Examen

Thomas Vrignaud - Komi Agblodoe

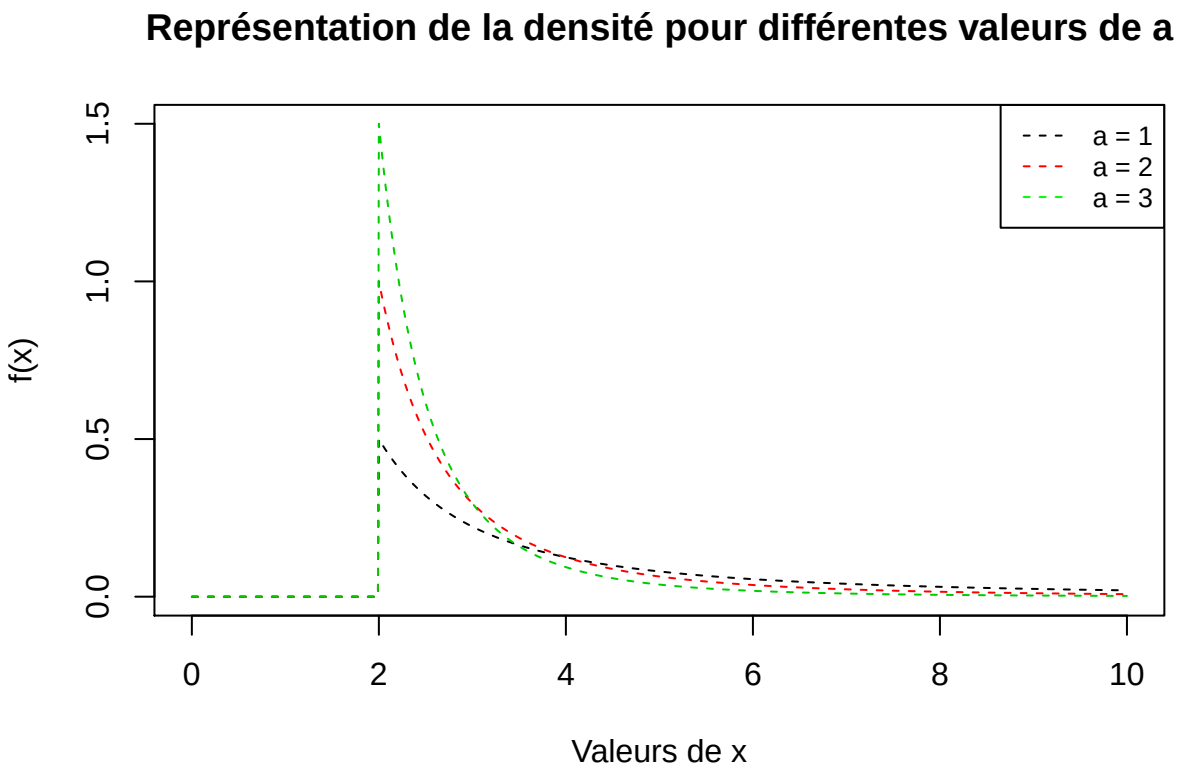
21/10/2019

Exercice 1 - Simulation par Inversion

On considère la fonction densité suivante, définie pour $a > 0, b > 0$:

$$\begin{cases} f(x) = \frac{ab^a}{x^{a+1}} \text{ si } x > b \\ 0 \text{ sinon} \end{cases}$$

1. Représenter cette densité pour $b = 2$ et $a1, 2, 3$.



Avec le graphique précédent, nous avons la densité définit auparavant pour pour trois valeurs de a différentes. Les trois courbes ont été affichées sur le graphiques pour pouvoir être comparées. Nous voyons que plus a augmente plus la valeur lorsque $x = b$ est élevée.

2. Implémenter une procédure d'inversion pour simuler une variable aléatoire selon cette densité.

Nous savons que la fonction est défini par :

$$f(x) = \frac{ab^a}{x^a + 1} \text{ si } x > b$$

Donc :

$$F(x) = \int_b^x ab^a y^{-a-1} dy \text{ si } y > b$$

Alors :

$$F(x) = [-b^a.y^{-a}]_b^x \text{ si } y > b$$
$$F(x) = -b^a.x^{-a} + b^a.b^{-a} \text{ si } x > b$$

Nous avons donc :

$$F(x) = 1 - b^a.x^{-a} \text{ si } x > b$$

Nous posons $ u U(0,1) $. Si $ u U(0,1) $ alors $ 1-u U(0,1) $.

$$u = 1 - b^a.x^{-a}$$

Alors

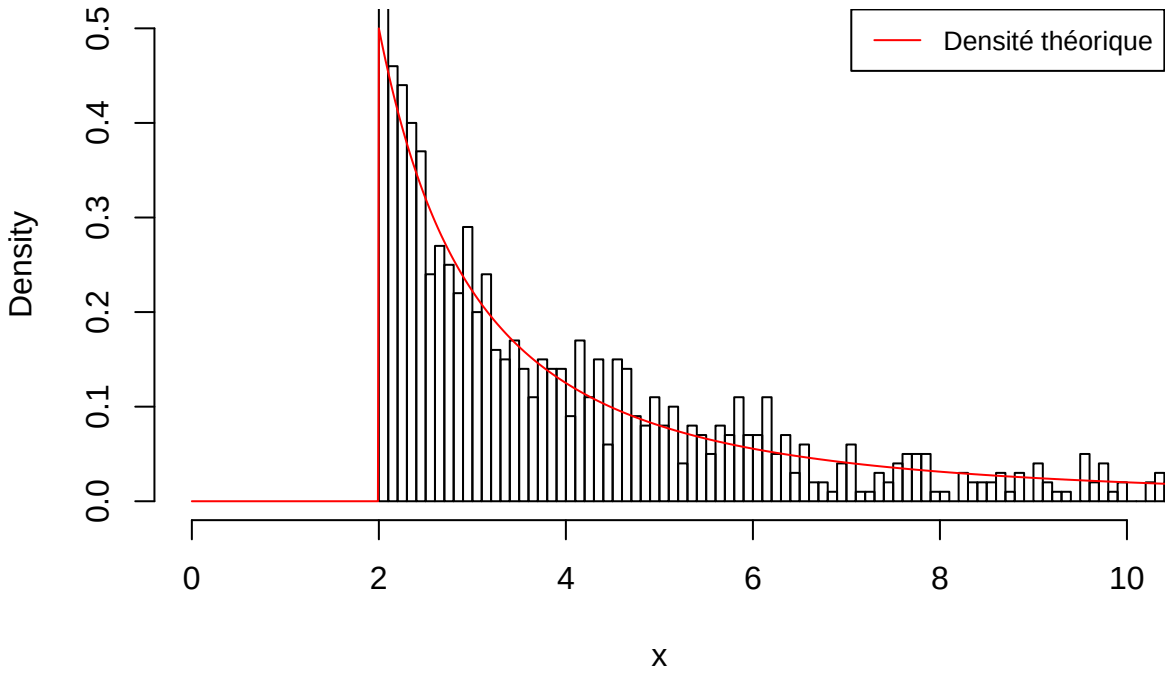
$$x = \frac{b}{(1-u)^{\frac{1}{a}}}$$

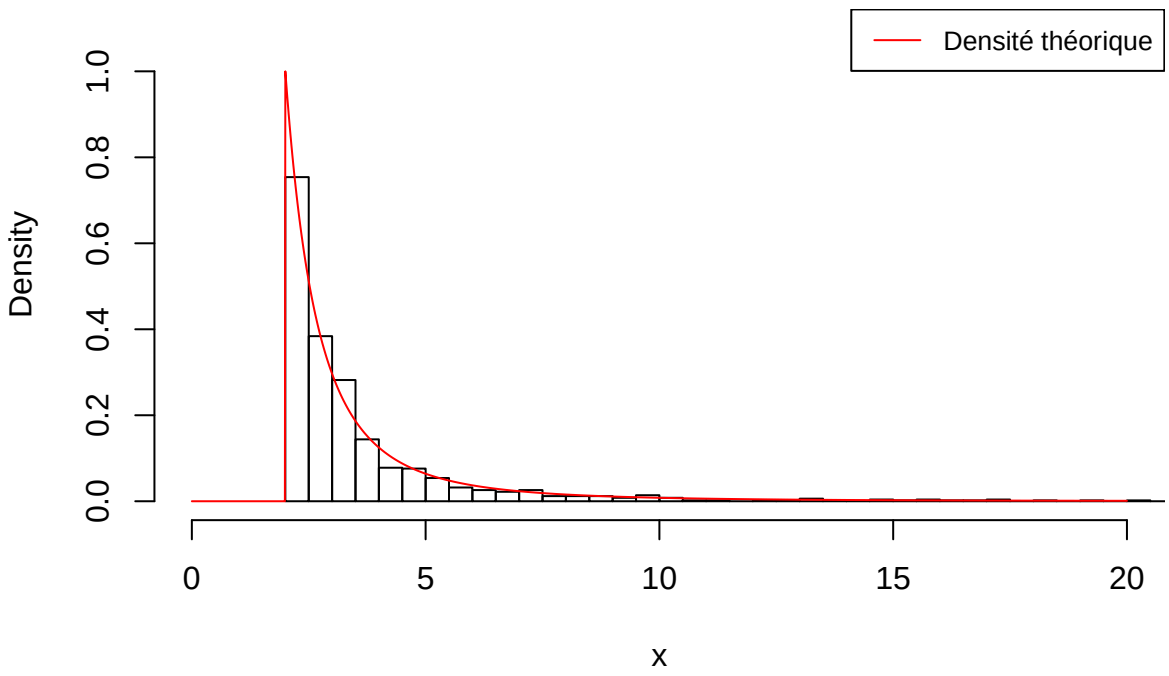
Avec nous pouvons donc utiliser cette formule pour mettre en place cette procédure d'inversion tel que $y = \frac{b}{(1-u)^{\frac{1}{a}}}$ avec $uU(0,1)$

