

# R in Action

---

## 简介

- `example("foo")` 函数foo的使用示例
- `apropos("foo",mode="function")` 列出名称中含有foo的所有可能函数
- `data()` 列出当前加载包中所含的所有可用示例数据集
- `vignette()` 列出当前安装包中所有可用的vignette文档
- `vignette("foo")` 为主题foo显示指定的vignette文档
- `history(#)` 显示最近使用过的#个命令
- `.libPaths()` 显示当前库所在位置
- `browseURL()` 网页浏览
- `download.load()` 下载文件到本地
- `dir.create()` 新建文件
- `list.files/list.dirs(path=".",all.files=F/T,full.names=F/T,recursive=F/T)` 显示目录下文件或目录下文件夹
- `file.create()` 文档创建
- `file.exists()` 判断文档是否存在
- `file.remove()` 文档删除
- `file.rename(from,to)` 文档重命名
- `file.append(file1,file2)` 文档添加
- `file.copy(from,to,recursive=F/T)` 文档复制
- `file.symlink(from,to)` 文档链接
- `file.show()` 显示文档内容
- `file.info()` 显示文档信息
- `file.edit()` 编辑文档
- `zip()/unzip()` 压缩/解压缩文档; `zip("plot.rdata.zip", "plot.rdata", extra=c("-r","-e"))`, enter password: verify password:
- `get()` 返回命名了的对象的值
- `attributes(my.group)` 产看一个对象所包含信息
- `download.file(url,destfile,method)`
- `update.packages()`
- `help(summarize, package="plyr")`
- `plyr::summarize()`
- `BiocManager::install("packages")`
- 代码(code): `print("hello, R!")`
- github安装: `install.packages("devtools"); devtools::install_github("tidyverse/tidyr")`
- 列举包中的函数及数据集 `ls("package:GO.db")`
- 查看包中所有函数 `help help(package="GO.db")`
- `Sys.getenv("HOME")`, 获得用户home目录
- `Sys.getenv("R_HOME")`, 获得R的home目录:
- `install.packages("./GO.db_3.3.0.tar.gz",type="source",repos = NULL)`安装本地下载包
- `View()` 以可修改表格形式查看对象, invoke a Data Viewer

## 特殊字符

NA 缺失值或为定义数据

NULL 空对象(null/empty lists)

Inf/-Inf 正或负无限值

NaN 不能定义结果

is.na()/is.finite()/is.nan()

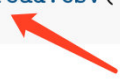
(v1>2) | (v2>3) '|' boolean OR, 返回向量

(v1>2) & (v2>3) '&' boolean AND, 返回向量

(v1>2) || (v2>3) '||' boolean OR, 返回单个值

(v1>2) && (v2>3) '&&' boolean AND, ditto

```
## read.csv is only executed if the file exists
file.exists("data.csv") && read.csv("data.csv")
#> [1] FALSE
```



However care must be taken not to use && or || on vectors since it will give the incorrect answer

```
x < 0.4 || x > 0.6
#> [1] TRUE
```



library(igraph)/detach(package:igraph)

## 创建数据集

- 向量

1. a <- c(1,2,5,3,6,-2,4)
2. b <- c("one","two","three")
3. c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)

- 矩阵

1. y <- matrix(1:20,nrow=5,ncol=4) 5x4矩阵
2. cells <- c(1,26,24,68)  
rnames <- c("R1","R2")  
cnames <- c("C1","C2")  
mymatrix <- matrix(cells,nrow=2,byrow=TRUE,dimnames=list(rnames,cnames))

- 数组

1. 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))

