Exercices

Cyril Dalmasso

11 janvier 2017

Aide à la syntaxe markdown :

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

http://kbroman.org/knitr_knutshell/pages/Rmarkdown.html

Vecteurs

Exercice 1

1. Suite des entiers de 1 à 12

```
1:12
seq(1,12)
```

2. Création du vecteur c(0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0)

```
seq(0.5,5,by=0.5)
seq(0.5,5,length=10)
1:10/2
sort(c(0.5:5,1:5))
```

3. Multiples de 2 compris entre 1 et 50

```
seq(2,50,by=2)
x<-1:50; x[x%%2==0]
```

4. Vecteur contenant 3 fois chacun des dix chiffres

```
rep(0:9,each=3)
```

5. Vecteur contenant une fois la lettre A, deux fois la lettre B, etc.

```
res <- rep(LETTERS,1:26)
length(res)
```

6. Vecteur c("individu 1", "individu 2", ..., "individu 100")

```
vindiv <- paste("individu",1:100)
head(vindiv)
```

Exercice 2

1. Génération d'une séquence de n=1000 bases

```
n <- 10000
myseq <- sample(c("A","T","G","C"),size=n,replace=TRUE)
```

Nombre d'occurrences de chaque lettre

```
sum(myseq=="T")
table(myseq)
```

Indices de la séquence où l'on trouve la lettre "T"

```
which(myseq=="T")
```

2. Vecteur contenant les 100 premiers entiers échantillonnés aléatoirement.

```
v1 <- sample(1:100)
```

Emplacement de la valeur minimale et de la valeur maximale.

```
which(v1==1); which(v1==100)
which(v1 %in% c(1,100))
```

vecteurs x et y des 100 premiers entiers ordonnés dans l'ordre croissant et décroissant.

```
x <- sort(v1)
y <- sort(v1,decreasing=TRUE)
```

Concatenation de x et y, et retrait du seul nombre apparaissant deux fois de suite

```
z <- c(x,y)
z2 <-
z[-(which(diff(z)==0))]
# ou bien z[(diff(z)!=0)]
```

Exercice 3

Création des deux vecteurs

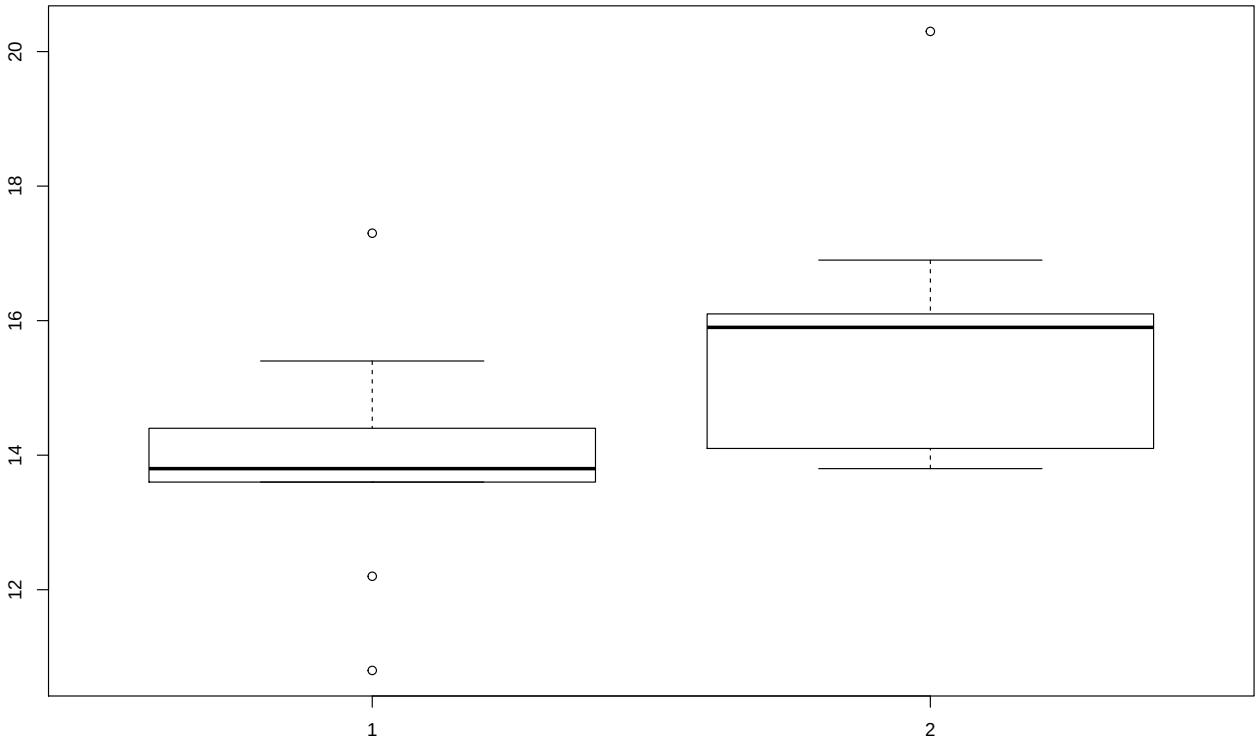
```
 grp1 <- c(14.40 , 13.70 , 14.20 , 17.30 , 13.90 , 13.60 , 15.40 , 10.80 , 12.20 , 13.60)
 grp2 <- c(14.00 , 15.90 , 16.90 , 14.10 , 13.80 , 20.30 , 16.00 , 15.30 , 16.10 , 15.90)
```

1. Calcul des moyennes, médianes, variances et écarts-types pour chaque groupe

```
  res1 <- c(mean(grp1), median(grp1), var(grp1), sd(grp1))
 res2 <- c(mean(grp2), median(grp2), var(grp2), sd(grp2))
 names(res1) <- names(res2) <- c("mean","mediane","variance","ecart-type")
 res1;  res2
```

2. Boîtes à moustaches

```
boxplot(grp1,grp2)
```



Facteurs

Exercice 4

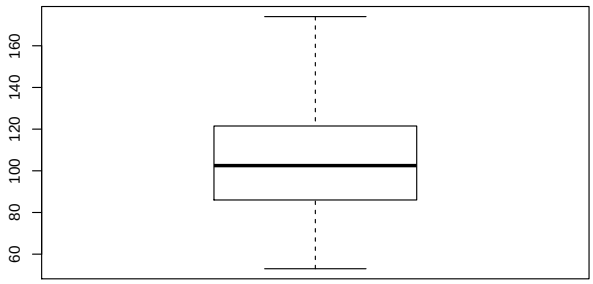
Création des vecteurs de données

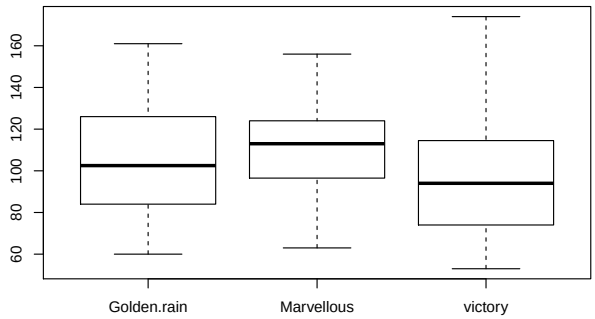
```
variete <- c("victory", "victory", "victory", "victory", "Golden.rain", "Golden.rain", "Golden.rain", "
"Marvellous", "Marvellous", "Marvellous", "Marvellous", "victory", "victory", "victory", "victory", "Go
"Golden.rain", "Golden.rain", "Golden.rain", "Marvellous", "Marvellous", "Marvellous", "Marvellous", "v
"victory", "victory", "victory", "Golden.rain", "Golden.rain", "Golden.rain", "Golden.rain", "Marvellou
"Marvellous", "Marvellous", "victory", "victory", "victory", "victory", "Golden.rain", "Golden.rain", "
"Golden.rain", "Marvellous", "Marvellous", "Marvellous", "Marvellous", "victory", "victory", "victory",
"Golden.rain", "Golden.rain", "Golden.rain", "Golden.rain", "Marvellous", "Marvellous", "Marvellous", "
"victory", "victory", "victory", "victory", "Golden.rain", "Golden.rain", "Golden.rain", "Golden.rain",
"Marvellous", "Marvellous", "Marvellous")
engrais <- c("0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "
"0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.
"0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.
"0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.
"0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.
"0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt", "0.
"0.2cwt", "0.4cwt", "0.6cwt", "0.0cwt", "0.2cwt", "0.4cwt", "0.6cwt")
rendement <- c(111, 130, 157, 174, 117, 114, 161, 141, 105, 140, 118, 156, 61, 91, 97, 100, 70, 108, 12
```

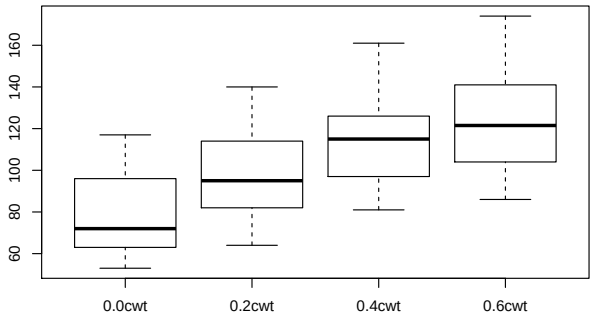
```
96, 124, 121, 144, 68, 64, 112, 86, 60, 102, 89, 96, 89, 129, 132, 124, 74, 89, 81, 122, 64, 103, 132,
70, 89, 104, 117, 62, 90, 100, 116, 80, 82, 94, 126, 63, 70, 109, 99, 53, 74, 118, 113, 89, 82, 86, 104
99, 119, 121)
```

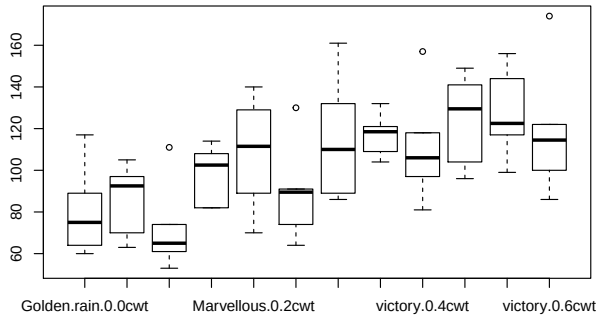
1. Répartition empirique des rendements par sous groupes

```
variete <- as.factor(variete)
engrais <- as.factor(engrais)
par(mfrow=c(2,2))
boxplot(rendement)
boxplot(rendement~variete)
boxplot(rendement~engrais)
boxplot(rendement~variete*engrais)
```









2. Moyennes et résumés numériques

```
tapply(rendement,variete,mean)
tapply(rendement,engrais,mean)
# tapply(rendement,paste(engrais,variete),mean)
tapply(rendement,list(engrais,variete),mean)
tapply(rendement,variete,summary)
tapply(rendement,engrais,summary)
res <- tapply(rendement,paste(engrais,variete),summary)
res <- tapply(rendement,list(engrais,variete),summary)
```

3.

a) Nombre de champs total, par variété, par dose d'engrais et par couple (variété,engrais)

```r
length(rendement)
table(variete)
table(engrais)
table(variete,engrais)
```

b) Même chose en ne conservant que les champs dont le rendement est supérieur au rendement moyen par groupe

```r
sum(rendement>mean(rendement))
myfct <- function(x){
  res <- sum(x>mean(x))
  return(res)
}
tapply(rendement,variete,myfct)
tapply(rendement,engrais,myfct)
tapply(rendement,list(variete,engrais),myfct)
```

c) Même chose en ne conservant que les champs dont le rendement est supérieur au rendement moyen total

```r
myfct2 <- function(x){
  res <- sum(x>mean(rendement))
  return(res)
}
tapply(rendement,variete,myfct2)
tapply(rendement,engrais,myfct2)
tapply(rendement,list(variete,engrais),myfct2)
```

d) Identification de la meilleure combinaison

```r
res <- tapply(rendement,list(variete,engrais),mean)
res <- tapply(rendement,paste(variete,engrais),mean)
names(res[res==max(res)])
```

Matrices, listes, tableaux

Exercice 5

1. Chargement des données

```r
data(iris)
class(iris[,5])
iris2 <- as.matrix(iris[,-5])
class(iris2)
```

2. Dimension de la matrice, valeur max, et indices correspondants

```r
dim(iris2)
max(iris2)
which(iris2==max(iris2),arr.ind=TRUE)
# arrayInd(which(iris2 == max(iris2)),dim(iris2))
```

