

精算建模：Basic R

张连增 庄源

南开大学精算学系

2023-09-05



1. 课程开始之前：作业与考勤

2. R 是什么？

3. 新手上路

4. R 数据结构与基本绘图

5. 逻辑判断与循环

6. 概率分布

7. 结语



注册南开邮箱

- ✎ 学生南开邮箱注册
- ✎ 南开邮箱可用于收取平时成绩



平时作业与考勤细节

- 讲完每一章后的下一次课上课前提交作业（纸质版）
 - 提交到讲台上即可
 - 若迟交作业，作业分打八折（即本次作业最终得分为实际得分的 80%）
 - 作业可手写，也可使用 \LaTeX 排版打印
- 助教在批改完后会将作业参考答案发至微信群中
 - 纸质版作业批改完后发回给同学们
- 如有任何身体不适或因事缺勤，请提前向助教请假

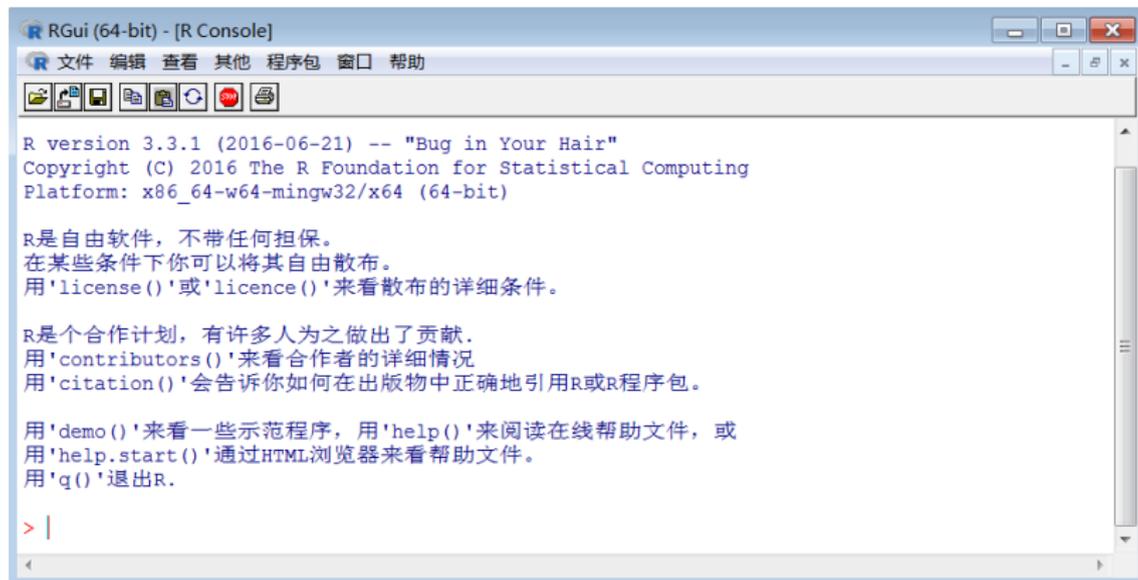


R 简介

- ⇒ R 是用于数据分析、统计分析的语言和操作环境，最早来源于 S 语言；
- ⇒ R 是一个免费开源软件，它有 Linux、MacOS 和 Windows 版本，都可以免费下载和使用；
- ⇒ R 语言具有丰富的网上资源和教材，搜索一下你就知道；
- ⇒ 最新的统计领域成果很多都在使用 R 语言编程（如 Journal of Statistical Software）；
- ⇒ 众多的画图功能；
- ⇒ 统计、金融工程、精算论文与教材附有 R 代码。



R GUI (Graphic User's Interface)



```
RGui (64-bit) - [R Console]
文件 编辑 查看 其他 程序包 窗口 帮助
R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R是自由软件，不带任何担保。
在某些条件下你可以将其自由散布。
用'license()'或'licence()'来看散布的详细条件。

R是个合作计划，有许多人之为之做出了贡献。
用'contributors()'来看合作者的详细情况
用'citation()'会告诉你如何在出版物中正确地引用R或R程序包。

用'demo()'来看一些示范程序，用'help()'来阅读在线帮助文件，或
用'help.start()'通过HTML浏览器来看帮助文件。
用'q()'退出R。

> |
```

图 1: RGUI (看起来不像是个 21 世纪的软件!)



