R处理数据

温灿红

中国科学技术大学管理学院

R处理数据

- 获取数据
- 导出数据
- 预处理数据
- 汇总数据
- 产生随机数

获取数据

导入数据——R语言系统格式数据

- 读取**R**二进制文档，格式为".RData"或".Rda"
  - load("name") 读取 name 文件到当前**R**工作空间中

```
load("cats.RData")                    # 读取数据
```

导入数据——带有分隔符的文本数据

- 带有分隔符的文本数据形式是常用的数据存储方式之一。
- 列与列之间以固定字符（如逗号、空格、分号等）进行分割
- 用 `read.csv()`，`read.table()`，`read.delim()`，`read.fwf()` 等函数读入。
- 文本文件也可通过 `readLines()` 逐行读取。
- 这些函数读取数据后会在**R**中保存成 `data.frame` 的格式。
- 可通过改变选择来调整是否需要包含列名，是否需要跳过开头的若干行以及指定缺失值符号等等。

导入数据——带有分隔符的文本数据

read.table()参数设置：

- file:文件名，后缀可以是txt、dat、csv等。
- header:导入数据时是否带有列标题，默认是TRUE。
- sep: 列与列之间的文本分隔符。
- stringsAsFactor: 导入数据时是否奖字符串数据转化成因子类型，默认是TRUE。
- fileEncoding:文本数据的文件编码，如果设计中文字符，根据文本格式更改GBK或UTF-8，默认是UTF-8。

读取数据的例子（一）

- header 是否包括列名，如果原文件中包含列名，则为TRUE，否则为FALSE。

```
d <- read.csv("covid19.csv", header = TRUE) #  header = TRUE 包括列名
head(d)                                     #  查看前面几行，默认是6行。
```

```
##   序号   分型 性别 年龄 入院时间 出院时间 疗程
## 1    1   重型   男   49 20200121 20200211   22
## 2    2   重型       47 20200121 20200214   25
## 3    3   重型   男   51 20200124 20200207   15
## 4    4 普通型   男   24 20200122 20200203   13
## 5    5 普通型   男   47 20200122 20200208   18
## 6    6 普通型   女   40 20200125 20200208   15
```

- 第二个病人的性别没有记载，应认定为缺失值。

```
table(d$性别)
```

```
##
##    男 女
##  1 43 31
```

读取数据的例子（二）

- na.strings = x 指定 x 为缺失值。如在该数据中，空的字符串为缺失值。

```
d <- read.csv("covid19.csv", header = TRUE, na.strings = "") #  第二个病人的性别没有记载，应认定
head(d, n = 3)
```

```
##   序号 分型 性别 年龄 入院时间 出院时间 疗程
## 1    1 重型   男   49 20200121 20200211   22
## 2    2 重型 <NA>   47 20200121 20200214   25
## 3    3 重型   男   51 20200124 20200207   15
```

```
table(d$性别)
```

```
##
## 男 女
## 43 31
```

```
table(d$性别, useNA = "always")
```

```
##
##   男   女 <NA>
##   43   31    1
```

读取数据的例子（三）

- row.names = x 指定列名为 x 的列为行名

```
dim(d)
```

```
## [1] 75  7
```

```
d <- read.csv("covid19.csv", header = TRUE, na.strings = "",
              row.names = "序号") #  指定第一列为行名
head(d,3)
```

```
##   分型 性别 年龄 入院时间 出院时间 疗程
## 1 重型   男   49 20200121 20200211   22
## 2 重型 <NA>   47 20200121 20200214   25
## 3 重型   男   51 20200124 20200207   15
```

```
dim(d)
```

```
## [1] 75  6
```

读取数据的例子（四）

- skip = x 跳过前面 x 行

```
d1 <- read.csv("covid19.csv", header = TRUE, na.strings = "", skip = 3)
dim(d1)
```

```
## [1] 72  7
```

```
dim(d)
```

```
## [1] 75  6
```

- nrows = x 只读取 x 行，若header = TRUE 则不包括列名那一行

```
d2 <- read.csv("covid19.csv", header = TRUE, na.strings = "",
               row.names = "序号", nrows = 10)
dim(d2)
```

```
## [1] 10  6
```

读取数据的例子（五）

- CSV文件是文本文件，是有编码问题的， 尤其是中文内容的文件。 read.csv默认是GBK格式，如果有时候发现读进去的数据是乱码，可试试看改成”UTF-8 “

```
d2 <- read.csv("covid19.csv", header = TRUE, na.strings = "",
               row.names = "序号", fileEncoding="UTF-8")
```

读取数据的例子（六）

- readLines() 按行读取，每一行记为一个字符串

```
d3 <- readLines("covid19.csv")
head(d3, 3)
```

```
## [1] "序号,分型,性别,年龄,入院时间,出院时间,疗程"
## [2] "1,重型,男,49,20200121,20200211,22"
## [3] "2,重型,,47,20200121,20200214,25"
```

- 若文件太大，可通过 readLines() 函数读取前面几行

```
d3 <- readLines("covid19.csv", n = 3)
d3
```

```
## [1] "序号,分型,性别,年龄,入院时间,出院时间,疗程"
## [2] "1,重型,男,49,20200121,20200211,22"
## [3] "2,重型,,47,20200121,20200214,25"
```

导入数据——excel

- readxl包是RStudio内置读取Excel数据文件的专用包。使用read_excel()读取。
- read_excel()参数设置：
  - path: 文件名，可以是xls或者是xlsx
  - sheet: 工作簿名称或序号
  - range：读取指定区域的数据，如B3:D87表示读取B3至D87区域的数据
  - col_names:判断第一行是否作为列名称，默认为TRUE