3. Calcul des moyennes

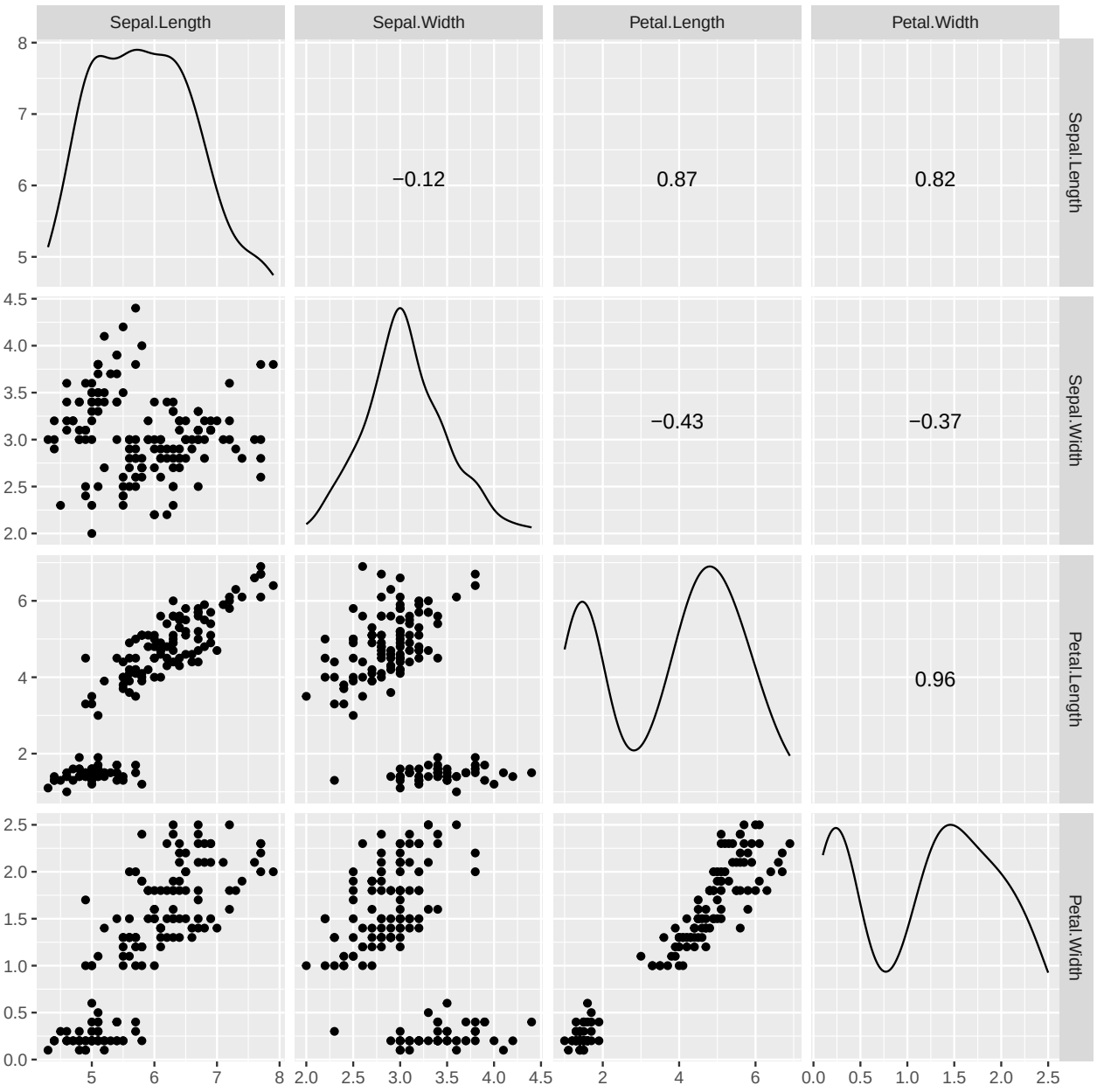
```r
colSums(iris2)/nrow(iris2)
apply(iris2,2,mean)
rowSums(iris2)/ncol(iris2)
apply(iris2,1,mean)
```

Identification des individus ayant les plus grande longueur de sépale et largeur de pétale

```r
which(iris2[,"Sepal.Length"]==max(iris2[,"Sepal.Length"]))
indmax <- which(iris2[,"Petal.Width"]==max(iris2[,"Petal.Width"]))
```

4. Graphe des paires de variables

```r
iris2[indmax, ]
# pairs(iris2)
library(GGally)
ggscatmat(data.frame(iris2))
```



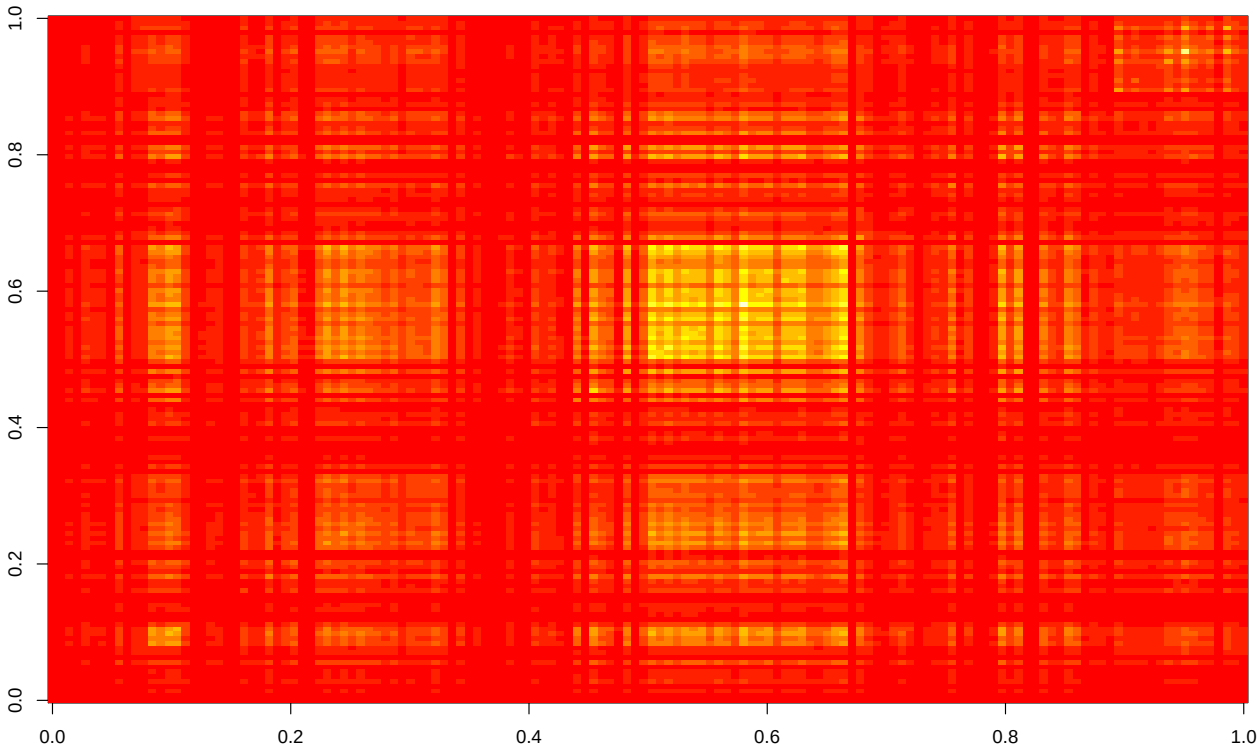
Exercice 6

1. Chargement des données

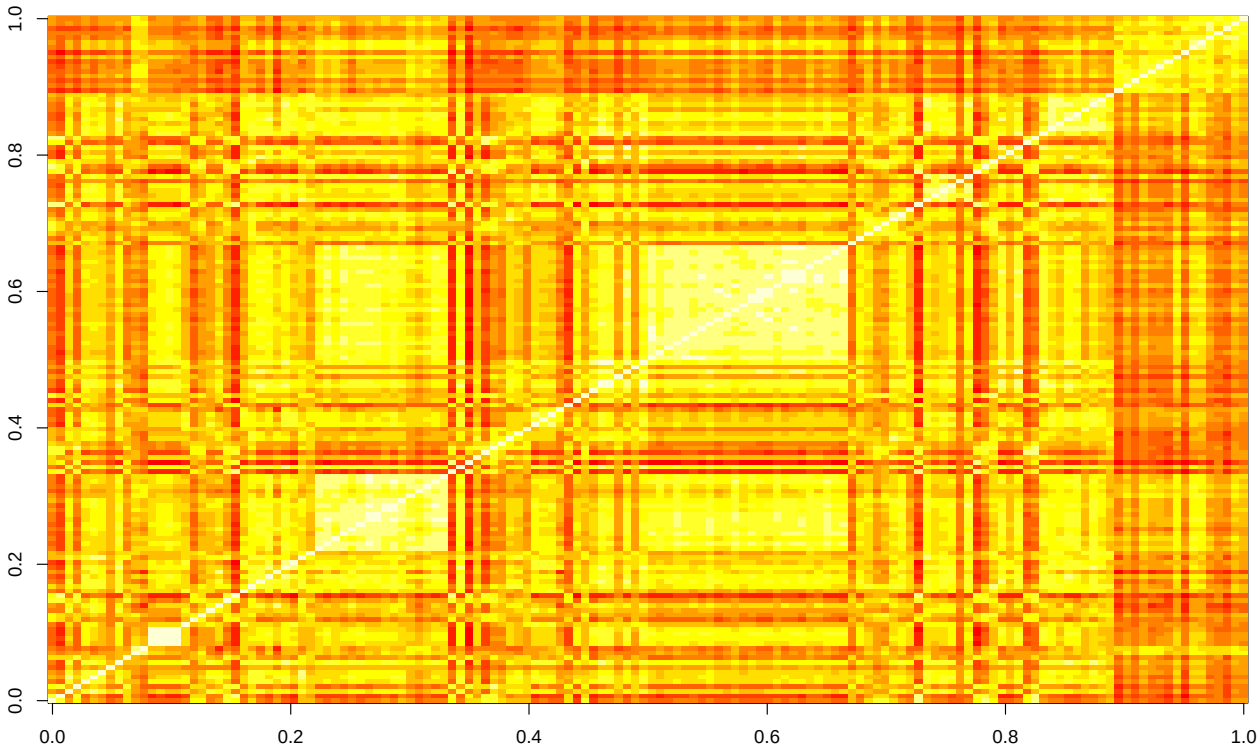
```
setwd("/Users/cdalmasso/IntroductionR/projet1")
microarray <- as.matrix(read.table("http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/14cancer.x

microarray[1:5,1:5]
dim(microarray)
is.matrix(microarray)
```