- 数据框

1. `patientID <- c(1,2,3,4)`  
`age <- c(25,34,28,52)`  
`diabets <- c("Type1","Type2","Type1","Type1")`  
`status <- c("Poor","Improved","Excellent","Poor")`  
`patientdata <- data.frame(patientID,age,diabetes,status)`
2. `patientdata[1:2]` or `patientdata[c("diabetes","status")]`
3. `attach()`将数据框添加到R搜索路径, `detach()`移除
4. `with(mtcars,{`  
`summary(mpg,disp,wt)`  
`plot(mpg,disp)`  
`})` 赋值仅在扩号内生效, 使用`<-`可将对象保存在`with()`外的全局环境中
5. `within(leadership,{`  
`agecat <- NA (not a available)`  
`agecat[age > 75] <- "Elder"`  
`agecat[age >75 & age <= 75] <- "Middle Aged"`  
`agecat[age <75] <- "Young"}`)
6. `transform(airquality,Ozone=-Ozone)`  
`transform(airquality,new=-Ozone,Temp =(Temp-32)/1.8)`
7. `subset()` 选入变量和观测  
`subset(leadership,gender=="M" & age > 25, select=gender:q4)`
8. `rename(dataframe,c(oldname="newname"...))`
9. `is.na()` 检测每个变量 / `na.omit()` 删除行/`na.rm=TRUE` 函数选项;置换data.frame中的NA:  
`dat[is.na(dat)] <- 0`
10. `which(x)` 给出对应逻辑值"TRUE" / "FALSE"  
`which((1:12)%2 == 0)` 输出能整除2的数的index
11. `order()` 默认升序, 得到排序后值位置index  
`leadership[order(leadership$age),]`  
`a <- c(3,4,5,1,2,3,9,3,0,2,0)`  
`a[order(a)]` 0 0 1 2 2 3 3 3 4 5 9
12. `sort(x, decreasing=F,...)` 将当前值从小到大排列
13. `reorder(x,X,FUN)` 第一参数为被排序的因子序列, 第二个参数是等长度的作为排序参考的序列, 第三个是构造排序标准的加工函数
14. `rev(x)`反向排序当前x向量顺序
15. 合并数据, `merge(dataframeA,dataframeB,by=c("ID","Country"))`
16. 选入数据, `dataframe[row indices,colnum indices]`
17. 删除数据`%in%`

```
myvars <- names(leadership) %in% c("q3","q4")
newdatas <- leadership[!myvars]
leadership$q3 <- leadership$q4 <- NULL(为定义)
```

#### 18. sample() 随机抽样

```
leadership[ sample(1:nrow(leadership),3,replace=F),]
```

```
1 data <- data.frame(
2     ticker=character(),
3     value=numeric(),
4     date = as.Date(character()),
5     stringsAsFactors=FALSE
6 )
7
8 # here read the contents of files into the dataframe
9 for (file in dir())
10 {
11     stocks <- read.csv2(file, sep = " ", header = FALSE)
12     stocks$date <- sub(".csv", "", file)
13     names(stocks)<- c("ticker", "value", "date")
14     data <- rbind(data, stocks)
15 }
```

先定义列宽度，再一次填充！！

#### • 列表

1. g <- "my first list"

```
h <- c(25,26,18,39)
```

```
j <- matrix(1:10,nrow=5)
```

```
k <- c("one","two","three")
```

```
mylist <- list(title=g,ages=h,j,k)
```

2. mylist[[2]] or mylist[["ages"]]

3. bash下标从0开始，awk下表从1开始，R下标从1开始，Python下标从0开始

#### • 数据输入

1. read.table(file,header=TREU/FALSE,sep=" ",stringsAsFactors=FALSE)

2. read.table(file,header=TRUE/FALSE,sep=" ",colClasses=c(logical, numeric,character,factor))

3. library(readr), read\_delim 读取文本，注意最后的空行会读取进入将命名为X行号—NA，可转换为 data.frame, na.omit=TRUE

4. gzfile(),bzfile(),xzfile(),unz() 读取压缩文件

5. 导入excel，library(xlsx),library(openxlsx),library(readxl)

6. library(xlsx), read.xlsx()

#### • 值标签

1. patientdata\$gender <- factor(patientdata\$gender,levels=c(1,2),labels=c("male","female"))  
levles代表了实际值，labels代表包含了理想值标签的字符向量

2. str(),class(),dim(),mode(),cbind(),rbind()

3. newobject <- edit(object) 编辑对象并另存为新object

fix(object) 直接编辑对象

#### • 因子

1. `factor(x, levels=sort(x,decreasing=F)/levels=rev(x))`
2. 将数字专为factor后反转需要: `as.numeric(as.charactor(x))`
3. `levels(factor)[factor]`

## 基本数据管理

- 符号,逻辑及转换

1. `+, -, *, /, ^` or `**`
2. `x%%y`, 求余; `x%/y`, 整数除法
3. `<, <=, >, >=, ==, !=, !x, x|y, x&y, isTRUE(x)`
4. `is.numeric()`, `is.character()`, `is.data.frame()`, `is.factor()`, `is.logical()`
5. `as.data.frame()`

- 日期

1. `date()`, 返回时间日期, `Sys.Date()` 日期
2. `as.Date(x, "input_format")`  
`%d 01~31; %A Monday; %m 00~12; %B January; %Y 2007`  
`mydates <- as.Date(c("2007-06-22", "2004-02-13"), "%Y-%m-%d")`
3. `format(Sys.Date(), format="%B %d %Y")`
4. `difftime()` 计算时间间隔  
`difftime(Sys.Date, as.Date("1956-1-22"), units="weeks")`
5. `strDates <- as.character(dates)` 将日期转为字符型

