mon document R

Yongcheng MA

16 janvier 2022

Table des matières

1 R

2 opération de données

2.1 vectoriel

— créer un vectoriel

```
        a <- c(1:3)
        b <- seq(from=0,to =100,by =2)
        d <- rep(1,4)

        a <- append(a,10)
        a <- append(a,4,after=30)
        a <- replace(a,2,10)
        a[-2] #  supprimer un element
        sum(a)
      sort(a)
      rev(sort(a))  #sum(),max(),min(),
            #range(),mean(),var(),sort(),
            #rev(),prod(),sd()

      sample(x,size,replace=F)
      rnorm(10)
      runif(20,min=0,max =100)

```

2.2 matrice

— générer une vectoriel aléatoire

```
      a <- rnorm(16)
      class(a)
      mode(a)

```

— créer une matrice

```
y <-matrix(5:24,nrow=4,ncol=5,byrow=TRUE)
x <- c(2,45,68,94)
rnames <- c("R1","R2")
cname <- c("C1","C2")
matrice <- matrix(x,nrow=2,ncol=2,byrow=TURE,
          dimnames=list(rnames,cnames))

```

— rbind cbind

```
x1 = c(2,4,6,8,0)
x2 = c(1,3,5,7,9)
rbind(x1,x2) # fusionner les lignes
cbind(x1,x2) # fusionner les colonnes

```

— which

```
a = c(2,3,4,2,5,1,6,3,2,5,8,5,7,3)
which.min(a)
which(a==2)
which(a>5)
a[which(a>5)]

```

— deux matrices multiplie

```
a =  matrix(1:12,nrow=3,ncol=4) \\
b =  matrix(1:12,nrow=4,ncol=3) \\
a \% * \% \\

```

— diag(a)

```
a =  matrix(1:16,nrow=4,ncol=4)
diag(a)
diag(diag(a))
diag(4)

```

2.3 array

— authentifier le type

```
x = c(1:6)
is.vector(x)
is.array(x)

```

— créer une array,array() :

```
dim1 <- c("A1","A2","A3")
dim2 <- c("B1","B2")
dim3 <- ("C1","C2","C3","C4")
d <- array(1:24,c(3,2,4),
     dimnames=list(dim1,dim2,dim3))

#comment prendre l'element dans la array
d[1,2,3]
```

```
        d[,,2]
        sum(d[1,,])

```

— attach()

```
        mtcars$mpg
        attach(mtcars)
        mpg
        deattach(mtcars)
        ls() #consulter le object dans le memoir de R

```

— importer des données par le clavier

```
        mydata <- data.frame(age= numeric(0),
                  gender=character(0),
                  weigth=numeric(0))
      mydata <- edit(mydata) ou fix(mydata)

```

2.4 data frame

— data.frame

```
    x1 <- c(10,13,45,26,23,12,24,78,23,43,31,56)
    x2 <- c(20,65,32,32,27,87,60,13,42,51,77,35)
    x <- data.frame(x1,x2)
    x <- data.frame('poits' = x1 , 'prix' =x2)
    plot(x) <- peintre la graphe

```

— créer un data frame :

```
      name <-c('kg','ky','kq')
      sex <- c('m','m','f')
      age <- c(22,21,24)
      mat <- data.frame(name,sex,age)

      mat$score <- c(80:82) #ajouter une propriete
      mat[which(mat$score == 82),] #consulter des lignes
                    #qui correspondant la condition

```

— convertir la matrice à data frame

```
      mat2 <- matrix(c(1:12),nrow = 3)\\
      mat3 <- as.data.frame(mat2)

```

merge

— union deux data frame merge() :

```
      mat0 <- data.frame(name=name,
                weiht = c(60:62),
                age=v(20,23,24))
      merge(mat,mat0,
```

```
          by.x = 'age',
          by.y = 'age')

```

— attacher des data frame :

```
rm(name)
attach(mat)
rbind(mat,mat0) #rbind() union tous les lignes,
        #mais les colonnes identique,
        cbind() union tous les colonnes,
        #mais les lignes identiques:
cbind(mat,mat0)
name

```

— lapply(),sapply()

```
lapply(mat0[-1,-2],sum) #calculer la somme de chaque
colonne,
             #retourner un list
sapply(mat0[-1,-2],sum) #calculer la somme de chaque
colonne,
             #retourner un vectoriel

