R 入门

目录

1 R 简介

R 的早期版本是 S 和 S-plus 商用软件。

- S
  - 1980's 由 Bell Lab 的 John Chambers 开发。
  - 为什么叫作 "S"? Statistics
  - S 后来发展为 Splus, 成为商业软件 (Insightful Corporation)
- R
  - 1990's 由新西兰 Auckland 大学统计系 Ross Ihaka, Robert Gentleman 开发。
  - 与 S 基本相同, 但作为 GNU Project 的一部分, 开源免费.
    (GNU 是一个开源化运动，将操作系统 Unix 开源化产生了 Linux，S 开源化产生了 R 等等。
    GNU 的递归定义: GNU = GNU's Not Unix)
  - 为什么叫作 "R"? 作者名字的首字母。
- 下载 R 以及 package (CRAN: The Comprehensive R Archive Network):

  `http://cran.r-project.org/`
- 入门教程:

  Quick-R 网上简明教程: `http://www.statmethods.net/`

2 体会 R

下载并安装 R 之后，打开进入 R 环境。执行下面的简单例子，体会一下 R 的基本操作和功能。注意 R 工作环境中的提示符为 >。

```
> 1+2
[1] 3
> log(2)
[1] 0.6931472
> 1:5
[1] 1 2 3 4 5
> 1:30 # 显示行的 [1],[21] 是编号
[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
[21] 21 22 23 24 25 26 27 28 29 30
> 1:5+10
[1] 11 12 13 14 15
> a=1:5 # 赋值
> a
[1] 1 2 3 4 5
> a+10
[1] 11 12 13 14 15
> b=c(3,1,1,0,2) #c(...) 将若干数字组成向量
> b
[1] 3 1 1 0 2
> a+b
[1] 4 3
> sum(a) # 求和
[1] 15
> sum(b)
[1] 7
> mean(a) # 求均值
[1] 3
> var(b) # 方差
[1] 1.3
> b^2 # 各个分量的平方
[1] 9 1 1 0 4

> A=matrix(1:6,nrow=2,ncol=3) #2x3 矩阵
> x=c(-1,0,1)
> x
[1] -1  0  1
> A%*%x # 矩阵与向量乘法
[,1]
[1,]    4
[2,]    4
> t(A) # 转置
[,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> A%*%t(A)->B # 矩阵乘法
> solve(B) # 逆矩阵
[,1]       [,2]
[1,]  2.333333 -1.833333
```

```
[2,] -1.833333  1.458333
> det(B) # 行列式
[1] 24
> eigen(a) # 特征分解/谱分解
eigen() decomposition
$‘values‘
[1] 90.7354949  0.2645051
$vectors
[,1]       [,2]
[1,] 0.6196295 -0.7848945
[2,] 0.7848945  0.6196295
> a
[1] 1 2 3 4 5
> b
[1] 3 1 1 0 2
> a*b # 分量相乘
[1]  3  2  3  0 10
> sum(a*b) # 内积 a'b
[1] 18
> t(a)%*%b # 内积 a'b
[,1]
[1,]   18
> a%*%t(b) # 外积 ab'
[,1] [,2] [,3] [,4] [,5]
[1,]    3    1    1    0    2
[2,]    6    2    2    0    4
[3,]    9    3    3    0    6
[4,]   12    4    4    0    8
[5,]   15    5    5    0   10
> (x=rnorm(5)) # 产生 5 个标准正态随机数
[1]  0.465987000  0.475899021 -0.924722447  2.753912505 -1.007803783
> plot(x) # 散点图
> hist(x) # 直方图
```

3 R object: 向量、矩阵、函数

R 中的变量、数据、程序等都是 R object，主要指向量、矩阵、函数。

3.1 Object 的命名、类型、定义和基本运算

命名

Object 名称由字母、数字或 "." (不能出现在开头) 构成。
- 区分大小写。
- 用户应避免定义与系统自带的 object 重名，避免如下 R 内部的固定名称：

无穷大: Inf
空值：NULL
缺失值: NA NaN
逻辑值：F T False TRUE
函数： c q t mean var rank sort order diff..
常数：pi letters LETTERS...
循环、判断: break else for function if in next repeat while...

