R 中基本数据类型及操作

周世祥

2020 年 5 月 17 日

R 中的数据类型

电影数据

```
rm(list = ls())
movie = read.csv(" 电影数据.csv", header = T, fileEncoding = "UTF-8")
head(movie)
```

```
##                       name boxoffice doubanscore type duration  showtime
## 1                    叶问3  77060.44         6.4 动作      105  2016/3/4
## 2                   美人鱼 338583.26         6.9 喜剧       93  2016/2/8
## 3           女汉子真爱公式   6184.45         4.5 喜剧       93 2016/3/18
## 4 西游记之孙悟空三打白骨精 119956.51         5.7 喜剧      120  2016/2/8
## 5               澳门风云三 111693.89         4.0 喜剧      112  2016/2/8
## 6               功夫熊猫3  99832.53         7.7 喜剧       95 2016/1/29
##   director      star1 index1        star2 index2
## 1   叶伟信      甄子丹  11385         张晋   4105
## 2   周星驰        邓超  41310         林允   9292
## 3   郭大雷      赵丽颖 181979         张翰  44277
## 4   郑保瑞      郭富城  12227         巩俐   8546
## 5     王晶      周润发  16731       刘德华  30277
## 6   吕寅荣 杰克布莱克     178 安吉丽娜朱莉   1540
```

基本数据类型

1. 数值型

电影数据示例

```
class(movie$"boxoffice");
```

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

```
class(movie$doubanscore)
```

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

```
# 自己为变量赋一个数值
a = 2; class(a)
```

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

```
exp(1000)  # 正无穷
```

```
## [1] Inf
```

```
-10 / 0  # 负无穷
```

```
## [1] -Inf
```

```
exp(1000) / exp(990)  # NaN 类型
```

```
## [1] NaN
```

```
exp(10)
```

```
## [1] 22026.47
```

```
##  [1] 22026.47
```

2. 字符型

字符的定义

```
a = "2"
class(a)
```

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

```
# 判断电影数据集中，变量“type","name" 是不是字符型变量
class(movie$type)
```

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

```
class(movie$name)
```

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

3. 逻辑型数据

读入数据时设置把字符数据保留，不转换为 factor

```
movie = read.csv(" 电影数据.csv", header = T, stringsAsFactors = F, fileEncoding = "UTF-8")
movie$type[movie$name == " 美人鱼"] == " 喜剧"
```

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

```
# 想在数据集中挑选大于 7 分的喜剧电影 name?
movie$name[movie$type == " 喜剧" & movie$"doubanscore" > 7]
```

```
## [1] "功夫熊猫3"
```

```
# 逻辑语句加减
(1 == 2) + (3 < 4)
```

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

4. 因子型数据

(1) 什么是因子型数据

```
(genders = factor(c(" 男", " 女", " 女", " 男", " 男")))
```

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

```
(class = factor(c("Poor", "Improved", "Excellent"), ordered = T))
```

```
## [1] Poor      Improved  Excellent
## Levels: Excellent < Improved < Poor
```

(2) 如何改变因子型数据各水平的编码顺序 ##“‘

```
(class = factor(c("Poor", "Improved", "Excellent"), ordered = T,
              levels = c("Poor", "Improved", "Excellent")))
```

```
## [1] Poor      Improved  Excellent
## Levels: Poor < Improved < Excellent
```

(3) 如何正确将因子型数据和字符型数据互相转化

输入原始字符变量

```
all = c(" 男", " 女", " 女", " 男", " 男")
# 将字符型变量变成因子型
gender = as.factor(all)
# 变换后的数据类型
is.factor(gender)
```

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

```
class(gender)
```

```
## [1] "factor"
```

```
# 将因子型变量变成字符型
genders = as.character(gender)
# 变换后的数据类型
is.character(genders)
```

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

```
class(genders)
```

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

5. 时间类数据

(1) 如何把字符转化成 Date 日期格式

函数 head 用来查看数据前 6 个元素，函数 class 用来查看对象数据类型

```
head(movie$showtime)
```

```
## [1] "2016/3/4"  "2016/2/8"  "2016/3/18" "2016/2/8"  "2016/2/8"  "2016/1/29"
```

```
class(movie$showtime)
```

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

```
movie$showtime = as.Date(movie$showtime)
head(movie$showtime)
```

```
## [1] "2016-03-04" "2016-02-08" "2016-03-18" "2016-02-08" "2016-02-08"
## [6] "2016-01-29"
```

```
class(movie$showtime)
```

```
## [1] "Date"
```

```
Sys.setlocale("LC_TIME", "C")
```

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

```
x = c("1jan1960", "2jan1960", "31mar1960", "30jul1960")
# y = as.Date(x)
(y = as.Date(x, format = "%d%b%Y"))
```

```
## [1] "1960-01-01" "1960-01-02" "1960-03-31" "1960-07-30"
```

(2) 如何把字符转化成 POSIXct/POSIXlt 时间格式

```
as.POSIXct("2015-11-27 01:30:00")
```

```
## [1] "2015-11-27 01:30:00 CST"
```

```
# as.POSIXct("November-27-2015 01:30:00")
as.POSIXct("November-27-2015 01:30:00", format = "%B-%d-%Y %H:%M:%S")
```

```
## [1] "2015-11-27 01:30:00 CST"
```