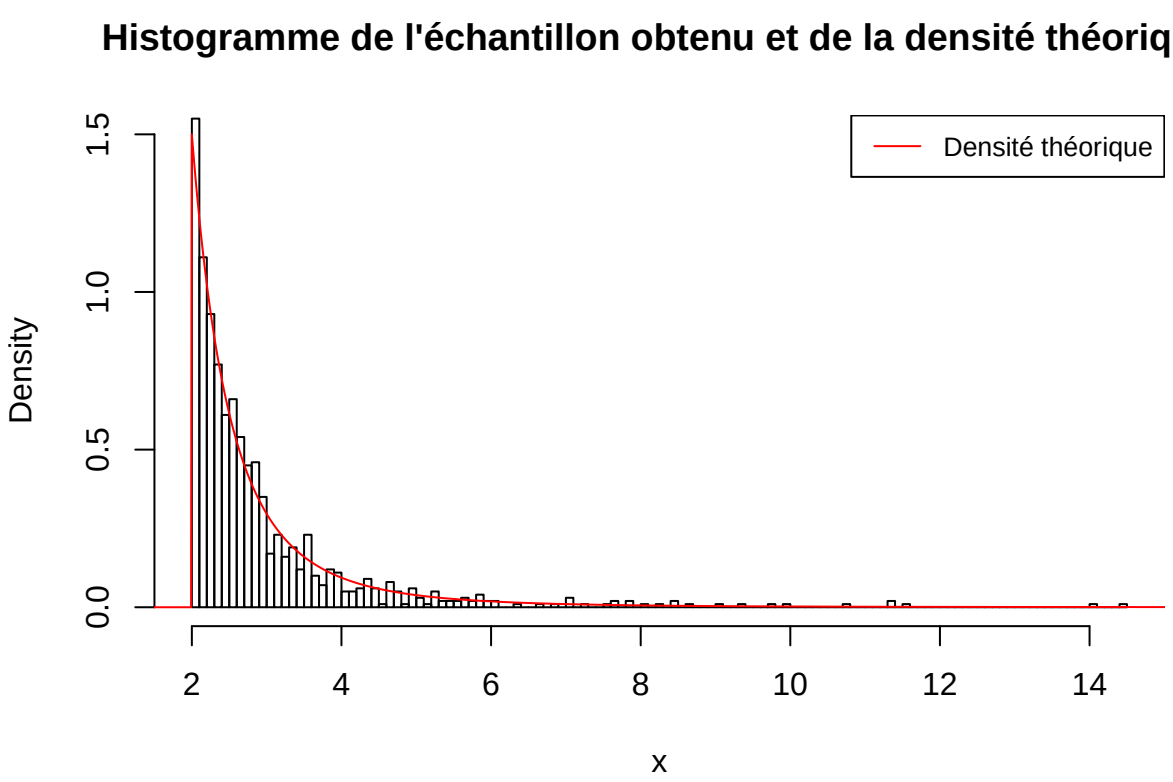
```
#fonction utilisation procédure inversion pour simulation
inverse <- function(a,b,u) {
  b/ ((1-u)**(1/a))
}
```

Nous avons donc mise en place une fonction permettant de simuler un échantillon de cette densité par une méthode d'inversion. Cette méthode va être utilisée à la question suivante.

3. Simuler un échantillon et comparer graphiquement la densité obtenue empiriquement à la densité théorique.


Histogramme de l'échantillon obtenu et de la densité théorique
Densité théorique
Density
0.0
0.1
0.2
0.3
0.4
0.5
0
2
4
6
8
10
x


Histogramme de l'échantillon obtenu et de la densité théorique
Densité théorique
Density
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
x



Nous avons simulé un échantillon à partir de la procédure d'inversion pour les différentes valeurs de a. Nous voyons que la densité théorique ajuste plutôt bien les valeurs empiriques.

Exercice 2 - Approximation de la fonction de répartition de la loi normale centrée réduite par approche MC

On cherche à approximer par une approche Monte-Carlo la fonction de répartition de la loi normale centrée réduite :

$$\phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}}, \mathrm{d}t$$

On peut réécrire cette fonction tel que :

$$\phi(x) = \frac{1}{n} \sum_{i=1}^{n} g(X_i)$$

1. Proposer une méthode basée sur la loi uniforme, et de préférence la loi U(0, 1).

Dans cette partie, nous proposon une méthode basée sur la loi uniforme. Pour cela, nous modifions la fonction de répartition donné ci-dessus. Sachant que $x \geq 0$, nous avons :

$$\phi(x) = \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} \mathrm{d}t + \int_{0}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} \mathrm{d}t$$

$$\phi(x) = \phi(0) + \int_0^x \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} \mathrm{dt}$$

avec $\phi(0) = 0.5$ car nous savons que la densité de la loi normale centrée réduite est symétrique autour de 0.
Nous posons ensuite

$$u = \frac{t}{x}$$

, nous obtenons donc :

$$\phi(x) = 0.5 + \int_0^1 \frac{x}{\sqrt{2\pi}} e^{\frac{-x^2u^2}{2}} \mathrm{du}$$

Alors :

$$\phi(x) = 0.5 + \int_{-\infty}^{\infty} I_{(0,1)}(u) \frac{x}{\sqrt{2\pi}} e^{\frac{-x^2u^2}{2}} \mathrm{du}$$

Donc :

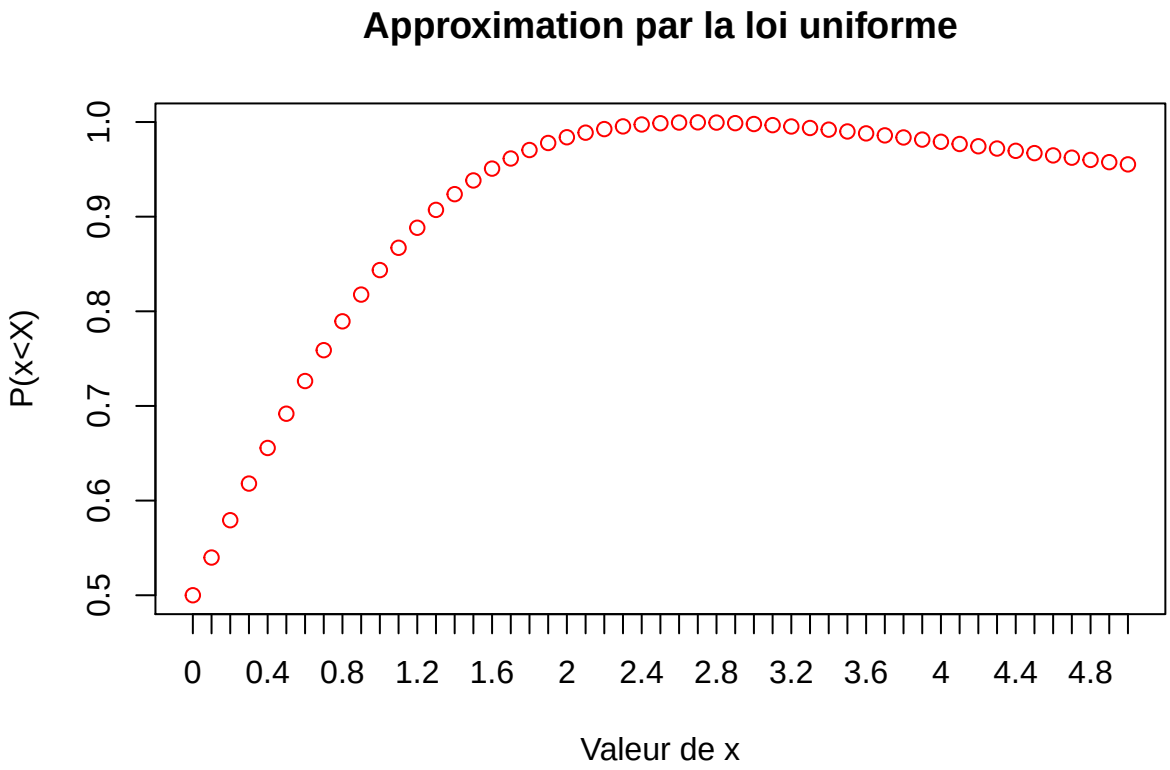
$$\phi(x) = 0.5 + E(g(U))$$

où u suit une loi uniforme $U(0,1)$ et nous avons

$$g(u) = \frac{x}{\sqrt{2\pi}} e^{\frac{-x^2u^2}{2}}$$

Avec tous ces calculs, nous pouvons approximer la fonction de répartition donnée initailement en calculant :

$$\phi(x) = 0.5 + \frac{1}{n} \sum_{i=1}^{n} g(U_i)$$



Pour $x = 0$, la probabilité est bien de 0.5 ce qui est cohérent. Ensuite, la courbe est croissante et semble tendre vers 1, ce se rapproche de la réalité. Cependant, une décroissance semble s'observer lorsque x augmente, ce qui ne correspond pas à la réparition théorique.

