonzero correlation  9.6417  1 0.0019021
cmeans  Col mean scores differ 10.9945  3 0.0117559
rmeans  Row mean scores differ  9.6417  1 0.0019021
general    General association 10.9945  3 0.0117559
```

例 9.9（程序 9.8）

：完全随机设计，多个独立样本，秩和检验

```
normal <- c(293, 409, 392, 244, 213, 409, 57, 97, 244, 254, 352, 168)
mild <- c(441, 538, 390, 589, 244, 409, 72, 168, 254, 374)
severe <- c(807, 833, 409, 914, 380, 883, 254, 993, 667)
cd8 <- c(normal, mild, severe)
group <- c(rep(1, length(normal)), rep(2, length(mild)), rep(3, length(severe)))
kruskal.test(cd8~group)
  # Kruskal Wallis test
```

结果：

```
> kruskal.test(cd8~group)
    Kruskal-Wallis rank sum test
data:  cd8 by group
Kruskal-Wallis chi-squared = 11.8201, df = 2, p-value = 0.002712
```

例 9.11（程序 9.9）

：等级资料，多个独立样本，秩和检验

```
freq <- c(1, 4, 7, 4,
    3, 6, 9, 7,
    10, 6, 5, 5,
    7, 2, 4, 1)

h <- 1

for(i in 1:4)
{
```

```
 for(j in 1:4)
 {
  for(k in 1:freq[(i-1)*4+j])
  {
   rank[h]=i
   group[h]=j
   no[h]=h
   h=h+1
  }
 }
}
  # 分组，生成变量 rank，赋值 1, 2, 3, 4，代表四种等级

kruskal.test(rank ~ group)
  # Kruskal-Wallis test
```

结果：

```
> kruskal.test(rank~group)
        Kruskal-Wallis rank sum test
data:  rank by group
Kruskal-Wallis chi-squared = 11.6155, df = 3,
p-value = 0.008823
```

例 9.12（程序 9.10）

：随机区组设计，秩和检验

```
# 第一种数据整理方式
cards <- c(48.02, 71.90, 66.27,
     52.70, 56.35, 60.59,
     60.22, 70.08, 66.12,
     44.49, 86.60, 55.36,
     49.31, 68.25, 53.39,
     46.23, 63.36, 52.34,
     55.16, 66.12, 55.16,
     42.48, 70.02, 58.64,
     50.84, 66.97, 44.01,
     39.38, 67.05, 52.49,
     45.16, 69.89, 59.99,
     53.47, 61.08, 61.08)
x <- c()
block <- c()
treat <- c()
h <- 1
for(i in 1:12)
{
```

```
 for(j in 1:3)
 {
  x[h] <- cards[(i-1)*3+j]
  block[h] <- i
  treat[h] <- j
  h <- h+1
 }
}

friedman.test(x ~ treat | block)
     # 明确指明处理因素和区组因素

# 第二种数据整理方式
dat <- matrix(c(48.02, 71.90, 66.27,
         52.70, 56.35, 60.59,
         60.22, 70.08, 66.12,
         44.49, 86.60, 55.36,
         49.31, 68.25, 53.39,
         46.23, 63.36, 52.34,
         55.16, 66.12, 55.16,
         42.48, 70.02, 58.64,
         50.84, 66.97, 44.01,
         39.38, 67.05, 52.49,
         45.16, 69.89, 59.99,
         53.47, 61.08, 61.08), nrow=12, byrow=T,
       dimnames <- list(block=1:12, treat=c("A", "B", "C")))
  # byrow=T 表示按行填充
friedman.test(dat)
  # friedman.test()函数将列矩阵的行作为处理因素，行作为区组因素
```

结果：

```
> friedman.test(dat)
     Friedman rank sum test
data:  dat
Friedman chi-squared = 19.1739, df = 2, p-value = 6.862e-05
```

例 9.13（程序 9.11）

：完全随机，多个样本，秩和检验，多重比较

```
normal <- c(293, 409, 392, 244, 213, 409, 57, 97, 244, 254, 352, 168)
mild <- c(441, 538, 390, 589, 244, 409, 72, 168, 254, 374)
severe <- c(807, 833, 409, 914, 380, 883, 254, 993, 667)
cd8 <- c(normal, mild, severe)
group <- as.factor(c(rep(1, length(normal)), rep(2, length(mild)), rep(3,
length(severe))))
kruskal.test(cd8~group)
```

```
    # Kruskal Wallis test

# install.packages("pgirmess")
# pgirmess 包中含有 kruskalmc()、friedmanmc()等非参数检验多重比较函数
library(pgirmess)
kruskalmc(cd8~group, probs=0.01)
  # probs=0.01 不写则默认以 0.05 计算

# coin 包中的 oneway_test 函数可做 Nemenyi-Damico-Wolfe-Dunn test
# install.packages("coin")
# install.packages("multcomp")
library(coin)
library(multcomp)
cd = data.frame(cd8, group)
mult <- oneway_test(cd8 ~ group, data = cd,
          ytrafo = function(data) trafo(data, numeric_trafo = rank),
          xtrafo = function(data) trafo(data, factor_trafo = function(x)
            model.matrix(~x - 1) %*% t(contrMat(table(x), "Tukey")))),
          teststat = "max", distribution = approximate(B = 90000))
pvalue(mult, method = "single-step")
```

结果：

```
> kruskalmc(cd8~group, probs=0.01)
Multiple comparison test after Kruskal-Wallis
p.value: 0.01
Comparisons
    obs.dif critical.dif difference
1-2  4.375000    11.42677      FALSE
1-3 13.652778    11.76794       TRUE
2-3  9.277778    12.26192      FALSE
> pvalue(mult, method = "single-step")
2 - 1 0.4118444444
3 - 1 0.0008333333
3 - 2 0.0792666667
```

例 9.14（程序 9.12）

：完全随机，多个样本，秩和检验，多重比较

```
freq <- c(1, 4, 7, 4,
    3, 6, 9, 7,
    10, 6, 5, 5,
    7, 2, 4, 1)

h <- 1

for(i in 1:4)
```

```
{
 for(j in 1:4)
 {
  for(k in 1:freq[(i-1)*4+j])
  {
   rank[h]=i
   group[h]=j
   no[h]=h
   h=h+1
  }
 }
}
  # 分组，生成变量 rank，赋值 1, 2, 3, 4，代表四种等级

kruskal.test(rank ~ group)
  # Kruskal-Wallis test

# install.packages("pgirmess")
# pgirmess 包中含有 kruskalmc()、friedmanmc()等非参数检验多重比较函数
library(pgirmess)
kruskalmc(rank ~ group, probs=0.05)
```

检验：

```
> kruskalmc(rank~group, probs=0.05)
Multiple comparison test after Kruskal-Wallis
p.value: 0.05
Comparisons
    obs.dif critical.dif difference
1-2 17.333333    19.93712      FALSE
1-3 19.693333    18.37279       TRUE
1-4 20.980392    20.25040       TRUE
2-3  2.360000    19.18686      FALSE
2-4  3.647059    20.99178      FALSE
3-4  1.287059    19.51220      FALSE
```

例 9.15（程序 9.13）

：随机区组设计，多个样本，秩和检验，多重比较

```
# 第一种数据整理方式
cards <- c(48.02, 71.90, 66.27,
      52.70, 56.35, 60.59,
      60.22, 70.08, 66.12,
      44.49, 86.60, 55.36,
      49.31, 68.25, 53.39,
      46.23, 63.36, 52.34,
      55.16, 66.12, 55.16,
      42.48, 70.02, 58.64,
```

```
      50.84, 66.97, 44.01,
      39.38, 67.05, 52.49,
      45.16, 69.89, 59.99,
      53.47, 61.08, 61.08)
x <- c()
block <- c()
treat <- c()
h <- 1
for(i in 1:12)
{
 for(j in 1:3)
 {
  x[h] <- cards[(i-1)*3+j]
  block[h] <- i
  treat[h] <- j
  h <- h+1
 }
}

friedman.test(x ~ treat | block)
      # 明确指明处理因素和区组因素
# install.packages("pgirmess")
# pgirmess 包中含有 kruskalmc()、friedmanmc()等非参数检验多重比较函数
library(pgirmess)
friedmanmc(x, treat, block, probs=0.05)

# 第二种数据整理方式
dat <- matrix(c(48.02, 71.90, 66.27,
         52.70, 56.35, 60.59,
         60.22, 70.08, 66.12,
         44.49, 86.60, 55.36,
         49.31, 68.25, 53.39,
         46.23, 63.36, 52.34,
         55.16, 66.12, 55.16,
         42.48, 70.02, 58.64,
         50.84, 66.97, 44.01,
         39.38, 67.05, 52.49,
         45.16, 69.89, 59.99,
         53.47, 61.08, 61.08), nrow=12, byrow=T,
        dimnames <- list(block=1:12, treat=c("A", "B", "C")))
  # byrow=T 表示按行填充
friedman.test(dat)
  # friedman.test()函数将列矩阵的列作为处理因素，行作为区组因素
friedmanmc(dat)
```

结果：

```
> friedmanmc(dat)
Multiple comparisons between groups after Friedman test
p.value: 0.05
```

```
Comparisons
  obs.dif critical.dif difference
1-2    21.0     11.72806       TRUE
1-3    10.5     11.72806      FALSE
2-3    10.5     11.72806      FALSE
```

例 9.16（程序 9.14）

：等级资料，Ridit 分析

