

# Introduction à R - Corrigés des exercices

*Emmanuelle Comets*

## 1 Cours 1

### 1.1 Premiers pas

Calculs simples, utilisation de l'aide :

```
10*9*8*7*6*5*4*3*2*1      ?log
log(2)                      log10(2)
log(2)/log(10)              log(2,base=10)
```

### 1.2 Vecteurs

#### Création d'un vecteur (1)

```
vec<-c(10,3,4,5,6,10,100,100,10,20,
        30,40)                vec<-c(10,3:6,10,rep(100,2),
                                seq(10,40,by=10))
```

#### Création d'un vecteur (2)

```
x<-c(1,4,5)                  y[-2]
x                             xy<-y[c(1,4,5)]
y<-seq(1,9,2) # ou bien :    # ou puisque x=c(1,4,5)
y<-2*(1:5)-1                 xy<-y[x]
y                             xy
y[2]                          y[2:4]
```

#### Opération sur les vecteurs

```
y+1                          (y+1)[1:4]
y[1:4]+1                      x<-2:4
# ou une écriture équivalente y*x
```

#### Manipulation de vecteurs

```
vec<-rnorm(10)                yvec<-log(vec)
vec                            vec2<-yvec[!is.na(yvec)]
length(vec[vec>0])             length(vec2)
```

## 1.3 Tableaux

### Création d'une matrice

```
mat<-matrix(1:15,ncol=5,byrow=T)      mat[mat[,1]<3,1]
mat                                     mat[mat[,1]<3,]
mat[2:3,c(2,4)]
```

## 2 Cours 2

### Remise en jambe (1) :

```
rep(c(0,6),3)
c(1:4)*3-2
rep(1:3,4)
rep(1:3,1:3)
rep(1:3,3:1)
1+0:2*4.5
rep(1:3,each=4)

1+(1:10)/10
c(0:9)*2+1
rep(-2:2,2)
rep(-2:2,each=2)
(1:10)*10
```

### Remise en jambe (2) :

```
d<-rep(-2:2,each=2)
d[d<0]<-NA
length(d[is.na(d)])
# ou

sum(is.na(d))
d<-rep(-2:2,each=2)
d[d<0]<--10
```

### Remise en jambe (3) :

```
x<-matrix(1:120,ncol=12)
x[2*(1:5),]
x[apply(x,1,mean)<60,]
x[,apply(x,2,sum)<500]

x[apply(x,1,mean)<60,apply(x,2,sum)<500]
x[apply(x,1,mean)<60 & x[,1]!=3,
  apply(x,2,sum)<500]
```

Dans la dernière écriture, on utilise le fait que `x[apply(x,1,mean)<60,apply(x,2,sum)<500]` est elle-même un tableau, donc on peut référencer ses éléments...Une écriture moins concise et plus claire serait de faire 2 lignes en utilisant un tableau intermédiaire :

```
x1<-x[apply(x,1,mean)<60,apply(x,2,sum)<500]
x1[-3,]
```

## 2.1 Dataframe

### Extraction de données

```
airquality
air1<-airquality[airquality$Temp>92,]
air1<-transform(air1,logTemp=log(Temp))
air1<-transform(air1,ftemp=ifelse(Temp>94,1,0))
air2<-air1[!is.na(air1$Ozone) & air1$Temp<=94,]
air2

air3<-airquality[!is.na(airquality$Ozone),]
air3[1:10,] # pour voir les 10 premières lignes de air3
head(air3) # pour voir les 6 premières lignes de air3

air3<-transform(air3,monindic=ifelse(Month<4 & Temp>80,1,0))
```

### Exercice sur match

```
dates<-c("1971-01-20","1971-01-28","1971-02-03","1971-02-11","1971-02-18",
"1973-01-17","1973-01-25","1973-01-31","1973-02-17","1974-01-07","1974-01-10",
"1974-01-15","1974-01-22","1974-01-29","1974-02-05","1974-02-12","1974-02-19")
mesure<-c(64,69,71,71,71,32,42,28,32,18,25,29,34,36,42,50,61)

dates[match(unique(mesure),mesure)]
mat<-cbind(mesure=mesure,dates=dates)
mat<-mat[order(mat[,1]),]
```

## 2.2 Statistiques descriptives

### Quantiles :

```
x<-matrix(1:100,ncol=4)
apply(x,2,quantile,c(0.1,0.9))
apply(x,1,quantile,c(0.1,0.9))
```

### Moyenne, variance :

```
x<-matrix(1:100,ncol=4)
mean(x)
apply(x,2,mean)
apply(x,2,var)
apply(x[1:3,],1,mean)
apply(x[1:3,],1,var)
```