数据对象的类型

- 数据对象 (object) 包括：向量 (vector), 矩阵 (matrix), 数组 (array), 列表 (list), 数据集 (data frame), ...

  判断类型 (is.something)：is.vector(x), is.matrix(x), ...。

  改变类型 (as.something): as.vector(x), as.matrix(x), ..

- 数据根据取值分为如下类型：数值 (numeric), 字符 (character), 逻辑 (logical), 缺失 (na), 因子 (factor), ...

  判断类型 (is.something)：is.numeric(x), is.character(x), ...。

  改变类型 (as.something)：as.numeric(x), as.character(x)

向量、矩阵、数组的元素取值通常都应是同一类型的。列表（list）是一种向量，数据集（data frame）类似于矩阵，但它们可以含不同取值类型的分量。首先看几个简单例子，后面会有细节。

例：
```
> 1:4 # 产生整数向量 (1,2,3,4) <---- "#" 代表注释
[1] 1 2 3 4
> c(2.5, 1.8, 4.3, 1.1) # 产生实数向量 (2.5,1.8, 4.3, 1.1)
[1] 2.5 1.8 4.3 1.1
> c("Linear", "Regression", "Anaysis") # 产生字符型向量
[1] "Linear" "Regression" "Anaysis"
> is.vector(1:4)
TRUE
> is.matrix(1:4)
FALSE
```

```
> is.character(1:3) # 判断类型
[1] FALSE
> as.character(1:3) # 改变类型
[1] "1" "2" "3"
> c(-1,1,3) > 2 # 向量 (-1,1,3) 的各分量是否大于 2?
[1] FALSE FALSE  TRUE
> x=1:4  # 把向量 (1,2,3,4) 赋值给 x
> is.vector(x)
TRUE
```

数据对象的定义/赋值和基本操作

- 赋值: myobject=3; myobject ←3; 3→ myobject

  上述三种方式等价，都是把数字 3 赋给 myobject.

- 运算结果：myobject = f(x,y,...) # f 是函数/程序, 其运算结果赋给 myobject.

- 外部读入（# 首先需确认 datafile.txt 在当前工作目录下或指定路径）:

  myobject = scan("datafile.txt")

  myobject = read.table("datafile.xls")

  myobject = read.table("http://..../datafile.xls")

3.2 向量

（**1**）定义向量 **(**数值**,** 字符**,** 逻辑**,** 列表**):**

```
> mydata <- 1:5
> m = mean(mydata) # 求 mydata 即 1,2,3,4,5 的平均值，并赋值给 m
> mydata=read.table("file.xls") # 将外部工作目录下的文档 file.xls 读入到 R

# 数值向量:
> x = 1:3
> 1:3 -> x
> x
[1] 1 2 3
> x <- c(9, 0, -5, 2, -1, 3) ## c:  concatenate 粘结
> x
[1]  9  0 -5  2 -1  3

# 字符向量
> course=c("multivariate", "analysis")
> course
[1] "multivariate" "analysis"

# 逻辑向量
> x=c(F,F,T)
> x
[1] FALSE FALSE  TRUE

> x = c(-1,1,3) > 2 # 判断每个分类是否大于 2
> x
```

```
[1] FALSE FALSE  TRUE

# 因子向量（因子变量的值称为水平，代表分类或分组，不具有实数含义，对因子变量不能做代数运算）
> x=factor(c(2,3,2,1,1,7))
> x
[1] 2 3 2 1 1 7
Levels: 1 2 3 7

> y=c("red","red", "yellow","blue")
> y
[1] "red"    "red"    "yellow" "blue"
> y=factor(y)
> y
[1] red    red    yellow blue
Levels: blue red yellow

# 缺失
> x=c(1:3, NA, 4:6)
> x
[1]  1  2  3 NA  4  5  6
> is.na(x) # 判断 x 的哪些分量是 NA
[1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE

# 列表向量 (可把不同类型的数据集中在一起，是一种特殊向量)
> x=list(1:2, c("a","d"))
> x
[[1]]  # [[1]] 是 x 的第一个分量
[1] 1 2

[[2]]  # [[2]] 是 x 的第一个分量
[1] "a" "d"

> x[[1]]
[1] 1 2

> unlist (x) # 解除 list 结构，成为一般向量，但由于有两种取值类型，统一为字符型
"1" "2" "a" "b"
```

