## 第二章

#显示当前目录

getwd()

#改变工作目录/或\\

setwd("C:/Users/Administrator/Desktop/0914140111李梦月上机作业六")

setwd("C:\\Users\\Administrator\\Desktop\\0914140111李梦月上机作业六")

#不退出R的情况下保存你的工作空间

save.image()

#使用history函数来查看最近输入的命令

history()

#.Last.value变量存储最近一个计算出的表达式值

x=5

x+10

y<-.Last.value

y

#保留字不能做对象名break、else、FALSE、for、function、

#if、Inf、NA、NaN、next、repeat、return、TURE、while

#开头必须是英文字母或“.”，以点为开头时，第二个字母不能是数字

#对象名称只能是字母、数字、“\_”和“.”

#、使用print函数输出所有变量和表达式值，也可以输出矩阵和列表值，每次只能显示一个对象

z<-100

print(z)

#cat一个输出函数，功能和print有相同之处,可以将多个对象连接并以连续的方式显示,print不行

a<-c(1,2,3)

cat("hellow cat",a)

#列出所有变量

ls.str()

#强制显示所有变量

ls(all.names=TRUE)

#删除变量-永久地从工作空间中删除一个或多个对象,无法撤销

x<-55

y<-66

ls.str()

rm(x,y)

ls.str()

#基本数学运算

#次方(^/\*\*)

x<-2^4

y<-2\*\*4

cat(x,y)

#平方根--默认7位

x=sqrt(16)

x

#对数--以m为底x的对数log(x,m)、log10()

log(100,base=10)

log10(100)

log(100)

#以e为低的对数log()

#科学符号e---15000=1.5\*10^4=1.5e4

x<-2e4/2e3

x

#四舍五入函数round(x, digits=k) / round(x, k)

round(982.546,digits=2)

round(982.546,2)

round(982.546,-2)

round(922.546,-2)

#实数x以四舍五入方式保留k位有效数字signif(x, digits=k)

signif(11.0592,digits=3)

signif(11.0592,3)

#近似函数(取整)

#floor(x)：得到小于等于x的最近整数

#ceiling(x)得到大于等于x的最近整数

#trunc(x)直接取整，小数全舍

floor(11.0592)

ceiling(11.0592)

trunc(11.0592)

#阶乘factorial(x)

factorial(3)

#特殊运算符

# %%：取模、求余；%/%：整除；%\*%：矩阵乘积；%in%：右侧变量中包含左侧变量时，为TRUE，否则，为FALSE

5%%2

5%/%2

c(1,2)%\*%c(1,2)

c(1,2)%in%c(1,2)

c(1,2)%in%c(3,4,1)

#特殊变量

#Inf无限大 -Inf负无限大

x=-5/0

x

5/Inf

#判断是否是无限大/无限小

is.infinite(x)

is.finite(x)

#NaN非数字或无定义数字

y<-Inf/Inf

y

is.nan(y)

#NA 缺失或无法得到

z<-NA+100

z

is.na(z)

## 第三章

一、#numeric()函数生成初始向量，向量的元素均为0 numeric(length=0)

numeric(length=4)

#数列seq(from, to, by=width, length.out=numbers)

seq(3,9)

seq(3,9,by=2)

seq(3,9,length.out=5)

#重复向量对象函数rep()rep(x, times=重复次数, each=每次每个元素的重

复次数, length.out = 向量长度)

rep(3, 3)

rep(1:3, times=3, each=3)

rep(1:3, times=3, each=3, length.out=8)

#简单向量对象的运算

x<-1:5

y<-x+3

y

#不同长度向量对象相乘

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

y<-c(1,2)

z<-x\*y

z

#常见向量对象的数学运算函数

#sum()求和

sum(22+33)

#max()最大值

x<-c(2,4,7,1)

max(x)

#min()最小值

min(x)

#length()计算向量对象长度函数结果为向量长度，即向量元素个数

length(x)

#prod()函数 计算所有元素的积

prod(1:3)