- 数学函数

1. `abs()`, `sqrt()`, `ceiling(x)`, `floor(x)`, `trunc(x)`, `round(x, digits=n)`
2. `log(x, base=n)`, `log(x)`, `log10(x)`, `exp(x)` 指数函数
3. `quantile(x, probs)` `probs[0,1]` 求分位数  
`quantile(x, c(0.3, 0.84))`
4. `scale(x, center=T, scale=T)` 将对象列中心化(`center=T`)或标准化(`center=T, scale=T`) 默认值  
`scale(mydata)*SD + M` 默认标准差为1, 均值为0

- 概率函数

1. `d=`密度函数(`density`); `p=`分布函数(`distribution function`); `q=`分位数函数(`quantile function`); `r=`生成随机数(随机偏差)
2. `nbinom`, `norm`, `pois`, `unif` 负二项式分布, 正态分布, 柏松分布, 均匀分布
3. `plot(rnorm(100), dnorm(rnorm(100)))` 绘制分布函数图
4. `set.seed(num)` 设立随机种子, 实现结果重复  
`set.seed(1234); runif(5, min=0, max=1)`

- 字符处理函数

1. `nchar(x)` 字符, `length(x)` 长度

2. `substr(x, start, stop)` 提取或替换一个字符向量的子串
3. `grep(pattern, x, ignore.case=FALSE, value = FALSE, fixed=FALSE)` 在x中搜索某种模式。fixed=F, pattern为正则。返回匹配下标, invert=T, 表示反向查询。
4. `grepl(pattern, x, ignore.case=FALSE, perl=FALSE, fixed=FALSE...)`返回TRUE/FALSE
5. `regexpr(pattern, text, ignore.case=FALSE, perl=FALSE, fixed=FALSE)`在文本中搜索, 返回第一个匹配的位置, 未匹配返回-1
6. `sub(pattern, replacement, x, ignore.case=F, fixed=F)` 在x中搜索pattern, 并以文本replacement将其替换。fixed=T, pattern为文本字符:

```
sub("(. *treated).*", "\\1", sampleFiles)
```

7. `strsplit(x, split, fixed=F)` 在split处分割字符向量x中的元素。若fixed=F, 则pattern为正则。返回一个list。 `unlist(y)[2]`, `sapply(y, "[", 2)`
8. `paste(..., sep="")` `paste("x", 1:3, sep="")`; `paste("x", 1:3, sep="m")`
9. `paste(list[[1]], collapse=",")` 变为长度为1的标量
10. `seq(from, to, by)` `seq(1, 10, 2)`  
`seq.int(from, to, by)`  
`seq_along(leadership)`  
`seq_len(leng.out)`
11. `rep(x, n)` `rep(1:3, 2)` `c(1, 2, 3, 1, 2, 3,)`  
`rep(1:4, each=2) == rep(1:4, c(2, 2, 2, 2))`  
`rep(1:4, each=2, len=4)`  
`rep_len(1:3, 8)` 1, 2, 3, 1, 2, 3, 1, 2
12. `tcut(x, n)` 将连续变量x分割为有着n个水平的因子, `ordered_result=T`创建一个有序因子
13. `pretty(x, n)` 创建美观分割点, 通过选取n+1个等间距的取整值, 将一个连续变量x分割为n个区间
14. `cat(..., file="my file", append=F)` 链接...中对象, 并将其输出到屏幕上或文件中(如果声明了一个的话) `cat("Hello", "Jane", "\n")`
15. `apply(矩阵或数据框, MARGIN, FUN, ...)`  
`apply(mydata, 1, mean, trim=0.2)`  
`sapply()`返回向量, `lapply()`返回列表, 将函数应用到list上  
`sapply(x, FUN, options)` `sapply(name, "[", 1, na.omit=T)`

#### ● 控制流

1. `for(var in seq) statement`  
`for(i in 1:10) print("Hello")`  
`next` 跳出本次循环  
`break` 跳出本层循环, 当存在多个for循环, 即跳出最近一个)
2. `while(cond) statement`  
`i <- 10; while(i>0){print ("Hello"); i<- i-1}`

```
3. x <- 1;
  repeat{
    print(x)
    x=x+1
    if(x == 6){
      break}
  }
```

```
4. if(cond) statement
   if(cond) statment1 else statement2
   ifelse(cond, statement1, statement2)
   outcome <- ifelse(score > 0.5, print("Passed"),print("Failed"))
   if(cond) {statements} else if(cond) {statements}
```