```
dat <- matrix(c(76, 4, 56, 7, 62, 20, 125, 49), nrow=2,
       dimnames=list(group=c("A", "B"), effect=c("no", "little", "obvious",
                                 "complete")))
# install.packages("Ridit")
library(Ridit)
ridit(dat, 1)
# dat 为 martix 类型列联表，1 表示分组信息在行(列有序)（若 2 则表示在列，即为行有序）
# ridit()要求矩阵行列必须有名字
chisq.test(dat)
```

结果：

```
> ridit(dat, 1)
Ridit Analysis:
Group   Label       Mean Ridit
-----        -----  ----------
1    A       0.469
2    B       0.6237
Reference: Total of all groups
chi-squared = 20.4459, df = 1, p-value = 6.134e-06
> chisq.test(dat)
    Pearson's Chi-squared test
data:  dat
X-squared = 22.5488, df = 3, p-value = 5.014e-05
```

例 9.17（程序 9.15）

：等级资料，Ridit 分析

```
dat <- matrix(c(76, 4, 56, 7, 62, 20, 125, 49), nrow=2,
       dimnames=list(group=c("A", "B"), effect=c("no", "little", "obvious",
                                 "complete")))
# install.packages("Ridit")
library(Ridit)
ridit(dat, 1)
# dat 为 martix 类型列联表，1 表示分组信息在行(列有序)（若 2 则表示在列，
```

```
即为行有序）
# ridit()要求矩阵行列必须有名字
chisq.test(dat)
```

结果：

```
> ridit(dat, 1)
Ridit Analysis:
Group   Label       Mean Ridit
-----   -----  ----------
1   A       0.4261
2   B       0.5739
Reference: Total of all groups
chi-squared = 14.39, df = 1, p-value = 0.0001486
```

例 9.18（程序 9.16）

：等级资料，多个样本，Ridit 分析

```
dat <- matrix(c(65, 18, 30, 13, 77, 16, 36, 18, 42, 6, 23, 11, 94, 11, 47, 36),
        nrow=4,  dimnames=list(effect=c("no",  "little",  "obvious",  "complete"),
        group=1:4))
# install.packages("Ridit")
library(Ridit)
ridit(dat, 2)
# dat 为 martix 类型列联表，2 表示分组信息在列(行有序)（若 1 则表示在行，
即为列有序）
# ridit()要求矩阵行列必须有名字
chisq.test(dat)
```

结果：

```
> ridit(dat, 2)
Ridit Analysis:
Group   Label       Mean Ridit
-----   -----  ----------
1   1       0.4817
2   2       0.4876
3   3       0.5016
4   4       0.5212
Reference: Total of all groups
chi-squared = 2.113, df = 3, p-value = 0.5493
```

例 9.19（程序 9.17）

：秩相关

```
cards <- c(1.1, 41.2, 2.3, 37.6,
```

```
    2.5, 38.8, 3.7, 24.3,
    3.8, 23.0, 4.0, 32.4,
    4.6, 16.1, 4.9, 18.4,
    7.6, 7.1, 8.1, 8.0,
    8.2, 9.1, 8.4, 4.2,
    8.8, 5.3, 18.7, 0.2, 23.5, 0.0)
x <- cards[seq(1, length(cards), 2)]
y <- cards[seq(2, length(cards), 2)]
cor.test(x, y, method="spearman")
  # Spearman 秩相关
```

结果：

```
> cor.test(x, y, method="spearman")
    Spearman's rank correlation rho
data:  x and y
S = 1100, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
-0.9642857
```

例 10.1-2

：判别分析

```
library(gdata)
dat <- read.xls("EYE1.xls", sheet=1)

library(MASS)
fit <- lda(group~age+vision+at+bt+qpv, data=dat)
  # lda()基于 Bayes 判别
fit

# 判别效果显著性检验，多元统计量
vars <- as.matrix(data.frame(dat$age, dat$vision, dat$at, dat$bt, dat$qpv))
  # 生成只含有训练样本的矩阵，用于多元分析
tr.manova <- manova(vars~dat$group)
summary(tr.manova, test="Wilks")
  # 输出 Wilk's Lambda 结果
summary(tr.manova)
  # 输出 Pillai's Trace 结果

# 进行组内考核
fit.id <- predict(fit)$class
  # 选取 predict 函数输出结果的 class 项
table(fit.id, dat$group)
  # 输出频数表，用于对比
```

```
# 对组外样本进行前瞻性考核
dat2 <- read.xls("EYE2.xls", sheet=1)
id2 <- predict(fit, dat2)$class
table(id2, dat2$group)
```

结果：

```
> fit
Call:
lda(group ~ age + vision + at + bt + qpv, data = dat)

Prior probabilities of groups:
     A1        A2        A3
0.5190840 0.3282443 0.1526718
Group means:
       age   vision       at       bt      qpv
A1 53.63235 1.0323529 13.44853 50.56985 37.74132
A2 64.48837 0.5837209 14.09302 51.34884 25.10000
A3 53.60000 0.4145000 15.92500 56.12500 16.86650
Coefficients of linear discriminants:
               LD1         LD2
age    -0.060922339 -0.07135878
vision -3.941019324  2.09928708
at      0.527380988  0.75293913
bt      0.008976821  0.02524835
qpv    -0.025218271  0.01019541
Proportion of trace:
  LD1    LD2
0.8253 0.1747
> summary(tr.manova, test="Wilks")
          Df   Wilks approx F num Df den Df    Pr(>F)
dat$group   2 0.19559   31.276     10    248 < 2.2e-16 ***
Residuals 128
> summary(tr.manova)
          Df Pillai approx F num Df den Df    Pr(>F)
dat$group   2 1.0416   27.169     10    250 < 2.2e-16 ***
Residuals 128
> table(fit.id, dat$group)
fit.id A1 A2 A3
   A1 63  8  1
   A2  4 35  1
   A3  1  0 18
> dat2 = read.xls("EYE2.xls", sheet=1)
> id2 = predict(fit, dat2)$class
> table(id2, dat2$group)
id2  A1 A2 A3
  A1 14  3  0
  A2  1  8  1
  A3  0  0  4
```

例 10.3

：逐步判别分析

```
group <- dat$group
vars <- as.matrix(dat[ , 1:(ncol(dat)-1)])
  # 生成只含有训练样本的矩阵，供后续使用

# SAS 中使用的方法为 Wilk's Lambda criterion，此处也使用此方法
# install.packages("klaR")
library(klaR)
greedy.wilks(vars, group)
  # 获得与 SAS 完全一样的结果
```

结果：

```
> greedy.wilks(vars, group)
Formula containing included variables:
group ~ vision + at + age + bv + qpv
<environment: 0x7fa54bca6d80>
Values calculated in each step of the selection procedure:
   vars Wilks.lambda F.statistics.overall p.value.overall F.statistics.diff p.value.diff
1 vision    0.4047650          94.11645    7.260843e-26         94.116448 7.260843e-26
2     at    0.3068206          51.13875    1.506579e-31         20.270686 2.250196e-08
3    age    0.2149055          48.59945    2.025339e-39         26.945124 1.760654e-10
4     bv    0.1813867          42.12487    3.025076e-42         11.549499 2.479915e-05
5    qpv    0.1708351          35.20164    3.398676e-42          3.829406 2.431472e-02
```

例 11.1

：危险度，RR

```
dat <- matrix(c(27, 44, 95, 443), nrow=2)
dimnames(dat) <- list(c("high", "low"), c("occur", "no"))

# install.packages("epitools")
library(epitools)
riskratio(dat, rev="both")
    # riskratio()默认的数据排列方式如下：
                     disease=0   disease=1
     exposed=0 (ref)    n00         n01
     exposed=1          n10         n11
     exposed=2          n20         n21
     exposed=3          n30         n31
    # 即第一行暴露最低（对照），第一列为对照
    # 因此需要用 rev="both"，表示行列顺序皆相反

# 为获得和 SAS 相同的数据，使用以下步骤
# install.packages("vcdExtra")
library(vcdExtra)
  # 可能需要手动启动 XQuartz，或直接通过 R 运行命令
CMHtest(dat)

chisq.test(dat)
```

结果：

```
RR = 2.449516

> riskratio(dat, rev="b")
$data
     no occur Total
low   443   44   487
high   95   27   122
Total 538   71   609

$measure
                 NA
risk ratio with 95% C.I.    estimate   lower    upper
               low  1.000000      NA      NA
               high  2.449516  1.583699  3.788679
$p.value
      NA
two-sided   midp.exact fisher.exact   chi.square
    low          NA          NA          NA
    high 0.0001822103  0.0002048979  5.561345e-05
$correction
[1] FALSE
attr(,"method")
[1] "Unconditional MLE & normal approximation (Wald) CI"
> CMHtest(dat)
Cochran-Mantel-Haenszel Statistics
             AltHypothesis Chisq Df      Prob
cor       Nonzero correlation 16.22  1 5.6402e-05
cmeans  Col mean scores differ 16.22  1 5.6402e-05
rmeans  Row mean scores differ 16.22  1 5.6402e-05
general    General association 16.22  1 5.6402e-05
> chisq.test(dat)
     Pearson's Chi-squared test with Yates' continuity
     correction
data:  dat
X-squared = 14.9998, df = 1, p-value = 0.0001075
```

例 11.2

：比数比，优势比，OR

```
dat <- matrix(c(55, 128, 19, 164), nrow=2)
dimnames(dat) <- list(c("use","no-use"), c("case", "control"))
# install.packages("epitools")
library(epitools)
# oddsratio(dat, rev="both")
oddsratio(dat, rev="both", method= "wald")
  # oddsratio()默认使用 median-unbiased estimation (mid-p)的 method，计计算结果和 SAS 不一致； method = "wald"(unconditional maximum likelihood estimation) 的选项使结果一致。
```

