# Ch2.4R的数据结构

本章概要

*♢* **R** 的对象与属性

*♢* 浏览对象信息

*♢* 向量的建立

*♢* 数组与矩阵的建立

*♢* 数据框的建立

*♢* 列表的建立

*♢* 时间系列的建立

2.4.1 **R的对象与属性**

R的所有对象都有两个内在属性：类型与长度。

1. 类型是对象元素的基本种类，共有四种：数值型、字符型、复数型和逻辑型(，通过函数mode()来得到。
2. 长度是对象中元素的数目，通过函数length()来得到。

如：

> x<-1

> mode(x)

[1] "numeric"

> length(x)

[1] 1

> A<-"Gomphotherium";compar<-TRUE;z<-1i

> mode(A);mode(compar);mode(z)

[1] "character"

[1] "logical"

[1] "complex"

NA(Not Available)表示缺失数据，无论什么类型;很大的数用科学计数法，如

> a<-NA

> a

[1] NA

> n<-2.1e23

> n

[1] 2.1e+23

#Inf表示正无穷的数值，-Inf表示负无穷的数值，NAN表示不是数字的值，如

> x<-5/0

> x

[1] Inf

> exp(x)

[1] Inf

> exp(-x)

[1] 0

> Inf+Inf

[1] Inf

> Inf-Inf

[1] NaN

> 0/0

[1] NaN

> sqrt(-16)

[1] NaN

警告信息：

In sqrt(-16) : 产生了NaNs

> sqrt(-16+0i) #按照复数运算

[1] 0+4i

字符型的值输入时可用双引号"界定，如果其中含有双引号的话，可在双引号前加上反斜杠“*\*”, 在某些函数如cat( )的输出显示或write.table( )写入磁盘时会被以特殊的方式处理. 例如

> x<-"Double quotes\"delimitate R's strings."

> x

[1] "Double quotes\"delimitate R's strings."

> cat(x)

Double quotes"delimitate R's strings.

字符串也可以用单引号来界定，如果字符串中含有单引号，则需要用用反斜杠来引用，双引号反而不用反斜杠。

x<-'Double quotes "delimitate R\'s strings.'

> x

[1] "Double quotes \"delimitate R's strings."

> cat(x)

Double quotes "delimitate R's strings.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表2.1 数据对象及类型 | | | | |
| 对象 | 类型 | | 是否允许同一对象中有多个类型？ | |
| 向量 | 数值型，字符型，复数型，逻辑型 | | 否 | |
| 矩阵 | 数值型，字符型，复数型，逻辑型 | | 否 | |
| 数组 | 数值型，字符型，复数型，逻辑型 | | 否 | |
| 因子 | 数值型，字符型 | | 否 | |
| 数据框 | 数值型，字符型，复数型，逻辑型 | | 是 | |
| 时间序列 | 数值型，字符型，复数型，逻辑型 | | 否 | |
| 列表 | 数值型，字符型，复数型，逻辑型  函数，表达式 | | 是 | |
| 表2.2 运算符 | | | | | |
| 数学运算 | | 比较运算 | | 逻辑运算 | |
| + 加法 | | < 小于 | | ！x 逻辑非 | |
| * 减法 | | > 大于 | | x&y 逻辑与 | |
| \* 乘法 | | <= 小于或等于 | | x&&y 同上 | |
| / 除法 | | >= 大于或等于 | | x|y 逻辑或 | |
| ^ 乘方 | | == 等于 | | x||y 同上 | |
| %% 两数相除的[余数](http://baike.baidu.com/view/1068391.htm) | | != 不等于 | | xor(x,y) 异或 | |
| %/% 两数相除的整数商 | |  | |  | |

注：异或，英文为exclusive OR，或缩写成xor。异或（[xor](http://baike.baidu.com/view/64546.htm)）是一个数学运算符。它应用于逻辑运算。异或的数学符号为“⊕”，计算机符号为“xor”。其运算法则为：

a⊕b = (¬a ∧ b) ∨ (a ∧¬b)

如果a、b两个值不相同，则异或结果为1(TRUE)。如果a、b两个值相同，异或结果为0(FALSE)。

> 9%%3

[1] 0

> 4%%3

[1] 1

> 9%/%3

[1] 3

> 4%/%3

[1] 1

>

> x<-FALSE;y<-FALSE;z<-TRUE;w<-TRUE

> !x; x&y ;y|z;z|w

[1] TRUE

[1] FALSE

[1] TRUE

[1] TRUE

> xor(x,y);xor(z,w)

[1] FALSE

[1] FALSE

> xor(x,z);xor(y,z)

[1] TRUE

[1] TRUE

> >

#注意：算术运算、比较运算、逻辑运算的运算优先级别（依次降低）。FALSE当作0,TRUE当作1

> 7!=6+1

[1] FALSE

> !(7==6)

[1] TRUE

> !(7==6)==2

[1] TRUE

> (7==9)|(7>0)

[1] TRUE

> (7==9)&(7>0)