#cumsum():连加，即前一个元素加上后一个元素的和作为后一个元素的新

值存入向量中。

cumsum(1:4)

#cumprod():连乘

cumprod(1:3)

#cummax():最大

cummax(1:3)

#cummin():最小

cummin(1:4)

#diff()返回各元素与下一个元素的差结果会比原先向量少一个元素

#sort(x, decreasing = FALSE)从小到大排序

#rank()传回向量对象，结果是原向量对象的各元素在元向量对象按从小到

大排序后所得向量对象中的位次

#rev()将向量对象颠倒排列

#mean()：计算样本的平均值

#media()：计算样本的中位数

#sd()：计算样本的标准偏差

#var()：计算样本的方差

#cor(x, y)：计算两变量间的相关系数

#cov(x, y)：计算两变量间的协方差

特殊数值向量（Inf、-Inf、NA）

任何整数或实数值与Inf相加，结果均是Inf。

任何整数或实数值与-Inf相加，结果均是-Inf。

函数中向量对象参数中有NA的，结果是NA。也可以使用na.rm=TURE去除NA，但diff()、cummax()、cumin()无法使用此方法。

二、逻辑向量

1. 构造：logical()函数构造初始逻辑向量

logical(3)

[1] FALSE FALSE FALSE

注意：对向量作逻辑运算，其返回值为逻辑向量

2. 逻辑运算包括：

==、！=、>、>=、<、<=

&、|、！

xor (x, y)如果x和y不同，传回TRUE

3. 逻辑向量函数

4. which()函数

x<-c(5,6,7,8,8)

which(x>6)

3,4,5

which.max()最大值的第一个索引值which.max(x)/4

which.min()最小值的第一个索引值

5. all()函数

如果要判断一个逻辑向量中的每一个元素是否都为TRUE，可以用all()

all(x, na.rm=FALSE)

6. any()函数

如果要判断一个逻辑向量中是否存在未TRUE的元素，可以用any()

any(x, na.rm=FALSE)

all(x>5)

any(x>5)

7. 特殊向量处理（Inf、-Inf、NA）

任何与NA的比较均返回NA

Inf和-Inf则按照实际进行比较，一般来说-Inf最小，Inf最大

三、字符向量

1. character()函数构造初始字符型向量

character(length=k)生成k个元素且元素内容为空的向量

character(length=3) "" "" ""

2. length()函数：返回字符串向量的长度

length("More")

1

x<-c("More", "and", "More")

length(x)

3

3. nchar()函数：获取字符串长度

nchar("More")

[1] 4

如果对一个字符向量运用nchar函数，返回每个字符串的长度

x<-c("More", "and", "More")

nchar(x)

4 3 4

4. paste()：连接字符串，

①　通过首位连接，将多个对象连在一起，构成一个新对象

paste("My", "Job")

[1] "My Job"

注意：默认情况下，函数paste在一对字符串中间插入一个空格

②　paste(…, sep="", collapse = NULL)

paste("My", "Job", sep="-")

[1] "My-Job"

③　 如果参数是一个或多个字符串向量，则paste会生成所有参数的组合

x <- c("Girls", "Boys")

paste("My", x)

[1] "My Girls" "My Boys"

④　collapse 将多个字符串连接成一个大的字符串

paste(…, sep="", collapse = NULL)

paste("My", x, collapse=" and ")

[1] "My Girls and My Boys"

5. strsplit()分解字符串：将字符型向量分解成多个字符串

strsplit(“ My Job ", " ")

[[1]]

[1] " My " " Job "

6. substr(string, start, end)提取子串：

①　提取或替换字符向量的部分子串，子串开始于start，结束于end

substr("Children", 1, 5)

[1] Child

②　如果参数是字符串向量，则应用于每个字符串元素并且返回一个子串向量

x <- c("Girls", "Boys")

substr(x, 1, 3)

[1] "Gir" "Boy"

③　substr(x, start, stop)<-value

substr(x, 1, 3)<-c("abcd", "abcd ")