## Corrélation :

```
?ToothGrowth                                xmat<-matrix(ToothGrowth[,1],ncol=6)
head(ToothGrowth)                             cor(xmat)
str(ToothGrowth)
```

## 2.3 Lois de probabilité et simulation

### Simulation d'échantillons :

```
#Tirage de 6 valeurs dans N(2,5), 4 dans N(-1,4)
vec<-c(rnorm(6,2,5),rnorm(4,-1,4))
#Tirage dans la mixture en utilisant une variable dans N(0,1) (loi symétrique)
v1<-c()
for(i in 1:10) {
  il<-rnorm(1)
  v1<-c(v1,ifelse(il>0,rnorm(1,2,5),rnorm(1,-1,4)))
}
m1<-mean(v1)
#Note : on peut le faire sans la boucle
il<-rnorm(10)
v1<-ifelse(il>0,rnorm(10,2,5),rnorm(10,-1,4))

#Calcul de la moyenne m1 de v1, et répétition (une nouvelle boucle) 10 fois
m1<-c()
for(j in 1:10) {
  v1<-c()
  for(i in 1:10) {
    il<-rnorm(1)
    v1<-c(v1,ifelse(il>0,rnorm(1,2,5),rnorm(1,-1,4)))
  }
  m1<-c(m1,mean(v1))
}
#Tirage dans une mixture de probabilités 10-90%
m2<-c()
for(j in 1:10) {
  v2<-c()
  for(i in 1:10) {
    il<-rnorm(1)
```

```

v2<-c(v2,ifelse(i1>qnorm(0.9),rnorm(1,2,5),rnorm(1,-1,4)))
}
m2<-c(m2,mean(v2))
}
#Moyennes de m1 et m2
mean(m1)
mean(m2)

```

### Probabilité que la moyenne soit inférieure à 130, n=10 sujets :

Notons X la variable 'clairance à la créatinine'.

$$P(\bar{X} \leq 130) = P\left(\frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \leq \frac{130 - \mu}{\sigma/\sqrt{n}}\right) = P\left(Z \leq \frac{130 - \mu}{\sigma/\sqrt{n}}\right) \quad (1)$$

où Z est la variable centrée réduite associée à X, qui suit une loi  $\mathcal{N}(0, 1)$ .

Donc on cherche la probabilité :

$$P(\bar{X} \leq 130) = P\left(Z \leq \frac{130 - 120}{40/\sqrt{10}}\right) = P(Z \leq 0.79)$$

```

moy<-120                                #En se ramenant à N(0,1)
ect<-40                                  z<-(a1-moy)/(ect/sqrt(nsubj))
a1<-130                                  pnorm(z)
nsubj<-10                                #En utilisant pnorm
                                           pnorm(a1,moy,ect/sqrt(nsubj))

```

### Probabilité que la moyenne soit comprise entre 120 et 130 :

$$P(120 \leq \bar{X} \leq 130) = P\left(\frac{120 - \mu}{\sigma/\sqrt{n}} \leq \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \leq \frac{130 - \mu}{\sigma/\sqrt{n}}\right) = P\left(Z \leq \frac{130 - \mu}{\sigma/\sqrt{n}}\right) - P\left(Z \leq \frac{120 - \mu}{\sigma/\sqrt{n}}\right) \quad (2)$$

Et on déduit :

```

b1<-120                                  pnorm(z1)-pnorm(z2)
nsubj<-10                                 #Directement
z1<-(a1-moy)/(ect/sqrt(nsubj))           pnorm(a1,moy,ect/sqrt(nsubj))-
z2<-(b1-moy)/(ect/sqrt(nsubj))           pnorm(b1,moy,ect/sqrt(nsubj))

```

### Nombre de sujets nécessaires pour que la probabilité ( 1) soit d'au moins 95% :

```

nsn<-(qnorm(0.95)*ect/(a1-moy))**2
nsn<-ceiling(nsn)

z<-(a1-moy)/(ect/sqrt(nsn))
pnorm(z)

```

## 3 Cours 3

### Remise en jambe (1) :

```
vec1<-rnorm(20,70,sqrt(10))          age=vec2,douleur=vec3)
vec2<-rnorm(20,25,sqrt(4))          mean(essai$poids)
vec3<-trunc(runif(20,0,5))          var(essai$poids)
essai<-data.frame(poids=vec1,
```