```
# install.packages("vcdExtra")
library(vcdExtra)
  # 可能需要手动启动 XQuartz，或直接通过 R 运行命令
CMHtest(dat)
chisq.test(dat)
```

结果：

```
> oddsratio(dat, rev="both", method = "wald")
$data
       control case Total
no-use     164  128   292
use         19   55    74
Total      183  183   366
$measure
                NA
odds ratio with 95% C.I.   estimate    lower    upper
                 no-use  1.000000       NA       NA
                 use     3.708882  2.096434  6.561524
$p.value
         NA
two-sided   midp.exact fisher.exact   chi.square
  no-use           NA           NA           NA
  use     2.334257e-06 3.716595e-06 2.795744e-06
$correction
[1] FALSE
attr(,"method")
[1] "Unconditional MLE & normal approximation (Wald) CI"
> CMHtest(dat)
Cochran-Mantel-Haenszel Statistics
                 AltHypothesis  Chisq Df       Prob
cor        Nonzero correlation 21.892  1 2.8845e-06
cmeans  Col mean scores differ 21.892  1 2.8845e-06
rmeans  Row mean scores differ 21.892  1 2.8845e-06
general    General association 21.892  1 2.8845e-06
```

例 11.4（程序 11.3）

：分层分析

```
dat <- array(c(4, 2, 62, 224,
     9, 12, 33, 390,
     4, 33, 26, 330,
     6, 65, 9, 362,
     6, 93, 5, 301),
    dim= c(2, 2, 5),
    dimnames= list(
     OC= c("taking", "none"),
     group= c("case", "control"),
     age= c("25~", "30~", "35~", "40~", "45~")))
mantelhaen.test(dat, correct= F)
  # 若 correct = T 则进行连续性矫正，结果与 SAS 不同
```

结果：

```
> mantelhaen.test(dat, correct=F)
    Mantel-Haenszel chi-squared test without continuity
    correction
data:  dat
Mantel-Haenszel X-squared = 34.723, df = 1, p-value =3.801e-09
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 2.426983 6.493688
sample estimates:
common odds ratio
       3.969895
```

例 11.5（程序 11.4a）

：Logistic 回归

```
library(gdata)
dat <- read.xls("table11_4a.xls", sheet=1)

fit <- glm(y~age+sex+ECG, data=dat, family=binomial)
  # 因 y 值为二分类变量，family=binomiall 表示二项分布
summary(fit)

# 计算比数比 OR
coef <- as.numeric(fit$coefficients)
age.coef <- coef[2]
sex.coef <- coef[3]
age.OR <- exp(age.coef)
age.OR
sex.OR <- exp(sex.coef)
sex.OR
```

结果：

```
> summary(fit)
Call:
glm(formula = y ~ age + sex + ECG, family = binomial, data = dat)
Deviance Residuals:
   Min       1Q   Median       3Q      Max
-2.0671  -0.9589   0.3285   0.9341   1.9914
Coefficients:
         Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.64176    1.80614  -3.124  0.00179 **
age          0.09285    0.03509   2.646  0.00815 **
sex          1.35643    0.54645   2.482  0.01306 *
ECG          0.87320    0.38433   2.272  0.02309 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 1)
   Null deviance: 107.926  on 77  degrees of freedom
```

```
Residual deviance:  86.811  on 74  degrees of freedom
AIC: 94.811
Number of Fisher Scoring iterations: 4
> age.OR
[1] 1.097299
> sex.OR
[1] 3.882291
```

例 11.5（程序 11.4b、c）

：Logistic 回归

```
library(gdata)
dat <- read.xls("table11_4b.xls", sheet=1)

# 因 agegroup 是否有序未能确定，而资料中表示为 1，2，3，故需生成哑变量 agegroup1 和 agegroup2，将 3 分类变量转化为两个二分类变量
dat$agegroup1 <- ifelse(dat$agegroup == 2, 1, 0)
dat$agegroup2 <- ifelse(dat$agegroup == 3, 1, 0)
  # ifelse()用于将某个变量转换为二分类变量，此例中，TRUE 取 1，FALSE 取 2

# 生成 sex 和哑变量的交互项
dat$sexage1 <- dat$sex*dat$agegroup1
dat$sexage2 <- dat$sex*dat$agegroup2

# For binomial families the response can also be specified as a factor (when the first level denotes failure and all others success)
# 即 0 表示 failure，1 表示 success
# 若为连续资料或多分类变量，则应与程度成正比
# 本例中，拟合的模型为 P(y=1 | x)
fit <- glm(y~sex+ECG+agegroup1+agegroup2+sexage1+sexage2, data=dat,
family=binomial)
summary(fit)

# 因 agegroup2 的 Estimate（2.14610）比 agegroup1 的（1.21904）高，故考虑 agegroup 可以等级变量进入模型
# 同时 sexage1 和 sexage2 均无统计学意义，故不考虑 sex 和 agegroup 的交互作用
# 重新拟合模型
fit <- glm(y~agegroup+sex+ECG, data=dat, family=binomial)
summary(fit)

# 计算 Odds Ratio（OR）及其可信区间
exp(cbind(coef(fit), confint.default(fit)))
  # coef()用于获取 Estimate 值，confint.default()用于获取 95%可信区间
```

结果：

```
> summary(fit)
Call:
glm(formula = y ~ sex + ECG + agegroup1 + agegroup2 + sexage1 +
  sexage2, family = binomial, data = dat)
Deviance Residuals:
  Min      1Q   Median      3Q     Max
-1.8823  -0.9771   0.2870   0.9330   1.9147
Coefficients:
         Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.63729    1.12137  -2.352   0.0187 *
sex         1.73394    1.25703   1.379   0.1678
ECG         0.97857    0.38986   2.510   0.0121 *
agegroup1    1.21904    1.26024   0.967   0.3334
agegroup2    2.14610    1.23039   1.744   0.0811 .
sexage1     -0.68782    1.49514  -0.460   0.6455
sexage2     -0.03067    1.59442  -0.019   0.9847
> fit = glm(y~agegroup+sex+ECG, data=dat, family=binomial)
> summary(fit)
Call:
glm(formula = y ~ agegroup + sex + ECG, family = binomial, data = dat)
Deviance Residuals:
  Min      1Q   Median      3Q     Max
-1.9865  -0.9846   0.3356   0.8399   1.8642
Coefficients:
         Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.5322     1.0219  -3.457 0.000547 ***
agegroup      1.0253     0.3697   2.774 0.005544 **
sex           1.3796     0.5488   2.514 0.011946 *
ECG           0.9628     0.3872   2.486 0.012913 *
> exp(cbind(coef(fit), confint(fit)))
                       2.5 %     97.5 %
(Intercept) 0.02923923 0.003241532  0.1866566
agegroup    2.78784708 1.398662600  6.0522849
sex         3.97332389 1.402508991 12.3080381
ECG         2.61889522 1.269533886  5.8920758
```

例 11.6（程序 11.5）

：频数资料，Logistic 回归

```
weight <- rep(c(750, 1150, 1550), c(68, 80, 75))
bpd <- rep(c(1, 0, 1, 0, 1, 0), c(49, 19, 18, 62, 9, 66))
fit <- glm(bpd~weight, family=binomial)
summary(fit)
exp(cbind(coef(fit), confint.default(fit)))
```

结果：

```
> summary(fit)
Call:
glm(formula = bpd ~ weight, family = binomial)
Deviance Residuals:
```

```
   Min      1Q   Median      3Q     Max
-1.5029  -0.8437  -0.4091   0.8836   2.2460
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.7179768  0.6386949   5.821 5.84e-09 ***
weight      -0.0039720  0.0005881  -6.754 1.44e-11 ***
> exp(cbind(coef(fit), confint.default(fit)))
Waiting for profiling to be done...
                          2.5 %      97.5 %
(Intercept) 41.1809937 12.2904102 151.8304310
weight       0.9960359  0.9948284   0.9971333
```

例 11.7（程序 11.6a）

：条件 Logistic 回归，分层，应用 Cox 回归思想

id	y	x1	x2	x3
1	1	1	3	0
2	1	0	3	1
3	1	0	1	2
4	1	1	2	0
5	1	1	1	1
6	1	0	2	2
7	1	1	1	1
8	1	1	1	2
9	1	3	3	2
10	1	2	2	2
1	0	1	0	1
2	0	1	3	0
3	0	0	2	0
4	0	1	0	0
5	0	1	2	1
6	0	2	0	0
7	0	0	0	0
8	0	0	0	0
9	0	2	2	0
10	0	0	0	0

整理数据如以上表格，命名为 x.csv

```
dat <- read.csv("x.csv")
library(survival)
fit <- clogit(y~x1+x2+x3+strata(id), data=dat)
      # cologit()函数是 Cox 回归的一种实现；
      # strata()指明 id 为分层变量
summary(fit)
```

结果：

```
> summary(fit)
Call:
coxph(formula = Surv(rep(1, 20L), y) ~ x1 + x2 + x3 + strata(id),
   data = dat, method = "exact")
 n= 20, number of events= 10
     coef exp(coef) se(coef)      z Pr(>|z|)
x1 -0.4790    0.6194   2.9548 -0.162    0.871
x2  1.2318    3.4274   0.8348  1.476    0.140
x3  2.2899    9.8735   1.7681  1.295    0.195
   exp(coef) exp(-coef) lower .95 upper .95
x1    0.6194     1.6145  0.001891     202.8
x2    3.4274     0.2918  0.667444      17.6
x3    9.8735     0.1013  0.308662     315.8
Rsquare= 0.393   (max possible= 0.5 )
Likelihood ratio test= 9.98  on 3 df,   p=0.01876
Wald test            = 2.59  on 3 df,   p=0.4592
Score (logrank) test = 6.91  on 3 df,   p=0.07472
```

例 11.7（程序 11.6b）

：条件 Logistic 回归，分层，使用差值做回归

