Data Analysis Using R: Chapter03

罗智超 (ROKIA.ORG)　　1814347@qq.com

1　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

2　R 的基本数据类型

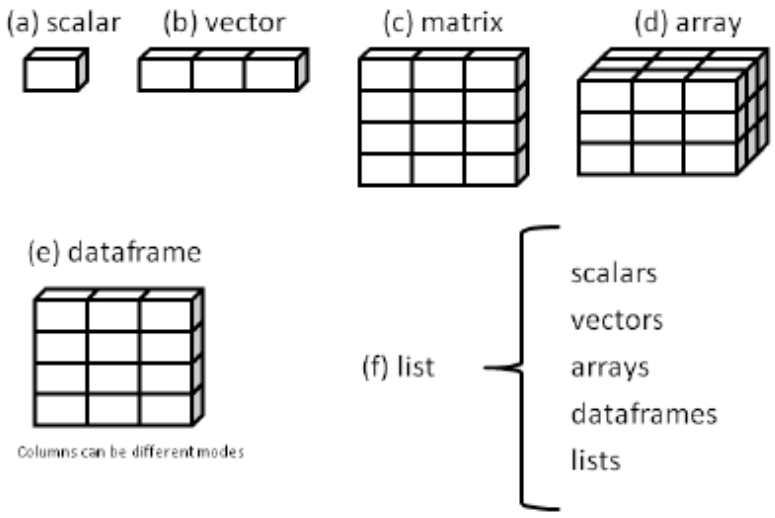

图 1: R 的基本数据类型

- 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
- 掌握好各种数据类型的特性是数据准备的基础，也是学习使用高级包 dplyr、reshape2 的基础

3 标量 scalar

- 只包含一个元素的向量，用于保存常量

```
a<-5
b<-"hello"
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

```
x<-c(88,NA,12,178,13)
mean(x)
mean(x,na.rm=T)
length(x)

x<-c(88,NULL,12,178,13)
mean(x)
length(x)
```

– 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

4 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

```
# a numeric vector
a <- c(1, 2, 5, 3, 6, -2, 4)
# a character vector
b <- c("one", "two", "three")
# a logic vector
c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
attributes(c)
```

5 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])}

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

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

6 增加或删除向量元素

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

7 获得向量长度

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

8 向量索引

```
y <- c(1.2,3.9,0.4,0.12)
y[c(1,3)] # extract elements 1 and 3 of y
y[2:3]
v <- 3:4
y[v]

#Note that duplicates are allowed
x <- c(4,2,17,5)
y <- x[c(1,1,3)]

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

9 all() 及 any() 的使用

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

10 扩展案例

```
#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)
```

11 向量运算

```
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)
x[x > 3] <- 0
x
```

12 向量过滤

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

13 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
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])
```

14 使用 ifelse() 函数

```
x <- 1:10
y <- ifelse(x %% 2 == 0,5,12) # %% is the mod operator
```

15 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)
```

- R 里面有一组 set operation，在做向量比较时比较有用。比如，我们要找出两列向量中相同的部分，我们就可以用 intersect(d1,d2)，如果我们要找出两个向量之间不同的部分，可以使用 setdiff(d1,d2)，当然我们也可以将两个向量进行合并，union(d1,d2)
- 如果要判断某个向量 1:10 的元素是否在向量 c(2,3,11) 中可以使用 1:10 %in% c(2,3,11) 来进行判断

- 还有一个很有用的函数 match(v1,v2)，可以计算出 v1 的元素在 v2 中第一次出现的位置，如果没有则显示 NA
- identical(v1,v2) 用于比较两个向量对应位置上的各个元素是否完全一样，注意 1:3 产生的是整数，而 c(1,2,3) 产生的是浮点数，如果比较这两个向量，identical(1:3,c(1,2,3)) 的结果是 FALSE
- unique() 提取唯一值

```
x <- c(3:5, 11:8, 8 + 0:5)
(ux <- unique(x))
(u2 <- unique(x, fromLast = TRUE))
length(unique(sample(100, 100, replace = TRUE)))
unique(iris)
nrow(iris)
```

- duplicated() 提取重复值

```
x <- c(9:20, 1:5, 3:7, 0:8)
## extract unique elements
(xu <- x[!duplicated(x)])
## similar, same elements but different order:
```

