Examen

Master parcours SSD - UE Statistique Computationnelle

RAMDÉ Ismaïl - N'DOYE EL Hadrami

2021-11-15

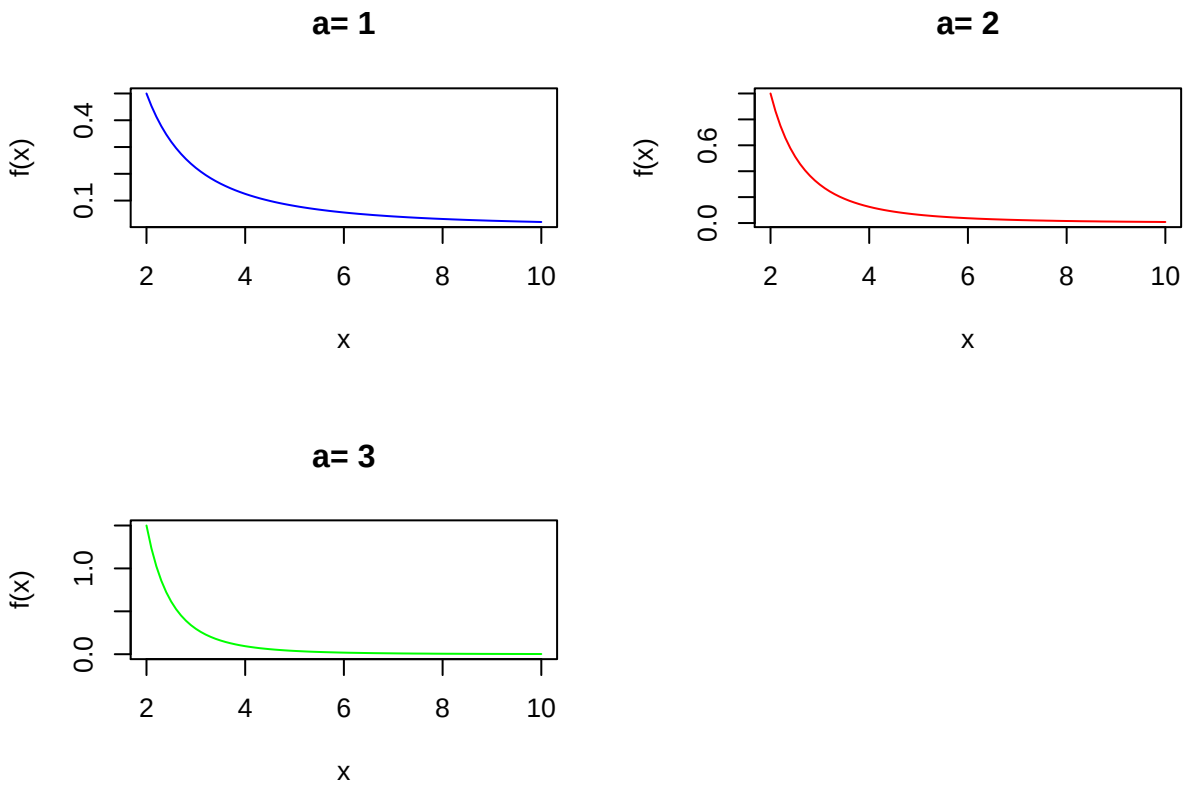
Exercice 1 - simulation par inversion

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

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

1. Représentons cette densité pour b = 2 et $a \in \{1, 2, 3\}$**.**

```
par(mfrow=c(2,2))
f_densite = function(x,a,b){
  f = c()
  for(i in x){
    if(i >= b)
      f = c(f,((a * (b^a))/(i^(a+1))))
    else{
      f = c(f,0)
    }
  }
  return(f)
}
x = seq(2,10,0.1)
a = c(1,2,3)
b = 2
cols = c("blue","red","green")
for ( i in 1 : length(a)){
plot(x,f_densite(x,a[i],b),col=cols[i],ylab = "f(x)",type = "l"
     ,main=c(paste("a=",a[i])))
}
```



Les figures ci-dessus représentent la fonction de densité $f(x)$ pour chaque valeur de $a \in \{1, 2, 3\}$. Les courbes de densité $f(x)$ sont croissantes et convergent vers 0, on constate que la courbe de densité pour $a = 3$ converge plus vite que les deux autres courbes.