[1] FALSE>

2.4.2 **浏览R的对象信息**

**1.函数ls()**

功能：列出内存中的所有对象名

注：选项pattern(可简写为pat)= " 指定字符"，可设定只显示出指定字符的对象名

选项pattern(可简写为pat)= "^ 指定字符"，可设定只显示出指定字符**开头**的对象名

主菜单：MISC → Remove all objects

> name<-"Carmen";n1<-10;n2<-100;m<-0.5

> ls()

[1] "m" "n1" "n2" "name"

> ls(pat="m") #列出名称中带有某个指定字符的对象

[1] "m" "name"

> ls(pat="^m") #列出名称中指定字符"m"开头的的对象

[1] "m"

>

**2.函数ls.str()**

功能：列出内存中所有对象的详细信息

> ls.str()

m : num 0.5

n1 : num 10

n2 : num 100

name : chr "Carmen"

> M<-data.frame(n1,n2,m)

> ls.str()

m : num 0.5

M : 'data.frame': 1 obs. of 3 variables:

$ n1: num 10

$ n2: num 100

$ m : num 0.5

n1 : num 10

n2 : num 100

name : chr "Carmen"

> ls.str(pat="M")

M : 'data.frame': 1 obs. of 3 variables:

$ n1: num 10

$ n2: num 100

$ m : num 0.5

>

**3.函数rm()**

功能：删除内存中的指定对象

ls()

rm(n1,n2) #删除对象n1,n2

ls()

rm(list=ls()) #删除所有对象

ls()

rm(list=ls(pat="^m")) #删除以指定字符“m”开头的对象

2.4.3向量的建立

1.数值型向量的建立

**1) 函数seq( )或“:”**

功能： 建立向量(序列)具有较为简单的规律

> 1:10

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

> 10:1

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

> 1:10-1 #等价于向量减法

[1] 0 1 2 3 4 5 6 7 8 9

> 1:(10-1) #注意括号有无的区别

[1] 1 2 3 4 5 6 7 8 9

> z<-seq(1,5,by=0.5) #等价于z<-seq(from=1,to=5,by=0.5)

> z<-seq(1,10,length=5)

> z

[1] 1.00 3.25 5.50 7.75 10.00

**2)函数 rep( )**

功能： 建立向量(序列)具有较为复杂的规律

> z<-rep(2,4) #等价于z<-rep(2,times=4)

> z

[1] 2 2 2 2

> z<-rep(2:5,2) #等价于z<-rep(2:5,times=2)

> z

[1] 2 3 4 5 2 3 4 5

> z<-rep(2:5,rep(2,4))

> z

[1] 2 2 3 3 4 4 5 5

> z<-rep(2:5,1:4)

> z

[1] 2 3 3 4 4 4 5 5 5 5

> z<-rep(1:3,times=4,each=2)

> z

[1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3

>

**3)函数 c( )**

功能： 建立向量(序列)没有什么规律

> c(3,5)

[1] 3 5

> c(2,4)

[1] 2 4

> rep(c(3,5),c(2,4))

[1] 3 3 5 5 5 5

**4)函数 scan()**

功能： 通过键盘逐个输入建立向量(序列)

> z<-scan()

1: 1

2: 2.3

3: 3.5

4: 3.5

5:

Read 4 items

> z

[1] 1.0 2.3 3.5 3.5

> z<-sequence(3:5)

> z

[1] 1 2 3 1 2 3 4 1 2 3 4 5

> z<-sequence(c(10,5)) #等价于c(seq(1,10),seq(1,5)

> z

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

2.字符型向量的建立

**1) 函数c( )**

功能：连接几个字符串形成向量,数字会被转化为字符看待。

> z<-c("green","blue sky","-99")

> z

[1] "green" "blue sky" "-99"

>

**2)函数 paste( )**

功能：顺次取出各部分参数的字符用分隔符连接，形成字符串向量，默认的连接符为空字符串

> labs<-paste(c("X","Y"),1:10,sep="") #尝试分隔符sep="\_" 和多几个空格

> labs

[1] "X1" "Y2" "X3" "Y4" "X5" "Y6" "X7" "Y8" "X9" "Y10"

> paste(c("X","Y"),1:10)

[1] "X 1" "Y 2" "X 3" "Y 4" "X 5" "Y 6" "X 7" "Y 8" "X 9" "Y 10"

> paste(c("X","Y"),1:10） #注意这两个语句的区别，最后一个反括号不一样

3.逻辑型向量的建立

> x<-c(10.4,5.6,3.1,6.4,21.7)

> temp<-x>13

> temp

[1] FALSE FALSE FALSE FALSE TRUE

>

**逻辑向量由条件可给，如**

4.因子型向量的建立

因子型向量：是一个分类变量，包括变量不同的可能水平（即使它们不在数据中出现）。可用spss软件中的值标签来解释什么是因子型向量

1. **函数factor( )**

功能：将给定对象(数值或字符型向量)转化成因子型向量

* 字符型向量转换为因子型向量

> a<-c("green","blue","green","yellow")

> a<-factor(a)

> a

[1] green blue green yellow

Levels: blue green yellow

* 数值型向量转换为因子型向量

>b<-c(1,2,3,1)

> b<-factor(b)