```
# 数据如程序 11.6a
dat <- read.csv("x.csv")
dat2 <- dat[1:10, 2:5]-dat[11:20, 2:5]
fit <- glm(y~x1+x2+x3+0, data=dat2, family=binomial)
# fit <- glm(y~x1+x2+x3-1, data=dat2, family=binomial)
  # model 中+0 或-1 均表示拟合无截距项的模型
summary(fit)
exp(cbind(coef(fit), confint.default(fit)))
```

结果：

```
> summary(fit)
Call:
glm(formula = y ~ x1 + x2 + x3 + 0, family = binomial, data = dat2)
Deviance Residuals:
   Min       1Q   Median       3Q      Max
0.02588  0.09819  0.28410  0.39038  1.72499
Coefficients:
  Estimate Std. Error z value Pr(>|z|)
x1  -0.4790     2.9548  -0.162    0.871
x2   1.2318     0.8348   1.476    0.140
```

```
x3   2.2899    1.7681   1.295    0.195
(Dispersion parameter for binomial family taken to be 1)
   Null deviance: 13.8629  on 10  degrees of freedom
Residual deviance:  3.8862  on  7  degrees of freedom
AIC: 9.8862
Number of Fisher Scoring iterations: 7
> exp(cbind(coef(fit), confint.default(fit)))
              2.5 %    97.5 %
x1 0.6193767 0.001891488 202.81788
x2 3.4273598 0.667444154  17.59967
x3 9.8734675 0.308664251 315.82977
```

例 12.1

：生存曲线，生存函数，Logrank 检验，乘积极限法，Kaplan-Meier

group	month	death
zhongyao	10	0
zhongyao	2	1
zhongyao	12	1
zhongyao	13	0
zhongyao	18	0
zhongyao	6	1
zhongyao	19	1
zhongyao	26	0
zhongyao	9	1
zhongyao	8	1
zhongyao	6	1
zhongyao	43	1
zhongyao	9	0
zhongyao	4	0
zhongyao	31	0
zhongyao	24	0
control	2	1
control	13	0
control	7	1
control	11	1
control	6	0
control	1	0

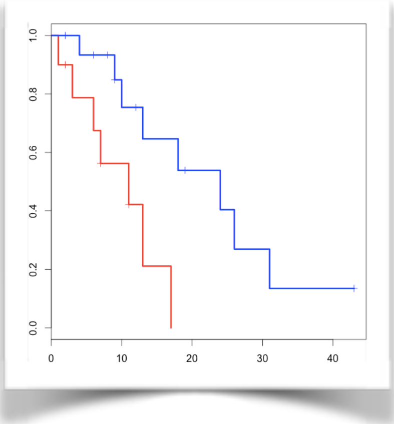
control	11	0
control	3	0
control	17	0
control	7	0

```
# 整理数据如以上表格，存储为 ch12.1.csv
library(survival)
dat <- read.csv("ch12.1.csv")
fit <- survfit(Surv(month, alive == 0) ~ group, data = dat)
  # alive ==0 用于指明完全数据（死亡）的取值，此例中 1 表示截尾数据（censor）
  # 可在原始数据中，直接用 T 表示完全数据，F 表示截尾数据
summary(fit)

# 画出生存曲线
plot(fit, col=c("red", "blue"), lwd=3)

# 比较两条生存曲线的差异
survdiff(Surv(month, alive == 0) ~ group, data = dat)
  # 此为经典 Log-Rank test，为渐进检验（asymptotic tests），未考虑条件方差（conditional variance）
# coin 包的 surv_test()可做精确检验
library(coin)
surv_test(Surv(month, alive == 0) ~ group, data = dat, distribution = exact())
  # 也可以写 distribution = "exact"
```

结果：

```
> summary(fit)
Call: survfit(formula = Surv(month, alive == 0) ~ group, data = dat)

                group=control
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    1     10       1    0.900  0.0949       0.7320        1.000
    3      8       1    0.787  0.1340       0.5641        1.000
    6      7       1    0.675  0.1551       0.4303        1.000
    7      6       1    0.562  0.1651       0.3165        1.000
   11      4       1    0.422  0.1737       0.1883        0.945
   13      2       1    0.211  0.1726       0.0424        1.000
   17      1       1    0.000     NaN           NA           NA
                group=zhongyao
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    4     15       1    0.933  0.0644       0.8153        1.000
    9     11       1    0.848  0.0999       0.6737        1.000
   10      9       1    0.754  0.1256       0.5441        1.000
   13      7       1    0.646  0.1468       0.4143        1.000
   18      6       1    0.539  0.1570       0.3043        0.954
   24      4       1    0.404  0.1657       0.1808        0.903
   26      3       1    0.269  0.1559       0.0866        0.837
   31      2       1    0.135  0.1231       0.0225        0.807
```

```
> survdiff(Surv(month, alive == 0) ~ group, data = dat)
Call:
survdiff(formula = Surv(month, alive == 0) ~ group, data = dat)
             N Observed Expected (O-E)^2/E (O-E)^2/V
group=control  10        7     3.21      4.47      6.58
group=zhongyao 16        8    11.79      1.22      6.58
> surv_test(Surv(month, alive == 0) ~ group, data = dat, distribution = exact())
     Exact Logrank Test
data:  Surv(month, alive == 0) by group (control, zhongyao)
Z = 2.1418, p-value = 0.02694
alternative hypothesis: two.sided
```



例 12.3（程序 12.2）

：生存函数，寿命表法，life table

```
dat2 <- c(59, 63, 69, 71, 43, 55, 30, 38, 13, 31, 7, 26, 14, 21, 4, 11, 3, 15, 2, 6, 1, 3)
death.no <- dat2[seq(1, length(dat2), 2)]
censor.no <- dat2[seq(2, length(dat2), 2)]
initial.no <- 585
  # 初入组人数

# KMsurv 包中含有 lifetab()函数可用于寿命表计算
# install.packages("KMsurv")
library(KMsurv)
years <- c(seq(0, 10, 1), NA)
  # lifetab()函数要求 tis（时间间距）选项长度比其他数据多 1 个，用 NA 代替
lifetab(tis=years, ninit=initial.no, nlost=censor.no, nevent=death.no)
```

结果：

```
> lifetab(tis=years, ninit=initial.no, nlost=censor.no, nevent=death.no)
    nsubs nlost nrisk nevent      surv        pdf     hazard    se.surv     se.pdf  se.hazard
0-1   585    63 553.5     59 1.0000000 0.10659440 0.11259542 0.00000000 0.01311695 0.01463543
1-2   463    71 427.5     69 0.8934056 0.14419880 0.17557252 0.01311695 0.01603730 0.02105485
2-3   323    55 295.5     43 0.7492068 0.10902163 0.15693431 0.01933155 0.01562379 0.02385847
3-4   225    38 206.0     30 0.6401852 0.09323085 0.15706806 0.02256215 0.01607281 0.02858800
4-5   157    31 141.5     13 0.5469543 0.05025022 0.09629630 0.02488209 0.01347657 0.02667681
5-6   113    26 100.0      7 0.4967041 0.03476929 0.07253886 0.02621023 0.01280537 0.02739907
6-7    80    21  69.5     14 0.4619348 0.09305162 0.22400000 0.02747321 0.02290228 0.05948985
7-8    45    11  39.5      4 0.3688832 0.03735526 0.10666667 0.03122833 0.01798687 0.05325743
```

```
8-9     30   15  22.5     3 0.3315279 0.04420372 0.14285714 0.03318472 0.02416728 0.08226794
9-10     12    6   9.0     2 0.2873242 0.06384982 0.25000000 0.03730446 0.04067117 0.17539019
10-NA    4    3   2.5     1 0.2234744         NA         NA 0.04926731         NA         NA
```

例 12.4（程序 12.3）

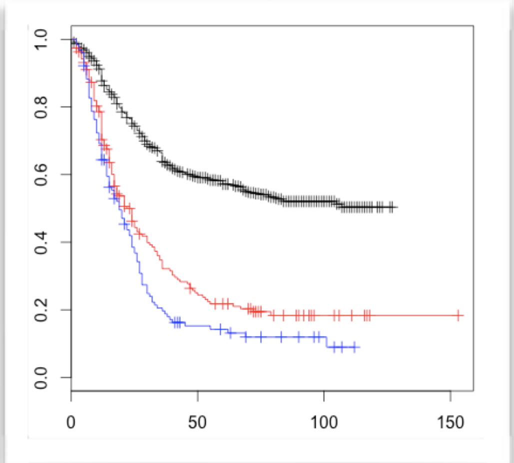
：生存函数，寿命表法，life table，原始数据转频数表

```
library(gdata)
dat <- read.xls("life.xls", sheet=1)
interval <- floor(dat$y/12)*12
  # floor()表示向下取整
dat <- data.frame(dat, interval)
dat.list <- split(dat, dat$x10)
  # 根据 x10 的取值，将数据分成三组，存储于清单 dat.list
dat0 <- dat.list[[1]]
  # list 类型定位第一个 list 需用[[ ]]
dat1 <- dat.list[[2]]
dat2 <- dat.list[[3]]
table0 <- data.frame(table(dat0$interval, dat0$censor))
  # 生成频数表。第一个为行变量，第二个为列变量
table1 <- data.frame(table(dat1$interval, dat1$censor))
table2 <- data.frame(table(dat2$interval, dat2$censor))