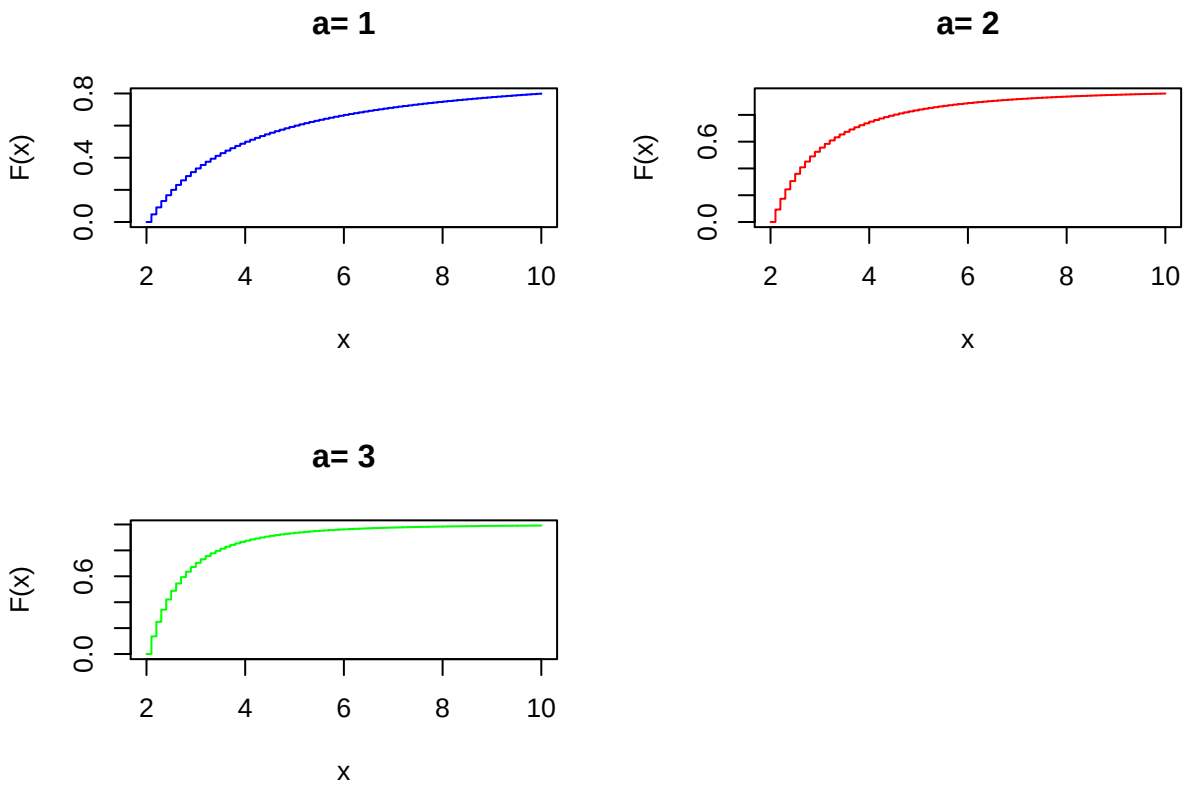
2. Implémentons une procédure d'inversion pour simuler une variable aléatoire selon cette densité.

Calcul de la fonction de répartition de la densité f:

$F(x) = \int_b^x f(t)\mathrm{d}t = \int_b^x \frac{ab^a}{t^{a+1}}\mathrm{d}t = ab^a \int_b^x \frac{1}{t^{a+1}}\mathrm{d}t = ab^a \left[\frac{-1}{at^a}\right]_b^x = ab^a(\frac{-1}{ax^a} + \frac{1}{ab^a}) = \frac{-b^a}{x^a} + 1 = 1 - (\frac{b}{x})^a$ d'ou :

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

```
par(mfrow=c(2,2))
n = length(x)
Fx = matrix(0,nrow=3,ncol=n)
for (i in 1:length(a)){
  Fx[i,] = 1 - (b / x)^a[i]
  plot(x,Fx[i,],type="s",col=cols[i],main = paste("a=",a[i]),ylab = "F(x)")
}
```



Les courbes ci-dessus donnent la fonction de répartition d'une variable aléatoire associée à une fonction de densité $f(x)$.