> b

[1] 1 2 3 1

Levels: 1 2 3

* 字符型因子转换为数值型因子

> a<-c("green","blue","green","yellow");a

[1] "green" "blue" "green" "yellow"

> a<-factor(a);a

[1] green blue green yellow

Levels: blue green yellow

> levels(a)<-c(1,2,3)

> a

[1] 2 1 2 3

Levels: 1 2 3

> ff<-factor(c("A","B","C"),label=c(1,2,3))

> ff

[1] 1 2 3

Levels: 1 2 3

>

* 数值型因子转换为字符型因子

>b<-c(1,2,3,1);b

[1] 1 2 3 1

> b<-factor(b);b

[1] 1 2 3 1

Levels: 1 2 3

> levels(b)<-c("low","middle","high");b

[1] low middle high low

Levels: low middle high

>

> ff<-factor(1:3,labels=c("A","B","C"));ff

[1] A B C

Levels: A B C

**注：#函数levels()，用来提取一个因子中可能的水平值**

> ff<-factor(c(2,4),levels=2:5)

> ff

[1] 2 4

Levels: 2 3 4 5

> levels(ff)

[1] "2" "3" "4" "5"

>

1. **函数gl(),**按指定的因子水平模式来产生规则的因子序列

格式：**gl(k,n ,length=,label=) ,k是水平数,n是每个水平的重复次数,length指定产生的数据个数，label指定每个水平因子的名称**

> gl(3,5)

[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3

Levels: 1 2 3

> gl(3,5,length=30)

[1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3

Levels: 1 2 3

> gl(2,6,label=c("Male","Female"))

[1] Male Male Male Male Male Male Female Female Female Female

[11] Female Female

Levels: Male Female

> gl(2,10)

[1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2

Levels: 1 2

> gl(2,1,length=20)

[1] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2

Levels: 1 2

> gl(2,2,length=20)

[1] 1 1 2 2 1 1 2 2 1 1 2 2 1 1 2 2 1 1 2 2

Levels: 1 2

>

5.数值型向量的运算

* 向量与一个常数的加、减、乘、除为向量的每一个元素与此常数进行加、减、乘、除;
* 向量的乘方( ˆ ) 与开方(sqrt) 为每一个元素的乘方与开方, 这对像log*,* exp*,* sin*,* cos*,* tan 等普通的运算函数同样适用;
* 同样长度向量的加、减、乘、除等运算为对应元素进行加、减、乘、除等;

>5+c(4,7,17)

[1] 9 12 22

> 5\*c(4,7,17)

[1] 20 35 85

> c(-1,3,-17)+c(4,7,17)

[1] 3 10 0

> c(1,2,3)\*c(4,5,6)

[1] 4 10 18

> c(2,4,5)^2

[1] 4 16 25

> sqrt(c(2,4,25))

[1] 1.414214 2.000000 5.000000

* 不同长度向量的加、减、乘、除遵从循环法则(recycling rule), 但要注意这种场合通常要求向量的长度为倍数关系, 否则会出现警告: “长向量并非是短向量的整数倍”.

循环法则：表达式中较短的向量会根据最长向量的长度被重复使用若干次，而常数将被不断重复使用，直到长度与最长向量相匹配。

> x<-c(1,2,3);x

[1] 1 2 3

> y<-c(x,0,x);y

[1] 1 2 3 0 1 2 3

> z<-2\*x+y+1;z

Warning message:

In 2 \* x + y :

longer object length is not a multiple of shorter object length

[1] 4 7 10 3 6 9 6

> 1:2+1:4

[1] 2 4 4 6

> 1:4+1:7 #重复次数非整数倍时会被警告

[1] 2 4 6 8 6 8 10

Warning message:

In 1:4 + 1:7 :

longer object length is not a multiple of shorter object length

6.常用统计函数P27

#常用统计函数

x<-c(3,5,1,4,2)

max(x) #求最大值

min(x) #求最小值

which.max(x) #求最大值的下标

which.min(x) #求最小值的下标

mean(x) #求均值

median(x) #求中位数

var(x) #求方差

sd(x) #求标准差

range(x) #返回最大值与最小值组成的向量

summary(x) #六个常用描述性统计量（最小，最大，平均值、中位数、第一个四分位数、第三个四分位数）

IQR(x) #计算四分位极差=第三个四分位数-第一个四分位数

quantile(x) #返回常用的分位数

quantile(x,0.9) #返回90%分位数

length(x) #返回向量长度

sum(x) #求元素的和

prod(x) #求元素的积

rev(x) #返回各元素的逆序排列

sort(x) #将向量元素按升序排列，选项decreasing=TRUE表示降序

order(x) #返回向量元素升序排列后的秩，选项decreasing=TRUE表示降序的秩

rank(x) #返回各元素的秩，可设定结的处理方式 ties.method =c("average", "first", "random", "max", "min"))

cumsum(x) #返回累积和组成的向量

cumprod(x) #返回累积积组成的向量

cummin(x) #返回累积最小值组成的向量

cumman(x) #返回累积最大值组成的向量

y<-c(1,2,3,6,5)

var(x,y) #求样本（向量）x和y的协方差

cov(x,y) #求样本（向量）x和y的协方差

cor(x,y) #求样本（向量）x和y的相关系数

7. 向量的下标与子集（元素）的提取

向量的下标与子集（元素）的提取,可通过在向量后追加一个方括号中的索引向量。有以下四类:

1）索引向量为正整数向量：提取向量中对应元素

> x<-c(11,22,33,44,55,66,77,88,99,100)

> x[1:3]

[1] 11 22 33

> x[4]

[1] 44

> x[c(1, 4)]

[1] 11 44

2）索引向量为负整数向量：去掉向量中对应元素

> y<-x[-(1:5)]

> y

[1] 66 77 88 99 100

3）索引向量为字符串向量：针对拥有name属性的向量并由它区分向量中的元素，如

> fruit<-c(5,10,1,20)

> names(fruit)<-c("orange","banana","apple","peach")

> fruit

orange banana apple peach

5 10 1 20

> lunch<-fruit[c("apple","orange")]

> lunch

apple orange

1 5

4)索引向量为逻辑向量：取出索引向量中返回值为TRUE所对应的元素，忽视FALSE所对应的元素

|  |
| --- |
| > x<-c(42,7,64,9)  > x>10  [1] TRUE FALSE TRUE FALSE  > x[x>10]  [1] 42 64  > x[x<40&x>10]  numeric(0)  > x[x>10]<-10  > x  [1] 10 7 10 9  > y<-runif(100,min=0,max=1) #(0,1)上100个均匀分布随机数  > y  [1] 0.53076753 0.97830478 0.38423384 0.72465699 0.67323699 0.73030272 0.40572310  ……………………………………………  > sum(y<0.5) #值小于0.5的元素的个数  [1] 50  > sum(y[y<0.5]) #值小于0.5的元素的值的和  [1] 12.53433  > x<-c(2,-3,4,NA)  > y<-x[!is.na(x)] #x中的非缺失值  > y  [1] 2 -3 4  > z<-x[(!is.na(x))&(x>0)] #x中的非负非缺失值  > z  [1] 2 4 |
|  |
| |  | | --- | |  | |

2.4.4数组与矩阵的建立

向量和矩阵都是数组的特例;向量、矩阵、数组中的所有元素必须是同一类型的。矩阵和数组，除了类型（mode）和长度(length)属性外，还多了一个维数(dim)属性。

1．矩阵的建立

* **函数**matrix **()**

**格式：matrix(data=向量,nr=行数,nc= 列数) 或matrix(向量,行数,列数)**

功能： 以data向量的数据为元素，建立指定行数和列数的矩阵

注意：填充方式为列

|  |
| --- |
| > matrix(1:6,nr=2,nc=3) #可简写为matrix(1:6,2,3)  [,1] [,2] [,3]  [1,] 1 3 5  [2,] 2 4 6  > matrix(1,nr=2,nc=3)  [,1] [,2] [,3]  [1,] 1 1 1  [2,] 1 1 1 |
|  |
| |  | | --- | |  | |

* **函数**diag**()**

**格式：diag(data=向量,nr=行数,nc= 列数) 或diag(向量,行数,列数)**

功能： 以data向量的数据为对角线上元素，建立指定行数和列数的对角矩阵

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| diag(3) #生成单位阵  [,1] [,2] [,3]  [1,] 1 0 0  [2,] 0 1 0  [3,] 0 0 1  > v<-c(10,20,30)  > diag(v) #等价于diag(c(10,20,30))  [,1] [,2] [,3]  [1,] 10 0 0  [2,] 0 20 0  [3,] 0 0 30   |  | | --- | | > diag(2.5,nr=3,nc=5)  [,1] [,2] [,3] [,4] [,5]  [1,] 2.5 0.0 0.0 0 0  [2,] 0.0 2.5 0.0 0 0  [3,] 0.0 0.0 2.5 0 0 | |  | | |  | | --- | | > | | |
|  |
| |  | | --- | |  | |

* **函数**rowname**()、colnames()：返回矩阵的行（列）名称（相当于标签）**
* **函数dim（）：返回矩阵的维数**

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | X<-matrix(1:4,2,2)  > X  [,1] [,2]  [1,] 1 3  [2,] 2 4  > rownames(X)<-c("a","b")  > colnames(X)<-c("c","d")  > X  c d  a 1 3  b 2 4  > dim(X)  [1] 2 2  > dimnames(X)  [[1]]  [1] "a" "b"  [[2]]  [1] "c" "d" | | |  | | --- | |  | | |
|  |
| |  | | --- | |  | |

注意事项：

* #循环准则适用于矩阵，要求数据个数等于矩阵列数的倍数，否则警告
* #默认按列填充，矩阵中数据填充方式可通过byrow=FALSE(缺省方式）或TRUE来设定。

|  |
| --- |
| > X<-matrix(1:4,2,4) #默认按列填充，循环准则  > X  [,1] [,2] [,3] [,4]  [1,] 1 3 1 3  [2,] 2 4 2 4  > X<-matrix(1:4,2,3) #数据项的个数不等于矩阵列数和倍数时会出现警告  Warning message:  In matrix(1:4, 2, 3) :  data length [4] is not a sub-multiple or multiple of the number of columns [3]  > X  [,1] [,2] [,3]  [1,] 1 3 1  [2,] 2 4 2  > X<-matrix(1:4,2,4,byrow=TRUE) #按行填充  > X  [,1] [,2] [,3] [,4]  [1,] 1 2 3 4  [2,] 1 2 3 4 |
|  |
| |  | | --- | | > | |

