#表示注释

Ch2:R 基本语法

1、向量

1.1 赋值：

x<-c(1,2,3,4,5,6) #用 x=c(1,2,3,4,5,6)也行

1.2 运算：

x*y #将向量 x 中的每个分量与 y 中每个分量相乘

x/y #同上，逐个相除

x^2 #逐个乘方

y^x #y 中每个元素与 x 中对应元素的运算

v<-2*x+y+1 # 生成的 v 也是向量，将 x、y 中对应分量代入计算得到 v 中元素

x+1 #x 中每个元素加 1

1.3 函数：

exp(x) #针对 x 中逐个项

sqrt(x) #对 x 中逐个项开方

min(x) #求向量 x 中最小值

max(x) #求向量 x 中最大值

range(x) #求向量 x 中数值的范围

#例如：x<-c(1,2,3,4,7)

#c=range(x)

#结果 c 是一个向量，其中项为 1，7

which.min(x) #返回 x 中最小值的位置

which.max(x) #同理

sum(x) #求 x 中各项和

prod(x) #求 x 中分量的阶乘

length(x) #向量 x 的长度，即 x 中分量的个数

median(x) #求向量 x 中的中位数

mean(x) #计算向量 x 的均值

sd(x) #计算向量 x 的标准差，分母为（n-1）

var(x) #计算向量 x 的方差

#var(x)=sum((x-mean(x))^2)/(length(x)-1)

sort(x) #将 x 中元素从小到大顺序排列

1.4 产生有规律的序列

#等差数列

x<-1:30 #表示 x=(1,2,3,…,30)

x<-30:1 #表示 x=(30,29,28,…,1)

#注意下面的例子(用这种方法产生序列，数据类型以左边为准，界限以右边为准)：

2.312:6 #返回向量 2.312 3.312 4.312 5.312

4：7.6 #返回向量 4 5 6 7

x<-2*1:15 #表示 x=(2,4,6,..,30) 即 x<-2*(1:15)

1:n-1 #表示(0,1,2,…,n-1)，即(1:n)-1

#上面这是因为：生成序列的计算优先级比数学计算高

#等间隔函数

seq(from=x1,to=x2,by=x3) #表示生成从 x1 到 x2 的向量，间隔为 x3，其中如果 by=多少没有赋值，则默认为 1。

seq(length=x1,from=x2,by=x3) #表示生成长度为 x1 的向量，从 x2 开始，间隔为 x3

#重复函数

s<-rep(x,times=3) #将变量 x 重复三次，放入向量 s 中

#例

x<-c(1,2,3)

s<-rep(x,times=3)#则 s 为（1，2，3，1，2，3，1，2，3）

1.5 逻辑向量

x<-1:7

a<-x>3

#则 a 为 F F F T T T T ，其中 F 表示 FALSE，T 表示 TRUE

#逻辑运算符有<,<=,>,>=,==(表示相等，只有一个=号那是赋值)和!=(表示不相等)

#还有&、|、!，表示且、或、非

#逻辑变量可以赋值

z<-c(TRUE,FALSE,F,T) #FT 的全写和简写等价

all(c(1,2,3,4,5,6,7)>3) #判断向量的分量是否全部大于 3，结果为 F

any(c(1,2,3,4,5,6,7)>3) #判断向量中是否存在元素大于 3，结果为 T

1.6 字符向量

y<-c(“er”,”sdf”,”eir”,”jk”,”dim”)

#或

c(“er”,”sdf”,”eir”,”jk”,”dim”)->y #把这些字符赋给 y，可以认为是一种向量

paste(“My”,”Job”) #把这两个字符拼接在一起，形成一个字符”My Job”

paste(“X”,1:6,sep=””) #将 X 字符和 1 到 6 的数字拼接在一起，最后还是形成字符向量，其中 sep 规定了用什么来分隔字符，这里是用空字符””，即生成”X1” ”X2” ”X3” ”X4” ”X5” ”X6”

paste("X",1:6,sep=".") #参考上一个，生成 "X.1" "X.2" "X.3" "X.4" "X.5" "X.6"

paste(1:10) #生成"1" "2" "3" "4" "5"

"6" "7" "8" "9"
"10"
paste(“Today is”,date()) #生成"Today is Sat Apr 16 17:07:15 2016" 后面的时间是执行这串代码时的系统时间。
paste(c(‘a’,’b’),collapse=’.’) #也可以这么做，结果是"a.b"

