R的数据结构

温灿红

中国科学技术大学管理学院

R的数据结构

- 向量(vector)
- 矩阵(matrix)
- 数组(array)
- 列表(list)
- 数据框(dataframe)

向量(vector)

向量(vector)

- **向量**是将若干个基础类型相同的值存储在一起， 各个元素可以按序号访问。
- 可以是数值型向量、逻辑型向量、字符型向量等
- 用c()函数把多个元素或向量组合成一个向量。如:

```
    c(7, 8, 10, 45)                          # 数值型向量
```

```
## [1]  7  8 10 45
```

```
    c(7, 8, 10, 45) > 10                     # 逻辑型向量
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
    s <- c('abc', 'A', 1, NA, '男')          # 字符型向量
    class(s)
```

```
## [1] "character"
```

```
    factor(c("男", "女", "男", "男",  "女")) # 因子型向量
```

```
## [1] 男 女 男 男 女
## Levels: 男 女
```

构建规则序列的向量

- 也可以通过以下的方式构建有规则的向量

```
    1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
    seq(from = 5, to = 25, by = 5)
```

```
## [1]  5 10 15 20 25
```

```
    c(1:10, seq(from = 5, to = 25, by = 5))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10  5 10 15 20 25
```

```
    seq(as.Date('2021-10-1'), by='days', length=5)
```

```
## [1] "2021-10-01" "2021-10-02" "2021-10-03" "2021-10-04" "2021-10-05"
```

```
    seq(as.Date('2021-9-8'),to=as.Date('2022-1-1'),by='2 weeks')
```

```
## [1] "2021-09-08" "2021-09-22" "2021-10-06" "2021-10-20" "2021-11-03"
## [6] "2021-11-17" "2021-12-01" "2021-12-15" "2021-12-29"
```

向量运算之四则运算

- 向量与标量的运算为每个元素与标量的运算。

```
    x <- c(7, 8, 10, 45);     x * 5
```

```
## [1]  35  40  50 225
```

```
    x > 9
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

- 等长向量的运算为对应元素两两运算。

```
    y <- c(-7, -8, -10, -45);     x + y
```

```
## [1] 0 0 0 0
```

- 两个不等长向量的四则运算，规则是每次从头重复利用短的一个。

```
    x + c(-7,-8)
```

```
## [1]  0  0  3 37
```

向量运算之比较运算

- 比较运算符： <, <=, >, >=, ==, !=, %in%
- 逻辑运算符： &, |, !,

```
    x > 9
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

```
    x == -y
```

```
## [1] TRUE TRUE TRUE TRUE
```

```
    c(1, NA, 3) > 2
```

```
## [1] FALSE    NA  TRUE
```

```
    (x > 9) & (x < 20)
```

```
## [1] FALSE FALSE  TRUE FALSE
```

```
    (x > 9) | (x < 20)
```

```
## [1] TRUE TRUE TRUE TRUE
```

向量运算之集合运算

- 可以把向量看成一个集合，对两个向量进行集合运算，如unique(),setdiff(),setequal(),union(),intersect()

```
    unique(c(1,2,3,2))                # 找出唯一的元素
```

```
## [1] 1 2 3
```

```
    intersect(c(1,2,3,2), c(1,2))     # 交集
```

```
## [1] 1 2
```

```
    union(c(1,2,3,2), c(1,2))         # 并集
```

```
## [1] 1 2 3
```

```
    setdiff(c(1,2,3,2), c(1,2))       # 差集
```

```
## [1] 3
```

```
    setequal(c(1,2,3,2), c(1,2,3))    # 集合是否相等
```

```
## [1] TRUE
```

向量函数

- 数学函数：sqrt(),sign(),abs(),log(),exp()等
- 数据统计函数：mean(),median(),sd(),var(),max(),min(),length(),sum(),cumsum() summary(),range()
- 排序函数：sort(),rev(),order()
- 逻辑运算函数：identical(),all.equal(),all(),any(),which(),match(),which.max(), which.min()
- 因子类型数据汇总函数：table(),tapply()等
- 字符型向量函数： paste(),strsplit()等
- 向量长度和属性：length(),attributes(),names()