2数组的建立

函数array（data,dim,dimnames）

功能：以data向量为数组元素，dim向量为维数向量（数值型向量），dimnames向量为各维名称组成的字符型向量（可缺省为空），从而建立一个数组。

注意：数据元素填充方式是先行、后列、再页

> A<-array(1:8,dim=c(2,2,2)) #可简写为array(1:8,c(2,2,2))

> A

, , 1

[,1] [,2]

[1,] 1 3

[2,] 2 4

, , 2

[,1] [,2]

[1,] 5 7

[2,] 6 8

> dim(A)

[1] 2 2 2

> dimnames(A)<-list(c("a","b"),c("c","d"),c("e","f"))

> A

, , e

c d

a 1 3

b 2 4

, , f

c d

a 5 7

b 6 8

> colnames(A)

[1] "c" "d"

> rownames(A)

[1] "a" "b"

> dimnames(A)

[[1]]

[1] "a" "b"

[[2]]

[1] "c" "d"

[[3]]

[1] "e" "f"

* 向量、矩阵作为数组的特例

|  |
| --- |
| > A<-array(1:6) #返回尽可能维数低的对象：向量  > A  [1] 1 2 3 4 5 6  > A<-array(1:6,c(2,3))  > A  [,1] [,2] [,3]  [1,] 1 3 5  [2,] 2 4 6  > A<-array(1:4,c(2,3)) #数据项不够  > A  [,1] [,2] [,3]  [1,] 1 3 1  [2,] 2 4 2  > A<-array(1:8,c(2,3)) #数据项有多的  > A  [,1] [,2] [,3]  [1,] 1 3 5  [2,] 2 4 6  > |
|  |
| |  | | --- | | > | |

3 数组与矩阵的下标与子集（元素）的提取

同向量一样，矩阵和数组的下标也可用正整数、负整数、逻辑表达式，从而实现子集的提取或修改。如

* 提取若干行（列）元素

> X<-matrix(1:6,2,3)

> X

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

> X[2,2] #提取一个元素

[1] 4

> X[2,] #提取一行元素

[1] 2 4 6

> X[,2] #提取一列元素

[1] 3 4

> X[,2,drop=FALSE] #默认的是drop=TRUE，返回维数尽可可能低的对象

[,1]

[1,] 3

[2,] 4

> X[,c(1,3)] #提取若干列元素

[,1] [,2]

[1,] 1 5

[2,] 2 6

> X[,2:3]

[,1] [,2]

[1,] 3 5

[2,] 4 6

* 去掉一个或若干个行（列）

> X<-matrix(1:6,2,3)

> X[-1,]

[1] 2 4 6

> X[,-2]

[,1] [,2]

[1,] 1 5

[2,] 2 6

* 修改或替换元素

> X<-matrix(1:6,2,3)

> X[,3]<-NA

> X

[,1] [,2] [,3]

[1,] 1 3 NA

[2,] 2 4 NA

> X[is.na(X)]<-0 #缺失值用0代替

> X

[,1] [,2] [,3]

[1,] 1 3 0

[2,] 2 4 0

>

1. 矩阵运算函数

（1）代数运算

* 转置函数t()

> X<-matrix(1:6,2,3);X

[,1] [,2] [,3]

[1,] 1 3 5

[2,] 2 4 6

> t(X)

[,1] [,2]

[1,] 1 2

[2,] 3 4

[3,] 5 6

* 提取对角线的元素diag()

> X<-matrix(1:4,2,2);X

[,1] [,2]

[1,] 1 3

[2,] 2 4

> diag(X)

[1] 1 4

* 几个矩阵按行合并rbind()与按列合并cbind()

> m1<-matrix(1,nr=2,nc=2);m1

[,1] [,2]

[1,] 1 1

[2,] 1 1

> m2<-matrix(2,nr=2,nc=2);m2

[,1] [,2]

[1,] 2 2

[2,] 2 2

> rbind(m1,m2)

[,1] [,2]

[1,] 1 1

[2,] 1 1

[3,] 2 2

[4,] 2 2

> cbind(m1,m2)

[,1] [,2] [,3] [,4]

[1,] 1 1 2 2

[2,] 1 1 2 2

>

* 矩阵的逐元乘积“\*”

> m2\*m2

[,1] [,2]

[1,] 4 4

[2,] 4 4

> >

* 矩阵的代数乘积“%\*%”

> rbind(m1,m2)%\*%cbind(m1,m2)

[,1] [,2] [,3] [,4]

[1,] 2 2 4 4

[2,] 2 2 4 4

[3,] 4 4 8 8

[4,] 4 4 8 8

> cbind(m1,m2)%\*%rbind(m1,m2)

[,1] [,2]

[1,] 10 10

[2,] 10 10

>

* 方阵的行列式det()