### 3.1 Tests statistiques

#### Tests de moyenne et de variance

Avec la base ci-dessus :

```
t.test(essai[,1]~as.integer(essai[,3]>=2))
wilcox.test(essai[,1]~as.integer(essai[,3]>=2))
#ou
wilcox.test(essai[essai[,3]>=2,1],essai[essai[,3]<2,1])
wilcox.test(essai[,1]~as.integer(essai[,3]>=2),exact=T)
```

Avec un test de Wilcoxon

```
A<-c(0,1,2)                          A<-c(0,1,2,3,4)
B<-c(100,150,5000)                    wilcox.test(A,B)
wilcox.test(A,B)                      B<-c(100,150,5000,6000,500)
B<-c(100,150,5000,6000)              A<-c(0,1,2,3,4,5)
wilcox.test(A,B)                    wilcox.test(A,B)
```

Malgré la différence des moyennes, il faut au moins 4 éléments à A et B pour arriver à détecter une différence.

Le test t fait même moins bien...

```
A<-c(0,1,2)                          t.test(A,B)
B<-c(100,150,5000)                    B<-c(100,150,5000,6000,500)
t.test(A,B)                          A<-c(0,1,2,3,4,5)
B<-c(100,150,5000,6000)              t.test(A,B)
A<-c(0,1,2,3,4)
```

## ANOVA

```
airquality[1:10,]
t.test(airquality$Ozone[airquality$Month==5],
       airquality$Ozone[airquality$Month==8])
#ou
ozconc<-airquality$Ozone
ozmonth<-airquality$Month
t.test(ozconc[ozmonth==5],ozconc[ozmonth==8])
wilcox.test(ozconc[ozmonth==5],ozconc[ozmonth==8])

anova(lm(Ozone~as.factor(Month),data=airquality))
kruskal.test(Ozone~as.factor(Month),data=airquality)
```

## Tests de distribution, $\chi^2$

Exercice sur airquality :

```
ozconc<-airquality$Ozone
oztemp<-airquality$Temp
v1<-ozconc[!is.na(oztemp) & !is.na(ozconc)]
oztemp<-oztemp[!is.na(oztemp) & !is.na(ozconc)]
ozconc<-v1
x1<-matrix(c(length(ozconc[ozconc>75 & oztemp>85]),
             length(ozconc[ozconc>75 & oztemp<=85]),length(ozconc[ozconc<=75 & oztemp>85]),
             length(ozconc[ozconc<=75 & oztemp<=85])),ncol=2,byrow=T)

chisq.test(x1)
ks.test(airquality$Ozone,"pnorm")
```

## 3.2 Graphes

### Graphes des concentrations d'ozone les 20 premiers jours de mai

```
tit<-"Ozone observée les 20 premiers jours du mois de mai"
ozc<-airquality$Ozone
ozm<-airquality$Month
ozd<-airquality$Day

plot(ozd[ozm==5],ozc[ozm==5],type="b",xlab="jours",
     xlim=c(0,20),ylim=c(0,50),pch=3,ylab="ozone",main=tit)
```

## Graphes de la relation entre l'ozone et le vent, selon la température

```
tit<-"Relation entre l'ozone et le vent, selon la température"
plot(ozw[!is.na(ozt) & ozt<85],ozc[!is.na(ozt) & ozt<85],xlab="Vent (mph)",
ylab="Ozone (ppb)",pch=2,xlim=c(2,15),ylim=c(35,125),main=tit)
points(ozw[!is.na(ozt) & ozt>=85],ozc[!is.na(ozt) & ozt>=85],pch=6)
legend(12,120,c("Temp>=85°F","Temp<85°F"),pch=c(2,6))
```

## Histogrammes de la base swiss

```
idens<-c(-1,-1,25,-1)
icol<-c("orange","white","red","peachpuff")
ibord<-c("gray0","red","red","red")
par(mfrow=c(2,2))
for(i in 2:5) {
hist(swiss[,i],xlab=names(swiss)[i],main="",breaks=20,border=ibord[i-1],
col=icol[i-1],density=idens[i-1])
}
```