(3) 如何把时间数据摆弄成你想要的形式

```
(m = head(movie$showtime))  # 原始日期数据
```

```
## [1] "2016-03-04" "2016-02-08" "2016-03-18" "2016-02-08" "2016-02-08"
## [6] "2016-01-29"
```

```
format(m, format = "%B %d %Y")  # 改成月日年的格式
```

```
## [1] "March 04 2016"    "February 08 2016" "March 18 2016"    "February 08 2016"
## [5] "February 08 2016" "January 29 2016"
```

```
format(m, format = "%B %d %Y %A")  # 加入星期信息
```

```
## [1] "March 04 2016 Friday"    "February 08 2016 Monday"
## [3] "March 18 2016 Friday"    "February 08 2016 Monday"
## [5] "February 08 2016 Monday" "January 29 2016 Friday"
```

```
format(m, format = "%B")  # 只提取出月份信息
```

```
## [1] "March"    "February" "March"    "February" "February" "January"
```

```
Sys.time()  # 输出系统时间
```

```
## [1] "2021-05-31 16:12:47 CST"
```

```
class(Sys.time())  # 查看时间类型
```

```
## [1] "POSIXct" "POSIXt"
```

```
format(Sys.time(), format = "%B %d %Y")  # 提取部分时间信息
```

```
## [1] "May 31 2021"
```

```
format(Sys.time(), format = "%Y/%B/%a %H:%M:%S")  # 提取部分时间信息
```

```
## [1] "2021/May/Mon 16:12:47"
```

(4) 一款处理时间数据的专用包 lubridate

```
# install.packages(lubridate)
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.0.5
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
x = c(20090101, "2009-01-02", "2009 01 03", "2009-1-4", "2009-1,5", "Created on 2009 1 6", "200901
ymd(x)
```

```
## [1] "2009-01-01" "2009-01-02" "2009-01-03" "2009-01-04" "2009-01-05"
## [6] "2009-01-06" "2009-01-07"
```

```
mday(as.Date("2015-11-20"))
```

```
## [1] 20
```

```
wday(as.Date("2015-11-20"))
```

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

```
hour(as.POSIXct("2015-11-20 01:30:00"))
```

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

```
minute(as.POSIXct("2015-11-20 01:30:00"))
```

```
## [1] 30
```

(5) 时间类数据的操作

```
# 做差
# 求任意两个日期距离的天数
begin = as.Date("2016-03-04")
end = as.Date("2016-05-08")
(during = end - begin)
```

```
## Time difference of 65 days
```

```
# 求任意两个日期距离的周数和小时数
difftime(end, begin, units = "weeks")
```

```
## Time difference of 9.285714 weeks
```

```
difftime(end, begin, units = "hours")
```

```
## Time difference of 1560 hours
```

排序

单独对时间进行排序

```
head(movie$showtime)
```

```
## [1] "2016-03-04" "2016-02-08" "2016-03-18" "2016-02-08" "2016-02-08"
## [6] "2016-01-29"
```

```
head(sort(movie$showtime))
```

```
## [1] "2016-01-29" "2016-02-08" "2016-02-08" "2016-02-08" "2016-03-04"
## [6] "2016-03-18"
```

对数据表格中的数据按照时间顺序排列, 这里只选取前 6 行，部分列做展示

```
head(movie[order(movie$showtime), c("name", "showtime")])
```

```
##                       name   showtime
## 6                 功夫熊猫3 2016-01-29
## 2                    美人鱼 2016-02-08
## 4 西游记之孙悟空三打白骨精 2016-02-08
## 5                 澳门风云三 2016-02-08
## 1                    叶问3 2016-03-04
## 3            女汉子真爱公式 2016-03-18
```

2.1 R 中的数据类型 rm(list = ls()) movie = read.csv("电影数据.csv", fileEncoding = "UTF-8", stringsAsFactors = F)

2.1.2 向量

一、基本操作

1. 向量的创建

```
c(1, 1, 1, 2, 3, 3, 1, 2, 4, 1, 2, 4, 4, 2, 3, 4, 1, 2, 3, 4)
```

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

```
c("a", "b", "c", "d")
```

```
## [1] "a" "b" "c" "d"
```

```
# seq(起始值, 终止值, 步长)
seq(0, 10, by = 2)
```

```
## [1]  0  2  4  6  8 10
```

```
1:10
```

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

```
# sample(被抽取的数据集合, 抽取数量)
set.seed(1234)
sample(1:10, 5)
```

```
## [1] 10  6  5  4  1
```

```
paste0("x_", 1:5)
```

```
## [1] "x_1" "x_2" "x_3" "x_4" "x_5"
```

2. 向量的引用

```
# 引用 x 向量中的第 5 个元素
x=c(1, 1, 1, 2, 3, 3)
x[5]
```

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

```
# 想看看 x 向量中 3 所在的位置
which(x == 3)
```

```
## [1] 5 6
```

```
which.max(x)
```

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

```
which.min(x)
```

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

3. 集合运算

```
intersect(c(1, 2, 3, 3, 12, 4, 123, 12), c(1, 2, 3))
```

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

```
union(c(" 狗熊会", " 聚数据英才"), c(" 狗熊会", " 助产业振兴"))
```

```
## [1] "狗熊会"     "聚数据英才" "助产业振兴"
```

```
setdiff(10:2, 5:3)
```

```
## [1] 10  9  8  7  6  2
```

二、常见类型

1. 数值向量的花式玩法

```
# match 函数
x = c(1, 1, 1, 2, 3, 3, 1, 2, 4, 1, 2, 4, 4, 2, 3, 4, 1, 2, 3, 4)
(y = letters[x])  # letters 是一个内置字符串，里面储存 26 个字母字符
```

```
##  [1] "a" "a" "a" "b" "c" "c" "a" "b" "d" "a" "b" "d" "d" "b" "c" "d" "a" "b" "c"
## [20] "d"
```

```
match(y, letters[1:4])
```

