Data Analysis Using R: Chapter03

罗智超 *(ROKIA.ORG)*

Contents

通过本章你将学会 . . . 2
R 的基本数据类型 . . . 2
标量 scalar . . . 2
向量 vector . . . 2
使用 seq()、rep() 创建向量 . . . 3
增加或删除向量元素 . . . 4
获得向量长度 . . . 4
向量索引 . . . 4
all() 及 any() 的使用 . . . 5
扩展案例 . . . 5
向量运算 . . . 5
向量过滤 (取子集) . . . 6
向量位置选择 . . . 6
使用 ifelse() 函数 . . . 7
向量元素比较 . . . 7
向量元素命名 . . . 7
扩展案例 . . . 7
扩展案例 . . . 8
矩阵 matrix . . . 8
矩阵的存储 . . . 8
矩阵运算 . . . 9
矩阵索引 . . . 9
自学题：检索，这些 apply 各自什么意思？ . . . 9
练习 . . . 9
数组 array . . . 9
数据框 dataframe . . . 9
数据框元素提取 . . . 10
缺失值处理 complete.cases() . . . 11
合并数据集 . . . 11
累加数据集 . . . 11
使用 apply 系列 . . . 11

扩展案例 11
列表 list 12
List 取子集 12
在 list 上应用函数 12
list 合并成 data.frame 12
因子 factor 13
数据类型属性 13

通过本章你将学会

- R 的基本数据类型 (向量、矩阵、数组、数据框、列表、因子)
- R 的基本数据类型的创建
- R 的基本数据类型之间的转换

R 的基本数据类型

Dim	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

- R 的数据类型的多样性是把双刃剑，由于多样所以灵活，由于灵活，所以掌握难度较大
- 掌握好向量的基本功是掌握其他数据类型的基础，数据框 (dataframe) 是最常用的一种类型

标量 scalar

- 只包含一个元素的向量，用于保存常量

```
a<-5
b<-" 厦门大学"
c<-TRUE
```

- NA 与 NULL 的区别

– 在 R 中 NA 表示为缺失值，NULL 表示为不存在的值,NULL 是特殊的对象，它没有模式 mode
– NULL 的一个用法是在循环中创建向量，其中每次迭代都在这个向量上增加一个元素

向量 vector

- 用于存储数值、字符或者逻辑数据的一维**数组**
- 向量的创建和索引是非常重要的基本功

- 正是 R 的向量运算功能使其效率极高
- 向量有两种形式：原子向量 (Atomic Vector 所有元素类型都一样) 和列表 (List)，有三个共同属性 typeof,length,attributes, 原子向量的四个类型 logical,integer,double,character，可以使用 unlist() 把 list 转成原子向量

```
x<-c(88,NA,12,178,13)
mean(x)
mean(x,na.rm=T)
mean(x,na.rm=T,trim = 0.1)
length(x)
x<-rnorm(1000)
y<-runif(1000)
d1<-density(x)
d2<-density(y)
hist(x)
lines(d2)
x<-c(88,NULL,12,178,13)
mean(x)
length(x)
```

```
# a numeric vector
a <- c(1, 2, 5, 3, 6, -2, 4)
a.i<-c(1L,2L)
# a character vector
b <- c("one", "two", "three")
# a logic vector
c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
attributes(a)
typeof(a)
typeof(a.i)
length(a)
is.atomic(a)
is.list(a)

is.numeric()
is.character()
is.logical()
is.integer()
is.double()
```

使用 seq()、rep() 创建向量

```
x<-seq(from=12,to=30,by=3)
for (i in 1:length(x)){print(x[i])}

y<-seq(rnorm(5))

# 重复常数
x <- rep(8,4)
rep(c(5,12,13),3)
rep(1:3,2)
```

```r
rep(c(5,12,13),each=2)

paste("x",1:5,sep="")
```

增加或删除向量元素

```r
x <- c(88,5,12,13)
x <- c(x[1:3],168,x[4])
x <- c(x[-c(1,2)])

# 练习：
# 选择第 1,3,5,7 个元素
# 向量一阶差分
# 向量二阶差分 =y(x+2) - 2y(x+1) + y(x)
x <- c(88,5,12,13,20,11)
y1<-x[-1]-x[-6]
```