> X<-matrix(1:4,2);X

[,1] [,2]

[1,] 1 3

[2,] 2 4

> det(X)

[1] -2>

* 略其它函数：求交叉乘积crossprod(),特征根与特征向量eigen(X)，QR分解qr()

（2）统计运算函数

* 函数max()、min()、var()。。cumsum()等统计函数对于矩阵的运行结果有方向性，默认是按列计算

> a<-matrix(c(21,31,43,56,40,23),3,2)

> a

[,1] [,2]

[1,] 21 56

[2,] 31 40

[3,] 43 23

> max(a)

[1] 56

> cumsum(a)

[1] 21 52 95 151 191 214

> var(a) #按列计算的协方差矩阵

[,1] [,2]

[1,] 121.3333 -181.6667

[2,] -181.6667 272.3333

> cov(a) #按列计算的协方差矩阵

[,1] [,2]

[1,] 121.3333 -181.6667

[2,] -181.6667 272.3333

> cor(a) #按列计算的相关系数矩阵

[,1] [,2]

[1,] 1.0000000 -0.9993895

[2,] -0.9993895 1.0000000

* 函数apply(X，MARGIN,FUN),对一个对象X按MARGIN（行或列）施加某种运算

> m<-matrix(rnorm(n=12),nrow=3)

> m

[,1] [,2] [,3] [,4]

[1,] -1.9310402 -1.5511957 -0.05394811 0.2806861

[2,] -0.2375811 1.0683468 1.00190358 -0.8780563

[3,] 1.2997472 -0.4325886 0.35878069 0.5013870

> apply(m,MARGIN=1,FUN=mean) #求各行的均值

[1] -0.8138745 0.2386533 0.4318316

* 标准化函数scale(),默认要求中心化（即每个元素减去均值，标准化（即减去均值后除以标准差）

> scale(m,center=T,scale=T)

[,1] [,2] [,3] [,4]

[1,] -1.01571338 -0.94798949 -0.9199923 0.4219857

[2,] 0.03220476 1.04494741 1.0643227 -1.1418237

[3,] 0.98350861 -0.09695793 -0.1443304 0.7198380

attr(,"scaled:center")

[1] -0.28962470 -0.30514583 0.43557872 -0.03199438

attr(,"scaled:scale")

[1] 1.6160223 1.3144132 0.5320988 0.7409742

* 函数sweep(X，MARGIN,STATS,FUN),对一个对象按MARGIN计算STATS,并从中施加某种运算

> row.med<-apply(m,MARGIN=1,FUN=median)

> row.med

[1] -0.8025719 0.3821612 0.4300839

> sweep(m,MARGIN=1,STATS=row.med,FUN="-") #减去中位数

[,1] [,2] [,3] [,4]

[1,] -1.1284683 -0.7486238 0.74862381 1.08325802

[2,] -0.6197423 0.6861856 0.61974233 -1.26021750

[3,] 0.8696633 -0.8626725 -0.07130316 0.07130316

2.4.5数据框（dataframe）的建立

数据框：一个完整的数据集是由若干个变量的若干个观测值组成的，在R里称为一个数据框。它也是一个数据对象，也具有dim属性，具有行、列两个维度（行表示变量，列表示观测），且各个变量的观测值有相同的长度，但可以是不同的数据类型。

1．数据框的直接建立

* **函数data.frame（）：由一些向量直接建立数据框**

> x<-c(42,7,64,9)

> y<-1:4

> z.df<-data.frame(INDEX=y,VALUE=x) #尝试简化形式data.frame(y,x)

> z.df

INDEX VALUE

1 1 42

2 2 7

3 3 64

4 4 9

* 数据框中的向量必须有相同的长度或长度有倍数关系，短的向量按循环法则重复整数次

> weight<-c(70.6,56.4,80,59.5)

> x<-c("adult","teen")

> wag<-data.frame(weight,age=x)

> wag

weight age

1 70.6 adult

2 56.4 teen

3 80.0 adult

4 59.5 teen

* x,y的长度不是倍数关系时，不能生成数据框，会出错警告提醒

> y<-2:4

> z<-data.frame(x,y)

Error in data.frame(x, y) :

arguments imply differing number of rows: 4, 3

> z

Error: object 'z' not found

2．数据框的间接建立

* **函数read.table（）：读取数据文件建立数据框**

#注意：数据文件中最后一行必须是空行

> foo<-read.table(file="E:/Rprogram/data/foo.txt",header=T)

> foo

treat weight

1 A 3.4

2 B NA

3 A 5.8

3．适用于数据框的函数

* **常用的统计计算函数max()，min(),var(),,cov(),cor()...等同样适用于数据框，注意：数值型的数据。**

> x<-c(1,4,5,6,4)

> y<-c(2,3,5,3,4)

> z<-data.frame(x,y);z

x y

1 1 2

2 4 3

3 5 5

4 6 3

5 4 4

> max(z)

[1] 6

> min(z)

[1] 1

> cumsum(z)

x y

1 1 2

2 5 5

3 10 10

4 16 13

5 20 17

> cov(z)

x y

x 3.50 1.25

y 1.25 1.30

> cor(z)

x y

x 1.000000 0.586009

y 0.586009 1.000000

> sd(z) #会出错，

Error in is.data.frame(x) :