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

```
# cut 函数
(Age = sample(21:100, 20, replace = T))
```

```
##  [1] 25 58 36 24 90 99 98 34 76 82 24 24 41 60 76 87 25 86 67 60
```

```
# 将年龄数据离散化
label = c('壮年', '中年', '长辈', '老年')
(ages = cut(Age, breaks = c(20, 30, 50, 70, 100), labels = label))
```

```
##  [1] 壮年 长辈 中年 壮年 老年 老年 老年 中年 老年 老年 壮年 壮年 中年 长辈 老年
## [16] 老年 壮年 老年 长辈 长辈
## Levels: 壮年 中年 长辈 老年
```

```
# sort 和 order 函数
set.seed(1234)
(x = sample(8, 5))
```

```
## [1] 4 2 6 5 8
```

```
sort(x)
```

```
## [1] 2 4 5 6 8
```

```
order(x)
```

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

```
x[order(x)]
```

```
## [1] 2 4 5 6 8
```

2. 字符向量的花式玩法

```
# nchar 用来提取字符串的长度
nchar(" 欢迎关注狗熊会")
```

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

```
# 看看数据集中的电影名字的长度分别是多少
nchar(movie$name)
```

```
##  [1]  3  3  7 12  5  5 12  7  8  4  5  7  4  4  6  4  4  3  2
```

```
# 中英文的字符长度计算方法有不同
nchar("Welcome to follow the CluBear")
```

```
## [1] 29
```

```
# substr 提取子字符串
substr(" 欢迎关注狗熊会", 1, 4)
```

```
## [1] "欢迎关注"
```

```
substr(" 一懒众衫小", 3, 5)
```

```
## [1] "众衫小"
```

```
# paste 基本玩法
paste(c(" 双 11", " 是个", " 什么节日"), collapse = "")
```

```
## [1] "双11是个什么节日"
```

```
paste("A", 1:4)
```

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

```
# paste 花式玩法
paste(1:4, collapse = "")
```

```
## [1] "1234"
```

```
paste(1:4, sep="")
```

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

```
paste("A", 1:4, sep="_")
```

```
## [1] "A_1" "A_2" "A_3" "A_4"
```

思考题

自己测试一下：

```
paste(LETTERS[1:4], 1:4, collapse = "_")
```

```
paste(LETTERS[1:4], 1:4, sep = "_", collapse = "|")
```

```
paste(LETTERS[1:4], 1:4)
```

```
txt = c("狗熊会", "CluBear", "双11", "生日")
# 返回含有关键字的字符位置
grep("Bear", txt)
```

```
gsub("生日", "happy birthday", txt)
```

```
# grep返回movie的name中包含“青春”的行号8，movie[8, ]即提取出movie数据集的第8行
(index = grep("青春", movie$name))

(young = movie[index, ])
##              name boxoffice doubanscore type duration  showtime director


# 看看它的豆瓣评分和票房处于我们电影数据集中的什么位置
young$doubanscore > mean(movie$doubanscore)

young$boxoffice > mean(movie$boxoffice)


salary = c("22万", "30万", "50万", "120万", "11万")
(salary0 = gsub("万", "0000", salary))

mean(as.numeric(salary0))

median(as.numeric(salary0))  # 结果是科学计数法的形式
```

2.1 R 中的数据类型 $\mathrm{rm(list = ls())}$

2.1.3 矩阵

1. 矩阵的创建与引用

生成全部是 0 的矩阵

```
(zero = matrix(0, nrow = 3, ncol = 3))
```

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

```r
# 生成一个对角全是 1 的矩阵，直接在 diag 中输入对角线向量即可
(dig = diag(rep(1, 4)))
```

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

```r
# 从已有数据转化成矩阵
(M = matrix(1:12, nrow = 3, ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```r
(N = diag(1:4))
```

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

2. 矩阵的常用操作

(1) 矩阵概览

```r
# 查看矩阵的维度
dim(M)
```

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

```r
# 提取矩阵的行数
nrow(M)
```

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

```r
# 提取矩阵的列数
ncol(M)
```

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

```
# 引用元素
M[1, 2]
```

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

```
M[1:2, 2:3]
```

```
##      [,1] [,2]
## [1,]    4    7
## [2,]    5    8
```

```
# 给行列命名
colnames(M) = paste0("x_", 1:4)
rownames(M) = 1:3; M
```

```
##   x_1 x_2 x_3 x_4
## 1   1   4   7  10
## 2   2   5   8  11
## 3   3   6   9  12
```

```
# 同样的命令可调用行列名
colnames(M)
```

```
## [1] "x_1" "x_2" "x_3" "x_4"
```

```
rownames(M)
```

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

(2) 将多个矩阵合并

```
(A = matrix(1:9, nrow = 3, ncol = 3, byrow = T))
```

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

```
(B = diag(11:13))
```

```
##      [,1] [,2] [,3]
## [1,]   11    0    0
## [2,]    0   12    0
## [3,]    0    0   13
```

```
rbind(A, B)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   11    0    0
## [5,]    0   12    0
## [6,]    0    0   13
```

```
cbind(A, B)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    2    3   11    0    0
## [2,]    4    5    6    0   12    0
## [3,]    7    8    9    0    0   13
```

3. 矩阵的数学操作

```
## (1) 矩阵的加减乘运算 ##
A + B
```

```
##      [,1] [,2] [,3]
## [1,]   12    2    3
## [2,]    4   17    6
## [3,]    7    8   22
```

```
A - B
```

```
##      [,1] [,2] [,3]
## [1,]  -10    2    3
## [2,]    4   -7    6
## [3,]    7    8   -4
```

```
A * B
```

```
##      [,1] [,2] [,3]
## [1,]   11    0    0
## [2,]    0   60    0
## [3,]    0    0  117
```

```
A %*% B
```

```
##      [,1] [,2] [,3]
## [1,]   11   24   39
## [2,]   44   60   78
## [3,]   77   96  117
```