1.7 向量下标运算

x<-c(1,4,7)
x[2] #返回 x 向量的第 2 个分量
(c(1,3,5)+5)[2] #结果是 8，圆括号里先计算，再返回方括号中指定位置的值
x[2]<-125 #给相应位置赋值
x[c(1,3)]<-c(144,169) #给 x 向量的第 1、第 3 个分量分别赋值 144 和 169
x<-c(1,4,7)
x<5 #返回 T T F
x[x<5] #返回 1 和 4

#也可以将向量中缺失数据赋为 0

z<-c(-1,1:3,NA)
z[is.na(z)]<-0 #is.na(z)对缺失值 NA 为 T，对其他为 F，所以结果为 -1 1 2 3 0

#定义分段函数

对 $y=\begin{cases}1-x, x<0\\1+x, x\geq 0\end{cases}$

有

x=numeric(100) #numeric 表示数值型，用在这里表示 x 包含 100 个数值型向量
y<-numeric(length(x))
y[x<0]<-1-x[x<0]
y[x>=0]<-1+x[x>=0]

#下标的正整数运算

v<-10:20
v[c(1,3,5,9)] #返回值为 10 12 14 18，即相应位置值
v[1:5] #返回值为 10 11 12 13 14 同理
v[c(1,2,3,2,1)] #允许重复取值
c(“a”,”b”,”c”)[rep(c(2,1,3),times=3)] #返回”b” “a” “c” “b” “a” “c” “b” “a” “c”

#下标的负整数运算

v[-(1:5)] #表示扣除第 1 到第 5 个数
v[-c(1,5)] #表示扣除第 1 和第 5 个数

#取字符型的下标向量

ages<-c(Li=33,Zhang=29,Liu=18)
#结果为
Li Zhang Liu
33 29 18
ages[“Zhang”]
#返回
Zhang
29
#向量元素名可以后加
fruit<-c(5,10,1,20)
names(fruit)<-
c(“orange”,”banana”,”apple”,”peach”) #可以达到和上一题相似的效果

2、tapply（）函数

(非考点)#因子

sex<-c("M","F","M","M","F")
sexf<-factor(sex)
#返回
M F M M F
Levels: F M
#factor 函数把向量编码成一个因子，一般形式为：factor(x,levels=sort(unique(x),na.list=TRUE),labels,exclude=NA,ordered=FALSE) 自己查阅资料，了解各参数作用，不要求掌握。
sex.level<-levels(sexf) #返回”F“和”M”两个。
sex.tab <-table(sexf)
#返回
sexf
F M
2 3

#tapply()函数

#已有上小节的 sex 向量
height<-c(174,165,180,171,160)
tapply(height,sex,mean)
#返回
F M
162.5 175.0
#tapply 格式为 tapply(X,INDEX,FUN=NULL,…,simplify=TRUE) 其中 X 为对象，一般为向量；INDEX 为与 X 等长度的因子；FUN 为需要计算的函数；simplify 是逻辑变量，默认为 TRUE(如果为 FALSE，返回 list 格式)

3、数组和矩阵

```
z<-1:12
dim(z)<-c(3,4) #将 z 按列排成 3x4 的数组
dim(z)<-12 #又转化为 1 维数组
```

#array()函数

```
#                格                式
array(data=NA,dim=length(data),dimnames=
NULL)
```

data 是向量数据；dim 定义数组维度，默认为原向量长度；dimnames 为各维度的名字

```
X<-array(1:20,dim=c(4,5))
#生成
X
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
Z<-array(0,dim=c(3,4,2)) #定义了一个 3x4x2
```

的三维数组，元素均为 0

#matrix()函数

```
格                式
matrix(data=NA,nrow=1,ncol=1,byrow=FALSE,
dimnames=NULL)
```

括号中各参数都是函数的默认值，data 指向量数据；nrow 代表矩阵行数，ncol 代表矩阵列数；byrow=TRUE 时，代表数据按行放置，默认为 FALSE；dimnames 为矩阵维名字

```
A<-matrix(1:15,nrow=3,ncol=5,byrow=T)
#返回
 A
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
#下面两种格式也等价
A<-matrix(1:15,nrow=3,byrow=T)
A<-matrix(1:15,ncol=5,byrow=T)
```

#数组下标

```
a<-1:24
dim(a)<-c(2,3,4)
a[2,1,2] #返回 8
a[1,2:3,2:3]
#返回
     [,1] [,2]
[1,]    9   15
[2,]   11   17
#如果略去某一维的下标，表示该维全选
a[1, , ]
#返回值
     [,1] [,2] [,3] [,4]
[1,]    1    7   13   19
[2,]    3    9   15   21
[3,]    5   11   17   23
a[]<-0 #将矩阵 a 中所有分量赋值为 0
a[3:10] #此时忽略维数，按列取值
#不规则数组下标
b<-
matrix(c(1,1,1,2,2,3,1,3,4,2,1,4),ncol=3,byrow
=T)
#返回 b 为
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    3
[3,]    1    3    4
[4,]    2    1    4
a[b] #返回 a 矩阵的(1,1,1) (2,2,3) (1,3,4) (2,1,4)
```