16 向量元素命名

```
x <- c(1,2,4)
names(x) <- c("a","b","ab")
x["b"]
```

17 扩展案例

```
#Kendall's 相关计算
# 方法一
findud<-function(v){
  vud<-v[-1]-v[-length(v)]
  return(ifelse(vud>0,1,-1))
}
udcorr<-function(x,y)
  ud<-laply(list(x,y),findud)
return(mean(ud[[1]]==ud[[2]]))

# 方法二
udcorr<-function(x,y) mean(sign(diff(x))==sign(diff(y)))
```

18 扩展案例

- 对鲍鱼数据重新编码

```
#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)
```

19 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

```
# 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
```

20 矩阵的存储

- 默认按列存储，即先存第一列，然后依次。

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

21 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
```

22 矩阵索引

- y[行, 列]
- apply(m,dimcode,f,fargs)

- tapply()
- lapply()
- sapply()

23 练习

- 计算 airquality 各行、列的均值

24 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
```

25 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
- 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

```
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
```

- 数据框元素索引

```
patientdata[[1]]
patientdata[,1]

patientdata$patientID

attach(patientdata)
patientID
detach(patientdata)
```

26 数据框元素提取

```
examsquiz[2:5,]
examsquiz[2:5,2]
examsquiz[2:5,2,drop=FALSE]
examsquiz[examsquiz$Exam.1 >= 3.8,]
subset(examsquiz,Exam.1 >= 3.8)
```

27 缺失值处理 complete.cases()

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

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

28 合并数据集

```
merge(d1,d2,by.x="kids",by.y="pals")
#cbind
#rbind
# 查看帮助文档
```

29 累加数据集

30 使用 apply 系列

- lapply 和 sapply 也可以用在 apply 上

31 library(data.table)

- 来源于 data.frame，但是运算速度更快

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

32 扩展案例

```
# 应用 Logistic 模型
aba <- read.csv("data/abalone.data",header=T)
abamf <- aba[aba$Gender != "I",]  # exclude infants from the analysis
lftn <- function(clmn) {
   glm(abamf$Gender ~ clmn, family=binomial)$coef
}
loall <- sapply(abamf[,-1],lftn)
```

33 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
```

34 List 索引

```
lst$c
lst[["c"]]
lst[[i]] #where i is the index of c within lst
# 使用单括号 [] 返回的是一个新的列表
```

35 在 list 上应用函数

- lapply() 返回结果也是 list

```
lapply(list(1:3,25:29),mean)
sapply(list(1:3,25:29),mean)
```

- sapply() 返回结果是向量或者矩阵

36 将 List 元素合并成 data.frame

```
#case data: UFO
city.state<-lapply(ufo$location,get.location)
head(city.state)
#[[1]]
#[1]"Iova City" "IA"
#[[2]]
#[1]"Milwaukee" "WI"

location.matrix<-do.call(rbind.city.state)
```

37 因子 factor

- 因子与水平

```
x<-c(5,12,13,12)
xf<-factor(x)
xf
str(xf)
```

- 因子常用函数

- tapply() 将向量分割为组，针对每组应用指定函数

```
ages<-c(25,26,55,37,21,42)
affils<-c("R","D","D","R","U","D")
tapply(ages,affils,mean)
```

-split() 将向量分割为组

```
g<-c("M","F","F","I","M","M","F")
split(1:7,g)
```

-by() 应用的对象不仅是向量, 返回 list

```
aba<-read.csv("data/abalone.data",header=TRUE)
lma<-by(aba,aba$Sex,function(m)lm(m[,2]~m[,3]))
typeof(lma)
```

-aggregate() 对分组中的每个变量调用 tapply() 函数

```
aggregate(aba[,-1],list(aba$Sex),median)
```

38 本周“大牛”

- 卡尔·弗里德里希·高斯（C.F.Gauss，1777 年 4 月 30 日— 1855 年 2 月 23 日），男，德国著名数学家、物理学家、天文学家、大地测量学家。是近代数学奠基者之一，高斯被认为是历史上最重要的数学家之一，并享有“数学王子”之称。高斯和阿基米德、牛顿并列为世界三大数学家。一生成就极为丰硕，以他名字“高斯”命名的成果达 110 个，属数学家中之最。高斯在历史上影响巨大，可以和阿基米德、牛顿、欧拉并列。