第一章 R语言介绍(一)

马学俊(主讲) 杜悦(助教)

苏州大学

数学科学学院

https://xuejunma.github.io/

内容

- R是为统计分析、绘图和数据挖掘而产生的语言和环境，它由Ross Ihaka 和Robert Gentleman 开发，现在由“R开发核心团队”负责开发。它的老祖先是S语言(John Chamber 和他的同事，贝尔实验室)，通常用S语言编写的代码都可以不作修改的在R环境下运行
- R软件的主页是`https://www.r-project.org/`，您可体验一下R网址的朴实而华丽，平凡而伟大。如果您打算和它相识，那么这将是最明智的选择，因为它会让您知道什么是自由、什么是奉献、什么是“走自己的路让别人也走自己的路”。

R不仅仅是一个软件，更是一个学习方式和生活态度。学习它，不仅仅可以提高我们的业务水平或者学术水平，也让我们深深感受到自由而快乐，奉献的快乐。写一串代码实现一个算法或模型，写一个包实现一个想法，留下的不是代码而是对知识和生活的享受和快乐。下面，我们一起看看R的朋友圈：

- 体积小。它只几十MB
- 免费，任何人都可以下载并且使用。
- 学习的资源多，算法更新非常快。如：
  - `https://blog.rstudio.org/`
  - `https://www.r-bloggers.com/`
  - `http://www.jstatsoft.org/`

下载和载入R包

我们在它主页上下载R并安装。R包是R软件的细胞。

```
install.packages("quantreg")
```

安装quantreg。需要注意加载包并不意味着可以使用该包函数，此时必须使用

```
library(quantreg)
```

将quantreg载入运行环境。比如我们利用该包rq建立分位数模型。

```
library(quantreg)
quantreg::rq
```

A::B表示使用A包中的函数B。这个写法虽然比较麻烦，但有时很重要。因为有时不同包的函数名可能相同。

帮助文件和代码

?x可以查看函数的使用方法比如，我们可以是使用?mean查看mean函数的帮助手册。它主要包括以下内容：

Title	标题：主要函数的用途，如mean的标题是Arithmetic Mean
Description	描述：更加详细介绍函数的用途
Usage	用法：函数的使用方法，如mean的用法是mean(x, ...)
Arguments	参数设置：函数的参数进行怎么设置或者要求
References	参考文献：函数的参考文献。这一部分很重要，我们通过它可以看到该函数或实现的模型的参考文献，可以进一步了解函数或者模型的表达、求解等细节，有助于我们学习新的方法或者模型。
Examples	例子：给一个实际或模拟数据的实现

查看源代码

x查些函数源代码。有些函数可以看到，有些看不到。如mean看不到源代码，lprq 可以看到源代码。

```
> library(quantreg)
> mean
function (x, ...)
UseMethod("mean")
<bytecode: 0x0000000016cc6cd8>
<environment: namespace:base>
> quantreg::lprq
function (x, y, h, tau = 0.5, m = 50)
{
    xx <- seq(min(x), max(x), length = m)
    fv <- xx
    dv <- xx
<environment: namespace:quantreg>
```

除了上面这些还有：

`getwd()` ：显示当前工作目录

`setwd()` ：修改当前工作目录。比如，我们处理的数据文件都在E中myr文件中，我们可以设置`setwd("E:/myr")`

`source()` :运行R的脚码(Script)。比如运行E中myr文件中my.R脚码，我们可以用`source("E:/myr/my.R")`

对象

- R中常用的对象(objects)包括: 向量(vector), 因子(factor), 数组(array), 矩阵(matrix), 数据框(data frame), 时间序列(ts), 列表(list)等等.
- 所有的对象都有两个内在属性: 类型(mode)和长度(length).
- 类型是对象元素的基本种类, 常用的有四种: 数值型(numeric), 字符型(character), 复数型(complex)和逻辑型(logical)(FALSE或TRUE). 虽然也存在其它的类型(raw), 但是并不能用来表示数据
- 例如函数或表达式; 长度是对象中元素的数目. 对象的类型和长度可以分别通过函数mode和length 得到.