位置的元素，本质上是把位置信息存在 b 矩阵中，再由 b 矩阵定位 a 中元素

```
a[b]<-c(1,2,3,4) #可以通过这种方式赋值
```

#数组四则运算

```
A<-matrix(1:6,nrow=2,byrow=T)
B<-matrix(1:6,nrow=2)
C<-matrix(c(1,2,2,3,3,4),nrow=2)
D<-2*C+A/B
#D 运算结果为
     [,1]     [,2]     [,3]
[1,]    3 4.666667  6.6
[2,]    6 7.250000  9.0
```

表明 R 语言中矩阵的加减乘除四则运算只针对矩阵间对应元素

```
x1<-c(100,200)
x2<-1:6
x1+x2 #结果为 101 202 103 204 105 206
x3<-matrix(1:6,nrow=3)
x1+x3
#结果为
     [,1] [,2]
[1,] 101  204
[2,] 202  105
```

[3,] 103 206
#上述两个例子表明形状不一样的向量或数组间也可以四则运算，一般规则是将二者对应位置元素进行运算(默认是按列的顺序，建议自己实验一下)，其中较短的向量或数组可以循环使用
#上述规则仅针对向量与向量、数组与向量，而数组之间的运算依然需要形状吻合
x2<-1:5
x1<-c(100,200)
x1+x2 #也不可以运算，因为长的对象长度并不是短对象的整数倍
#矩阵运算
#转置
A<-matrix(1:6,nrow=2)
t(A) #将 A 按照数学规则转置为 3x2 的矩阵
#求矩阵行列式
det(A)
det(matrix(1:4),ncol=2) #这两种方法都可以
#向量内积
x<-1:5;y<-2*1:5
x%*%y #这就是数学上的矩阵乘法
crossprod(x,y) #计算内积，表示”t(x)%*%y”，结果为 110
tcrossprod(x,y) #表示”x%*%t(y)”，即外积
x%o%y #也是外积，此次运算结果为：

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	2	4	6	8	10
[2,]	4	8	12	16	20
[3,]	6	12	18	24	30
[4,]	8	16	24	32	40
[5,]	10	20	30	40	50

outer(x,y) #也是外积
#outer 格式
outer(X,Y,fun=”*”,…) #fun 默认为乘法，即常规的外积，其他很少用，outer()函数在绘制三维曲面图时很有用，可以生成一个 X 和 Y 的网格
#矩阵乘法
A<-array(1:9,dim=c(3,3))
B<-array(9:1,dim=c(3,3))
t(x)%*%A%*%x #求矩阵二次型
#生成对角阵
v<-c(1,4,5)
diag(v) #生成以向量 v 为对角线元素的对焦矩阵，其余部分补 0
M<-array(1:9,dim=c(3,3))
diag(M) #取矩阵对角线元素
#解线性方程组和矩阵求逆
#对 Ax=b 的求解，其形式为 solve(A,b)
#单个 solve(A)，结果为求 A 的逆
（应该不考）#求矩阵特征值和特征向量
eigen(Sm) #求矩阵 Sm 的特征值和特征向量，返回的是列表形式
#一些函数
#取维数
A<-matrix(1:6,nrow=2)
dim(A) #取 A 的维数，结果为 2 3
nrow(A) #取 A 的行数
ncol(A) #取 A 的列数
#矩阵合并
x1<-rbind(c(1,2),c(3,4)) #按行拼接对象，结果为 [,1] [,2]
[1,] 1 2
[2,] 3 4
x2<-10+x1
cbind(x1,x2) #按列拼接对象
#结果

	[,1]	[,2]	[,3]	[,4]
[1,]	1	2	11	12
[2,]	3	4	13	14

#矩阵拉直
A<-matrix(1:6,nrow=2)
as.vector(A) #按列拉直成向量
as.vector(t(A)) #按行拉直
#数组维名字
X<-
matrix(1:6,ncol=2,dimnames=list(c(“one”,”two”,”three”),c(“First”,”Second”)),byrow=T)
#这样就给矩阵的 3 行 2 列命名了
#也可以这样，效果一样
X<-matrix(1:6,ncol=2,byrow=T)
dimnames(X)<-
list(c(“one”,”two”,”three”),c(“First”,”Second”))
（应该不考）#数组广义转置
A<-array(1:24,dim=c(2,3,4))
B<-aperm(A,c(2,3,1)) #把 A 的第 2 维、第 3 维、第 1 维存在 B 的第 1、2、3 维

