R In Action Notes

Author : Yunqiu Xu

`R` `DataAnalysis`

- 4 Parts, 16 chapters:
- Introduction-getting data into R-visualisation-data managemen
- Basic tutorials : 1-6,11,16
- Advanced tutorials : 7,10
- Some difficult topics : 8,9,12-15

Part I Getting Started

Chapter 1 Introduction to R

1.1 Managing the workspace

```
getwd()  #工作区目录，相当于git中的"pwd"
setwd("C:/mydirectory")
#注意一点，R中"\"为转义，不可直接出现在路径内，需要改为"/"or "\\"
dir() #展示wd中的文件
file.exists("directoryName")/dir.create("directoryName") #查询并创建工作目录
```

- 一个栗子：

```
if(!file.exists("xyq"){ #如果不存在则返回TRUE
    dir.create("xyq")} #创建工作目录
```

- 常用函数

```
source("filename") #从某个文件中读取R编码，见后
list.files("C:\data") #类似ls()
ls() #类似git "ls"
```

```
options() #view or set current options
history(#) #查询命令历史，默认为前25个
savehistory("myfile") #将命令历史储存到myfile，载入用loadhistory("myfile")
save.image("");
save(objectlist,file="myfile"); load("myfile")
q() #退出R，退出前会询问是否保存
demo(graphics/Hershey/persp/image or none) #展示R自带的可视化类型
```

- 注意&为交集，返回向量，而&&则返回值

1.2 Getting help

```
help.start() #general help
?foo or help("foo") #get help for a function
example("foo") #the examples of the function you want to know
```

1.3 Packages

- Install packages:install+library+search

```
install.packages("package_name")
library(package_name) #when the installation is ended, you need to loa
d it
search() #you will find packages that have been loaded
help("package_name") #seekhelp
```

Chapter 2 Creating a datas

- R data structures
- data entry
- importing data
- annotating data
- Dataset：本书用observations/variables来称谓行或列
- 基本数据类型：

```
numeric#数字，整数或小数
integer#整形
```

```
character#字母
logical#逻辑正误
complex#复数
```

2.1 数据结构

- scalars标量,vectors向量,matrices矩阵,arrays数组,data frames数据框架,lists

2.1.1 向量vectors

```
a <- c(1,3,5,7,9) #使用函数c()来形成向量
#向量元素必须为同一种类型
#如果添加不同元素也可以，但是你class一下这个向量，会发现会被强制转换成一种类型
b <- c("one","two","three") #字母向量
c <- c(TRUE,FALSE,TRUE,TRUE) #逻辑向量
a[1] #向量a中第一个元素（这个跟python从零开始不同）
a[c(2,4)] #列出向量a中第二个和第四个元素
a[2:6] #列出向量a中第二个到第六个元素，a <-c(2:6) 等价于 a <- c(2,3,4,5,6)
#两个向量x/y可以加减乘除，元素一一对应
```

2.1.2 矩阵Matrices

```
myymatrix <- matrix(vector, #矩阵中元素
        nrow, #number of rows
        ncol, #number of columns
        byrow=logical_value, #矩阵排列形式，默认False按列排列，TRUE则为按行排列
        dimnames=list(char_vector_rownames, char_vector_colnames) #名称
        )
```

- 当然，也可以先把元素整出来，再给这些元素dim一下：

```
m<-1:10
dim(m)<-c(2,5)
```

- 再举个栗子：将1-20按列排列到5*4矩阵内，且规定矩阵的行名与列名

```
y<-matrix(1:20,
    nrow=5,ncol=4,
    dimnames=list(c("a","b","c","d","e"),
    c("大傻","二傻","三傻","四傻")))
y[2,] #矩阵y第二行元素
```

```
y[,2] #矩阵y第二列元素
y[2,c(3,4)] #矩阵y第二行第3,4个元素（相当于第二行与第三列第四列的交叉点）
```

- 两个类型相同的矩阵可以加减乘除，元素一一对应
- 矩阵乘法：`x %*% y`

2.1.3. 阵列Arrays

- 矩阵是特殊的阵列（只有二维），但阵列可以有多个维度

```
myarray <- array(vector,
        dimensions, #每个维度的最大数值,如c(3,4,5)
        dimnames=list()#每个维度的名称，行名/列名...
        )
```

- 举个栗子：建立一个三维（3*4*5）阵列

```
dim1=c("大傻子","二瘸子","三蹦子")
dim2=c("001","002","003","004")
dim3=c("d1","d2","d3","d4","d5")
y <-array(1:60,c(3,4,5),dimnames=list(dim1,dim2,dim3))
```

2.1.4 数据框架DataFrames——比矩阵更适用于R

- 与矩阵的不同之处在于不同列可以放置不同类型的数据

```
mydata <- data.frame(col1,col2,col3)
```

- 举个栗子：将姓名、年龄、学号、籍贯集合成数据框架

```
name <-c("大傻子","二瘸子","三蹦子")
age <-c(18,14,22)
studentID <-c(12306,57652,61365)
birthplace <-c("天津","黑龙江","广东")
mydata <- data.frame(name,age,studentID,birthplace)
mydata[2:4] #从第二列到第四列的元素
mydata[c("age","name")] #年龄与姓名信息
mydata$name #这是一种新的查询子集方法
```

- 举个栗子：创建年龄与籍贯的交叉分析表

```
table(mydata$age,mydata$birthplace)
```

```
attach/detach(mydata) #将数据框架引入/移出搜索路径
#后面学习时看到过，一般在最开始引入如attach(mtcars)
#attach/detach相当于创造了一个封闭空间，在这个空间里调用指定数据集子集不需要主数据集前缀
#attach后，之后再调用列不需要再标注mydata$
with(mydata,table(age,birthplace)) #另一种省略标注的办法，加with
#如果有多个表格，用{}括起来
#注意下with的限制，with里的生成任务只能在with括号里使用，如在外面使用age会显示错误
#解决措施，使用特殊的任务生成器<<-替代<-
with(mydata,xyq<<-table(age,birthplace))
```

- 建议使用with而不是attach
- 一些别的小tips

```
read.table()/read.csv() #这个不是创建，是从外部读取
data.matrix() #将数据框架转化为矩阵
nrow()/ncol()#行数与列数
```

2.1.5 系数Factors

- 一种特殊的整形向量，其中每个整形都带有一个标签
- factor要强于单单使用整型变量
- 例如使用标签"male/female"要强于整形1/2

```
bp=factor(birthplace,order=TRUE)
```

- 举个栗子：

```
#将逻辑分类的字符集组成一个向量
x<-factor(c("Y","Y","N","NG","Y","N","NG","Y","NG"))
table(x) #对不同的标签计数成表
unclass(x) #将不同标签转化为整形权重
#一般情况下，我们直接在factor函数里给出不同的level
#这在线型模拟中十分重要，因为第一个标签将作为基准物
levels=c("Y","N","NG")
```

2.1.6 Lists: 可添加不同类型的object，同数据框架比还能添加不同长度

```
mylist <- list(name1=object1,name2=object2,...)
```

- 举个栗子：

```
g<-"my first list"
h<-c(23,24,25,21)
j<-matrix(1:10,nrow=5,ncol=2)
k<-c("xyq1","xyq2","xyq3")
mylist=list(title=g, ages=h,j,k)
```

2.2 导入(读取)及写入数据

- 导入数据前需要考虑所需内存及硬盘的容量
- 所有的文件都要先存放到工作区目录中才能被找到,或者将你的工作区目录改到这个文件的所在地

2.2.1 键盘导入

```
fix()/edit() #编辑界面
```

2.2.2 从文本文件导入数据 read.table()/read.csv()

```
mydataframe <- read.table(
        file, #工作目录中的某个文件名，如xxx.csv
        header=TRUE, #第一行是否要包含变量名，TRUE/FALSE
        sep=",", #分割标点
        nrows=10 #读取10行
        row.names="name" #可选参数，选择一个或多个变量作为列标识符，类似SQL中的主键，例如"STUDENT'S ID"
        skip=8 #跳过前8行再开始读取
        na.strings="sb", #使用该字符指代NA
        )
```

2.2.3 从EXCEL导入数据

- xlsx package

```
library(xlsx)
read.xlsx("filename",
        sheetIndex=1,#which sheet do we want to store the data on
        header=TRUE,
        colIndex=2:4, #读取2-4列
        rowIndex=3:5, #读取3-5行
        )
```

- RODBC包: 安装并使用RODBC包install+library

```
channel <- odbcConnectExcel("xxx.xls")
mydataframe <- sqlFetch(channel,"mysheet")
odbcClose(channel) #也可以导入Access数据
```

2.2.4 从XML导入数据 : XML package

```
library(XML)
doc<-xmlTreeParse(fileUrl,useInternal=TRUE) #连接url
rootnode<-xmlRoot(doc) #get root 并进入url
xmlName(rootNode)/names(rootNode)
rootNode[[1]]#the 1st component of rootNode
rootNode[[1]][[1]] #进一步查询子集
xmlSApply(rootNode,xmlValue) #从XML文件中提取信息，类似爬虫！！
```