向量函数之例子（一）

```
    log(x)
```

```
## [1] 1.945910 2.079442 2.302585 3.806662
```

```
    summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.00    7.75    9.00   17.50   18.75   45.00
```

```
    sort(x, decreasing = TRUE)
```

```
## [1] 45 10  8  7
```

```
    order(x, decreasing = TRUE)
```

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

```
    rev(sort(x))
```

```
## [1] 45 10  8  7
```

向量函数之例子（二）

```
    all(x>0)
```

```
## [1] TRUE
```

```
    identical(c(0.5-0.3,0.3-0.1),c(0.3-0.1,0.5-0.3))
```

```
## [1] FALSE
```

```
    all.equal(c(0.5-0.3,0.3-0.1),c(0.3-0.1,0.5-0.3))
```

```
## [1] TRUE
```

```
    match(c(1, 3), c(2,3,4,3))      # 若能在右边找到对应的元素，则返回第一个找到的位置，否则返回
```

```
## [1] NA  2
```

```
    which(c(1, 3) %in% c(2,3,4,3)) # 找到符合条件的下标
```

```
## [1] 2
```

向量函数之例子（三）

```
table(c(1,2,3,2))
```

```
##
## 1 2 3
## 1 2 1
```

```
gender <- c("男", "女","男", "女","男", "女","男")

table(gender)
```

```
## gender
## 男 女
##  4  3
```

向量函数之例子（四）

```
    presidents = c("Clinton", "Bush", "Reagan", "Carter", "Ford")
    substr(presidents, 1, 2) # Grab the first 2 letters from each
```

```
## [1] "Cl" "Bu" "Re" "Ca" "Fo"
```

```
    paste(presidents, c("D", "R", "R", "D", "R"))
```

```
## [1] "Clinton D" "Bush R"    "Reagan R"  "Carter D"  "Ford R"
```

```
    paste(presidents, c("D", "R"))
```

```
## [1] "Clinton D" "Bush R"    "Reagan D"  "Carter R"  "Ford D"
```

```
    paste(presidents, collapse="; ")
```

```
## [1] "Clinton; Bush; Reagan; Carter; Ford"
```

```
    paste(presidents, " (", c("D", "R", "R", "D", "R"), 42:38, ")", sep="", collapse="; ")
```

```
## [1] "Clinton (D42); Bush (R41); Reagan (R40); Carter (D39); Ford (R38)"
```

向量函数之例子（五）

```
  length(x)
```

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

```
  attributes(x)
```

```
## NULL
```

向量下标和子集（一）

- 正整数下标：访问对应位置的元素和子集。如：

```
x[2]
```

```
## [1] 8
```

```
x[c(1,3)]
```

```
## [1]  7 10
```

- 负整数下标：扣除相应的元素后的子集。如：

```
x[-2]
```

```
## [1]  7 10 45
```

```
x[-c(2,4)]
```

```
## [1]  7 10
```

向量下标和子集（二）

- 下标超界： 假设向量长度为n，提取不在 $\{1, 2, \ldots, n\}$内的子集时，返回缺失值，应尽量避免。如：

```
    x[5]
```

```
## [1] NA
```

```
    x[0]
```

```
## numeric(0)
```

```
    x[6] <- 9
    x
```

```
## [1]  7  8 10 45 NA  9
```

向量下标和子集（三）

- 逻辑下标：下标可以是与向量等长的逻辑表达式，一般是关于本向量或者与本向量等长的其它向量的比较结果。如：

```
    x[x>9]
```

```
## [1] 10 45 NA
```

```
    x[which(x>9)]
```

```
## [1] 10 45
```

```
    x[!is.na(x) & x > 9]
```

```
## [1] 10 45
```

```
    c(which.min(x), which.max(x))
```