# 生成 life table
library(KMsurv)
lifetab(tis=seq(0, max(as.numeric(table0[,1]))*12, 12),
     ninit=sum(table0[ ,3]),
     nlost=table0[(length(table0[,3])/2+1):length(table0[,3]), 3],
     nevent=table0[1:(length(table0[,3])/2), 3])

lifetab(tis=c(seq(0, max(as.numeric(table1[,1]))*11, 12), max(interval)),
     ninit=sum(table1[ , 3]),
     nlost=table1[(length(table1[, 3])/2+1):length(table1[, 3]), 3],
     nevent=table1[1:(length(table1[, 3])/2), 3])

lifetab(tis=seq(0, max(as.numeric(table2[, 1]))*12, 12),
     ninit=sum(table2[ , 3]),
     nlost=table2[(length(table2[, 3])/2+1):length(table2[, 3]), 3],
     nevent=table2[1:(length(table2[, 3])/2), 3])

# 比较生存函数，并作图
library(survival)
survdiff(Surv(y, censor == 0) ~ x10, data = dat)
plot(survfit(Surv(y, censor == 0) ~ x10, data = dat), col=c("black", "red", "blue"))
```

结果：

```
x10=0
      nsubs nlost nrisk nevent      surv          pdf      hazard    se.surv
0-12      684    27 670.5     59 1.0000000 0.0073328362 0.007670307 0.00000000
12-24     598    17 589.5     95 0.9120060 0.0122477476 0.014606396 0.01094022
24-36     486    24 474.0     62 0.7650330 0.0083389672 0.011662904 0.01658182
36-48     400    17 391.5     39 0.6649654 0.0055201469 0.008736559 0.01865782
48-60     344    41 323.5     10 0.5987236 0.0015423071 0.002616431 0.01958358
60-72     293    41 272.5     17 0.5802159 0.0030164131 0.005366162 0.01983349
72-84     235    67 201.5      7 0.5440190 0.0015749102 0.002946128 0.02044703
84-96     161    54 134.0      2 0.5251201 0.0006531344 0.001253133 0.02094730
96-108    105    59  75.5      2 0.5172824 0.0011419039 0.002237136 0.02135521
108-120    44    36  26.0      0 0.5035796 0.0000000000 0.000000000 0.02288232
120-132     8     8   4.0      0 0.5035796           NA          NA 0.02288232
           se.pdf    se.hazard
0-12    0.0009116850 0.0009975308
12-24   0.0011602349 0.0014928186
24-36   0.0010037678 0.0014775593
36-48   0.0008529300 0.0013970457
48-60   0.0004827660 0.0008272862
60-72   0.0007158647 0.0013008106
72-84   0.0005878172 0.0011133577
84-96   0.0004591161 0.0008860737
96-108  0.0007980750 0.0015817519
108-120          NaN          NaN
120-132           NA           NA

x10=1
      nsubs nlost nrisk nevent      surv         pdf      hazard    se.surv
0-12      191     8 187.0     40 1.0000000 0.017825312 0.019960080 0.00000000
12-24     143    13 136.5     50 0.7860963 0.023995612 0.037369208 0.02998653
24-36      80     2  79.0     24 0.4981489 0.012611365 0.029850746 0.03757580
36-48      54     1  53.5     13 0.3468125 0.007022684 0.023049645 0.03672515
48-60      40     1  39.5      7 0.2625403 0.003877178 0.016203704 0.03444502
60-72      32     6  29.0      2 0.2160142 0.001241461 0.005952381 0.03252140
72-84      24     7  20.5      2 0.2011167 0.001635095 0.008547009 0.03193909
84-96      15     7  11.5      0 0.1814955 0.000000000 0.000000000 0.03169361
96-108      8     3   6.5      0 0.1814955 0.000000000 0.000000000 0.03169361
108-120     5     4   3.0      0 0.1814955 0.000000000 0.000000000 0.03169361
120-144     1     1   0.5      0 0.1814955          NA          NA 0.03169361
           se.pdf   se.hazard
0-12    0.0024988777 0.003133252
12-24   0.0028522619 0.005150251
24-36   0.0023491778 0.005994730
36-48   0.0018506456 0.006331391
48-60   0.0014232668 0.006095411
60-72   0.0008674101 0.004206284
72-84   0.0011286179 0.006035696
84-96            NaN         NaN
96-108           NaN         NaN
108-120          NaN         NaN
120-144           NA          NA

x10=2
      nsubs nlost nrisk nevent       surv          pdf      hazard    se.surv
0-12      128     2 127.0     39 1.00000000 0.0255905512 0.030232558 0.00000000
12-24      87     5  84.5     33 0.69291339 0.0225504356 0.040441176 0.04093248
24-36      49     0  49.0     25 0.42230816 0.0179552788 0.057077626 0.04443833
36-48      24     3  22.5      6 0.20684481 0.0045965514 0.025641026 0.03719256
48-60      15     1  14.5      1 0.15168620 0.0008717597 0.005952381 0.03340297
60-72      13     2  12.0      2 0.14122508 0.0019614594 0.015151515 0.03269641
72-84       9     2   8.0      0 0.11768757 0.0000000000 0.000000000 0.03119678
84-96       7     1   6.5      0 0.11768757 0.0000000000 0.000000000 0.03119678
96-108      6     4   4.0      1 0.11768757 0.0024518243 0.023809524 0.03119678
108-120     1     1   0.5      0 0.08826567           NA          NA 0.03459310
          se.pdf   se.hazard
0-12    0.003411040 0.004760771
```

```
12-24   0.003341601 0.006829515
24-36   0.003144205 0.010725232
36-48   0.001807056 0.010343283
48-60   0.000862790 0.005948584
60-72   0.001345092 0.010669376
72-84          NaN         NaN
84-96          NaN         NaN
96-108  0.002220584 0.023565317
108-120         NA          NA
```




```
> survdiff(Surv(y, censor == 0) ~ x10, data = dat)
Call:
survdiff(formula = Surv(y, censor == 0) ~ x10, data = dat)
      N Observed Expected (O-E)^2/E (O-E)^2/V
x10=0 684      293    415.3      36.0     164.2
x10=1 191      138     77.1      48.0      57.6
x10=2 128      107     45.6      82.6      93.1
 Chisq= 174  on 2 degrees of freedom, p= 0
```

例 12.5（程序 12.4）

：Cox 回归，逐步回归

```
library(gdata)
dat <- read.xls("life.xls", sheet=1)
# x3 和 x8 为无序变量，需由 3 分类转换为两个 2 分类变量
dat$x3b <- ifelse(dat$x3 == 1, 1, 0)
dat$x3c <- ifelse(dat$x3 == 2, 1, 0)
dat$x8b <- ifelse(dat$x8 == 1, 1, 0)
dat$x8c <- ifelse(dat$x8 == 2, 1, 0)

library(survival)
fit <- coxph(Surv(y, censor==0)~1, data=dat, ties="breslow")
  # ~1 表示拟合无因子的模型（只有截距），用于 forward 逐步回归时向其中
      添加因子；也可写 lower= ~1-1，-1 表示无截距
  # R 默认使用 ties="efron"。因 SAS 使用 Breslow，故此处亦如此
summary(step(object=fit,
                 scope=list(upper=~(x1+x2+x3b+x3c+x4+x5+x6+x7+x8b+x8c+x9
```

```
                    +x10), lower=~1), direction="forward"))
  # SAS 默认使用 direction="forward"，故此处也以此方法拟合模型
  # upper=~(x1+x2+x3b+x3c+x4+x5+x6+x7+x8b+x8c+x9+x10)指明选入因子的
      上限
  # lower=~1 指明选入因子的下限，即为空模型，可不写
  # R 使用 AIC 筛选模型，和 SAS 默认的根据 p 值筛选不同，目前认为，AIC 的
      方法更为稳健

# 根据上述步骤，选取 p<0.05 的因子重新拟合模型
fit <- coxph(Surv(y, censor==0)~x3b+x3c+x4+x5+x8c+x9, data=dat,
              ties="breslow")
summary(fit)
```

结果：

```
> summary(step(object=fit, scope=list(upper=~(x1+x2+x3b+x3c+x4+x5+x6+x7+x8b+x8c+x9+x10),
lower=~1), direction="forward"))
Call:
coxph(formula = Surv(y, censor == 0) ~ x5 + x9 + x4 + x3c + x8c +
   x7 + x3b + x2, data = dat, ties = "breslow")
  n= 1003, number of events= 538
       coef exp(coef) se(coef)      z Pr(>|z|)
x5   0.15205   1.16422  0.04403  3.453 0.000554 ***
x9   0.26835   1.30781  0.04656  5.764 8.21e-09 ***
x4   0.32162   1.37936  0.10769  2.986 0.002822 **
x3c -1.02894   0.35738  0.37204 -2.766 0.005680 **
x8c  0.74977   2.11650  0.33994  2.206 0.027413 *
x7   0.14890   1.16055  0.07632  1.951 0.051051 .
x3b -0.71855   0.48746  0.36027 -1.994 0.046102 *
x2   0.13342   1.14273  0.08919  1.496 0.134646
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
   exp(coef) exp(-coef) lower .95 upper .95
x5     1.1642     0.8589    1.0680    1.2692
x9     1.3078     0.7646    1.1938    1.4328
x4     1.3794     0.7250    1.1169    1.7035
x3c    0.3574     2.7981    0.1724    0.7410
x8c    2.1165     0.4725    1.0871    4.1207
x7     1.1606     0.8617    0.9993    1.3478
x3b    0.4875     2.0515    0.2406    0.9876
x2     1.1427     0.8751    0.9595    1.3610
> fit = coxph(Surv(y, censor==0)~x3b+x3c+x4+x5+x8c+x9, data=dat, ties="breslow")
> summary(fit)
Call:
coxph(formula = Surv(y, censor == 0) ~ x3b + x3c + x4 + x5 +
   x8c + x9, data = dat, ties = "breslow")
  n= 1003, number of events= 538
       coef exp(coef) se(coef)      z Pr(>|z|)
x3b -0.71693   0.48825  0.36072 -1.987 0.046869 *
x3c -1.00766   0.36507  0.37234 -2.706 0.006805 **
x4   0.35849   1.43117  0.10569  3.392 0.000694 ***
x5   0.16029   1.17385  0.04396  3.646 0.000266 ***
x8c  0.70192   2.01763  0.33922  2.069 0.038522 *
x9   0.27026   1.31030  0.04661  5.798 6.72e-09 ***
---
```

例 13.2

：临床诊断试验，敏感度，特异度，似然比