注：list, data.frame 是两种把不同类型数据放置一起的方法，前者是“向量”，后者是“矩阵”

（**2**）判断或改变向量类型

```
> x=1:3
> is.numeric(x)
[1] TRUE
> is.logical (x)
[1] FALSE
> as.character(x)
[1] "1" "2" "3"
> y=as.factor(x) # 或 y=factor(x)
> x
[1] 1 2 3
> y
[1] 1 2 3
Levels: 1 2 3
```

```
> is.na( c(1:3, NA, 4:6)) # 判断哪个分量缺失
[1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

（**3**）命名各个分量**:** **names(x) =** 字符向量

```
> x=1:3
> names(x)
NULL
> names(x)=c("a", "b", "c")
> x
a b c
1 2 3
> mylist = list(1:2, c("a","d"))
> names(mylist)=c("x1", "x2")
> mylist
$x1
[1] 1 2
$x2
[1] "a" "d"
> names(mylist)=NULL # 去除名称
> mylist
[[1]]
[1] 1 2
[[2]]
[1] "a" "d"
```

（**4**）子集、下标操作**:** **x[sub]** 或 **x["变量名"]** 或 **x[logical]**

```
> x=-(3:4)
> x
[1] -3 -4

> x=-3:4
> x
[1] -3 -2 -1  0  1  2  3  4
> x[3]
[1] -1
> x[c(3,6)]
[1] -1  2
> x[c(6,3)]
[1] 2 -1
> x[-(1:6)] # "-" 号代表 "不取"
[1] 3  4

> letters
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r"。。。
> letters[1:4]
[1] "a" "b" "c" "d"

> names(x)=letters[1:5]
> x
a  b  c  d  e
```

```
0.59601640  0.27809609 -1.10968357  0.02912560 -1.34511581

> x[c("a", "d")] # 使用名称 (name) 取子集在数据集较大时更方便
a         d
0.59601640 0.02912560


# 逻辑下标: 取出 T 对应的分量

逻辑运算:   交: &        并:  |       否: !

>  x=-2:2
>
>  x>1
[1] FALSE FALSE FALSE FALSE  TRUE
>
>  x[x>1] # 取出那些大于 1 的分量
[1] 2
>
>  x>1 | x<= -2
[1]  TRUE FALSE FALSE FALSE  TRUE
>  x[x>1 | x< -2]
[1] 2
>
> x>=0 &  1:5 < 4
[1] FALSE FALSE  TRUE FALSE FALSE
> x[x>=0 &  1:5 < 4]
[1] 0
>
> x[ c(T, T, F, F, F) ]
[1] -2 -1
>
> x==2
[1] FALSE FALSE FALSE FALSE  TRUE

> x!=2
[1]  TRUE  TRUE  TRUE  TRUE FALSE

> x[x!=2]
[1] -2 -1 0 1

# 对于 list，使用 x[[ sub ]]
> mylist
[[1]]
[1] 1 2

[[2]]
[1] "a" "d"


> mylist[[2]]
[1] "a" "d"
```

3.3 矩阵

相关函数: matrix, dim, dimnames, is.matrix, as.matrix