**The double equals operator can't tolerate an `NA` on either side. If I define: `x = NA` and then do an `if (x == NA){ ... }` then this error will be thrown at runtime when the parser examines the left hand side of the double equals. To remedy this error, make sure every variable in your conditional is not NA using `is.na(your_variable)`**

```
5. switch(expr,...)
   feelings <- c("sad","afraid")
   for(i in feelings){
     print(
       switch(i,
         happy="I am glad you are happy",
         afraid="There is nothing to fear",
         sad="Cheer up",
         angry="Calm down now"
       )
     )
   }
```

```
6. function(arg1,arg2,...){
   statements
   return(object)
}
```

- 整合和重构

1. `t(mtcars)` 将矩阵或数据框进行转置
2. `aggregate(x,by,FUN)` `x`为待折叠数据对象, `by`是一个变量组成列表, 将这些列表变量通过FUN变为新的观测 `aggregate(mtcars, by=list(mtcars$cyl,mtcars$gear), FUN=mean, na.rm=T)`
3. `library(reshape2)`

```
melt(data, ..., na.rm=F)
```

`melt(mydata, id=c("id","time"))` 融合，根据id对其重新排列  
`cast(md, rowvar1+rowvar2 + ... ~ colvar1+ colvar2+... , FUN)`

`rowvar1+rowvar2 + ... ~ colvar1+ colvar2+...` 定义了要选入的变量集合，以确定各行内容

`dcast(data,formula=Var1~Var2,fun.aggregate=mean,margins=NULL)`解析melt的融合后的数据

## Miscellaneous

1. `rm(list=ls()[! ls() %in% c("find_the_lost_sample","data_list","lab_samples")])` 删除不想要的对象
2. `library(plyr); round_any(x,X,FUN); round_any(24.534, 10, ceiling) [1]30; round_any(24.534, 10, floor) [1] 20`
3. `2:7 %o% 1:2 == outer(2:7,1:2)`

---

## plyr & dplyr

**summarize()**是对数据做汇总, 不同于**mutate**, 它不会增加列到现存数据集, 它创造一个新的数据框

```
library(plyr)
```

```
library(ggplot2)
```

```
summarize(mpg, a=quantile(displ), b=quantile(displ))
```

```
ddply(mpg, .(manufacturer), summarize,q=quantile(displ))
```

**mutate()**类似**transform()**, 但是它能迭代使用新生成的列信息来产生新的列信息

**transform**可行

```
mutate(airquality, Ozone=-Ozone)
```

```
mutate(airquality, new=-Ozone, Temp=(Temp - 32)/1.8)
```

**transform()**不可行

```
mutate(airquality, Temp=(Temp-32), OzT=Ozone/Temp) #存在迭代使用第一次计算得到的Temp
```

**transmute()**增加新的数据同时丢弃原有数据, 而**mutate()**保留原有数据同时增加新的数据

```
mtcars %>% mutate(displ_1=displ/61)
```

```
mtcars %>% transmute(displ_1=displ/61)
```

**group\_by()**要搭配**summarize()**使用, 对现有数据根据变量进行分组

```
mpg %>% group_by(manufacturer,cyl) %>% summarize(m=min(displ),n=max(displ)) ##返回  
data.frame, 对应分组后的m和n
```

```
mpg %>% summarize(m=min(displ),n=max(displ)) ##返回两个值m和n
```

**filter()**选择条件为真的行内容(row/case)

```
== > >= & | ! xor() is.na() between() near()
```

```
filter(starwars, mass>7)
```



```
starwars %>% filter(mass > mean(mass, na.rm = T))
```

```
starwars %>% group_by(gender) %>% filter(mass > mean(mass, na.rm = TRUE))
```

### **arrange()**根据变量对数据排序

```
arrange(mtcars, mpg, disp)
```

```
arrange(mtcars, desc(mpg))
```

### **select()**选择或重命名变量名, **select()**仅保留所选的变量; **rename()**保留所有变量

starts\_with() ends\_with() contains() matches() num\_range() one\_of() everything() group\_cols()

```
select(mtcars, cyl:wt)
```

```
select(mtcars, -cyl) #符号表删除
```

---

## tidyr

### **gather(data, key="Key", value="value", ..., na.rm=F, convert=F, factor\_key=F)**

将多重列转成key-value对的结构, key和value都是新生成列的名字; 后接一段列名称, 空置表示全选; x: y一段范围; -y排除y列, 具体规则同select()函数。

```
> billboard %>% gather(week, rank, wk1:wk76, na.rm=T) %>%  
mutate(week=extract_numeric(week), date=as.Date(date.entered) + 7 * (week - 1)) %>%  
select(-date.entered) %>% arrange(artist, date, rank)
```