(2) rARPACK 的应用

```
# 打开这个包
# install.packages("rARPACK")
library(rARPACK)
```

```
## Warning: package 'rARPACK' was built under R version 4.0.5
```

```
# 构造一个 1000 维的大型矩阵
T = matrix(1:1000000, 1000, 1000)
# 正常分解与快速分解的对比，此处以选择前 5 个特征（奇异）值为例
system.time(svd(T))
```

```
##    user  system elapsed
##    2.41    0.00    2.41
```

```
system.time(svds(T, 5))
```

```
##    user  system elapsed
##    0.06    0.00    0.06
```

```
system.time(eigen(T))
```

```
##    user  system elapsed
##    4.00    0.01    4.02
```

```
system.time(eigs(T, 5))
```

```
##    user  system elapsed
##    0.19    0.00    0.19
```

矩阵的转置、求逆及分解

```
solve(B)  # 求矩阵逆
```

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

```
## [1,] 0.09090909 0.00000000 0.00000000
## [2,] 0.00000000 0.08333333 0.00000000
## [3,] 0.00000000 0.00000000 0.07692308
```

```
t(A)      # 求矩阵转置
```

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

```
eigen(A)  # 特征值分解
```

```
## eigen() decomposition
## $values
## [1]  1.611684e+01 -1.116844e+00 -1.303678e-15
##
## $vectors
##            [,1]        [,2]       [,3]
## [1,] -0.2319707 -0.78583024  0.4082483
## [2,] -0.5253221 -0.08675134 -0.8164966
## [3,] -0.8186735  0.61232756  0.4082483
```

```
##  eigen() decomposition
##  $values

svd(A)    # 奇异值 svd 分解
```

```
## $d
## [1] 1.684810e+01 1.068370e+00 4.418425e-16
##
## $u
##            [,1]       [,2]       [,3]
## [1,] -0.2148372  0.8872307  0.4082483
## [2,] -0.5205874  0.2496440 -0.8164966
## [3,] -0.8263375 -0.3879428  0.4082483
##
## $v
##            [,1]        [,2]       [,3]
## [1,] -0.4796712 -0.77669099 -0.4082483
## [2,] -0.5723678 -0.07568647  0.8164966
## [3,] -0.6650644  0.62531805 -0.4082483
```

(3) 稀疏矩阵

```
# install.packages("Matrix")
library(Matrix)
# 生成普通矩阵
vector = c(1:3, rep(0, 5), 6:9)
(m1 = matrix(vector, nrow = 3, ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    7
## [2,]    2    0    0    8
## [3,]    3    0    6    9
```

```
# 生成稀疏矩阵方法 1
(m2 = Matrix(vector, nrow = 3 ,ncol = 4, sparse = TRUE))
```

```
## 3 x 4 sparse Matrix of class "dgCMatrix"
##
## [1,] 1 . . 7
## [2,] 2 . . 8
## [3,] 3 . 6 9
```

```
(m3 = Matrix(vector, nrow = 3 ,ncol = 4, sparse = FALSE))
```

```
## 3 x 4 Matrix of class "dgeMatrix"
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    7
## [2,]    2    0    0    8
## [3,]    3    0    6    9
```

```
# 生成稀疏矩阵方法 2
(m4 = spMatrix(10, 20, i = c(1, 3:8), j = c(2, 9, 6:10), x = 7 * (1:7)))
```

```
## 10 x 20 sparse Matrix of class "dgTMatrix"
##
##  [1,] . 7 . . .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
##  [2,] . . . . .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
##  [3,] . . . . .  .  .  . 14  .  .  .  .  .  .  .  .  .  .
##  [4,] . . . . . 21  .  .  .  .  .  .  .  .  .  .  .  .  .
##  [5,] . . . . .  . 28  .  .  .  .  .  .  .  .  .  .  .  .
##  [6,] . . . . .  .  . 35  .  .  .  .  .  .  .  .  .  .  .
##  [7,] . . . . .  .  .  . 42  .  .  .  .  .  .  .  .  .  .
```

```
##  [8,] . . . . . . . . 49 . . . . . . . . . .
##  [9,] . . . . . . . . . . . . . . . . . . . .
## [10,] . . . . . . . . . . . . . . . . . . . .
```

```
summary(m4)
```

```
## 10 x 20 sparse Matrix of class "dgTMatrix", with 7 entries
##    i  j  x
## 1  1  2  7
## 2  3  9 14
## 3  4  6 21
## 4  5  7 28
## 5  6  8 35
## 6  7  9 42
## 7  8 10 49
```

```
# 当行列数分别为 10000 时，稀疏矩阵的内存大小和生成时间优势均很明显。
n = 10000
m1 = matrix(0, nrow = n, ncol = n)
m2 = Matrix(0, nrow = n, ncol = n, sparse = TRUE)
object.size(m1); object.size(m2)
```

```
## 800000216 bytes
```

```
## 41728 bytes
```

```
system.time(matrix(0, nrow = n, ncol = n))
```

```
##    user  system elapsed
##    0.15    0.03    0.19
```

```
system.time(Matrix(0, nrow = n, ncol = n, sparse = TRUE))
```

```
##    user  system elapsed
##       0       0       0
```

```
# 两种矩阵计算区别
n = 1000
dat = sample(c(0, 1), n^2, replace = TRUE, prob = c(0.9, 0.1))
m1 = matrix(dat, nrow = n, ncol = n); m1[1:6, 1:6]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    0    0    0    0    0
## [2,]    0    0    0    0    0    0
```

```
## [3,]    0    1    0    0    0    0
## [4,]    0    1    0    1    0    1
## [5,]    0    0    0    0    0    0
## [6,]    0    0    1    1    0    0
```

```
m2 = Matrix(dat, nrow = n, ncol = n, sparse = TRUE); m2[1:6, 1:6]
```

```
## 6 x 6 sparse Matrix of class "dgCMatrix"
##
## [1,] . . . . . .
## [2,] . . . . . .
## [3,] . 1 . . . .
## [4,] . 1 . 1 . 1
## [5,] . . . . . .
## [6,] . . 1 1 . .
```

```
# 求乘积运算时间对比
system.time(m1 %*% t(m1))
```

```
##    user  system elapsed
##     0.7     0.0     0.7
```

```
system.time(m2 %*% t(m1))
```

```
##    user  system elapsed
##    0.07    0.00    0.06
```

```


rm(list = ls())
```