```
> x=c(9,-5,-1,0,2,3)
> x1 = matrix(x, nrow=2, ncol=3) # 生成 2x3 矩阵，元素为 x, 首先是第一类，其次第二列..
> x1
[,1] [,2] [,3]
[1,]    9   -1    2
[2,]   -5    0    3

> class(x1) #x1 的类别/属性
[1] "matrix"
> dim(x1)
[1] 2 3

> x2 = matrix(x, nrow=2, ncol=3, byrow=T)
> x2
[,1] [,2] [,3]
[1,]    9   -5   -1
[2,]    0    2    3

> dimnames(x2)=list(c("Group1", "Group2"), c("x1", "x2", "x3")) # 行, 列命名
> x2
x1 x2 x3
Group1  9 -5 -1
Group2  0  2  3

> dimnames(x2)
[[1]]
[1] "Group1" "Group2"

[[2]]
[1] "x1" "x2" "x3"


# 下标 : mat[sub1,sub2]

> x1[1:2, c(1,3)]  # 1,2 行,1,3 列
> x1[2:1, c(1,3)]  # 2,1 行,1,3 列
> x1[, c(3,1)]     # 所有汉, 3,1 列
> x1[2,]           # 第 2 行, 所有列
> x1[c(T,F,T), ]   # 第 1,3 行
```

3.4 数组（**array**）

(较少使用，可略过)

```
> a = array(1:12, c(2,3,2))
> a
, , 1

[,1] [,2] [,3]
[1,]    1    3    5
```

```
[2,]    2    4    6

, , 2

[,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

# 命名：
> dimnames(a) =list(c("Group1", "Group2"), c("x1","x2","x3"), c("Region1", "Region2"))
> a
```

3.5 数据集（data frame）

data frame 具有“矩阵”形式，但不是矩阵，不同的列可以是不同的数据类型, 而矩阵的所有元素都是实数。一般的数据格式都是 data.frame。

```
# 生成数据集：使用 data.frame, 或从外部读入 (read.table,read.csv...).

> x=1:5
> y=letters[1:5]
> z=factor(c(1,2,2,1,2)
> mydataframe = data.frame(x=x, y=y, z=z)
> mydataframe
x y z
1 1 a 1
2 2 b 2
3 3 c 2
4 4 d 1
5 5 e 2

## 从外部文档读入数据
> read.table("http://staff.ustc.edu.cn/~ynyang/vector/databook/pottery.txt", header=T,sep="\t")
```

3.6 序列（sequence）

特殊的向量或数组，函数 seq, rep 等可以方便地产生数值、字符或因子序列。

```
> 1:4
> 4:1
> (-4):1
> -4:1
> -(4:1)
# seq(start, end, by=, lenght= ) 等差序列
> seq(0,1, by=0.1) #by: 间隔
> seq(1,3, length=10)
# rep 重复
> rep(1, 4) # 产生向量 1,1,1,1
> rep(1:2, 3) # 1,2,1,2,1,2
> rep(1:2, c(3,2))# 1,1,1, 2,2
> rep(letters,2)
```

4 数据的输入和输出

4.1 直接输入数据

```
> x=c(2.9, 3.1, 3.4) # 向量
> x=scan() # 手工输入向量
1: 2.9 3.1 3.4
4:
Read 3 items
> x
[1] 2.9 3.1 3.4

> x =matrix(1:4,2,2) # 矩阵
> x
[,1] [,2]
[1,]    1    3
[2,]    2    4

> cbind(1:2, 3:4)  # 按列合并成矩阵
> rbind(c(1,3), c(2,4)) # 按行合并成矩阵

> data.frame(x1=1:3, x2=LETTERS[1:3], x3=factor(6:8)) # 将几个向量/矩阵案列合并成 data frame
x1 x2 x3
1  1  A  6
2  2  B  7
3  3  C  8
```

4.2 修改, 编辑