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

向量下标和子集（四）

- 元素名下标：向量可以为每个元素命名，命名后即可用元素名或者元素名向量作为向量的下标。如：

```
    names(x) <- c("v1","v2","v3","v4")
    names(x)
```

```
## [1] "v1" "v2" "v3" "v4" NA   NA
```

```
    x["v1"]
```

```
## v1
##  7
```

```
    x[c("v4", "v1")]
```

```
## v4 v1
## 45  7
```

```
    x["v5"]
```

```
## <NA>
##   NA
```

- 用字符串作为下标时，如果该字符串不在向量的元素名中，读取时返回缺失值结果，赋值时该向量会增加一个元素并以该字符串为元素名。

向量下标和子集（五）

- 重复下标：R在使用整数或元素名作为向量下标时，允许使用重复下标。如：

```
    x[c(1,3,1,2)]
```

```
## v1 v3 v1 v2
##  7 10  7  8
```

```
    gender
```

```
## [1] "男" "女" "男" "女" "男" "女" "男"
```

```
    # 把“男”记为1，“女”记为2.
    code <- c(1,2)
    names(code) <- c("男", "女")
    code[gender]
```

```
## 男 女 男 女 男 女 男
##  1  2  1  2  1  2  1
```

矩阵(matrix)

构建矩阵（一）

- 矩阵在**R**中实际上是按列存储成一个向量，根据行数和列数对应到矩阵的元素。
- matrix()函数把矩阵元素以向量的形式输入，用 nrow 和 ncol 规定行数和列数，向量元素填入的缺省次序是按列填入，用 byrow=TRUE 选项可转换成按行填入。

```
    A <- matrix(1:6, nrow=2, ncol=3); print(A)  # 数值型矩阵
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
    B <- matrix(c(1,-1, 1,1), nrow=2, ncol=2, byrow=TRUE); print(B)  # 改变元素填入顺序为按行填
```

```
##      [,1] [,2]
## [1,]    1   -1
## [2,]    1    1
```

```
    C <- matrix(LETTERS[1:6], nrow=2, ncol=3); print(C)  # 字符型矩阵
```

```
##      [,1] [,2] [,3]
## [1,] "A"  "C"  "E"
## [2,] "B"  "D"  "F"
```

构建矩阵（二）

- 也可以通过 rbind() 和 cbind() 函数把向量或者矩阵构建成新的矩阵。如：

```
    cbind(1:2,3:4,c(5,6))
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
    cbind(A, 1)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    1
## [2,]    2    4    6    1
```

```
    rbind(C, c(1,2,3))
```

```
##      [,1] [,2] [,3]
## [1,] "A"  "C"  "E"
## [2,] "B"  "D"  "F"
## [3,] "1"  "2"  "3"
```

构建矩阵（三）

- 可通过函数构建特殊矩阵，如对角阵 diag()。

```
    diag(2)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```
    diag(c(3,4))
```

```
##      [,1] [,2]
## [1,]    3    0
## [2,]    0    4
```

```
    diag(A)
```

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

矩阵的属性（一）

- attributes(A) 返回矩阵的属性，包括 dim, dimnames等

```
    rownames(A) <- c("行一", "行二"); colnames(A) <- c("列一", "列二", "列三");  attributes(A)
```

```
## $dim
## [1] 2 3
##
## $dimnames
## $dimnames[[1]]
## [1] "行一" "行二"
##
## $dimnames[[2]]
## [1] "列一" "列二" "列三"
```

```
    rownames(B) <- c("行一", "行二"); attributes(B)
```

```
## $dim
## [1] 2 2
##
## $dimnames
## $dimnames[[1]]
## [1] "行一" "行二"
##
## $dimnames[[2]]
## NULL
```

矩阵的属性（二）

- dim() 返回矩阵的行数和列数

```
dim(A)
```

```
## [1] 2 3
```

- nrow() 和 ncol() 分别返回矩阵行数和列数

```
nrow(A)
```

```
## [1] 2
```

```
ncol(A)
```

```
## [1] 3
```