对象	类型	是否允许同一个对象中有多种类型?
向量	数值型, 字符型, 复数型, 或逻辑型	否
因子	数值型或字符型	否
数组	数值型, 字符型, 复数型, 或逻辑型	否
矩阵	数值型, 字符型, 复数型, 或逻辑型	否
数据框	数值型, 字符型, 复数型, 或逻辑型	是
时间序列(ts)	数值型, 字符型, 复数型, 或逻辑型	否
列表	数值型, 字符型, 复数型, 或逻辑型 函数, 表达式...	是

- 向量是一个变量, 其意思也即人们通常认为的那样;
- 因子是一个分类变量; 数组是一个k维的数据表; 矩阵是数组的一个特例, 其维数$k = 2$. 注意, 数组或者矩阵中的所有元素都必须是同一种类型的;
- 数据框是由一个或几个向量和(或)因子构成, 它们必须是等长的, 但可以是不同的数据类型;
- `ts`表示时间序列数据, 它包含一些额外的属性, 例如频率和时间;
- 列表可以包含任何类型的对象, 包括列表!
- 对于一个向量, 用它的类型和长度足够描述数据; 而对其它的对象则另需一些额外信息, 这些信息由外在的属性(attribute)给出.
- 例如我们可以引用这些属性中的dim属性, 其是用来表示对象的维数, 比如一个2行2列的的矩阵, 它的dim属性是一对数值[2,2], 但是其长度是4.例如

```
> x
C1  C2
R1   1   3
R2   2   4
>   attributes(x)
$dim [1] 2 2
$dimnames
$dimnames[[1]] [1] "R1" "R2"
$dimnames[[2]] [1] "C1" "C2"
>  attributes(x)$dim
> [1] 2 2
```

数据结构

在介绍数据结构(Data Structure)之前，我们先介绍数据类型(Data Type)，它是每个数据的所属的类型，基本的数据类型主要有四种：逻辑性(Logical)、整数型(Integer)、双精度型(Double)和字符型(Character)，其中整数型和双精度型合成数值型(Numeric)。还有两种不常见的类型：原型(Raw)和复数型(Complex)。

- 逻辑型只有两个取值，`TURE`(简称`T`)和`FALSE`(简称`F`)。

  `c(F,T)`

- 整数型取值是整数，一个整数的长度有4-byte或者23-bit(bit就只0和1)，它可以表示为

$$\sum_{i=1}^{32} x_i 2^{i-1} - 2^{31}$$

  其中$x_i \in \{0,1\}$(注意的是+和−各占一个bite)。

```
> #最大的整数
> max_inter <- .Machine$integer.max
> max_inter
[1] 2147483647
> #错误，因为已经是最大了
> max_inter + 1L
[1] NA
Warning message:
In max_inter + 1L : NAs produced by integer overflow
```

- 双精度型取值为实数，一般可以表示为

$$(-1)^{x_0}\left(\sum_{i=1}^{t} x_i 2^{-1} 2^k\right)$$

其中$x_i \in \{0,1\}$，k是一个整数，称为指数部分(Exponent)，x_0称为符号bit(Sign Bit)，$\sum_{i=1}^{t} x_i 2^{-1}$是significand。一个标准的双精度型可以有一个sign bit 、11 bit 和52 bits significand，组成。$2^{11} = 2048$，所以，Exponent可以从-1022取到1024

```
>    #最小的正双精度型取值
> .Machine$double.eps
[1] 2.220446e-16
> .Machine$double.base
[1] 2
>
> .Machine$double.min.exp
[1] -1022
> .Machine$double.max.exp
[1] 1024
```