2. Covariances et corrélations

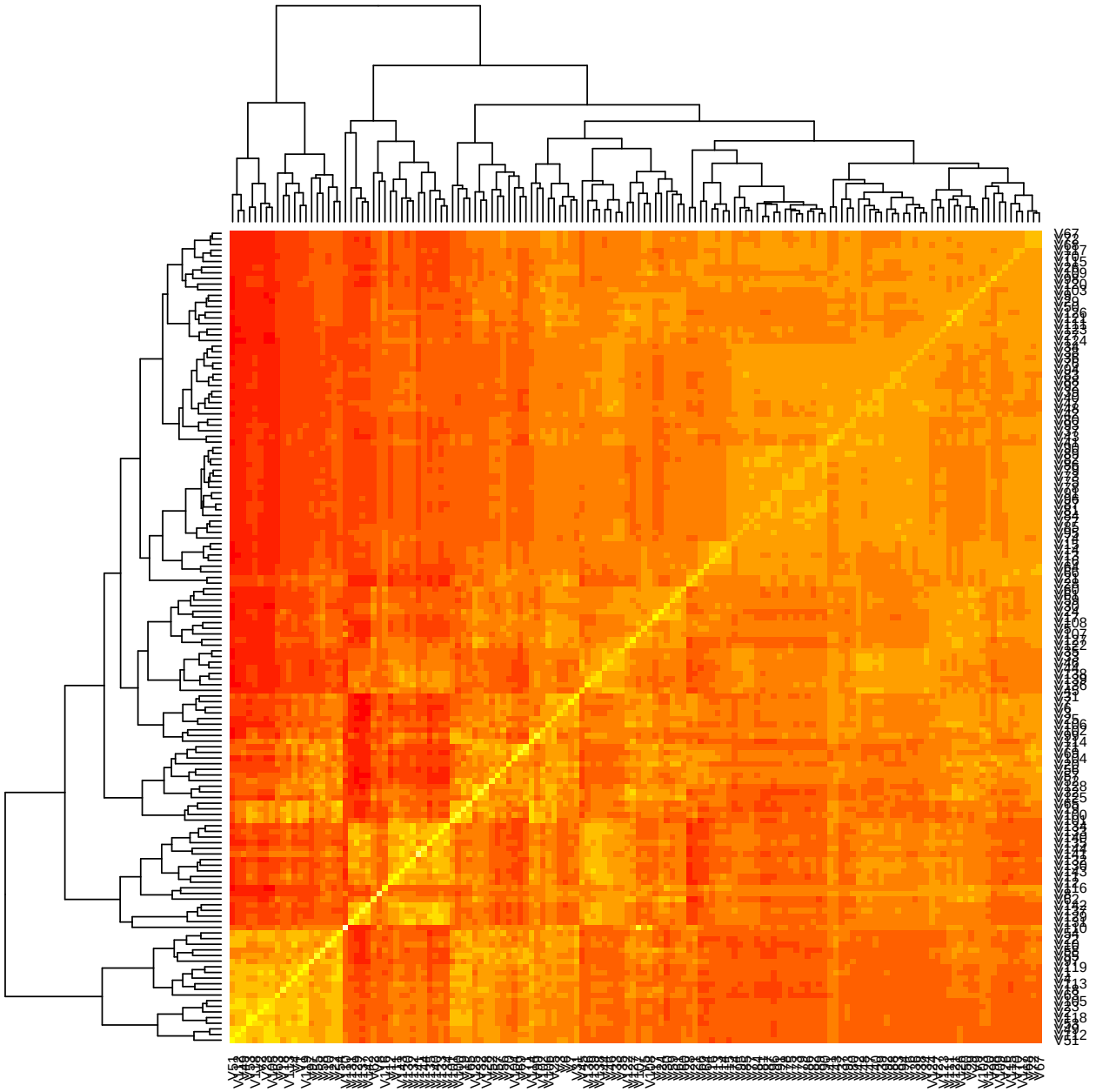
```
rescov <- cov(microarray)
image(rescov)
```




```
rescor <- cor(microarray)
image(rescor)
```



```
heatmap(rescor)
```



Exercice 7

1.

a) Résultat du programme

```
strsplit("AATTCCTCCCGTGACGAAATATA","")
sequence <- strsplit("AATTCCTCCCGTGACGAAATATA","")[[1]]
res <- list("A"=which(sequence=="A"),
"T"=which(sequence=="T"),
```

```
"G"=which(sequence=="G"),
"C"=which(sequence=="C"))
sapply(res,length)
```

Ecriture de la fonction :

```
indicesOccurences <- function(v){
res <- list("A"=which(v=="A"),
"T"=which(v=="T"),
"G"=which(v=="G"),
"C"=which(v=="C"))
return(res)
}
resio <- indicesOccurences(sequence)
```

b) Nombre d'occurences de chaque lettre

```
sapply(resio,length)
```

2.

a) Liste des trois chaines et indices des occurences

```
chaine1 <- strsplit("ATTCG","")[[1]]
chaine2 <- strsplit("CCGT","")[[1]]
chaine3 <- strsplit("GCGAGG","")[[1]]
chaines <- list(chaine1,chaine2,chaine3)
names(chaines) <- c("chaine1","chaine2","chaine3")
reslapply <- lapply(chaines,indicesOccurences)
```

b) Longueur de chaque séquence

```
sapply(chaines, length)
```

c) Nombre d'occurences de chaque nucléotide dans chacune des listes

```
fcttemp <- function(vlist){
  sapply(vlist,length)
}
lapply(reslapply, fcttemp)
# sapply(reslapply, fcttemp)
# Ecriture plus compacte : lapply(reslapply, function(x) sapply(x,length))
```

Exercice 8

1. Chargement des données, noms de variables et résumés numériques

```
library(ggplot2)
data("diamonds")
is.data.frame(diamonds)
names(diamonds)
unlist(sapply(diamonds, class))
summary(diamonds)
```

2. Extraction des entrées du tableau telles que

- les diamands soient de qualité `Premium`

```
head(subset(diamonds, cut == "Premium"))
attach(diamonds)
diamonds[cut=="Premium",]
```

- le carat soit supérieur à 3

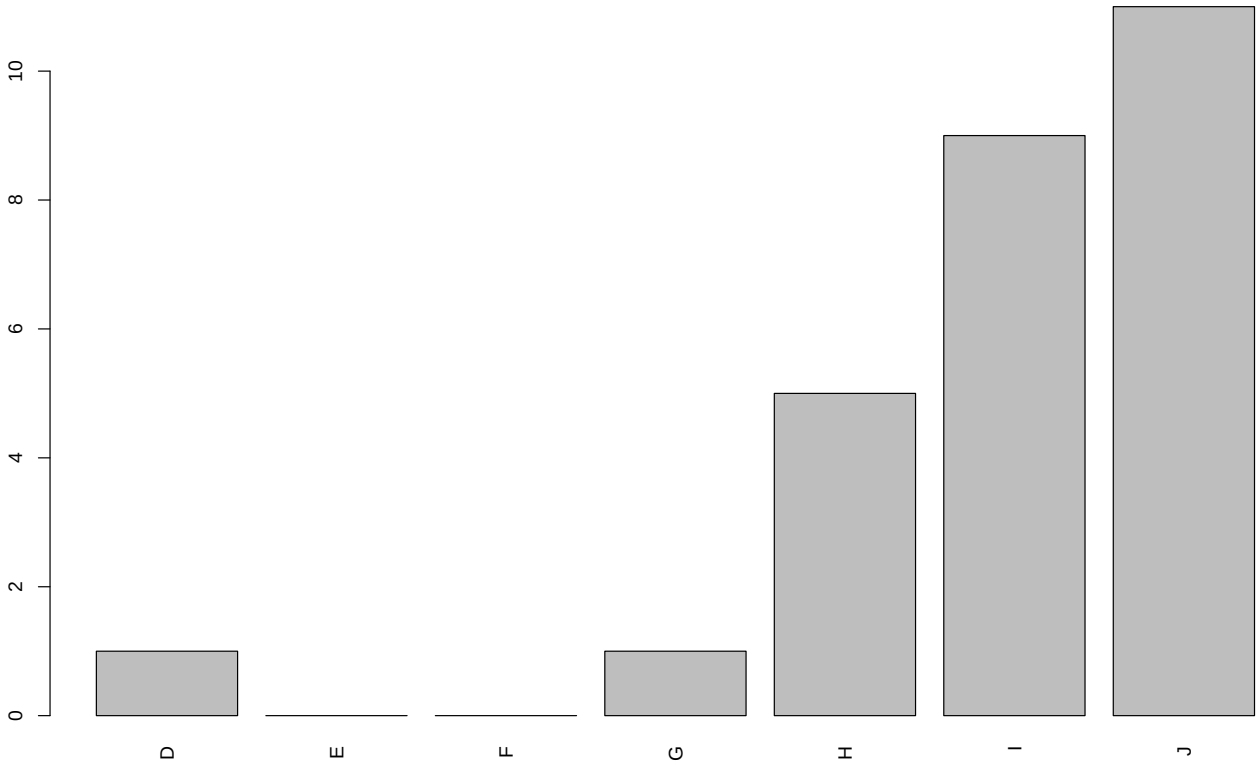
```
head(subset(diamonds, carat > 3))
```

- le volume (approximatif) soit supérieur à $500mm^3$

```
diamonds$volume <- diamonds$x*diamonds$y*diamonds$z
head(subset(diamonds, volume > 500))
head(subset(diamonds, x*y*z > 500))
```

- la qualité soit idéale, le prix inférieur à 1000 et le carat supérieur à .5. Déterminer la répartition des couleurs pour ce sous-ensemble

```
set <- subset(diamonds, cut == "Ideal" & price < 1000 & carat > 0.5 )
barplot(table(set$color), las=3)
```



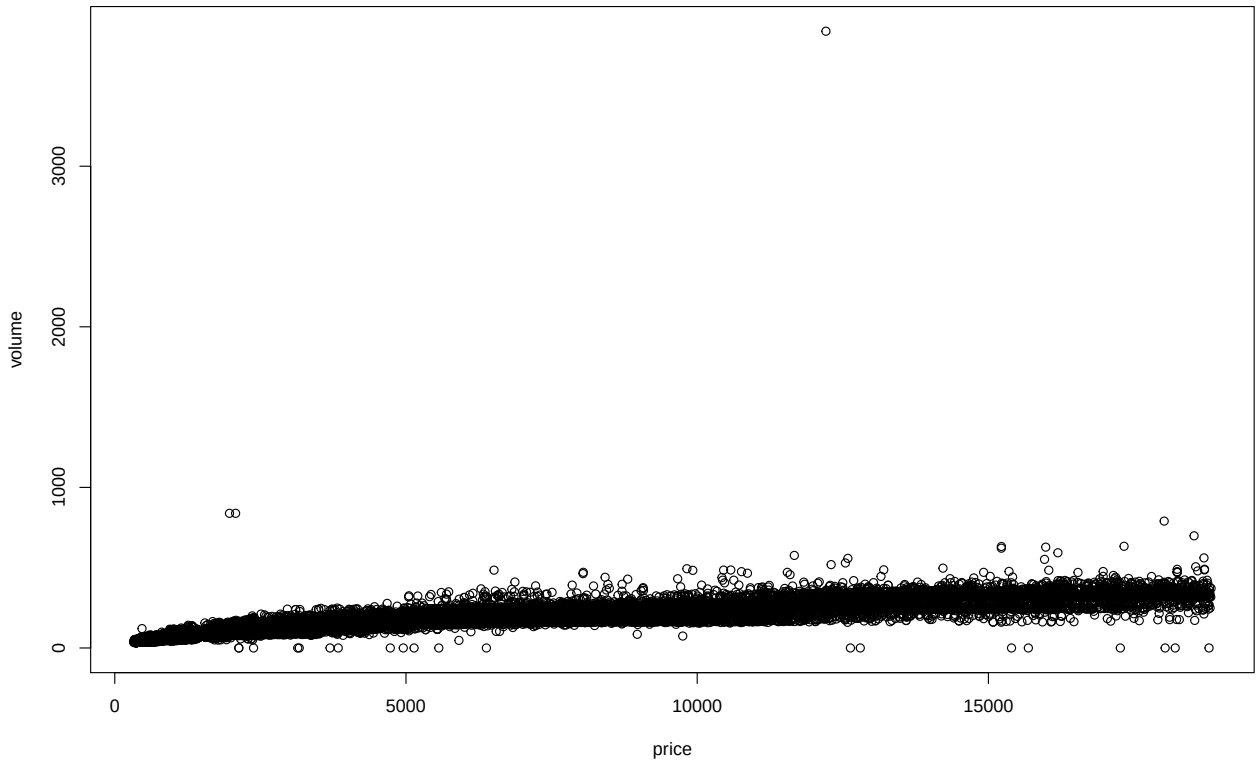
3. Prix moyen par classe de qualité et par intervalle de carat.

```
with(diamonds, tapply(price, cut, mean))
```