矩阵的属性（三）

- dimnames() 返回矩阵的行名和列名

```
    dimnames(A)
```

```
## [[1]]
## [1] "行一" "行二"
##
## [[2]]
## [1] "列一" "列二" "列三"
```

```
    dimnames(B)
```

```
## [[1]]
## [1] "行一" "行二"
##
## [[2]]
## NULL
```

矩阵运算之四则运算（一）

- 矩阵与标量的运算为每个元素与标量的运算。如：

```
A + 1
```

```
##      列一 列二 列三
## 行一    2    4    6
## 行二    3    5    7
```

- 两个同形状的矩阵进行加、减运算， 即对应元素相加、相减， 用 A + B，A - B 表示。如：

```
A + matrix(1, nrow=2, ncol=3)
```

```
##      列一 列二 列三
## 行一    2    4    6
## 行二    3    5    7
```

```
A + B  # 返回错误：Error in A + B : non-conformable arrays
```

矩阵运算之四则运算（二）

- 对两个同形状的矩阵， 用 * 表示两个矩阵对应元素相乘(注意这不是线性代数中的矩阵乘法)， 用 / 表示两个矩阵对应元素相除。 如:

```
A * matrix(2:7, nrow=2, ncol=3)
```

```
##       列一 列二 列三
## 行一    2   12   30
## 行二    6   20   42
```

矩阵运算之矩阵乘法

- 用 %*% 表示矩阵乘法，要求左边的矩阵的列数等于右边的矩阵的行数。

```
B %*% A          # 如果是 `A %*% B`，那会出来什么结果？
```

```
##      列一 列二 列三
## 行一   -1   -1   -1
## 行二    3    7   11
```

- 矩阵与向量进行乘法运算时， 向量按需要解释成列向量或行向量。 当向量左乘矩阵时，看成行向量； 当向量右乘矩阵时，看成列向量。如：

```
A %*% c(1,2,3)
```

```
##      [,1]
## 行一   22
## 行二   28
```

```
1:2 %*% A
```

```
##      列一 列二 列三
## [1,]    5   11   17
```

矩阵算子（一）

- 转秩： t()

```
t(A)
```

```
##       行一 行二
## 列一     1    2
## 列二     3    4
## 列三     5    6
```

- 行列式： det()

```
det(B)
```

```
## [1] 2
```

矩阵算子（二）

- 矩阵的逆： `solve(A)`
- 方程求解：`solve(A, b)` 返回的是线性方程组 $Ax = b$的解。

```
    solve(B)
```

```
##      行一 行二
## [1,]  0.5  0.5
## [2,] -0.5  0.5
```

```
    solve(B, 1:2)
```

```
## [1] 1.5 0.5
```

```
    B %*% solve(B, 1:2)
```

```
##      [,1]
## 行一    1
## 行二    2
```

矩阵算子（三）

- 矩阵的内积：
  - A^TB: crossprod(A, B),
  - AB^T: tcrossprod(A,B)
  - A^TA: crossprod(A)
  - AA^T: tcrossprod(A)

```
crossprod(A, B) #  tcrossprod(A, B)  返回Error in tcrossprod(A, B) : non-conformable argume
```

```
##      [,1] [,2]
## 列一    3    1
## 列二    7    1
## 列三   11    1
```

```
t(A) %*% B
```

```
##      [,1] [,2]
## 列一    3    1
## 列二    7    1
## 列三   11    1
```

矩阵算子（四）

```
    crossprod(A)
```

```
##      列一 列二 列三
## 列一    5   11   17
## 列二   11   25   39
## 列三   17   39   61
```

```
    tcrossprod(A)
```

```
##      行一 行二
## 行一   35   44
## 行二   44   56
```

```
    A %*% t(A)
```

```
##      行一 行二
## 行一   35   44
## 行二   44   56
```

矩阵按行/列汇总函数（一）

- apply(A, i, FUN) 把矩阵 A 的每一列分别输入到函数 FUN 中，得到对应于每一维度的结果，其中 i = 1 表示对行进行运算， i = 2 表示对列进行运算。如：

```
    apply(A, 2, sum)
```

```
## 列一 列二 列三
##    3    7   11
```

```
    apply(A, 1, range)
```

```
##      行一 行二
## [1,]    1    2
## [2,]    5    6
```