- 字符型取值的是字符串。如

```
c("R","statistics")
```

- 原型是计算机识别的最直接的数据类型，它是以二进制形式保存。如

```
raw(2)
```

- 复数型取值是复数。如

```
 3 + 5i
```

- 判断数据类型，可以使用mode和typeof(思考这两个命令的区别)。
- 逻辑性(Logical)、整数型(Integer)、双精度型(Double)和字符型(Character)四种数据类型与运算符号一样，也有高低。
- 我们可以用is.x看是否是x类型(如as.integer)，用as.x将数据强制转化为x类型。但这种转化不能将含有字母的字符串的转化为整数型。
- 大多数数学函数(+,- log)会将数据转化为整数型或双精度型；大多数逻辑运算符(&,any)会将数据转化为逻辑型。
- 几种数据结构之间的级别

```
> x_int <- c(1L, 2L)
> mode(x_int)
[1] "numeric"
> typeof(x_int)
[1] "integer"
> x_dou <- c(1, 2)
> x_log <- c(T, F)
> as.integer(x_log)
[1] 1 0
> x_cha <- c("R","S")
> as.integer(x_cha)
[1] NA NA
Warning message:
NAs introduced by coercion
> x_cha1 <- c("1","2")
> as.integer(x_cha1)
[1] 1 2
> as.integer(x_dou)
[1] 1 2
```

向量

经常使用c()创建，如

```
#数值
a <- c(1, 2, 5, 3, 6, -2, 4)
#字符
b <- c("one", "two", "three")
#逻辑
c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE)
#下标
c[1]
c[-1]
```

矩阵

dim rownames, colnames查看维度，行和列的变量名

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
       dimnames = NULL)
> 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
C1 C2
R1 1 26
R2 24 68
>mymatrix[,1]
```

数组

```
array(data = NA, dim = length(data), dimnames = NULL)
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
, , C1

   B1 B2 B3
A1  1  3  5
A2  2  4  6
```

数据框

colnames ,rownames，cbind,rbind查看列，行变量名，合并列和行。

```
data.frame(col1, col2, col3,···)
```

```
> age <- c(25, 34, 28, 52)
> diabetes <- c("Type1", "Type2", "Type1", "Type1")
> status <- c("Poor", "Improved", "Excellent", "Poor")
> patientdata <- data.frame(age, diabetes, status)
> patientdata
  age diabetes    status
1  25    Type1      Poor
2  34    Type2  Improved
3  28    Type1 Excellent
4  52    Type1      Poor
```

因子

```
x <- factor(c("a", "b", "b", "a"))
x
#> [1] a b b a
#> Levels: a b
class(x)
#> [1] "factor"
levels(x)
#> [1] "a" "b"
# You can't use values that are not in the levels
x[2] <- "d"
#> Warning: invalid factor level, NA generated
x
#> [1] a <NA> b a
#> Levels: a b
```

怎样将添加因子水平d

读入txt 和csv格式数据

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".",
           stringsAsFactors = default.stringsAsFactors())

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", )
```

- sep separation 的简写，表示文件的分隔符。默认是空格，我们可以设置为逗号。
- header 取值为T表示第一行为变量名
- stringsAsFactors 默认TRUE将字符型数据转化为因子变量。

```
read.fwf(file, widths, sep="\t", as.is = FALSE,
skip = 0, row.names, col.names, n = -1, ...)
```

widths有时很有用。

数据

```
A1.501.2
A1.551.3
B1.601.4
B1.651.5
C1.701.6
C1.751.7
> read.fwf("data.txt",  widths=c(1,  4,  3))
```

xlsx格式

xlsx数据，需要下载安装xlsx包，read.xlsx可以读入。

```
read.xlsx(file, sheetIndex, sheetName=NULL, rowIndex=NULL,
          startRow=NULL, endRow=NULL, colIndex=NULL)
```

file 指定的工作博，一种是数字形式指标第几个工作簿；另一种指定工作簿的名字

rowIndex，colIndex 指定读取行和列

startRow，endRow 没有指定rowIndex时，指定读入数据的行范围

SAS,SPSS Stata格式

`foreign`包:

read.ssd	SAS数据
read.spss	SPSS数据
read.dta	Stata数据

数据导出

```
write.table(x, file = "")
```


hán lù
寒露
@木壳人
weibo.com/sunguoyan1
木壳人