```
> fix(x)

> x[2] = -3.1

> x =matrix(1:4,2,2)
> x
[,1] [,2]
[1,]    1    3
[2,]    2    4

> as.vector(x) # 按列拉直
[1] 1 2 3 4

> as.matrix(1:4) # 转成 4x1 矩阵

> data.frame(x) # 将矩阵转化成 data frame
X1 X2
1  1  3
2  2  4
```

4.3 外部读入数据

向量: scan,

矩阵：data.frame: read.table, read.csv, read.csv2, read.delim, read.delim2

```
x=scan("mydata.txt") ## scan 读入的数据是向量。如果 mydata.txt 不在你的工作目录，需指定路径，
# 比如， scan("c:\mywork\mydata.txt"),
# scan("http://staff.ustc.edu.cn/~ynyang/matrix/databook/pottery.txt")

read.table(mydata.xls, header=FALSE, sep = "\t", ...) # 同样指定 mydata.xls 的路径

# 其中 header=T: 指定数据文件第一行为列变量名， sep="\t": tab 键分隔各数字单元;

> read.csv("mydata.xls", head=T)
> read.csv2("mydata.csv", sep="\t")
```

4.4 输出数据: write, write.table, write.csv, write.csv2

```
write(x, file = "data",
ncolumns = if(is.character(x)) 1 else 5,
append = FALSE, sep = " ")

write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
eol = "\n", na = "NA", dec = ".", row.names = TRUE,
col.names = TRUE, qmethod = c("escape", "double"))
```

4.5 读入, 输出一般 R 对象

将 R object 以 binary, ascii 形式储存成外部文件, 或读入外部储存的 R 源码

dput: dput(x, "mydata.txt")

dget: dget("mydata.txt") -> x

save: save(x1, x2, file="mydump.rda") # binary

load: load("mydump.rda")

dump: dump(ls(), "mydump.txt")

sink

5 基本运算及函数

5.1 基本运算

---

1. 运算

```
> 1+2
[1] 3

> x=1:3
> y=c(1,-1,0)
> x
[1] 1 2 3
> y
[1]  1 -1  0

> x+y # 逐分量相加
[1] 2 1 3
> x*y
[1]  1 -2  0
> x/y
[1]   1  -2 Inf

> x^2  # 每个分量平方
[1] 1 4 9

> 2^x
```

2. 循环规则（两个长度不同的向量运算时，将短的向量循环扩展至长度相同)

```
> 1:3 + 1 # 1 --> c(1,1,1)
[1] 2 3 4

> a=1:4
> b=1:2
> a+b ## 把 b 扩为 c(1,2,1,2), 然后相加
[1] 2 4 4 6

> a=1:5; b=1:2
> a*b # 把 b 扩为 c(1,2,1,2,1), 然后相乘
[1] 1 4 3 8 5
```

3. 标准运算符

```
/   除法
*   乘法
+   加法
-   减法
%/% 整除
%%  余数
%*% 矩阵相乘 A%*%B
^   幂次 a^b
```

4. 逻辑运算

```
&   #AND  #x&y: 逐分量运算，结果为向量
|   #OR
!   #NOT

any(x) # true if any of x is true
all(x) # true if all are true
isTRUE(x)   # test if x is true
is.na(x)  # 判断是否缺失 (NA)
>,
<,
>=,
<=,
==, (判断相等)
!=   （判断不等）