```
# install.packages("caret")
dat <- matrix(c(20, 8, 5, 85), nrow=2,
        dimnames=list(pred=c("+", "-"), truth=c("+", "-")))
dat <- as.table(dat)
library(caret)
confusionMatrix(dat)
  # 此包在 CRAN 中有，但不输出阳性似然比、阴性似然比等内容

# 从 http://cran.r-project.org/src/contrib/Archive/DiagnosisMed/下载
   DiagnosisMed 源码包
# 在终端通过以下命令修改包（R3.0 要求源码包包含 NAMESPACE 的文件）
# tar -xvf DiagnosisMed_0.2.3.tar.gz
# cd DiagnosisMed
# echo 'exportPattern("." )' > NAMESPACE
# cd ..
# tar -zcf DiagnosisMed_0.2.3.tar.gz DiagnosisMed
# 安装 DiagnosisMed 包
# install.packages("~/Desktop/DiagnosisMed_0.2.3.tar.gz", repos = NULL, type =
   "source")
library(DiagnosisMed)
diagnosis(dat)  # 输出结果包含似然比、Youden 系数等
```

结果：

```
> confusionMatrix(dat)
Confusion Matrix and Statistics
   truth
pred  +  -
   + 20  5
   -  8 85
               Accuracy : 0.8898
                 95% CI : (0.819, 0.94)
    No Information Rate : 0.7627
    P-Value [Acc > NIR] : 0.0003744
                  Kappa : 0.684
 Mcnemar's Test P-Value : 0.5790997
            Sensitivity : 0.7143
            Specificity : 0.9444
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.9140
             Prevalence : 0.2373
         Detection Rate : 0.1695
   Detection Prevalence : 0.2119
       'Positive' Class : +
> diagnosis(dat)
Reference standard: truth
Index test        : pred
---------------------------------------------------------------
    truth
pred   +   - Sum
  +   20   5  25
```

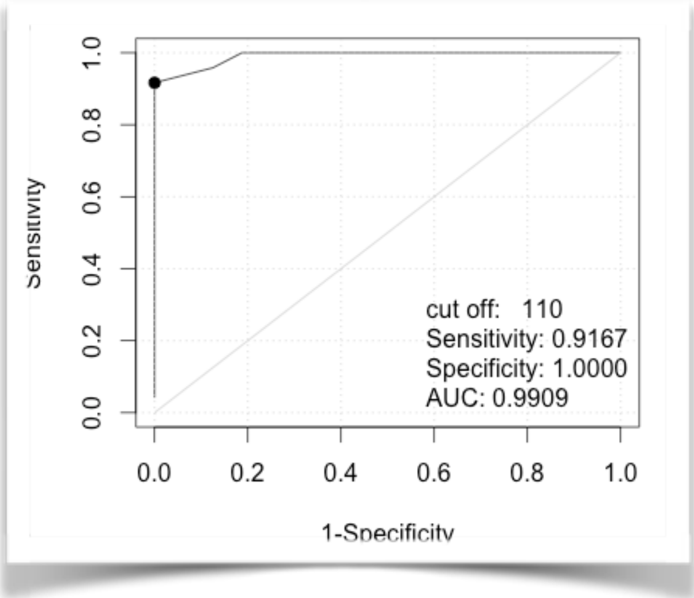
```
 -    8  85  93
 Sum  28  90 118
The test has the following parameters [95% confidence interval]
--------------------------------------------------------------
Sample size:                    118
Prevalence considered(%):       76.27
Sensitivity(%):                 94.44  [ 87.65  -  97.60 ]
Specificity(%):                 71.43  [ 52.94  -  84.75 ]
Positive predictive value(%):  91.40  [ 83.93  -  95.58 ]
Negative predictive value:(%): 80.00  [ 60.87  -  91.14 ]
Positive likelihood ratio:     3.31   [ 1.84  -  5.95 ]
Negative likelihood ratio:     0.08   [ 0.03  -  0.19 ]
Diagnostic odds ratio:         39.10  [ 12.37  -  148.75 ]
Error trade off (FN : FP)      0.62 : 1
Error rate(%):                  11.02  [ 6.55  -  17.94 ]
Accuracy(%):                    88.98  [ 82.06  -  93.45 ]
Youden index:                   0.6587  [ 0.6643  -  0.6532 ]
Area under ROC curve:           0.8294
--------------------------------------------------------------
```

例 13.3

：单条 ROC 曲线

```
library(pROC)
gold <- rep(c(0, 1), c(16, 24))
  # 0 和 1 表示金标准的诊断结果
test <- c(77, 83, 86, 93, 94, 94, 97, 97, 98, 99, 99,
      101, 102, 104, 108, 108, 104, 108, 110, 110,
      112, 119, 120, 120, 121, 121, 123, 124, 126,
      128, 129, 130, 130, 131, 131, 134, 137, 138,
      139, 140)
test2 <- rep(c(1, 1), c(16, 24))

# 画出 ROC 曲线
library(Epi)
a <- ROC(gold, test)
  # ROC()函数位于 Epi 包中，可以处理多变量联合检测的问题
roc(gold, test, ci=T)
  # 输出 AUC 及其可信区间
roc.test(roc(gold, test), roc(gold, test2))
  # roc(gold, test2)用于生成 AUC=0.5 的模型，用于与实际模型比较
```

结果：




```
> roc(gold, test, ci=T)
Call:
roc.default(response = gold, predictor = test, ci = T)
Data: test in 16 controls (gold 0) < 24 cases (gold 1).
Area under the curve: 0.9909
95% CI: 0.9737-1 (DeLong)
> roc.test(roc(gold, test), roc(gold, test2))
    DeLong's test for two correlated ROC curves
data:  roc(gold, test) and roc(gold, test2)
Z = 56.1186, p-value < 2.2e-16
alternative hypothesis: true difference in AUC is not equal to 0
sample estimates:
AUC of roc1 AUC of roc2
 0.9908854   0.5000000
```

例 13.4

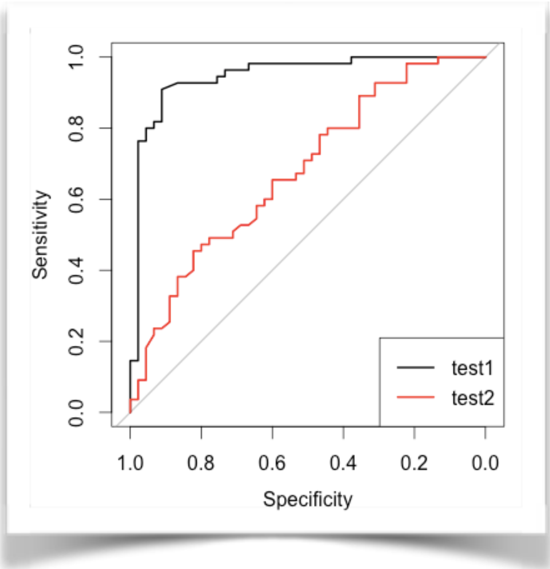
：两条 ROC 曲线比较

```
library(pROC)
dat <- read.csv(text="id, diag, test1, test2
          1, 1, 112.700, 124.000
          2, 1, 104.000, 135.800
          3, 1, 126.700, 122.700
          4, 1, 123.300, 158.400
          5, 1, 120.500, 141.200
          6, 1, 130.300, 131.100
          7, 1, 129.600, 148.000
          8, 0, 97.900, 130.600
          9, 0, 94.900, 120.000
          10, 1, 140.200, 140.900
          11, 1, 119.700, 142.100
          12, 0, 98.600, 133.000
          13, 0, 77.300, 121.700
          14, 1, 139.900, 128.800
          15, 0, 97.900, 116.600
```