[1] "abcls" "abcs"

7. 替代子串

sub：替代字符串中的第一个子串实例

sub(old, new, string)

y<-c("More", "and", "More")

sub("More", "Less", y)

gsub：替代字符串的所有子串实例

gsub(old, new, string)

gsub("More", "Less", y)

8. 大小写

toupper()将字符串改成大写

tolower()将字符串改成小写

 x <- c("Girls", "Boys")

 toupper(x)

 “GIRLS" “BOYS"

 tolower(x)

 "girls" "boys"

9. unique()：让向量内容没有重复的出现

 x<-c(5,8,4,5)

 unique(x)

 [1] 5 8 4

对于字符向量则需要先将分成个别单词，然后在处理重复单词

10. noquote()去掉字符型变量中的引号

 noquote(x)

 [1] girls boys

11. 日期的设置与使用

①　as.Date()函数设置日期向量， “YYYY-MM-DD”

x.date<-as.Date(“2017-8-30”)

x.date<-as.Date(“2017/8/30”)

日期向量进行加减法运算，即加上几天或减去几天

②　weekdays()：返回某个日期是星期几

weekdays(x.date)

③　months()：返回某个日期对象是几月

months(x.date)

④　quarters()：返回某个日期对象是第几季

quarters(x.date)

⑤　Sys.localeconv()了解目前所使用系统的本地化的各项参数的使用格式。

⑥　Sys.Date()返回目前的系统日期

⑦　seq()

seq(x.date, by=“1 months”, length.out=12)

⑧　使用不同格式表示日期

as.Date(“4 9 2017”, format=“%d %m %Y”)

%B、%b

%d、%m

%Y、%y

help(strptime)

⑨　Sys.time()返回目前的系统时间

⑩　as.POSIXct()用于设定时间向量，默认由1970年1月1日开始计数，以秒为单位。

x.time<- “1 1 1970, 02:00:00”

x.time.fmt<- “%d %m %Y, %H:%M:%S”

x.Times<-as.POSIXct(x.time, format=x.time.fmt)

%H、%I、%M、%S、%p：AM/FM

注意：可进行加减运算，表示时间的前移或延后

可进行比较运算，早晚

11　seq()

seq(x.Times, by=“1 years”, length.out=6)

12　练习

对islands数据集作向量相关操作，如大小，排序等

## 第四章

类型分类

数值向量

逻辑向量

字符向量

向量运算

与常数的四则运算

数值向量之间的运算

字符向量的运算函数

nchar()、paste()、strsplit()、substr()、sub()

matrix()

matrix(data, nrow=?, nol=?, byrow=FALSE, dimnames=NULL)

byrow为FALSE时，表示按列填充；byrow为TRUE时，表示按行填充

dimnames设置行名和列名

dimnames = list(c(colnames), c(rownames))

矩阵的建立

现有一向量(1,2,3,4,5,6)，希望得到一个2×3的矩阵。

theData <-c(1,2,3,4,5,6)

mat1<-matrix(theData, 2, 3)

mat2<-matrix(theData, 2, 3, byrow=FALSE)

mat3<-matrix(theData, 2, 3, byrow=TRUE)

mat4<-matrix(theData, 2, 3, byrow=TRUE, dimnames = list(c('1st','2nd'), c('1st','2nd','3rd')))

取得或修改矩阵对象的行名和列名

rownames()：取得和修改矩阵对象的行名

colnames() ：取得和修改矩阵对象的列名

rownames(mat3) <- c(‘1st’, ‘2nd’)

cownames(mat3) <- c(‘1st’, ‘2nd’, ‘3rd’)

mat3[‘1st’, ‘2nd’]