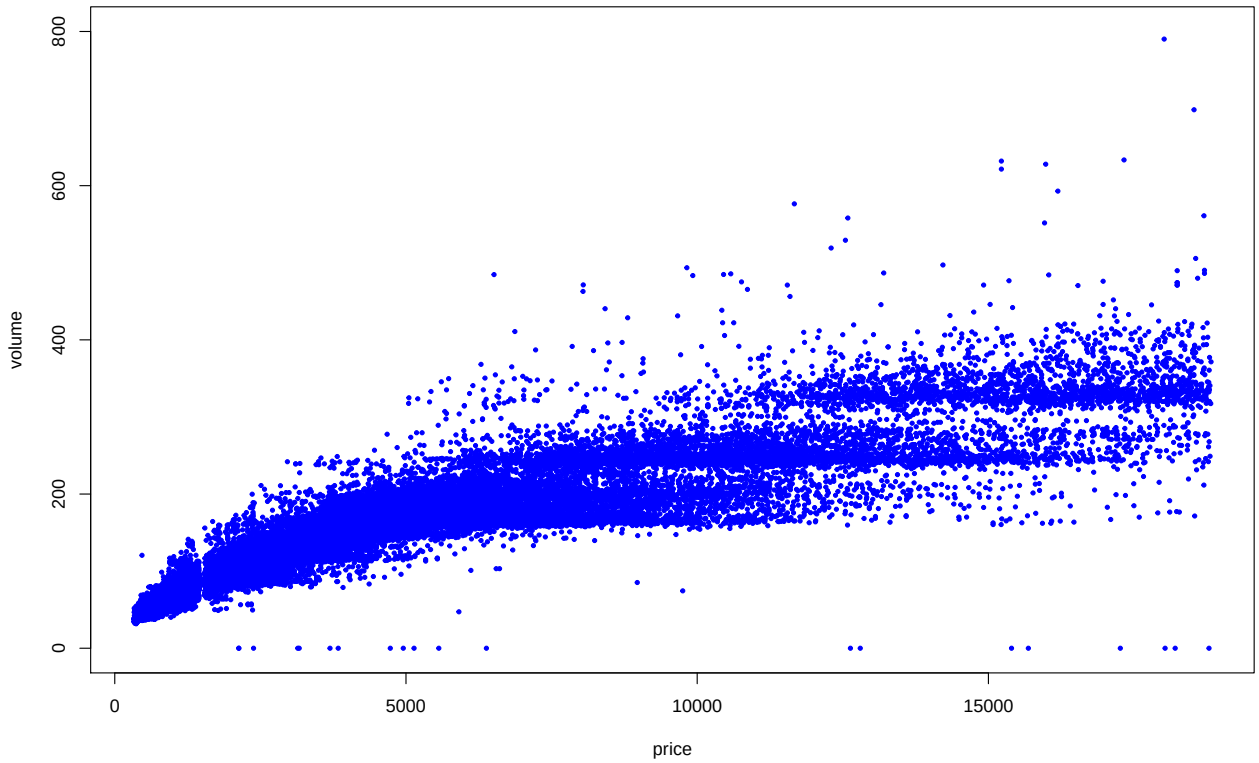
```
diamonds$carat.inter <- cut(diamonds$carat, 6)
with(diamonds,tapply(price,carat.inter,mean))
```

4. Graphe du volume en fonction du prix, le carat en fonction du prix. Représenter les boxplot de carat, prix et profondeur par classe de qualité et par couleur.

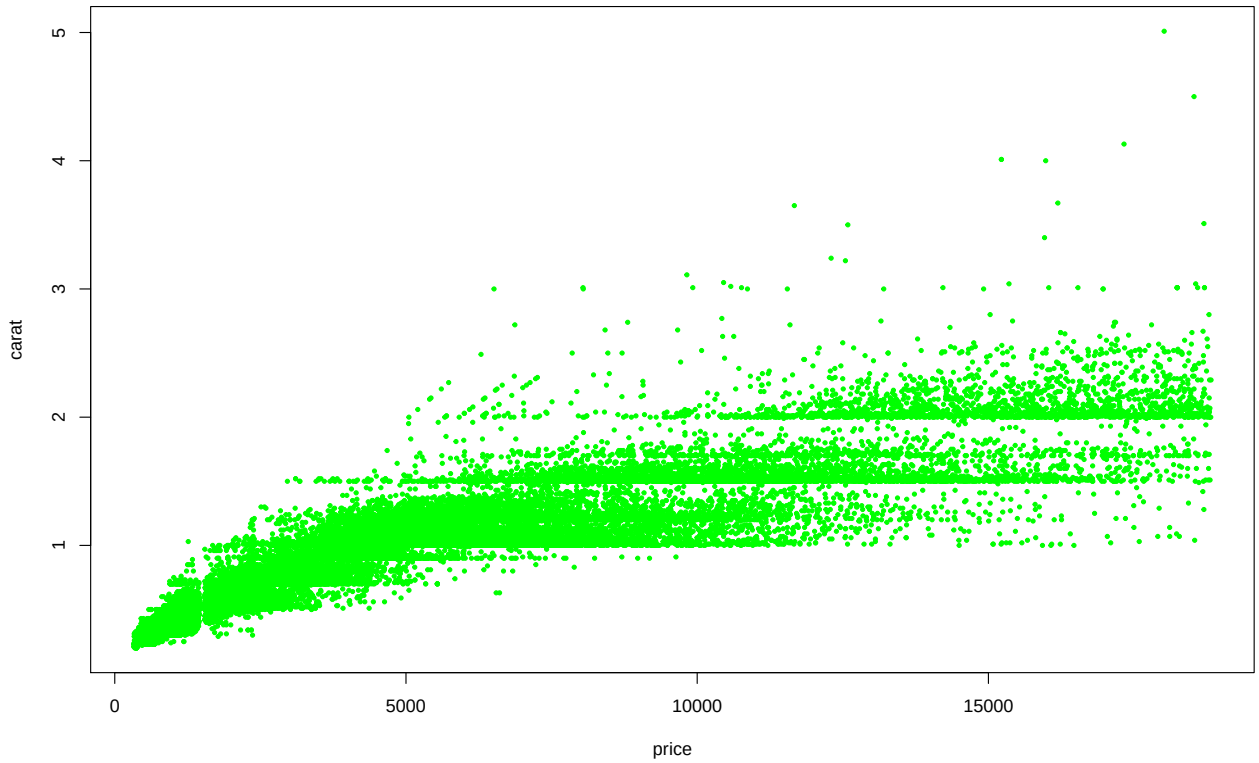
```
plot(volume~price,data=diamonds)
```



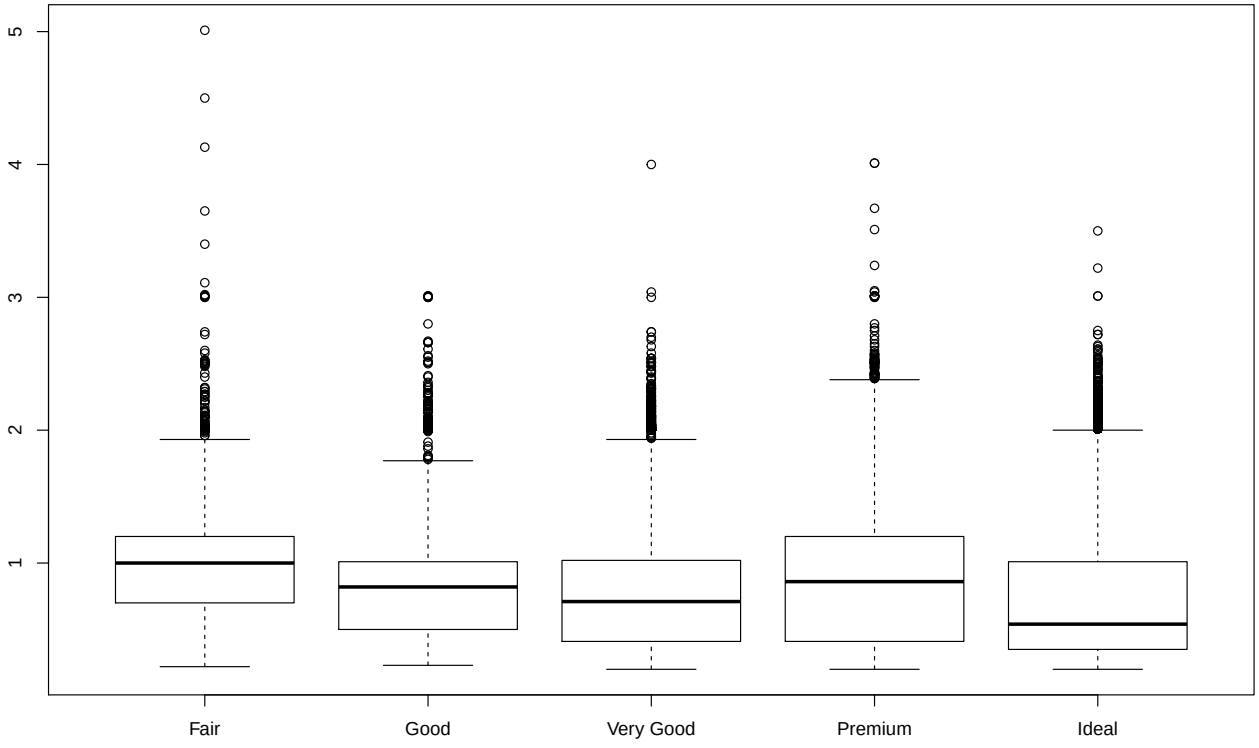

```
plot(volume~price, diamonds, ylim=c(0,800),pch=19,col="blue",cex=.5)
```



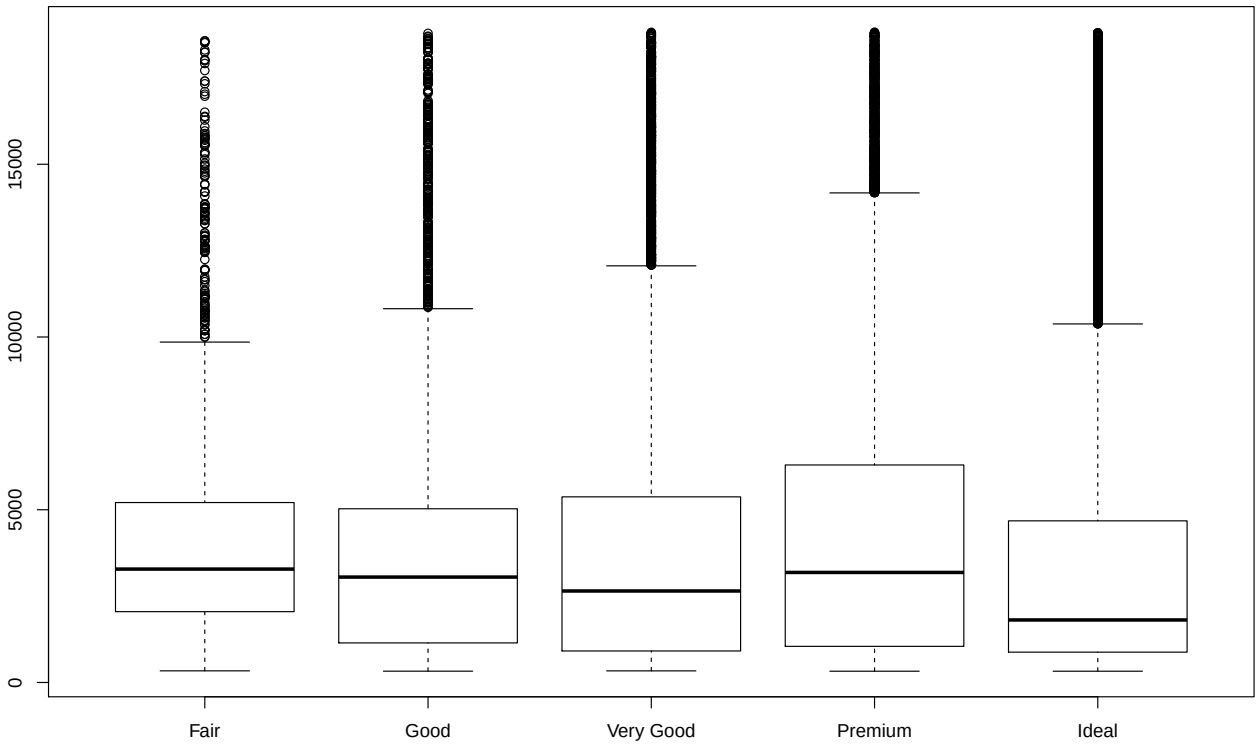

```
plot(carat~price, diamonds,pch=19,col="green",cex=.5)
```