Méthode d'inversion

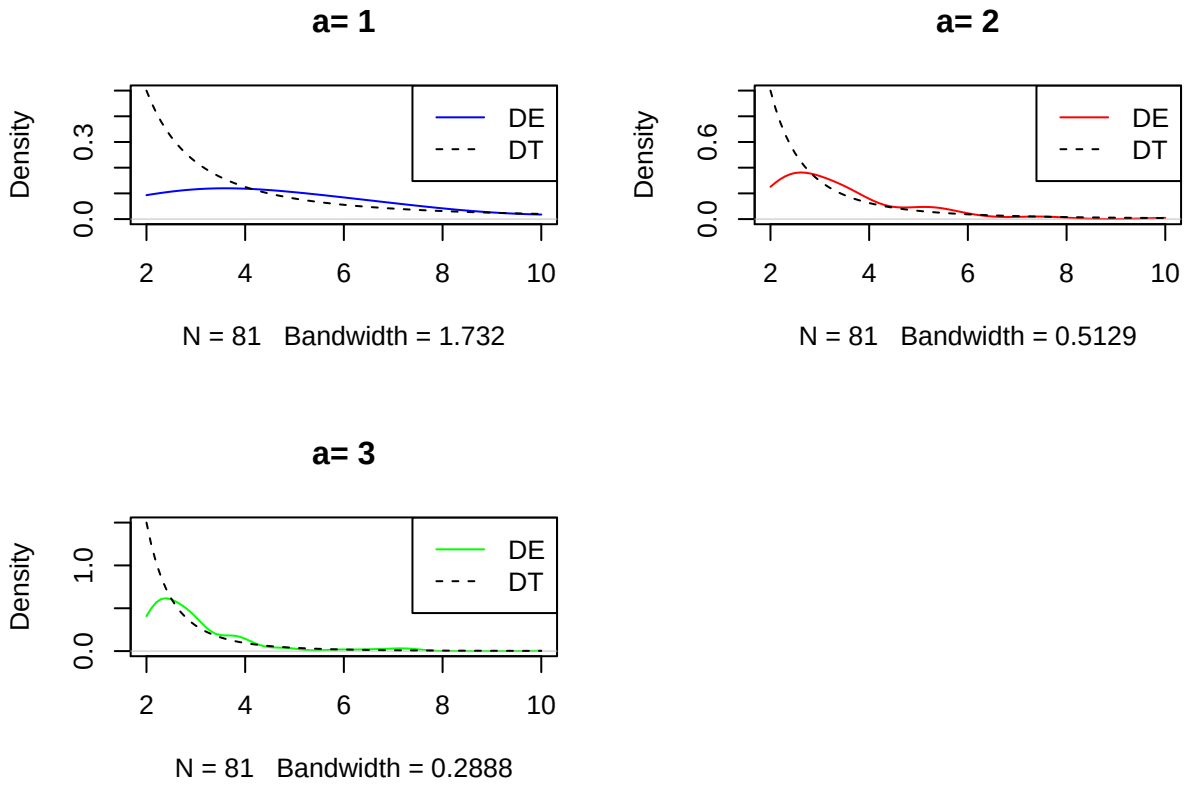
Pour simuler par la méthode d'inversion on doit d'abord calculer la réciproque de la fonction de répartition de la variable d'intérêt.
On pose $u = F(x) = 1 - (\frac{b}{x})^a$, d'ou :
$x = \frac{b}{(1-u)^{1/a}}$, avec $u \sim U(0,1)$.

```
procedure_inversion = function(ui,a,b=2){
  # u = F(x)
  # a = [1,2,3]
    xi = (b / ((1-ui)^(1 / a)))
    return(xi)
}
```

3. Simulons un échantillon et comparons graphiquement la densité obtenue empiriquement à la densité théorique.

```
par(mfrow=c(2,2))
# Calcul des max pour gerer les graphiques
max_f = c()
for (k in a){
  max_f = c(max_f,max(f_densite(x,k,b)))
}
a = c(1,2,3)
ui = runif(n)
for (i in 1 :length(a)){
```

```
  xi =  procedure_inversion(ui,a[i])
  plot(density(xi,n=length(x),from = 2,to=10),col=cols[i],
         lty=1,main = paste("a=",a[i]),ylim=c(0,max_f[i]))
  lines(x,f_densite(x,a[i],b),col="black",lty=2,ylab = "f(x)",type = "l")
  legend("topright",legend = c("DE","DT")
         ,col = c(cols[i],"black"),lty = c(1,2))
}
```