删除行名或列名

rownames(mat3) <-NULL

使用索引取得矩阵元素的值

将多个向量合并成矩阵

rbind()：将两个或多个向量组成矩阵，每个向量各占用一行。

cbind()：将两个或多个向量组成矩阵，每个向量各占用一列。

注意：向量长度相等。如向量对象类型不同，则会转换为字符或者数字。

str()：查看矩阵对象的结构

nrow()：得到矩阵的行数

ncol()：得到矩阵的列数

dim()：得到矩阵的行数和列数

length()：取得矩阵或高维数组对象的元素个数

is.matrix()：检查对象是否是矩阵

is.array()：检查对象是否是Array

rowSums()：计算行中元素的总和

colSums()：计算列中元素的总和

rowMeans()：计算行中元素的平均值

colMeans()：计算列中元素的平均值

矩阵的运算

转置矩阵t()

实现矩阵的行列元素互相对调

%\*%矩阵相乘

矩阵的运算

diag()：当第一个参数是矩阵时，可传回矩阵对角线的向量值。

solve()：可传回反矩阵

det()：计算矩阵的行列式值。

三维或高维数组

array()可用于建立三维数组

identical()比较两个对象是否完全相同。

## 第五章

factor(x)或as.factor(x)

其中，x向量是欲转换为因子的向量。levels是原x向量内元素的可能值。

因子的转换

as.character()：将因子转换成字符串向量

as.numeric()：将因子转换成数值向量

当要转换的因子为数值型因子时，可以使用as.numeric(as.character())

str()函数查看此因子的结构

levels()函数查看此因子的levels

nlevels()函数可返回Levels的数量

table()函数自动统计在因子的所有元素中，levels中各值出现的次数，其结果按照levels的顺序显示。

有序因子主要是处理有序的数据

可用两种方法建立：

factor()函数，增加参数“ordered=TRUE”

ordered()函数

建立列表list()

使用names()添加元素名称

length()函数获取列表的元素个数

获得列表内对象的元素内容

$ stu.inf2$Year

[[]] stu.inf2[[1]]

[] stu.inf2[1:2]

编辑列表内对象的元素值

修改和添加 $或[[]]

建立第一个数据框

ZS<-c(95.5,85,78)

LS<-c(80,85,75)

WW<-c(82,56,70)

stu.inf2<-data.frame(ZS, LS, WW)

如不想字符串向量变为因子变量，则建立数据框时，增加参数“stringsAsFactors = FALSE”

特殊符号$ 读取数据框列名内的数据 stu.newinfo$stu.name

添加一条列记录 weight<-c(65,71,58,55)

添加行数据和列数据

rbind()函数：添加单行数据

添加多行数据，现将这些数据组合成数据框，在使用rbind()函数将来给你个数据框组合即可。

cbind()：添加单列数据

添加多列数据，同rbind()，建立数据框，再添加。

## 第六章

函数

 R语言拥有丰富的内建函数，或一些R语言专家提供的额外的数据集。

 根据需要解决的问题，编写函数

程序脚本Source

函数

 函数名->function (参数1,参数2…)

 {

 程序代码

 程序代码

 …

 }

将数值向量转成百分比

 Percent<-function(x)

 {

 x.percent<-round(x\*100,digits=2)

 x.final<-paste(x.percent,sep=“”, “%”)

 return(x.final)

 }

函数

 函数可以看做是一个对象，直接输入函数名称，可以显示函数具体内容。

 注意，多个函数时避免重名

 函数在运行时，有时会面临某些状况的出现，需要提早结束函数，不再往下执行，这时可以使用return()函数

 如果函数主体只有一行代码，则可以省略大括号

 Percent<-function(x) paste(round(x\*100,digits=2), sep=“”, “%”)

 break语句和next语句

if语句

 格式1：

 if(逻辑表达式){

 系列运算命令

 }

 格式2：

 if(逻辑表达式){

 系列运算命令A

 }else{

 系列运算命令B

 }

向量数据处理

 ifelse(逻辑判断,TURE表达式,FALSE表达式)

 ifelse(c(1,5)>3,10,1)

switch语句

 switch(判断运算,表达式1,表达式2,…)