2.1 R 中的数据类型

2.1.4 数据框

1. 创建数据框

```
# 读入一个 txt,csv 等格式数据，即自成一个数据框
movie = read.csv(" 电影数据.csv", fileEncoding = "UTF-8", stringsAsFactors = F)
class(movie)
```

```
## [1] "data.frame"
```

```
# 自己创建
star1 = c(" 邓超", " 赵丽颖", " 郭富城", " 周润发", " 杰克布莱克", " 汤唯", " 白敬亭", " 陈晓", " 梁家
birthyear = c(1979, 1987, 1965, 1955, 1969, 1979, 1993, 1987, 1958, 1979, 1987, 1980, 1977)
gender = c(" 男", " 女", " 男", " 男", " 男", " 女", " 男", " 男", " 男", " 女", " 女", " 男", " 男"
stars = data.frame(star1, birthyear, gender); head(stars)
```

```
##        star1 birthyear gender
## 1       邓超      1979     男
## 2     赵丽颖      1987     女
## 3     郭富城      1965     男
## 4     周润发      1955     男
## 5 杰克布莱克      1969     男
## 6       汤唯      1979     女
```

2. 汇总

```
str(movie)
```

```
## 'data.frame':    19 obs. of  11 variables:
##  $ name       : chr  "叶问3" "美人鱼" "女汉子真爱公式" "西游记之孙悟空三打白骨精" ...
##  $ boxoffice  : num  77060 338583 6184 119957 111694 ...
##  $ doubanscore: num  6.4 6.9 4.5 5.7 4 7.7 6.5 6.4 5 5.6 ...
##  $ type       : chr  "动作" "喜剧" "喜剧" "喜剧" ...
##  $ duration   : int  105 93 93 120 112 95 131 108 95 102 ...
##  $ showtime   : chr  "2016/3/4" "2016/2/8" "2016/3/18" "2016/2/8" ...
##  $ director   : chr  "叶伟信" "周星驰" "郭大雷" "郑保瑞" ...
##  $ star1      : chr  "甄子丹" "邓超" "赵丽颖" "郭富城" ...
##  $ index1     : int  11385 41310 181979 12227 16731 178 13499 14759 13251 6911 ...
##  $ star2      : chr  "张晋" "林允" "张翰" "巩俐" ...
##  $ index2     : int  4105 9292 44277 8546 30277 1540 77260 755 9549 5614 ...
```

```
summary(movie)
```

```
##      name             boxoffice         doubanscore        type
##  Length:19          Min.   :   924.9   Min.   :3.400   Length:19
##  Class :character   1st Qu.:  3799.5   1st Qu.:4.600   Class :character
##  Mode  :character   Median : 12561.5   Median :5.300   Mode  :character
##                     Mean   : 50813.3   Mean   :5.568
##                     3rd Qu.: 77700.9   3rd Qu.:6.450
##                     Max.   :338583.3   Max.   :8.000
```

```
##     duration       showtime           director           star1
##  Min.   : 84.0   Length:19          Length:19          Length:19
##  1st Qu.: 94.5   Class :character   Class :character   Class :character
##  Median : 99.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   :101.5
##  3rd Qu.:107.5
##  Max.   :131.0
##      index1          star2              index2
##  Min.   :   178   Length:19          Min.   :  521
##  1st Qu.:  8232   Class :character   1st Qu.: 3650
##  Median : 12227   Mode  :character   Median : 9292
##  Mean   : 27861                      Mean   :17369
##  3rd Qu.: 24663                      3rd Qu.:20763
##  Max.   :181979                      Max.   :77260
```

```
head(movie)
```

```
##                     name boxoffice doubanscore type duration  showtime
## 1                 叶问3  77060.44         6.4 动作      105  2016/3/4
## 2               美人鱼 338583.26         6.9 喜剧       93  2016/2/8
## 3          女汉子真爱公式   6184.45         4.5 喜剧       93 2016/3/18
## 4 西游记之孙悟空三打白骨精 119956.51         5.7 喜剧      120  2016/2/8
## 5              澳门风云三 111693.89         4.0 喜剧      112  2016/2/8
## 6              功夫熊猫3  99832.53         7.7 喜剧       95 2016/1/29
##   director      star1 index1        star2 index2
## 1   叶伟信      甄子丹  11385         张晋   4105
## 2   周星驰        邓超  41310         林允   9292
## 3   郭大雷      赵丽颖 181979         张翰  44277
## 4   郑保瑞      郭富城  12227         巩俐   8546
## 5     王晶      周润发  16731       刘德华  30277
## 6   吕寅荣 杰克布莱克     178 安吉丽娜朱莉   1540
```