```

2.5 lire les fichier

— charger fichier excel par rodbc

```
local({ pkg <- select.list(
                sort(.packages(all.available = TRUE)),
                 graphics=T$ + if(nchar(pkg))
library(pkg, character.only=TRUE )})
library(RODBC)
z <- ordbcConnectExcel('chemain de fichier de excel')
w <- sqlFetch(z,"Sheet1")

```

— lire et écrire dans le fichier

```
  data <- read.table("chemin de fichier txt",header=TRUE,
sep=",") \\
  data <- read.csv("chemin de fichier de csv ",header =
TRUE, sep=",") \\
  write.csv(data,'chemin de fichier',row.names=F)\\
  write.table(data,'chemin de fichier',sep=' ') \\

```

2.6 écrire des fichier

— seq()

```
  num <- seq(10378001,10378100)
  x1 <- round(runfi(100,min =80, max =100))
  x2 <- round(rnom(100,mean = 80,sd =7))
  x3 <- round(rnom(100,mean = 83,sd =18))
```

```
      x3[which(x3 > 100)]  <- 100
      x <- data.frame(num,x1,x2,x3)
      write.table(x,file='chemain de fichier qu'on veux enregistrer',
              col.names=F,row.names=F,sep=' ' )

```

— sauvegarder des données dans un fichier

```
   png('france.png')
   map('france')
   dev.off()

```

```
   setwd('chemin de fichier')
   image <- c(1,2,3)
   for (i in c(1:10)) {
      filename <- paste0(i,'jep')
      jpeg(filename)
      plot(d)
      dev.off()

```

2.7 function

2.7.1 la function de statistique

— mean sum max min var prod sd

```
    x = c(1:100)
    sum(x)
    max(x)
    min(x)
    var(x)
    prod(x) -> factoriel\\
    sd(x)
    abs()
    sqrt()
    log2()
    log10()
    log(c,base=)
    log(a,base=exp(1))

```

calculer en math

—

$$+, -, *, /$$

```
     /* +, - ,* , /, %%, %/%,sqrt(),prod()
       abs(),sign(),log2(),log10(),
       log(number,base=3),
     log(numner,base=exp(1)), ?identical  */

     cumsum(1:5)
     cumprod(1:5)
```

```
8
9     a <-c(rep(1,3),
10        rep(2,3),
11        rep(6,7),
12        1:10)
13    a
14    unique(a)
15
16      sin(),
17      cos(),
18      tan()
19
```

2.7.2 tirer des donnée

— rev sort ,order

```
1           a = 1:20
2           rev(a)
3           a = c(2,3,4,2,5,1,6,3,2,5,8,5,7,3)
4           b <- sort(a)
5           rev(b)
6           index <- order(a,decreasing=FLASE) or order(-a)
7           a[index]
8
9           dat <- data.frame(name=c('kg','ky','kq'),score=c
    (89,90,79))
10          dat[order(dat$score,decreasing=TRUE),]
11
```

2.7.3 transposition

— t() transposition

```
1           a = matrix(1:12,nrow=3,ncol=4)
2           a
3           t(a)
4
```

— eigen() chercher le caractéristique de la matrice

```
1     a <- diag(4) + 1
2     a.e <- eigen(a,symmetric=T)
3     a.e
4     a.e$vectors %*%diag(a.e$values) %*% t(a.e<$vectors)
5
```

2.7.4 boucle for,while

— for

```
1       for ( i in 1:59)
2       {
3         a[i] = i * 2 + 3
4       }
```

```
5
6       a[1] = 5
7       i = 1
8       while(a[i] < 121){
9       i = i +1;
10      a[i] = a[i-1] + 2
11      }
12
13        e <- 1
14        i <- 1
15        while(1/prod(1:i) -1 / prod(1:(i+1)) > 0.001){
16          print(i)
17          e <- e+1 / prod(1:i)
18          i <- i+1
19        }
20
```

— récursif

```
1       f <- function(a=1,b=1){
2         d <- a + b
3         a <- b
4         b <- d
5         if (d>1000)
6         {
7           return(d)
8         }
9         f(a,b)
10      }
11    s <- f()
12    s
13
```