读取Excel数据的例子

```
data <- readxl::read_excel("covid19.xlsx", col_names = TRUE)
head(data)                                    #  查看前面几行，默认是6行。
```

```
## # A tibble: 6 × 7
##    序号 分型   性别   年龄 入院时间 出院时间  疗程
##   <dbl> <chr>  <chr> <dbl>    <dbl>    <dbl> <dbl>
## 1     1 重型   男       49 20200121 20200211    22
## 2     2 重型   <NA>    47 20200121 20200214    25
## 3     3 重型   男       51 20200124 20200207    15
## 4     4 普通型 男       24 20200122 20200203    13
## 5     5 普通型 男       47 20200122 20200208    18
## 6     6 普通型 女       40 20200125 20200208    15
```

导出数据

导出数据

- 导出**R**二进制文档，格式为".RData"或".Rda"
  - save(thing, file = "name") 把指定的若干个变量thing（直接用名字，不需要表示成字符串）保存到用 name 文件中
  - save.image("name") 把当前工作空间中的所有变量保存到 name 文件

```
data(cats,package="MASS")              # 读取扩展包MASS中自带的数据"cats"
save(cats, file = "cats.RData")        # 保存数据“cats”到“cats.RData“
rm(cats)
ls()
```

```
## [1] "d"    "d1"   "d2"   "d3"   "data"
```

导出数据到文本文件

- 可用 write.csv() , write.table() , write.delim() , write.fwf() , writeLines() 等函数输出data.frame至外面文件。

```
write.table(d, file="mydata1.txt") # 可能会导致乱码
write.table(d, file="mydata2.txt", fileEncoding = "UTF-8")
```

导出数据到Excel文件

- openxlsx包的write.xlsx()。

```
openxlsx::write.xlsx(d, file="mydata.xlsx", asTable=TRUE)
```

预处理数据

查看数据

- 通过行序号选择
  - head(x, n) 选择数据框 x 的前 n 行
  - tail(x, n) 选择数据框 x 的倒数 n 行
- View(x)：打开界面查看数据。

```
head(d)
```

```
##      分型 性别 年龄 入院时间 出院时间 疗程
## 1    重型   男   49 20200121 20200211   22
## 2    重型 <NA>   47 20200121 20200214   25
## 3    重型   男   51 20200124 20200207   15
## 4 普通型   男   24 20200122 20200203   13
## 5 普通型   男   47 20200122 20200208   18
## 6 普通型   女   40 20200125 20200208   15
```

```
tail(d, n = 2)
```

```
##     分型 性别 年龄 入院时间 出院时间 疗程
## 83 重型   男   72 20200221 20200225    5
## 84 重型   男   29 20200208 20200220   13
```

```
View(d)
```

筛选数据

- 按条件选出符合条件的行组成的子集，用逻辑判断语句
- subset()：提出子集，参数subset提取行，参数select提取列。

```
d[d$年龄<18,]
```

```
##       分型 性别 年龄 入院时间 出院时间 疗程
## 21 普通型   女   16 20200130 20200212   14
## 82 普通型   男    5 20200131 20200213   NA
```

```
subset(d, d$年龄<18)
```

```
##       分型 性别 年龄 入院时间 出院时间 疗程
## 21 普通型   女   16 20200130 20200212   14
## 82 普通型   男    5 20200131 20200213   NA
```

```
with(d, d[年龄<18,])
```

```
##       分型 性别 年龄 入院时间 出院时间 疗程
## 21 普通型   女   16 20200130 20200212   14
## 82 普通型   男    5 20200131 20200213   NA
```

筛选数据

- 同时选择行和列的子集

```
    d[d$年龄<18 & d[,"性别"]=="男", c("性别","年龄","疗程")]
```

```
##    性别 年龄 疗程
## 82   男    5   NA
```

```
    subset(d, d$年龄<18 & d[,"性别"]=="男", select = c("性别","年龄","疗程"))
```

```
##    性别 年龄 疗程
## 82   男    5   NA
```

排序数据

1. 使用order()对需要排序的列进行排序，返回排序后各个元素的原位置信息。
2. 基于order()函数的输出结果对数据框排序。

```
    idx1 <- order(d[,"年龄"]) # 按照一列进行排序
    head(d[idx1,],6)
```

```
##       分型 性别 年龄 入院时间 出院时间 疗程
## 82 普通型   男    5 20200131 20200213   NA
## 21 普通型   女   16 20200130 20200212   14
## 17 普通型   男   21 20200129 20200224   27
## 36 普通型   男   21 20200203 20200218   16
## 15 普通型   男   22 20200129 20200213   16
## 16 普通型   男   22 20200129 20200225   28
```

```
    idx2 <- order(as.factor(d[,"性别"]), d[,"年龄"], decreasing = c(T, T), method="radix") # 按
    head(d[idx2,],6)
```

```
##       分型 性别 年龄 入院时间 出院时间 疗程
## 47 普通型   女   69 20200207 20200301   NA
## 31 普通型   女   66 20200201 20200220   20
## 35 普通型   女   56 20200203 20200225   23
## 63   重型   女   56 20200209 20200226   18
## 80   重型   女   56 20200131 20200223   24
## 51 普通型   女   55 20220205 20200222   18
```

合并数据（一）

- 如果行号都一样，那么使用data.frame(dat1,dat2)或者cbind(dat1,dat2)直接合并即可。

```
hos <- read.csv("covid19.csv", header = TRUE, na.strings = "")
cli <- read.csv("covid19-2.csv", header = TRUE)
newdata <- data.frame(hos, cli)
head(newdata)
```

```
##   序号   分型 性别 年龄 入院时间 出院时间 疗程 序号.1     职业 病史 吸烟
## 1    1   重型   男   49 20200121 20200211   22      1     职员 首诊   无
## 2    2   重型 <NA>   47 20200121 20200214   25      2     无业 首诊   无
## 3    3   重型   男   51 20200124 20200207   15      3     职员 首诊   有
## 4    4 普通型   男   24 20200122 20200203   13      4 地铁工作 首诊   无
## 5    5 普通型   男   47 20200122 20200208   18      5     销售 首诊   有
## 6    6 普通型   女   40 20200125 20200208   15      6     个体 首诊   无
```