Rstudio

The screenshot displays the RStudio environment with several key components:

- Code Editor (R脚本编辑区):** Contains R code for loading packages (RColorBrewer, tm, NLP, tm), processing a crude oil dataset, and creating a word cloud. A red circle highlights the **Run** button.
- Environment Pane (环境变量):** Lists loaded objects such as `cement.eigen`, `cement.k`, `china`, `choosePeo`, `c1`, `control`, and `crude`.
- Console (R命令控制台):** Shows the execution of `inspect` and the output of `data(crude)`, `crude <- tm_map(crude, removePunctuation)`, and `crude <- tm_map(crude, function(x) removeWords(x, stopwords()))`.
- Word Cloud (结果展示):** A visualization of words from the processed crude oil data, with prominent terms like `production`, `reuter`, `crude`, `industry`, and `accord`.

图 2: RStudio (欢迎来到 21 世纪!)



简单四则运算

```
x <- 1 ; y <- 2 # 赋值语句使用 <-, 快捷键是 ALT + -  
x + y
```

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

```
x - y
```

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

```
x * y
```

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

```
x / y
```

```
## [1] 0.5
```



别用交互命令啦！ 让我们把代码存起来

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

(D. Donoho)

- 新建 R 文件
- 新建 Rmd 文件



R Packages

- ⇒ 包是 R 函数、数据、预编译代码以一种定义完善的格式组成的集合
- ⇒ 一个包可以实现一些特殊功能，如 t 检验、回归或机器学习等
- ⇒ 截一个你的图放到微信群中，看看 R 赞美了你什么？

```
# 可以使用 install.packages 安装包
```

```
# 双引号中填写包名
```

```
install.packages("praise")
```

```
# 引入这个包
```

```
library(praise)
```

```
# 包中有一个函数
```

```
praise()
```

```
## [1] "You are dazzling!"
```



标量：数值型

```
a <- 2L ; b <- 2.26
```

```
is.numeric(a) # 查看一个变量是否为数值型变量
```

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

```
is.integer(a) # 查看一个变量是否为整数型变量
```

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

```
class(a)
```

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

```
class(b)
```

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



有奖练习 (2 Opportunities)

⇒ 输入以下代码：

```
c <- 2
```

⇒ 判断 `c` 是整数吗？把你的代码截屏于微信群中。



数值型标量的其它运算

```
# a <- 2 ; b <- 2.26
```

```
b ** a # 乘方
```

```
## [1] 5.1076
```

```
b %/% a # 整除
```

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

```
b %% a # 余数
```

```
## [1] 0.26
```



数值型标量的其它运算 (Cont'd)

```
# b <- 2.26  
ceiling(b) # 向上取整
```

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

```
floor(b) # 向下取整
```

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

```
round(b, digits = 1) # 四舍五入
```

```
## [1] 2.3
```



标量：文本型

```
t <- "ABCD"  
class(t)
```

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

```
nchar(t) # 计算文本的长度
```

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



标量：日期

```
date1 <- as.Date("2023-09-05")  
class(date1)
```

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

```
# 需要表示更为复杂的时间，可以使用 as.POSIXct 函数  
date2 <- as.POSIXct("2023-09-05 17:42")  
class(date2)
```

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

- 更为复杂的日期操作需要使用 `lubridate` 包和 `chron` 包，此处不涉及



标量：逻辑型

```
k <- TRUE
```

```
class(k)
```

```
## [1] "logical"
```

```
# TRUE 和 FALSE 在数值上是 1 和 0
```

```
TRUE * 5; FALSE * 5
```

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

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

```
T; F # TRUE 和 FALSE 可简写为 T 和 F
```

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

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



标量：逻辑型 (Cont'd)

⇒ 逻辑型变量经常在变量比较时产生

```
2 == 3 # 2 等于 3 吗?
```

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

```
2 != 3 # 2 不等于 3 吗?
```

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

```
2 <= 3 # 2 小于等于 3 吗?
```

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

```
"math" < "actuary" # "math" 小于 "actuary" 吗?
```

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



向量

⇒ 可以直接使用 `c()` 新建一个向量

```
x <- c(1, 2, 3, 4, 5)
```

⇒ 可以简单操作生成一些有规律的向量

```
1:10
```

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

```
seq(from = 2, to = 8, by = 2) # 2 到 8, 公差为 2 的等差数列
```