#apply()函数

apply(A,MARGIN,FUN,…) #A 为数组，MARGIN 是固定某些维不变，FUN 是用来计算的函数

A<-matrix(1:6,nrow=2)

apply(A,1,sum) #固定第一维（行数）不变，计算每行元素之和

apply(A,2,mean) #固定第二维（列数）不便，计算每列均值

4、列表与数据框

#构造列表

```
Lst<-
list(name="Fred",wife="Mary",no.children=3,
child.ages=c(4,7,9))
#Lst 结果为
$name
[1] "Fred"

$wife
[1] "Mary"

$no.children
[1] 3

$child.ages
[1] 4 7 9
```

Lst[[2]] #返回列表第 2 项，即”Mary”

Lst[[4]][2] #返回列表第 4 项中的第 2 项，即 7

#也可以这么来

Lst[[“name”]]

Lst$name

#上述二者都是调用列表相应名称的项

#构造列表的格式

```
Lst<-
list(name_1=x1,name_2=x2,…name_n=xn)
```

#name 是列表元素的名称，x 是列表元素的对象

#列表修改

Lst$name<-“John” #直接重新赋值，另外，若想删除某一元素，可以直接赋值 NULL

#也可以增加列表元素

Lst$income<-c(1980,1600) #增加了收入项

list.ABC<-c(list.A,list.B,list,C) #直接用 c()就可以拼接不同的 list，这里的 list.A、list.B、list.C 都是列表

#数据框

#生成数据框

```
df<-
data.frame(Name=c("Alice","Becka","James",
"Jeffrey","John"),Sex=c("F","F","M","M","M"),
Age=c(13,13,12,13,12),Height=c(56.5,65.3,57.
3,62.5,59.0),Weight=c(84.0,98.0,83.0,84.0,99.
5))
#返回 df 为
      Name Sex Age Height Weight
1    Alice   F  13   56.5   84.0
2    Becka   F  13   65.3   98.0
3    James   M  12   57.3   83.0
4  Jeffrey   M  13   62.5   84.0
5     John   M  12   59.0   99.5
#也可以将列表转换为数据框
Lst<-
list(Name=c("Alice","Becka","James","Jeffrey"
,"John"),Sex=c("F","F","M","M","M"),Age=c(1
3,13,12,13,12),Height=c(56.5,65.3,57.3,62.5,5
9.0),Weight=c(84.0,98.0,83.0,84.0,99.5))
#结果为
$Name
[1] "Alice"   "Becka"   "James"   "Jeffrey"
[5] "John"

$Sex
[1] "F" "F" "M" "M" "M"

$Age
[1] 13 13 12 13 12

$Height
[1] 56.5 65.3 57.3 62.5 59.0

$Weight
[1] 84.0 98.0 83.0 84.0 99.5

as.data.frame(Lst)
#转换为
      Name Sex Age Height Weight
1    Alice   F  13   56.5   84.0
2    Becka   F  13   65.3   98.0
```