## Boîtes à moustache de la base swiss

```
binedu<-as.integer(swiss$Education>10)
binmor<-as.integer(swiss$Infant.Mortality>20)
par(mfrow=c(1,2))
boxplot(swiss$Fertility~binedu,xlab="Education",ylab="Fertility",
boxwex=0.2,col="orange")
legend(0.5,45,c("0: Edu<10","1: Edu>10"))
boxplot(swiss$Fertility~binmor,xlab="Education",ylab="Fertility",
boxwex=0.2,col="red")
legend(1.2,45,c("0: Inf.Mort<20","1: Inf.Mort>20"))
```

## Exercice

Densité de 4 lois :

```
par(mfrow=c(2,2))
#N(0,1)
xpl<-c(-50:50)/10
ypl<-dnorm(xpl)
plot(xpl,ypl,xlab="X",ylab="Densite loi normale",type="l")

#loi log-normale
```

```

xpl<-c(0:50)/10
ypl<-dlnorm(xpl)
plot(xpl,ypl,xlab="X",ylab="Densite loi log-normale",type="l")

#loi de Poisson
lambda<-2
xpl<-c(0:40)/2
ypl<-dpois(xpl,lambda)
plot(xpl,ypl,xlab="X",ylab=paste("Densite loi de Poisson (lambda=",lambda,")",
sep=" "),type="l")

#loi Gamma
gam<-2
bet<-1
xpl<-c(0:50)/5
ypl<-dgamma(xpl,gam,bet)
plot(xpl,ypl,xlab="X",ylab=paste("Densite loi Gamma (gamma=",gam,", beta=",
bet,")",sep=""),type="l")

```

## 4 Cours 4

### Remise en jambe (1) :

```

?women
dat<-women
dat[,1]<-dat[,1]*2.5
dat[,2]<-dat[,2]/2
plot(dat[,1],dat[,2],xlab="Taille (cm)",ylab="Poids (kg)",type="b",
main="Taille et poids moyen chez des femmes américaines de 30 à 39 ans")

```

### Remise en jambe (2) :

```

?CO2
dat<-CO2
par(mfrow=c(1,2))
hist(dat[dat[,3]=="chilled",5],xlab="CO2 uptake",main="Chilled")
hist(dat[dat[,3]=="nonchilled",5],xlab="CO2 uptake",main="Non-chilled")
wilcox.test(dat[,5]~dat[,3])

```

## 4.1 Analyses statistiques

### Régression linéaire :

```
library(MASS)
pairs(cats)
cats.lm <- lm(Hwt~Bwt*Sex,data=cats)
summary(cats.lm)

par(mfrow=c(1,1))
qqnorm(studres(cats.lm))
qqline(studres(cats.lm))

plot(predict(cats.lm),studres(cats.lm),xlab="Predictions",
ylab="Residus standardises")
abline(h=0)
plot(cats.lm,which=4)

attributes(summary(cats.lm))
summary(cats.lm)$r.squared
summary(cats.lm)$df
summary(cats.lm)$cov.unscaled

cats.lm1<-update(cats.lm,Hwt~Bwt+Sex)
summary(cats.lm1)
anova(cats.lm1,cats.lm)
```

### Régression logistique :

```
library(MASS)
?birthwt
lbwt <- data.frame(low=factor(birthwt$low),age=birthwt$age,ptl=birthwt$ptl,
smoke=(birthwt$smoke>0),ht=(birthwt$ht>0))
lbwt.glm <- glm(low~age+ptl+smoke+ht,data=birthwt,family=binomial)
summary(lbwt.glm)
drop1(glm(low~age+ptl+smoke+ht,data=birthwt,family=binomial),test="Chisq")

lbwt.glm1<-update(lbwt.glm,.-smoke)
anova(lbwt.glm1,lbwt.glm,test="Chisq")
drop1(lbwt.glm1,test="Chisq")
```

```

lbwt.glm2<-update(lbwt.glm1, .~-age)
anova(lbwt.glm2, lbwt.glm1, test="Chisq")
drop1(lbwt.glm2, test="Chisq")
lbwt.glm3<-update(lbwt.glm2, .~pt1*ht)
summary(lbwt.glm3)
# on reste avec lbwt.glm2
summary(lbwt.glm2)

plot(lbwt.glm2)

p1<-predict(lbwt.glm2, type="response")
p1<-as.integer(p1>=0.5)
table(p1, birthwt$low)

#Avec step
scope.lm<-paste(names(birthwt)[2:10], collapse="+")
scope.lm<-paste(names(birthwt)[1], scope.lm, sep="~")
lbwt.full <- glm(as.formula(scope.lm), family=binomial, data=birthwt)
step(lbwt.full)