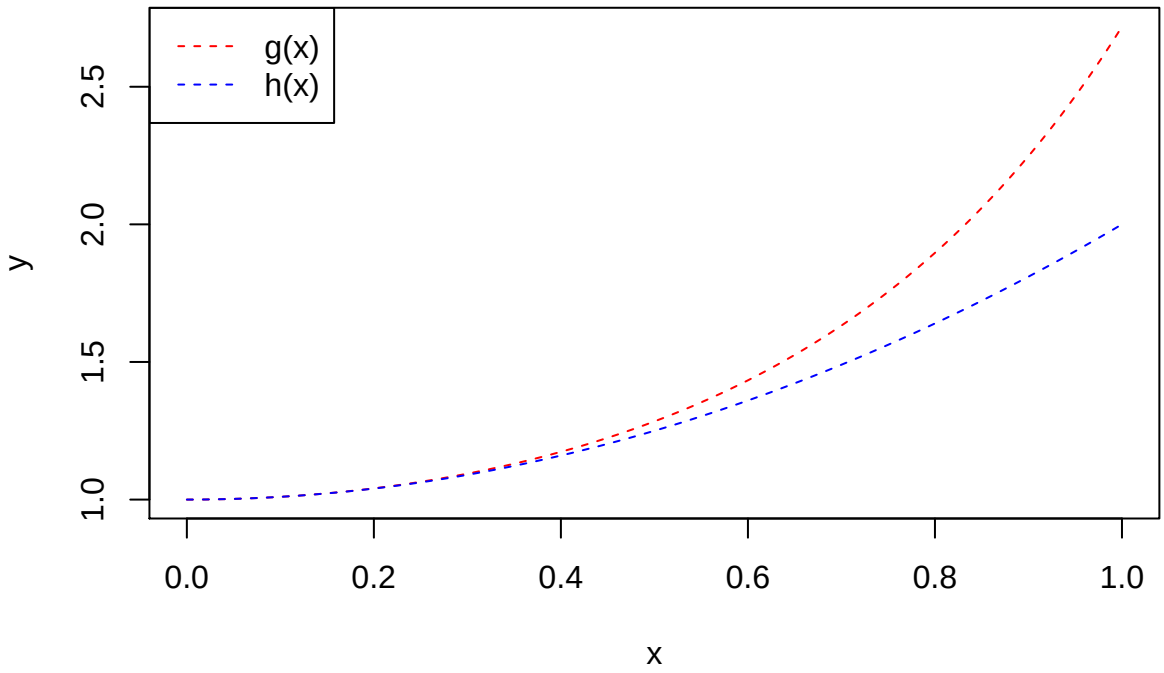


Les différentes courbes affichent les densités empiriques (DE) et théoriques (DT) pour chaque valeur de a. On constate que la densité théorique converge vers la densité empirique à partir d'un certain point x.

Exercice 2 - réduction de variance par variable de contrôle

1. Représentons la fonction à intégrer et la fonction de contrôle pour $x \in [0,1]$

```
g = function(x){
  return (exp(x^2))
}
x = seq(0,1,0.01)
gx = g(x)
h = function(x){
  return(1 + x^2)
}
hx = h(x)
plot(x,gx,type="l",col="red",lty=2,ylab = "y")
lines(x,hx,col="blue",lty=2)
legend("topleft",legend = c("g(x)","h(x)"),col = c("red","blue"),lty=2)
```



La figure ci-dessus représente la courbe de la fonction à intégrer et celle de la fonction de contrôle pour chaque $x \in [0, 1]$. Elles sont toutes les deux croissantes.

2. Fournissons une estimation de l'intégrale et l'intervalle de confiance associé par l'approche Monte Carlo standard, en simulant un échantillon de taille $n = 1000$.

```
n = 1000
X = runif(n)
for(i in 1:n){
  integral_gx = mean(g(X))
}
integral_gx
```

```
## [1] 1.468608
```

```
integrate(g,0,1)
```

```
## 1.462652 with absolute error < 1.6e-14
```

```
cat("L'estimation de l'integral de la fonction g est : ",integral_gx,"\n")
```

```
## L'estimation de l'integral de la fonction g est :  1.468608
```

```
# intervalle de confiance
Sn = mean(g(X))
Vn = (n / (n-1)) * mean((g(X)-Sn)^2)
alpha = 0.05
born_inf = Sn - (qnorm(1-(alpha/2))* sqrt(Vn/n))
born_sup = Sn + (qnorm(1-(alpha/2))* sqrt(Vn/n))
cat("La borne inferieur de l'intervalle de confiance est : ", born_inf,"\n")
```

```
## La borne inferieur de l'intervalle de confiance est :  1.439931
```

```
cat("La borne superieur de l'intervalle de confiance est", born_sup,"\n")
```

```
## La borne superieur de l'intervalle de confiance est 1.497285
```

```r
cat("L'intervalle de confiance est : Ic=[",born_inf,",",born_sup,"]")
```

```
## L'intervalle de confiance est : Ic=[ 1.439931 , 1.497285 ]
```

La valeur de l'estimation de l'intégral par la méthode de Monte-carlo est sur la borne de l'intervalle de confiance (Ic) ce qui montre l'efficacité de la méthode de Monte-carlo sur l'estimation de la valeur de l'intégral.