3 James M 12 57.3 83.0
4 Jeffrey M 13 62.5 84.0
5 John M 12 59.0 99.5
这样就和前面的 df 一样了
#一个矩阵可以直接用 data.frame()转换为数据框
X<-array(1:6,c(2,3))
#X 为
[,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6
data.frame(X)
#转换为数据框，如果本来矩阵有列名，则直接套用，否则系统自动取个名字，如下
X1 X2 X3
1 1 3 5
2 2 4 6
#数据框的引用
df[1:2,3:5] #返回 df 的 1 至 2 行，3 至 5 列
#结果为
Age Height Weight
1 13 56.5 84
2 13 65.3 98
df[["Height"]] #也可以根据列名来调取数据
df$Height
#数据框命名
names(df) #获取变量列名
names(df)<-c() #可以直接赋值
rownames(df)<-c() #对行名赋值
#attach()函数
attach(df) #可以把已有的数据框 df 链接到内存中，方便直接调用
#若没有用 attach(df)链接
df$Height
#若已经链接，可以直接使用 Height 列的数据
Height
attach(df)
r<-Height/Weight #这样可以很方便的使用数据框中数据
df$r<-Height/Weight #这样可以把运算后的结果保存进数据框 df 中，成为其一部分
detach() #取消调用(应该不考)
xnew<-edit(xold) #可以直接以表格形式编辑原有数据框的数据，保存在 xnew 中
5、读写数据
#读 txt 格式文件
rt<-read.table("house.data") #读取括号中名称的数据文件
is.data.frame(rt) #判断数据是否是 data.frame 格式，返回 T 或者 F
rt<-read.table("house.data",header=TRUE)
#header 的 T 或者 F 决定是否读取数据时在第一列加上数字标明顺序
#read.table()的更多用法可以自己查阅资料
#scan 函数和可以读纯文本文件
w<-scan("weight.data") #读取"weight.data"文件
inp<-scan("h_w.dat",list(height=0,weight=0))
#这里将"h_w.dat"中数据以列表形式赋值给 inp，右边的 list(height=0,weight=0)标明了新列表的变量名 height、weight，以及数据类型为数值型（这里等于号右边只要是数字就行了，不一定要写 0），也可以用 list(height="",weight="")将数据转为字符型，注意这里是按列读取，依次交替将数据给 height 和 weight
x<-
matrix(scan("weight.data",0),nrow=3,ncol=5.byrow=T) #将读取的数据存为矩阵
x<-scan() #像 c 语言一样，可以直接读取屏幕上用户输入的数据
#读其他格式文件
library(foreign) #载入 foreign 程序包，这样可以用很多程序包中的命令
read.spss() #读取 spss 的 sav 格式文件
read.xport() #读取 SAS 的 xpt 格式文件
read.S() #读取 S—PLUS 文件
read.dta() #读取 Stata 文件
read.csv() #读取 Excel 的 csv 文件
#read 系列函数括号内可以有多个参数，使函数达成各种效果，可以通过相关资料了解
data() #括号为空时显示 R 的基本软件包的数据集，也可以针对某个软件包查阅
#写数据
write(x,file="data",ncolumns=if(is.character(x))1 else 5,append=FALSE) #x 是所要写的数据，file 是文件名，append=T 时，在原文件上添

加数据，否则新写一个文件(默认 F)，其他参数可以查阅相关资料

#针对列表数据或者数据框

```
df<-
  data.frame(Name=c("Alice","Becka","James"
  ,"Jeffrey","John"),Sex=c("F","F","M","M","M
  "),Age=c(13,13,12,13,12),Height=c(56.5,65.3
  ,57.3,62.5,59.0),Weight=c(84.0,98.0,83.0,84.
  0,99.5))
write.table(df,file=”foo.txt”) #写 txt 文件
write.csv(df,file=”foo.csv”) #写 csv 文件
#格式
write.table(x,file=””,append=F,sep=””) # 其他参数前面部分有提到，sep 标明数据间隔字符，sep 参数在 read 里面作用一样
write.csv()格式差不多，其他参数查阅资料
load() #加载.Rdata 格式文件
```

6、控制流

#分支语句

#if 语句

```
if(any(x<=0)) y<-log(1+x) else y<-log(x) #如果 if()括号内为 T，执行第一个命令，否则执行 else 之后的命令
y<-if(any(x<=0)) log(1+x) else log(x) #与上等价
#标准形式
if (cond_1)
   statement_1
else if (cond_2)
   statement_2
else
   statement_3
```

#switch 语句

```
switch (statement_1,list) #statement_1 是表达式，指定 list 中的位置，函数返回 list 中对应位置的值，如果 statement_1 的值超出了 list，返回 NULL
#例
x<-3
> switch(x,2+2,mean(1:10),rnorm(4))
[1]  1.2596343  0.2801656  0.5852562
  0.6728395
> switch(2,2+2,mean(1:10),rnorm(4))
[1] 5.5
> switch(4,2+2,mean(1:10),rnorm(4))
>NULL
#也可以用有名定义
y<-“fruit”
switch(y,fruit=”banana”,vegetable=”broccoli”,
  meat=”beef”) #结果是”banana”
```

#循环语句

#for 循环

```
for(name in ex1) ex2 #name 是循环变量，ex1 是一个向量，ex 是所要循环的表达式
n<-4;x<-array(0,dim=c(n,n))
for (i in 1:n){
   for (j in 1:n){
        x[i,j]<-1/(i+j-1)
}
}
#x 结果为一个矩阵
```

#while 循环

```
while (condition) ex #当括号内为 TRUE 时，执行 ex，用法参考 for 循环
```

#repeat 循环

```
repeat {
   ex
   if(condition) break
}
#重复 ex 语句，如果 condition 为 TRUE，执行 break 跳出循环
```