- 举个栗子：

```
xpathSApply(rootNode,"//name",xmlValue) #获取商品名
xpathSApply(rootNode,"//price",xmlValue) #获取价格
```

2.2.5 从网络爬虫webscraping导入数据

- 注意: python爬取网页的能力强于R
- 简单爬取网页

```
con=url("uelname") #connect to url
htmlCode=readLines(con) #get the data
close(con)
```

- 结合之前学的XML: xmlTreeParse+xpathSApply

```
library(XML)
doc<-xmlTreeParse(fileUrl,useInternal=TRUE)
xpathSApply(html,"//title",xmlValue)
```

- httr package

```
library(httr)
html2<- GET(url)
content2<- content(html2,as="text")
```

```
parseHTML<- htmlParse(content2,asText=TRUE)
xpathSApply(html,"//title",xmlValue)
```

- 对于有password的：

```
GET(urlname,authenticate("user","passwd"))
```

2.3.6 其他统计语言SAS/SPSS

- foreign package #可以将经其他统计语言获取的数据转换为R

```
read.xport(SAS)
read.spss(SPSS)
```

2.3.7 从MySQL导入数据: #RMySQL package

- 连接数据库并进行查询dbConnect+dbGetQuary+dbDisconnect

```
ucscDb<-dbConnect(MySQL(),user="XXX",
        host="xxx") #connect the database
result<-dbGetQuary(ucscDb,
        "show databases;")#query command
dbDisconnect(ucscDb) #解除连接
```

- 打开某个数据库

```
ucscDb<-dbConnect(MySQL(),user="XXX",
        db="hg19",host="xxx")
allTables<-dbListTables(hg19)
allTables[1:5]
```

- get dimensions of a table

```
dbListFields(hg19,"aircar") #给出aircar表中的所有列名
dbGetQuery(hg19,"select count(*) from airfly")
```

- 读取数据库中某个表dbReadTable

```
dbReadTable(hg19,"tablename")
```

- 选取某个子集

```
query<-dbSendQuery(hg19,"yourquery") #提交查询
ff<-fetch(query) #获取查询结果，滞后就可以对查询结果进行R操作
dbDisconnect(hg19)
```

- 其他DBMS:

```
RPostresSQL package
RODBC package #PostgreSQL/MySQL/Access/SQLite
RMongo package #MongoDb
```

2.3.8 JSON 导入数据: `jsonlite package`

```
library(jsonlite)
jsonData<-fromJSON("urlname")
jasonData$subset1$subset2 #逐层次查询子集
myjson<-toJSON(data,
        pretty=TRUE #自动缩进，美化格式
        ) #将数据写入为JSON形式
cat(myjson) #连接并print，如果有多个objects相当于paste，这里仅仅为print
myjson2<-fromJSON(myjson) #使用刚刚的fromJson()函数可以再将转化成JSON的数据转化回来
```

2.3.9 导入图片及音频

- 读取image数据：`jpeg/readbitmap/png/EBImage packages`
- 读取GIS数据：`rdgal/rgeos/raster packages`
- 读取music数据：`tuneR/seewave packages`

2.3.10 结合 JHU Data Science Series总结一下读取和写入的办法

```
read.table()/read.csv() #常规读取表格类文本类数据的方法
write.table
readLines #读取文本文件的行，可以用来爬网页
writeLines
source/dget #读取文件中的R编码
dump/dput
load #读取已经保存的workplace
save
unserialize #二进制形式读取R对象
serialize
```

- 这里着重讲一下source-dump；`dget-dput`，`dump/dput`最后生成的是textual format而

非一般的表格

- Textual Format的特点：
  - dumping/dputing的文本格式是可编辑的;
  - 同table/csv不同，dump/dput保存了元数据（牺牲部分可读性），不需要多次具体说明;
  - 文本格式与版本控制系统更兼容;
  - 文本格式可长期保存;
  - 文本格式的不足：对空间的利用效率低;
- 举一个dput+dget的栗子：适合写入单个对象的情况

```
y<- data.frame(a=1,b="a") #建立一个data frame
dput(y) #简单演示一下
dput(y,file="y.R") #将y写入y.R
new.y<-dget("y.R") #将读取y.R的结果赋值给new.y
```

- 举一个dumping+source的栗子：适用于多个对象

```
x<- "foo"
y<- data.frame(a=1,b="a")
dump(c("x","y"),file="data.R") #将对象x, y都写入data.R
rm(x,y) #删除x,y
source("data.R") #如果直接查询x,y会找不到因为已经被删掉了，但是source读取后又可以找到
```

2.3.12 连接文件或者网络connecting to file/gzip/bzip/url

- 连接网络：

```
file(description = "",
        open = "", #打开方式：r-reading/w-writing/a-appending;rb/wb/ab为以二进制编码形式操作
        blocking = TRUE,
        encoding = getOption("encoding"),
        raw = FALSE)
url(description, #同理不加赘述
        open = "",
        blocking = TRUE,
        encoding = getOption("encoding"),
        method)
gzfile(description, #同理不加赘述
        open = "",
```

```
          encoding = getOption("encoding"),
          compression = 6)
```

- 举个栗子：

```
connection<-file("foo.txt","r") #与foo.txt建立连接，操作方式为读取
data<-read.csv(connection) #读取数据
close(connection) #断开连接
#等价于
data<-read.csv("foo.txt") #在这个例子里，完全可以用一行代码来替代
```

- 再举个栗子，如果仅仅需要读取文件中的部分数据，使用file()更简单：

```
x<-readlines(connection,10) #仅仅读取前10行数据就够了
```

- 再最后举个栗子，使用connection+readLines连接并读取网页内容

```
con<-url("urlname") #连接网站
x<-readLines(con) #读取数据
head(x) #显示head数据
```

- 从网上下载数据：

```
download.file(url, #copy link address
        destfile="C:\data\shabi.csv", #save as
        method="curl",
        #下载方式，对于Mac打开https时需要特别说明，win则是默认的
        )
```

2.4 标注数据集

2.4.1 给变量加标签

```
names(patientdata)[2] <- "Age at hospitalization (in years)"
```

2.4.2 带值的标签: 使用`factor() function`

2.5 将R数据以文档/pdf/表格等形式输出

2.5.1 以R编码文档输出

```
dput(y,file="y.R") #将y写入y.R，使用dget()读取
dump(c("x","y"),file="data.R") #将多个对象写入data.R,使用source()读取
```

2.5.2 保存为文档

```
sink("123.txt") #创建一个文档
source("xxxx") #读取数据，或者用别的方法导入数据
sink() #保存
#另一种形式：read.csv()
write.csv(data, file="D:/bearf2.csv")
```

2.5.3 保存为pdf

```
pdf("mygraph.pdf")
#写入需要保存的内容
dev.off() #关闭图像生成函数，自动保存
```

Chapter 3 Getting started with graphs

- creating&saving graphs;
- customizing factors;
- annotating with text and titles;
- controling a graph's dimensions;
- combining mutiple graphs into one

3.1 基本步骤：

```
pdf("mygraph.pdf") #保存为pdf图像
attach(mtcars) #连接Motor Trend Car Road Tests数据集
plot(dose,drugA,type="b") #散点图，typeb意指点+线状图，p仅仅是点，l仅仅是线，别的可以参考help(plot)
abline(lm(mpg~wt)) #拟合
title("Regression of MPG on Weight") #add a title
detach(mtcars) #断开数据集
dev.off() #关闭图像生成函数
```

- 同时连接多个图表：

```
dev.new() #创造一个新图表
dev.new() #再创造一个新图表
```

- 一些其他用于操作图表的命令：

```
dev.next()
dev.prev()
dev.set()
dev.off()
dev.cur() #seek help
```

3.3 图表参数 par() function

```
opar <- par(no.readonly=TRUE)
```

- 使用par函数设置并查询图表参数

```
par(..., no.readonly = FALSE)
    #...表示所有类似于tag=value形式的参数
    #no.readonly默认为FALSE，如果为TRUE表示保存当前设置
par(lty=2,pch=17) #线型更改为曲线(lty=2)，并将点改为实心三角(pch=17)
#也可以分成两行写
plot(dose,drugA,type="b") #这里就和之前一样了
#也可以用另一种写法plot(dose, drugA, type="b", lty=2, pch=17)
par(opar)
```

3.3.1 symbols and lines

```
pch #点类型 P51图3.4 如之前的pch=17
cex #点大小，默认为1,1.5大50%，0.5小50%
lty #line_type
lwd #line_width，默认为1
```

3.3.2 colors col参数

```
col=c("red","blue") #点与线的颜色参数
#只有一条线，一种颜色,则整条线以及所有点都是红色的
#只有一条线，两种颜色，线线是一种颜色，点是两种交替
#两条线，两种颜色，一条线一种颜色
```

```
#多条线，两种颜色，交替
col=1
col="red"
col="#FFFFFF"
col=rgb(1,1,1)
col=hsv(0,0,1)
col.axis #坐标轴颜色参数
col.lab #坐标轴标识颜色参数
col.main #title
col.sub #subtitle
fg #前景
bg #背景
```