3. Faisons de même en utilisant la fonction h(x) comme variable de contrôle.

```r
integral_gx = mean(h(X))
cat("L'estimation de l'integral de la fonction g est : ",integral_gx)
```

```
## L'estimation de l'integral de la fonction g est :  1.339825
```

```r
Snh = mean(h(X))
Vnh = (n / (n-1)) * mean((h(X)-Snh)^2)
alpha = 0.05
born_inf = Snh - (qnorm(1-(alpha/2))* sqrt(Vnh/n))
born_sup = Snh + (qnorm(1-(alpha/2))* sqrt(Vnh/n))
cat("La borne inferieur de l'intervalle de de confiance est : " , born_inf,"\n")
```

```
## La borne inferieur de l'intervalle de de confiance est :  1.321792
```

```r
cat("La borne superieur de l'intervalle de de confiance est :", born_sup)
```

```
## La borne superieur de l'intervalle de de confiance est : 1.357859
```

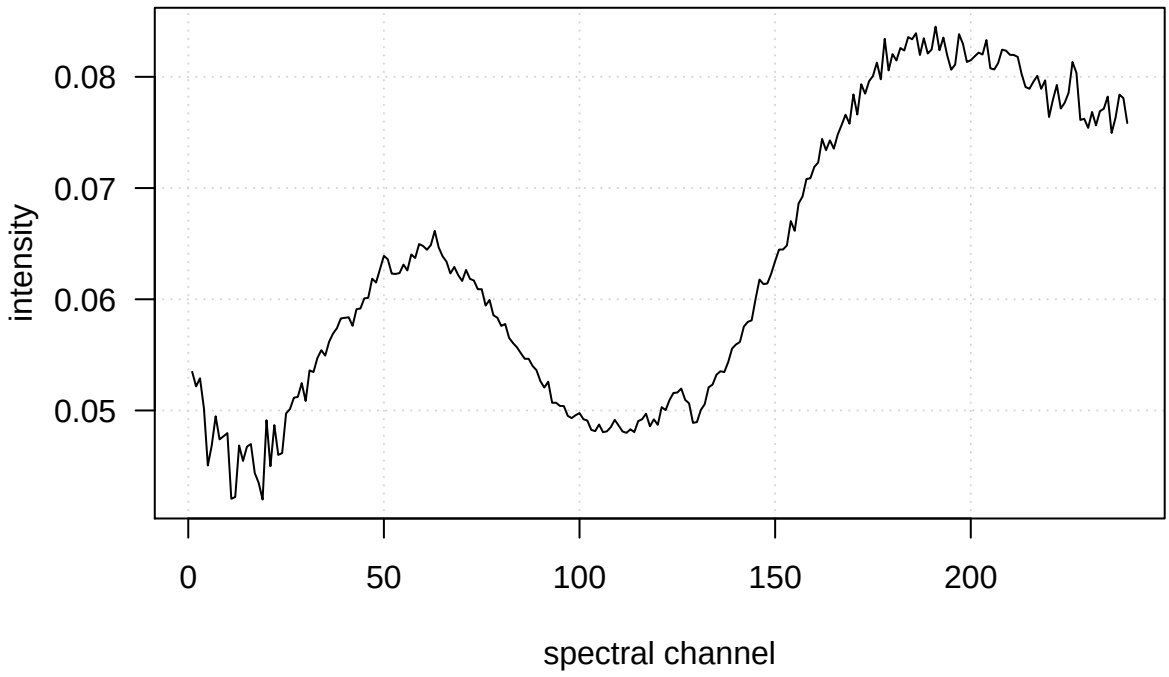
$$var(g(X) - h(X)) = var(g(X)) + var(h(X)) - 2cov(g(X), h(X))$$

```r
vargxhx = Vn+Vnh-2*cov(g(X),h(X))
cat("La reduction de la variance est : ",Vn / vargxhx)
```

```
## La reduction de la variance est :  6.750212
```