7、编写自己的函数

#记住编写解方程的函数时大多要考虑精度问题

```
name<-function(x,y) ex #编写一个名为 name 的函数，参数为 x，y，根据 ex 决定函数的作用，调用时直接使用 name(x=a,y=b)
#例子
对于 x^2+y-1
#可以这样
f<-function(x,y) x^2+y-1 #如果目标函数很简短，可以直接这么编辑
#如果目标函数比较复杂
twosam<-function(y1,y2){
   n1<-length(y1);n2<-length(y2)
   yb1<-mean(y1);yb2<-mean(y2)
   s1<-var(y1);s2<-var(y2)
   s<-((n1-1)*s1+(n2-1)*s2)/(n1+n2-2)
   (yb1-yb2)/sqrt(s*(1/n1+1/n2))
```

```
}
```
#这里定义了一个求两组数据 y1、y2 的 T 统计量的函数，花括号内最后一行为最终返回的值

#T 统计量公式

$$T=\frac{(\bar{X}-\bar{Y})}{S\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}$$

#二分法解方程：取中点(a+b)/2，若 f(a)与 f(b)异号，则置 b=x，否则 a=x，当 a，b 区间长度小于精度时，停止运算

```
fzero=function(f,a,b,eps=1e-5){
    if(f(a)*f(b)>0)
        list(fail="finding root is fail!")
    else{
         repeat{
           if(abs(b-a)<eps)break
           x=(a+b)/2
           if(f(a)*f(x)<0)b=x else a=x
                }
           list(root=(a+b)/2,fun=f(x))
          }
}
f=function(x)x^3-x-1
fzero(f,1,2,1e-6)
```

#用模拟方法求定积分的近似值

#对定积分

$$I=\int_1^5 \varphi(x)dx$$

有以下几种方法

```
den=function(x) 1/sqrt(2*pi)*exp(-x^2/2) #定义被积函数，即 dnorm，这里是正态分布的密度函数
a=1;b=5
m=100000
```

#黎曼近似

```
d=(b-a)/m    #区间长度
x=seq(a-d,b+d,by=d)    #取点
I.hat1=sum(den(x)*d)
I.hat1
```

#蒲丰投针

```
x=runif(m,a,b)
y=runif(m,0,den(a))
freq=mean(y<=den(x))
I.hat2=(b-a)*den(a)*freq
I.hat2
```

#大数律

```
x=runif(m,a,b)
I.hat3=(b-a)*mean(den(x))
I.hat3
```

#用 pnorm（仅对分布函数有效）

```
I=pnorm(b)-pnorm(a)
I
```

#也可以写成函数，例如对蒲丰投针法

```
int=function(a,b,fun,m){
   x=runif(m,a,b)
   y=runif(m,0,den(a))
   freq=mean(y<=den(x))
   I.hat2=(b-a)*den(a)*freq
   I.hat2
}
int(a,b,den,m)
```

Ch3、数据描述性分析

1、 描述统计量

#位置的度量

mean(x,trim=0,na.rm=F) #求 x 数据均值，trim 表示计算前去除极端数据的比例，默认为 0，即全部保留，na.rm=T 时，允许数据 x 中有缺失数据

weight.mean(x,w) #计算加权平均值，w 是个向量，代表权重，这个不要求掌握

sort(x,decreasing=F) # 这个前面讲过，decreasing 为 T 时，数据从大到小排列，默认为 F

median(x,na.rm=F) #返回 x 数据的中位数，若 x 内元素数量为奇数，则返回大小最中间的数；为偶数，则返回中间两个数的均值。na.rm 作用和 mean 中的一样

quantile(x,probs=seq(0,1,0.25),na.rm=F) # 此函数可以找到数据的百分位数，probs 规定了具体哪个分位，默认为 0、0.25、0.5、0.75、1。其他参数可以查阅相关资料

#例

```
> w<-c(75.0,64.0,47.4,66.9,62.2,58.7,63.5,66.6,64.0
```

,57.0,69.0,56.9,50.0,72.0)

> quantile(w)

0%	25%	50%	75%	100%
47.400	57.425	63.750	66.825	75.000

#分散程度的度量

var(x) #返回数据方差，数学上对应分母为 n-1，而不是 n

sd(x) #数据标准差，为 var(x)的开方

#上面两个函数也有 na.rm 参数，作用与 mean()函数中的一样

#变异系数

cv<-100*sd(x)/mean(x) #前面乘一百是因为变异系数用百分数表示

css<-sum((w-mean(w))^2) #校正平方和

uss<-sum(w^2) #未校正平方和

R=max(x)-min(x) #极差

n=length(x)