```

### **Calcul du nombre de sujets nécessaire :**

```

power.prop.test(power=0.95, p1=0.4, p2=0.6)

xpmin<-0.42
delmin<-xpmin-0.4
for(xp2 in seq(0.6, xpmin, by=-0.02)) {
y<-power.prop.test(n=1000, p1=0.4, p2=xp2)
cat("delta=", xp2-0.4, " puissance=", y$power, "\n")
if(y$power>=0.95) delmin<-xp2-0.4
}
y<-power.prop.test(n=1000, p1=0.4, p2=0.4+delmin)
cat("Avec une puissance de ", y$power, " on detecte un effet de ", delmin, "\n")

```

## **4.2 Boucles**

### **Ranger les élèves :**

Prendre un élève au hasard et le mettre debout

Pour chaque élève encore assis :

- \* se lever
- \* se comparer à l'élève debout le plus à gauche
- \* si l'élève est plus grand, avancer d'un cran
- \* recommencer tant que l'élève n'a pas atteint la fin de la ligne

Arrêt de l'algorithme lorsque tous les élèves sont debout

### Tests :

```
{
x<-scan("",nlines=1)
if(x>0) cat("Ce nombre est positif\n") else cat("Ce nombre est negatif\n")
}
```

Les accolades permettent d'exécuter un bloc de commande en une seule fois. Comparez avec et sans les accolades.

Le script suivant doit être exécuté en deux temps, d'abord les 2 premières lignes puis les suivantes :

```
itir<-sample(0:100,1)
x<-scan("",nlines=1)
if(x==itir) cat("C'est gagné!\n") else {
  if(x>itir) cat("Trop haut\n") else cat("Trop bas\n")
  cat("Le chiffre était :",itir,"\n")
}
```

Une façon plus jolie de faire consiste à définir une fonction (voir cours 5) :

```
devinez<-function() {
  itir<-sample(0:100,1)
  x<-scan("",nlines=1)
  if(x==itir) cat("C'est gagné!\n") else {
    if(x>itir) cat("Trop haut\n") else cat("Trop bas\n")
    cat("Le chiffre était :",itir,"\n")
  }
}
```

puis de l'exécuter :

```
> devinez()
1: 45
Read 1 item
Trop haut
Le chiffre était : 35
```

Exercice supplémentaire : modifier la fonction pour jouer (nécessite les boucles dans la suite du cours 4) à deviner le bon nombre par essais successifs.

### Boucles

```
x<-matrix(1:120,ncol=12)
#Ne pas oublier de définir et initialiser lmoy avant la boucle
lmoy<-c()
for(i in 1:dim(x)[1]) lmoy<-c(lmoy,mean(x[i,]))
#autre possibilité
lmoy<-NULL
# ou lmoy<-0
for(i in 1:dim(x)[1]) lmoy[i]<-mean(x[i,])

cmoy<-c()
for(i in 1:dim(x)[2]) cmoy<-c(cmoy,mean(x[,i]))
#Note : on peut aussi utiliser apply
apply(x,1,mean)
apply(x,2,mean)
```

### Exercice :

```
{
for(i in 1:3) cat(LETTERS[i])
cat("\n")
}
{
for(i in c(5,24,5,18,3,9,3,5))
  cat(letters[i])
cat("\n")
}
x<-matrix(1:120,ncol=12)
rowSums(x)
apply(x,1,sum)
colSums(x)
apply(x,2,sum)
rowMeans(x)
apply(x,1,mean)
colMeans(x)
apply(x,2,mean)
```

## 4.3 TCL

### Exercice d'illustration du TLC, tirage avec nobs=3 observations :

```
x<-rexp(500,5)
hist(x)
nobs<-3
```

```
x<-rexp(nobs,5)
mean(x)
moy<-c()
for(i in 1:500) moy<-c(moy,mean(rexp(nobs,5)))
hist(moy)
```

### **Boucle pour faire varier nobs :**

```
nobs<-c(3,5,10,30)
par(mfrow=c(2,2))
for (i in nobs) {
  moy<-c()
  for (j in 1:50) {
    vec<-rexp(i,5)
    moy<-c(moy,mean(vec))
    hist(moy,breaks=20)
  }
}
```

Cet exercice est une illustration du théorème de la limite centrale : lorsque nobs est petit, la distribution de la moyenne de nobs observations est proche de la distribution de la variable échantillonnée. En revanche quand nobs augmente, la distribution des moyennes se rapproche de celle d'une loi normale. On constate que pour la loi exponentielle, il faut au moins attendre nobs=30 pour commencer à avoir quelque chose de satisfaisant.