3. 变大–数据框的增列、合并

```
# 添加一列数据 prefer
prefer = 1:19
movie$pre = prefer
head(movie)
```

```
##                     name boxoffice doubanscore type duration  showtime
```

```
## 1                    叶问3  77060.44         6.4 动作      105  2016/3/4
## 2                   美人鱼 338583.26         6.9 喜剧       93  2016/2/8
## 3           女汉子真爱公式   6184.45         4.5 喜剧       93 2016/3/18
## 4 西游记之孙悟空三打白骨精 119956.51         5.7 喜剧      120  2016/2/8
## 5                澳门风云三 111693.89         4.0 喜剧      112  2016/2/8
## 6                 功夫熊猫3  99832.53         7.7 喜剧       95 2016/1/29
##   director      star1 index1        star2 index2 pre
## 1   叶伟信      甄子丹  11385         张晋   4105   1
## 2   周星驰        邓超  41310         林允   9292   2
## 3   郭大雷      赵丽颖 181979         张翰  44277   3
## 4   郑保瑞      郭富城  12227         巩俐   8546   4
## 5     王晶      周润发  16731       刘德华  30277   5
## 6   吕寅荣 杰克布莱克    178 安吉丽娜朱莉   1540   6
```

```
# merge 实现的效果是: 将 movie 和 stars 按照列 star1 匹配并合并起来
(movie.star = merge(movie[1:3, ], stars,by = "star1"))
```

```
##    star1          name boxoffice doubanscore type duration  showtime director
## 1   邓超        美人鱼 338583.26         6.9 喜剧       93  2016/2/8   周星驰
## 2 赵丽颖 女汉子真爱公式   6184.45         4.5 喜剧       93 2016/3/18   郭大雷
##   index1 star2 index2 pre birthyear gender
## 1  41310  林允   9292   2      1979      男
## 2 181979  张翰  44277   3      1987      女
```

```
# all.x=T, 即取前一个数据框 movie 中 star1 列所有的值做合并, 匹配不到赋值 NA
(movie.star = merge(movie[1:3, ], stars[1:5, ], by = "star1", all.x = T))
```

```
##    star1          name boxoffice doubanscore type duration  showtime director
## 1   邓超        美人鱼 338583.26         6.9 喜剧       93  2016/2/8   周星驰
## 2 赵丽颖 女汉子真爱公式   6184.45         4.5 喜剧       93 2016/3/18   郭大雷
## 3 甄子丹          叶问3  77060.44         6.4 动作      105  2016/3/4   叶伟信
##   index1 star2 index2 pre birthyear gender
## 1  41310  林允   9292   2      1979      男
## 2 181979  张翰  44277   3      1987      女
## 3  11385  张晋   4105   1        NA   <NA>
```

4. 变小–数据的筛选、引用

引用

```
movie[3, ]  # 查看第 3 行的电影信息
```

```
##             name boxoffice doubanscore type duration  showtime director  star1
## 3 女汉子真爱公式   6184.45         4.5 喜剧       93 2016/3/18   郭大雷 赵丽颖
##   index1 star2 index2 pre
## 3 181979  张翰   44277   3
```

```
movie[, 8]  # 查看第 8 列主演者的名字
```

```
##  [1] "甄子丹"     "邓超"       "赵丽颖"     "郭富城"     "周润发"
##  [6] "杰克布莱克" "汤唯"       "白敬亭"     "陈晓"       "梁家辉"
## [11] "姚晨"       "宋茜"       "黄宗泽"     "黄晓明"     "洪金宝"
## [16] "陈坤"       "陶泽如"     "刘亦菲"     "何润东"
```

```
# 筛选
movie$star1  # 用 $ 符号通过列名引用
```

```
##  [1] "甄子丹"     "邓超"       "赵丽颖"     "郭富城"     "周润发"
##  [6] "杰克布莱克" "汤唯"       "白敬亭"     "陈晓"       "梁家辉"
## [11] "姚晨"       "宋茜"       "黄宗泽"     "黄晓明"     "洪金宝"
## [16] "陈坤"       "陶泽如"     "刘亦菲"     "何润东"
```

```
(action = movie[movie$type == " 动作", ])  # 选择数据中的动作电影
```

```
##             name boxoffice doubanscore type duration  showtime director  star1
## 1         叶问3  77060.44         6.4 动作      105  2016/3/4   叶伟信 甄子丹
## 10     冰河追凶   4262.14         5.6 动作      102 2016/4/15     徐伟 梁家辉
## 15 我的特工爷爷  32009.37         5.3 动作       99  2016/4/1   洪金宝 洪金宝
## 19         钢刀    924.86         4.3 动作       94 2016/5/20     阿甘 何润东
##    index1  star2 index2 pre
## 1   11385   张晋   4105   1
## 10   6911 佟大为   5614  10
## 15   9148 刘德华  30277  15
## 19  11822 李学东    521  19
```

```
(action_long = movie[movie$type == " 动作" & movie$duration > 100, ])  # 放映时间超过 100 分钟的动作
```

```
##        name boxoffice doubanscore type duration  showtime director  star1
```

```
## 1      叶问3  77060.44         6.4 动作      105  2016/3/4   叶伟信 甄子丹
## 10 冰河追凶   4262.14         5.6 动作      102 2016/4/15     徐伟 梁家辉
##    index1  star2 index2 pre
## 1   11385   张晋   4105   1
## 10   6911 佟大为   5614  10
```