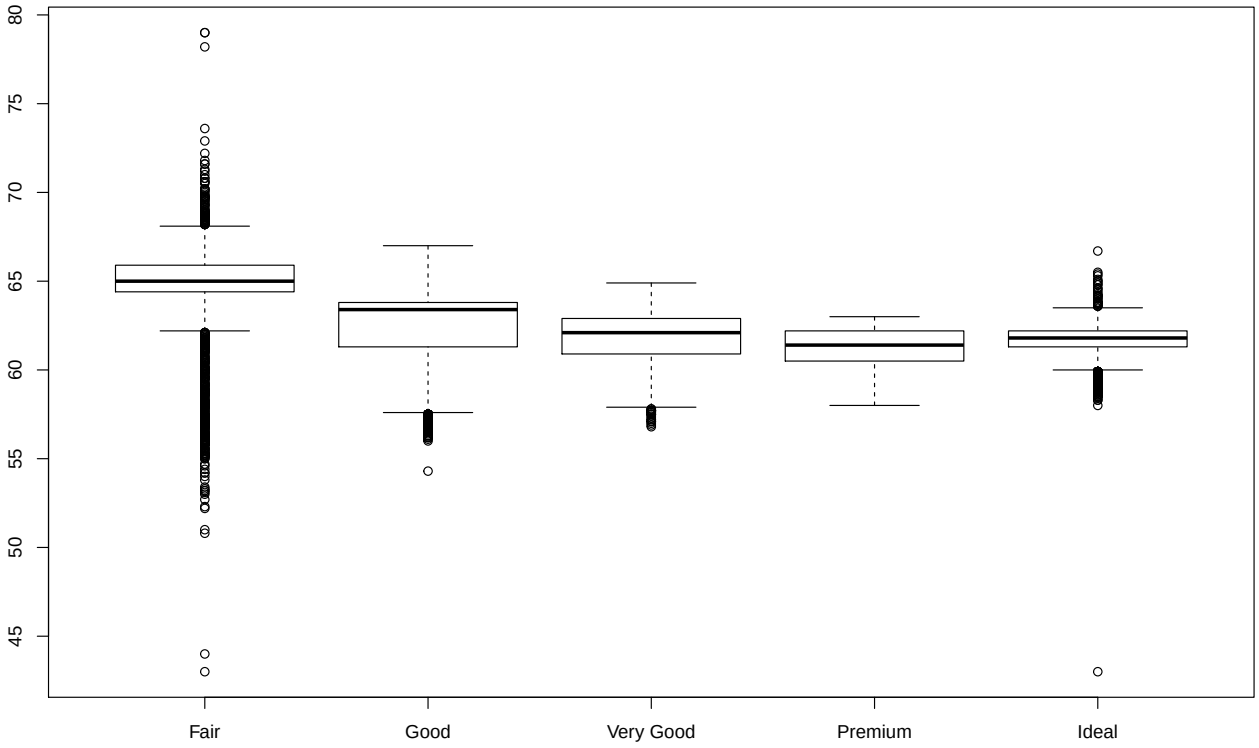

```
boxplot(carat~cut, diamonds)
```




```
boxplot(price~cut, diamonds)
```




```
boxplot(depth~cut, diamonds)
```



5. Prix moyen par triplet

```r
with(diamonds,tapply(price,paste(as.factor(cut),as.factor(color),as.factor(clarity)),mean))
# head(aggregate(price~cut+color+clarity,diamonds, mean))
```

Statistiques descriptives

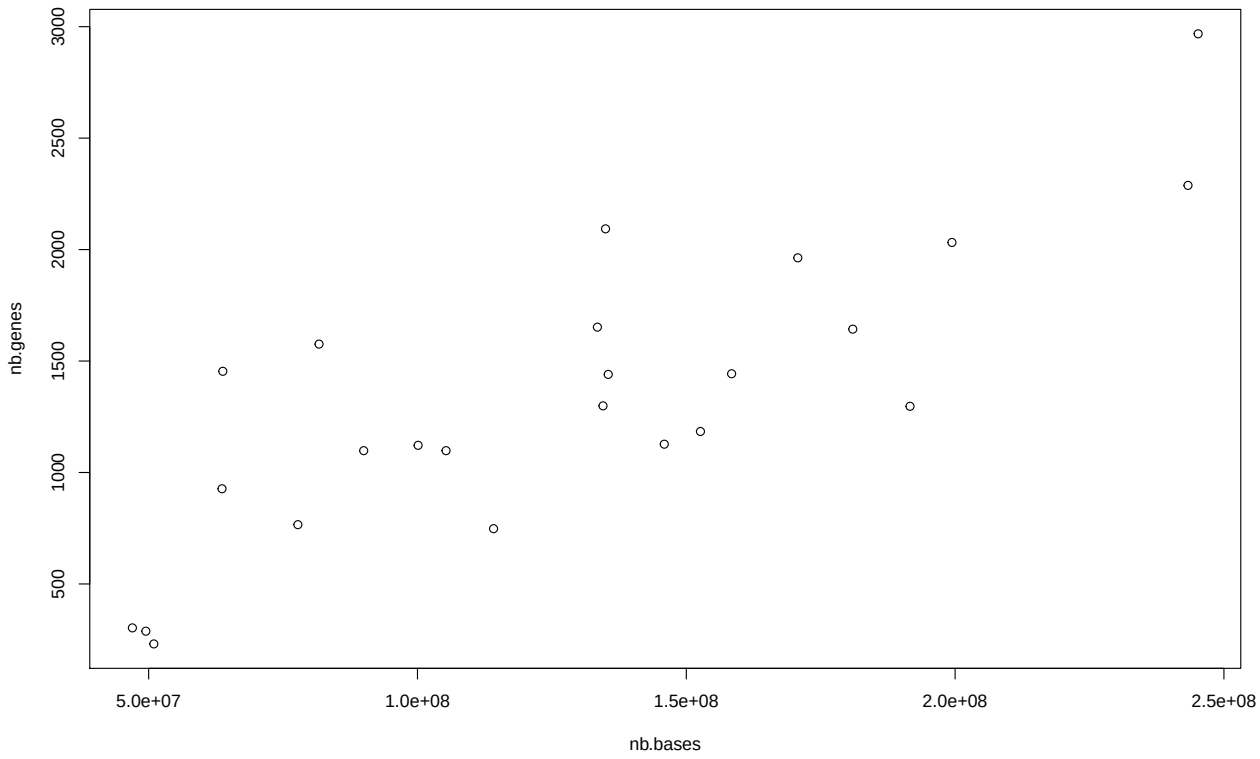
Exercice 9

1. Chargement des données

```r
setwd("/Users/cdalmasso/IntroductionR/projet1")
chromosomes <- read.table("./data/chromosomes.txt")
names(chromosomes ) <- c("chr","nb.genes","nb.bases")
getwd()
chromosomes <- read.table("./data/chromosomes.txt", col.names=c("chr","nb.genes","nb.bases"))
```

2. Nombre de gène en fonction du nombre de bases.

```r
plot(nb.genes~nb.bases, chromosomes)
```




```r
# ou bien : plot(chromosomes$nb.bases,chromosomes$nb.genes)
# ou encore : attach(chromosomes); plot(nb.bases,nb.genes)
```

3. Ajout d'une colonne supplémentaire au tableau indiquant pour chaque chromosome s'il est autosome ou pas.

```
chromosomes$autosome <- as.factor(c(rep(TRUE,22),rep(FALSE,2)))
```

4. Nombre total de paires de base d'un génome humain pour un homme, puis pour une femme.

```
commun <- colSums(subset(chromosomes, autosome==TRUE,  "nb.bases"))*2
homme <- commun + colSums(subset(chromosomes, autosome==FALSE, "nb.bases"))
femme <- commun + 2 * subset(chromosomes, chr == "X",  "nb.bases")
```

5. Export du tableau complété

```
setwd("/Users/cdalmasso/IntroductionR/projet1")
write.table(chromosomes, file="./output/chrpomosomes2.txt",row.names=FALSE,quote=FALSE)
```

Exercice 10

1. Chargement du jeu de données `hdpg`

```
# install.packages("ade4")
library(ade4)
data(hdpg)
```

2. Sélection du tableau `hdpg$ind` qui décrit l'échantillon des 1066 individus

```
ind <- hdpg$ind
head(ind)
```

3. Nombre de populations différentes

```
nlevels(ind$population)
```

4. Tableaux des effectifs des variable population, région et sexe.

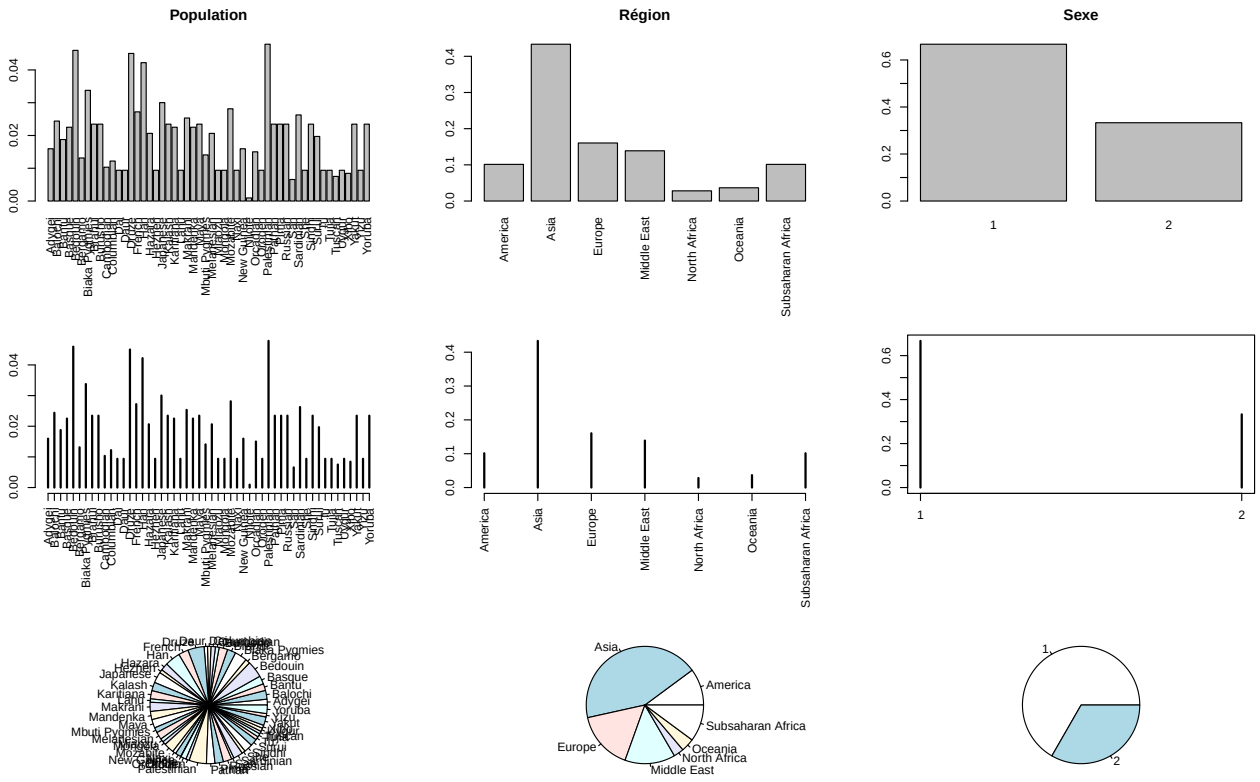
```
eff.pop <- table(ind$population)
eff.reg <- table(ind$region)
eff.sex <- table(ind$sex)
```

5. Transformation des tableaux en tableaux de fréquences.

```
freq.pop <- eff.pop/sum(eff.pop)
freq.reg <- eff.reg/sum(eff.reg)
freq.sex <- eff.sex/sum(eff.sex)
```