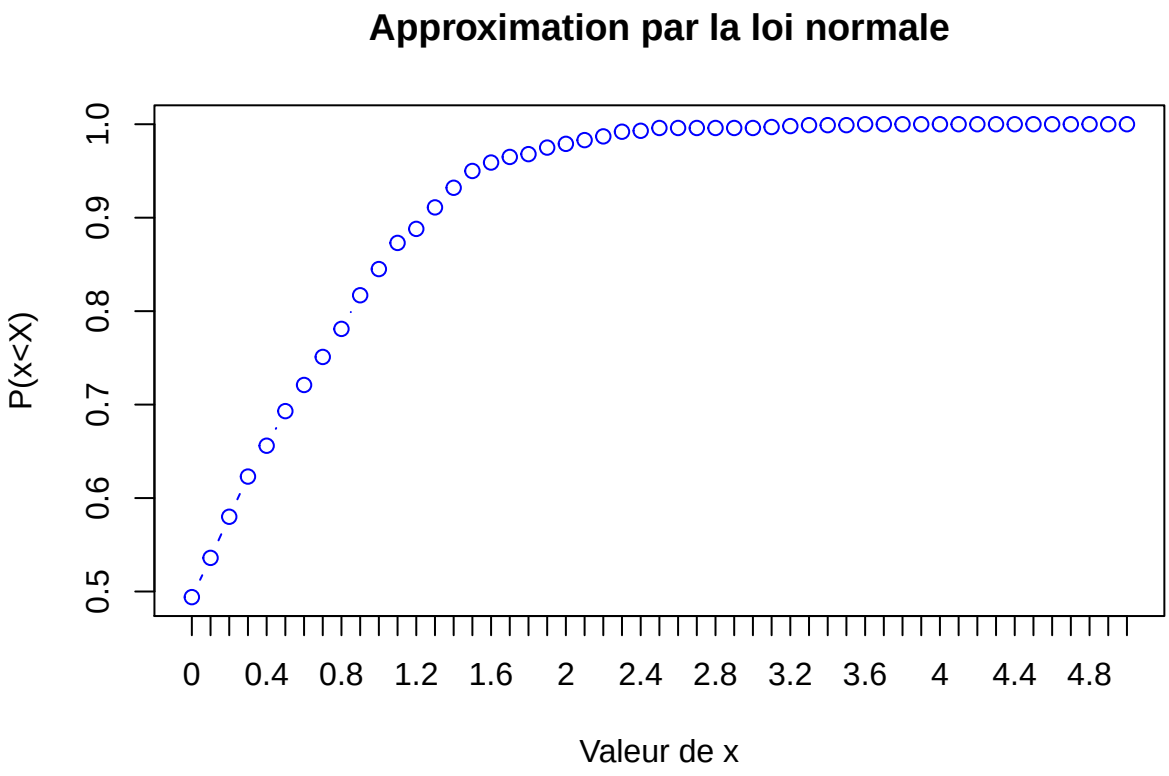
2. Proposer une méthode basée sur la loi normale

Pour la méthode la loi normale cela est beaucoup plus simple. Il sufit de modifier les limites de notre intégrale.

$$\phi(x) = \int_{-\infty}^{\infty} I_{(-\infty,x)}(t) \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} \mathrm{d}t$$

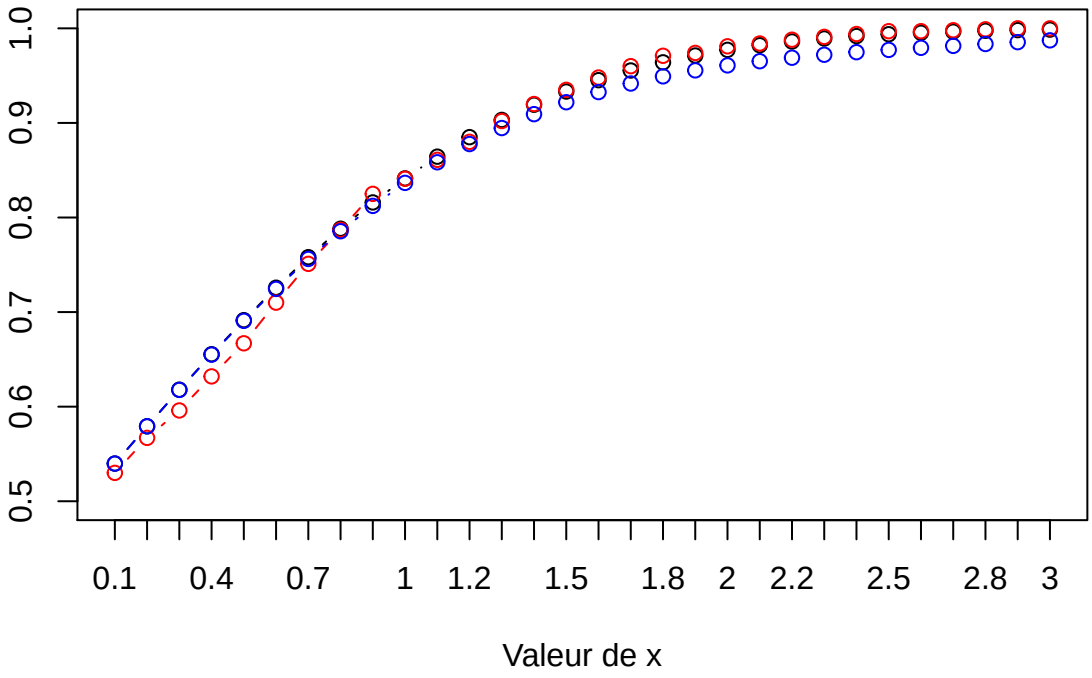
A noter, cela revient à calculer la fonction de répartition empirique pour l'échantillon $(X_1, X_2, ..., X_n)$ donc nous pouvons écrire

$$\hat{\phi}(x) = \hat{F}_n(x)$$



Les résultats nous montrant bien que nous retombons sur la fonction de répartition de la loi normale centrée réduite.

3. Comparer les valeurs obtenues par les deux procédures à la valeur réelle donnée par R via la fonction pnorm pour des valeurs croissantes de $x[0.1, 3]$. On considérera un nombre de tirages n = 1000.



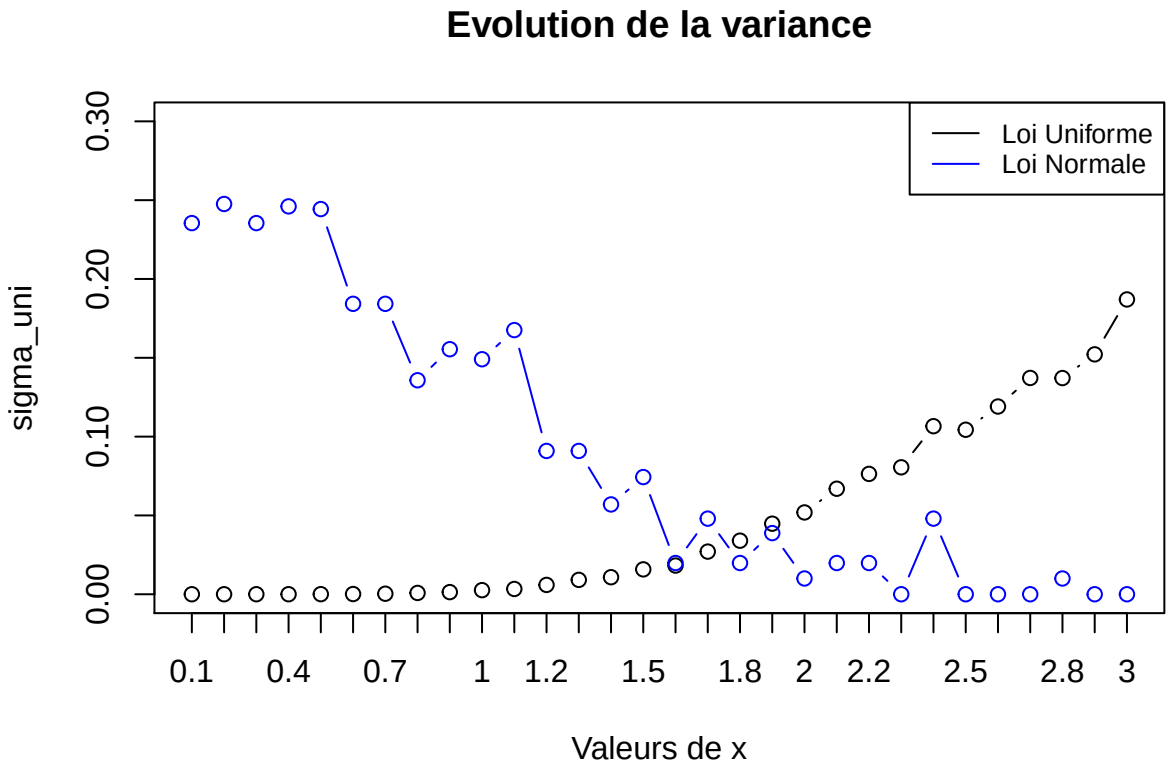
Nous voyons que les deux méthodes que nous avons estimées sont assez proche de la réalité représentée par la courbe noire qui est la fonction de répartition théorique. Les deux méthodes d'approximations semble être cohérente.

4. Comparer la variance et les intervalles de confiance des deux estimateurs pour les valeurs croissantes de $x[0.1, 3]$ et interpréter les résultats obtenus.