5. 变序–数据框的内部排序

```
# 按照票房降序排列
movie = movie[order(movie$boxoffice, decreasing = T), ]; head(movie)
```

```
##                      name boxoffice doubanscore type duration  showtime
## 2                  美人鱼 338583.26         6.9 喜剧       93  2016/2/8
## 4 西游记之孙悟空三打白骨精 119956.51         5.7 喜剧      120  2016/2/8
## 5              澳门风云三 111693.89         4.0 喜剧      112  2016/2/8
## 6               功夫熊猫3  99832.53         7.7 喜剧       95 2016/1/29
## 7 北京遇上西雅图之不二情书  78341.38         6.5 喜剧      131 2016/4/29
## 1                  叶问3  77060.44         6.4 动作      105  2016/3/4
##   director      star1 index1        star2 index2 pre
## 2   周星驰        邓超  41310         林允   9292   2
## 4   郑保瑞       郭富城  12227         巩俐   8546   4
## 5     王晶       周润发  16731       刘德华  30277   5
## 6   吕寅荣 杰克布莱克    178 安吉丽娜朱莉   1540   6
## 7   薛晓路        汤唯  13499       吴秀波  77260   7
## 1   叶伟信       甄子丹  11385         张晋   4105   1
```

```
# 先按电影类型排序，再按照豆瓣评分排序
movie = movie[order(movie$type, movie$doubanscore, decreasing = T), ]; head(movie)
```

```
##                      name boxoffice doubanscore type duration  showtime
## 6               功夫熊猫3  99832.53         7.7 喜剧       95 2016/1/29
## 2                  美人鱼 338583.26         6.9 喜剧       93  2016/2/8
## 7  北京遇上西雅图之不二情书  78341.38         6.5 喜剧      131 2016/4/29
## 4  西游记之孙悟空三打白骨精 119956.51         5.7 喜剧      120  2016/2/8
## 13                刑警兄弟   3005.96         5.2 喜剧       97 2016/4/22
## 3           女汉子真爱公式   6184.45         4.5 喜剧       93 2016/3/18
##    director      star1 index1        star2 index2 pre
## 6    吕寅荣 杰克布莱克    178 安吉丽娜朱莉   1540   6
## 2    周星驰        邓超  41310         林允   9292   2
## 7    薛晓路        汤唯  13499       吴秀波  77260   7
```

```
## 4    郑保瑞     郭富城  12227         巩俐   8546   4
## 13   戚家基     黄宗泽   9823         金刚   4010  13
## 3    郭大雷     赵丽颖 181979         张翰  44277   3
```

6. 变形–长宽表互换

```
# install.packages("reshape")
library(reshape)
```

```
## Warning: package 'reshape' was built under R version 4.0.5
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:Matrix':
##
##     expand
```

```
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```
# install.packages("reshape2")
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.0.5
```

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:reshape':
##
##     colsplit, melt, recast
```

```
## (1) 宽表变长表 ##
mWide = data.frame(Name = c(" 熊大", " 水妈"), Type = c(" 帅哥", " 美女"),
                   GF2013 = c(300, 100), GF2014 = c(500, 350), GF2015 = c(1000, 886))
# 由于构造数据框时列名不可以为纯数字，在数字前添加 GF
# 将列名中的 GF 去掉
colnames(mWide)[3:5] = gsub("GF", "", colnames(mWide)[3:5])
mWide # 查看原表
```

```
##   Name Type 2013 2014 2015
```

```
## 1 熊大 帅哥  300  500 1000
## 2 水妈 美女  100  350  886
```

```
(mLong = reshape::melt(mWide, id.vars = c("Name", "Type"), variable_name = "Year"))
```

```
##   Name Type Year value
## 1 熊大 帅哥 2013   300
## 2 水妈 美女 2013   100
## 3 熊大 帅哥 2014   500
## 4 水妈 美女 2014   350
## 5 熊大 帅哥 2015  1000
## 6 水妈 美女 2015   886
```

```
# 将列 Year 从字符型变成数值型
mLong$Year = as.numeric(mLong$Year)

# 长表变宽表
reshape2::dcast(mLong, Name + Type ~ Year)
```

```
##   Name Type   1   2    3
## 1 水妈 美女 100 350  886
## 2 熊大 帅哥 300 500 1000
```

7.R 中的数据透视表-神奇的 ddply

Excel 中，常用的功能是 vlookup 和数据透视表，在 R 中，ddply 函数可以完成类似数据透视表的分组计算不同量的功能。

```
# install.packages(plyr)
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:reshape':
##
##     rename, round_any
```

```
# 根据电影类型进行分组，查看不同类型电影票房的平均水平
popular_type = ddply(movie, .(type), function(x) {mean(x$boxoffice)}); head(popular_type)
```

```
##   type       V1
```

```
## 1 爱情 11206.95
## 2 动作 28564.20
## 3 犯罪 36624.84
## 4 剧情  6671.91
## 5 喜剧 95116.85
```

```
# 根据电影类型和电影时长同时分组，查看电影票房的平均水平
long = ddply(movie, .(type,duration), function(x) {mean(x$index1)}); head(long)
```

```
##   type duration    V1
## 1 爱情       84 58355
## 2 爱情       95 13251
## 3 爱情      108 14759
## 4 动作       94 11822
## 5 动作       99  9148
## 6 动作      102  6911
```

```
rm(list = ls())
```

2.1 R 中的数据类型

2.1.5 列表

1. 创建

```
(example = list("abc", 3:5, matrix(1, nrow = 3, ncol = 4), data.frame(x = 1:4, y = paste0("boy_",
```

```
## [[1]]
## [1] "abc"
##
## [[2]]
## [1] 3 4 5
##
## [[3]]
##      [,1] [,2] [,3] [,4]
## [1,]    1    1    1    1
## [2,]    1    1    1    1
## [3,]    1    1    1    1
##
```

```
## [[4]]
##   x     y
## 1 1 boy_1
## 2 2 boy_2
## 3 3 boy_3
## 4 4 boy_4
```

```
### 2. 基本操作 ###
# 查看
(complex = list(first = list(1:2), second = list(letters, list(matrix(1:4, nrow = 2, ncol = 2)))))
```

```
## $first
## $first[[1]]
## [1] 1 2
##
##
## $second
## $second[[1]]
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
##
## $second[[2]]
## $second[[2]][[1]]
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```
# 利用名字引用元素
complex$first
```

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

```
# 利用序号引用元素
complex[[1]]
```

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