获得向量长度

```r
x <- c(1,2,4)
# 遍历向量
#seq_along(x) = 1:length(x)
n<-length(x)
for (i in seq_along(x)){
    print(x[i] )
}
```

向量索引

```r
y <- c(1.2,3.9,0.4,0.12)
y[c(1,3)]
y[2:3]
v <- 3:4
y[v]
# 向量索引的原理
# 允许重复向量位置
x <- c(4,2,17,5)
y <- x[c(1,1,3)]

x<-1:10
x[3:4]
x[c(3,5,7)]
x[x>8]
x[which(x>8)]
#
x[order(x)]
x[c(FALSE,TRUE)]
y<-setNames(x,letters[1:10])
```

```r
y[c("a","b")]


y[-1]-y[-length(y)]

x<-sample(1:100,20,replace = T)
#question:
#(1) 提取 x 中符合>=21 <=55 条件的数
#(2) 列出大于 50 的数据的位置
#(3) 将 (2) 中计算的结果提取出来
x[x>=21 & x<=55]
which(x>50)
x[which(x>50)]
```

all() 及 any() 的使用

```r
x <- 1:10
any(x > 8)
all(x > 88)
all(x > 0)
```

扩展案例

```r
#Suppose that we are interested in finding runs of consecutive 1s
#in vectorsthat consist just of 1s and 0s.
findruns1 <- function(x,k) {
   n <- length(x)
   runs <- vector(length=n)
   count <- 0
   for (i in 1:(n-k+1)) {
      if (all(x[i:(i+k-1)]==1)) {
         count <- count + 1
         runs[count] <- i
      }
   }
   if (count > 0) {
      runs <- runs[1:count]
   } else runs <- NULL
   return(runs)
}
y<- c(1,0,0,1,1,1,0,1,1)

findruns1(y,2)
```

向量运算

```r
u<-c(5,2,8)
v<-c(1,3,9)
```

```
u>v

z <- c(5,2,-3,8)

w <- z[z*z > 8]

x <- c(1,3,8,2,20)

# 赋值
x[x > 3] <- 0

x<-sample(c(1:100,rep(NA,20)),50,replace = T)
# 将 NA 替换为 x 的均值
x[is.na(x)] <- mean(x,na.rm = T)
```

向量过滤（取子集）

```
#subset(dataset,subset,select=c())
x <- c(6,1:3,NA,12)
x[x > 5]
y<-subset(x,x > 5)
```

向量位置选择

```
z <- c(5,2,-3,8)
which(z*z > 8)
```

```
# 寻找向量中第一个等于 1 的位置
x<-c(3,2,6,1,7,1,1)
# 向量运算
which(x==1)[1]

# 向量计算中短的向量会自动循环补充
c(1,2)+1:10


first1 <- function(x) {
    for (i in 1:length(x))
        {if (x[i] == 1) break # break out of loop
         }
    return(i)
    }
# 另外一种方法
first1a <- function(x) return(which(x == 1)[1])
```

```
x<-sample(1:100,50,replace = T)
# 请将奇数赋值为 1，偶数赋值为 0
x %% 2 == 0
y<-vector(length=length(x))
for (i in 1:length(x)){
```

```
  if (x[i]%%2==0){y[i]<-0}
else {y[i]<-1}
}
```

使用 ifelse() 函数

```
x<-sample(1:100,50,replace = T)

y <- ifelse(x %% 2 == 0,0,1) # %% is the mod operator
```

向量元素比较

```
# Compare whether two datasets are same and
# which array indics is different.
a1<-c(1,3,4,5,6)
a2<-c(1,3,7,8,7)
which(a1!=a2,arr.ind = TRUE)

#identical 比较的是完全一样
identical(x,y)
# : 产生的是整数, c() 产生的是浮点数
x<-1:2
y<-c(1,2)
identical(x,y)

#match(a,b) 类似于 excel 里面的 vlookup, 在 b 中查找是否存在 a 的元素
```