矩阵按行/列汇总函数（二）

- summary()函数按列输出汇总信息

```
summary(A)
```

```
##       列一           列二           列三     
##  Min.   :1.00   Min.   :3.00   Min.   :5.00  
##  1st Qu.:1.25   1st Qu.:3.25   1st Qu.:5.25  
##  Median :1.50   Median :3.50   Median :5.50  
##  Mean   :1.50   Mean   :3.50   Mean   :5.50  
##  3rd Qu.:1.75   3rd Qu.:3.75   3rd Qu.:5.75  
##  Max.   :2.00   Max.   :4.00   Max.   :6.00
```

矩阵按行/列汇总函数（三）

- rowMeans(), colMeans(), rowSums(), colSums()等

```
    rowMeans(A)
```

```
## 行一 行二
##    3    4
```

```
    rowSums(A)/ncol(A)
```

```
## 行一 行二
##    3    4
```

矩阵下标和子集（一）

- 矩阵本质上是一个向量添加了 dim 属性，实际保存还是保存成一个向量，其中元素的保存次序是按列填入，所以和向量类似，可以通过正整数、负整数下标、逻辑以及名字下标来取子集。如：

```
    A[1:4]
```

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

```
    A[6]
```

```
## [1] 6
```

```
    A[-5]
```

```
## [1] 1 2 3 4 6
```

矩阵下标和子集（二）

- 但更常用的是通过行号和列号来提取子集，如：

```
A[1,2] # 提取A的第一行第二列的元素
```

```
## [1] 3
```

```
A[1,] # 提取A的第一行，返回一个向量
```

```
## 列一 列二 列三
##    1    3    5
```

```
A[,1] # 提取A的第一列，返回一个向量
```

```
## 行一 行二
##    1    2
```

```
A[1:2,c(1,3)] # 提取指定行、列对应的子矩阵
```

```
##      列一 列三
## 行一    1    5
## 行二    2    6
```

矩阵下标和子集（三）

- 若矩阵有列名、行名，可以通过字符型向量提取子集

```
    A["行一",]        # 取出A的行名为“行一”的行，返回一个向量
```

```
## 列一 列二 列三
##    1    3    5
```

- 逻辑下标取子集

```
    A[A>2]
```

```
## [1] 3 4 5 6
```

```
    C[C %in% c("A","C")]
```

```
## [1] "A" "C"
```

矩阵下标和子集（四）

- 提取A的第一行结果为向量，维数会有不同，drop = FALSE 可保留原有维度。

```
    A["行一",]                  # 取出A的行名为“行一”的行，返回一个向量
```

```
## 列一 列二 列三
##    1    3    5
```

```
    A["行一", , drop=FALSE]     # 取出A的行名为“行一”的行，返回一个矩阵
```

```
##      列一 列二 列三
## 行一    1    3    5
```

- 寻找矩阵中的最小元素，并返回其位置

```
    mat <- matrix(rnorm(40), 10, 4)
    which(mat == min(mat, na.rm=TRUE))  # 返回的是向量的位置
```

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

```
    which(mat == min(mat, na.rm=TRUE), arr.ind = TRUE)  # 返回行号和列号
```

```
##      row col
## [1,]   4   1
```

数组(array)

数组(array)

- 数组是矩阵的推广， **R**支持任意的n维数组
- 所有矩阵的运算（除了如矩阵乘法等特殊的运算外）和函数，都可以运用到数组中。如apply()等
- 多维数组的一般定义语法为

```
数组名 <- array(数组元素,
                dim=c(第一下标个数, 第二下标个数, ..., 第s下标个数))
```

数组的例子

```
arr <- array(1:24, dim=c(2,3,4)); arr
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]    7    9   11
## [2,]    8   10   12
##
## , , 3
##
##      [,1] [,2] [,3]
## [1,]   13   15   17
## [2,]   14   16   18
##
## , , 4
##
##      [,1] [,2] [,3]
## [1,]   19   21   23
## [2,]   20   22   24
```