- 有两列是重复的，因此列名改成了“序号”，“序号.1”。

合并数据（二）

- 如果行号不一样，用 merge(dat1, dat2, by = x) 按照列名为x来合并，只保留x元素相同的行，即同时在两个数据框中的行。

```
hos <- read.csv("covid19.csv", header = TRUE, na.strings = "")
dim(hos)
```

```
## [1] 75  7
```

```
cli <- read.csv("covid19-3.csv", header = TRUE)
dim(cli)
```

```
## [1] 69  4
```

```
newdata <- merge(hos, cli, by = "序号")
dim(newdata)
```

```
## [1] 69 10
```

合并数据（三）

- 用 merge(dat1, dat2, by.x = x, by.y = y) 把 dat1中的x列和 dat2中的y列作为合并的标准。

```
newdata <- merge(hos, cli, by.x = "序号", by.y = "序号")
dim(newdata)
```

```
## [1] 69 10
```

- 如果想要保留 dat1 中的所有行，则指定 all.x = TRUE。

```
newdata <- merge(hos, cli, by.x = "序号", by.y = "序号", all.x = TRUE)
dim(newdata)
```

```
## [1] 75 10
```

标准化数据

- scale() 把每一列都标准化， 即每一列都减去该列的平均值，然后除以该列的样本标准差。
- scale(x, center=TRUE, scale=FALSE) 仅中心化而不标准化。
- 仅适用于数值型的变量

```
    ds <- scale(d[,3])
    head(ds)
```

```
##             [,1]
## [1,]  0.2552160
## [2,]  0.1370604
## [3,]  0.3733716
## [4,] -1.2217285
## [5,]  0.1370604
## [6,] -0.2764840
```

案例分析

初看数据

```
summary(d)
```

```
##      分型               性别               年龄          入院时间
##  Length:75          Length:75          Min.   : 5.00   Min.   :20200121
##  Class :character   Class :character   1st Qu.:31.00   1st Qu.:20200129
##  Mode  :character   Mode  :character   Median :47.00   Median :20200131
##                                        Mean   :44.68   Mean   :20200433
##                                        3rd Qu.:54.00   3rd Qu.:20200206
##                                        Max.   :91.00   Max.   :20220205
##
##     出院时间            疗程
##  Min.   :20200129   Min.   : 5
##  1st Qu.:20200213   1st Qu.:13
##  Median :20200220   Median :16
##  Mean   :20200226   Mean   :17
##  3rd Qu.:20200224   3rd Qu.:21
##  Max.   :20200307   Max.   :29
##                     NA's   :11
```

调整数据的类型

- 日期数据没有正确读入

```
    class(d$出院时间)
```

```
## [1] "integer"
```

```
    d[,"入院时间"] <- as.Date(as.character(d[,"入院时间"]), format = "%Y%m%d")
    d[,"出院时间"] <- as.Date(as.character(d[,"出院时间"]), format = "%Y%m%d")
    class(d$出院时间)
```

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

```
    summary(d[, 4:5])
```

```
##      入院时间              出院时间
##  Min.   :2020-01-21   Min.   :2020-01-29
##  1st Qu.:2020-01-29   1st Qu.:2020-02-13
##  Median :2020-01-31   Median :2020-02-20
##  Mean   :2020-02-11   Mean   :2020-02-18
##  3rd Qu.:2020-02-06   3rd Qu.:2020-02-24
##  Max.   :2022-02-05   Max.   :2020-03-07
```

增加新的列变量（一）

- 可以为数据框计算新变量， 返回含有新变量以及原变量的新数据框

```
d[,"住院时间"] <- as.numeric(d[,"出院时间"] - d[,"入院时间"]) + 1
summary(d[, 4:7])
```

```
##      入院时间             出院时间             疗程        住院时间
##  Min.   :2020-01-21   Min.   :2020-01-29   Min.   : 5   Min.   :-713.000
##  1st Qu.:2020-01-29   1st Qu.:2020-02-13   1st Qu.:13   1st Qu.:  14.000
##  Median :2020-01-31   Median :2020-02-20   Median :16   Median :  18.000
##  Mean   :2020-02-11   Mean   :2020-02-18   Mean   :17   Mean   :   8.787
##  3rd Qu.:2020-02-06   3rd Qu.:2020-02-24   3rd Qu.:21   3rd Qu.:  23.000
##  Max.   :2022-02-05   Max.   :2020-03-07   Max.   :29   Max.   :  35.000
##                                            NA's   :11
```

- 增加的新变量有异常值，最小值竟然为负数！！！

增加新的列变量（二）

```
d[d$住院时间<0,]
```

```
##       分型 性别 年龄     入院时间     出院时间 疗程 住院时间
## 51 普通型    女    55 2022-02-05 2020-02-22    18      -713
```

- 发现是原始数据输入有误，入院时间写成了"2022-02-05"，通过查看原始文档，调整为“2020-02-05”。

```
d[d$住院时间<0,"入院时间"] <- as.Date("2020-02-05")
d[,"住院时间"] <- as.numeric(d[,"出院时间"] - d[,"入院时间"]) + 1
summary(d[, 4:7])
```

```
##      入院时间             出院时间             疗程        住院时间
##  Min.   :2020-01-21   Min.   :2020-01-29   Min.   : 5   Min.   : 5.00
##  1st Qu.:2020-01-29   1st Qu.:2020-02-13   1st Qu.:13   1st Qu.:14.00
##  Median :2020-01-31   Median :2020-02-20   Median :16   Median :18.00
##  Mean   :2020-02-01   Mean   :2020-02-18   Mean   :17   Mean   :18.53
##  3rd Qu.:2020-02-06   3rd Qu.:2020-02-24   3rd Qu.:21   3rd Qu.:23.00
##  Max.   :2020-02-21   Max.   :2020-03-07   Max.   :29   Max.   :35.00
##                                            NA's   :11
```