```
16, 1, 134.200, 130.900
17, 1, 137.500, 150.500
18, 1, 131.200, 131.000
19, 1, 110.000, 140.200
20, 0, 99.700, 117.500
21, 1, 121.000, 135.500
22, 1, 131.100, 131.500
23, 1, 108.900, 147.500
24, 1, 121.200, 138.000
25, 0, 83.000, 132.100
26, 1, 124.300, 135.400
27, 0, 102.500, 133.900
28, 0, 104.500, 147.000
29, 1, 128.700, 133.800
30, 1, 130.800, 119.300
31, 0, 108.900, 108.400
32, 0, 93.200, 115.800
33, 0, 101.300, 114.700
34, 1, 138.800, 137.100
35, 1, 110.400, 141.800
36, 0, 99.800, 119.700
37, 0, 108.300, 108.700
38, 0, 86.000, 137.900
39, 1, 120.600, 125.500
40, 0, 94.900, 126.600
41, 1, 102.700, 142.800
42, 1, 126.600, 147.500
43, 0, 103.200, 122.400
44, 1, 123.000, 151.000
45, 1, 119.900, 149.800
46, 1, 95.000, 131.300
47, 1, 143.600, 136.200
48, 0, 84.000, 128.300
49, 0, 84.200, 138.800
50, 0, 112.900, 126.800
51, 1, 110.500, 129.100
52, 1, 126.800, 143.400
53, 1, 115.600, 155.400
54, 1, 110.500, 157.400
55, 1, 127.000, 159.400
56, 1, 131.600, 175.700
57, 1, 128.200, 157.200
58, 0, 106.900, 141.700
59, 0, 107.900, 141.000
60, 1, 118.400, 153.600
61, 1, 128.000, 153.900
62, 1, 126.800, 154.600
63, 1, 104.900, 164.000
64, 0, 100.300, 129.300
65, 0, 133.400, 136.000
66, 0, 90.600, 144.800
67, 0, 102.900, 136.600
68, 1, 134.800, 165.800
69, 0, 86.400, 144.000
70, 1, 132.800, 166.600
71, 0, 107.700, 167.500
72, 1, 128.900, 144.900
73, 1, 123.100, 152.400
74, 1, 135.700, 139.100
75, 1, 124.500, 160.600
76, 0, 98.800, 142.200
77, 0, 100.200, 144.400
78, 0, 105.400, 155.400
```

```
          79, 0, 95.100, 155.900
          80, 1, 110.700, 160.900
          81, 0, 85.600, 149.900
          82, 0, 102.500, 132.100
          83, 0, 108.900, 133.500
          84, 0, 112.200, 152.800
          85, 0, 102.800, 139.000
          86, 1, 119.200, 144.600
          87, 1, 131.100, 154.500
          88, 0, 92.400, 127.700
          89, 1, 133.100, 157.400
          90, 1, 114.600, 171.200
          91, 0, 94.000, 162.500
          92, 1, 131.800, 141.900
          93, 0, 94.100, 142.100
          94, 0, 77.400, 138.100
          95, 0, 96.800, 157.400
          96, 0, 114.800, 142.800
          97, 0, 86.200, 144.500
          98, 1, 113.100, 136.900
          99, 0, 88.900, 149.800
          100, 1, 132.500, 158.900")
# 参考了帖子 http://www.dxy.cn/bbs/topic/26751203
roc1 <- plot.roc(dat$diag, dat$test1, col="1")
roc2 <- lines.roc(dat$diag, dat$test2, col="2")
legend("bottomright", legend=c("test1", "test2"), col=c(1,2), lwd=2)
  # 添加 ROC 曲线
roc.test(roc1, roc2)
  # 比较 AUC
```

结果：



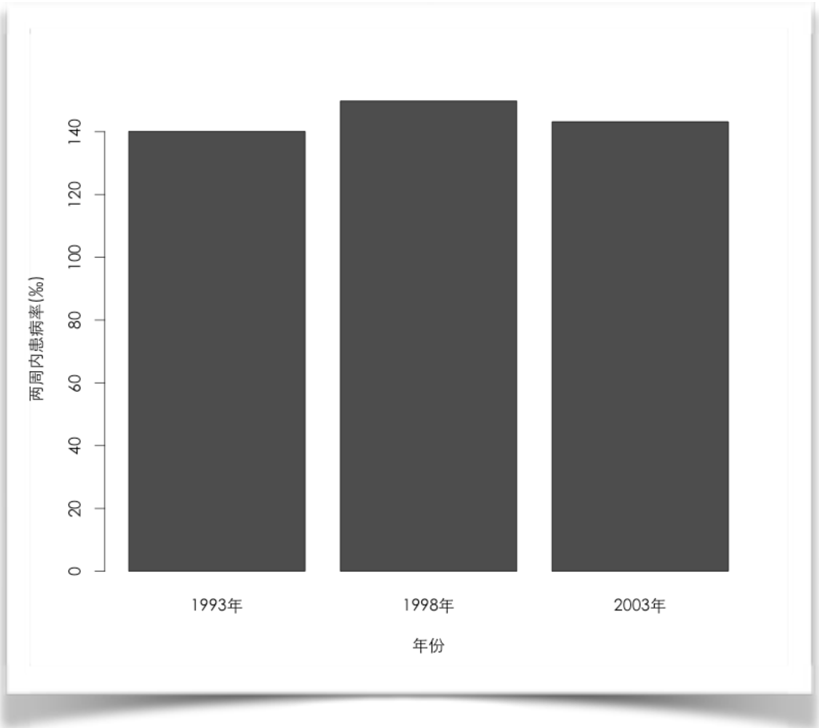

```
> roc.test(roc1, roc2)
    DeLong's test for two correlated ROC curves
data:  roc1 and roc2
Z = 4.6415, p-value = 3.458e-06
alternative hypothesis: true difference in AUC is not equal to 0
sample estimates:
AUC of roc1 AUC of roc2
  0.9466667   0.6787879
```

例 15.3（程序 15_1）

：直条图

```
dat <- matrix(c(140.1, 149.8, 143.2), ncol=3)
  # 生成一个单行的矩阵
dimnames(dat) <- list(x=c(), year=c("1993 年", "1998 年", "2003 年"))

par(family="STXihei")
  # R 默认使用的字体不支持中文显示，此处改为其他字体
barplot(dat, xlab="年份", ylab="两周内患病率(‰)")
  # barplot 函数以矩阵的各列作图
```

结果：



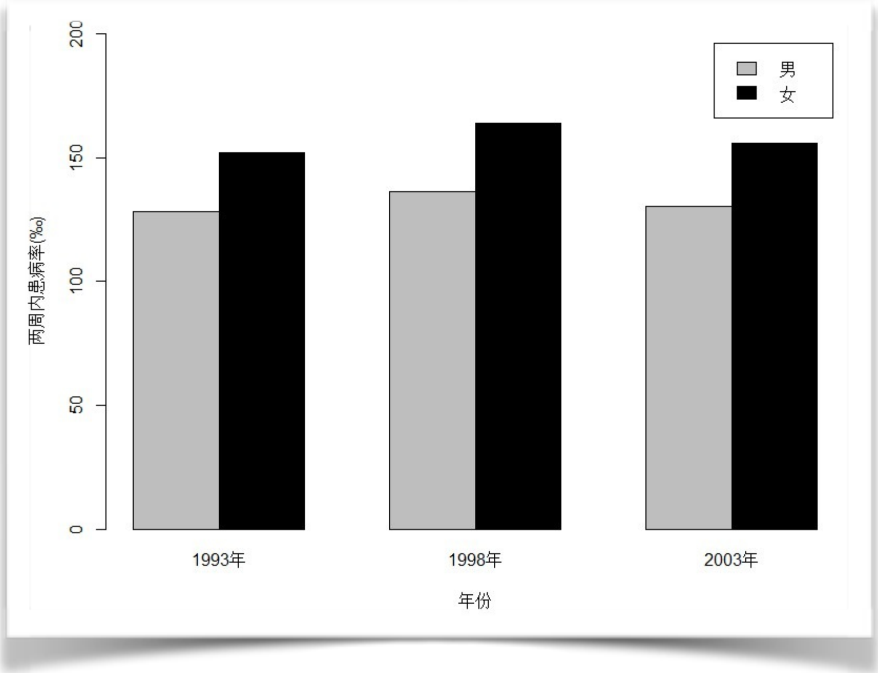
例 15.4（程序 15_2）

：复式直条图

```
dat <- matrix(c(128.4, 151.9, 136.2, 164.1, 130.4, 155.8), ncol=3)
dimnames(dat) <- list(sex=c("男", "女"), year=c("1993 年", "1998 年", "2003 年
"))
```