3.3.3 Text chrarcteristics

```
cex #可以使用之前控制图标大小的参数来控制字体大小
cex.axis
cex.lab
cex.main
cex.sub
font #字体类型，默认为1,2加粗，3斜体，4粗斜体，5Adobe默认编码字体
#同理axis/lab/main/sub

ps #字体大小，像素点表示，the text size=ps*cex
windowsFont(family) #字体集类型
windowsFont(
        A=windowsFont("字体A")
        B=windowsFont("字体B")
        ) #将字体进行赋值，调用更方便
#如果以pdf形式输出，可以直接改成pdf里的字体
names(pdfFonts()) #查找pdf字体
pdf(file="xyq.pdf",family="fontname") #将xyq.pdf以某个字体输出
```

3.3.4 Graph and margin dimensions

```
pin #控制维度(宽/高，单位为英寸)
mai #控制边距(上下左右，单位为英寸)
mar #和上面那个一样，只是单位为1/12英寸
```

- 举个栗子:

```
par(pin=c(4,3),mai=c(1,.5,1,.2))#4*3,上边距1以此类推
```

3.3.5 最后一个栗子，综合前面所有的图表设置参数

```
opar <- par(no.readonly=TRUE) #保存当前设置
par(pin=c(3,4)) #图表大小3*4

par(lwd=2,cex=1.5) #线宽2，点大小1.5

par(cex.axis=0.75,font.axis=3) #坐标轴字体大小0.75，斜体

plot(dose,drugA,type="b",pch=19,lty=2,col="red") #点+线，点型19，线型为2，红色

plot(dose,drugB,type="b",pch=23,lty=6,col="blue",bg="green") #点型23，线型6，蓝色，绿色背景
par(opar)
```

3.4 添加文本，个性化坐标轴

3.4.1 titles

```
title(main="",sub="",xlab="",ylab="") #也可以添加颜色，尺寸等参数
```

3.4.2 axes

```
axis(side, #位置1底2左3顶4右
        at #加入刻度
        labels
        pos #交叉点
        lty #线型
        col #颜色
        las #labels平行(0)/垂直(2)于坐标轴
        tck #刻度长度，正数为图表内负数为外默认-0.01
        )
```

- 举个栗子：

```
x<-c(2:16)
y<-x+1
z<-10/y
opar<-par(no.readonly=TRUE)

par(mar=c(5,4,4,8)+0.1) #确定边界，单位为1/12英寸

```

```r
plot(x,y,type="b",pch=21,col="red",yaxt="n",lty=3,ann=FALSE) #x对y作图，点型，线型

lines(x,z,type="b",pch=22,col="blue",lty=2) #向已有的图表添加散点图而非重新做一个图表

axis(2,at=x,labels=x,col.axis="blue",lty=2)

axis(4,at=z,labels=round(z,digits=2),col.axis="blue",las=2,cex.axis=0.7,tck=-.01) #平行，字体大小0.7，刻度长度及方向为-0.1

mtext("y=x+1", side=4, line=3, cex.lab=1, las=2, col="blue")

title("An Example of Creative Axes",xlab="X values",ylab="Y=X+1")

par(opar)
```

3.4.3 reference lines基准线: `abline() function`

```r
abline(a = NULL, #intercept截距
       b = NULL, #slope斜率
       h = yvalues, #水平数值，只有这个则是一条y值固定的基准线
       v = xvalues, #竖直数值，只有这个则是一条x值固定的基准线
       reg = NULL,
       coef = NULL,
       untf = FALSE,
       ...) #该函数用于添加线条
```

- 举个栗子：

```r
abline(v=c(1,10,2),lty=2,col="blue") #x=1/x=10/x=2三条线
abline(h=seq(1,10,2),lty=4,col="yellow") #y=1,3,5,7,9
```

3.4.4 legend标注 `legend() function`

```r
legend("topleft", #location
       inset=.05,
       title="Drug Type",
       c("A","B"),
       lty=c(1, 2),
       pch=c(15, 17),
       col=c("red", "blue"))
```

3.4.5 text annotations文本注释`text()/mtext() functions`

```
text(location,
        "text to place",
         pos, #相对于location的位置1below, 2left, 3above, 4right
         ...) #图表内部
```

- text()可以用于给图表里的点做标注，这里有个栗子：

```
attach(mtcars)
plot(wt, mpg,
        main="Mileage vs. Car Weight",
        xlab="Weight", ylab="Mileage",
        pch=18,
        col="blue")
text(wt, mpg,row.names(mtcars),
        cex=0.6, pos=4, col="red")
        detach(mtcars)
mtext("text to place",
        side, #1为下边界，2为左边界，3为上边界，4为右边界
        line=n, #文本标注与边界线的相对位置，0为最接近
        ...) #四面边界
```

- 再举一个栗子：

```
opar <- par(no.readonly=TRUE)
par(cex=1.5)
plot(1:7,1:7,type="n") #n for "no plotting"
text(3,3,"小婊砸")
text(4,4,family="mono","我赵日天第一个不服")
mtext(family="serif","ASS WE CAN",side=1,line=1)
par(opar)
```

3.5 组合图表 `par() /layout()`

- 举个栗子先：`par()`

```
attach(mtcars)
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2)) #组合图表2行2列
plot(wt,mpg, main="Scatterplot of wt vs. mpg")
```

```
plot(wt,disp, main="Scatterplot of wt vs disp")
hist(wt, main="Histogram of wt") #hist是柱状图
boxplot(wt, main="Boxplot of wt") #箱图
par(opar)
detach(mtcars)
```

- 再举个栗子：`layout()`

```
attach(mtcars)
layout(matrix(c(1,1,2,3),2,2,byrow=TRUE),
       widths=c(3, 1), heights=c(1, 2))
#layout函数里通过矩阵指定位置，这里构造一个2*2按行排列的矩阵
#第一个图在第一行，剩下两个图在第二行
#指定相对高度与宽度
hist(wt)
hist(mpg)
hist(disp)
detach(mtcars)
```

3.5.1 精密地控制图表: `fig= graphical parameter`

```
opar <- par(no.readonly=TRUE)

par(fig=c(0, 0.8, 0, 0.8))    #这里的数字都是相对值，把图表左下角原点当作(0,0)，右上角当作(1,1)
#本图表的范围是x轴0-0.8, y轴0-0.8

plot(mtcars$wt, mtcars$mpg, xlab="miles per gallon",ylab="car weight")
par(fig=c(0,0.8,0.55,1),new=TRUE) #在图表上方加箱图
#这个箱图在x轴上0-0.8, y轴上0.55-1, 相当于在散点图上方

boxplot(mtcars$wt, horizontal=TRUE, axes=FALSE)
par(fig=c(0.65, 1, 0, 0.8), new=TRUE) #在图表右侧加箱图
#这个箱图在x轴上0.65-1, y轴上0-0.8, 相当于在散点图右侧
boxplot(mtcars$mpg, axes=FALSE)
mtext("Enhanced Scatterplot", side=3, outer=TRUE, line=-3)
par(opar)
```

Chapter 4 Basic data management

- Manipulating dates and missing values

- Understanding data type conversions
- Creating and recoding variables
- Sorting, merging, and subsetting datasets
- Selecting and dropping variables

4.1 Creating new variables

```
s1<-seq(1,10,by=2) #1-10,by=2
s2<-seq(1,10,length=3) #元素数量3个，平均分配（1,5.5,10)
```

- 创建二进制变量：

```
s3<-ifelse(data<0,TRUE,FALSE) #设置condition,如果data<0返回TRUE
table(s3,data<0)
```

- 分割数据

```
s4<-cut(data,breaks=quantile(data)) #获取分割数据，此处分割点为默认分位点
table(s4)
#更简洁的分割方法：Hmisc package
s4S<-cut2(data,g=4) #使用包内函数cut2，分割成4组数据
s5<-factor(data) #创建factor 变量
```

- 举个栗子：

```
yesno<--sample(c("Y","N"),size=10,replace = TRUE)
yesnofac<-factor(yesno,levels=c("Y","N"))#建立factor
relevel(yesnofac,ref="N") #向量顺序不变，但是level顺序变成了N在前
as.numeric(yesnofac) #转换为数字形式，也可以as.character
```

4.2 重新编码变量recoding

- 例如：
  - 将一系列连续变量导入新目录
  - 替换错误的值
  - 基于缺失的数据创建变量

- 可以使用一系列逻辑操作符返回`TRUE/FALSE`，如>、==等

```
x|y #或
x&y #与
```

- 举个栗子：将一系列年龄数据转换为老中青三个年龄层

```
total$age1[total$age==99]<-NA #将年龄为99的数据重编码为NA
total$age1[total$age>75]<-"老人"
total$age1[total$age>=55 & total$age <= 75] <- "中年人"
#青年以此类推，且将分组后的数据保存到新的列age1中
```

- 另一种更简化的写法，使用`within`函数：

```
total<-within(total,{age1<-NA #创建空的新目录
        age1[age>75]<- "老人"
        age1[age>=55 & age<=75]<-"中年人"
        age1[age<55]<-"青年人"
        ) #within函数类似以前学的with，不同在于可以对数据框架进行修改
```