 判断运算的最终值可能是数字或文字，如果最终值是1，则执行表达式1，如果最终值是2，则执行表达式2，以此类推。如果最终值是文字，则执行相应的表达式。

 注意：switch语句无法处理向量数据

复习

 函数function

 if

 if…else…

 if…else if…else

 ifelse(逻辑判断,TURE表达式,FALSE表达式)

 switch语句

for循环

 for循环可用于向量对象的操作

 格式1：

 for(循环索引in区间)

 单一运算命令

 格式2：

 for(循环索引in区间){

 系列运算命令

 }

while循环

 while(逻辑表达式){

 系列运算命令

 }

repeat循环

 repeat{

 单一或系列运算命令

 if(逻辑表达式) break

 其他运算命令

 }

break、next

 break语句

 和循环语句一起用，跳出循环

 next语句

 需与if语句，也就是逻辑表达式配合使用，但跳出目前这次循环的剩下的命令，直接进入下一轮循环

输入与输出

 认识文件夹

 数据输出cat()函数

 读取数据scan()函数

 输出数据write()函数

 其他输入输出

认识文件夹

 在进行程序设计时，可能常需要将执行结果储存至某个文件夹

 getwd()：获得目前的工作目录

 setwd()：更改目前的工作目录

 file.path()：将片段数据路径组合起来变成一个完整的路径

 dir()：列出某个工作目录下的所有文件名以及子目录名称

 list.files()：列出某个工作目录底下的所有文件名以及子目录名称

认识文件夹

 file.exist()：检查指定的文件是否存在，如果是则返回TRUE，否则返回FLASE

 file.rename()：更改文件名

 file.create()：建立文件

 file.copy()：进行文件的复制。这个函数会将第一个参数的原目录文件复制到第二个参数的目的目录文件

 file.remove()：删除指定的文件，须填写绝对地址

数据输出cat()函数

 cat()可以在屏幕或文件输出R语言的计算结果数据或是一般输出数据

 cat(系列变量或字符串, file=“”, sep=“”, append=FALSE)

 输出到文件

 cat(“R Language”, file=“~/example.txt”)

读取数据scan()函数

 scan()：读取屏幕输入或外部文件的数据，结束读取屏幕输入时可以直接按回车键

 scan(file=“”,what=double(), namx=-1, n=-1, sep=“”, skip=0, nlines=0, na.strings=“NA”)

输出数据write()函数

 write()：将一般向量或矩阵数据输出到屏幕或外部文件

 write(x, file=“data”, ncolumns=k, append=FALSE, sep=“”)

 write(letters, file=“”, ncolumns=4)

 write(letters, file=“~/example12.txt”)

 x1<-1:10

 write(x1, “”, ncolumns=5, sep=“,”)

 x2<-matrix(1:10,nrow=2)

 write(x2, file=“”, ncolumns=5)

其他输入输出

 readClipboard()：读取剪贴板数据，不能用于Mac OS系统

 read.table()：读取表格形式的文件

 读取Excel文件，使用XLConnect扩展包，再进行路径定位，读取

 library(“XLConnect”)

 excel1<-file.path(“”)

 readWorksheetFromFile(excel1, sheet=“sheet1”)

 read.csv()：读取.CSV文件数据

 read.delim()：读取.txt文本数据

 writeClipboard()：将数据输出至剪贴板

 write.table()：将数据写成表格形式的文本文件

 write.table(x, file=“”, quote=TURE, sep=“”, eol=“\n”, na=“NA”, dec=“.”, row.names=TURE, col.names=TURE)

 SAS、SPSS等

 library(foreign)

 读取数据

 read.S() read.spss() read.ssd()

 read.xport() read.mtp()

 输出数据

 write.foreign(df, datafile, codefile, package=c(“SPSS”, “Stata”, “SAS”), …)

## 第七章

导入数据

 程序-安装程序-XLConnect，XlConnectJars，rJava

 library(“XLConnect”)

 excel1<-file.path(“H:/2017.10.18/学生成绩单.xlsx”)

 CJD<-readWorksheetFromFile(excel1, sheet=“sheet1”)