```
# par(family="STXihei")
barplot(dat, xlab="年份", ylab="两周内患病率(‰)", beside=T, ylim=c(0, 200),
        col=c("gray", "black"), legend=T)
```

结果:



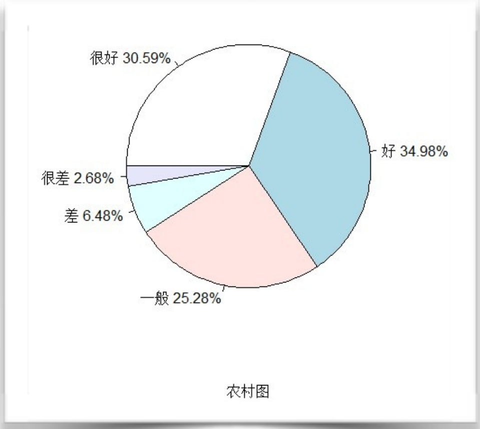
例 15.5（程序 15_3）

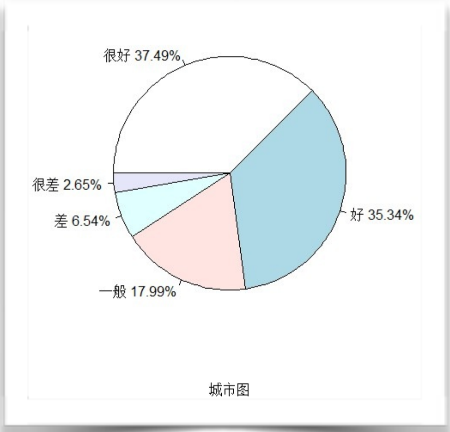
：饼图，百分比条图

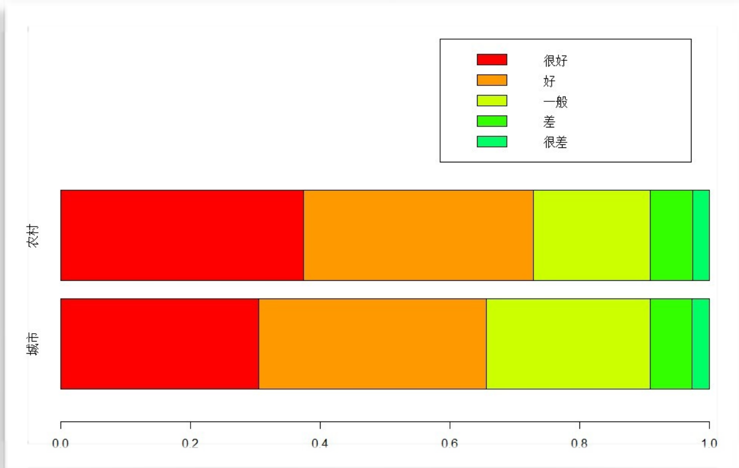
```
country <- c(30.59, 34.98, 25.28, 6.48, 2.68)
city <- c(37.49, 35.34, 17.99, 6.54, 2.65)
label <- c("很好", "好", "一般", "差", "很差")
percent1 <- round(country/sum(country)*100, digits=2)
  # 保留小数点后两位
label1 <- paste(label, percent1, sep=" ")
label1 <- paste(label1, "%", sep="")
  # 连接字符串, sep 表示分隔符
pie(country, init.angle=180, labels=label1, xlab="农村图", clockwise=T)

percent2 <- round(city/sum(city)*100, digits=2)
label2 <- paste(label, percent2, sep=" ")
label2 <- paste(label2, "%", sep="")
pie(city, init.angle=180, labels=label2, xlab="城市图", clockwise=T)

pct <- matrix(c(country/sum(country), city/sum(city)), ncol=2)
dimnames(pct) <- list(result=c("很好", "好", "一般", "差", "很差"), area=c("城市",
```

```
"农村"))
barplot(pct, horiz=T, legend=T, ylim=c(0, 4), col=rainbow(10))
```

结果：







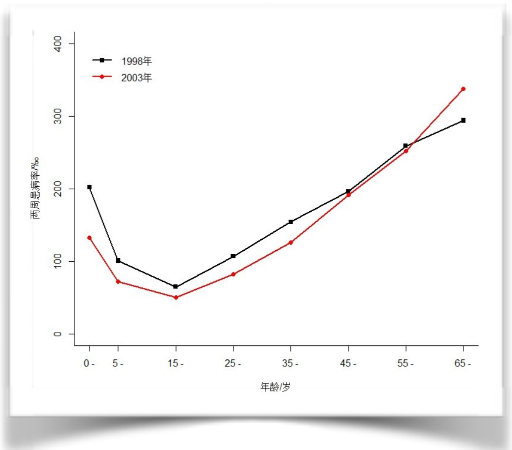
例 15.6（程序 15_4）

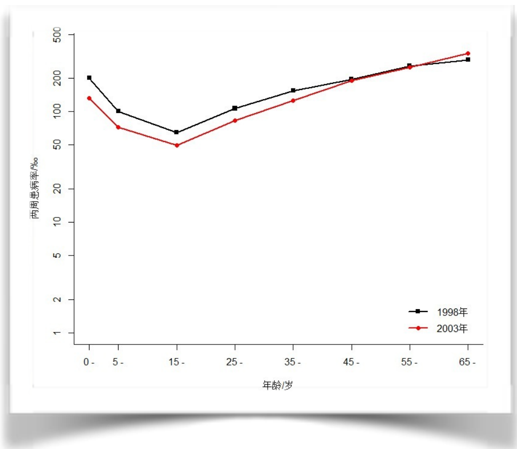
：线图

```
dat <- read.csv(text="age, x1, x2
              0, 201.8, 133.0
              5, 100.6, 72.2
              15, 64.7, 49.8
              25, 106.8, 82.5
              35, 154.3, 126.2
              45, 196.2, 191.5
              55, 259.1, 251.8
              65, 294.1, 338.3")
plot(dat$age, dat$x1, pch=15, col=1, ylim=c(0, 400), bty="l", xaxt="n", xlab="年
      龄/岁", ylab="两周患病率/‰")
```

```
  # bty="l"表示边框为 L 形
lines(dat$age, dat$x1, lwd=2)
axis(side= 1, at=c(0, seq(5, 65, 10)), labels=paste(dat$age, "-"))

points(dat$age, dat$x2, pch=16, col=2)
lines(dat$age, dat$x2, col=2, lwd=2)
legend(0, 400, legend=c("1998 年", "2003 年"), col=c(1,2), pch=c(15, 16), lwd=2,
        bty="n")

# 画半对数坐标
plot(dat$age, dat$x1, ylim=c(1, 400), log="y", pch=15, col=1, bty="l", xaxt="n",
        xlab="年龄/岁", ylab="两周患病率/‰")
  # log="y"表示对 y 轴进行对数转换
lines(dat$age, dat$x1, log="y", lwd=2)
axis(side= 1, at=c(0, seq(5, 65, 10)), labels=paste(dat$age, "-"))

points(dat$age, dat$x2, log="y", pch=16, col=2)
lines(dat$age, dat$x2, log="y", col=2, lwd=2)
legend("bottomright", legend=c("1998 年", "2003 年"), col=c(1,2), pch=c(15, 16),
        lwd=2, bty="n")
```

结果：



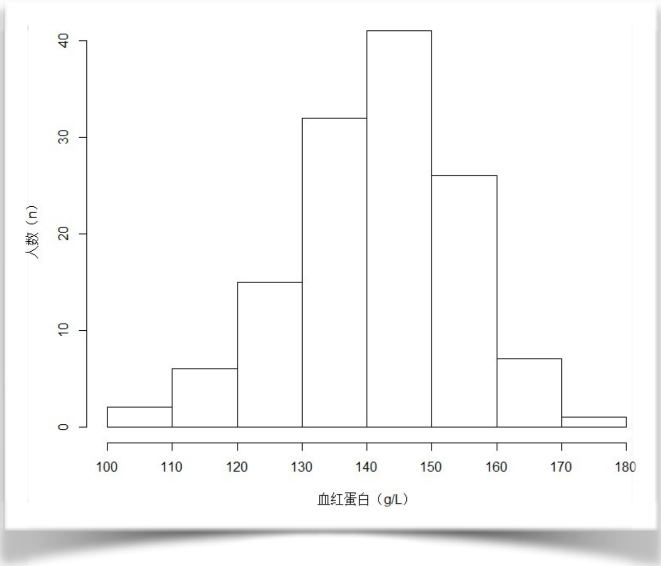


例 15.7（程序 15_5）

：直方图

```
dat <- rep(seq(105, 175, 10), c(2, 6, 15, 32, 41, 26, 7, 1))
hist(dat, xaxt="n", main="", xlab="血红蛋白（g/L）", ylab="人数（n）")
  # hist 默认以频数作图
axis(side=1, at=seq(100, 180, 10))
```

结果：

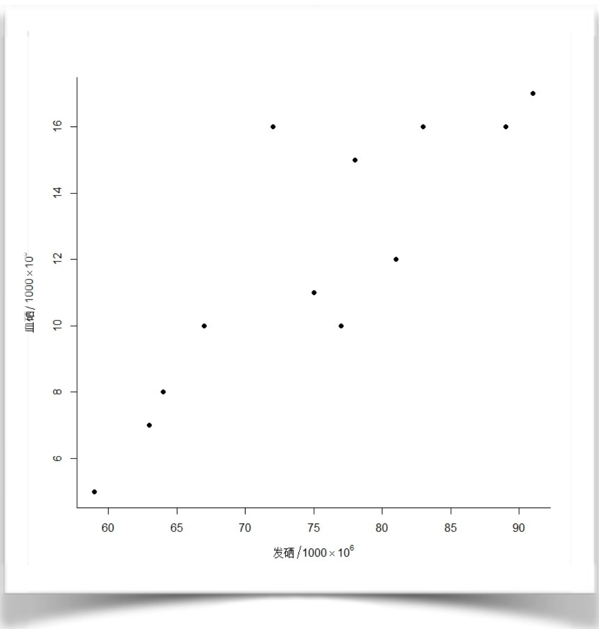


例 15.8（程序 15_6）

：直方图

```
cards <- c(78, 15, 83, 16, 67, 10, 64, 8, 89, 16, 72, 16, 63, 7, 59, 5, 75, 11, 81, 12,
           91, 17, 77, 10)
x <- cards[seq(1, length(cards), 2)]
y <- cards[seq(2, length(cards), 2)]
plot(x, y, pch=16, bty="l", xlab=expression(发硒/1000%*%10^6),
      ylab=expression(血硒/1000%*%10^6))
  # expression()函数用于 Tex 格式化，详细的信息可以用 demo(plotmath)查询
```

结果：



例 15.9（程序 15_7）

：直方图

```
x <- c(21, 19, 23, 23, 16, 17, 18, 22, 15, 24, 20, 26, 23, 25, 22, 26, 28, 34, 32, 24,
        21, 25, 19, 20, 22, 29, 23, 20, 17, 27)
group <- rep(c("常规药组", "新药组"), c(length(x)/2, length(x)/2))
boxplot(x~group, ylab=expression(血红蛋白增加量/g%.%L^-1))
```

结果：