Nous souhaitons maintenant déterminer la variance empirique et l'intervalle de confiance pour $\phi(x)$.

On peut estimer la variance théorique σ^2 par la variance empirque qui est donné par

$$\overline{\sigma}_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)^2$$



Nous voyons que pour la loi uniforme la variance augmente au fur et à mesure que x augmente alors que pour la loi normale c'est l'inverse, la variance diminue au fur et à mesure que les valeurs de x augmentent.

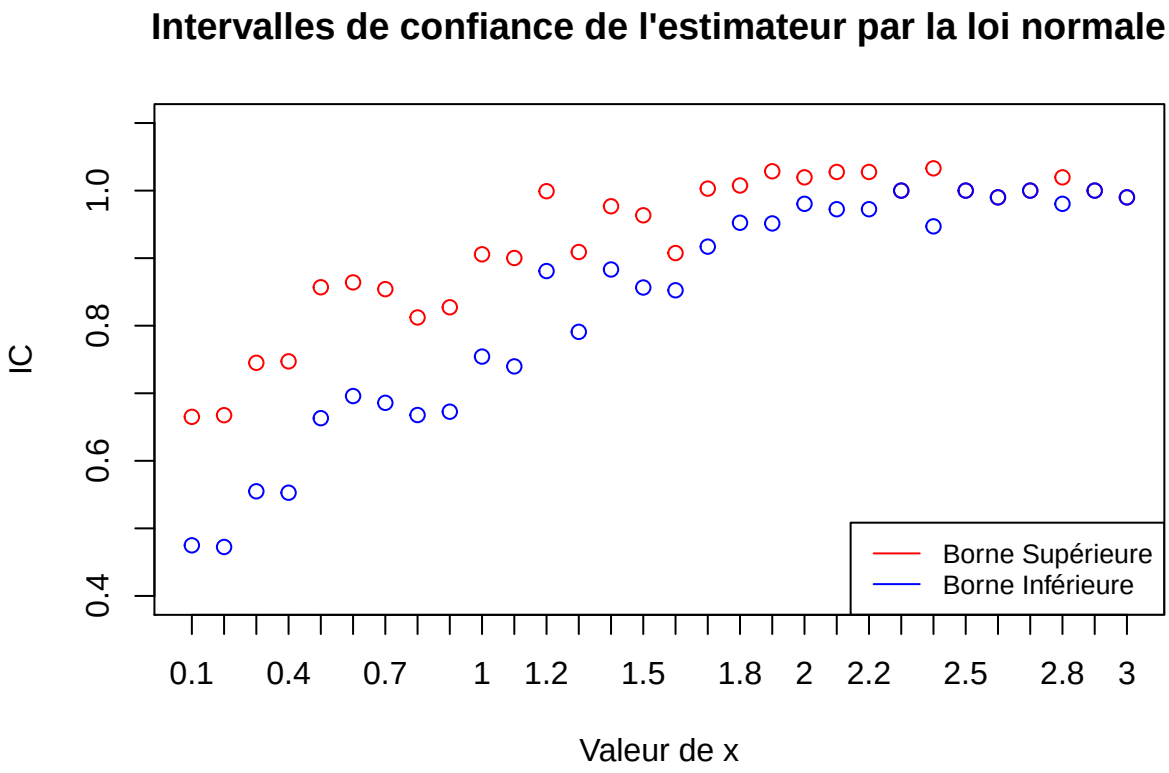
et pour l'intervalle de confiance :

$$[\overline{X}_n - 1.96\frac{\sigma}{\sqrt{n}}; \overline{X}_n + 1.96\frac{\sigma}{\sqrt{n}}]$$

avec $\frac{\alpha}{2}$ le quantile de la normale centrée réduite.

```
n <- 100
#IC
x <- seq(0.1,3,0.1)
Xnbarre <- numeric(length(x))
#loi normale
l = 0
#on fait varier les valeurs de x
for (z in x) {
  l = l + 1

  Xi = numeric()
  for (i in (1:n)){
    t <- rnorm(1,0,1)
    Xi[i] = sum(t < z)/length(t)
  }

  Xnbarre[l] <- mean(Xi)
```

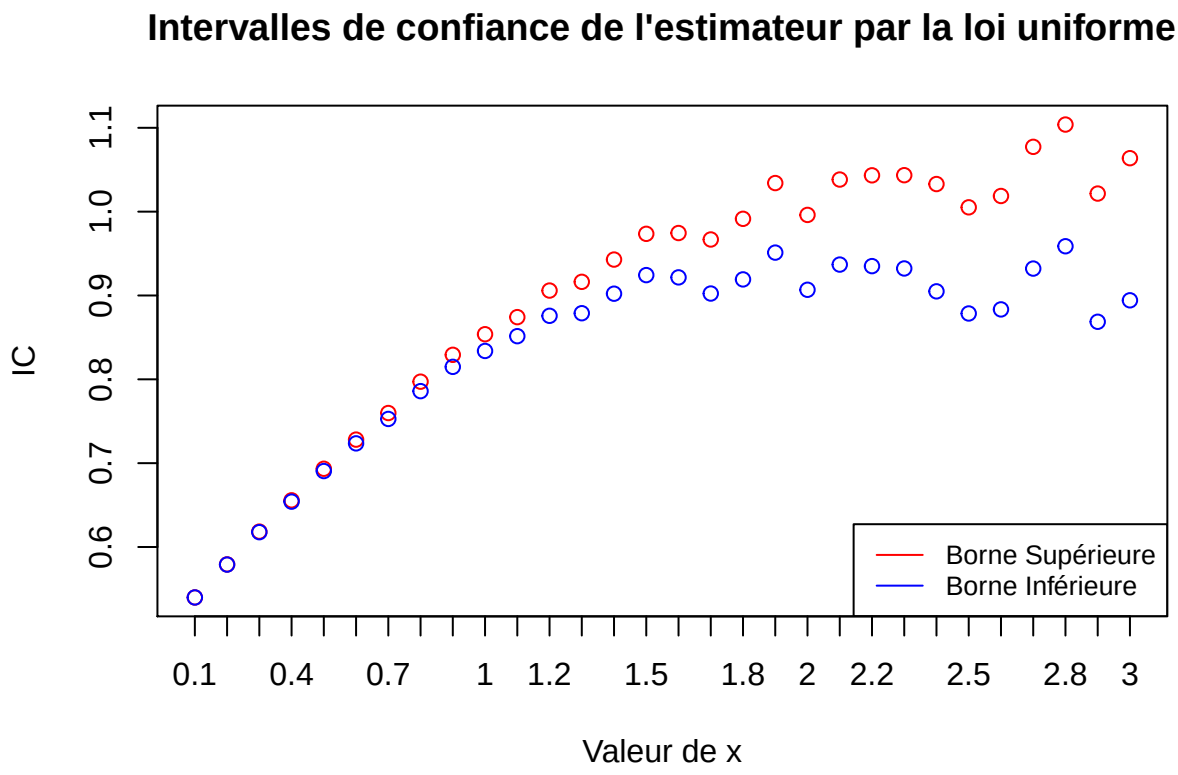
```
}
#normale
ICinf <-  Xnbarre - 1.96*sqrt(sigma_norm)/ sqrt(n)
ICsup <-  Xnbarre + 1.96*sqrt(sigma_norm)/ sqrt(n)

Interval <- cbind(ICinf, ICsup)
#représentation graphique
plot(x,ICsup, type="p",col="red", main="Intervalles de confiance de l'estimateur par la loi normale", yl
lines(x,ICinf, type="p",col="blue")
axis(side=1,at= seq(0.1,3,0.1),labels= seq(0.1,3,0.1))
legend("bottomright", legend=c("Borne Supérieure", "Borne Inférieure"),
       col=c("red", "blue"), lty=1, cex=0.8)
```



Pour la loi normale, nous voyons que les intervales ont tendance à diminuer au fur et à mesure que x augmente

```
##Uniformel=0
#on fait varier les valeurs de x
Xnbarre <- numeric()
l = 0
for (z in x) {
  l = l+1
Xi = numeric()
j=0
for (i in (1:n)){
  u <- runif(1,0,1)
```

```
  j = j +1
Xi[j] = 0.5 + z/sqrt(2*pi)*exp(-((z**2)*(u**2)/2))
}
Xnbarre[l] <- mean(Xi)
}
#uniform
ICinf <-  Xnbarre - 1.96*sqrt(sigma_uni)/ sqrt(n)
ICsup <-  Xnbarre + 1.96*sqrt(sigma_uni)/ sqrt(n)

Interval <- cbind(ICinf, ICsup)
#représentation graphique
plot(x,ICsup, type="p",col="red", main="Intervalles de confiance de l'estimateur par la loi uniforme", x
lines(x,ICinf, type="p",col="blue")
axis(side=1,at= seq(0.1,3,0.1),labels= seq(0.1,3,0.1))
legend("bottomright", legend=c("Borne Supérieure", "Borne Inférieure"),
       col=c("red", "blue"), lty=1, cex=0.8)
```



Plus la taille de x augmente plus l'intervalle de confiance augmente également.