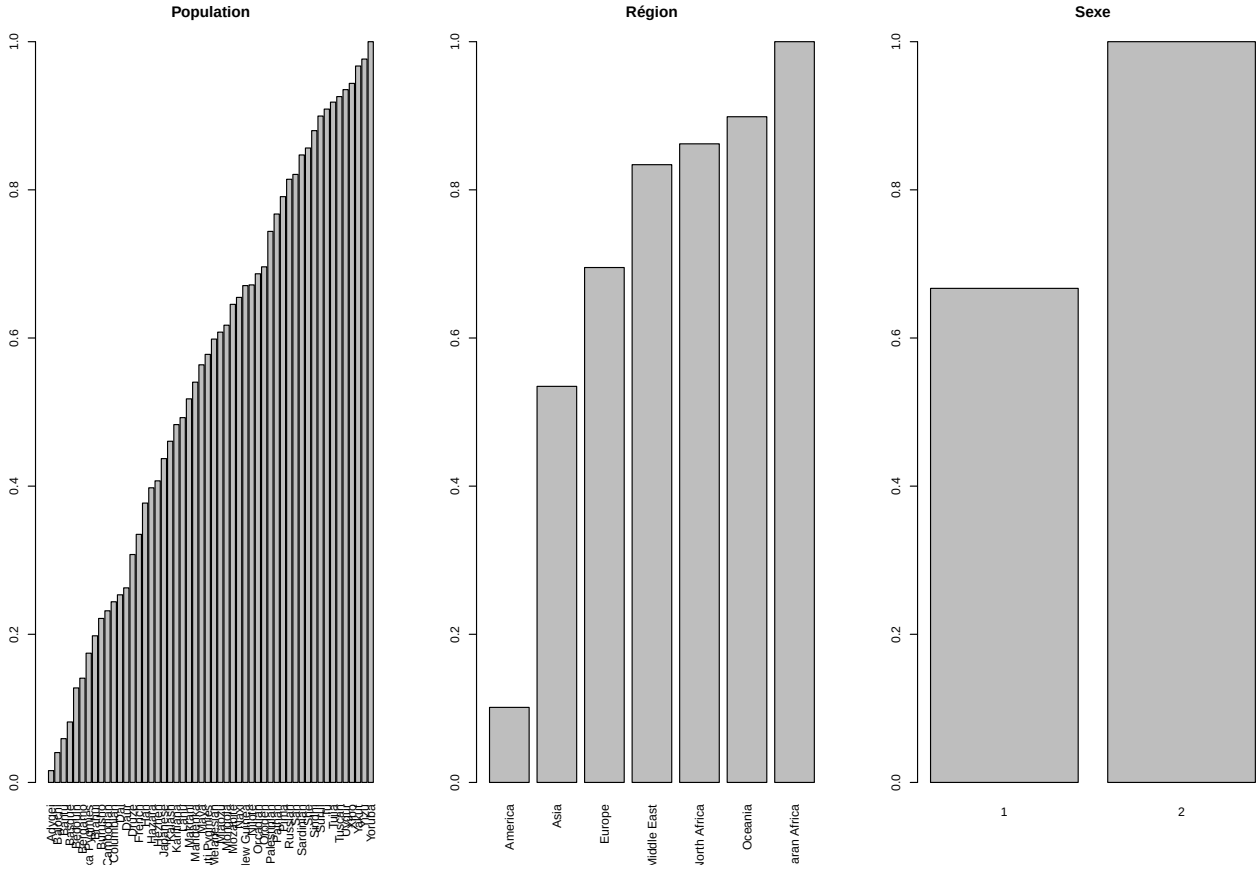
6. Représentation des tableaux de fréquence par des diagrammes en bâton et par des diagrammes circulaires

```
par(mfrow=c(3,3))
barplot(freq.pop, las=3, main="Population")
barplot(freq.reg, las=3, main="Région")
barplot(freq.sex, main="Sexe")
plot(freq.pop, type="h", las=3, xlab="", ylab="")
plot(freq.reg, type="h", las=3, xlab="", ylab="")
plot(freq.sex, type="h", xlab="", ylab="")
pie(freq.pop); pie(freq.reg) ; pie(freq.sex)
```



7. Représentation des fréquences cumulées

```
par(mfrow=c(1,3))
barplot(cumsum(freq.pop), las=3, main="Population")
barplot(cumsum(freq.reg), las=3, main="Région")
barplot(cumsum(freq.sex), main="Sexe")
```



Exercice 11

1. Lecture du fichier

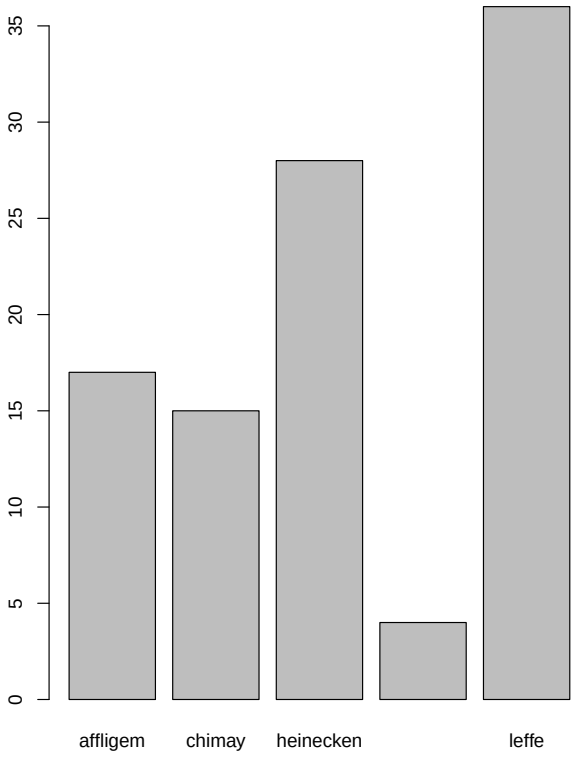
```
setwd("/Users/cdalmasso/IntroductionR/projet1")
bieres <- read.table(file="./data/bieres.csv",header=TRUE,sep=",",row.names=1)
bieres <- read.csv(file="./data/bieres.csv")
```

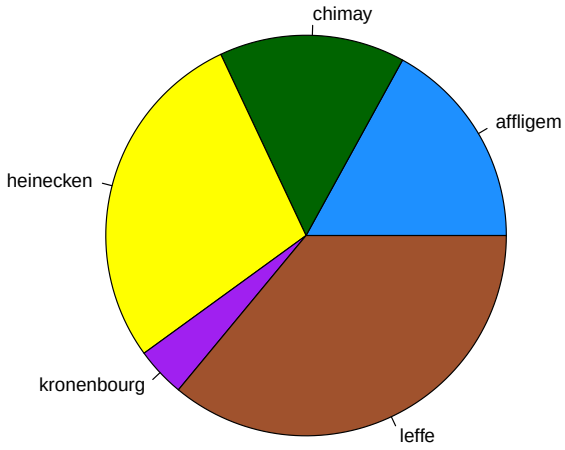
2. Nombre de marques (et noms des marques)

```
nlevels(bieres$x)
levels(bieres$x)
```

3. Nombre d'occurences par groupe

```
repartition <- table(bieres$x)
par(mfrow=c(1,2))
barplot(repartition)
pie(repartition,col=c("dodgerblue","darkgreen","yellow","purple","sienna"))
```





Exercice 12

1. Saisie ces données dans un vecteur

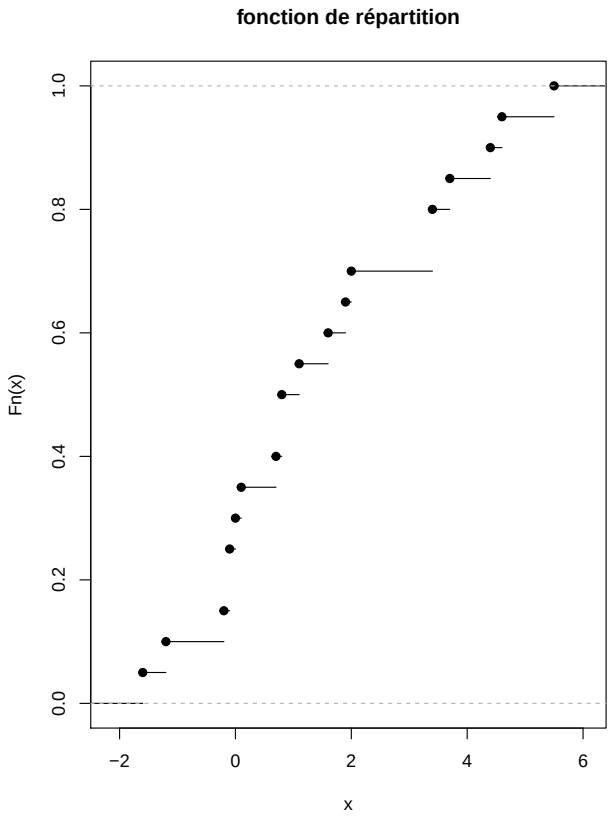
```
somnifere <- scan(text="0.7 -1.6 -0.2 -1.2 -0.1 3.4 3.7 0.8 0.0 2.0 1.9 0.8 1.1 0.1 -0.1 4.4 5.5 1.6 4.6
```

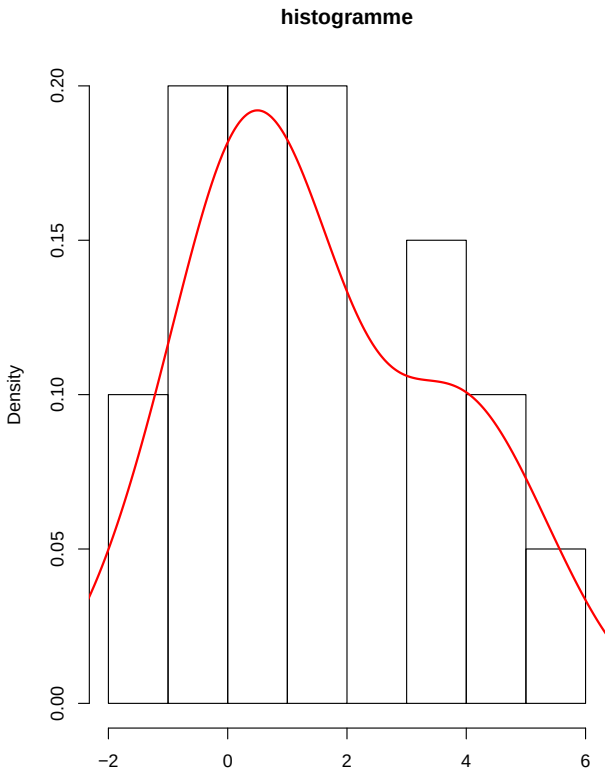
2. Résumés numériques

```
summary(somnifere)
fivenum(somnifere)
```

3. Fonction de répartition empirique puis histogramme normalisé des données

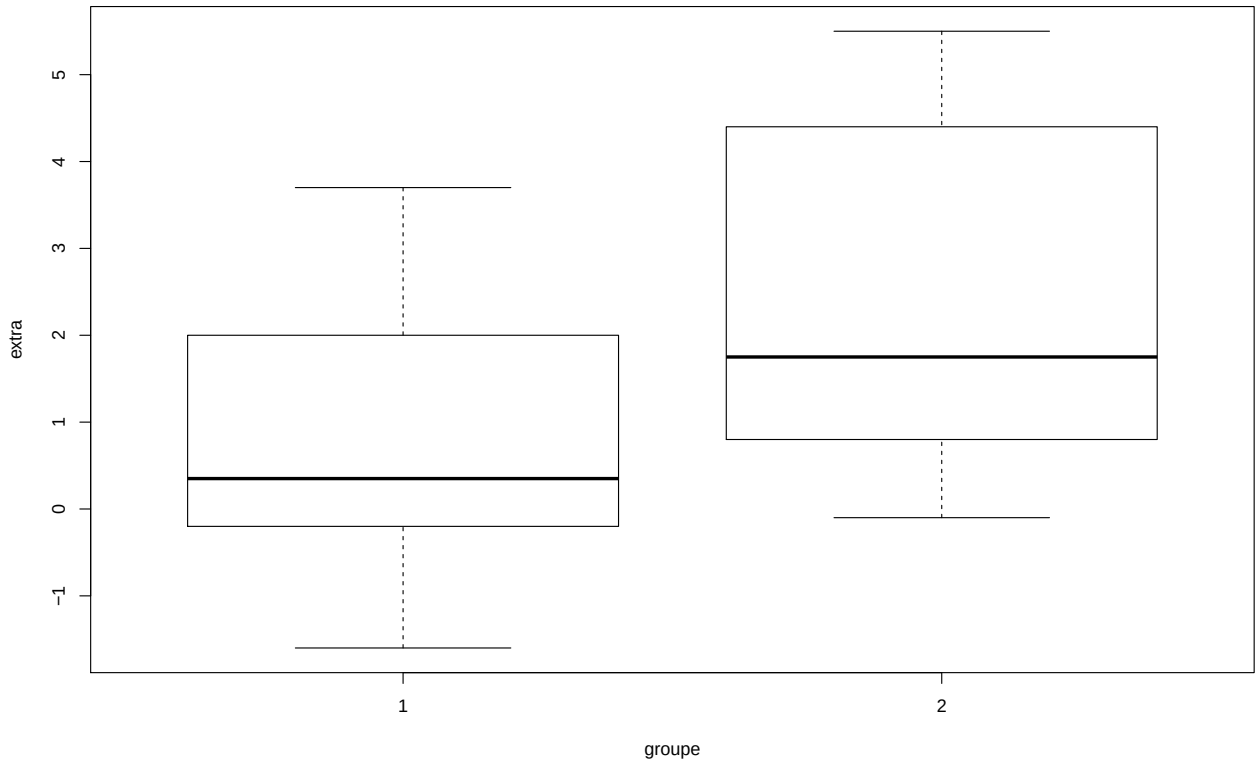
```
par(mfrow=c(1,2))
plot(ecdf(somnifere), main="fonction de répartition")
hist(somnifere, freq=FALSE, main="histogramme", xlab="")
lines(density(somnifere), col="red", lwd=2)
```