### **spread(data, key, value, fill=NA, convert=F, drop=T, sep=NULL)**

将key-value对分不到多个列中, 其中key为拆分为新的列, value的值对应到新的列下。

```
weather3 %>% spread(element, value)
```

### **separate(data, col, into, sep, remove=F...)**

针对给定的正则表达式或字符向量位置将col列转化into成多个列, 其中sep为字符串时表正则表达式, 为数字时, 表示col分开的位置。

```
> tb2 %>% separate(demo, c("sex", "age"), 1)
```

```
separate_row(data, ..., sep="[\\^[:alnum:]]+", convert=F)
```

当一个列的值包含多个限定的值, 将其分开到当前列的不同行中

```
separate_rows(table3, rate)
```

### **unite(data, col, ..., sep="\_", remove=T)**

将多个列合并到一个列

```
unite(table5, century, year, col="year", sep="")
```

### **unique(x, incomparables=F, ...)**

unique返回一个向量, 数据框或数组, 对应的重复的元素或行都被删去

```
> billboard3 %>% select(artist, track, year, time) %>% unique() %>%  
mutate(song_id=row_number())
```

## Tibbles

增强型数据结构，新的S3数据结构用于存储表格型数据，具有以下优点: subsetting，一贯返回一个新的tibble, 而[[和\$返回向量；no partial matching，当行subsetting时，需使用全列名称；display，当屏幕打印tibble，输出一个匹配屏幕的简洁视野。

**View(), glimpse():** 查看全部数据

**as.data.frame():** 转成data frame

**as\_tibble(x, ...):** 将数据框转成tibble

**enframe(x, name="name", value="value"):** 将名称向量转成tibble

**is\_tibble(x):** 检查是否为tibble

## CONSTRUCT A TIBBLE IN TWO WAYS

**tibble(...)**

Construct by columns.

*tibble(x = 1:3, y = c("a", "b", "c"))*

**Both  
make this  
tibble**

**tribble(...)**

Construct by rows.

*tribble(~x, ~y,  
1, "a",  
2, "b",  
3, "c")*

```
A tibble: 3 × 2  
      x     y  
  <int> <chr>  
1     1   a  
2     2   b  
3     3   c
```

**as\_tibble(x, ...)** Convert data frame to tibble.

**enframe(x, name = "name", value = "value")**  
Convert named vector to a tibble

**is\_tibble(x)** Test whether x is a tibble.

## Strings

函数str\_开头，将向量作为第一个参数

```
x <- c("why", "video", "cross", "extra", "deal", "authority") str_length(x)
```

```
[1] 3 5 5 5 4 9
```

```
str_c(x, collapse = ", ")
```

```
[1] "why, video, cross, extra, deal, authority"
```

```
str_sub(x, 1, 2)
```

```
[1] "wh" "vi" "cr" "ex" "de" "au"
```

```
str_subset(x, "[aeiou]")
```

```
[1] "video" "cross" "extra" "deal" "authority"
```

```
str_count(x, "[aeiou]")
```

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

```
str_detect(x, "[aeiou]")
```

```
[1] FALSE TRUE TRUE TRUE TRUE TRUE
```

```
str_count(x, "[aeiou]")
```

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

```
str_subset(x, "[aeiou]")
```

```
[1] "video" "cross" "extra" "deal" "authority"
```

extract the characters on either side of the vowel

```
str_match(x, "(.)([aeiou])(.)")
```

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

```
[1,] NA NA NA
```

```
[2,] "vid" "v" "d"
```

```
[3,] "ros" "r" "s"
```

```
[4,] NA NA NA
```

```
[5,] "dea" "d" "a"
```

```
[6,] "aut" "a" "t"
```

```
str_replace(x, "[aeiou]", "?")
```

```
[1] "why" "v?deo" "cr?ss" "?xtra" "d?al" "?uthority"
```

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

```
[[1]]
```

```
[1] "a" "b"
```

```
[[2]]
```

```
[1] "c" "d" "e"
```



antigenome  
新药研发

16 人赞同了该回答

苹果用户需要五行代码解决问题：

```
install.packages('devtools') #assuming it is not already installed
library(devtools)
install_github('andreacirilloac/updateR')
library(updateR)
updateR(admin_password = 'Admin user password')
```

安装完成后能看到

```
everything went smoothly
open a Terminal session and run 'R' to assert that latest version was instal
```

从[stackoverflow](#)上找到的解决方法[Update R using RStudio](#)