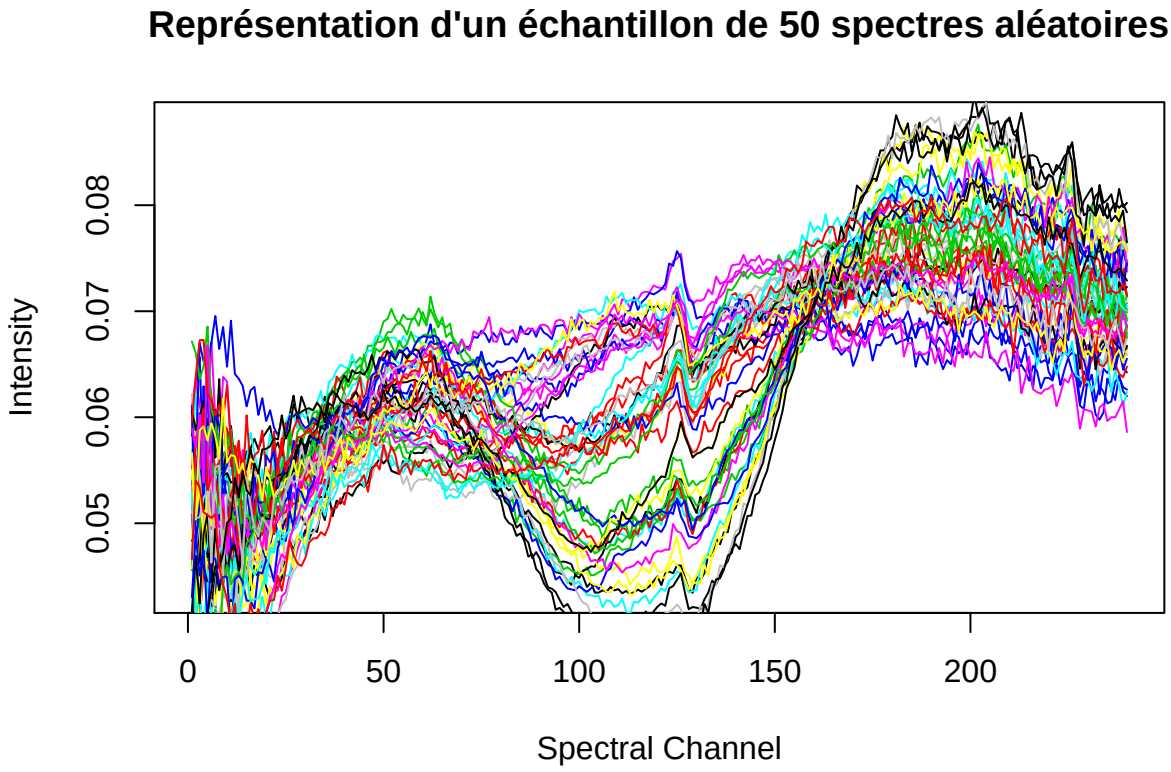
5. Comment modifier la procédure si $x < 0$?

Si le x est inférieur à 0, nous savons que le résultat sera inférieur à 0.5. Nous pouvons utilisé ce que nous avons en calculant l'intégrale entre 0 et x. En effet, puisque la fonction est symétrique autour de 0, cet intégrale est égale à celui entre x et 0 avec $x < 0$. Puis, ce résultat devra être retranché à 0.5 pour connaître l'intégrale entre $-\infty$ et x avec $x < 0$.

Exercice 3 - Bootstrap & ACP

1. Représenter (sur une même figure) un échantillon aléatoire de 50 spectres pour en apprécier leur variabilité, et commenter le résultat obtenu.

```
library(bootstrap)
library(FactoMineR)
set.seed(27)

data<-load("H:/Grenoble/SSD_M2/Statistique Computationnelle/Examen/examen-dataset/spectra.Rdata")

#informations sur le jeu de données
 #546 spectres sur 240 canaux

#taille de l'échantillon
n=50
z <- X[sample(1:nrow(X),n),] #tirage de 50 spectres
plot(z[1,], main="Représentation d'un échantillon de 50 spectres aléatoires", ylab="Intensity", xlab="Sp
for (i in 2:nrow(z)){
lines(z[i,],col=i)
}
```

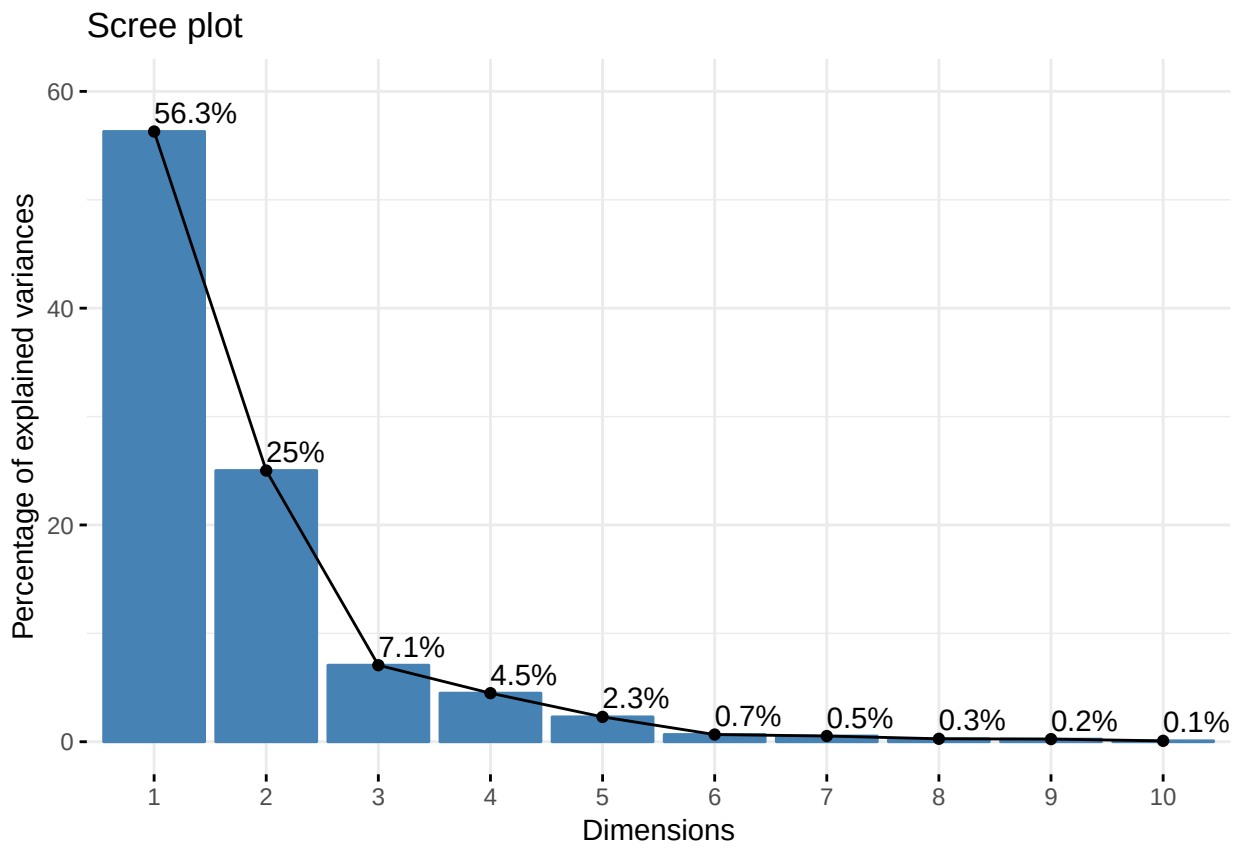


Le graphique ci-dessus nous montre la représentation d'un échantillon de 50 spectres aléatoires. Cela nous permet de voir la variabilité de tous ces spectres.

2. Effectuer une ACP et représenter les proportions de variance expliquées par les 10 premières composantes principales : quelle proportion de la variance totale est expliquée par les deux premières

composantes principales ? Représenter sous la forme de spectres les deux axes principaux (i.e., les opérateurs linéaires permettant de passer de l'espace des canaux aux deux premières composantes principales) : sont-ils cohérents avec vos observations de la question 1 ?

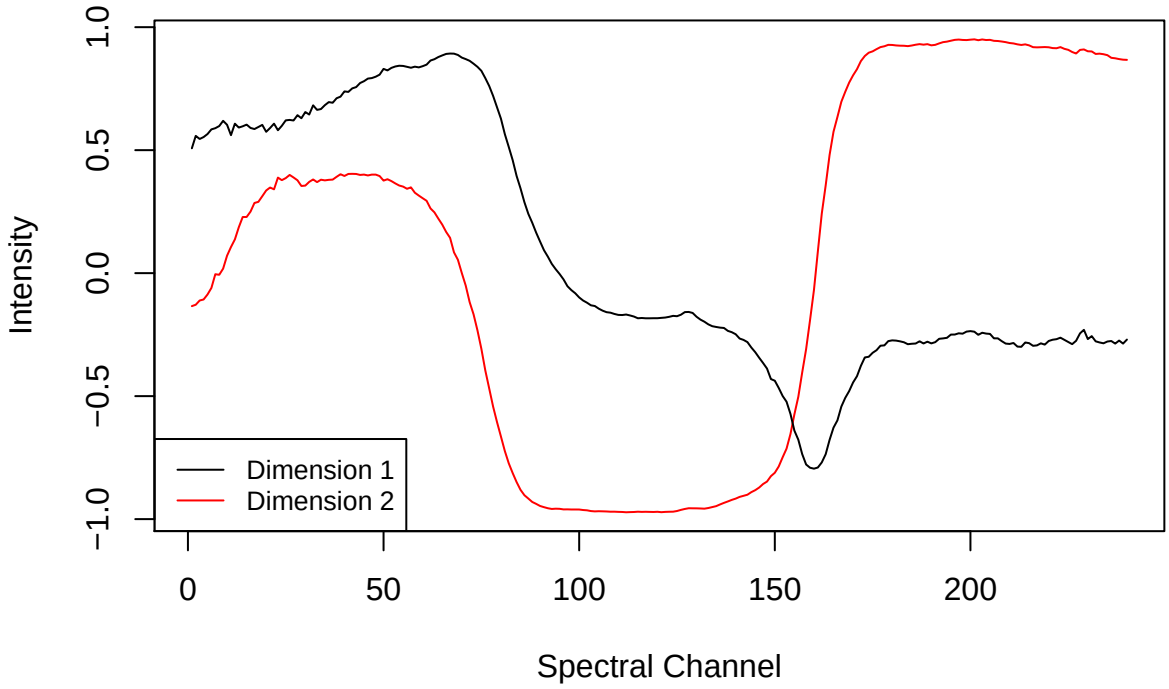
```
#package
library("factoextra")

#mise en place de l'ACP
res.pca <- PCA(X,scale.unit=TRUE,ncp = 10, graph = FALSE)
eig.val <- get_eigenvalue(res.pca) #importance des variables
#eig.val[1:10,2] #pourcentage de variance pour chaque dimension


#Representation graphique
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 60))
```