4.3 重命名变量：`fix()/rename()/names()`

```
fix(yourdataset) #方法1：使用fix调出交互编辑器
rename(dataframe,
        c(oldname="newname",
        oldname="newname",…)
        ) #方法2：加入reshape包，使用其中的rename()function
names(leadership)[2] <- "testDate" #方法3，同样需要安装并导入reshape包
#如果后面不加赋值则是单纯滴显示该列列名
```

4.4 日期date values

- 三种基本日期格式：`Date class`转化为日期,`POSIXct/POSIXlt Class`转化为秒，lt指list

```
%d #日期
%a/%A #星期，前者为缩写
%m/%b/%B #数字月份，字母缩写月份，完整字母月份
%y/%Y #2位年份，4位年份
%H #HOUR
```

```
Sys.Date() #当前日期
date() #当前时间
format(yourtime,format="%B %d %Y %a") #格式转换
as.Date() #将数值转换为时间格式，as.POSIXct/POSIXlt()同理
    #可以通过两个时间格式的相减得到时间差，但必须为同一种时间格式
    #unclass 这个object you will find the 时间差 between your object and the original time
as.character(dates) #将日期转为字符
strptime(datestring,"%B %d %Y %a") #将字符串向量结构转化为POSIXlt日期格式
```

- 可以看下帮助文档，有关`as.Date()/strftime`字符日期转换/`ISOdatetime`数字日期转换

4.5 type conversions 数据类型转换

- is判断，as转换

```
is/as.numeric/character/vector/matrix/data.frame/factor/logical
```

4.6 sorting排序

- order():

```
newdata<-total[order(total$x,decreasing=FALSE),] #注意这个格式,按x排序，如果是反序则decreasing为TRUE
```

- sort():

```
sort(X$var1,
        decreasing=TRUE,  #倒序(默认为FALSE)
        na.last=NA/TRUE/FALSE,
        )
```

- pylr package:

```
library(pylr)
arrange(X,colname)
arrange(X,desc(colname)) #倒序
X$newvar<-newdata #增加一个变量(列)
# 另一种方法cbind(X,newdata)
```

4.7 merging datasets 合并数据集

4.7.1 新增列`merge()`

```
cbind() #这个是简单的方法，把不同列合并成矩阵或数据集，适用于两个数据集没有相同列名的情况

merge(dataframeA, dataframeB,
        by.x=c("ID","Country"),
        by.y="solution_id", #by参数表示内键（相同行/列名），将两个合并的数据集相关联
        all=TRUE
        ) #如果某个变量在set A有而set B没有，合并后set B此处为NA

merge(dataframeA, dataframeB,all=TRUE)#自动默认合并
```

4.7.2 新增行

```
rbind(dataframeA,dataframeB)
#注意，两个数据及必须有相同的变量（列名），但列的顺序不一定相同
#如果列数不同会发生：列多的数据集多出的列被删除or列少的数据集多出空列，用NA填充
```

4.8 缺失值missing values及移除缺失值的几种方法

- 缺失值的栗子：问卷调查中空缺或格式错误的填写

```
x<-c(1,2,NA,NaN,10,3) #NA为缺失值，NaN为缺失数字
```

- 使用NA/NaN来标注，用于指代未定义的数学操作

```
is.na() #检测某个对象是否为NA
is.NaN() #检测某个对象是否为NaN
#NA 缺失值可以有不同的类型如integer NA
#NaN缺失值是NA的一种，但NA不一定为NaN
```

- 得到NA数量的方法，先用is.na()转换为逻辑格式，再sum(正确为1错误为0)
- 移除缺失值的几种方法：

```
#方法1：先查出缺失值并赋值给bad，再调用其反义
bad<-is.na(x)
```

```
    x[!bad]

#方法2，适用于多组向量，但无法移除NAN
good<-complete.cases(x,y,z)
    x[good] #得到无缺失值的向量x
    #对于矩阵，也可以使用complete.cases,只是之后的调用方法有些不同
    x[good,][1:6,] #记住good后的逗号
newdata<-na.omit(olddata) #方法3,na.omit()，但不足是会把NAN变成"NAN"而不是消除
```

4.9 Subsetting查询子集

4.9.1 基本方法： [] 给出子集 / [[]]和 $ 给出元素

- 查看子集名称并取名：可参考之前的重命名变量

```
names(mydata) #查看名字
names(mydata) <- c/list/matrix("name","name","name") #取名
```

- 查询向量：

```
a[1] #向量a中第一个元素（这个跟python从零开始不同）
a[c(2,4)] #列出向量a中第二个和第四个元素,同样适用于其他数据集
a[2:6] #列出向量a中第二个到第六个元素，a <-c(2:6) 等价于 a <- c(2,3,4,5,6),同样适用于其他数据集
a[a>5] #列出向量a中大于5的元素
```

- 查询矩阵：

```
y[2,] #矩阵y第二行元素
y[,2] #矩阵y第二列元素
y[2,c(3,4)] #矩阵y第二行第3,4个元素（相当于第二行与第三列第四列的交叉点）
x[1,2,drop=FALSE] #一般情况下查询返回的元素组成向量，修改drop参数为FALSE，返回一个矩阵而非向量
```

- 查询阵列及list：引入`[[]]`及`$`提取list中的某个子集，不同在于`$`后接的子集名称，而`[[]]`中的是数字或是"名称"

```
table(mydata$age,mydata$birthplace)
x[[c(1,3)]] #提取list中第一个子集的第三个元素，等价于x[[1]][[3]]
```

- 部分匹配partial matching：

```
x<-list(aard=1:5)
x$a--1 2 3 4 5
x[["a"]]--null #[[]]默认不进行partial matching
x[["a",exact=FALSE]] #将exact参数设为FALSE 开启partial matching
```

4.9.2 条件查询

- 栗子1：选取列查询（列即是变量）

```
newdata <- leadership[paste("q", 1:5, sep="")]
#等价于：
newdata <-leadership[c("q1", "q2", "q3", "q4", "q5")]
#paste()创建相同的字母向量，在第五章还会见到
```

- 栗子2：排除列查询查询
  - 注意，NULL为未定义，和NA空值不一样

```
myvars <- names(leadership) %in% c("q3", "q4") #转化为逻辑格式
newdata <- leadership[!myvars] #选取反义

#等价于：
newdata <- leadership[c(-8,-9)] #排除第8、9列数据

#还等价于：
leadership$q3<- leadership$q4<- NULL #将这两行赋值为NULL
```

- 栗子3：行查询（行即是观察值observations）

```
newdata <- leadership[which(leadership$gender=="M" & leadership$age > 30),]#大于30岁的男性
#行查询类似于SQL中where查询
#为了简化变量条件，可以attach/detach，这样就不用在变量前面加一长串修饰了
```

4.9.3 `subset()`函数——比前面条件查询简单得多的方法

- 栗子1：不小于35或者小于24，显示q1,q2,q3,q4四列数据

```
newdata <- subset(leadership, age >= 35 | age < 24, select=c(q1, q2, q3, q4))
```

- 栗子2：大于35的男性，显示列gender与q4之间的所有数据

```
newdata <- subset(leadership, gender=="M" & age > 25,select=gender:q4)
```

4.9.4 随机样本`sample() function`

```
sample(x, #the vector u wanna take samples
        size, #the number of samples you wanna take,不加的话则给出整条向量的随机排列
        replace = FALSE,
        prob = NULL)
```

- 栗子：

```
sample(1:10,4)
sample(letters,5) #选取5个字母
sample(1:10) #未设定样本数则为排序
sample(1:10,replace=TRUE) #数字可重复
```

4.10 使用SQL命令来操作数据框架：`sqldf package`

```
newdf <- sqldf(
        "select * from mtcars where carb=1 order by mpg",
        row.names=TRUE
        ) #直接使用SQL进行条件查询
```

可参考帮助文档

Chapter 5 Advanced data management

- mathematical and statistical functions
- character functions
- looping and conditional execution
- user-written functions
- ways to aggregate and reshape data

5.1 Numerical and character functions

5.1.1 Mathematical functions

```
abs(x)
sqrt(x)
ceiling(x) #不小于x的最小整数
floor(x) #不大于x的最大整数
trunc(x) #截取trancate x的整数部分
round(x,digits=n) #四舍五入保留n位小数（不包含整数数位）
signif(x,digits=n) #保留n位significant digits有效数字（相当于n包含了整数数位）
cos/sin/tan/acos/asin/atan(x)
cosh/sinh/tanh/acosh/asinh/atanh(x) #双曲线hyperbolic三角函数
log(x,base=n)
log(x)/log10(x)/log2(x) #默认base=e/10/2
exp(x)
```