知乎用户  
腾讯科技（深圳）有限公司 数据分析师

59 人赞同了该回答

如果你是在windows环境下，可以安装一个包：installr。不仅能帮你更新R，还能将原来安装的扩展包配置到新版本下，不用再重新一一安装。

```
library(installr)
updateR()
```

发布于 2015-12-18

▲ 赞同 59



● 17 条评论

➦ 分享

★ 收藏

❤ 感谢

## Efficient R Programming

- Startup Files

`.Renviron` 为环境变量。所包含的设置和操作系统相关，包含查询额外的程序的路径，和用户指定的变量

`.Rprofile` 为文本文件，是R每次启动时运行的R codes。

当R启动时，首先搜索 `.Renviron`，然后搜索 `.Rprofile` 文件。可通过 `R --no-environ` 取消执行 `.Renviron` 文件

startup files存在与3个重要路径：`R_HOME` (`R.home()`)，R安装路径；`HOME` 为用户home路径；R当前的工作路径(`getwd()`)。

```
> R.home()
[1] "/Library/Frameworks/R.framework/Resources"
>
> setwd(R.home())
> getwd()
[1] "/Library/Frameworks/R.framework/Versions/3.6/Resources"
>
> list.files()
 [1] "bin"          "COPYING"      "doc"          "etc"          "fontconfig"
 [6] "include"      "Info.plist"   "lib"          "library"      "man1"
[11] "modules"      "R"           "Rscript"      "share"        "SVN-REVISION"
[16] "tests"
```

可在project的根目录创建该project特异性的 `.Rprofile` 文件：

```
file.edit(file.path("~", ".Rprofile")) # edit .Rprofile in HOME
file.edit(".Rprofile") # edit project specific .Rprofile
```

该例子在用户home路径创建了 `.Rprofile` 文件，`.Renviron` 同之。

- `.Rprofile` file

```
# A fun welcome message
message("Hi Robin, welcome to R")
# Customise the R prompt that prefixes every command
# (use " " for a blank prompt)
options(prompt = "R4geo> ")
# Don't convert text strings to factors with base read functions
options(stringsAsFactors = FALSE)
```

`.Rprofile`和`.Renviron`中的内容仅且在每个R session中运行一次，因此其中选项可以在随后的操作中修改；例如修改换行带来的+号：`options(continue=" ")`

通过查看例子和文档了解设置：`help("Startup")`，`?options`

**miscellaneous**，`fortunes`包提供名言警句，因此可在每次开始一个session时打印一次：

```
## fortunes~
if(interactive()){~
  try(fortunes::fortune(),silent=TRUE)~
}~

## .Last functions runs at the end of the session~
.Last=function(){~
  cond=suppressWarnings(!require(fortunes,quietly=TRUE))~
  if(cond){~
    try(install.pacakges("fortunes"),silent=TRUE)~
    message("Goodbye at",date(),"\n")~
  }~
}
```

其中 `interactive` 用于检测R是否在一个交互式终端使用，使用 `try` 调用 `fortune` 函数。若 `fortunes` 不可用，则避免报告错误，继续接下来操作。通过使用 `::` 可避免将 `fortunes` 包添加到 attached 包列表。

最后，使用 `require` 装载包，若未安装，则 `require` 函数返回 `FALSE` 和警告信息。

在 `.Rprofile` 中添加 'helper' 函数或重新定义现存函数以便快速访问

```
## useful functions predefined or redefine existing ones~
## ht == headtail~
ht=function(d,n=6) rbind(head(d,n),tail(d,n))~
## hh show the first 5 row & 5 columns of a data frame~
hh=function(d) d[1:5,1:5]~
~
## a function for setting a nice plotting window~
#setnicepar=function(mar=c(3,3,2,1),mgp=c(2,0.4,0),tck=-0.01,cex.axis=0.9,~
#.....las=1,mfrow=c(1,1),...){~
# par(mar=mar,mgp=mgp,tck=tck,cex.axis=cex.axis,las=las,mfrow=mfrow,...)~
#}~
## set the abbreviated form for the frequently used function~
v=utils::View~
~
```

在 `.Rprofile` 中创建隐藏环境，当使用 `ls()` 时，`.Rprofile` 中的 functions 会显示，使得当前工作环境杂乱(clutter-up)；同时当 `rm(list=ls())` 时，`.Rprofile` 中 functions 也将会被删除。使用隐藏的对象和环境来解决该问题

```
## creating hiding environments with .Rprofile~
## beside of the objects, also could create a environment~
.env=new.env()~
.env$ht=function(d,n=6) rbind(head(d,n),tail(d,n))~
## attach to makes it possible to refer to objects in the environment by~
## their names alone~
attach(.env)~
~
```