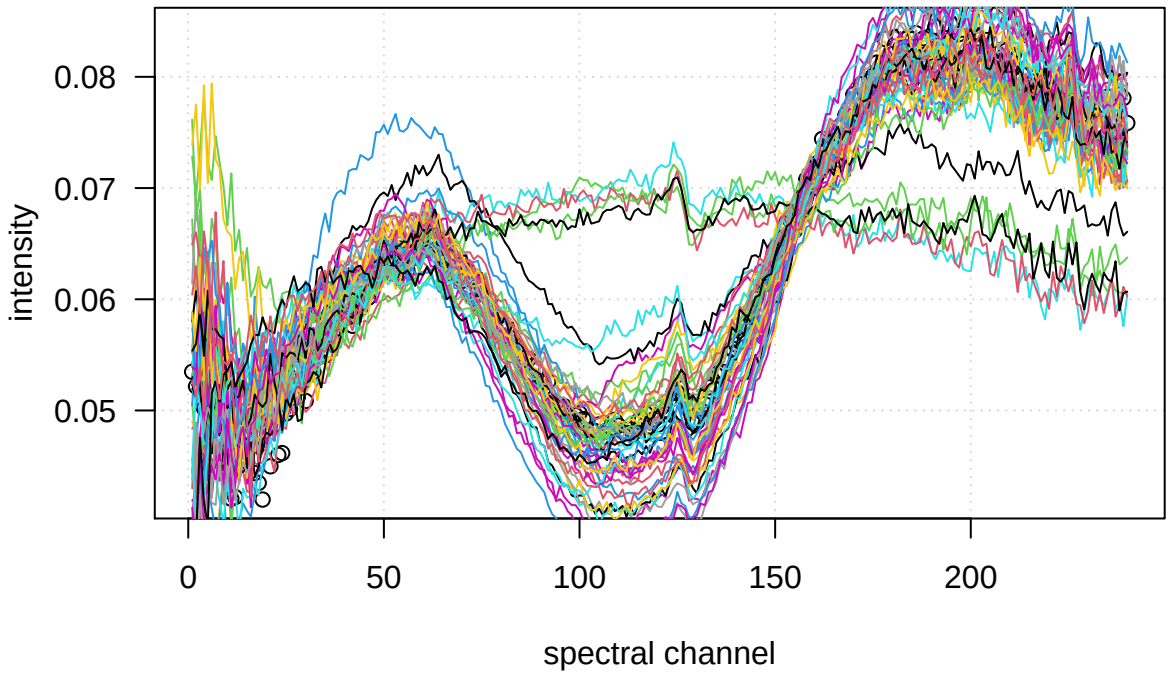
3. Exercice 3 : bootstrap & ACP

```r
# load data
load("spectra.Rdata")
plot(1:240,X[1,],type = "l",xlab = "spectral channel",ylab="intensity",
     las=1,panel.first = grid())
```



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

```
ind_alea = sample(1:240,50)
X_50_spec = X[ind_alea,]
cols = 1:50
spec_channel = 1:240
plot(spec_channel,X_50_spec[1,],col=cols[1],panel.first = grid(),xlab = "spectral channel",
     ylab="intensity",las=1)
for(i in 2:dim(X_50_spec)[1]){
  lines(spec_channel,X_50_spec[i,],col=cols[i])
}
```



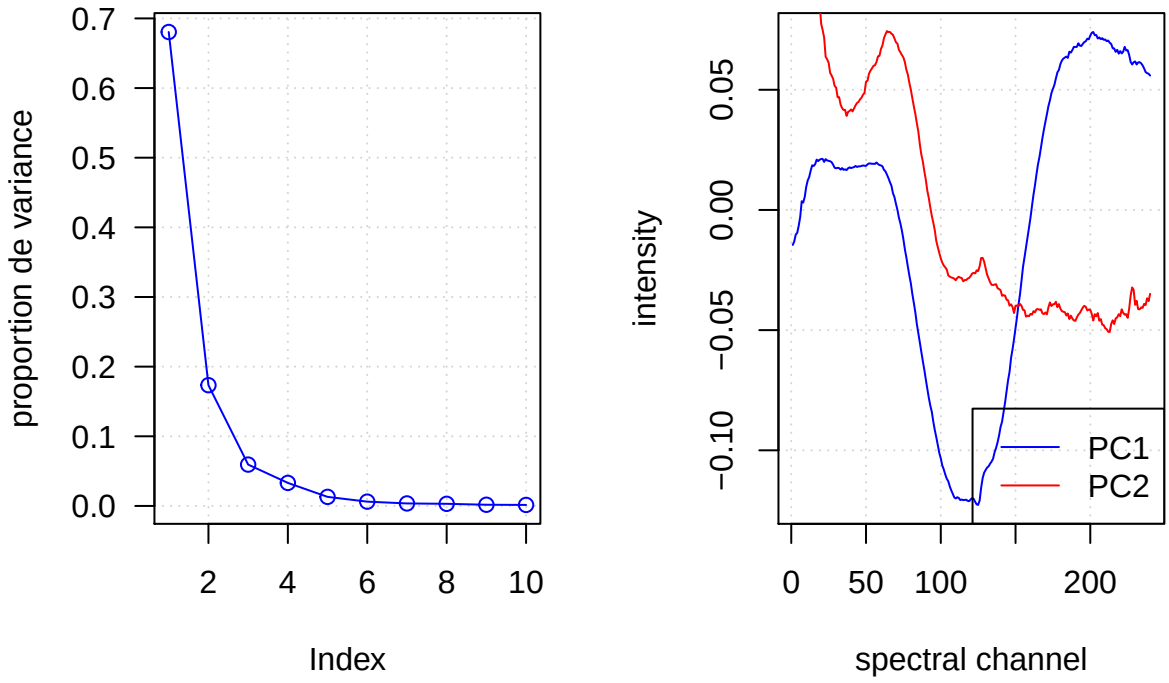
Les courbes ci-dessus représentent la variabilité de l'intensité des 50 spectres choisis de façon aléatoire. Ces courbes permettent de classifier l'intensité des spectres sur un plan (X,Y), tout en conservant le maximum d'informations. A partir de l'ensemble de ces spectres se dégagent deux principales tendances.