2.7.5 la tube

la function de tube

1 utiliser function de tube,pour l'équation suivante %>%

$$f(x) = \sin((x+1)^2)$$

```
1         a %>% f(b) => f(a,b)
2         a %>% f(b,.,c) => f(b,a,c)
3         f <- function(x){
4             return(x+1)
5           }
6           f2 <- function(x){
7                 return(x**2)
8             }
9           f3 <- function(x){
10              return(sin(x))
11            }
12      x <- 4
13      x %>% f() %>% f2() %>% f3()
14
```

2 utilise unite

```
        library(tidyr)  #importer package
        data <-Sys.Date()
        date <- as.Date('2017-6-22') + 0:14
        hour <- sample(1:24,15)
        min <- sample(1:60,15)
        second <- sample(1:60,15)
        dat <- data.frame(date,hour,min,second)
        dat %>% unite(dateHour,date,hour,sep=' ')%>%
        unite(dateHourMinute,dateHour,min,sep=':')%>%
        unite(DH,dateHourMinute,second,sep=':')
        # identique que au dessus
        dat %>% unite(dateHour,date,hour,sep=' ')%>%
        unite(dh,dateHour,min,second,sep=':')

```

2.7.6 apply

apply()
— function apply

```
      mean(x)
      cloMeans(x)
      cloMeans(x)[c('x1','x2','x3')]
      apply(x,2,mean) # 2 reprisent dans le colum,
              #1 reprisent dans le ligne
      apply(x[c('x1','x2','x3')],1,sum) #- 2 reprisent
    dans le colum,
                        #1 reprisent dans le ligne

      x <- cbind(3,c(1:5,4:1))
      apply(x,1,mean)
      x <- array(c(1:24),dim=c(2,3,4))
      apply(x,3,mean)

```

2.7.7 function édition

— la function simple

```
        f <- function(x){
          print(x)
        }
        f('bonjour')

```

— défénier une function

```
        mydata <- function(type){
          swith(type,long = format(sys.time(),"%A %B %d %Y
    "),\\
          short = format(sys.time(),"%m-%d-%y"),\\
          cat(type,"is not recoginzed type")
          }
        mydata("long")

         sum <- function(num){
```

```
            x <- 0
            for(i in 1:num){
            x <- x + i
           }
           return(x)
          }

```

— calculer le produit de deux matrices

```
    mat0 <- matrix(c(1:12),nrow = 3,byrow = T)
    mat1 <- matrix(c(1:24),nrow = 4,byrow = T)
    f <- function(mat0,mat1){
        xcol = dim(mat0)[2]
        yrow = dim(mat1)[1]
        if (xcol != yrow){
          print('dexu matrices ne peuvent pas munipuler')
          return(0)
        }
        n = dim(mat0)[1]
        m = dim(mat1)[2]
        mat_pro = matrix(0, nrow =n, ncol = m)
        for (i in c(1:n) ){
          for (j in c(1:m)) {
              mat_pro[i,j] =sum(mat0[i,] * mat1[,j] )
          }
        }
        return(mat_pro)
    }
    mrt <- f(mat0,mat1)
    print(mrt)
    mat0 % * % mat1

```

2.7.8 rnorm,runif,normalisation

— rnorm

```
          n <- 100
          a <- rnorm(n,mean=0,sd=1)

          n <- 100
          a <- rnorm(n,mean=5,sd=2) #mean(moyeen) sd(
    variance)

```

[language= python]

— runif

```
          n <- 100
          a <- rnorm(n,min=0,max=1) #min(valeur minimale)
    max(valeur maximum)

```

[language= python]

2.7.9 table

— table()

```
    a <- c(1,4,6,8,3,4,6,6,6,8,6,1)
    b <- table(a)
    b == max(b)
    names(b)
    as.numeric(names(b)[b == max(b)])

```

2.7.10 chercher des solution pour équation

— calculer valeur d'éauqtion

$$y = a * x + b$$

```
      f <- function(x,a,b){
          return(a*x + b)
          }
      root <- uniroot(f,
              c(-10,10),
              a=5,
              b=10,
              tol=0.0001) #la marge d'erreur
      root$root
      f <- function(x,a,b,c){
            return a*(x**2) +b*x +c
            }
        uniroot(f,c(-10,10),a=5,b=10,c=6,tol=0.0001)