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

```
seq(from = 0, to = 10, length = 5) # 生成 0-10 间均匀的 5 个数
```

```
## [1] 0.0 2.5 5.0 7.5 10.0
```



向量 (Cont'd)

 针对标量的运算可以直接应用于向量的所有元素

```
x * 3
```

```
## [1] 3 6 9 12 15
```

```
sqrt(x) # 开平方
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```



向量 (Cont'd)

⇒ 向量之间可以直接做四则运算

```
x <- 1:10; y <- -5:4 # 当两个向量长度一致时，可对应位置直接加或
x + y
```

```
## [1] -4 -2 0 2 4 6 8 10 12 14
```

```
v1 <- c(4,6,8,10); v2 <- c(10,2) # 当两个向量长度不一致……
v1 + v2
```

```
## [1] 14 8 18 12
```



向量索引

⇒ 向量索引从 1 开始

```
x <- c(1, 2, 3, 4, 5)
```

```
x[1]
```

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

```
x[-1] # 除了 1 以外的元素
```

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

```
x[c(1,4)] # 第 1 和第 4 个元素
```

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

```
length(x)
```

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



向量的统计操作

```
x <- c(1, 2, 3, 4, 5, 8)
```

```
mean(x)
```

```
## [1] 3.833333
```

```
max(x)
```

```
## [1] 8
```

```
median.default(x)
```

```
## [1] 3.5
```

```
summary(x)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  1.000  2.250   3.500   3.833  4.750   8.000
```



数据框 (Dataframe)

```
x <- 1:4
y <- -4:-1
q <- c("Hockey", "Football", "Baseball", "Curling")
theDF <- data.frame(x, y, q) # 重点看这一行
theDF
```

```
##   x  y      q
##  1  1 -4  Hockey
##  2  2 -3 Football
##  3  3 -2  Baseball
##  4  4 -1   Curling
```



指定 Dataframe 的列名

```
theDF <- data.frame(First=x, Second=y, Sport=q)
theDF
```

```
##      First Second   Sport
## 1         1     -4 Hockey
## 2         2     -3 Football
## 3         3     -2 Baseball
## 4         4     -1  Curling
```



Dataframe 的参数

```
nrow(theDF) # 行数
```

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

```
ncol(theDF) # 列数
```

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

```
dim(theDF) # 维度
```

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

```
rownames(theDF) # 行名
```

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

```
colnames(theDF) # 列名
```

```
## [1] "First" "Second" "Sport"
```



Dataframe 索引：列名索引

```
theDF$Sport
```

```
## [1] "Hockey" "Football" "Baseball" "Curling"
```

```
theDF["First"]
```

```
##      First
## 1         1
## 2         2
## 3         3
## 4         4
```



Dataframe 索引：数字索引

```
theDF[3, 2] # 第 3 行第 2 列
```

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

```
theDF[1:3, 1] # 第 1-3 行, 第 1 列
```

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

- ⇒ 取出第 3 行，怎么做？
- ⇒ 同时取出第 2 列和第 3 列，怎么做？



矩阵：生成

```
A <- matrix(1:10, nrow=5) # 5 行 2 列  
C <- matrix(21:30, nrow=2) # 2 行 5 列
```



矩阵：运算

- ⇒ 相同维度的矩阵可以加减乘除
- ⇒ 真正意义的矩阵乘法是下面这个

```
# C: 2 行 5 列; A: 5 行 2 列
C %*% A
```

```
##      [,1] [,2]
## [1,]  395 1020
## [2,]  410 1060
```

- ⇒ 矩阵索引和 Dataframe 差不多，这里不赘述



有奖练习 (3 Opportunities)

⇒ 当不明白一个代码是什么时，可以使用`?`，带上你想询问的代码，如：