> (x=c(-5,0, 5:2) )  # 赋值命令放在 () 中，既执行赋值，也把复制后的结果列出
[1] -5  0  5  4  3  2
> x>3
[1] FALSE FALSE  TRUE  TRUE FALSE FALSE
> x>=3
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE
> (i=x>=3)
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE
> i
[1] FALSE FALSE  TRUE  TRUE  TRUE FALSE
> x[i]
[1] 5 4 3
> x[!i]
[1] -5  0  2
>  x>2 & x<4
[1] FALSE FALSE FALSE FALSE  TRUE FALSE
> x<0 |x>3
[1]  TRUE FALSE  TRUE  TRUE FALSE FALSE
> any(x>6)
[1] FALSE
> any(x>4)
[1] TRUE
> all(x>=0)
[1] FALSE
> x==0 # 判断 x 的各分量是否为 0
[1] FALSE  TRUE FALSE FALSE FALSE FALSE
> x!=0 # 判断非 0
[1]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
> log(x)
[1]       NaN      -Inf 1.6094379 1.3862944 1.0986123 0.6931472
Warning message:
In log(x) : 产生了 NaNs
> is.na(log(x))
[1]  TRUE FALSE FALSE FALSE FALSE FALSE
> x[!is.na(log(x))]
[1] 0 5 4 3 2
```

5.2 运算有关的函数（使用方式: function.name(x)）

```
abs     # 绝对值 abs(x) 给出所有 x 元素的绝对值
sign    # 符号
log     # 对数, log(x) 自然对数; log(x, base=2) 以 2 为底
sqrt    # 开方
exp     # 指数
sin; asin; cos; acos; tan; atan; cosh; acosh;tanh; atanh
gamma   # Gamma 函数
lgamma  # log-gamma 函数

round   # round(x,d)
# 截取至 10^(-d), 例如: round(839.1982, 2) : 839.20;  round(839.1982, -2): 800



sum     # sum(x): x 的所有分量相加
prod    # prod(x): x 的所有分量相乘
cumsum  # 累积求和
cumprod # 累积乘积

max     # max(x): x 所有元素的最大值
min     # 最小
cummax  # 累积最大
cummin
range   # 最大及最小值

pmax; pmin    # pmax(x,y): 逐点取最大或最小

diff    # 差分


# 集合运算:

unique     # 去掉重复值
duplicated # 判断每个分量是否重复出现
> c(1:4, 1, 2, 9,1)
> unique(x)
[1] 1 2 3 4 9
> duplicated(c(1:4, 1, 2, 9,1))
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE


union      # 并集 union(1:2, c(1,3,5))  1,2,3,5
interset   # 交集
setdiff    # 集合之差
```

6 运算与操作

6.1 矩阵运算

```
A%*%B      # 矩阵乘积，A，B 为矩阵或向量
A*B        # 同阶矩阵元素相乘; 若 B 是向量, 则 A 每一列乘 B (即按循环法则)

t(A)       # 转置
solve(A)   # 逆
eigen(A)   # 特征值和特征向量
svd(A)     # 奇异值分解
qr(A)      #Q-R 分解
chol(A)    #Cholsky 分解
x%*%t(y)   #xy'
x%*%A%*%x  #x'Ax

outer(x,y, FUN="*") # 外积，(i,j) 元素 =xi*xj
## 等于 x%*%t(y) 或 x%o%y
outer(x,y, "+")     #(i,j) 元素 =xi+xj
```

矩阵行、列操作

apply(X, MARGIN, FUN,...) # 按行 (MARGIN=1) 或列 (MARGIN=2) 运行 FUN

sweep(x, MARGIN, STATS, FUN="-",...) # 按行 (MARGIN=1) 或列 (MARGIN=2) 与 STATS 进行 FUN 的运算

scale # 标准化

```
> x=matrix(1:12, 3,4, byrow=T)
> x
[,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12

> apply(x, 1, sum) # 行和 (1 代表行)
[1] 10 26 42

> apply(x, 2, sum) # 列和 (2 代表列)
[1] 15 18 21 24

> sweep(x, 2, 1:4, "*")
[,1] [,2] [,3] [,4]
[1,]    1    8   21   40
[2,]    2   10   24   44
[3,]    3   12   27   48