向量元素命名

```
x <- c(1,2,4)
names(x) <- c("a","b","ab")
x["b"]
letters[3:8]
LETTERS[5:6]
```

扩展案例

```
#Kendall's 相关计算
# 方法一
x <- sample(1:50,10,replace = T)
y <- sample(1:50,10,replace = T)
x1<-x[-1]-x[-length(x)]# diff(x)
x2<-ifelse(x1>0,1,0)#sign(x)
y1<-y[-1]-y[-length(y)]
y2 <- ifelse(y1>0,1,0)
c <- ifelse(x2==y2,1,0)
```

```r
mean(c)

# 方法二
findud<-function(v){
  vud<-v[-1]-v[-length(v)]
  return(ifelse(vud>0,1,-1))
}
udcorr<-function(x,y)
  ud<-lapply(list(x,y),findud)
return(mean(ud[[1]]==ud[[2]]))

# 方法三
udcorr<-function(x,y) mean(sign(diff(x))==sign(diff(y)))
```

扩展案例

- 对鲍鱼数据重新编码

```r
#ifelse 可以嵌套使用
#for() 循环可以对字符串向量进行循环，甚至文件名
g<-c("M","F","F","I","M","M","F")
ifelse(g=="M",1,ifelse(g=="F",2,3))
m<-which(g=="M")
f<-which(g=="M")
i<-which(g=="M")
grps<-list()
for(gen in c("M","F","I")) grps[[gen]]<-which(g==gen)
```

矩阵 matrix

- 矩阵是二维**数组**，每个元素具有相同模式 (mode)
- 通过 matrix() 函数创建

```r
# create a 2 x 2 matrix with labels
# fill in the matrix by rows
cells <- c(1,26,24,68)
rnames <- c("R1", "R2")
cnames <- c("C1", "C2")
mymatrix <- matrix(cells, nrow=2, ncol=2,
                   byrow=TRUE,dimnames=list(rnames, cnames))
mymatrix
```

矩阵的存储

- 默认按列存储，即先存第一列，然后依次。

```r
m1 <- matrix(c(1,2,3,4,5,6),nrow=2)
m2 <- matrix(c(1,2,3,4,5,6),nrow=2,byrow=T)
```

矩阵运算

```
y<- matrix(c(1:4),nrow=2)
# mathematical matrix multiplication
y %*% y
# mathematical multiplication of matrix by scalar
3*y
# mathematical matrix addition
y+y
```

矩阵索引

自学题：检索，这些 apply 各自什么意思？

- y[行, 列]
- apply(m,dimcode,f,fargs)
- tapply()
- lapply()
- sapply()
- vapply()

练习

- 计算 airquality 各行、列的均值

数组 array

-数组与矩阵类似，但维度可以大于 2

- 由 array(vetcor,dimensions,dimnames) 创建

```
dim1 <- c("A1", "A2")
dim2 <- c("B1", "B2", "B3")
dim3 <- c("C1", "C2", "C3", "C4")
z <- array(1:24, c(2,3,4), dimnames=list(dim1,dim2,dim3))
z
```

数据框 dataframe

- 数据框是最常用的数据类型，类似于 SAS 里面的 dataset
- 数据框是特殊的 List, 是包含向量的 list
- 不同的列可以包含不同的模式 (数值、字符、逻辑、因子)
- 由 data.frame(col1,col2,col3,...) 创建

```r
patientID <- c(1, 2, 3, 4)
age <- c(25, 34, 28, 52)
diabetes <- c("Type1", "Type2", "Type1", "Type1")
status <- c("Poor", "Improved", "Excellent", "Poor")
patientdata <- data.frame(patientID, age, diabetes,                status,stringsAsFactors=FALSE)
patientdata
```

```r
df<-as.data.frame(replicate(10,sample(1:100,50,replace = T)))
# 问题：请给 df 的 10 个变量，分别命名为 x1-x10
names(df)<-paste0("x",1:10)
df[1:2,1:2]

# 提取 df
# 行 1,3,5,7,9
# 列 2,4,6,8,10

# 提取 x3>=40 或者 x4 <=60 的数据，只显示 x3,x4 的数据
df2<-df[df$x3>=40 | df$x4<=60,c("x3","x4")]
# 请问，以上提取的数据共有多少行
nrow(df2)
ncol(df2)
# 请问，x5 这列中有多少唯一的数字
length(unique(df$x5))
# 请问，y=x7+x8
y <- df$x7+df$x8
z<-cbind(df,y)
z[,]
subset()
```