Avec cet histogramme, nous voyons le pourcentage de variance expliquée pour chacun des 10 premières dimensions. Les deux premiers dimensions expliquent plus de 80% de la variance totale.

```
#Représentation sous forme des spectres des deux axes principaux
var <- get_pca_var(res.pca)
axes_pr<-var$coord[,1:2]
axes<-t(axes_pr)
#representation graphique
plot(axes[1,], col="red", type="l", xlab="Spectral Channel",ylab="Intensity")
lines(axes[2,], type="l")
```

```
legend("bottomleft", legend=c("Dimension 1", "Dimension 2"),
       col=c("black", "red"), lty=1, cex=0.8)
```



Pour cette partie, les deux axes principaux ont été representées sous formes de spectres. Ces résultats sont plutôt adéquation avec nos résultats de la question 1 car ils reprennent l'ensemble de la variabilité. Cependant, il reste toujours une part de la variance, environ 20%, qui n'est pas expliquée par ces deux axes.

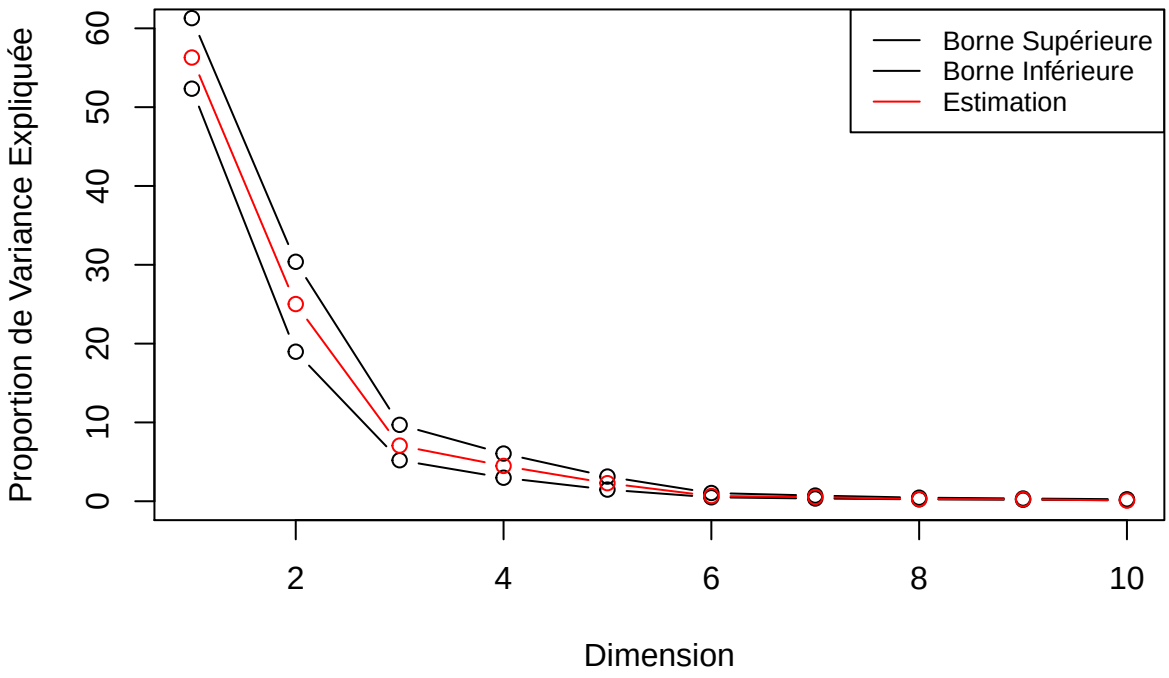
3. Implémenter une procédure bootstrap pour calculer les intervalles de confiance associées aux proportions de variance expliquées par les 10 premières composantes principales, par la méthode de votre choix (e.g., quantile, basique, ...). Proposer une représentation graphique de vos résultats.

```
library(boot)

#creating one replication of the dataset
one.replication <- function(dataset){
n <- nrow(dataset)
index <- sample.int(n,size=50,replace=T)
out.dataset <- dataset[index,]
return(out.dataset)
}

#performing a pca on a replication of the dataset
pca.replication <- function(dataset){
one.dataset <- one.replication(dataset)
pca <- PCA(one.dataset,ncp = 10, graph = FALSE)
#prct_var <- (pca$sd^2/sum(pca$sdev^2))*100
```

```r
prct_var <- as.vector(get_eigenvalue(pca)[1:10,2])
return(prct_var)
}

#bootstraping pca
T<-1000 #remplacer T par 1000
res.boot <- replicate(T,pca.replication(X))

#Intervalle de confiance
Iperc <- matrix(data = NA,nrow=10,ncol=2)
q1 <- numeric()
q2 <- numeric()
for(i in 1:10){
q1[i]=quantile(res.boot[i,],0.025)
q2[i]=quantile(res.boot[i,],0.975)
Iperc[i,]=c(q1[i],q2[i])
}

#resultat
resultat <- cbind(eig.val[1:10,2], Iperc)
colnames(resultat) <- c("Estimation","Borne Inférieure" ,"Borne Supérieure")
resultat
```

```
##          Estimation Borne Inférieure Borne Supérieure
## Dim.1  56.29879485       52.3430590       61.2977299
## Dim.2  25.01858258       18.9716587       30.3806518
## Dim.3   7.06122797        5.2019880        9.6932481
## Dim.4   4.48333600        2.9818046        6.0518371
## Dim.5   2.29345349        1.4758017        3.1315827
## Dim.6   0.66560661        0.4995949        1.0451438
## Dim.7   0.52860963        0.3480878        0.7187057
## Dim.8   0.27414543        0.2433401        0.4516415
## Dim.9   0.24473150        0.1851781        0.3467701
## Dim.10  0.08359934        0.1447075        0.2459593
```

```r
#Représentation graphique
plot(Iperc[,1], type='b', xlab="Dimension", ylab="Proportion de Variance Expliquée", ylim=c(0,60))
lines(Iperc[,2], type='b')
lines(eig.val[1:10,2],col="red", type='b')
legend("topright", legend=c("Borne Supérieure", "Borne Inférieure", "Estimation"),
       col=c("black","black", "red"), lty=1, cex=0.8)
```

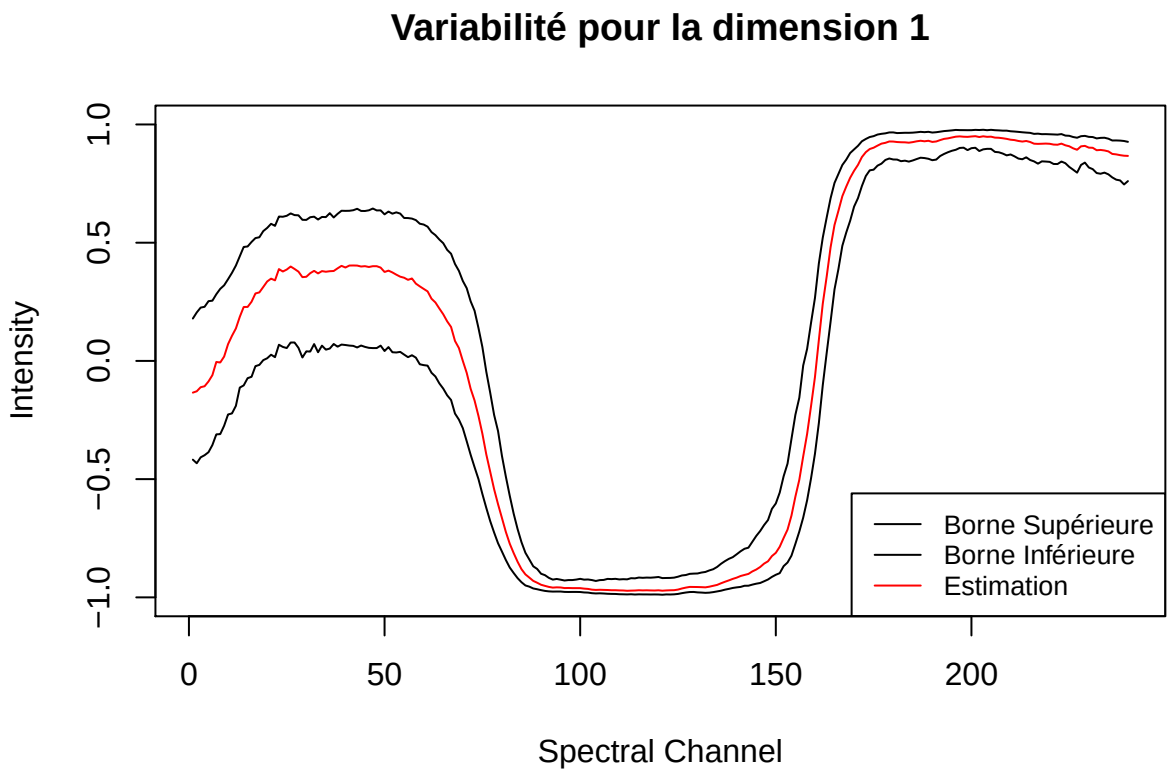


Avec le graphique et les résultats dans le tableau nous pouvons les intervalles de confiance associés aux proportions de variances expliquées par les 10 premières valeurs.