```
# 利用序号添加元素
complex[[3]] = matrix(1, 2, 3); complex
```

```
## $first
```

```
## $first[[1]]
## [1] 1 2
##
##
## $second
## $second[[1]]
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
##
## $second[[2]]
## $second[[2]][[1]]
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
##
##
## [[3]]
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    1    1    1
```

```
# 利用名字添加元素
complex$new = 1:5; complex
```

```
## $first
## $first[[1]]
## [1] 1 2
##
##
## $second
## $second[[1]]
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
##
## $second[[2]]
## $second[[2]][[1]]
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```
##
##
##
## [[3]]
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    1    1    1
##
## $new
## [1] 1 2 3 4 5
```

3. 列表中的 **ply 函数

```
# 老王耗子药的单价，单位（元/袋）
(price = list(year2014 = 36:33, year2015 = 32:35, year2016 = 30:27))
```

```
## $year2014
## [1] 36 35 34 33
##
## $year2015
## [1] 32 33 34 35
##
## $year2016
## [1] 30 29 28 27
```

```
# lapply 返回列表
lapply(price, mean)
```

```
## $year2014
## [1] 34.5
##
## $year2015
## [1] 33.5
##
## $year2016
## [1] 28.5
```

```
lapply(price, sd)
```

```
## $year2014
## [1] 1.290994
```

```
##
## $year2015
## [1] 1.290994
##
## $year2016
## [1] 1.290994
```

```
lapply(price, quantile)
```

```
## $year2014
##    0%   25%   50%   75%  100%
## 33.00 33.75 34.50 35.25 36.00
##
## $year2015
##    0%   25%   50%   75%  100%
## 32.00 32.75 33.50 34.25 35.00
##
## $year2016
##    0%   25%   50%   75%  100%
## 27.00 27.75 28.50 29.25 30.00
```

```
# sapply 默认返回向量或矩阵
sapply(price, mean)
```

```
## year2014 year2015 year2016
##     34.5     33.5     28.5
```

```
sapply(price, sd)
```

```
## year2014 year2015 year2016
## 1.290994 1.290994 1.290994
```

```
sapply(price, quantile)
```

```
##      year2014 year2015 year2016
## 0%      33.00    32.00    27.00
## 25%     33.75    32.75    27.75
## 50%     34.50    33.50    28.50
## 75%     35.25    34.25    29.25
## 100%    36.00    35.00    30.00
```

```
# mapply 实现了将 price 与 amount 对应元素相乘的效果
(amount = list(year2014 = rep(200, 4), year2015 = rep(100, 4), year2016 = rep(300, 4)))
```

```
## $year2014
## [1] 200 200 200 200
##
## $year2015
## [1] 100 100 100 100
##
## $year2016
## [1] 300 300 300 300
```

```
(income_quarter = mapply("*", price, amount))
```

```
##      year2014 year2015 year2016
## [1,]     7200     3200     9000
## [2,]     7000     3300     8700
## [3,]     6800     3400     8400
## [4,]     6600     3500     8100
```

```
# 练习题：总收入
(income_year = mapply(function(x, y) {sum(x*y)}, price, amount))
```

```
## year2014 year2015 year2016
##    27600    13400    34200
```

4.list 对象的其他快捷玩法

```
# do.call
Sunday1 = data.frame(" 经度" = rep(39.95, 5), " 纬度" = rep(116.3, 5), " 地点" = rep(" 熊孩子玩耍基地
Sunday2 = data.frame(" 经度" = rep(39.96, 5), " 纬度" = rep(116.4, 5), " 地点" = rep(" 论文生产基地"
Sunday3 = data.frame(" 经度" = rep(39.97, 5), " 纬度" = rep(116.5, 5), " 地点" = rep(" 工业实践基地"
(example = list(Sunday1, Sunday2, Sunday3))
```

```
## [[1]]
##    经度  纬度           地点
## 1 39.95 116.3 熊孩子玩耍基地
## 2 39.95 116.3 熊孩子玩耍基地
## 3 39.95 116.3 熊孩子玩耍基地
## 4 39.95 116.3 熊孩子玩耍基地
## 5 39.95 116.3 熊孩子玩耍基地
##
## [[2]]
##    经度  纬度         地点
```

```
## 1 39.96 116.4 论文生产基地
## 2 39.96 116.4 论文生产基地
## 3 39.96 116.4 论文生产基地
## 4 39.96 116.4 论文生产基地
## 5 39.96 116.4 论文生产基地
##
## [[3]]
##    经度  纬度          地点
## 1 39.97 116.5 工业实践基地
## 2 39.97 116.5 工业实践基地
## 3 39.97 116.5 工业实践基地
## 4 39.97 116.5 工业实践基地
## 5 39.97 116.5 工业实践基地
```

```
do.call(rbind, example)
```

```
##     经度  纬度           地点
## 1  39.95 116.3 熊孩子玩耍基地
## 2  39.95 116.3 熊孩子玩耍基地
## 3  39.95 116.3 熊孩子玩耍基地
## 4  39.95 116.3 熊孩子玩耍基地
## 5  39.95 116.3 熊孩子玩耍基地
## 6  39.96 116.4   论文生产基地
## 7  39.96 116.4   论文生产基地
## 8  39.96 116.4   论文生产基地
## 9  39.96 116.4   论文生产基地
## 10 39.96 116.4   论文生产基地
## 11 39.97 116.5   工业实践基地
## 12 39.97 116.5   工业实践基地
## 13 39.97 116.5   工业实践基地
## 14 39.97 116.5   工业实践基地
## 15 39.97 116.5   工业实践基地
```

参考文献

朱雪宁，《R 语言从数据思维到数据实战》，中国人民大学出版社，2019