2. Effectuons une ACP et représentons 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 ? Représentons 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 ?

```
par(mfrow=c(1,2))
pca = prcomp(X)
s = summary(pca)
# proportion de variance expliquées
pv = s$importance[2,][1:10]
plot(pv,ylab = "proportion de variance",col="blue",panel.first = grid(),las=1)
lines(pv,col="blue")
cat("La proportion de la variance totale expliquée par les deux premières composantes
      principales est :",sum(pv[1:2]),"\n")
```

```
## La proportion de la variance totale expliquée par les deux premières composantes
##       principales est : 0.85381
```

```
plot(1:240,pca$rotation[,"PC1"],type="l",col="blue",lty=1,xlab="spectral channel",
     ylab="intensity",panel.first = grid())
lines(1:240,pca$rotation[,"PC2"],type="l",col="red",lty=1)
legend("bottomright",legend=c("PC1","PC2"),col = c("blue","red"),lty=1)
```

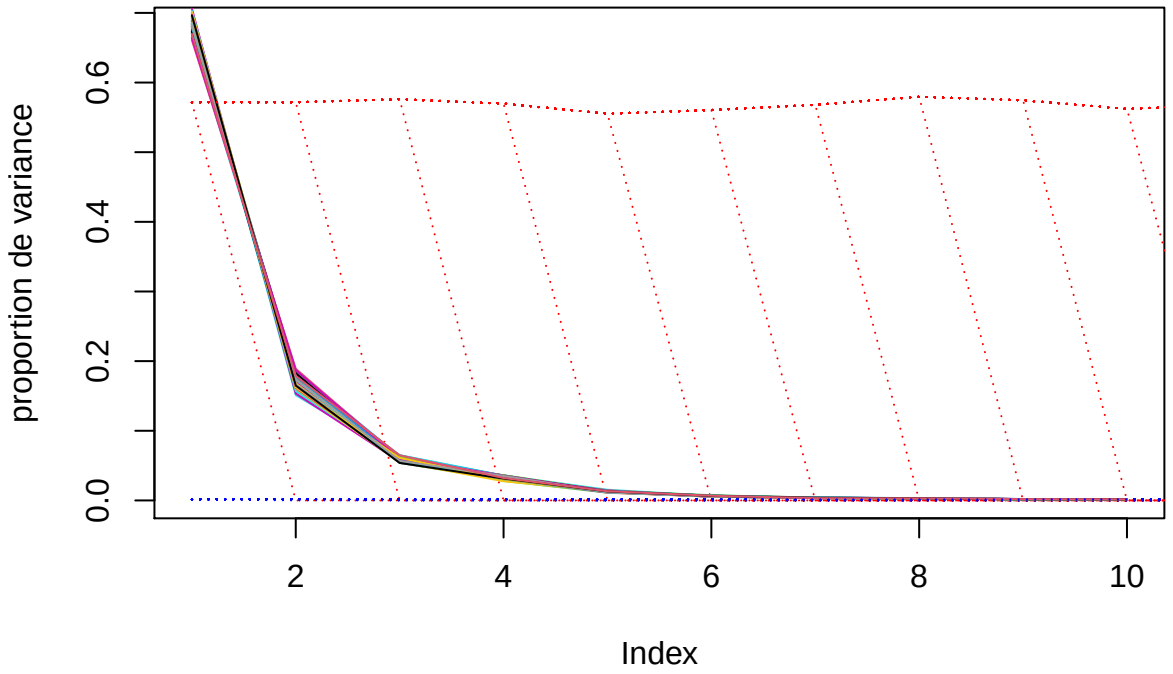


La figure 1 représente les proportions des variances expliquées par les dix premières composantes principales. On constate que la différence de proportion de variance entre les deux premières composantes est très élevée puis elle diminue largement au delà des deux premières composantes avec une variabilité presque constante et elle converge vers 0 à partir de la 7ème composantes.
Au travers de la figure 2, on constate que la composante 1 et la composante 2 suivent les deux tendances évoquées dans la question 1. Nos résultats sont donc cohérents avec nos observations.