- `.Renvron` file

用于保存系统变量。和 `.Rprofile` 一样，先查看全局 `.Renvron` 文件，然后查看本地版本。

使用 `.Renvron` 用于指定 `R_LIBS` 路径，决定了新的 packages 将会安装在哪里

```
> Sys.getenv("R_HOME")
[1] "/Library/Frameworks/R.framework/Resources"
> Sys.getenv("HOME")
[1] "/Users/carlos"
> R.home()
[1] "/Library/Frameworks/R.framework/Resources"
```

使用函数 `Sys.getenv()` 查看所有当前环境变量设置

使用 `Sys.setenv()` 设置或取消当前环境变量

```
Sys.setenv("TEST" = "test-string") # set an environment variable for the session
Sys.unsetenv("TEST") # unset it
```

**miscellaneous**, `TMPDIR=/data/R_tmp`，当 R 运行时，构建临时拷贝文件

`R_COMPILE_PKGS=3` byte compile all packages

`R_LIBS_SITE=/usr/lib/R/site-library:/usr/lib/R/library`，查询 packages 的路径

`R_DEFAULT_PACKAGES=utils,grDevices,graphics,stats,methods` 需要载入的默认包

- Efficient programming



Speeding up code using the **compiler** packages and multiple **CPUs**

**R编程的金标准是使可能快的方式来访问潜在的C/Fortran规则，越少的函数操作越好。**

比较三种构建向量方式

```
method1 = function(n){vec = NULL;for(i in 1:n)vec=c(vec,i);vectorized}
```

```
method2 = function(n){vec=numeric(n);for(i in 1:n)vec[i]=i;vec}
```

```
method3 = function(n)1:n
```

Table 3.1: Time in seconds to create sequences. When  $n = 10^7$ , method 1 takes around an hour while the other methods take less than 3 seconds.

$n$	Method 1	Method 2	Method 3
$10^5$	0.21	0.02	0.00
$10^6$	25.50	0.22	0.00
$10^7$	3827.00	2.21	0.00

**Remember the golden rule: access the underlying C/Fortran code as quickly as possible**

输入输出格式一致性: **Type consistency: when programming it is helpful if the return value from a function always takes the same form**

并非所有的R函数都遵循输入输出格式一致性, `sapply` 和 `[.data.frame]` 不能实现一致性, 函数 `lapply` 和 `vapply` 可以实现一致性。

当我们创建函数时, 常希望函数针对当前情况给出反馈信息, 例如, 是否存在缺失的参数, 或者数字计算失败等。

致命错误: **stop**

当调用函数 `stop` 时, 执行终止, 此时函数无法继续执行。

```
try(expr, silent=FALSE, outFile=getOption("try.outFile",default=stderr()))
```

用于回收错误信息

```
> good <- try(1+1,silent=T)
> bad <- try(1+"1",silent=T)
> good
[1] 2
> bad
[1] "Error in 1 + \"1\" : non-numeric argument to binary operator\n"
attr(,"class")
[1] "try-error"
attr(,"condition")
<simpleError in 1 + "1": non-numeric argument to binary operator>
```

因此可以根据返回值设置下一步操作

```
if(class(bad) == "try-error") Do something
```

警告错误: **warning**

使用 `warning` 函数即可生成警告信息。当警告产生时，表明出现潜在的问题。例如，`mean(NULL)` 返回 `NA` 同时还有警告信息

```
> mean(NULL)
[1] NA
Warning message:
In mean.default(NULL) : argument is not numeric or logical: returning NA
>
```

使用 `suppressWarnings(mean(NULL))` 隐藏警告信息

```
> suppressWarnings(mean(NULL))
[1] NA
> suppressWarnings(mean(NULL))
[1] NA
> |
```

信息类输出：**message** 和 **cat**

包中存在输出有用信息的 `message` 函数，类似警告信息，可使用 `suppressMessages` 抑制该信息输出；另一个就是 `cat` 函数，同 `print/show` 方式


不可见返回值：`invisible` 函数允许用户返回一个不可见的对象拷贝值。This is particularly useful for functions that return values which can be assigned ,but are not printed when they are not assigned.

```
> regression_plot = function(x,y,...){
+   plot(x,y)
+   model = lm(x~y)
+   abline(model)
+   invisible(model)
+ }
> regression_plot(1:10,(1:10)^2 +9)
> dev.off()
null device
      1
> out=regression_plot(1:10,(1:10)^2 +9)
> out

Call:
lm(formula = x ~ y)

Coefficients:
(Intercept)          y
      1.39874      0.08634

> |
```



