第 1 章简介

目录

第一章简介

1.2 R 简介

1.2.1 R 起步

首先安装本书提供的 R 包，该包包括本书用到的数据集和多个函数。

```
# install.packages("DMwR")
# update.packages()
```

1.2.2 R 对象

```
a <- 1
ls()  # 列举当前内存中的对象
```

```
## [1] "a"
```

```
objects()
```

```
## [1] "a"
```

```
rm(a) # 删除对象 a
```

1.2.3 向量

```
# 使用 c() 函数和相应的参数创建向量
v <- c(4, 7, 23.5, 76.2, 80)
v
```

```
## [1]  4.0  7.0 23.5 76.2 80.0
```

```
length(v)
```

```
## [1] 5
```

```
mode(v)
```

```
## [1] "numeric"
```

```
class(v)
```

```
## [1] "numeric"
```

一个向量的所有元素都必须属于同一类型，若不是，R 会强制执行类型转换。

```
v <- c(4, 7, 23.5, 76.2, 80, "rrt")
v
```

```
## [1] "4"    "7"    "23.5" "76.2" "80"   "rrt"
```

1.2.5 因子

因子用水平来表示所有可能的取值，可以用来处理分类（名义）数据。

```
g <- c("f", "m", "m", "m", "f", "m", "f", "m", "f", "f") # 性别向量
g
```

```
##  [1] "f" "m" "m" "m" "f" "m" "f" "m" "f" "f"
```

```
g <- factor(g) # 转换为一个因子
g
```

```
##  [1] f m m m f m f m f f
## Levels: f m
```

假设有另外 5 个人，需要把他们的性别信息存储在另一个因子对象中，假设他们都是男性。如果按照如下设置：

```
other.g <- factor(c("m", "m", "m", "m", "m"))
other.g
```

```
## [1] m m m m m
## Levels: m
```

结果显示因子对象 other.g 只有一个水平，若需要 other.g 与对象 g 一样具有两个相同的因子水平，必须使用如下命令：

```
other.g <- factor(c("m", "m", "m", "m", "m"), levels = c("f", "m"))
other.g
```

```
## [1] m m m m m
## Levels: f m
```

利用因子类型数据，可以计算每个可能值的发生次数，

```
table(g)
```

```
## g
## f m
## 5 5
```

table() 函数也可以用于获取多个因子的交叉表。假设因子 a 存储 10 个人的年龄，那么可以得到这两个向量的交叉表：

```
a <- factor(c("adult", "adult", "juvenile", "juvenile", "adult","adult", "adult", "juve
table(a, g)
```

```
##           g
```

```
## a          f m
##   adult    4 2
##   juvenile 1 3
```

计算列联表的边际和相对频率：

```
t <- table(a, g)
margin.table(t, 1) # 1 表示行, 2 表示列
```

```
## a
##    adult juvenile
##        6        4
```

```
margin.table(t, 2)
```

```
## g
## f m
## 5 5
```

```
prop.table(t, 1)
```

```
##           g
## a                   f         m
##   adult    0.6666667 0.3333333
##   juvenile 0.2500000 0.7500000
```

```
prop.table(t, 2)* 100
```

```
##           g
## a           f  m
##   adult    80 40
##   juvenile 20 60
```

1.2.6 生成序列

```
x <- 1:10
x
```

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

```
10:15 - 1  # ":" 的优先级高于减法
```

```
## [1]  9 10 11 12 13 14
```

```
10:(15-1)
```

```
## [1] 10 11 12 13 14
```

```
5:0
```

```
## [1] 5 4 3 2 1 0
```

```
seq(-4, 1, 0.5)
```

```
##  [1] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0
```

```
seq(1, 5, length = 4)
```

```
## [1] 1.000000 2.333333 3.666667 5.000000
```

```
seq(length = 10, from = -2, by = 0.2)
```

```
##  [1] -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2
```

```
rep(5,4)
```

```
## [1] 5 5 5 5
```

```
rep("hi", 3)
```

```
## [1] "hi" "hi" "hi"
```

```
rep(1:2, 3)
```

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

```
rep(1:2, each = 3)
```

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

gl() 函数可用于生成带有因子的序列，gl(k, n)，其中 k 是因子个数，n 是每

个水平的重复数。

```
gl(3,5)
```

```
##  [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
## Levels: 1 2 3
```

```
gl(2,5, labels = c("female", "male"))
```

```
##  [1] female female female female female male   male   male   male   male
## Levels: female male
```

许多可以根据不同概率密度函数来生成随机序列的函数：

```
rnorm(10) #10 个服从均值为 0, 标准差为 1 正太分布的随机数值
```

```
##  [1]  1.1146191 -2.0272787  0.7048391  0.3730833 -0.6105335  0.8769237
##  [7] -1.0635239  2.1229553  0.3062213 -0.6200478
```

```
rnorm(4, mean = 10, sd = 3)
```

```
## [1] 13.12245 10.71744 14.94453 10.62214
```

```
rt(5, df = 10)
```

```
## [1] -3.4894221  0.1314390 -0.3604728 -0.3571094  2.6639701
```