```

— chercher des valeurs d'équation

$$\begin{cases} 3x_1 + 5x_2 = 4 \\ x_1 + 2x_2 = 1 \end{cases}$$

```
      f <- matrix(c(3,5,1,2),nrow=2,byrow=TRUE)
      rf <- matrix(c(4,1),nrow=2)
      solve(f,rf)

```

2.7.11 solve

— solve() régler un système d'équation

```
    a <- matrix(rnorm(16),4,4)
    b <- c(1:4)
    solve(a,b)

```

2.7.12 authendifier deux type de valeur

— identical

```
          identical(1,NULL)
          identical(1,is.integer())
          is.integer(1)
          identical('1',is.character('1'))

```

[language= python]

2.7.13 cumulation de sum

— cumsum(), cumprod()

```
        cumsum(1:15)
        cumprod(1:15)

```

[language= python]

2.7.14 unique,which,in

— unique()

```
        a <- c(rep(1,3),rep(2,3),rep(4,7),1:10)
        b <- unique(a)

```

[language= python]

— which()

```
        a <- c(rep(1,3),rep(2,3),rep(4,7),1:10)
        c <- a[which(d>5)]

        a <-c(1:10)
        b <-c(1:10)
        dat <- data.frame(a,b)
        dat[which(dat[,1] > 5),]
        dat[which(dat[,1] > 5),1]

```

[language= python]

— %in%

```
        x %in% y # verifier x dans le y

```

[language= python]

2.8 executer des fichier scripte de R

— source()

```
    path <- 'repertoire de fichier'
    setwd(path)
    getwd()
```

```
    list.files()
    source('nom de scripte de R')
    source("chemain de scripte de R")

```

2.9 facteur

— créer un facteur

```
    a <- factor(c('A','B','C','A','B'))
    a <- factor(c('A','B','C','A','B'),
          levels =c('A','B'))
    a <- factor(c('A','B','C','A','B'),
          levels =c('A','B','C'),
          labels= c('Tres Bien','Bine','Mauvais'))

    a <- factor(c('A','B','C','A','B'),
          exclude ='A'))

    score <- c('A','B','A','C','B')
    score1 <- ordered(score,
            levels=c('C','B','A'))

    dataset <- c(....)
    examen <- cut(dataset,breaks=3)

```

— tapply()

```
    gender <- c('f','m','m','m','f')
    age <- c(12,35,25,12,25)
    tapply(age,gender,mean)

```

2.10 chaine caractère

— strsplit()

```
    s <- '123,234,56'
    strsplit(s,',')
    unlist(strsplit(s,','))

      s <- 'a b c'
    s1 <- 'w x v'
    paste(s,s1,sep=',')
    paste0(s,s1) #paste() paste0()

```

— nchar(),length()

```
    s <- 'a b c'
    nchar(s)
    length(s)

```

— substr() substring()

```
s <- 'a b c'
substr(s,2,3)
substring(s,2)

```

2.11 nettoyer des données

2.11.1 examiner des valeur manquant

— examiner des valeur manquant complete.cases() :

```
          a <- matrix(c(1:6,NA,8,9),nrow=3)
        complete.cases(a);
        a[complete.cases(a)]
        a[complete.cases(a),]
        b <- c(1,2,NA)
        complete.cases(a,b);
        na.omit(a)

```

2.11.2 valeur manquant

— remplie des valeur manquant

```
        x <- c(1,NA,2,NA,3)
      x[!is.na(x)]
        y <- c('a','b',NA,'c',NA)
        z <-complete.cases(x,y);
        x[z]
        y[z]
        g <- complete.cases(airquality)
        airquality[g,]

```

2.11.3 explorer

— explorer des données

```
        mat <- iris[1:4]
        mean(mat[,1])
        median(mat[,1])
        quantile(mat[,1])
        range(mat[,1]) #afficher le valeur maximum et minimum
        var(mat)
        sd(mat[,1])
        summary(mat)

```

2.12 debug

— test code

```
        cou <- function(count)
            {
              s <- 0
              i <- 1
              while(i < count +1)
              {
              s <- s + i
              i <- i + 1
              browser()
              }
              return(s)
            }
        debug(cou)
        cou(10)
        appuyer n #next
        appuyer i # afficher la valeur de variable
        print(s) #afficher la valeur de variable
        q # quitter debug
        browser() # appuyer c pour debug le boucle