 可以将xlsx文件转换成csv文件，然后导入，免除程序包的加载

 CJD<-read.csv (file=“H:/2017.10.18/学生成绩单.csv”)

 数值自动添加NA

 字符缺省

 CJD<-read.cvs (file=“H:/2017.10.18/学生成绩单.csv”,na.strings=c(“NA”, “”))

查看数据及结构

 CJD

 str(CJD)

 结构不同

 ls.str(CJD)

数据预处理

 将成绩小数位设为2位小数

 options(digits=3)

 #本次R程序中通篇改变为2位小数

 CJD[,4:6]<-round(CJD[,4:6],2)

数据预处理

 是否存在缺省值

 complete.cases(CJD)

 去除缺省值

 CJD[complete.cases(CJD),]

 修正错误信息(数据不能有NA值)

 for和if

 去除重复值

 duplicated(iris)

 which(duplicated(iris))

 !/-

数据处理

 求平均值

 lapply(CJD[,4:6],mean)

 sapply(CJD[,4:6],mean)

 求中位值

 sapply(CJD[,4:6],median)

 求最大值

 sapply(CJD[,4:6],max)

 求最小值

 sapply(CJD[,4:6],min)

数据汇总

 summary(CJD)

 数值变量：最小值、最大值、平均值、第1个四分位数、中位数、第3个四分位数。如果有NA值，则会列出NA值的数量

 因子：列出频率表，如果有NA值，则会列出NA值的数量

 字符串变量：列出字符串的长度

按性别，姓氏排序

 dec.order<-order(CJD2$SEX，CJD2$NAME,decreasing=FALSE)

 CJD[dec.order,]

对C语言进行绘图

 彩色图

 rainbows<-rainbow(20,start=0,end=0.9)

 barplot(CJD$C,main="C Score",xlab="Name",ylab="Score",names.arg=CJD$NAME,col=rainbows)

 颜色少时，可以直接写颜色的英文名字

 col=c(“red”, “blue”)

对C语言进行绘图

 灰度图

 heights<-tapply(CJD$NAME,CJD$C)

 rel.hts<-rank(heights)/length(heights)

 grays<-gray(1-rel.hts)

 barplot(CJD$C,main="C Score",xlab="Name",ylab="Score",names.arg=CJD$NAME,col=grays)

对C语言进行绘图

 按照成绩不同，绘图

 fill\_colors<-c()

 for(i in 1:length(CJD$C)){

 if(CJD$C[i]==100){

 fill\_colors<-c(fill\_colors,"red") }

 else if(CJD$C[i]<100&&CJD$C [i]>=80){

 fill\_colors<-c(fill\_colors,"yellow") }

 else if(CJD$C[i]<80&&CJD$C [i]>=60){

 fill\_colors<-c(fill\_colors,"blue") }

 else{

 fill\_colors<-c(fill\_colors,"green")

 }

 }

 更改标题和xy轴字体

 font.main=4,

 font.lab=3,

 不设边框线

 border=NA

 添加图例

 legend(x=15,y=100, #左上角点的坐标

 cex=.8, #字体大小缩放比例

 inset=5, c("Excellent","Good","Ordinary","Bad"),

 pch=c(15,16,17,19), #图例中的符号

 col=c("red","yellow","blue","green"),

 bg="#821122", #背景色

 xpd=TRUE, #可以在绘图区之外显示

 text.font=8, text.width=3.5,

 text.col=c("red","yellow","blue","green")

 )

数据处理

 数据中心化和标准化，得到综合得分

 afterscale<-scale(CJD[,4:6])

 afterscale

 scale中的两个参数center和scale的解释：

 1.center和scale默认为真,即T或者TRUE

 2.center为真表示数据中心化

 3.scale为真表示数据标准化

数据分析

 标准差均值，填到表中

 score<-apply(afterscale,1,mean)

 CJD<-cbind(CJD,score)