### **Même exercice pour une loi uniforme :**

```
par(mfrow=c(2,2))
for (i in nobs) {
  moy<-c()
  for (j in 1:100) {
    vec<-runif(i)
    moy<-c(moy,mean(vec))
    hist(moy,breaks=20)
  }
}
```

Ici, l'histogramme commence à ressembler à celui d'une loi normale presque tout de suite.

## **5 Cours 5**

### **Remise en jambe :**

```
library("ISwR")
plot(bp~obese,pch = ifelse(sex==1, 1,2), data = bp.obese,xlab="Obesity ratio",
ylab="Blood pressure (mm Hg)")
legend(2,200,c("Women","Men"),pch=1:2)

y<-lm(bp~obese,data = bp.obese)
```

```

summary(y)
y2<-lm(bp~obese+sex,data = bp.obese)
summary(y2)
anova(y2,y,test="Chisq")
co<-coef(y2)
plot(bp~obese,pch = ifelse(sex==1, 1,2), data = bp.obese,xlab="Obesity ratio",
ylab="Blood pressure (mm Hg)")
abline(y)
abline(a=co[1]+co[3],b=co[2],col="red",lty=2)
abline(a=co[1],b=co[2],col="blue",lty=2)
legend(1.6,200,c("Both (model 1)","Women (model 2)","Men (model 2)"),lty=c(1,2,2),
col=c("black","red","blue"))

```

## 5.1 Chaînes de caractères

```

grep("er$",month.name) # renvoie les mois finissant en er (en anglais)
grep("^[A-J]",month.name) # renvoie les mois commençant par une lettre
# entre A et J (en anglais)
grep("^[^J]",month.name) # renvoie les mois ne commençant pas par J

```

```

chaine<-"chaine"
gsub("e","i",chaine)
gsub("[a-e]","x",chaine)

```

### Exercices sur grep :

```

chaine<-"Ceci est une chaine"
gsub("e","i",chaine)
gsub("[a-e]","x",chaine)

```

### Manipulation de chaînes :

```

eleve<-c("Anne Dubois","Julie Bertrand","Emmanuelle Comets","Caroline Bazzoli",
"Hervé Le Nagard")

```

Ici je définis 2 fonctions pour extraire les 2 parties prénoms/nom (voir deuxième partie du cours 5). Je suppose pour ne pas compliquer les choses que si le prénom est composé, c'est de la forme Jean-Pierre (et que donc la première case du vecteur extrait contient le prénom en entier).

```

extract.prenom<-function(vec) {
return(vec[1])}

```

```

extract.nom<-function(vec) {
return(paste(vec[2:length(vec)],collapse=" "))}

eleve.split<-strsplit(eleve," ")
eleve.prenom<-unlist(lapply(eleve.split,extract.prenom))
eleve.nom<-unlist(lapply(eleve.split,extract.nom))
print(eleve.prenom)
print(eleve.nom)

eleve.format<-paste(eleve.prenom,eleve.nom,sep="-")
print(eleve.format)

eleve.prenom[grepl("i",eleve.prenom) | grepl("I",eleve.prenom)]
eleve.prenom[grepl("^[A-M]",eleve.prenom)]

```

### **Lecture/écriture :**

```

date.naiss<-c("20-2-1978","12-10-1978","25-05-1971","8-8-1979","1-6-1971")
anniv<-data.frame(prenom=eleve.prenom,nom=eleve.nom,naiss=date.naiss)
write.table(anniv,"Anniversaires.txt",row.names=F)

anniv2<-read.table("Anniversaires.txt",header=T)
all.equal(anniv,anniv2)

vec<-strsplit(date.naiss,"-")
date.bis<-unlist(lapply(vec,paste,collapse="/"))
anniv2[,3]<-date.bis
anniv[order(anniv[,2]),]

demog<-data.frame(prenom=eleve.prenom[order(eleve.nom)],
nom=sort(eleve.nom),taille=c(1.7,1.85,1.75,1.65))
write.csv(demog,"demog.txt",row.names=F)

#Fichier contenant noms et taille
demog<-data.frame(prenom=eleve.prenom[order(eleve.nom)],
nom=sort(eleve.nom),taille=c(1.7,1.85,1.75,1.65))
write.csv(demog,"demog.txt",row.names=F)

#Nombre d'élèves nés ce mois