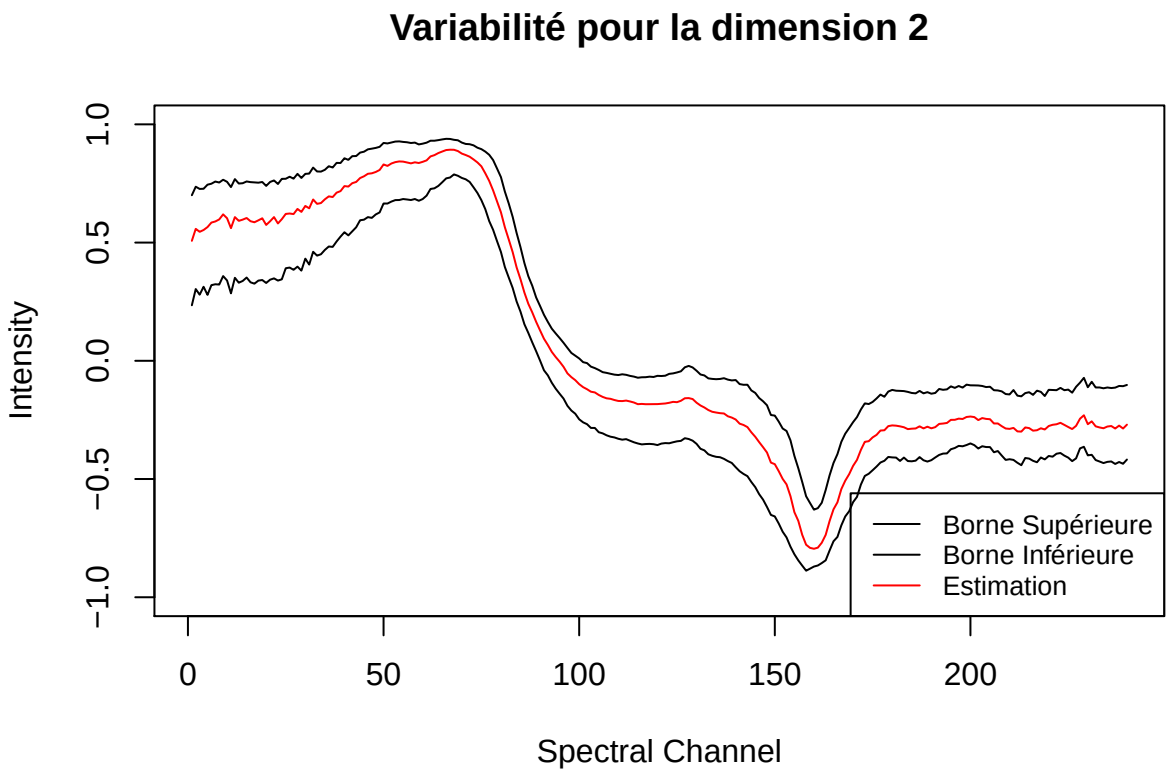
4. Enfin, modifier votre procédure pour pouvoir représenter la variabilité induite par le bootstrap sur les axes principaux de la question 2 et commenter les résultats obtenus.

```
#nouvelle fonction pour récuper informations necessaire
pca.replication2 <- function(dataset){
one.dataset <- one.replication(dataset)
pca <- PCA(one.dataset,ncp = 10, graph = FALSE)
var <- get_pca_var(pca)
axes_pr<-t(var$coord[,1:2])
return(axes_pr)
}

T<-1000 # remplacer T par 1000
variabilite_dim.boot <- replicate(T,pca.replication2(X))
#presence d'un cube
dim1 <- variabilite_dim.boot[1,,]
dim2 <- variabilite_dim.boot[2,,]

#Intervalle de confiance
Iperc1 <- matrix(data = NA,nrow=240,ncol=4)
dim1_q1 <- numeric()
dim1_q2 <- numeric()
dim2_q1 <- numeric()
```

```r
dim2_q2 <- numeric()
for(i in 1:nrow(dim1)){
dim1_q1[i]=quantile(dim1[i,],0.025)
dim1_q2[i]=quantile(dim1[i,],0.975)
dim2_q1[i]=quantile(dim2[i,],0.025)
dim2_q2[i]=quantile(dim2[i,],0.975)
Iperc1[i,]=c(dim1_q1[i],dim1_q2[i],dim2_q1[i],dim2_q2[i])
}

#Représentation graphique
#dim1
plot(Iperc1[,1],xlab="Spectral Channel",ylab="Intensity", type="l",ylim=c(-1,1), main="Variabilité pour
lines(Iperc1[,2], type="l")
lines(axes[1,], col="red", type="l")
legend("bottomright", legend=c("Borne Supérieure", "Borne Inférieure", "Estimation"),
       col=c("black","black", "red"), lty=1, cex=0.8)
```




```r
#dim2
plot(Iperc1[,3], type="l",xlab="Spectral Channel",ylab="Intensity", ylim=c(-1,1),main="Variabilité pour
lines(Iperc1[,4], type="l")
lines(axes[2,], type="l",col="red")
legend("bottomright", legend=c("Borne Supérieure", "Borne Inférieure", "Estimation"),
       col=c("black","black", "red"), lty=1, cex=0.8)
```



Les deux graphiques ci-dessus, nous donne l'intervalle de confiance pour la dimension 1 et la dimension 2 de la variabilité induite par le bootstrap. Nous voyons que ces résultats correspondent aux résultats obtenues aux questions précedentes.

Exercice 4 : tests par permutation

```
#Importation du jeu de données
load("/ext/u123/agblodok/Desktop/Examen/examen-dataset/permutation-dataset.Rdata")

#ajout une variable pour connaitre l'apparteance aux groupes
couleur <- c(rep(1,100),rep(0,100))
data <- cbind(X,couleur)
head(data)
```

```
##                              couleur
## [1,] -0.2085937  0.29245270       1
## [2,] -1.0542989 -0.98181481       1
## [3,]  1.3490941  0.98081296       1
## [4,] -1.0276848 -0.44742986       1
## [5,] -0.5852872 -0.43621016       1
## [6,] -0.7368039 -0.06049774       1
```

1. Que mesure cette statistique ?

$$T = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_{n(x_i)} = y_i)$$

Cette statistique nous donne la moyenne de points dont leur plus proche voisin appartient au même ensemble de points, c'est à dire à la même couleur, que le leur.

2. Appppliquer une procédure par permutation pour B = 1000 tirages et évaluer la p-valeur obtenue. La différence entre les deux nuages de points est-elle significative ?

Pour réaliser cette permutation, nous devons avoir la même variance sous H_0 dans ces groupes.

```
#calcul de la distance euclidienne entre tous les points
mat_dist <- as.matrix(dist(X[,1:2],method="euclidean"))
#NA dans la diagonale
diag(mat_dist) <- "NA"

ppp <- numeric()#ppp pour points le plus proche
t0 <- numeric()
for (i in 1:200){
 ppp[i] <- order(mat_dist[,i])[1]
}
#nombre de plus proche voisins de la meme couleur
for (i in 1:200){
  t0[i] <- data[i,3] == data[ppp[i],3]
}
t0 <- mean(t0)
```

Le nuage de point initial donne une p-valeur de 0.655. Nous avons donc 65.5% de point dont leur plus proche voisin est de la même couleur qu'eux.

```
#seed
set.seed(1)
B = 1000 # nombre de permutations
n = 100 #valeur de n
t.perm = numeric(B)
ppp <- numeric()#ppp pour points le plus proche
res <- numeric()
```

```
for(b in 1:B){
ind = sample(nrow(X),n)
noir <- cbind(X[ind,],rep(1,n))
rouge<- cbind(X[-ind,],rep(0,n))
#recréation de la matrice pour calcul distance
data1 <- rbind(noir,rouge)

#calcul de la distance euclidienne entre tous les points
mat_dist <- as.matrix(dist(data1[,1:2],method="euclidean"))
#NA dans la diagonale
diag(mat_dist) <- "NA"


for (i in 1:200){
 ppp[i] <- order(mat_dist[,i])[1]
 res[i] <- data1[i,3] == data1[ppp[i],3]
}
t.perm[b] = mean(res)
}


#calcul de la pvaleur
pval = mean(t.perm >= t0)
pval
```

```
## [1] 0
```

Nous obtenons une p-valeur de 0, elle est inférieure à 0.05. Donc nous rejettons l'hypothèse nulle, alors nous pouvons conclure que les deux nuages de points sont significativement différents.