有缺失值的行

```
    idx.na <- apply(is.na(d), 1, any)
    d[idx.na, ]
```

```
##       分型 性别 年龄    入院时间    出院时间 疗程 住院时间
## 2     重型 <NA>   47 2020-01-21 2020-02-14   25       25
## 19 普通型   男   22 2020-01-29 2020-03-03   NA       35
## 29 普通型   女   27 2020-01-31 2020-03-05   NA       35
## 37 普通型   女   53 2020-02-03 2020-03-05   NA       32
## 42 普通型   男   35 2020-01-28 2020-03-01   NA       34
## 45 普通型   女   45 2020-02-07 2020-03-01   NA       24
## 46 普通型   女   42 2020-02-07 2020-03-02   NA       25
## 47 普通型   女   69 2020-02-07 2020-03-01   NA       24
## 50 普通型   女   35 2020-02-05 2020-02-27   NA       23
## 55    重型   男   63 2020-02-11 2020-03-07   NA       26
## 57    重型   男   57 2020-02-01 2020-02-29   NA       29
## 82 普通型   男    5 2020-01-31 2020-02-13   NA       14
```

汇总数据

汇总数据

常用的汇总函数有：

- 总体信息： `summary()`, `table()`
- 位置度量： `mean()`,`median()`。
- 分散程度（变异性）度量： `sd()`,`IQR()`,`mad()`。
- 分位数： `min()`,`max()`,`quantile()`。

汇总数据之例子（一）

```
summary(d)
```

```
##      分型               性别               年龄          入院时间
##  Length:75          Length:75          Min.   : 5.00   Min.   :2020-01-21
##  Class :character   Class :character   1st Qu.:31.00   1st Qu.:2020-01-29
##  Mode  :character   Mode  :character   Median :47.00   Median :2020-01-31
##                                        Mean   :44.68   Mean   :2020-02-01
##                                        3rd Qu.:54.00   3rd Qu.:2020-02-06
##                                        Max.   :91.00   Max.   :2020-02-21
##
##     出院时间             疗程        住院时间
##  Min.   :2020-01-29   Min.   : 5   Min.   : 5.00
##  1st Qu.:2020-02-13   1st Qu.:13   1st Qu.:14.00
##  Median :2020-02-20   Median :16   Median :18.00
##  Mean   :2020-02-18   Mean   :17   Mean   :18.53
##  3rd Qu.:2020-02-24   3rd Qu.:21   3rd Qu.:23.00
##  Max.   :2020-03-07   Max.   :29   Max.   :35.00
##                       NA's   :11
```

汇总数据之例子（二）

-对于因子类型的变量，可以通过table()查看其在每一类的频数分布。

```
table(d$分型)
```

```
##
## 普通型    重型
##     54     21
```

```
table(d$性别)
```

```
##
## 男 女
## 43 31
```

汇总数据之例子（三）

-对于数值型变量，则可以通过一些数字特征来描绘它的分布。

```
summary(d$年龄)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   31.00   47.00   44.68   54.00   91.00
```

```
cat("平均年龄为", mean(d$年龄), "中位数为", median(d$年龄), "标准差为",
    sd(d$年龄), "下90%的分位数为", quantile(d$年龄,0.9))
```

```
## 平均年龄为 44.68 中位数为 47 标准差为 16.92684 下90%的分位数为 67.6
```

```
cat("平均疗程为", mean(d$疗程), "标准差为", sd(d$疗程))
```

```
## 平均疗程为 NA 标准差为 NA
```

- 汇总出错了！！！

汇总数据之例子（四）

- 可通过 na.rm = TRUE 将其中NA的数值去除来计算平均值、标准差、中位数等。

```
cat("平均疗程为", mean(d$疗程, na.rm=TRUE), "标准差为", sd(d$疗程, na.rm=TRUE))
```

```
## 平均疗程为 17 标准差为 5.327378
```

- 在quantile(),IQR()等函数中，如果输入有缺失值，则直接报错，错误如下：

> Error in quantile.default(as.numeric(x), c(0.25, 0.75), na.rm = na.rm, : 'na.rm'如果设为FALSE的话不允许有遗漏值和NaN

汇总数据之例子（五）

-对于时间日期型变量，可类似于数值型变量进行汇总。

```
mean(d$入院时间)
```

```
## [1] "2020-02-01"
```

```
min(d$入院时间)
```

```
## [1] "2020-01-21"
```

```
max(d$入院时间)
```

```
## [1] "2020-02-21"
```

```
sd(d$入院时间)
```

```
## [1] 5.815481
```

分组汇总数据（一）

- aggregate() 函数对输入的数据框用指定的分组变量（或交叉分组） 分组进行概括统计。

```
    aggregate(d[,c(3:5,7)], by = d["分型"], mean)
```

```
##      分型      年龄    入院时间    出院时间 住院时间
## 1 普通型 40.00000 2020-01-30 2020-02-18 19.11111
## 2   重型 56.71429 2020-02-04 2020-02-20 17.04762
```

```
    aggregate(d[,c(3:5,7)], by = d[c("分型","性别")], mean)
```

```
##      分型 性别      年龄    入院时间    出院时间 住院时间
## 1 普通型   男 37.46154 2020-01-29 2020-02-16 19.11538
## 2   重型   男 58.11765 2020-02-05 2020-02-20 16.05882
## 3 普通型   女 42.35714 2020-02-01 2020-02-19 19.10714
## 4   重型   女 52.00000 2020-02-04 2020-02-23 20.00000
```

分组汇总数据（二）

- tapply() 函数对向量进行分组概括

```
    tapply(d[,"性别"], INDEX = d["分型"], table)
```

```
## $普通型
##
## 男 女
## 26 28
##
## $重型
##
## 男 女
## 17  3
```