(list) object cannot be coerced to type 'double'

> var(z) #var(z)=cov(z)

x y

x 3.50 1.25

y 1.25 1.30

* **函数summary(),pairs(),xtable()的使用（注：描述性统计部分会再讲，掌握数据集的使用）**

> attach(Puromycin) #挂接数据集使之激活

The following objects are masked from Puromycin (pos = 3):

conc, rate, state

> help(Puromycin)

> Puromycin

conc rate state

1 0.02 76 treated

2 0.02 47 treated

3 0.06 97 treated

4 0.06 107 treated

5 0.11 123 treated

6 0.11 139 treated

7 0.22 159 treated

8 0.22 152 treated

9 0.56 191 treated

10 0.56 201 treated

11 1.10 207 treated

12 1.10 200 treated

13 0.02 67 untreated

14 0.02 51 untreated

15 0.06 84 untreated

16 0.06 86 untreated

17 0.11 98 untreated

18 0.11 115 untreated

19 0.22 131 untreated

20 0.22 124 untreated

21 0.56 144 untreated

22 0.56 158 untreated

23 1.10 160 untreated

>

> summary(Puromycin) #显示主要的描述性统计量

conc rate state

Min. :0.0200 Min. : 47.0 treated :12

1st Qu.:0.0600 1st Qu.: 91.5 untreated:11

Median :0.1100 Median :124.0

Mean :0.3122 Mean :126.8

3rd Qu.:0.5600 3rd Qu.:158.5

Max. :1.1000 Max. :207.0

> pairs(Puromycin,panel=panel.smooth) #对数据框中变量画出成对散点图

> xtabs(~state+conc,data=Puromycin)

conc

state 0.02 0.06 0.11 0.22 0.56 1.1

treated 2 2 2 2 2 2

untreated 2 2 2 2 2 1

4.数据框的下标与子集的提取

数据框的下标与子集提取与矩阵基本相同，如

* **#提取单个元素**

> Puromycin[1,1]

[1] 0.02

* **提取一个子集**

> Puromycin[c(1,3,5),c(1,3)] #提取第1，3，5行的第1，3列的元素

conc state

1 0.02 treated

3 0.06 treated

5 0.11 treated

> Puromycin[c(1,3,5),] #提取第1，3，5行的所有列的元素

conc rate state

1 0.02 76 treated

3 0.06 97 treated

5 0.11 123 treated

> Puromycin[c(1,3,5),c(1,3)] #提取第1，3，5行的第1，3列的元素

conc state

1 0.02 treated

3 0.06 treated

5 0.11 treated

> Puromycin[c(1,3,5),c("conc","state")] #常用使用变量名称来指定列的位置

conc state

1 0.02 treated

3 0.06 treated

5 0.11 treated

* **提取一列（变量的值）：数据框名$变量名**

> Puromycin$conc #等价于Puromycin[,1]

[1] 0.02 0.02 0.06 0.06 0.11 0.11 0.22 0.22 0.56 0.56 1.10 1.10 0.02 0.02

[15] 0.06 0.06 0.11 0.11 0.22 0.22 0.56 0.56 1.10

> Puromycin$state

[1] treated treated treated treated treated treated treated

[8] treated treated treated treated treated untreated untreated

[15] untreated untreated untreated untreated untreated untreated untreated

[22] untreated untreated

Levels: treated untreated

* **提取满足条件的子集**

> subset(Puromycin,state=="treated"&rate>160) #注意恒等于“==”

conc rate state

9 0.56 191 treated

10 0.56 201 treated

11 1.10 207 treated

12 1.10 200 treated

5. 数据框中添加新变量

例如：以在Puromycin中添加变量iconc,其值为1/conc,有如下三种方法：

* 基本方法：赋值符号

> Puromycin$iconc<-1/Puromycin$conc

* 函数with()

> Puromycin$iconc<-with(Puromycin,1/conc)

* 函数transform（），一次性定义多个变量

Puromycin<-transform(Puromycin,iconc=1/conc,sqrtconc=sqrt(conc))

head(Puromycin) #查看对象的前面部分内容

2.4.6列表(list)的建立

列表：是可由不同类型的对象组成的复杂的数据对象形式，它的建立与数据框类似，下标与子集的提取也与数据框没有本质区别。

* 函数list():以给定的对象为元素建立列表

> L1<-list(1:6,matrix(1:4,nrow=2)) #一个向量和一个矩阵组成

> L1

[[1]]

[1] 1 2 3 4 5 6

[[2]]

[,1] [,2]

[1,] 1 3

[2,] 2 4

> L2<-list(x=1:6,y=matrix(1:4),nrow=2) #一个向量X和一个矩阵Y组成

> L2

$x

[1] 1 2 3 4 5 6

$y

[,1]

[1,] 1

[2,] 2

[3,] 3

[4,] 4

$nrow

[1] 2

* **列表名$变量名：取出列表的子集**

> L2$x #等价于L2[[1]]