1.2.7 数据子集

```
x <- c(0, -3, 4, -1, 45, 90, -5)
x[x > 0]
```

```
## [1]  4 45 90
```

```
x[ x <= -2 | x > 5]  # 逻辑或
```

```
## [1] -3 45 90 -5
```

```
x[ x > 40 & x < 100] # 逻辑与
```

```
## [1] 45 90
```

```
x[c(4, 6)]
```

```
## [1] -1 90
```

```
x[1:3]
```

```
## [1]  0 -3  4
```

```
x[-1] # 表示删除第一个元素
```

```
## [1] -3  4 -1 45 90 -5
```

```
x[-(1:3)]
```

```
## [1] -1 45 90 -5
```

```
x[- c(4,6)]
```

```
## [1]  0 -3  4 45 -5
```

可通过 R 函数 names() 给向量中元素命名

```
pH <- c(4.5, 7, 7.3, 8.2, 6.3)
names(pH) <- c("area1", "area2", "mud", "dam", "middle")
pH
```

```
##  area1  area2    mud    dam middle
##    4.5    7.0    7.3    8.2    6.3
```

```
pH <- c(area1 = 4.5, area2 = 7, mud = 7.3, dam = 8.2, middle = 6.3)
pH
```

```
##  area1  area2    mud    dam middle
##    4.5    7.0    7.3    8.2    6.3
```

```
pH["mud"]
```

```
## mud
## 7.3
```

```
pH[c("area1", "area2")]
```

```
## area1 area2
##   4.5   7.0
```

1.2.8 矩阵和数组

```
m <- matrix(c(45,23,66,77,33,44,56,12,78,23), 2,5)
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   45   66   33   56   78
## [2,]   23   77   44   12   23
```

注意到，向量中的数据通过矩阵中的列进行扩展。首先将数据填到第一列，然后填到第二列，以此类推。也可以设定函数 matrix() 的参数进行按行填充：

```
m <- matrix(c(45,23,66,77,33,44,56,12,78,23), 2,5, byrow = T)
m
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   45   23   66   77   33
## [2,]   44   56   12   78   23
```

```
m[2, 3]
```

```
## [1] 12
```

```
m[-2, 1]
```

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

```
m[1, -c(3, 5)]
```

```
## [1] 45 23 77
```

```
m[1,]
```

```
## [1] 45 23 66 77 33
```

按照上面操作，得到的结果可能是一个向量，可以使用下面命令使结果仍为一个矩阵

```
class(m[1, ])
```

```
## [1] "numeric"
```

```
m[1, , drop = F]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   45   23   66   77   33
```

```
class(m[1, , drop = F])
```

```
## [1] "matrix" "array"
```

可以使用函数 cbind() 和 rbind() 分别按列和行把两个或两个以上的向量合并。

```
m1 <- c(1,2)
m2 <- c(3,4)
cbind(m1, m2)
```

```
##      m1 m2
## [1,]  1  3
## [2,]  2  4
```

```
rbind(m1, m2)
```

```
##    [,1] [,2]
## m1    1    2
## m2    3    4
```

可以使用 colnames() 和 rownames() 给矩阵的行和列命名：

```
colnames(m) <- c("c1", "c2", "c3", "c4", "c5")
rownames(m) <- c("r1", "r2")
m
```

```
##    c1 c2 c3 c4 c5
## r1 45 23 66 77 33
## r2 44 56 12 78 23
```

数组是矩阵的扩展，它把数据维度扩展到两个以上。这意味数组中的元素需要两个以上的索引。可以使用函数 array() 创建数组。

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

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

```
a[1,3,2]
```

```
## [1] 21
```

```
a[1, , 2]
```

```
## [1] 13 17 21
```

```
a[c(2, 3), , -2]
```

```
##      [,1] [,2] [,3]
## [1,]    2    6   10
## [2,]    3    7   11
```

1.2.9 列表

列表的成分和向量元素不同，它们不一定是同一种数据类型、模式或者相同长度。

```
my.lst <- list(stud.id = 34453, stud.name = "John", stud.marks = c(14.3, 12, 15, 19))
my.lst$stud.id
```

```
## [1] 34453
```

```
my.lst[[1]]  # 双方括号
```

```
## [1] 34453
```

```
my.lst[1]
```

```
## $stud.id
## [1] 34453
```

```
names(my.lst)
```

```
## [1] "stud.id"    "stud.name"  "stud.marks"
```

```
names(my.lst) <- c("id", "names", "marks")
my.lst
```

```
## $id
## [1] 34453
##
## $names
## [1] "John"
##
```

```
## $marks
## [1] 14.3 12.0 15.0 19.0
```

```
# 添加元素，扩展列表
my.lst$parents.names <- c("Ana", "Mike")
my.lst
```

```
## $id
## [1] 34453
##
## $names
## [1] "John"
##
## $marks
## [1] 14.3 12.0 15.0 19.0
##
## $parents.names
## [1] "Ana"  "Mike"
```

```
my.lst <- my.lst[-1] # 剔除列表成分
my.lst
```

```
## $names
## [1] "John"
##
## $marks
## [1] 14.3 12.0 15.0 19.0
##
## $parents.names
## [1] "Ana"  "Mike"
```