```

3 marche learning

3.0.1 régression linéaire

— calculer le paramètre de régression de linéaire

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}(x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}, \beta_0 = \bar{y} - \beta_1 * \bar{x} \tag{1}$$

— régression linéaire

```
        y <-matrix(5:24,nrow=4,ncol=5,byrow=TRUE)
        x <- c(2,45,68,94)
        rnames <- c("R1","R2")
        cname <- c("C1","C2")
        matrice <- matrix(x,nrow=2,ncol=2,byrow=TURE,dimnames
    =list(rnames,cnames))
        res <- lm(para1 ~ para2, data)

```

3.0.2 PCA

— scale

```
            sl <- scale(iris[,1])
            pw <- scale(iris[,4])
            model <- lm(pw~sl)
            summary(model)
            model <- lm(pw~sl-1) #-1 (supprimer la valeur
    constante)
```

```
6             plot(pw,sl,col='green')
7
8             sc <- scale(swiss)
9             pri <- princomp(sc,cor=TRUE)
10            summary(pri)
11
12            screeplot(pri,type='line')
13            pre <- predict(pri)
14            summary(pre)
15            y <- eigen(cor(sc))
16            y$values
17            scores <-(y$values[1] * pre[,1] +y$values[2] *
    pre[,2] +
18                  y$values[3] * pre[,3] +y$values[4] * pre[,4]
     ) / sum(y$values)
19            plot(scores)
20
```

3.0.3 régression multiple linéaire

— régression multiple linéaire

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_k * x_k + \epsilon$$

pour quelque soit

$$x_i$$

on a

$$x_i = (x_{1i}, x_{2i}, x_{3i}, ..., x_{ki}), x_1 = (x_{11}, x_{21}, x_{31}, ..., x_{k1})$$

marge d'erreur

$$\epsilon_i = y_i - (\beta_0 + \beta_1 * x_{i1} + \beta_2 * x_{i2} + ... + \beta_k * x_{ik})$$

```
1         rm(list=ls()) # vider la variable de la memoire
2         s.l <- scale(iris[,1])
3         s.w <- scale(iris[,2])
4         p.l <- scale(iris[,3])
5         p.w <- scale(iris[,4])
6
7         model <- lm(s.l,~s.w + p.l + p.w)
8
9         plot(model,1,col='red') # 1.merge d'erreur
10        plot(model,2,col='red') # 2 qq graphe
11        s.l <- 0.34*s.w + 1.511p.l - 0.512 *p.w
12
13
14        blood <- data.frame(
15              x1 = c(76.0,91.5,85.5,82.5,79.0,80.5,
16                   74.5,79.0,85.0,76.5,82.0,95.0,92.5)
17              x2 = c(50,20,20,30,30,50,60,50,40,55,40,40,20)
18              y =  c(120,141,124,126,112,125,
19                   123,125,132,123,132,155,147))
20        lm.sol <- lm(y~x1+x2,data=blood)
21        summary(lm.sol)
```

```
22      s <- lm(y ~ ., data = blood)
23      summary(s)
24      toothpaste <- data.frame(
25              x1=c(-0.05,0.25,0.06,0,0.25,0.20,0.15,
26
    0.05,-0.15,0.15,0.20,0.10,0.40,0.45,0.35,
27
    0.30,0.50,0.50,0.40,-0.05,-0.05,-0.01,
28                 0.20,0.10,0.50,0.60,-0.05,0,0.05,0.55)
29              x2 = c(5.50,6.75,5.00,4.98,7.25,5.50,7.00,
30                     6.50,6.75,25.5,25.6,6.00,6.50,6.25,
31                     7.00,6.90,6.80,7.10,7.00,6.80,6.50,
32
    6.25,6.00,6.50,7.00,6.80,6.50,5.75,5.80,6.80)
33                          y = c
    (7.38,8.51,9.52,7.50,9.33,8.28,8.75,
34
    7.87,7.10,8.00,7.89,8.15,9.10,8.86,8.90,
35
    8.87,9.26,9.00,8.75,7.95,7.65,7.27,8.00,
36
    8.50,8.75,9.21,8.27,7.67,7.93,9.26)
37                          )
38      lm.sol <-lm(y~x1+x2,data = toothpaste)
39      summary(lm.sol)
40      lm.new <-update(lm.sol,.~.+I(x2**2)) #mettre a jour
    notre modele
41      summary(lm.new)
42      lm2.new <-update(lm.new,.~.-x2) #supprimer facteur x2
43      summary(lm2.new)
44      lm3.new <-update(lm2.new,.~.+x1 * x2) #ajouter un
    facteur x1 * x2
45      summary(lm3.new)
46      lm.step <- step(lm3.new)
47      drop(lm.step)
48
```