Sm=sd(x)/sqrt(n) #Sm 为样本标准误，数学上 Sm^2 是 var(mean(x))的估计，即 x 均值的方差代表

2、 数据的分布

#随机种子

#由于电脑生成的随机数组每次都不同，所以有时候需要设定随机种子来固定生成的数据

set.seed(123) #括号里的数字可以随便填，一般填 123

#介绍 r、d、p、q 四种前缀

#r 生成某种分布的随机数

#d 返回某种分布的密度函数

#q 返回某种分布给定概率 p 后的下分位点

#p 返回某种分布的分布函数

#例如，对正态分布(norm)

rnorm(n,mean=0,sd=1) #mean 代表均值，sd 代表标准差，这里生成 n 个服从规定分布的随机数

dnorm(x,mean=0,sd=1,log=F) #生成服从规定分布的概率密度函数，log 参数为 T 时，返回对数正态分布（很少用）

qnorm(p,mean=0,sd=1,lower.tail=T,log.p=F) #产生服从规定分布的 p 分位数，lower.tail=F 时，返回上分位点，默认为下分位点，log.p 的作用和前面的 log 差不多

pnorm(q,mean=0,sd=1,lower.tail=T,log.p=F) #生成该分布的分布函数

#上面是连续情况，对于离散情况，如 Poisson 分布，dpois(k,lambda)的 k 参数必须为整数，否则结果为 0；ppois(q,lambda)的 q 参数可以不是整数，计算时系统自动取[q]；qpois(p,lambda)的计算结果返回符合 P{X=k}>=p 的最小整数 k。

#备注，其他分布的形式可以查阅资料，常用的有 beta(beta 分布)、binom(二项分布)、chisq(卡方分布)、exp(指数分布)、f(F 分布)、t(T 分布)等

#直方图、经验分布图与 QQ 图

hist(x,breaks="Sturges",freq=T,col=NULL) #绘制以 x 数据为基础的直方图，breaks 规定图的组距，freq=T 时绘制频率图，否则为密度图，col 决定颜色，其他参数可查阅资料

density(x,bw="nrd0") #此函数与 hist 直方图配套使用，估计样本密度，用于画出核密度估计曲线，大概能看出数据服从哪种分布。x 是向量数据，bw 规定带宽，当 bw 为默认值时，画出光滑曲线，其他参数自行查阅资料。

#例

w<-
c(75.0,64.0,47.4,66.9,62.2,58.7,63.5,66.6,64.0
,57.0,69.0,56.9,50.0,72.0)

hist(w,freq=F)

lines(density(w),col="blue")

x<-44:76

lines(x,dnorm(x,mean(w),sd(w)),col="red")

#观察结果中直方图、密度估计曲线与概率密度曲线的相似性

#ecdf(x) 用于画经验分布函数，用法与 density(x)相似，不要求掌握

#QQ 图用于鉴别样本的分布是否近似于某种类型分布

w<-
c(75.0,64.0,47.4,66.9,62.2,58.7,63.5,66.6,64.0
,57.0,69.0,56.9,50.0,72.0)