```

```

mat.anniv<-matrix(as.integer(unlist(strsplit(date.naiss,"-"))),ncol=3,byrow=T)
mois.naiss<-rep(1,12)
for (i in 1:12) {
nba<-length(mat.anniv[mat.anniv[,2]==i,1])
if(nba>0) cat("Nb d'anniversaires en",month.name[i],":",nba,"\n")
mois.naiss[i]<-nba
}
plot(c(1:12),mois.naiss,type="h")

if(length(unique(date.naiss))<length(date.naiss))
cat("Au moins 2 élèves sont nés le même jour")

#Histogramme des années de naissance
hist(mat.anniv[,3])

#Calcul de l'âge
now<-date()
dal<-unlist(strsplit(now," "))
dal<-dal[dal!=""]
mon<-pmatch(dal[2],month.name)
day<-as.integer(dal[3])
year<-as.integer(dal[5])
age<-year-mat.anniv[,3]
for(i in 1:length(age)) {
if(mat.anniv[i,2]>mon | (mat.anniv[i,2]==mon & mat.anniv[i,1]>day))
age[i]<-age[i]-1
}
# Prochain et dernier anniversaire de l'année
now<-date()
dal<-unlist(strsplit(now," "))
dal<-dal[dal!=""]
yr.now<-as.integer(dal[5])
today<-mdy.date(pmatch(dal[2],month.name),as.integer(dal[3]),yr.now)

jd.anniv<-mdy.date(mat.anniv[,2],mat.anniv[,1],yr.now)
vec<-today-jd.anniv
#on vérifie qu'il y a au moins un anniversaire de passé cette année

```

```

#sinon on considère les anniversaires de l'année précédente
if(length(vec[vec>=0])<=0)
jd.anniv<-mdy.date(mat.anniv[,2],mat.anniv[,1],yr.now-1)
vec<-today-jd.anniv

indx<-1:length(vec)
ind1<-indx[vec==min(vec[vec>=0])]
cat("Le dernier anniversaire a eu lieu le",date.mmddy(jd.anniv[ind1[1]]),"\n")

#reste-t-il au moins un anniversaire de passé cette année
#sinon on considère les anniversaires de l'année prochaine
if(length(vec[vec<0])<=0)
jd.anniv<-mdy.date(mat.anniv[,2],mat.anniv[,1],yr.now+1)
vec<-today-jd.anniv

cat("Le prochain anniversaire aura lieu dans",min(-vec),"jours\n")

```

## 5.2 Fonctions

### Variance biaisée

```

ma.variance<-function(x) {
  sum((x-mean(x))**2)/length(x)
}
x<-1:100
ma.variance(x)
var(x)
ma.variance<-function(x,biased=F) {
  if(biased)

```

```

sum((x-mean(x))**2)/length(x) else
sum((x-mean(x))**2)/(length(x)-1)
}
x<-1:100
ma.variance(x)
var(x)
ma.variance(x,biased=T)

```

### Densité

```

phi <- function(x) {
  exp(-x^2/2) / sqrt(2 * pi)
}

```

```

phi(0)
dnorm(0)

```

**Fonction à corriger** Pour déboguer, exécuter la fonction en lisant attentivement les messages d'erreur (syntaxe, objet non trouvé, avertissements), et modifier petit à petit la fonction.

```
# Fonction originelle
```

```

toto<-mafonction(x,y=true) {
  if(y=T) mean(x*a) else print(mean(x*a,na.rm=T))
  return(res=sum(x),)
}
#Fonction corrigée
toto<-function(x,a,y=TRUE) {
  if(y) print(mean(x*a)) else print(mean(x*a,na.rm=T))
  return(res=sum(x))
}

```

### **C'est la note finale**

```

# Fonction originelle
notes.finales <- function(notes, p) {
  netud <- nrow(notes)
  neval <- ncol(notes)
  final <- (1:netud) * 0
  for(i in 1:netud) {
    for(j in 1:neval) {
      final[i] <- final[i] + notes[i, j] * p[j]
    }
  }
  final
}
#Fonction optimisée
notes.finales2 <- function(notes, p)
  apply(t(notes)*p,2,sum)
#Simulation d'un tableau de notes et de pondérations pour essayer la fonction
ns<-20;np<-5
notes<-matrix(trunc(runif(ns*np,3,21)),ncol=np)
pond<-trunc(runif(4,1,4))/20
pond<-c(pond,1-sum(pond))
pond
#on vérifie que nos deux fonctions renvoient le même résultat...
notes.finales(notes,pond)
notes.finales2(notes,pond)