3.0.4 non-linéaire

— la modèle non-linéaire

```
1       x <- c
    (1.5,2.8,4.5,7.5,10.5,13.5,15.1,16.5,19.5,22.5,24.5,26.5)
2       y <- c
    (7.0,5.5,4.6,3.6,2.9,2.7,2.5,2.4,2.2,2.1,1.9,1.8)
3       lm.log <- lm(y~ log(x))
4       summary(lm.log)
5       lines(x,exp(fitted(lm.log)))
6       lm.exp <- lm(log(y) ~ x)
7       summary(lm.exp)
8       lines(x,exp(fitted(lm.exp)))
9       lm.pow <- lm(log(y) ~ log(x))
10      summary(lm.pow)
11      lines(x,exp(fitted(lm.pow)))
12
```

3.0.5 K-means

— k-means

```
        c1 <- (14,22,15,20,30,18,32,13,
                 23,20,21,22,23,24,35,18)
        c1 <- (15,28,18,30,35,20,30,15,
                 25,23,24,25,26,27,30,16)
        kc <- kmeans(dat,3)
        summary(kc)
        kc$centers
        kc$cluster
        qplot(dat[1,],dat[,2],colour=kc$cluster)

```

4 connecter database

4.0.1 mysql

11 R connecter base de donnée mysql,

```
            /**1. install.packages("RODBC"). */

            library(RODBC)
            myconne <- odbcConnect("nom de data base",uid="
    root",pwd="password")
            data1 <-sqlFetch(myconn,"nom de table")

           data2 <- sqlQuery(myconn,"select * from nom de
    table")
             close(myconn)

```

— examiner la modèle est normalisation

```
        shapiro.test(x$x1)
        shapiro.test(x$x3)
        y.res <- residuals(lm.sol)
        norell <- data.frame(x=0:5,
               n = rep(70,6),
               success = c(0,9,21,47,60,63)
        norell$Ymat<-cbind(norell$success,
                  norell$n-norell$success)

        glm.sol <- glm(Ymat~x,family = binomial,
             data = norell)
        summary(glm.sol)

```

5 graph

— barplot

```
        colors()# lister toutes les couleurs dans le R
```

```
    palette()#lister toutes les couleurs dans la palette d'
  actuelle
    barplot(c(88,78,100),
       names.arg=c('eric','alen','fabien'),
       ylim=c(0:100),
       col=rainbow(3),
      legend.text = c('eric','alen','fabien'))

    hist(iris[,1],freq=T)
    hist(iris[,1],freq=F)
    pie(c(1,2,3),lables=c('chine','france','etat-unis'),
             radius=0.5)
    boxplot(iris[,c(1:4)],
      notch=T,
      col=rainbow(4),
      names=c('sl','sw','pl','l'))

```

— qplot()

```
      dat <- diamonds[sample(nrow(diamonds),200),]
      names(dat)
      qplot(dat$carat,dat$price)
      unique(dat$color)
      qplot(carat,price,data=dat,colour=color)
      qplot(carat,price,
            data=dat,colour=color,
            shape=mat$cut)

      qplot(carat,price,
              data=dat,colour=color,
            shape=mat\$cut,alpha=I(1/10))

```

— comment peintre un image

```
      apply(x[c('x1','x2','x3')],1,sum) # 2 reprisent dans
  le colum,
                   #1 reprisent dans le ligne

      whitch.max(x[c('x1','x2','x3')],1,sum) #chercher plus
   grand de note

      x$num[whitch.max(x[c('x1','x2','x3')],
               1,sum)] #chercher le numero d'etudiant
                  # qui est plus grand de note
      hist(x$x1)
      plot(x1,x2)
      plot(x$x1,x$x2)
      pie(table(x$x1)) # circle
      boxplot(x$x1,x$x2,x$x3) \\
      boxplot(x$x1,x$x2,x$x3,horizontal = T)
      boxplot(x[2:4],
      col=c('red','green','blue'),
      notch = T)
      stars(x[c('x1','x2','x3')])
      stars(x[c('x1','x2','x3')],
      full = T,draw.segment = T)