5.1.2 Statistical functions

```
mean(mean(x,trim=0.05,na.rm=TRUE)) #trim[0,0.05]代表从首尾去除一定百分比的数据
median(x) #median中位数
sd(x) #standard deviation 标准差
var(x) #variance 方差

mad(x) #Median absolute deviation 中位数绝对偏差
    #先求出中位数，然后测算每个元素的绝对值与中位数的差，得到一组非负向量

quantile(x,probs=c(0.25,0.75)) #分位数，中位数是一种特殊的分位数（probs=0.5），type参数有9种，默认为1

range(x, na.rm = FALSE, finite = FALSE) #return c(minx,maxx), finite默认为FALSE，自动移除无限循环

min/max(x)
sum(x)
diff(x, #lagged difference滞后差分项
        lag=n1, #默认为1，对滞后n1阶的数据进行差分
        differences=n2)
    #就是说每个数据减去其前第n1个数据，形成一个新向量
    #差分阶数，默认为1即一阶差分,lag执行一次
```

```
      #differences=2 意味着差分执行一次后的新向量再按lag执行一次差分

scale(x, #数据的标准化与中心化——消除量纲对数据结构的影响
         #中心化即各项数据减去均值得到新向量
         #标准化指中心化后的新向量除以数据集标准差得到的新向量
         center=TRUE, #可以为逻辑变量或矩阵某列
         scale=TRUE) #可以为逻辑变量或矩阵某列
```

5.1.3 Probability functions

- 基本格式：`[dpqr]distribution_abbreviation()`
- 首字母d/p/q/r：
  - d-density密度
  - p-cumulative distribution 累积分布函数
  - q-quantile function分位函数
  - r-random generation/deviates 随机生成/模拟
- distribution_abbreviation:
  - Beta-beta
  - Binomal-binom
  - Cauthy-cauthy
  - Chi-squared (noncentral)-chisq
  - Exponential-exp
  - F-f
  - Gamma-gamma
  - Geometric-geom , 几何的
  - Hypergeometric-hyper , 超几何的
  - Lognormal-lnorm , 对数正态
  - Logistic-logis
  - Multinominal-multinom
  - Begative binomal-nbinom
  - Normal-norm 传说中的正态分布
  - Poisson-pois
  - Wilcoxon Signed Rank-signrank , 魏克森讯号等级检定
  - T-t
  - Uniform-unif

- Weibull-weibull
- Wilcoxon Rank Sum-wilcox , 魏克森秩和

- 栗子1：正态分布与泊松分布

```
rnorm(n,mean=0,sd=1) #随机正态分布，n为输出元素数量，mean默认0，sd默认1

dnorm(d,mean=0,sd=1,log=FALSE) #正态概率密度，d为分位点，log参数为TRUE时还会给出结果的log值

pnorm(p,mean=0,sd=1,lower.tail=TRUE,log.p=FALSE) #正态累积分布，p为概率
#lower.tail为TRUE意指计算分布左侧函数(即P(X<=p))，若想计算upper.tail(P(X>=p))，则为FALSE

qnorm(q,mean=0,sd=1,lower.tail=TRUE,log.p=FALSE) #正态分位分布（逆累积分布），q为分位点
#p/qnorm互为反函数：如果F()为某个标准正态分布的累积分布函数：
    pnorm(q)=F(q)
    qnorm(p)=F-1(p)

rpois(n,rate) #随机泊松分布
ppois(p,rate) #累积泊松分布，Pr(x<=p)
```

- 栗子2：从线性模型中产生随机数
  - 线性模型$y = b_0 + b_1 x + e$
  - 假定误差$e = N(0, 2^2), x = N(0, 1^2), b_0 = 0.5, b_1 = 2$

```
set.seed(20)
x<-rnorm(100) #随机选取100个x点
e<-rnorm(100,0,2) #随机选取100个e点，根据假设，mean=0,sd=2
y<-0.5+2*x+e
summary(y) #大致看一下结果
plot(x,y) #绘制散点图
#如果期望x为二进制数，将x<-rbinom(100,1,0.5)即可，其余不变
```

- 栗子3：从泊松模型中产生随机数（广义generalized线性模型）
  - 模拟泊松模型$Y - Poisson(\mu)$
  - $log\mu = b_0 + b_1 x; b_0 = 0.5, b_1 = 0.3$

```
set.seed(1)
x<-rnorm(100) #随机选取100个x点
logμ<-0.5+0.3*x
```

```
y<-rpois(100,exp(logμ)) #随机泊松模型，rate=exp(logμ)=μ
summary(y)
plot(x,y)
```

- 栗子4：均匀取样：`set.seed+runif`
  - set.seed的作用：初始化随机种子，如果以后还想产生相同的一组随机样本的话，调用相同的种子即可
  - set.seed不仅适用于随机取样，别的需要随机数的场合，也可以引入setseed进行重复

```
set.seed(1234) #初始化随机种子1234
runif(5) #在[0,1]之间产生5个随机数
```

- 注意几种调用的情况：

```
#情况1：
    set.seed(1234)
    a<-runif(5)
    b<-runif(5) #a不等于b
#情况2：
    set.seed(1234)
    a<-runif(5)
    set.seed(1234)
    b<-runif(5) #a等于b
```

5.1.4 Character functions

```
nchar(x) #返回x中每个元素的字符数量
substr(x,start,stop) #返回x中从start到stop的字符值，如果赋值的话则为替换
grep/grepl(pattern, #打算进行匹配的内容，如"A"
        x,
        ignore.case = FALSE, #是否区分大小写
        fixed = FALSE #FALSE-pattern为正则表达式格式；TRUE-pattern为单纯的字符串
        ) #字符匹配，grep返回匹配位置，grepl返回TRUE/FALSE的数量

sub/gsub(pattern,replacement,x,
        ignore.case=FALSE,fixed=FALSE
        ) #在x中查找pattern并替换，
#sub和gsub的区别是前者只做一次替换（不管有几次匹配）
#而gsub把满足条件的匹配都做替换

strsplit(x,split,fixed=FALSE) #将x中字符用split(如" ")分割开来
```

```
#注意，分隔符需要是x中有的字符

paste (..., #多个字符串
        sep = " " #分隔符，可以不加
        ) #将多个字符串连接起来

toupper/tolower(x) #全部大写/小写
```

5.1.5 其他常用函数

```
length(object) #元素数量
seq(from,to,by) #创建一个大小为[from,to]的数列，间隔数为by
seq_along(vector_name) #相当于seq_len(length(vector_name)
rep(x,n) #repeat x for n times
cut(x,n) #相当于在n中加n个点，将x分为n+1个区间
pretty(x,n) #在x这个范围里(如c(1:5)均匀选取n个数字
```

```
cat(..., #这里同paste，可以添加不同元素及分隔符，但注意结尾要加一个换行字符"\n"
        file="", #可以加文件地址，将paste写入文件
        append = FALSE, #覆盖原文，如果TRUE是不覆盖
        ) #连接并打印，也可以写入文件，如果只有一个object的话等同于print

#举个栗子；cat("i = ", 1, "\n", file="c:/work/result.txt")将i=1写入结果
```

```
dim(object) #维数（秩）
str(object) #利用str()可以对函数结构、运算结果及矩阵等object进行了解，比
head()/summary()更直观
fivenum() #5-number
summary(min,lower-hinge,median,upper-hinge,max)
head(object, n=6) #前6行，等价于object[c(1,2),]
tail(object, n=6) #后6行
colSums()/rowSums() #行列sums
table(data %in% c("1212","1234")) #查看表中数据包含"1212"或"1234”的数量
restData(data %in% c("1212","1234")) #得到表中包含这两个或数据的元素
xtabs(Xcol~Ycol+Zcol,data=mtcars) #形成交叉表cross tables
```

```
class(object) #对象类型
object.size(data) #查看data的大小，默认用bytes表示
mode(object) #存储方式
names(object) #元素名称
c(object1,object2...) #将多个对象组成一个向量
object/print(object) #打印对象
```

```
cbind(object1,object2...) #将多个对象组成一列，可以形成矩阵等
rbind(object1,object2...) #将多个对象组成一行，可以形成矩阵等
ls() #列出当前的所有对象
rm(object1,object2...) #移除某个或多个对象
newobject <- edit(object) #编辑对象并另存为新对象
fix(object) #在原位编辑对象
```

```
lm(y~x) #Fitting Linear Model拟合线性模型，会给出intercept截距与slope斜率
gl(n,k,length) #分级，n为级数，k为每级重复次数，length为向量长度默认为n*k

gl(3,5) #栗子1：3levels，每个level重复5次，15个元素
gl(3,1,10) #栗子2: 3levels,每个level重复1次，10个元素（1231231231）
```

5.1.6 apply系列函数，对matrix/array/dataframe/list进行操作

- lapply: Loop over a [list] and evaluate a function on each element
  - 对list的每一个子集做计算
  - lapply 可加入辅助函数[split]
- sapply : Same as lapply but try to [simplify the result]
- apply: Apply a function over the margins of an [array] 阵列，保留某一个维度做计算
- tapply: Apply a function over [subsets of a vector]
- mapply: Multivariate version of lapply , 多元的lapply

5.1.6.1 lapply

```
lapply(X, #should be a list.If not, use as.list() to change its type
        FUN, #这个函数(e.g. sum,mean)会对X[1]计算一次，再计算X[2]...
        ...)
```

- 这里有个例子再深入理解下：

```
x<-1:4
lapply(x,runif) #最后第一行得到一个结果，第二行两个...第四行四个
lapply(x,runif,max=10,min=0) #默认为[0,1]，这里改成[0,10]
```