数组下标和子集

- 类似于矩阵，可通过对应的维度的子集来提取数组的子集。如：

```
    arr[,2,2:3] # 多维数组在取子集时如果某一维下标是标量， 则结果维数会减少
```

```
##      [,1] [,2]
## [1,]    9   15
## [2,]   10   16
```

```
    arr[,2,2:3,drop = FALSE]  # 保持和原有数组同样的维度
```

```
## , , 1
##
##      [,1]
## [1,]    9
## [2,]   10
##
## , , 2
##
##      [,1]
## [1,]   15
## [2,]   16
```

列表(list)

列表(list)

- 不同于以往的数据结构，列表(list)是用来保存不同类型的数据。 比如， 一个元素是数值型向量， 一个元素是字符串， 一个元素是标量， 一个元素是另一个列表。
- 可通过 list() 定义列表

```
  dist <- list("exponential", 7, FALSE);  dist
```

```
## [[1]]
## [1] "exponential"
##
## [[2]]
## [1] 7
##
## [[3]]
## [1] FALSE
```

- 注意和向量区分

```
  c("exponential", 7, FALSE)
```

```
## [1] "exponential" "7"           "FALSE"
```

列表元素的命名（一）

- 可通过 names() 来命名。

```
names(dist) <- c("family", "mean", "is.symmetric")
dist
```

```
## $family
## [1] "exponential"
##
## $mean
## [1] 7
##
## $is.symmetric
## [1] FALSE
```

列表元素的命名（二）

- 也可在一开始定义的时候就命名好。

```
dist1 <- list(family = "exponential", mean = 7, is.symmetric = FALSE)
dist1
```

```
## $family
## [1] "exponential"
##
## $mean
## [1] 7
##
## $is.symmetric
## [1] FALSE
```

列表元素的访问

- 列表的一个元素也可以称为列表的一个“变量”，单个列表元素必须用两重方括号格式访问。如：

```
    dist[[1]]
```

```
## [1] "exponential"
```

```
    dist[["mean"]]
```

```
## [1] 7
```

```
    dist[[2]]^2
```

```
## [1] 49
```

- 如果使用单重方括号对列表取子集，结果还是列表而不是列表元素。如：

```
    dist[2]^2   # 无法直接对列表进行四则运算
```

列表元素的增加

- 直接给列表不存在的元素名定义元素值就添加了新元素。如：

```
dist$was.estimated <- FALSE  # 美元符号和双中括号的方式是等价的
dist[['sd']] <- 1;
dist
```

```
## $family
## [1] "exponential"
##
## $mean
## [1] 7
##
## $is.symmetric
## [1] FALSE
##
## $was.estimated
## [1] FALSE
##
## $sd
## [1] 1
```

列表元素的删除（一）

- 把某个列表元素赋值为 NULL 就删掉这个元素。如：

```
dist[['was.estimated']] <- NULL; dist
```

```
## $family
## [1] "exponential"
##
## $mean
## [1] 7
##
## $is.symmetric
## [1] FALSE
##
## $sd
## [1] 1
```

列表元素的删除（二）

- 要把已经存在的元素修改为 NULL 值而不是删除此元素， 或者给列表增加一个取值为 NULL 的元素， 这时需要用单重的方括号取子集，这样的子集会保持其列表类型，给这样的子列表赋值为 list(NULL)。如：

```
dist['was.estimated'] <- list(NULL); dist
```

```
## $family
## [1] "exponential"
##
## $mean
## [1] 7
##
## $is.symmetric
## [1] FALSE
##
## $sd
## [1] 1
##
## $was.estimated
## NULL
```

列表类型的转换

- 用 as.list() 把一个其它类型的对象转换成列表。

```
as.list(1:2)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
```

- 用 unlist() 函数把列表转换成基本向量。

```
unlist(list(a = 1, b = 2:4))
```

```
##  a b1 b2 b3
##  1  2  3  4
```

数据框(dataframe)

数据框(dataframe)