```
?data.frame
```

⇒ 试着找出 `t()`、`solve()` 和 `eigen()` 在对矩阵做什么操作？



使用数据集——鸢尾花

```
iris.new <- iris[,1:3] # 仅选取前三列
head(iris.new) # 展示前几个数据
```

```
##      Sepal.Length Sepal.Width Petal.Length
## 1           5.1           3.5           1.4
## 2           4.9           3.0           1.4
## 3           4.7           3.2           1.3
## 4           4.6           3.1           1.5
## 5           5.0           3.6           1.4
## 6           5.4           3.9           1.7
```

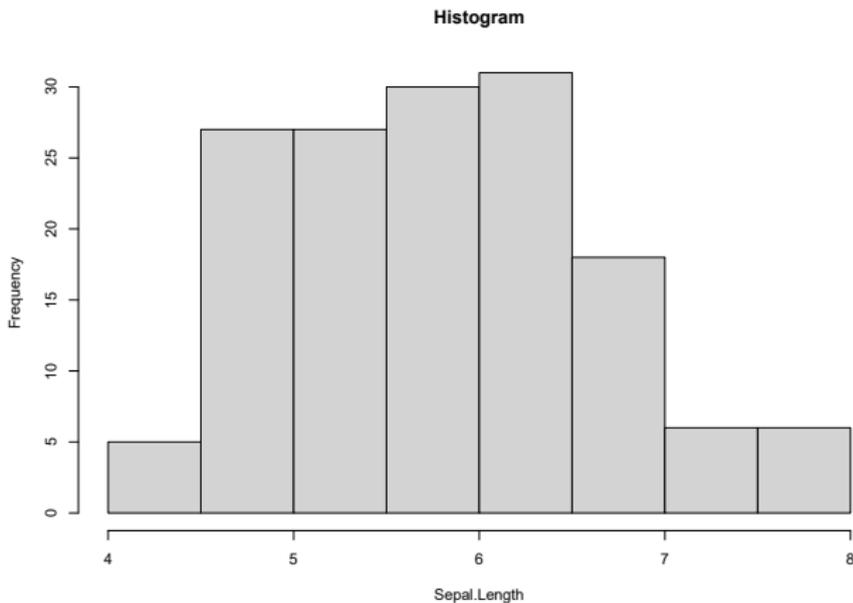
```
colnames(iris.new) # 看看有些什么变量
```

```
## [1] "Sepal.Length" "Sepal.Width"   "Petal.Length"
```



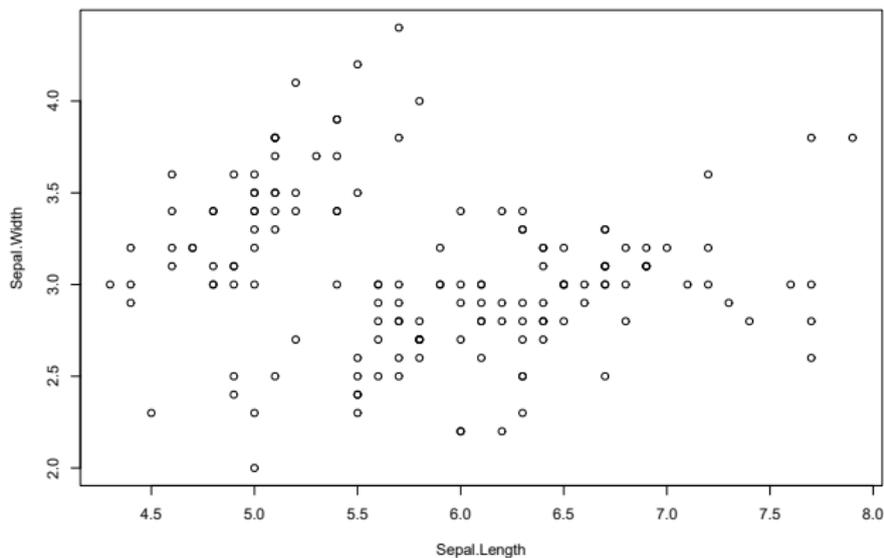
直方图