- lapply可以用于匿名函数, 举个构建匿名函数的栗子：

```
lapply(x,
        function(elt) elt[,1]  #注意不加逗号！
        #匿名函数提取list中每一个子集中第一行元素，对x中每列子集都代入这个函数
        )
```

5.1.6.2 sapply

- sapply是简化的lapply
  - 如果结果是1*n list(每个元素长度均为1)，返回一个向量
  - 如果结果是m*n list(每个元素长度相等)，返回一个矩阵

5.1.6.3 apply

- apply 对一个数组按行/列计算
  - most often used to apply a function to the rows/columns of a matrix;
  - can be used with general arrays;
  - works in one line

```
array(X, #array
        MARGIN, #integer vector, indicate which margins should be retai
ned
        #就是保留一个维度，然后打散其他的维度进行函数运算
        #Margin也可以为向量，如c(1,2)表示保留第一维度和第二维度
        FUN,
        ...)
```

- 举个栗子：

```
x<- matrix(c(1:60),10,6) #10行6列矩阵，维度为2

apply(x,2,mean) #保留6列，打散10行进行平均值计算
>5.5 15.5 25.5 35.5 45.5 55.5 #最后得到6个数据为每列的平均值

apply(x,1,mean) #保留10行，打散6列进行平均值计算
>26:35 #最后得到10个数据为每行的平均值
```

- Apply的应用，几个简化的计算（这几个函数可以直接使用）：

```
rowSums=apply(x,1,sum) #保留行，做每行的sum，最后得到的数据数量=行数
rowMeans=apply(x,1,mean)
colSums=apply(x,2,sum)
colMeans=apply(x,2,mean)
```

- 再举个栗子：array

```
x<-array(c(1:60),c(3,4,5)) #3*4*5阵列
apply(x,1,mean) #行平均值，3行总共3个数据(把第三维度接起来)
#列平均值同理
apply(x,3,mean) #第三维为每个3*4matrix的平均值，总共5个数据
```

5.1.6.3 mapply

- mapply 需要引入多个参数时

```
mapply(FUN,
        ...,
        MoreArgs = NULL, #a list of other arguments in Function
        SIMPLIFY = TRUE,  #whether the results should be simplified
        USE.NAMES = TRUE #use names if the first ... argument has
names
        )
```

- 举个栗子：

```
mapply(rep,1:4,4:1)#rep函数的第一个参数执行1:4，第二个参数执行4:1
#等价于list(rep(1,4),rep(2,3),rep(3,2),rep(4,1))
```

5.1.6.4 tapply

- tapply 对不规则阵列使用函数（分组统计）

```
tapply(X, #vector
        INDEX, #该list中的每一个元素都是与X有同样长度的因子，可通过gl()实现
        FUN = NULL,
        ...,
        simplify = TRUE) #可见前面的sapply
```

- 举个栗子：

```
index<-gl(3,3) #111222333分成3个level
tapply(c(1:9),index,mean)
>2 5 8
```

5.1.6.5 split

- split: lapply的辅助函数,可用于将一个object进行分组

```
function (x, #vector/list/dataframe
        f, #a factor or a list of factors
        drop = FALSE, #whether empty factors levels should be dropped
        ...)
```

- 举个栗子：将1:30均分为3组并取平均值

```
x<-c(1:30)
f<-gl(3,1,30)
y<-split(x,f) #可以得到一个有三个子集的list，每个子集10个元素
#[(1,4,...28),(2,5,...29),(3,6,...30)]
lapply(y,mean)
```

- 再举个复杂一点的栗子：将空气质量按月分组并求取均值

```
library(datasets) #得到空气质量总数据集
aqbymonth<-split(airquality,airquality$Month) #将空气质量按月分组
lapply(aqbymonth,function(x) colMeans(x[,c("Ozone","Solar.R","Wind")],na.rm=TRUE))
#匿名函数colMeans计算其中3列的平均值
```

- 最后一个栗子：多个levels 1.1,1.2,2.1,...

```
split(x,list(f1,f2),drop=TRUE) #and drop the empty levels
```

5.2 Control Flow

- statement/cond/expr/seq

- if/else/for/while/repeat/break/next/return

5.2.1 if-else

```
if(x<10){
10
}else if(x<15){
15
}else{
0
}
```

5.2.2 for loop

```
for(i in 1:10){
print(i)
}
```

5.2.3 while loop

```
count<-0
while(count<10){
print(count)
count<-count+1
}
```

5.2.4 repeat: 初始化无限循环，只有break才能退出，用得比较少(因为需要尽可能避免无限循环)

```
x<-0
repeat{
    x<-x+1
    if(x>100){
    print(x)
    break}
    }
```

5.2.5 next/return

```
next：skip
return：exit and return a given value
```

5.3 User-written functions

5.3.1 Create an easy function

- create a new R script,and your function should be written there
- 举个栗子:

```
summy<-function(x,y){x+y}
highlight these statements
then press run to run your function in console windows
#注意一定要选中，否则无法实现
```

5.3.2 Take a matrix and calculate the mean of each column

- for loop

```
columnmean<- function(y,removeNA=TRUE){ #除去NA
    nc<-ncol(y) #nc为y的列数
    means<- numeric(nc)
    for(i in 1:nc){
        means[i]<-mean(y[,i],na.rm=removeNA)     #求y第i列的平均值,na.rm=TRUE
        print(means[i])
    }
}
```

5.3.3 参数

```
f<-function(a,b=1,c=2,d=NULL,...){} #b,c,d均为默认参数,...为可变参数，这个可以参见python
```

- 惰性编译lazy evaluation:

```
f<-function(a,b){a+2}
f(2)->4
#但函数式中涉及到关于b的操作而输入没有b会返回错误
```

- 可变参数的栗子：

```
args(paste)
function (..., sep = " ", collapse = NULL)
paste("a","b",sep=":")——"a:b" #传入字符a,b并使用：分隔
paste("a","b",se=":")——"a b :" #这里传入的是可变参数se，而分隔符则采用默认参数
```

```
空格
```

5.3.4 scoping rules作用域规则

- 自由变量的值首先取决于函数被定义时它们被赋予的值，而不是函数被调用时的值。
- 只要不是局部参数的都是自由变量
- 感觉R与python类似，先查看局部，在查看全局
- 两种环境：global environment及calling environment
  - global environment: 即为workspace，指代某函数被创建的环境，调用词汇/静态作用域
  - calling environment: 指某函数内部框架的环境，调用动态作用域
- 举个栗子来说明lexical scoping与动态作用域:

```
y = 10 #全局变量，这里是函数f/g被创建的环境
f = function(x) { #不是x的都是自由变量
    y = 2 #x为形参，y为自由变量
    y ^ 2 + g(x) #g同样为自由变量
}
g = function(x) {
    x * y #这里的y也是自由变量，调用g被创建的环境中的y，即y=10
}

f(3) #f调用局部y=2，这里为动态作用域，而g则调用全局y=10，最后返回34
```

- 在一个Function中，对一个变量的lexical scoping rule的检索次序是：
  - 先检查本Function中的calling environment，如果能找到该变量，则返回该变量;
  - 如果不能，则检索这个Function被创建(不是被调用)的Environment;
  - 对于"<<-"函数变量的赋值符，检索时跳过函数内部环境直接全局;
  - 如果一个函数被定义且调用于全局环境，那么calling/global环境没什么区别，相当于动态作用域
- 查询某个函数所处的环境：

```
ls(environment(function_name)) #返回一系列参数
get("para_name", environment(function_name)) #返回该参数的值
```

- 静态作用域造成的结果：
  - All objects must be stored in memory

- All functions must carry a pointer to their environments
- In S-PLUS, free variables are always looked up in the global workspace, so everything can be stored on the disk because the "defining environment" of all functions is the same.

5.4 Aggregation聚类 and reshaping重构

- Aggregation： replace groups of observations with summary statistics based on those observations
- Reshaping: alter the structure (rows and columns) determining how the data is organized

5.4.1 Transpose 转置t(x)

```
t(x) #行列互换
```

5.4.2 Aggregating data

```
aggregate(x, #the data object to be collapsed
        by, #[a list] of variables that will be crossed to form new obs
ervations
        FUN)
```

5.4.3 reshaping data:melting+casting

`library(reshape2)`

- melting: 保留一些变量，将指定数据变为新变量

```
carM<-melt(mtcars,id=c("cyl","gear"), #当做维度的变量，每个变量占一列
        measure.vars =c("mpg","hp")) #被当做观测值的变量，列变量名称和值组成v
ariable和value两列
```

- casting: recast data set into a particular shape
- 举个栗子：得到cyl关于mpg,hp的数据数量

```
clyData<-dcast(carM, #刚melt的数据
        cyl~variable)
#得到结果为"4轮车的mpg数据有11个hp数据有11个，6轮车***，8轮车数据***
```

- 再举个栗子：得到平均值

```
clyData<-dcast(carM,cyl~variable,mean)
```

Part II Basic Method

Chapter6 Basic graphs

- bar,box,dot plots
- pie,fan charts
- Histograms直方图,kernel density plots

6.1 Bar plots柱状图

```
barplot(height,#height is a vector or matrix
        main="the Title of the plot",
        sub="subtitle",
        xlab=" ",ylab=" ",
        names.arg=c("a","b","c"), #定义各柱子的名称
        col=c("red","yellow","green"), #三列的颜色
        legend=rownames(counts), #添加标注
        )
```