通过函数 c() 合并列表

```
other <- list(age = 19, sex = "male")
lst <- c(my.lst, other)
lst
```

```
## $names
## [1] "John"
##
## $marks
## [1] 14.3 12.0 15.0 19.0
##
## $parents.names
## [1] "Ana"  "Mike"
##
## $age
## [1] 19
##
## $sex
## [1] "male"
```

可以通过函数 unlist() 把列表中所有元素转换为向量元素，转换后的向量元素个数和列表中所有数据对象个数相同，并且会把列表中不同类型数据统一，觉得多数情况下会转换为字符型。

```
unlist(my.lst)
```

```
##          names         marks1         marks2         marks3         marks4
##         "John"         "14.3"           "12"           "15"           "19"
## parents.names1 parents.names2
##          "Ana"         "Mike"
```

1.2.10 数据框

数据框结构类似于二维矩阵，不同的是，数据框每列可以有不同的数据类型的数据，在这个意义上，数据框和列表相似。对 R 而言，数据框是一类特殊的列表。

```
my.dataset <- data.frame(site = c("A", "B", "A", "A", "B"), season = c("Winter", "Summe
my.dataset
```

```
##   site season  pH
## 1    A Winter 7.4
## 2    B Summer 6.3
## 3    A Summer 8.6
## 4    A Spring 7.2
## 5    B   Fall 8.9
```

```
my.dataset[my.dataset$pH > 7, ]
```

```
##   site season  pH
## 1    A Winter 7.4
## 3    A Summer 8.6
## 4    A Spring 7.2
## 5    B   Fall 8.9
```

可以使用函数 attach() 简化查询。attach() 函数可以直接访问数据框的列，无需添加相应的数据框名：

```
attach(my.dataset)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     pH
```

```
my.dataset[pH > 7, ]
```

```
##   site season  pH
## 3    A Summer 8.6
## 4    A Spring 7.2
```

```
season
```

```
## [1] "Winter" "Summer" "Summer" "Spring" "Fall"
```

函数 attach() 的反向操作是函数 detach()，它禁止直接访问数据框的列。

```
detach(my.dataset)
# season
```

如果要该表数据框中数据的值：

```
my.dataset[my.dataset$season == "Summer", "pH"] <- my.dataset[my.dataset$season == "Sum
my.dataset
```

```
##   site season  pH
## 1    A Winter 7.4
## 2    B Summer 7.3
## 3    A Summer 9.6
## 4    A Spring 7.2
## 5    B   Fall 8.9
```

加入新列：

```
my.dataset$NO3 <- c(2,3,4,5,6)
my.dataset
```

```
##   site season  pH NO3
## 1    A Winter 7.4   2
## 2    B Summer 7.3   3
## 3    A Summer 9.6   4
## 4    A Spring 7.2   5
## 5    B   Fall 8.9   6
```

```
nrow(my.dataset)
```

```
## [1] 5
```

```
ncol(my.dataset)
```

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

可以使用函数 data() 获取 R 内置数据集的信息，使用 data(USArrests) 获取数据集 USArrests。

1.2.11 构建新函数

在创建新函数前，需要检查 R 中是否已经有一个同名的函数存在。如果的确有同名的函数存在，那么可以选择一个不同的名称；否则，会把 R 中已经存在的 R 函数对用户隐藏起来。此时 R 标准函数仍然存在，但是搜索路径中与它具有相同名称的用户定义函数位于顶部，因此“隐藏”了 R 的标准函数。

检查某个函数是否在 R 中存在，在命令提示符处输入函数名即可：

```
# se
```

创建函数的基本形式：function() { }

函数的返回值可以由函数 return() 确定，或者 R 返回函数中最后运算表达式的结果。

1.2.13 管理 R 会话

假设当前工作目录中有一个名称为“mycode.R”的文本文件，可以使用如下命令执行文件

```
# source('mycode.R')
```

有时可能需要保存对象以备以后应用，比如在名称为 mysession.RData 的文件中保存了名为 f 和 my.dataset 的两个对象：

```
# save(f, my.dataset, file = "mysession.RData")
```

在新的 R 会话中，可以使用如下命令重新读入这些对象：

```
# load("mysession.RData")
```

也可以使用下面命令来保存当前 R 工作空间中所有对象：

```
save.image()
```

上面命令会在当前工作目录中把 R 工作空间保存为“.RData”的文件。从该目录启动 R 时，该文件会自动载入 R 中。注意：当 R 退出时，如果回答

“是”，会达到上面相同效果。