```
hist(iris.new$Sepal.Length, main="Histogram",  
     xlab="Sepal.Length")
```



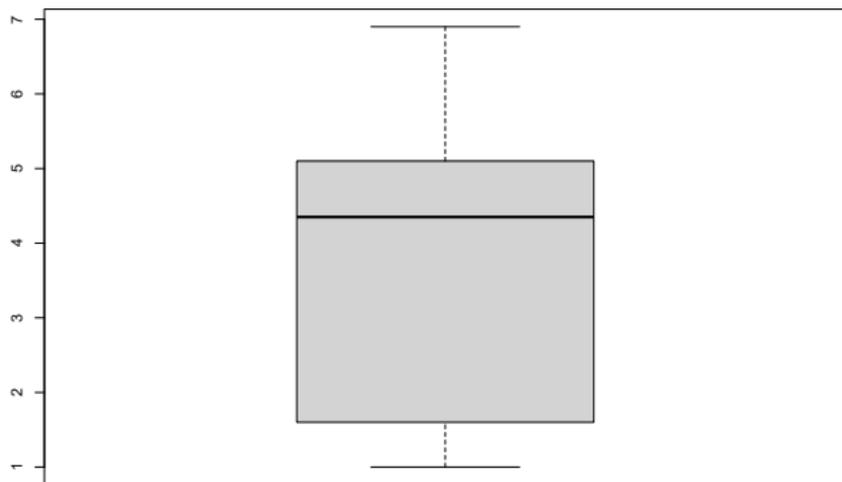
散点图

```
plot(iris.new$Sepal.Length,iris.new$Sepal.Width,  
     xlab="Sepal.Length",ylab="Sepal.Width")
```



箱线图

```
boxplot(iris.new$Petal.Length)
```



有奖练习 (5 Opportunities)

- ⇒ R 语言内置了一个数据集，名为 `cars`；
- ⇒ 绘制 `cars` 中，以 `speed` 为 x 轴、`dist` 列为 y 轴的散点图；
- ⇒ 图的标题为 “Cars+ 自己名字的拼音”，如 “Cars Yuanzhuang”；
- ⇒ 散点为实心圆形（提示：可以在 `plot` 函数里加上 `pch = 19`）；
- ⇒ 我要把点染成绿的！（提示：可以在 `plot` 函数里设置 `col` 这个参数）



if-else 语句

```
Check <- 2
if(Check == 1){ # 如果 Check 是 1……
  print("Hello")
}else if(Check == 0){ # 要如果 Check 是 0……
  print("Bye")
}else{ # 如果是其它情况……
  print("Confused")
}
```

```
## [1] "Confused"
```



简单的 ifelse 函数

```
toTest <- 1  
ifelse(toTest == 1, "Yes", "No")
```

```
## [1] "Yes"
```

```
toTest <- c(1, 1, 0, 1, 0, 1)  
ifelse(toTest == 1, "Yes", "No") # ifelse 函数还可以对向量判断
```

```
## [1] "Yes" "Yes" "No" "Yes" "No" "Yes"
```



逻辑上的交、并、非

- 多个条件之间可用 `&`、`|` 连接，表示“与”和“或”；
- `!` 加在一个条件前，表示“非”



for 循环

```
for(i in 1:5){  
  print(i)  
}
```

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



while 循环

```
x <- 1 # 设定 x 的初值
while(x <= 5){
  print(x)
  x <- x + 1
}
```

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



next 和 break (有奖练习: 3 Opportunities)

- next: 跳过当次循环, 开始下一次循环;
- break: 跳出所有循环;
- 使用 for 循环语句从 1 开始打印到 15
 - 跳过 3 不打印
 - 当打印完 10 后, 直接跳出循环, 打印你的名字拼音



四类函数

- ⇒ d= 密度函数 (f)
- ⇒ p= 累计分布函数 (F)
- ⇒ q= 分位数函数
- ⇒ r= 生成随机数

标准正态分布在 1.645 上的 F 是多少？

```
pnorm(1.645, mean = 0, sd = 1)
```

```
## [1] 0.9500151
```

标准正态分布的 97.5% 分位数是多少？

```
qnorm(0.975, mean = 0, sd = 1)
```

```
## [1] 1.959964
```



有奖问答 (Last Chance, 3 Opportunities)

- ⇒ 在命令行中键入以下代码：

```
RSiteSearch("Density of the Lomax Distribution")
```

- ⇒ 尝试寻找计算 Lomax 分布的密度函数所需的包，并使用 `library()` 加载这个包；
- ⇒ 计算 `scale = 1`，`shape = 3` 时，Lomax 分布的密度函数，保留两位小数。



未来值得学习的知识

- ⇒ ggplot 画图
- ⇒ actuar 精算建模
- ⇒ tidyverse 进行数据分析
- ⇒ 多种多样的机器学习包
- ⇒ 我们的征途是星辰大海.....



感谢

 谢谢!