当调用该函数时，图形中拟合线不可见，但是当将函数assign给一个值时，该值当输出将包含 `lm` 拟合曲线公示值

因子：**Factors**

因子可用于存储分类变量。分类变量常保存为数字，且数字对应有其解释。

例如：**Months of the year**



```
> m <- c("January", "December", "March")
> fac_m <- factor(m, levels=month.name)
> sort(fac_m)
[1] January March December
12 Levels: January February March April May June July August ... December
> |
```

例如: **Graphics**

同months, 避免默认根据字母顺序排序

```
boxplot(y ~ type)
boxplot(y ~ factor(type, levels=c("Small", "Medium", "Large")))
```

例如: **data input**

使用 `read.csv()` 读取文件时, 列的字符自动转换为factors, 可通过使用 `stringsAsFactors=FALSE` 避免

闭包: **Function closures**

```
> simple_counter = function(){
+   no = 0
+   function(){
+     no <- no+1
+     no
+   }
+ }
> sc = simple_counter()
> sc()
[1] 1
> sc()
[1] 2
```

`simple_counter` 函数返回一个函数

`sc` 的闭包环境不是 `.GlobalEnv`, 而是 `sc` 的绑定环境

`sc` 拥有一个可以存储/缓存值的环境

`<-` 操作符用于改变 `sc` 环境中的对象 `no` 的值

- Efficient workflow

Importing data: 通过下载或使用 `read.csv` 直接打开

```
url <- "https://www.monetdb.org/sites/default/files/voc_tsvs.zip"
download.file(url, "voc_tsvs.zip")
unzip("voc_tsvs.zip", exdir="data")
file.remove("voc_tsvs.zip")
##
url <- "https://www.osha.gov/dep/fatcat/FatalitiesFY10.csv"
df <- read.csv(url)
```

Fast data reading: 基本的R 文本读取函数 `read.delim`; 使用函数 `fread` 的 `data.table` 途径; 提供 `read_csv` 等 `read_*` 函数的 `readr` 包

针对小于1MB的文件, `read.delim` 快与 `read_csv` 和 `fread`, 而对于大于100MB的文件, `fread` 和 `read_csv` 远快与 `read.delim`

Data processing with dplyr: `dplyr`; `%>%`

```
idata %>%
  filter(grepl("g", Country)) %>%
  group_by(Year) %>%
  summarise(gini = mean(gini, na.rm = TRUE)) %>%
  arrange(desc(gini)) %>%
  top_n(n = 5)
#> Selecting by gini
#> Source: local data frame [5 x 2]
#>
#>   Year  gini
#>   (int) (dbl)
#> 1  1980  46.9
#> 2  1993  46.0
#> 3  2013  44.5
#> 4  1981  43.6
#> 5  2012  43.6
```

Data processing with data.table: `data.table`

- Efficient visualisation
- Efficient base graphics
  - Set up par function
  - Custom palettes
- ggplot2
- Interactive graphics
  - plotly, shiny, leaflet, htmlwidgets
- Efficient performance

```
ifelse(test, yes, no)
```

```
sort() / order()
```

```
sort(t, decreasing=TRUE) == rev(sort(x))
```

```
which(x == min(x)) == which.max/which.min
```

```
as.numeric(levels(f))[f] == as.numeric(as.character(f))
```

 去因子水平

```
is.na() / anyNA()
```

 查询是否含有缺失值

Parallel computing: `parallel` 包

```
library("parallel")
```

```
no_of_cores = detectCores()
```

```
lapply, sapply, apply
```

```
parLapply(cl, x, FUN, ...)
```

```
parApply(cl = NULL, X, MARGIN, FUN, ...)
```

```
parSapply(cl = NULL, X, FUN, ..., simplify = TRUE, USE.NAMES = TRUE)
```

- Efficient Learning

```
?plot
```

```
example(plot)
```

```
??regression
```

```
help.search("regression")
```

## reading R source code

默认的下载地址都在国外，所以一般来说下载速度十分慢。但其实在国内都有相关的镜像，可以用来下载相关包

Rstudio直接在Tools-->Global Options-->Packages-->CRAN mirror 选项中选择一个距离你最近的国内镜像就行，如China (Shanghai) [https] - Tongji University。这样以后下载包就都从这个站点下载了

**Review the installed packages information** 使用 `library()` 查看已经安装包列表

使用 `installed.packages()` 查看各个包安装路径, 版本信息

也可使用 `.packages(all.available=T)` 在控制台现实已经安装包名字

卸载删除包, `remove.packages(c("pkg1", "pkg2"), lib=file.path("path", "to", "library"))`

---