4. Résumé statistique pour chaque groupe

```
somnifere <- data.frame(extra=somnifere, groupe=factor(rep(c(1,2), each=10)))
plot(extra ~ groupe, somnifere)
```



Exercice 13

1. Chargement des données

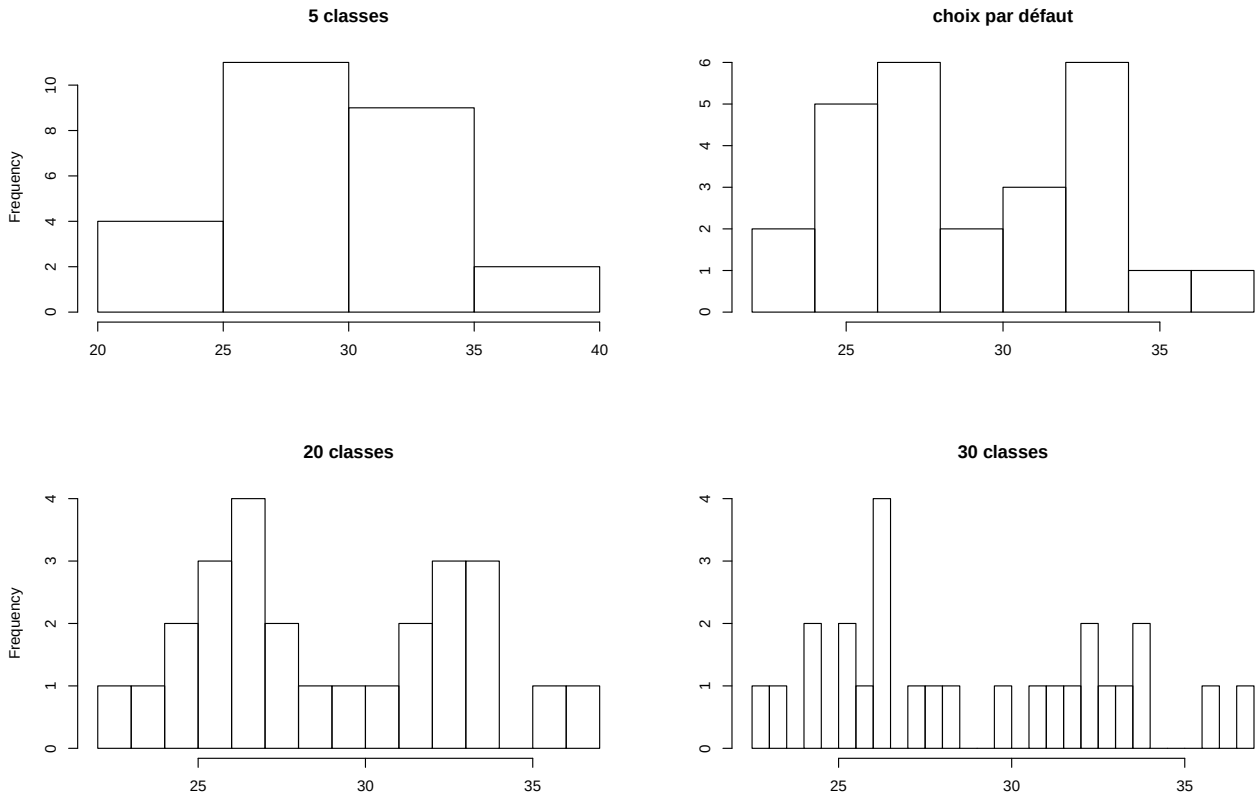
```
setwd("/Users/cdalmasso/IntroductionR/projet1")
load("./data/gini.RData")
head(gini)
```

2. Sélection des lignes du tableau correspondant à l'année 2007

```
gini2007 <- subset(gini, year == 2007, -year)
```

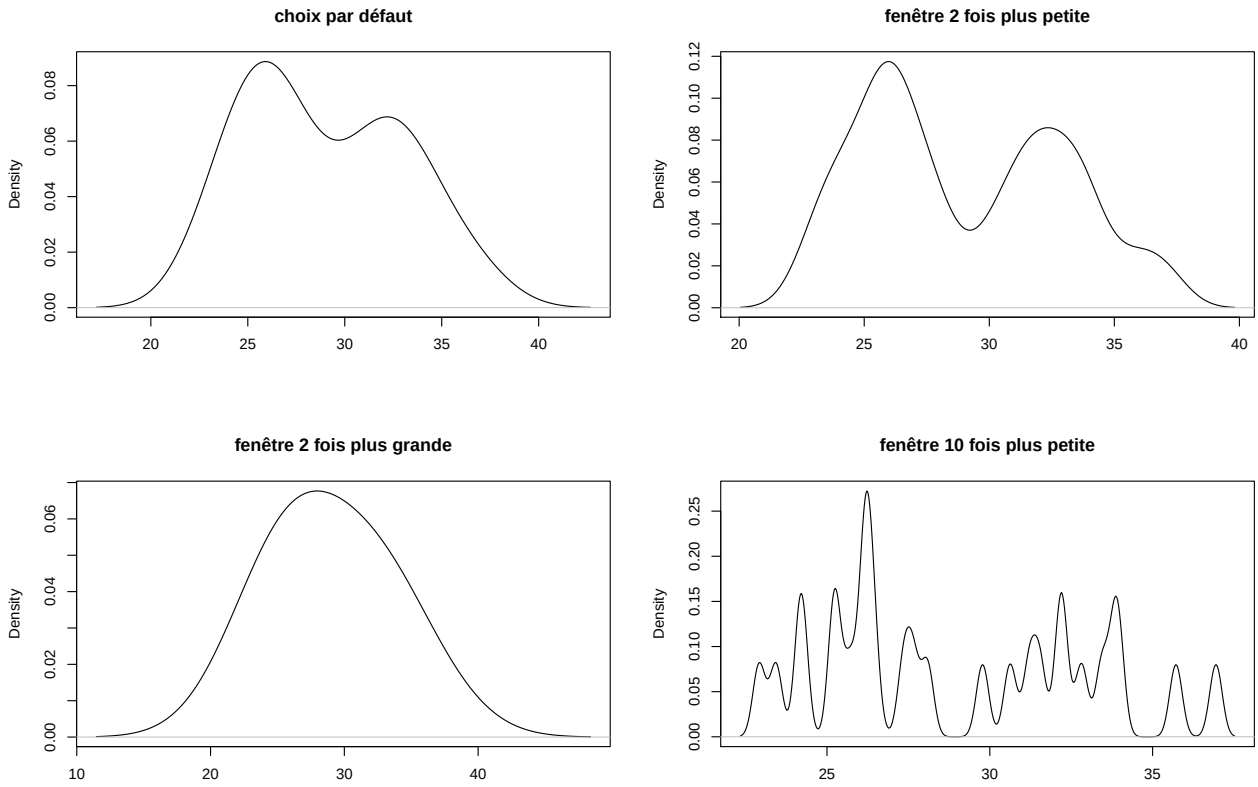
3. Histogramme des coefficients

```
par(mfrow=c(2,2))
hist(gini2007$gini, nclass = 5, main="5 classes", xlab="")
hist(gini2007$gini, main="choix par défaut", xlab="", ylab="")
hist(gini2007$gini, nclass = 20, main="20 classes", xlab="")
hist(gini2007$gini, nclass = 30, main="30 classes", xlab="", ylab="")
```



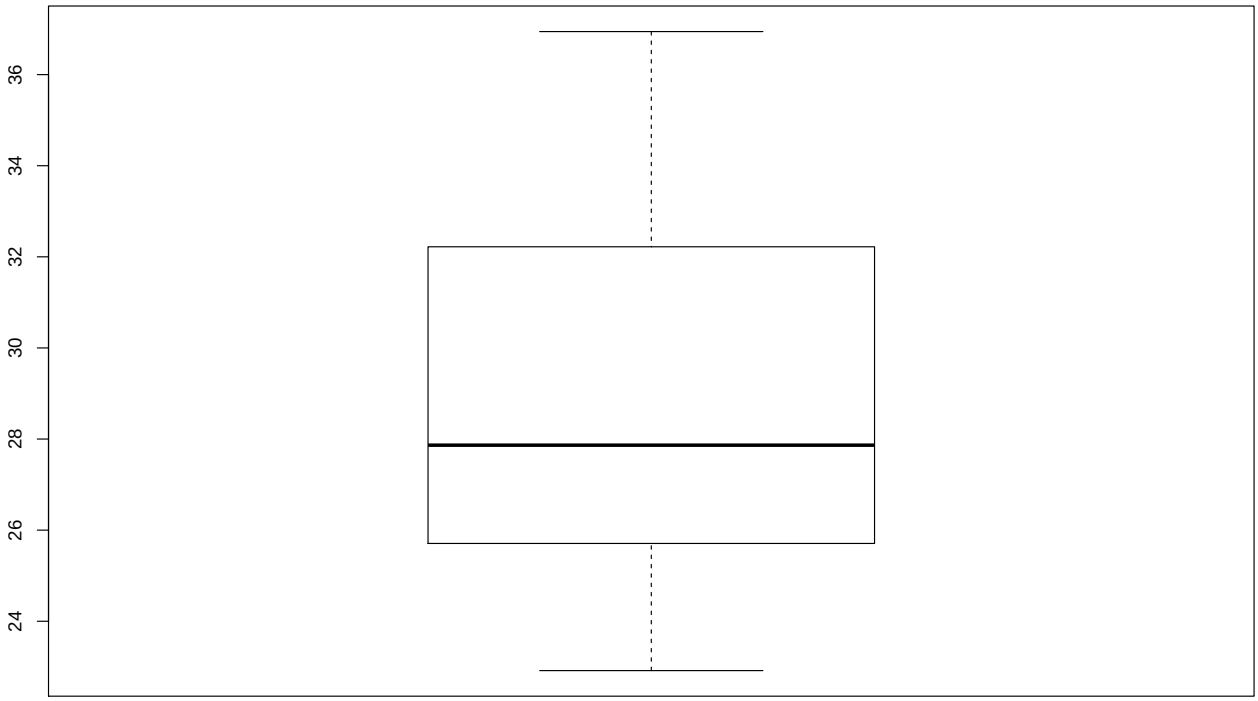
4. Histogramme lissé des coefficients

```
par(mfrow=c(2,2))
plot(density(gini2007$gini), main="choix par défaut", xlab="")
plot(density(gini2007$gini, adjust=.5),main="fenêtre 2 fois plus petite", xlab="")
plot(density(gini2007$gini, adjust=2) , main="fenêtre 2 fois plus grande", xlab="")
plot(density(gini2007$gini , adjust=.1), main="fenêtre 10 fois plus petite", xlab="")
```