- 其他参数:

```
width=1 #柱宽度
space=c(0,1) #柱间隔,这个例子是双柱矩阵，间隔0+间隔1
beside=FALSE #对于矩阵向量，FALSE为堆叠的柱状图，而TRUE为并列的
horiz=FALSE #FALSE通常的竖直柱状图，TRUE为水平的柱状图
density=NULL #暗线密度
angle=45 #暗线角度
border="red" #边界颜色
    border=NA #无边界线
    border=TRUE #边线与暗影线颜色一致
x/ylim #limits for the axis
xpd=TRUE #是否允许柱形图超越图表区域
axes=TRUE #是否添加axis
```

```
axisnames=TRUE
cex.axis/cex.names #点与字符大小，之前学过，1.5为1.5倍
inside=TRUE#当space=0时是否添加柱与柱的分界线
plot=TRUE
axis.lty=0 #line type
offset=0 #a vector indicating how much the bars should be shifted relative to the x axis
add=FALSE #是否将该柱形图加入已存在的散点图默认为否
```

- spinegrams: 一种特殊的柱状图 `spine() function` 柱高为1，堆叠柱状图，通过比例来展示不同

6.2 Pie charts

```
pie(x, labels = names(x), #或者可以为c(),
        col = rainbow(length(x)) #使用颜色中的rainbow来填充
        main = "TITLE",
        edges = 200, #边数，默认200边近似圆形，而edge=length(x)则给出多边形
        radius = 0.8, #调整大小，pie图画在一个边长为[-1,1]的正方形内
         #如果标签过长，需要让直径小一些
        clockwise = FALSE, #默认逆时针
        init.angle = if(clockwise) 90 else 0,
        #起始角，默认为3点钟方向，如果顺时针为TRUE则默认为12点钟方向
        density = NULL, angle = 45, , border = NULL, #跟前面一样有关暗线的参数
        lty = NULL , ...)
```

- `pie3D()/fanplot()` 3D pie/扇形图，需要先`library(plotrix)`

6.3 Histograms直方图

```
hist(x, breaks = "Sturges", #可以指定直方柱数量，breakpoints基于pretty函数
        freq = NULL, #FALSE纵坐标密度，TRUE纵坐标频率
        #仅有一种情况freq默认为TRUE：breaks为等距且且下面的probability未指明
        probability = !freq, #!freq的别名
        include.lowest = TRUE,
        right = TRUE,# the histogram cells are right-closed (left open) intervals
        density = NULL, angle = 45, col = NULL, border = NULL,
        main = paste("Histogram of" , xname),xlab , ylab,
```

```
          xlim = range(breaks), ylim = NULL, #坐标系范围
          axes = TRUE, plot = TRUE, labels = FALSE,
          nclass = NULL, warn.unused = TRUE, ...)
```

- 举个栗子：
  - 给出直方图
  - 添加密度曲线density curve
  - 添加rug plots

```
hist(mtcars$mpg,freq=FALSE,breaks=12,col="yellow",
        xlab="Miles Per Gallon",main="FUCK YOUR ASSHOLE")

rug(jitter(mtcars$mpg)) #这个函数是下一张密度点的一种
#一维函数rugplots可以给出在不同区间有多少个x，如果样本太多则jitter大致数量

lines(density(mtcars$mpg), col="red", lwd=2) #密度曲线
```

6.4 Kernel density plots

```
`plot/line(density(x))`
```

- 举个栗子：

```
d<-density(mtcars$mpg)
plot(d,main="son of bitch")
polygon(d,col="red",border="blue") #填充红色，边界线蓝色
rug(mtcars$mpg,col="brown") #棕色密度曲线
```

- 再举个栗子：把不同直方图的密度点叠加在一张图上

```
#需要用到sm.density.compare(x,factor) 新的package sm
par(lwd=2) #将线宽加倍,更易于观察
library(sm)
attach(mtcars)
cyl.f<-factor(cyl,levels=c(4,6,8),
        labels=c("4 cylinder,6 cylinder,8 cylinder")
        ) #将cyl转换为factor

sm.density.compare(mpg,cyl,xlab="MPG") #将4轮、6轮、8轮车的MPG按密度点展示出来
```

```
title(main="FUCK YOU")

colfill<-c(2:(1+length(levels(cyl.f)))) #这里colfill=c(2:4)
legend(locator(1), levels(cyl.f), fill=colfill) #添加标签
detach(mtcars)
```

6.5 Box plots 箱图

- 箱图可以给出minimum，lower quartile(25%分度点),median,upper quartile(75%分度点),maximum
- 一个简单的箱图：

```
boxplot(mtcars$mpg, main="", ylab="")
```

6.5.1 使用平行箱图来比较组间数据parallel box

```
boxplot(mpg~cyl, #给出一个formula,
        data=mtcars,
        main,xlab="cyl",ylab="mpg",col="red"
        varwidth=TRUE, #箱图宽度与样本大小的平方根成比例
        horizontal=TRUE #水平坐标系
        notch=TRUE #有凹口的箱图
        ) #更细化的栗子可见P136 Figure 6.14
```

6.5.2 violin plots `vioplot() function`

```
vioplot(x1, x2, … , names=, col=)
```

6.6 Dot plots `dotchart() function`

- 适用于：将大量带标签的值展示在水平坐标轴图中

```
dotchart(mtcars$mpg, labels=row.names(mtcars),...)
```

- 举个栗子：将标签点排序输出，且对不同的范围给予不同的颜色

```
x<- mtcars[order(mtcars$mpg)] #按mpg进行排序
x$cyl<-factor(x$cyl) #创建factor
```

```
x$color[x$cyl==4] <- "red"
x$color[x$cyl==6] <- "blue"
x$color[x$cyl==8] <- "darkgreen"
dotchart(x$mpg,
        labels=row.names(x),
        cex=.7, pch=19 #之前学过的点大小,点类型
        groups=x$cyl,
  color=x$color,
        main="",xlab="")
```

Chapter 7 Basic statics

- Descriptive statistics
- Frequency and contingency tables
- Correlations and covariances
- t-tests
- Nonparametric statistics

7.1 Descriptive statistics描述统计

- 基本任务：
  - mtcars dataset；
  - examine descriptive statistics by transmission type (am) and number of cylinders (cyl)；
  - 传动方式am：[0:automatic,1:manual];cyl:4,5,6

7.1.1 A menegerie of methods

`summary()/apply()/sapply()/fivenum`

- you can get mean/sd/var/min/max/median/length/range/quantile
  - skew偏度：用于衡量分布的不对称程度或偏斜程度的指标
    - α=变量的三阶中心动差除以标准差的三次方
      `skew<-sum((x-m)^3/s^3)/n` m为均值，s为标准差
    - α图像为以x均值为基准线向左边或右边偏斜的曲线
    - α>0正偏斜；

    - α<0负偏斜；
    - α=0，对称的正态分布
  - kurtosis峰度：衡量分布的集中程度或分布曲线的尖峭程度的指标
    - β=四阶中心动差除以标准差的四次方(即方差平方)的差-3
    - `kurt<-sum((x-m)^4/s^4)/n-3` m为均值，s为标准差
    - β图像为以x均值为中心的对称曲线，区别为尖锐/平缓
    - β>0,分布比正态分布更集中在平均数周围，更尖锐
    - β=0，正态分布
    - β<0,分布比正态分布更分散，更平缓
- 举个栗子：建立函数来求取上述统计量

```
mystats<-function(x,na.omit=FALSE){
                  if(na.omit)
                  x<-x[!is.na(x)]
                  m<-mean(x)
                  n<-length(x)
                  s<-sd(x)
                  skew<-sum((x-m)^3/s^3)/n
                  kurt<-sum((x-m)^4/s^4)/n-3
                  return(c(n=n,mean=m,stdev=s,skew=skew,kurtosis=kurt))
}

sapply(mtcars[vars], mystats)
```

- 这里推荐了几个外部的packages:Hmisc/pastecs/psych

```
describe() #Hmisc, 返回变量数/样本数/缺失值数/均值/分位/5个最大最小值
#psysh package也有一个同名函数

stat.desc() #pastecs, 同样可以给出 descriptive statistics
```

7.1.2 Descriptive statistics by group

- 使用aggregate():

```
aggregate(mtcars[vars], by=list(am=mtcars$am), mean)
aggregate(mtcars[vars], by=list(am=mtcars$am), sd)
#不足：aggregate()仅能使用单值函数如mean，无法一次返回多个统计学结果
```

- alternative：`by(data, INDICES, FUN) #INDICES向量用于定义groups`

```
dstats <- function(x)(c(mean=mean(x), sd=sd(x)))
by(mtcars[vars], mtcars$am, dstats)
```

- other alternatives:

```
summaryBy() in doBy package
describe.by() in psych package
melt+cast in reshape package #可参见5.4.3
```