```

```
        stars(x[c('x1','x2','x3')],
            full = F,draw.segment = T)

        faces(x[c('x1','x2','x3')]) #image de face install
    pakage aplpack

        faces2(x) #autre face de image,
          #install pakage TeachingDemo
        stem(x$x1) #image de tige et feuille
        qqnorm(x1)
        qqline(x1)
        qqnorm(x3)
        qqline(x3)
        plot(x$x1,x$x2,
           main='titre de image',
           xlab='x de lable',
           ylab='y de label',
           xlim = c(0,100),
           ylim=c(0,100),
           xaxs='i', le sytel de x
           yaxs='i', le sytel de y
           col = 'red',\\
             pch = 19 le sytel de point de image)

```

```
        a = c(2,3,4,5,6)
        b = c(4,7,8,9,12)
        plot(a,b,type='l') utile un fil lie les points
        plot(rain$Tokyo,type='l',col='red',
            ylim=c(0,300),
            main = 'le pleut de mois dans la ville',
              xlab = 'mois d'annee',
                ylab = 'quantite de pleut(mm)',
              lwd = 2 ) la large du fil

        lines(rain$NewYork,type='l',col='blue',lwd=2)
        lines(rain$London,type='l',col='green',lwd=2)
        lines(rain$Berlin,type='l',col='orange',lwd=2)

```

— density()

```
        plot(density(rnorm(1000)))

```

— consulter la dataset in system de Rstudio *data() on peut trouver mtcars*
— heatmap

```
        heatmap(as.matrix(mtcars),Rowv =NA,
              Colv =NA, col = heat.colors(256),
              scale = 'column',margins = c(2,8),
              main = 'Car caracteristique de model')

```

— par

```
        par(mfrow = c(3,1)) #mettre 3 lignes 1 colonne
                #pour mettre les trois graphes
        plot(x1,x2)
```

```
        plot(x2,x3)
        plot(x3,x1)
        install pakages sctterplot3d

        sctterplot3d(x[2:4])
        x <-y<-seq(-2*pi,2*pi,pi/15)

        f <- function(x,y) sin(x)*sin(y)
        z <- outer(x,y,f)

        contour(x,y,z,col='red')
        persp(x,y,z,theta=30,phi=30,expand=0.7,col='lightnlue
    ')

        source('chemin de scripte de R')
        unison(x[2:4])
        install pakages maps et geosphere
        map('state',interior=FALSE)
        map('state',bounday = FALSE,col='red',add=TRUE)
        map('world',fill=TRUE,col=heat.colors(10))
        map('state')
        map('world')

```

— la mode de probabilité

```
        rnorm(n,mean=0,sd=1)
        rexp(n,rate=1)
        rgamma(n,shape,scale=1)
        rpois(n,lambda)
        rweibull(n,shape,scale=1)
        recauchy(n,location=0,scale=1)
        rbeta(n,shape1,shape2)
        rt(n,df)
        rf(n,df1,df2)
        rchisq(n,df)
        rbinom(n,size,prob)
        rgeom(n,prob)
        rhyper(nn,m,n,k)
        rlogis(n,location=0,scale=1)
        rlnorm(n,meanlog)0,sdlog=1)
        rnbinom(n,siez,prob)
        runif(n,min=0,max=1)
        rwilcox(nn,m,n),
        rsignrank(nn,n)

```

```
        plot(iris[1,2])
        i1 = iris[which(iris$Species=='setosa'),1:2]
        plot(i1)
        cor(i1[1],i1[2]) #consulter la coefficient de
    correlation

```

```
        h <- c
    (171,175,159,155,152,158,154,164,168,166,159,164)
        w <- c(57,64,41,38,35,44,41,51,57,49,47,46)
        plot(w~h+1)
```

```
        lxy <- function(x,y){
          n = length(x);
          sum(x*y) - sum(x)*sum(y) /n;
        }
        b <- lxy(h,w)/lxy(h,h)
        a <- mean(w) - b*mean(h)
        lines(h,a+b*h)
        a <- lm(w~1+h)
        z <- data.frame(x=185)
        predict(a,z)
        predict(a,z,interval='prediction',level=0.95)

```

6 R avec logiciel de bigdata, RHadoop,RHive,RHipe