- 并没有把 NA计算在内。

分组汇总数据（三）

- 通过 useNA = "always" 或useNA = "ifany"来把 NA计算在内。

```
tapply(d["性别"], INDEX = d["分型"],
       table, useNA="always")
```

```
## $普通型
##
##   男   女 <NA>
##   26   28    0
##
## $重型
##
##   男   女 <NA>
##   17    3    1
```

```
tapply(d["性别"], INDEX = d["分型"],
       table, useNA="ifany")
```

```
## $普通型
##
## 男 女
## 26 28
##
## $重型
##
##   男   女 <NA>
##   17    3    1
```

分组汇总数据（四）

- 对两个分类变量进行交叉分组计算频数

```
    table( d[,"分型"], d[,"性别"])
```

```
##
##          男 女
##   普通型 26 28
##   重型   17  3
```

```
    table( d[,"分型"], d[,"性别"], useNA="ifany")
```

```
##
##          男 女 <NA>
##   普通型 26 28    0
##   重型   17  3    1
```

```
    table( d[,"分型"], d[,"性别"], useNA="always")
```

```
##
##          男 女 <NA>
##   普通型 26 28    0
##   重型   17  3    1
##   <NA>    0  0    0
```

dplyr包

dplyr包

- filter(): 按行筛选数据
- select(): 按名称选取变量/列
- arrange(): 对行排序数据
- mutate(): 创建新变量（列）
- summarise(): 汇总数据
- group_by(): 分组汇总数据
- xxx_joins(): 合并数据
- %>%: 管道

导入需要的R包和数据集

```
library(dplyr)
```

```
##
## 载入程辑包：'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
d <- readxl::read_excel("covid19.xlsx", col_names = TRUE)
```

filter()筛选数据

- filter(x,...)：对数据框按照后面的参数来筛选行，后面的参数是用来筛选数据的判断表达式

```
    filter(d, d$年龄<18)
```

```
## # A tibble: 2 × 7
##    序号 分型    性别    年龄 入院时间 出院时间  疗程
##   <dbl> <chr>   <chr> <dbl>    <dbl>    <dbl> <dbl>
## 1    21 普通型 女        16 20200130 20200212    14
## 2    82 普通型 男         5 20200131 20200213    NA
```

```
    filter(d, d$年龄<18, 性别=="男")
```

```
## # A tibble: 1 × 7
##    序号 分型    性别    年龄 入院时间 出院时间  疗程
##   <dbl> <chr>   <chr> <dbl>    <dbl>    <dbl> <dbl>
## 1    82 普通型 男         5 20200131 20200213    NA
```

select()按名称选取变量/列

```
filter(select(d, 性别,年龄, 疗程), d$年龄<18)
```

```
## # A tibble: 2 × 3
##   性别   年龄  疗程
##   <chr> <dbl> <dbl>
## 1 女       16    14
## 2 男        5    NA
```

arrange()排序数据（一）

- arrange(x, ...)：对数据框x排序，后面的参数为一组作为排序依据的列名。如果列名不止一个，那么就后面的列是在前面排序的基础上继续排序。
- desc()降序排序
- 缺失值总是排在最后。

```
head(arrange(d, 年龄),6)
```

```
## # A tibble: 6 × 7
##    序号 分型   性别    年龄 入院时间 出院时间  疗程
##   <dbl> <chr>  <chr> <dbl>    <dbl>    <dbl> <dbl>
## 1    82 普通型 男        5 20200131 20200213    NA
## 2    21 普通型 女       16 20200130 20200212    14
## 3    17 普通型 男       21 20200129 20200224    27
## 4    36 普通型 男       21 20200203 20200218    16
## 5    15 普通型 男       22 20200129 20200213    16
## 6    16 普通型 男       22 20200129 20200225    28
```

arrange()排序数据（二）

```
    head(arrange(d, 性别, desc(年龄)),6) #按两列进行排序
```

```
## # A tibble: 6 × 7
##    序号 分型   性别   年龄 入院时间 出院时间  疗程
##   <dbl> <chr>  <chr> <dbl>    <dbl>    <dbl> <dbl>
## 1    47 普通型 女       69 20200207 20200301    NA
## 2    31 普通型 女       66 20200201 20200220    20
## 3    35 普通型 女       56 20200203 20200225    23
## 4    63 重型   女       56 20200209 20200226    18
## 5    80 重型   女       56 20200131 20200223    24
## 6    51 普通型 女       55 20220205 20200222    18
```

mutate()新增变量

- 可以为数据框计算新变量， 返回含有新变量以及原变量的新数据框。

```
tmp <- mutate(d, 住院时间 = 出院时间 - 入院时间 + 1)
head(tmp)
```

```
## # A tibble: 6 × 8
##    序号 分型   性别    年龄 入院时间 出院时间  疗程 住院时间
##   <dbl> <chr>  <chr>  <dbl>    <dbl>    <dbl> <dbl>    <dbl>
## 1     1 重型   男        49 20200121 20200211    22       91
## 2     2 重型   <NA>     47 20200121 20200214    25       94
## 3     3 重型   男        51 20200124 20200207    15       84
## 4     4 普通型 男        24 20200122 20200203    13       82
## 5     5 普通型 男        47 20200122 20200208    18       87
## 6     6 普通型 女        40 20200125 20200208    15       84
```

summarise()汇总数据（一）

- 按照分型来进行分组汇总

```
    summarise(group_by(d, 分型), mean(年龄), mean(入院时间), mean(出院时间))
```

```
## # A tibble: 2 × 4
##   分型   `mean(年龄)` `mean(入院时间)` `mean(出院时间)`
##   <chr>         <dbl>            <dbl>            <dbl>
## 1 普通型         40          20200528.         20200226.
## 2 重型           56.7        20200188.         20200224.
```

summarise()汇总数据（二）

- 按照分型和性别来进行分组汇总

```
    summarise(group_by(d, 分型,性别), mean(年龄), mean(入院时间), mean(出院时间))
```

```
## `summarise()` has grouped output by '分型'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 5 × 5
## # Groups:   分型 [2]
##   分型   性别  `mean(年龄)` `mean(入院时间)` `mean(出院时间)`
##   <chr>  <chr>        <dbl>            <dbl>            <dbl>
## 1 普通型 女            42.4        20200884.        20200232.
## 2 普通型 男            37.5        20200146.        20200220.
## 3 重型   女            52          20200182.        20200224.
## 4 重型   男            58.1        20200194.        20200225.
## 5 重型   <NA>         47          20200121         20200214
```