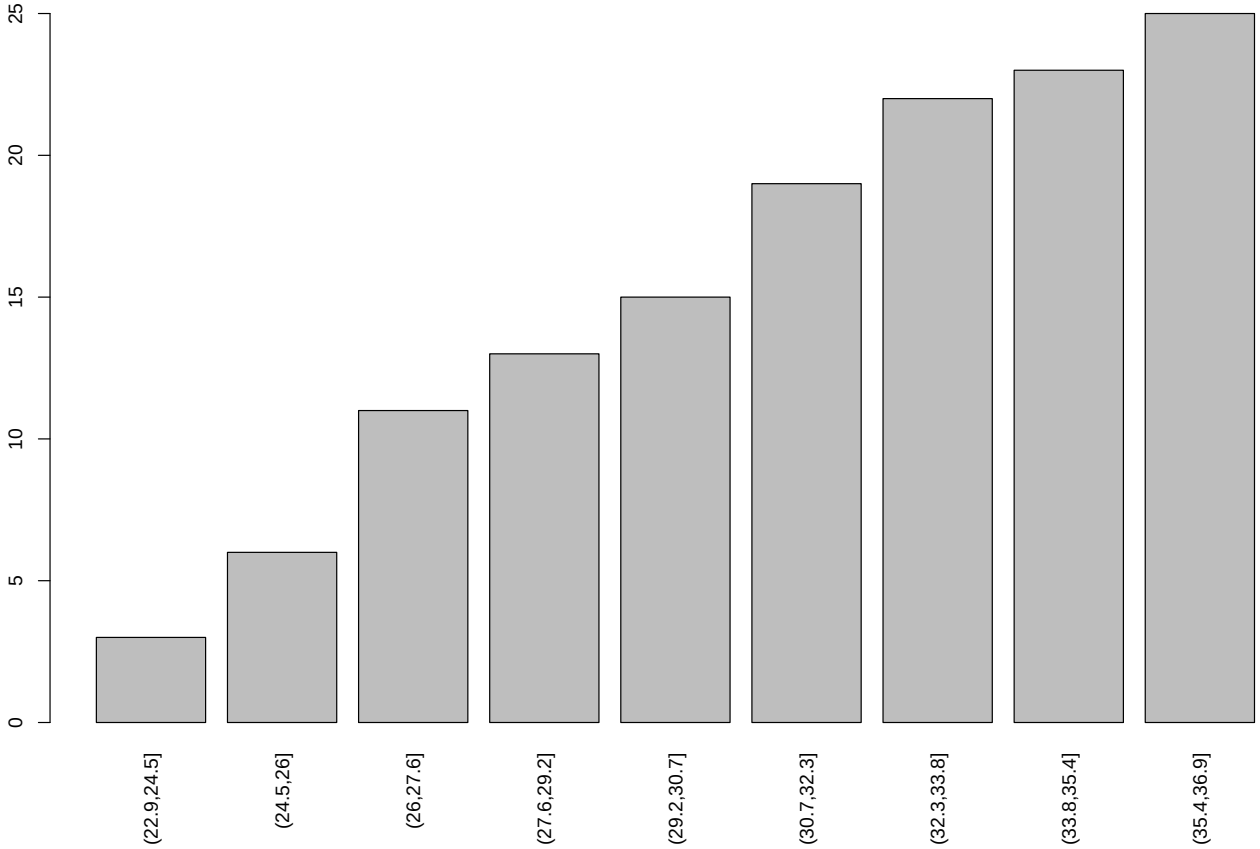
5. Boxplot des coefficients

```
boxplot(gini2007$gini)
```



6. Diagramme des fréquences cumulées

```r
bornes <- seq(min(gini2007$gini), max(gini2007$gini), len=10)
freq.gini <- table(cut(gini2007$gini, bornes))
barplot(cumsum(freq.gini), las=3)
```



7. Écriture d'une fonction R qui donnant les pays de coefficient Gini d'index maximum et minimum.

```r
extremeGini <- function(ginicoef, country) {

  return(c( min.gini = country[which.min(ginicoef)],
            max.gini = country[which.max(ginicoef)]))

}
extremeGini(gini2007$gini, gini2007$country)
```

8. Classer les pays par leur coefficient de Gini.

```r
with(gini2007, country[order(gini)])
```

9. Calcul de la moyenne, la variance, du coefficient d'asymétrie, du coefficient d'aplatissement

```r
library(moments)
mean(gini2007$gini); var(gini2007$gini); skewness(gini2007$gini); kurtosis(gini2007$gini)
```

10. Nombre de pays plus égalitaires que la France en Europe

```r
gini.France <- with(gini2007, gini[which(country == "France")])
sum(gini2007$gini > gini.France)
```

Programmation

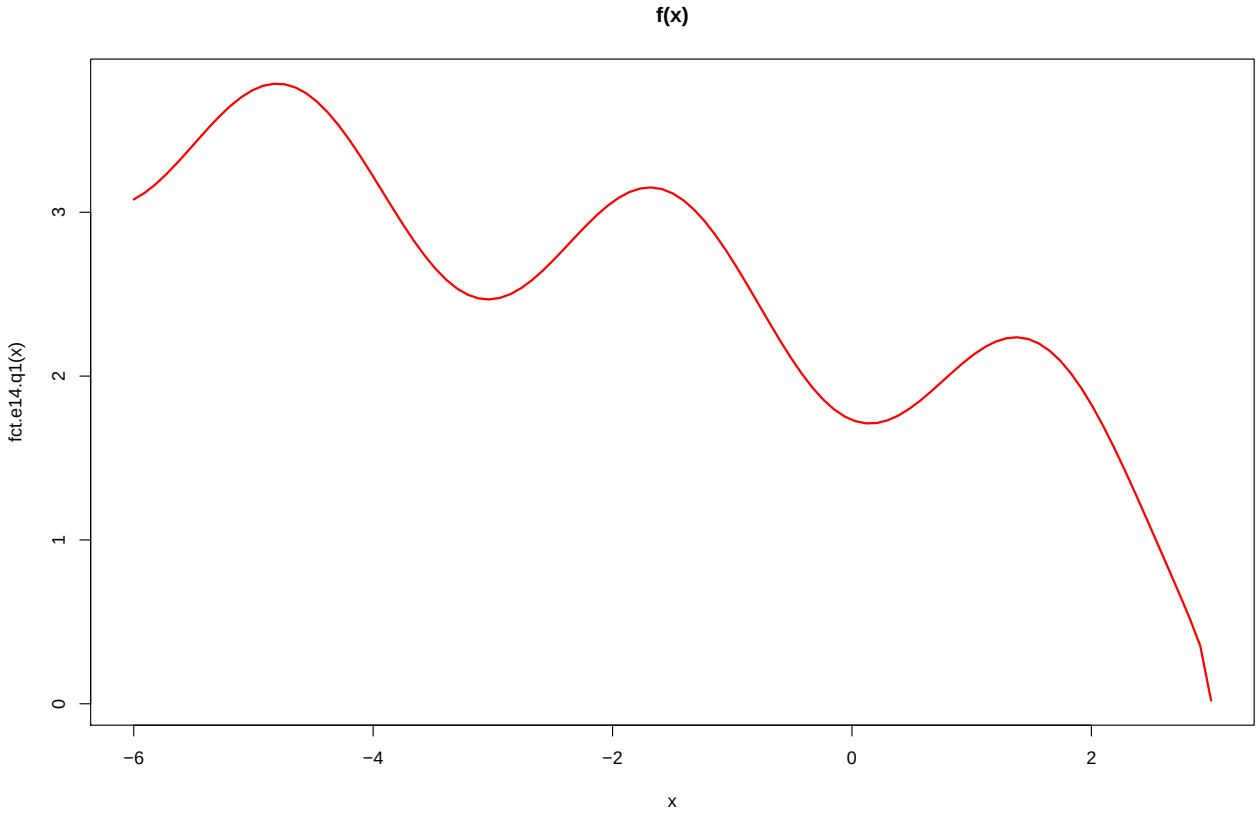
14.

1. Construction de la fonction

```r
fct.e14.q1 <- function(x){
  res <- sin(x)^2+sqrt(abs(x-3))
  return(res)
}
fct.e14.q1(x=1)
```

2. Graphe

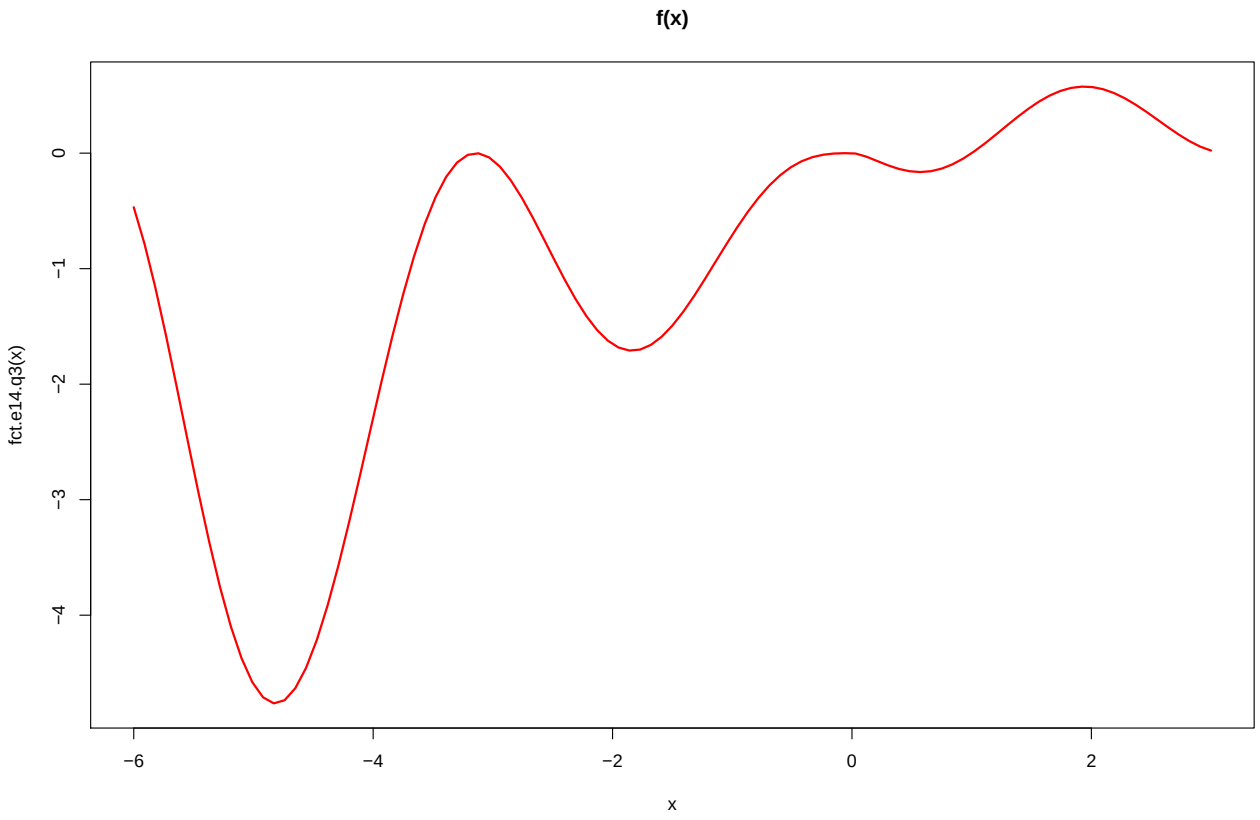
```r
# curve(fct.e14.q1,from=-6,to=3)
curve(fct.e14.q1,from=-6,to=3,col="red",lwd=2,main="f(x)")
```



3. Idem pour g

```r
fct.e14.q3 <- function(x){
  res <- ifelse(x>0,sin(x)^2*log(x),sin(x)^2*x)
  return(res)
```

```r
}

fct.e14.q3.v2 <- function(x){
  if(x>0){
    res <- sin(x)^2*log(x)
  }
  else{
    res <- sin(x)^2*x
  }
  return(res)
}

curve(fct.e14.q3,from=-6,to=3,col="red",lwd=2,main="f(x)")
```




```r
# ou bien :
# x <- seq(-6,3,0.001)
# y <- fct.e14.q3(x)
# y.v2 <- apply(as.matrix(x),1,fct.e14.q3.v2)
# plot(x,y.v2,type="l",col="red",lwd=2)
```

Exercice 15

1. Calcul de l'ICM

```r
calculICM <- function(poids,taille){
  ICM <- poids/taille^2
  return(ICM)
}

# Autre possibilité :
calculICM.v2 <- function(poids,taille){
  if(max(taille)> 1.2 & min(taille) < 3 & min(poids) > 15 & max(poids) < 200){
    ICM <- poids/taille^2
  }
  else{
    print("ERREUR : variables non conformes !")
    ICM <- NULL
  }
  return(ICM)
}
```

2. Application

```r
calculICM(64,1.64)
mesdonnees <- data.frame(c(64,56,102,51),c(1.64,1.61,1.72,1.65))
names(mesdonnees) <- c("poids","taille")
attach(mesdonnees)
calculICM(taille,poids)
calculICM.v2(taille,poids)
calculICM.v2(poids,taille)
```

3. Classification

```r
fct.classification <- function(poids,taille){
  ICM <- calculICM.v2(poids,taille)
  classif <- cut(ICM,c(0,18.5,25,30,100),
                 labels=c("insuffisance pondérale","corpulence normale","surpoids","obesite"))
  return(classif)
}
fct.classification(64,1.64)
```

4. Application

```r
fct.classification(poids,taille)
```