# 矩阵标准化
> scale(x)
```

6.2 字符操作

paste # 粘结字符
substring # 提取或替换字符向量的子串
substr
nchar
strsplit
特殊字符：

\\ , \b, \t, \n, \r

```
> paste("x", 1:4, sep="")
[1] "x1" "x2" "x3" "x4"
> paste("x", 1:4, sep="_")
[1] "x_1" "x_2" "x_3" "x_4"
> paste(letters[1:5], collapse="")
[1] "abcde"
> paste(" 今天是", date())
[1] " 今天是 Tue Mar 13 15:16:01 2012"

> substr("abcdef",2,4)
[1] "bcd"
```

6.3 排序（sort、order）

```
> x=c(3, 1, -3, 5, -2)
> sort(x) # 次序统计量
[1] -3 -2  1  3  5
> rank(x) # x 各个分量的排名 (从小到大)
[1] 4 3 1 5 2
> order(x) # 次序统计量在原数据集中的下标
[1] 3 5 2 1 4
> x[order(x)]
[1] -3 -2  1  3  5
> x
[1]  3  3 -1 -4  3  2
> y
[1] 9 8 7 6 5 4
> sort(x)
[1] -4 -1  2  3  3  3
> order(x)
[1] 4 3 6 1 2 5
> order(x,y) # 求 x 的 order，有结，按 y 再排
[1] 4 3 6 5 2 1
```

6.4 合并

```
> c(1,3)
> c("a", "d")
> c(x,y,z) # 合并向量 x,y,z
```

```
> cbind(x,y,z) # 合并成矩阵
> rbind(x,y,z)
> cbind(A, B) # 按列合并矩阵 A,B
> rbind(A,B) # 按行合并矩阵
> merge(x, y, z) # 合并 data frames x,y,z
```

6.5 分拆 (split, by, aggregate)

split(x, f) # x=vector, f=factor

by(data, INDICES, FUN, ..., simplify = TRUE) # data=dataframe, INDICES=factor, FUN=function applied.

```
> x=1:5
> group=c(1,1,2,2,1)
> split(x, group)
$‘1‘ [1] 1 2 5

$‘2‘ [1] 3 4

> warpbreaks
breaks wool tension
1      26    A       L
2      30    A       L
...
53     16    B       H
54     28    B       H
> attach(warpbreaks)
> by(warpbreaks, wool, data.frame)
> by(warpbreaks[,1], wool, mean)
```

6.6 交叉分类 (cross-classificaiotn, tabulation): table, xtabs

```
> attach(quine)
> quine
Eth Sex Age Lrn Days
1    A   M  F0  SL    2 2     A   M  F0  SL   11 ...

> table(Age)
Age F0 F1 F2 F3 27 46 40 33

> table(Sex, Age)
Age
Sex F0 F1 F2 F3
F 10 32 19 19
M 17 14 21 14
> xtabs(~Sex+Age)
Age
Sex F0 F1 F2 F3
F 10 32 19 19
M 17 14 21 14

# tapply(X, INDEX, FUN = NULL) # 对 table 的各组应用函数 FUN
```

lapply 类似，对列表的个分类应用某函数

> tapply(Days, Age, mean) # 对每个年龄组的 Days 求平均
F0 F1 F2 F3
14.85185 11.15217 21.05000 19.60606

> tapply(Days, list(Sex, Age), mean) # 对每个年龄, 性别组的 Days 求平均
F0 F1 F2 F3
F 18.70000 12.96875 18.42105 14.00000 M 12.58824 7.00000 23.42857
27.21429

---

7 自定义函数

自定义函数的一般格式如下

```
myfun = function(arg1,arg2,...) { ...  }
```

其中 function(arg1,arg2...) 中的 arg1, arg2,... 是用户自定义的函数的参数 (arguments)，大括号中包含若干命令, 最后一行为输出内容。例如计算 $x^2 + y$

```
myfun = function(x,y) {
z=x^2+y
return(z)
}
```