管道 %>%

- 管道 %>%帮助你以清晰易懂的方式编写代码
- x%>% f(y)转换为f(x, y)

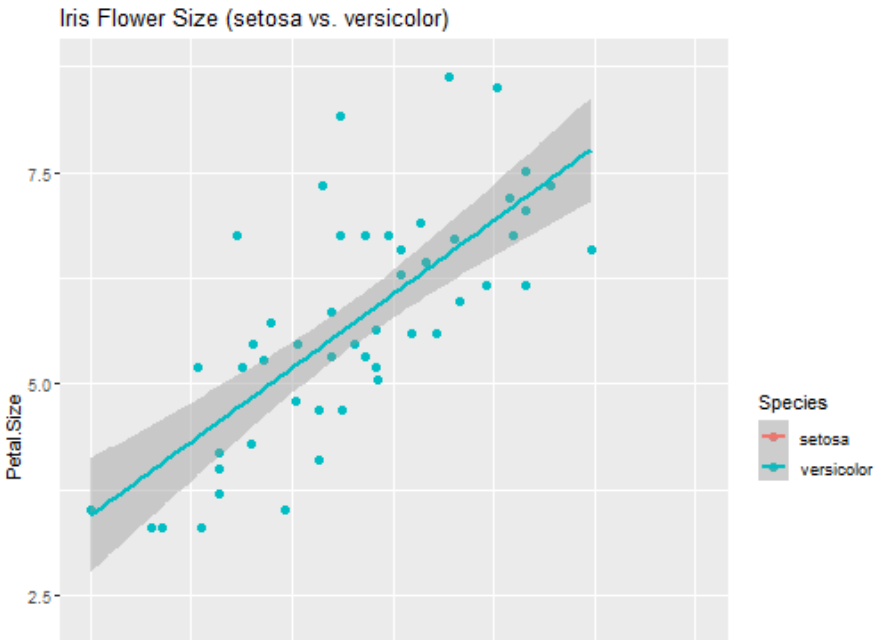
```
d %>%
  mutate(住院时间 = 出院时间 - 入院时间 + 1) %>%
  filter(年龄<18)
```

```
## # A tibble: 2 × 8
##    序号 分型    性别    年龄 入院时间 出院时间  疗程 住院时间
##   <dbl> <chr>   <chr> <dbl>    <dbl>    <dbl> <dbl>    <dbl>
## 1    21 普通型 女        16 20200130 20200212    14       83
## 2    82 普通型 男         5 20200131 20200213    NA       83
```

管道 %>%和画图函数结合

```
library(ggplot2)
iris %>%
  mutate(Sepal.Size = Sepal.Length*Sepal.Width,
         Petal.Size = Petal.Length*Petal.Width) %>%
  filter(Species != 'virginica') %>%
  ggplot(aes(x=Sepal.Size, y=Petal.Size, colour=Species)) +
  geom_point(size=2) +
  geom_smooth(method = "lm") +
  labs(title = 'Iris Flower Size (setosa vs. versicolor)')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



随机数

随机数

- 在计算机中，所谓随机数，实际是“伪随机数”，是从一组起始值（称为种子），按照某种递推算法向前递推得到的。所以，从同一种子出发，得到的随机数序列是相同的。
- 为了得到可重现的结果，随机模拟应该从固定不变的种子开始模拟。用 `set.seed(k)` 来实现固定模拟实验中的种子。
- 语法如下：

```
set.seed(seed, kind = NULL, normal.kind = NULL, sample.kind = NULL)
```

- `seed = k` 指定一个编号为 k的种子，这样每次从编号 k 种子运行相同的模拟程序就可以得到相同的结果;
- `kind =`指定后续程序要使用的随机数发生器名称;
- `normal.kind=` 指定要使用的正态分布随机数发生器名称。

例子

```
    runif(5)
```

```
## [1] 0.1884734 0.6677992 0.5920902 0.9050837 0.2802835
```

```
    runif(5)
```

```
## [1] 0.1182230 0.6643650 0.3519240 0.9321326 0.0329193
```

```
    set.seed(5)
    runif(5)
```

```
## [1] 0.2002145 0.6852186 0.9168758 0.2843995 0.1046501
```

```
    set.seed(5)
    runif(5)
```

```
## [1] 0.2002145 0.6852186 0.9168758 0.2843995 0.1046501
```

从古典概型开始： sample()

- sample() 函数从一个有限集合中无放回或有放回地随机抽取，产生随机结果。
- 语法如下:

```
sample(x, size, replace = FALSE, prob = NULL)
```

- x 用以存储有限集合的向量，也可以为一个正整数（此时集合为1:x）；
- size 指定抽样个数，即样本数;
- replace =指定是否为有放回抽样，TRUE是有放回抽样,FALSE是无放回抽样;
- prob =指定以各种权重抽取，默认是等概率。

例子（一）

- 有放回抽样

```
sample(0:1, size = 10, replace = TRUE)
```

```
##  [1] 0 0 0 1 0 0 0 0 1 1
```

- 无放回等概率抽样：

```
set.seed(1)
sample(1:100, size = 6, replace = FALSE)
```

```
## [1] 68 39  1 34 87 43
```

- 以下为等价命令：

```
set.seed(1)
sample(1:100, size = 6)
```

```
## [1] 68 39  1 34 87 43
```

例子（二）

- 随机排序:

```
sample(10)
```

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

```
sample(letters)
```

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

- 多项分布的随机抽样

```
x <- sample(1:3, size = 100, replace = TRUE, prob = c(.2, .3, .5))
table(x)
```

```
## x
##  1  2  3
## 16 32 52
```

R的随机数函数（一）

- 提供了多种分布的随机数函数，如:
- runif(n) 产生 n 个标准均匀分布随机数;

```
    round(rnorm(5), 2)
```

```
## [1]  0.14 -0.12 -0.91 -1.44 -0.80
```

- rnorm(n) 产生 n 个标准正态分布随机数。

```
    round(rnorm(5), 2)
```

```
## [1]  1.25  0.77 -0.22 -0.42 -0.42
```

R的随机数函数（二）

- 每一种分布都有自己的名字，在其前面添加如下的字母分别代表不同的功能：
  - `p`：probability，分布函数
  - `q`：quantile, 分位数
  - `d`：density, 概率密度函数
  - `r`：random，随机数
- 查看**R**中支持的概率分布

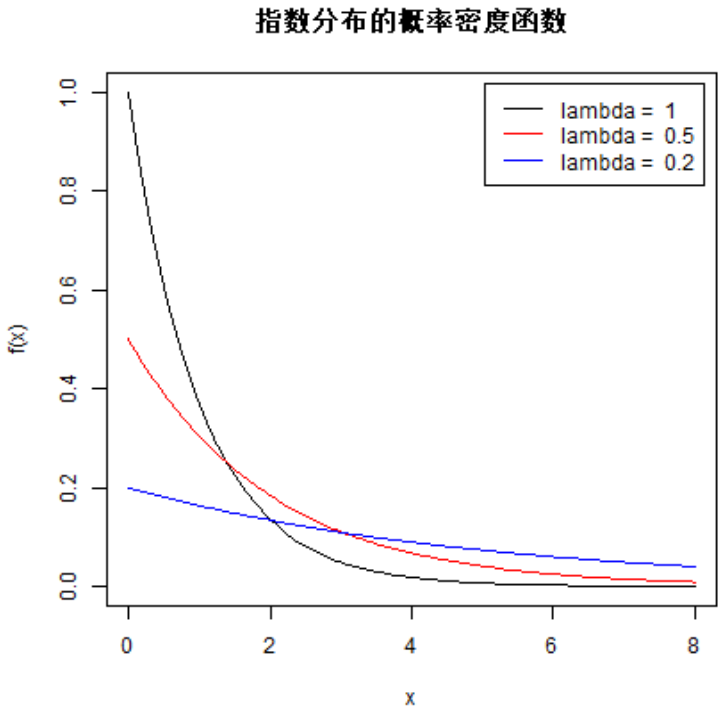
```
?Distributions
```

常用分布表

Distribution	cdf	parameter
二项分布	pbinom	size, prob
卡方分布	pchisq	df
指数分布	pexp	rate
F分布	pf	df1, df2
伽玛分布	pgamma	shape, rate or shape
几何分布	pgeom	prob
对数正态分布	plnorm	meanlog, sdlog
负二项分布	pnbinom	size, prob
正态分布	pnorm	mean, sd
Poisson分布	ppois	lambda
t 分布	pt	df
均匀分布	punif	min, max

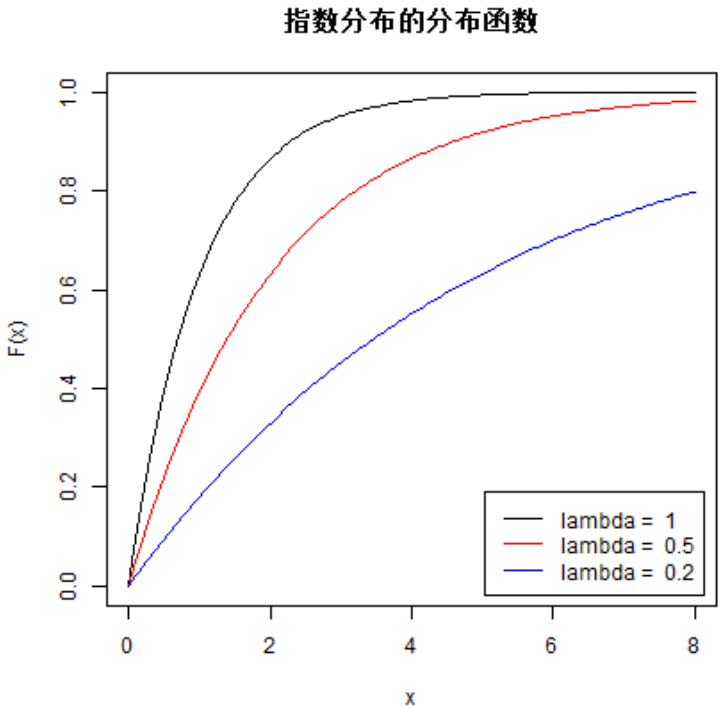
例子：指数分布的概率密度函数