```

## Le retour du TCL :

```
#Définition de la fonction                                }
extcl<-function(nobs,ntir=1) {                            #Exécution
  moy<-c()                                                nobs<-c(3,5,10,30)
  for(i in 1:ntir) {                                     ntir<-50
    vec<-runif(nobs)                                     par(mfrow=c(2,2))
    moy<-c(moy,mean(vec))                               for (i in nobs)
  }                                                      hist(extcl(nobs,ntir),breaks=20)
return(moy)
```

On peut mais c'est beaucoup plus sioux, inclure le type de distribution et ses paramètres dans la définition de la fonction. Cela fait appel à une nouvelle fonction, `do.call`, qui admet comme argument le nom d'une fonction et une liste avec ses arguments. Ici, on va donner le nom de la fonction comme une chaîne de caractères, et utiliser l'argument `"..."` pour permettre de spécifier des arguments à donner à cette fonction. Lorsque R reçoit des arguments à la place de `"..."`, il les répercute à la fonction où ces `"..."` apparaissent, ici dans le `do.call`.

```
extcl2<-function(nobs,ntir=1,fonc="rnorm",...) {
  moy<-c()
  for(i in 1:ntir) {
    vec<-do.call(fonc,list(nobs,...))
    moy<-c(moy,mean(vec))
  }
  return(moy)
}
```

On peut alors utiliser cette fonction avec toutes les fonctions de tirage connues par R :

```
nobs<-c(3,5,10,30)
ntir<-50
par(mfrow=c(2,2))
for (i in nobs)
  hist(extcl2(i,ntir,"runif",0,1),xlab=paste(i,"tirages"),
  main="Uniform distribution",breaks=20)
par(mfrow=c(2,2))
for (i in nobs)
  hist(extcl2(i,ntir,"rexp",5),xlab=paste(i,"tirages"),
  main="Exponential distribution",breaks=20)
```

### 5.3 Concepts avancés

```
rpareto <- function(n, alpha, lambda)
  lambda * (runif(n)^(-1/alpha) - 1)
malist<-vector(length=5,mode="list")
il<-1
for(i in c(100,150,200,250,300)) {
  malist[[il]]<-rpareto(i,2,5000)
  il<-il+1
}
names(malist)<-paste("echantillon",1:5,sep=" ")
vec<-lapply(malist,mean)
ppareto<-function(x,alpha,lambda) {
  1-exp(alpha*log(lambda/(x+lambda)))
}
dpareto<-function(x,alpha,lambda) {
  alpha*(lambda^alpha)/exp((alpha+1)*log(x+lambda))
}
vec1<-sort(ppareto(unlist(malist),2,5000))
hist(malist[[5]])
hist(malist$echantillon5)
vec2<-unlist(lapply(malist, function(x) sort(ppareto(x, 2,5000))))
vec3<-lapply(lapply(malist, sort), ppareto, alpha = 2,lambda = 5000)
```

### 5.4 Librairies

#### Librairie combinat :

```
library(combinat)
x<-c("rouge", "orange", "jaune", "vert", "bleu", "indigo", "violet")
tab<-permn(x)
length(tab)
#note : ce nombre est évidemment égal à
factorial(length(x))

tab<-combn(x,4)
dim(tab)
choose(7,4)
```

## Librairie date :

```
now<-date()

naiss<-mdy.date(5,25,1971)
today<-mdy.date(pmatch(da1[2],month.name),as.integer(da1[3]),as.integer(da1[5]))
cat("Je suis née depuis ",today-naiss,"jours\n") #tout ça...

mon<-pmatch(da1[2],month.name)+2
yea<-as.integer(da1[5])
while(mon>=13) {
mon<-mon-12
yea<-yea+1}
unmois<-paste(as.integer(da1[3]),mon,yea,sep="/")
cat("Dans deux mois, nous serons le",unmois,"\n")

da1<-unlist(strsplit(now," "))
da1<-da1[da1!=""]
today<-paste(da1[3],da1[2],da1[5])
cat("Aujourd'hui, nous sommes le",today,"\n")
cat("      et il est",da1[4],"\n")
heur<-as.integer(unlist(strsplit(da1[4],":")))
heur[1]<-heur[1]+1
if(heur[1]==24) heur[1]<-0
#on ne sait jamais, vous révisez peut-être à minuit
heur2<-paste(heur,collapse=":")
cat("      dans une heure il sera",heur2,"\n")
```