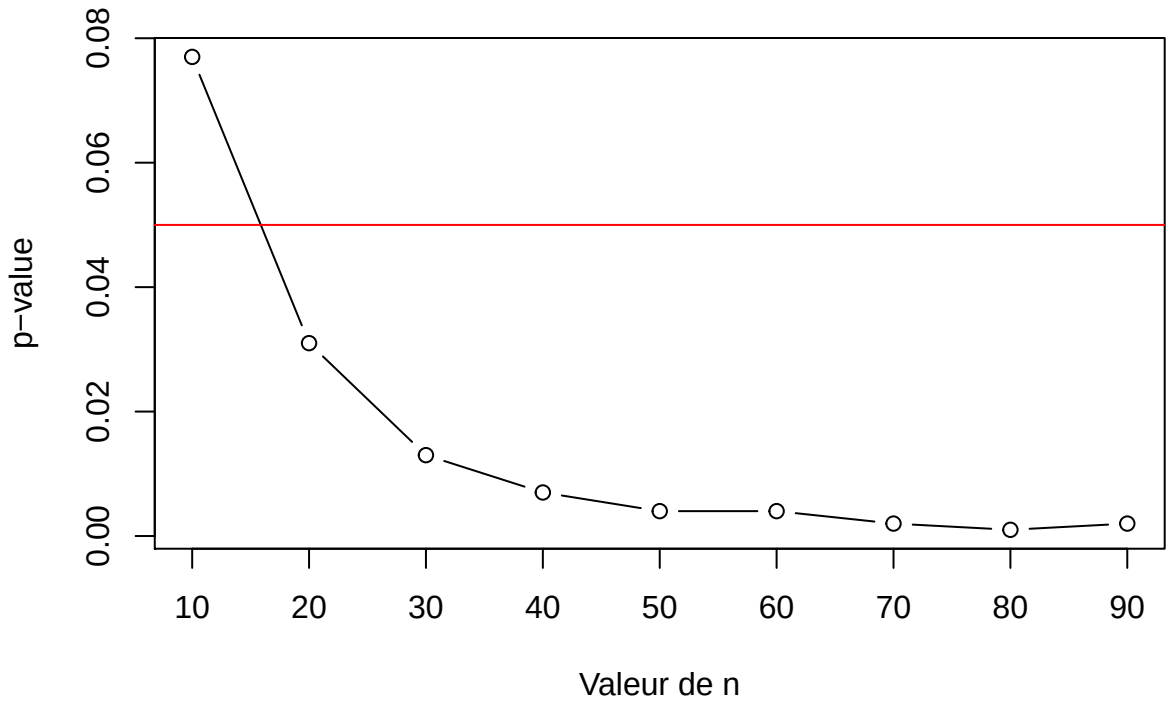
3. Reproduire cette analyse en tirant aléatoirement un ensemble de n = 10, 20, ..., 90 points parmi chaque population. Comment la p-valeur évolue t'elle ? Représenter les résultats sous la forme d'un graphique.

```
#seed
set.seed(1)
B = 1000 # nombre de permutations
n = seq(10,90,10) #valeur de n
t.perm = numeric(B)
ppp <- numeric()#ppp pour points le plus proche
res <- numeric()
pval<-numeric(length(n))

for (k in 1:length(n)){
  for(b in 1:B){
    ind = sample(nrow(X),2*n[k])

    noir <- cbind(X[ind[1:n[k]],],rep(1,n[k]))
    rouge<- cbind(X[ind[(n[k]+1):(2*n[k])],],rep(0,n[k]))
    #recréation de la matrice pour calcul distance
    data1 <- rbind(noir,rouge)

    #calcul de la distance euclidienne entre tous les points
    mat_dist <- as.matrix(dist(data1[,1:2],method="euclidean"))
    #NA dans la diagonale
    diag(mat_dist) <-"NA"
```

```
    for (i in 1:(2*n[k])){
      ppp[i] <- order(mat_dist[,i])[1]
       res[i] <- data1[i,3] == data1[ppp[i],3]
    }
    t.perm[b] = mean(res)
  }
  #calcul de la pvaleur
pval[k] = mean(t.perm >= t0)
}
```

Nous allons maintenant représenter les p-valeurs pour les différentes valeurs de n.

```
plot(n,pval, type="b", col="black",xlab="Valeur de n", xlim=c(10,90), ylab="p-value", xaxt='n')
axis(side=1,at=n,labels=n)
abline(h=0.05,col="red")
```



Dans cette partie, nous faisons varier la taille de chaque population. Nous voyons que les p-values ont tendance à diminuer au fur et à mesure que la taille de n augmente. La ligne rouge est tracée pour represénter la limite pour considérer les deux populations significativement différentes ou non.

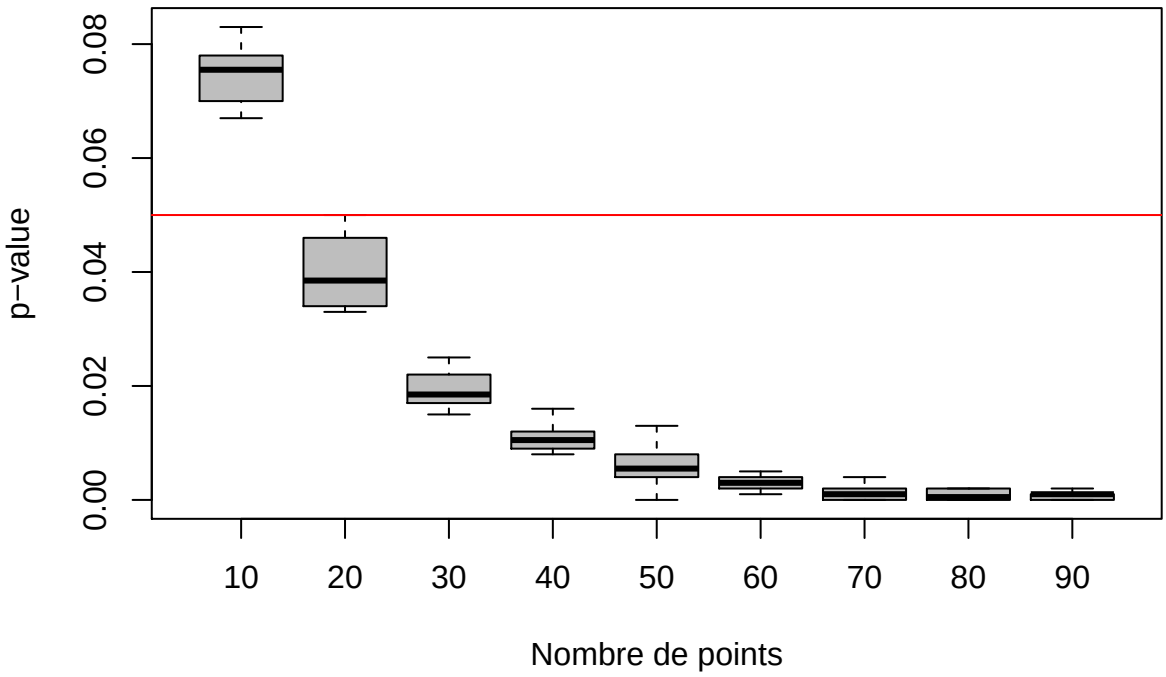
4. Enfin, reproduire cette seconde analyse 10 fois et représenter la variabilité dans les pvaleurs obtenues en fonction du nombre de points considérés. A partir de quelle taille d'échantillon est-on capable de détecter une différence significative en considérant que la médiane des p-valeurs obtenues sur les 10 répétitions doit être (et rester) inferieure à 0.05 ?

```
#reproduire l'analyse 10 fois
set.seed(1)
B = 1000 # nombre de permutations
n = seq(10,90,10) #valeur de n
t.perm = numeric(B)
ppp <- numeric()#ppp pour points le plus proche
res <- numeric()
pval<-matrix(data=NA, nrow=10,ncol=9)
```

```
for (k in 1:length(n)){
  for(l in 1:10){
    for(b in 1:B){
      ind = sample(nrow(X),2*n[k])

      noir <- cbind(X[ind[1:n[k]],],rep(1,n[k]))
      rouge<- cbind(X[ind[(n[k]+1):(2*n[k])],],rep(0,n[k]))
      #recréation de la matrice pour calcul distance
      data1 <- rbind(noir,rouge)

      #calcul de la distance euclidienne entre tous les points
      mat_dist <- as.matrix(dist(data1[,1:2],method="euclidean"))
      #NA dans la diagonale
      diag(mat_dist) <-"NA"


      for (i in 1:(2*n[k])){
        ppp[i] <- order(mat_dist[,i])[1]
        res[i] <- data1[i,3] == data1[ppp[i],3]
      }

      t.perm[b] = mean(res)
    }
    #calcul de la pvaleur
  pval[l,k] = mean(t.perm >= t0)
  }
}
```

Représentation des résultats

```
boxplot(pval, col="grey", main="Représentation de la variabilité des p-values", ylab="p-value", xlab="N
axis(1,at=c(1:9), labels=c(10,20,30,40,50,60,70,80,90))
abline(h=0.05, col="red")
```



Nous avons refait la même analyse mais en la répétant 10 fois. Cela nous permet d'avoir une variabilité de la p-value selon le nombre de points dans chaque population. Nous retrouvons le fait que les p-values diminuent plus on augmente la taille de n. De plus la variabilité diminue lorsque la taille de n augmente.