```
x <- seq(0, 8, .05)
plot (x, dexp(x), ty="l", main="指数分布的概率密度函数", xlab="x", ylab="f(x)")
lines (x, dexp(x, rate=0.5), col="red")
lines (x, dexp(x, rate=0.2), col="blue")
legend("topright", legend = paste("lambda = ", c(1, 0.5, 0.2)), col=c("black", "red", "blue"),
```



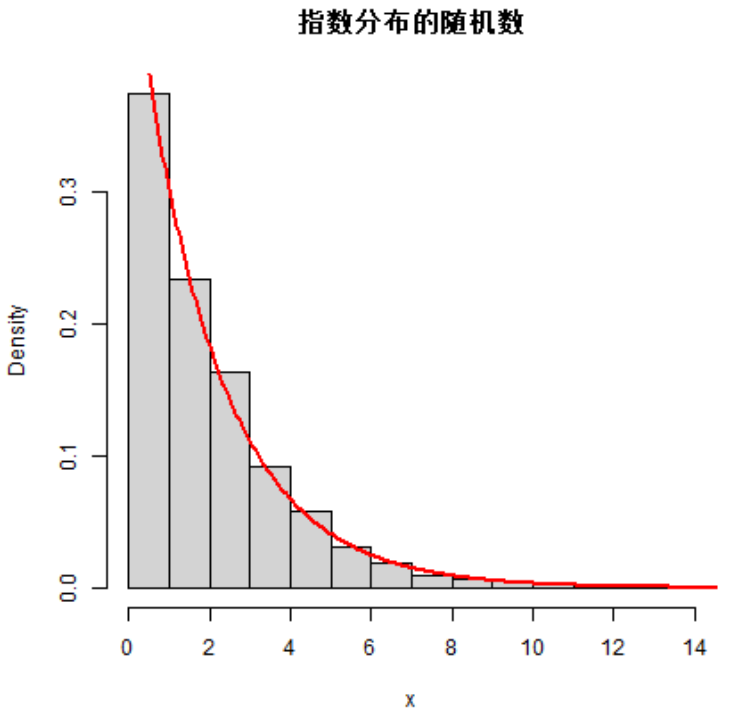
例子：指数分布的分布函数

```
x <- seq(0, 8, .05)
plot (x, pexp(x), ty="l", main="指数分布的分布函数", xlab="x", ylab="F(x)")
lines (x, pexp(x, rate=0.5), col="red")
lines (x, pexp(x, rate=0.2), col="blue")
legend("bottomright", legend = paste("lambda = ", c(1, 0.5, 0.2)), col=c("black", "red", "blue"
```



例子：指数分布的随机数

```
x <- seq(0, 16, .05)
hist(rexp(1000, 0.5), freq = FALSE,  xlab="x", main="指数分布的随机数")
lines (x, dexp(x, 0.5), col="red", lwd=2)
```



例子：大数定律

```
for(n in 2^c(1:20)){
  x <- rnorm(n, 0, 1)
  cat(abs(mean(x) - 0), "\n")
}
```

```
## 0.9632246
## 0.03062479
## 0.3383492
## 0.1319352
## 0.1006042
## 0.05585653
## 0.1118207
## 0.05027381
## 0.02897839
## 0.02046115
## 0.003602319
## 0.01314447
## 0.003361388
## 0.009055725
## 0.01005668
## 0.004535021
## 0.002457723
## 0.001133291
## 0.0003931444
## 0.0001127032
```

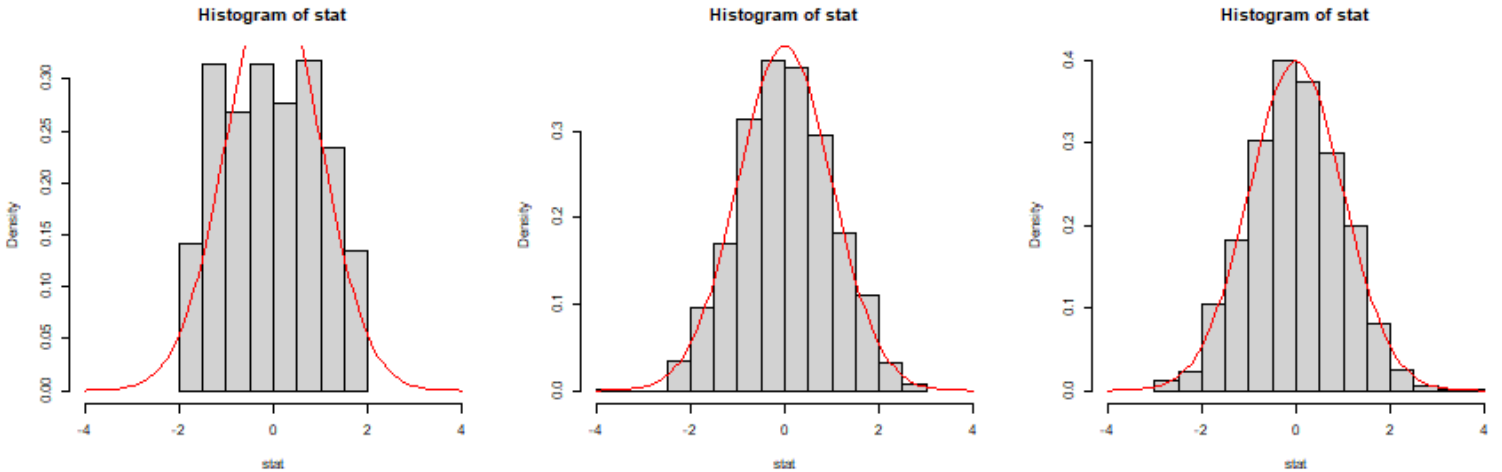
```
for(n in 2^c(1:20)){
  x <- rt(n, df = 1)
  cat(abs(mean(x) - 0), "\n")
}
```

```
## 0.4587143
## 0.06919593
## 2.901435
## 2.706213
## 0.730981
## 3.416372
## 1.008312
## 2.697938
## 0.3737971
## 6.013019
## 0.1435414
## 0.2877687
## 0.6573467
## 0.43103
## 0.8191331
## 0.1633895
## 60.54722
## 1.301394
## 0.8898035
## 8.422956
```

例子：中心极限定理

```
iter <- 1000

op <- par(mfrow=c(1, 3))
for(n in c(1, 10, 100)){
  x <- replicate(iter, mean(runif(n)))
  stat <- (x-0.5)*sqrt(12*n)
  hist(stat, freq = FALSE, xlim = c(-4, 4))
  curve(dnorm(x),  col="red", add = TRUE)
}
```




```
par(op)
```

谢 谢