数据框元素提取

- 从数据框中提取列有两种方法；

```r
df <- data.frame(x = 1:3, y = 3:1, z = letters[1:3],stringsAsFactors = F)
str(df)

#-- 像 list
df[c("x","y")]
#-- 像 matrix
df[,c("x","y")]
# 如果你选择单个列，那么有重要的区别：
# 默认情况下，矩阵的 ** 取子集操作 ** 会对结果进行简化
# 而列表方式却不会。
# 比较下面语句
str(df["x"])
str(df[,"x"])
str(df[,"x",drop=F])
str(df[,c("x","y")])
```

```r
df[2:5,]
df[2:5,2]
df[2:5,2,drop=FALSE]
```

```r
df[df$x >= 1,]
subset(df,x1 >= 1)
#Homework "Data Manipulation with R"
#C01 & C06
```

缺失值处理 complete.cases()

```r
library(mice)
#mice :Multivariate Imputation by Chained Equations

d5 <- d4[complete.cases(d4),]
# na.rm=TRUE in function
```

合并数据集

```r
merge(d1,d2,by.x="kids",by.y="pals")
#cbind
#rbind
```

累加数据集

使用 apply 系列

- lapply 和 sapply 也可以用在 apply 上

library(data.table)

- 来源于 data.frame，但是运算速度更快

```r
library(data.table)
df<- data.table()
```

扩展案例

```r
apply()
mean(anscombe$x1)
apply(anscombe,2,mean)
lapply(datafame,fun)

df(y,x1,x2,x3)


# 应用 Logistic 模型
aba <- read.csv("data/abalone.data",header=T,stringsAsFactors = T)
abamf <- aba[aba$Sex != "I",]  # exclude infants from the analysis
str(abamf)
```

```
levels(abamf$Sex)
lftn <- function(clmn) {
   glm(abamf$Sex ~ clmn, family=binomial)
}

loall <- sapply(abamf[,-1],lftn)
```

列表 list

- list 是最复杂的数据类型
- list 可以包含之前提到的所有数据类型及 list 自己
- 由 mylist <- list(object1, object2, ...) 创建

```
g <- "My First List"
h <- c(25, 26, 18, 39)
j <- matrix(1:10, nrow=5)
k <- c("one", "two", "three")
mylist <- list(title=g, ages=h, j, k)
x<-(mylist[[2]])
(x)
#Sparse Matrix
#Return Parameters
```

List 取子集

- “如果列表 x 是一列载有对象的火车的话, 那么 x[[5]] 就是在第 5 节车厢里的对象; 而 x[4:6] 就是第 4-6 节车厢。” ——@RLangTip

```
x <- list(a = 1:5, b = 2:6,c=letters[1:5])
x[[1]]
x[["a"]]
x$a
x[1:2]

lst$c
lst[["c"]]
lst[[i]] #where i is the index of c within lst
```

在 list 上应用函数

- lapply() 返回结果也是 list
- sapply() 返回结果是向量或者矩阵

list 合并成 data.frame

```r
#create sample list
l <- replicate(5,list(sample(letters, 20)),
  simplify = FALSE
)
# 方法一：
df1 <- data.frame(matrix(unlist(l), nrow=20, byrow=F))

# 方法二：
df2<-do.call(cbind.data.frame, l)

# 方法三：
df3 <- do.call(cbind, lapply(l, as.data.frame))

df3.1 <- do.call(cbind,l)

# 方法四
df4 <- as.data.frame(l)
```

因子 factor

```r
x<-factor("a","b","a","a","b")
class(x)
levels(x)

#
z <- read.csv(text = "value\n12\n1\n.\n9")
str(z)
z <- read.csv(text = "value\n12\n1\n.\n9",na.strings = ".",stringsAsFactors = F)
str(z)
```

数据类型属性

- name()
- dimension()
- class()