[1] 1 2 3 4 5 6

> L2$x[2] #等价于 L2[[1]][2] ，取出列表的子集中的元素

[1] 2

> L2$y[4]

[1] 4

2.4.7时间序列(ts)的建立

* 函数ts（）：通过一向量或矩阵创建一个一元或多元的时间序列，可通过选项表明序列特征

> cost=c(1100,2100,3100,4100,5100,6100,1100,2100,3100,4100,5100,6100)

> income=c(2900,3400,5600,6700,7800,8900,1100,2100,3100,4100,5100,6100)

> data=data.frame(cost,income)

>

> ts(cost,start=2001) #等价于ts(cost,start=2001，frequency=1) ，产生一 元的时间序列

Time Series:

Start = 2001

End = 2012

Frequency = 1

[1] 1100 2100 3100 4100 5100 6100 1100 2100 3100 4100 5100 6100

> ts(data,start=2001) #产生多元的时间序列

Time Series:

Start = 2001

End = 2012

Frequency = 1

cost income

2001 1100 2900

2002 2100 3400

2003 3100 5600

2004 4100 6700

2005 5100 7800

2006 6100 8900

2007 1100 1100

2008 2100 2100

2009 3100 3100

2010 4100 4100

2011 5100 5100

2012 6100 6100

> ts(data,start=2001,frequency=2) #一个时间单位内的观测频率frequency：一年两次

Time Series:

Start = c(2001, 1)

End = c(2006, 2)

Frequency = 2

cost income

2001.0 1100 2900

2001.5 2100 3400

2002.0 3100 5600

2002.5 4100 6700

2003.0 5100 7800

2003.5 6100 8900

2004.0 1100 1100

2004.5 2100 2100

2005.0 3100 3100

2005.5 4100 4100

2006.0 5100 5100

2006.5 6100 6100

> ts(data,start=2001,frequency=4)

cost income

2001 Q1 1100 2900

2001 Q2 2100 3400

2001 Q3 3100 5600

2001 Q4 4100 6700

2002 Q1 5100 7800

2002 Q2 6100 8900

2002 Q3 1100 1100

2002 Q4 2100 2100

2003 Q1 3100 3100

2003 Q2 4100 4100

2003 Q3 5100 5100

2003 Q4 6100 6100

> ts(data,start=2001,frequency=12)

cost income

Jan 2001 1100 2900

Feb 2001 2100 3400

Mar 2001 3100 5600

Apr 2001 4100 6700

May 2001 5100 7800

Jun 2001 6100 8900

Jul 2001 1100 1100

Aug 2001 2100 2100

Sep 2001 3100 3100

Oct 2001 4100 4100

Nov 2001 5100 5100

Dec 2001 6100 6100

> ts(data,start=2001,frequency=10) #注意：时间系列输出有详细说明

Time Series:

Start = c(2001, 1)

End = c(2002, 2)

Frequency = 10

cost income

2001.0 1100 2900

2001.1 2100 3400

2001.2 3100 5600

2001.3 4100 6700

2001.4 5100 7800

2001.5 6100 8900

2001.6 1100 1100

2001.7 2100 2100

2001.8 3100 3100

2001.9 4100 4100

2002.0 5100 5100

2002.1 6100 6100

> ts(data,start=c(2001,3),frequency=10) #开始时间为2001年的第3个观测开始

Time Series:

Start = c(2001, 3)

End = c(2002, 4)

Frequency = 10

cost income

2001.2 1100 2900

2001.3 2100 3400

2001.4 3100 5600

2001.5 4100 6700

2001.6 5100 7800

2001.7 6100 8900

2001.8 1100 1100

2001.9 2100 2100

2002.0 3100 3100

2002.1 4100 4100

2002.2 5100 5100

2002.3 6100 6100

>

#再理解以下书上的例子

> ts(1:10,start=1959)

Time Series:

Start = 1959

End = 1968

Frequency = 1

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

> ts(1:47,frequency=12,start=c(1959,2))

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1959 1 2 3 4 5 6 7 8 9 10 11

1960 12 13 14 15 16 17 18 19 20 21 22 23

1961 24 25 26 27 28 29 30 31 32 33 34 35

1962 36 37 38 39 40 41 42 43 44 45 46 47

> ts(1:10,frequency=4,start=c(1959,2))

Qtr1 Qtr2 Qtr3 Qtr4

1959 1 2 3

1960 4 5 6 7

1961 8 9 10

> ts(matrix(rpois(36,5),12,3),start=c(1961,1),frequency=12)

Series 1 Series 2 Series 3

Jan 1961 6 5 3

Feb 1961 5 6 2

Mar 1961 7 5 4

Apr 1961 3 3 8

May 1961 8 3 5

Jun 1961 2 3 5

Jul 1961 3 4 5

Aug 1961 6 8 7

Sep 1961 6 7 6

Oct 1961 7 5 9

Nov 1961 7 9 5

Dec 1961 4 4 4

本节学习的R函数小结

本表列出了本节讨论的所需掌握的重点函数

|  |  |  |
| --- | --- | --- |
| 表2.2 第2章2.4R的数据结构所介绍的R函数 | | |
| 函数 | 功能 | 示例 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**作业：练习本节所有命令**