- 数据框(data.frame)存储似于数据库表或Excel数据表的数据。
- 类似于矩阵，有 n行， p列，但各列之间允许有不同的类型，同一列中的元素保持相同类型。
- 每一行可看作一个列表，每一列可看作一个向量。
- data.frame()生成数据框

```
d <- data.frame(A,logicals=c(TRUE,FALSE), character = c("男","女"))
d
```

```
##      列一 列二 列三 logicals character
## 行一    1    3    5     TRUE        男
## 行二    2    4    6    FALSE        女
```

数据框的命名

```
    names(d)     # 默认名字为列名
```

```
## [1] "列一"     "列二"     "列三"     "logicals" "character"
```

```
    rownames(d)  # 提取行名
```

```
## [1] "行一" "行二"
```

```
    rownames(d) <- c("R1", "R2")
    d
```

```
##    列一 列二 列三 logicals character
## R1    1    3    5     TRUE        男
## R2    2    4    6    FALSE        女
```

数据框的增加行或列

- 可通过 rbind() 或 cbind() 来增加

```
    rbind(d,list(列一=-3, 列二=-5, 列三=-7, logicals=TRUE, character="男"))
```

```
##    列一 列二 列三 logicals character
## R1    1    3    5     TRUE        男
## R2    2    4    6    FALSE        女
## 1    -3   -5   -7     TRUE        男
```

```
    cbind(d, date = as.Date(c("2021/09/10", "2021/09/14")))
```

```
##    列一 列二 列三 logicals character       date
## R1    1    3    5     TRUE        男 2021-09-10
## R2    2    4    6    FALSE        女 2021-09-14
```

数据框内容访问（一）

- 数据框中的内容可采用类似于矩阵的方式进行访问。如：

```
    d[2,3]
```

```
## [1] 6
```

- 访问行

```
    d[1,]
```

```
##    列一 列二 列三 logicals character
## R1    1    3    5     TRUE        男
```

- 因为数据框的一行不一定是相同数据类型， 所以数据框的一行作为子集， 结果还是数据框，而不是向量。

```
    is.data.frame(d[1,])
```

```
## [1] TRUE
```

数据框内容访问（二）

- 访问列

```
    d[, 2]
```

```
## [1] 3 4
```

```
    d[,"列二"]
```

```
## [1] 3 4
```

```
    d[["列二"]]
```

```
## [1] 3 4
```

```
    d$列二
```

```
## [1] 3 4
```

数据框内容访问（三）

- 同时取行子集和列子集

```
    d[1:2,3:4]
```

```
##    列三 logicals
## R1    5     TRUE
## R2    6    FALSE
```

```
    d[d[,"character"]=="男",3:5]
```

```
##    列三 logicals character
## R1    5     TRUE        男
```

with() 函数

- 在对数据框中的数据进行运算时，常常需要提取多列，这时候用 with() 函数能够很好的简化代码。
- 如以下两种方式是等价的：

```
    (d$列一 + d$列二) / exp(d$列三)
```

```
## [1] 0.02695179 0.01487251
```

```
    with(d, (列一 + 列二) / exp(列三))
```

```
## [1] 0.02695179 0.01487251
```

数据框与矩阵的区别

- 数据框不能作为矩阵参加矩阵运算。 需要时，可以用 as.matrix() 函数转换数据框或数据框的子集为矩阵。
- 可通过 ad.data.frame() 把矩阵转化成data.frame，以方便用在回归模型等的函数命令中（因为这些都需要输入的是data.frame而不是matrix）。

```
  as.matrix(d)       # 转化成字符型矩阵，无法进行矩阵运算
```

```
##    列一 列二 列三 logicals character
## R1 "1"  "3"  "5"  "TRUE"    "男"
## R2 "2"  "4"  "6"  "FALSE"   "女"
```

```
  as.matrix(d[,1:4]) # 逻辑型数据直接转化成0，1数值型数据
```

```
##    列一 列二 列三 logicals
## R1    1    3    5        1
## R2    2    4    6        0
```

谢 谢