qqnorm(w);qqline(w) #根据数据先后画出正态 QQ 图和相应直线，如果 QQ 图的点近似在直线附近，则认为数据服从正态分布

3、 绘图命令

plot(x,y) #生成 y 关于 x 的散点图

plot(x) #如果 x 是时间序列，则生成时间序列图形；若为向量，则产生 x 关于下标的散点图；若 x 是复向量，则绘出实部与虚部的散点图。
plot(f)
plot(f,y)
#f 是因子，y 是数值向量，第一个生成 f 的直方图，第二个生成 y 关于 f 水平的箱线图（不要求掌握）
plot(df) #df 为数据框，则绘出数据框中不同指标构成的散布图
attach(df)
plot(~A+B) #绘出数据框中 A 关于 B 的散点图
plot(C~A+B) #分别绘出 C 关于 A、C 关于 B 的散点图（两张图）
#参数(用在函数的括号里)
add=TRUE #表示在原图上加图，否则替换原图
axes=FALSE #表示所绘图形没有坐标轴
log="x",log="y" #分别表示 x，y 轴数据取对数，也可以有 log="xy"
#type 命令
type="p" #(默认值)绘散点图
type="l" #画实线
type="b" #用实线连接所有点
type="o" #实线通过所有点
type="h" #绘出点到 x 轴的竖线
type="s" or "S" #绘出阶梯形曲线
type="n" #不画任何点或曲线，常用于预置一张空图，便于后来添加曲线。
xlim=c(a,b) #表示 x 轴范围
ylim=c(a,b) #表示 y 轴范围
col="blue" #表示颜色
pch=1 #数据点的形状
cex=1 #点的大小
xlab="字符串" #字符串内容为 x 轴的说明
ylab="字符串" #对 y 轴的说明
main="字符串" #对图表的说明（标题）
sub="字符串" #对子图的说明，当把多个图排在一起的时候会用到
#加点、线和标记
point(x,y) #根据 x,y 的定位在图中加点
lines(x,y) #根据 x,y 的定位在图中加线
text(x,y,labels) #x,y 用于定位，label 可以给图中点做标记
#用法
w<-
c(75.0,64.0,47.4,66.9,62.2,58.7,63.5,66.6,64.0,57.0,69.0,56.9,50.0,72.0)
y=1:14
plot(y,w,type="l",main="haha");text(y,w,labels=c(1,2,3,4,5,6,7,8,9,10,11,12,"ha","ho")) #text 中的标记数量要与点对应，默认情况下是依次填数字
abline(a,b) #在图中加一条 y=a+bx 的直线
abline(h=y) #在图中画一条高度为 y 的水平直线
abline(v=x) #在图中画一条过 x 轴上对应点的垂直线
abline(lm.obj) #画出线性模型得到的线性方程
polygon(x,y) #连接所有坐标画多边形
title(main="Main Title",sub="sub title",) #最右边的逗号可以不加，main 加在图片顶部，sub 加在图片底部
axis(side,……) #在坐标轴上加标记，side=1 表示加在图底部，side=2 表示在左侧，side=3 表示在顶部，side=4 表示在右侧
#图例
Legend("bottomright",legend=c("A","B"),col=c("blue","red"),pch=c(5,24),lty=1) #在原图上添加图例，"bottomright"定位在图片右下角，也可以用 x,y 定位，legend 决定添加图例的内容，col 决定颜色，pch 确定图例点的类型，lty 决定线的类型，其他参数可查阅资料
4、 多元数据
```
ore=data.frame(
   x=c(67,54,72,64,39,22,58,43,46,34),
   y=c(24,15,23,19,16,11,20,16,17,13)
)
ore
ore.m=colMeans(ore); #求数据框均值，也可以用 apply
ore.s=cov(ore); #求协方差或矩阵
ore.r=cor(ore) #求相关系数或矩阵
#格式
cov(x,y)
```

cor(x,y)　#其他参数可查阅资料
#cov 与 var 的用法有一点相似
#cov(ore) 与 var(ore)结果一样。但如果对单个对象求方差，则要这样：
cov(orex,orex) 或 var(ore$x)

Ch4、参数估计(从这章开始数学比较多)

1、点估计

#矩估计都是数学

#极大似然估计，基本理论都是数学，这里假设读者已经了解数学背景，仅讨论怎么求解

uniroot(f,interval,lower=min(interval),upper=max(interval),tol=.Machine$double.eps^0.25,maxiter=1000,…) #此函数用来解一元方程，f 为方程的表达式，interval 为求值区间，可以用 c(a,b)来规定，tol 为计算精度，maxiter 是最大迭代次数，其他参数不常用
#对于 Cauchy 分布，其似然函数为：

$$L(\theta;x)=\prod_{i=1}^{n} f(x_i;\theta)=\frac{1}{\pi^n}\prod_{i=1}^{n}\frac{1}{1+(x_i-\theta)^2}$$

对数似然函数为：

$$\ln L(\theta;x)=-n\ln(\pi)-\sum_{i=1}^{n}\ln(1+(x_i-\theta)^2)$$

求偏导，得到对数似然方程：

$$\sum_{i=1}^{n}\frac{x_i-\theta}{1+(x_i-\theta)^2}=0$$

代码：

```
x=rcauchy(1000,1)
f=function(p) sum((x-p)/(1+(x-p)^2))
out=uniroot(f,c(0,5)) #通过解对数似然方程，得出了θ的最大似然估计
optimize(f,interval,lower,upper,maximum=F,tol,…) #用于求一元函数的极小值，也可以写成 optimise，maximum 默认为 F，代表求极小值，否则求极大值
loglike=function(p) sum(log(1+(x-p)^2))
out=optimize(loglike,c(0,5)) #这是直接求柯西分布的对数似然函数的极值，跳过了似然方程
nlm(loglike,0) #直接求 loglike 的极小值点，0 是初值
```

2、模拟大数定律

参考作业 3

3、单个正态总体的区间估计

#数学部分可以查阅资料
#均值 μ 的区间估计
#置信度为 1-α，的双侧区间估计
剩下的都是数学，自己看书吧，考试前写不完了