7.1.3 Visualizing results 见chapter 6

7.2 Frequency and contingency tables

7.2.1 fenerating frequency tables

```
table()/xtabs() #生成频数表，交叉表
prop.table(table, margins) #将频数表转换为比例
margin.table(table, margins) #得到表中某个变量的频数
addmargins(table, margins) #增加一行 summary margins (sums by default)
ftable(table) #将多维表格转化为 compact and attractive manner
```

- 举个栗子说明以上函数：ONE-WAY TABLES

```
mytable <- with(Arthritis, table(Improved)) #生成简单的频数表
prop.table(mytable) #将频数按比例显示(0.500,0.167...)
prop.table(mytable)*100 #比例*100，便于观察
```

- 再举个栗子：TWO-WAY TABLES

```
mytable <- table(mpg,cyl)
mytable <- xtabs(~ mpg + cyl, data=mydata) #需要交叉的数据在~右边，这里mpg为行名，cyl为列名
margin.table(mytable, 1) #得到某个变量的频数，1代表表中第一个变量,输出不同mpg的counts
prop.table(mytable, 1)

addmargins(mytable) #在交叉表里给所有行列增加总数值
addmargins(prop.table(mytable, 1), 2) #仅仅给出行变量(1)的总值，输出为一新列
addmargins(prop.table(mytable, 2), 1) #仅仅给出列变量(2)的总值，输出为一新行
```

```
#alternatives:
CrossTable() in gmodels package
```

- 最后一个栗子：多维表格

```
mytable <- table(mpg,cyl,hp)
ftable(mytable) #三维表扁平化
margin.table(mytable,c(1,2)) #给出第1,2个变量的counts
ftable(addmargins(prop.table(mytable, c(1, 2)), 3)) #添加第1,2个变量的sum
```

7.2.2 Test of independence in 3 methods

- chi-square test of independence

  `chisq.test(mytable)#给出X-squared, df, p-value`
- Fisher exact test

  `fisher.test(),给出p+alternative hypothesis`
- Cochran-Mantel–Haenszel test

  `mantelhaen.test(),给出Cochran-Mantel-Haenszel M^2,df,p`

7.2.3 Measures of association

- assocstats() in vcd package
  - phi coefficient+contingency coefficient+Cramer’ s V for a two-way table
  - In general, larger magnitudes indicated stronger associations
- kappa() in vcd package
  - Cohen’ s kappa and weighted kappa for a a confusion matrix

7.2.4 visualisation (chapter 6+mosaic and association plots in chapter 11)

7.2.5 converting tables to flat files(自己写一个函数)

```
table2flat <- function(mytable) {
    df <- as.data.frame(mytable)
    rows  <- dim(df)[1]
    cols  <- dim(df)[2]
    x <- NULL
    for (i in 1:rows){
        for (j in 1:df$Freq[i]){
            row <- df[i,c(1:(cols-1))]
            x <- rbind(x,row)
        }
```

```
    }
    row.names(x)<-c(1:dim(x)[1])
    return(x)
}
```

- 举个栗子来实现这个函数：

```
treatment <- rep(c("Placebo", "Treated"), times=3)
improved <- rep(c("None", "Some", "Marked"), each=2)
Freq <- c(29,13,7,17,7,21)
mytable <- as.data.frame(cbind(treatment, improved, Freq))
mydata <- table2flat(mytable)
```

7.3 Correlations

7.3.1 type of correlation `spearson/spearman/kendall`

```
cor(x, #your dataframe,or simply cor(a,b)
        use= , #对缺失数据的操作，包括all.obs(assumes no missing data)
        #everything(any correlation involving a case with missing
values will be set to missing);
        #complete.obs(listwise deletion);
        #pairwise.complete.obs(pairwise deletion)
        method= #the type of correlation:pearson/spearman/kendall
        ) #default:everything+pearson
cov() #covariances 协方差
#notice:显著性需要参考7.3.2
```

- Other types of correlations:
  - partial correlations 偏相关 `pcor() in ggm package`
  - correlation with 2 quantitative variables,controlling for one or more other quantitative variables
    即将其他变量都看做常数，就找两个变量的相关性

```
pcor(u, # numeric vector, 前两个数字是打算找相关性的，后面为控制变量
        S) #covariance matrix among the variables
hetcor() in polycor package
```

- Pearson product-moment correlations#numeric variables

- polyserial correlations 多项相关 between numeric and ordinal variables(有序变量)
- polychoric correlations 多序频相关,ordinals
- tetrachoric correlations 四分相关between two dichotomous variables
  假定有序变量和二分变量从正态分布中被移除

7.3.2 testing correlations for [significance]

- 对于null hypothesis：假定变量是彼此独立无关的`cor.text()`

```
cor.test(x, y,
        alternative = "two.side/less/greater", #two-tailed/one-tailed test
        #hepothesis:correlation<0时, "less";同理, >0时用"greater"
        #two.side is the default 默认双侧检验
        method="pearson/kendall/spearman"
        )
#不足：only one correlation per time
```

- corr.test() in psych package
  - correlations and significance levels for [matrices] of P/S/K

```
library(psych)
corr.test(data,
        use="complete/pairwise", #整列或成对删除缺失值
        method) #默认为pearson
```

- r.test() in psych package
  - significance of a correlation coefficient;
  - difference between 2 independent correlations ;
  - difference between 2 dependent correlations sharing 1 single variable ;
  - difference between 2 dependent correlations based on completely different variables ;

7.3.3 visualisation correlations (scatter plots/correlograms in cpter 11)

7.4 t tests

- 回归模型个体系数的显著性进行假设检验[样品与均值的差异]

- |t|越大-样品偏离均值越远,样品与均值的差别越显著
- 举几个应用的栗子：
  - 接受新药相比接受原有治疗是否显著提高疗效？
  - 某个制造过程是否比其他过程缺陷率更低？

7.4.1. Independent t-test独立t检验

- Are you more likely to be imprisoned if you commit a crime in the South?
  - 比较南方/非南方州被监禁的probability-two-group independent t-test
- hypothesis：人口mean相等，two groups are independent

```
t.test(y ~ x, #y:numeric,x: dichotomous二分变量;或者t.test(y1,y2)
        alternative #同前
        mu=0,#a number indicating the true value of the mean
        paired = FALSE, #是否需要paired t-test对偶t检验
        var.equal = FALSE, #默认var不等，且使用 Welsh degrees of freedom m
odification
        #if TRUE,方差相等and specify a pooled var estimate联合方差估计法
        conf.level = 0.95,
        data)
```

- 实现实例：

```
head(Uscrime)
t.test(Prob ~ So, data=UScrime) #入狱概率~南方州
#结论：t大p小，不接受原假设，南方/非南方州入狱概率不等
```

7.4.2 Dependent t-test

- if unemployment rate for younger males is greater than for older males the two groups aren’t independent
- hypothesis：不同组的差异是正态分布的

  `t.test(y1, y2, paired=TRUE)`
- 实现实例：

```
sapply(UScrime[c("U1","U2")], function(x)(c(mean=mean(x),sd=sd(x))))
with(UScrime, t.test(U1, U2, paired=TRUE))
#结论：差异显著，不接受原假设，年轻人就业率更高
```

7.5 Nonparametric tests of group differences

- 在总体方差未知或知道甚少的情况下，利用样本数据对总体分布形态等进行推断
- 不涉及有关总体分布的参数

7.5.1 Comparing two groups#note that these 2 groups are independent

- Wilcoxon rank sum test(Mann–Whitney U test): 用于检验观察值是否取自同一概率分布(某一组取得更好结果的可能性是否大于另一组)

  `wilcox.test(y ~ x, data)/wilcox.test(y1, y2)#其余参数跟前面类似`
- 这里举个栗子：之前南方州入狱问题：

```
with(UScrime, by(Prob, So, median))#给出median
wilcox.test(Prob ~ So, data=UScrime)#结果p很小，不接受原假设
#若t-tests的假定很reasonable，参数检验更易找到差异
#但当假定unreasonable时，非参数检验更适合(如有序排列的数据)
```

7.5.2 Comparing more than 2 groups

- 比较四个地区的文盲率——需要进行one-way design单项设计
- Kruskal–Wallis test : non-paramater, independent groups

  `kruskal.test(y ~ A, data) #A:grouping variable with >= 2 levels`
- Friedman test: 感觉跟上面那货差不多

  `friedman.test(y ~ A | B, data) #B:blocking variable限制条件`
- 实现实例：结论，文盲率并不一样(p很小)

```
states <- as.data.frame(cbind(state.region, state.x77))
kruskal.test(Illiteracy ~ state.region, data=states)
#以上两个test的不足：无法了解which rigions differ signifily froms other
```

- Mann–Whitney U test: `npmc() in npmc package`

```
class <- state.region
var <- state.x77[,c("Illiteracy")]
mydata <- as.data.frame(cbind(class, var))
rm(class, var)
A <- summary(npmc(mydata), type="BF") #给出所有组两两对比的结果C(n,2)
B <- aggregate(mydata, by=list(mydata$class), median)

#A：南方-其他的p明显要小，而其他地区的两两对比差异不大
#B：南方中位文盲率更高
```

- `anova()`比较几个模型哪个更合适

Part III Intermediate methods