 CJD

综合得分的百分位数

 quantile(x,probs):求分位数，其中x为待求分位数的数值型向量，probs为一个由[0,1]之间的概率值组成的数值向量

 afterquantile<-quantile(score,c(.8,.6,.4,.2))

 afterquantile

score转为等级（离散型）

 使用逻辑运算符，把score转为等级

 CJD$grade[score>=afterquantile[1]]<-"A"

 CJD$grade[score<afterquantile[1]&score>=afterquantile[2]]<-"B"

 CJD$grade[score<afterquantile[2]&score>=afterquantile[3]]<-"C"

 CJD$grade[score<afterquantile[3]&score>=afterquantile[4]]<-"D"

 CJD$grade [score<afterquantile[4]]<-"E"

 CJD

图形

 软件包graphics

 散点图plot

 条形图barplot

 箱线图boxplot

 直方图hist

 正态Q-Q图qqnorm

 函数curve

散点图

 如果在x和y之间有任何关系，plot是看清数据间关系的快捷方法

 plot(x,y)

 在图形窗口绘制(x,y)数据对的图形

 plot(CJD)

 plot(CJD$NO,CJD$C)

 plot(CJD[,4:6])

 plot(CJD$C, type=“l”)

添加网格

 plot(x, y, type=“n”)

 grid() #绘制网格

 points(x, y) #绘制点

 注意：在绘制图形之前，网格必须先绘制。如果先画图形，网格将覆盖图形或者部分隐藏它们。

改变线的类型

 参数lty

 lty=“solid”或者lty=1(默认)

 lty=“dashed”或者lty=2

 lty=“dotted”或者lty=3

 lty=“dotdash”或者lty=4

 lty=“longdash”或者lty=5

 lty=“twodash”或者lty=6

 lty=“blank”或者lty=0(抑制绘图)

改变线的宽度

 参数lwd，默认值为1

 plot(x, y, type=“l”, lwd=2)

改变图的颜色

 col

 colors<-ifelse(x>0, “black”, “gray”)

 plot(x, type=“h”, lwd=3,col=colors)

绘制回归线

 绘制数据点对，同时需要添加一条说明它们线性回归关系的线。

 m<-lm(y~x)

 plot(y~x)

 abline(m) #绘制最佳拟合回归线

添加垂直线和水平线

 abline(v=x)

 abline(h=y)

 m<-mean(CJD$C)

 abline(h=m)

多组散点图

 plot(x, y, pch=as.integer(f))

 with(CJD,plot(CJD$NO,CJD$C,pch=as.integer(as.factor(CJD$grade))))

绘制多个数据集

 使用plot或者curve来初始化图形

 lines和points来添加额外的数据集

 xlim<-range(c(x1,x2))

 ylim<-range(c(y1,y2))

 plot(x1,y1,type=“l”,xlim=xlim,ylim=ylim)

 lines(x2,y2,lty=“dashed”)

图

 条形图barplot()

 barplot(c(height , height ,…, height))

 不同类型数据之间的比较

 直方图hist()

 在函数hist中，必须决定为了容纳数据需要创建多少直方块

 samp<-rgamma(500,2,2)

 hist(samp, 20,prob=T)

 lines(density(samp)) #近似估计样本密度

 箱线图boxplot()

 data(UScereal, package="MASS")

 boxplot(sugars ~ shelf, data=UScereal)

 提供可视化总结，用于因子数据

 summary()

 boxplot(CJD$JAVA)

 boxplot(JAVA~SEX,data=CJD)

 正态Q-Q图qqnorm()

 data(Cars93,package = "MASS")

 qqnorm(Cars93$Price)

 qqline(Cars93$Price)#添加对角线

 判断数据是否服从正态分布

 绘制函数curve()

 curve(sin, -3, +3)

 -3到+3区间内sin函数的值

 curve可以绘制任何接受一个参数并且返回一个值的函数

小技巧

 par(ask=TRUE)#图形暂停覆盖

 par(mfrow=(c(N,M)))#一页显示多个图

 par(mfcol=(c(N,M)))#一页显示多个图

 win.graph()#打开第二个图形窗口

 savePlot(filename=“.ext”,type=“type”)#将图保存到当前目录

 help(Devices)查看可用类型列表

图的类型

 散点图plot

 条形图barplot

 箱线图boxplot

 直方图hist

 正态Q-Q图qqnorm

 函数curve

图的属性参数

 main：标题

 xlab：x轴标签

 ylab：y轴标签

 legend()：添加图例

 lty：线的类型

 lwd：线的粗细

 col：线的颜色

 abline(v=x)：添加垂直线

 abline(h=y)：添加水平线

软件包graphics 散点图plot 条形图barplot

箱线图boxplot 直方图hist 正态Q-Q图qqnorm

函数curve 散点图

如果在x和y之间有任何关系，plot是看清数据间关系的快捷方法plot(x,y)

在图形窗口绘制(x,y)数据对的图形

plot(CJD$NO,CJD$C)

plot(CJD[,4:6])

plot(CJD$C, type=“l”)

添加网格

grid() #绘制网格

points(x, y) #绘制点

注意：在绘制图形之前，网格必须先绘制。如果先画图形，网格将覆盖图形或者部分隐藏它们。

改变线的类型

参数lty

lty=“solid”或者lty=1(默认)

lty=“dashed”或者lty=2

lty=“dotted”或者lty=3

lty=“dotdash”或者lty=4

lty=“longdash”或者lty=5

lty=“twodash”或者lty=6

lty=“blank”或者lty=0(抑制绘图)

改变线的宽度

参数lwd，默认值为1

改变图的颜色

col

colors<-ifelse(x>0, “black”, “gray”)

plot(x, type=“h”, lwd=3,col=colors)

绘制回归线

绘制数据点对，同时需要添加一条说明它们线性回归关系的线。

m<-lm(y~x) plot(y~x)

abline(m) #绘制最佳拟合回归线

添加垂直线和水平线

abline(v=x) abline(h=y)

m<-mean(CJD$C) abline(h=m)

多组散点图

plot(x, y, pch=as.integer(f))

with(CJD,plot(CJD$NO,CJD$C,pch=as.integer(as.factor(CJD$grade))))

绘制多个数据集

使用plot或者curve来初始化图形

lines和points来添加额外的数据集

xlim<-range(c(x1,x2))

ylim<-range(c(y1,y2))

plot(x1,y1,type=“l”,xlim=xlim,ylim=ylim)

lines(x2,y2,lty=“dashed”)

条形图barplot()

barplot(c(height , height ,…, height))

直方图hist()

在函数hist中，必须决定为了容纳数据需要创建多少直方块

samp<-rgamma(500,2,2)

hist(samp, 20,prob=T)

lines(density(samp)) #近似估计样本密度

箱线图boxplot()

data(UScereal, package="MASS")

boxplot(sugars ~ shelf, data=UScereal)

提供可视化总结，用于因子数据

summary()

boxplot(CJD$JAVA)

boxplot(JAVA~SEX,data=CJD)

正态Q-Q图qqnorm()

data(Cars93,package = "MASS")

qqnorm(Cars93$Price)

qqline(Cars93$Price)#添加对角线

判断数据是否服从正态分布

绘制函数curve()

curve(sin, -3, +3)

-3到+3区间内sin函数的值

curve可以绘制任何接受一个参数并且返回一个值的函数

小技巧

par(ask=TRUE)#图形暂停覆盖

par(mfrow=(c(N,M)))#一页显示多个图

par(mfcol=(c(N,M)))#一页显示多个图

win.graph()#打开第二个图形窗口

savePlot(filename=“.ext”,type=“type”)#将图保存到当前目录

help(Devices)查看可用类型列表

main：标题

xlab：x轴标签

ylab：y轴标签

legend()：添加图例

lty：线的类型

lwd：线的粗细

col：线的颜色

abline(v=x)：添加垂直线

abline(h=y)：添加水平线