3. Implémentons une procédure bootstrap pour calculer les intervalles de confiance associés aux proportions de variance expliquées par les 10 premières composantes principales, par la méthode de notre choix (e.g., quantile, basique, ...). Propospons une représentation graphique de nos résultats.

```
alpha = 0.05
B = 50
n = dim(X)[1]
x_size = length(pv)
prop_variance_est = matrix(0,B,x_size)
born_inf = numeric(B)
born_sup = numeric(B)
cols= 1:B
plot(pv,type = "l",col="red",ylab = "proportion de variance")
# Methode de bootstrap
for(b in 1:B){
  ind = sample(1:n,size=n,replace=TRUE)
  pca = prcomp(X[ind,])
  s=summary(pca)
  prop_variance_est[b,] = s$importance[2,][1:10]
  # Calcul des intervalles de confiance
  born_inf[b] = quantile(prop_variance_est[b,],alpha/2)
  born_sup[b] = quantile(prop_variance_est[b,],1-(alpha/2))
  lines( prop_variance_est[b,],col=cols[b])
  lines(born_inf,col="blue",lty=3)
  lines(born_sup,col="red",lty=3)
```

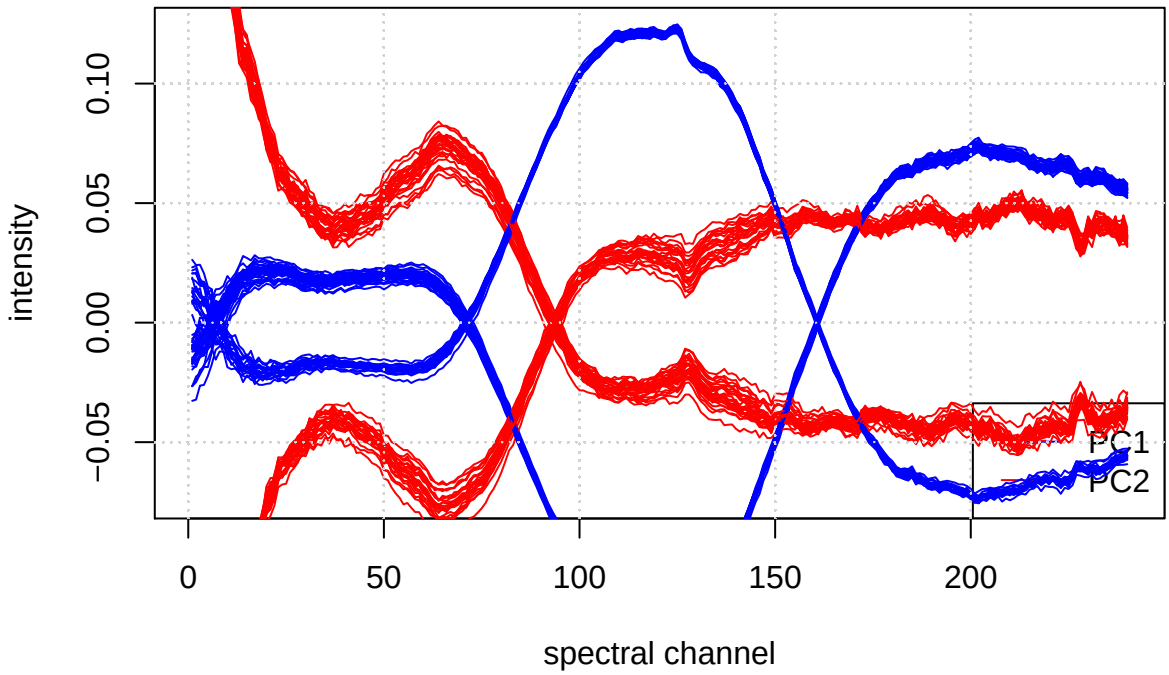
```
}
```



Les intervalles de confiances sont représentés ici par les deux lignes en pointillé : en haut par la couleur rouge (borne supérieur) et en bas par la couleur bleue (borne inférieur).
Les proportions des variances des dix premières composantes principales estimées par la méthode de bootstrap(B=50) sont situées dans l'intervalle de confiance et converge vers les vraies proportions de variances calculées à l'aide d'une ACP sans la méthode bootstrap.

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

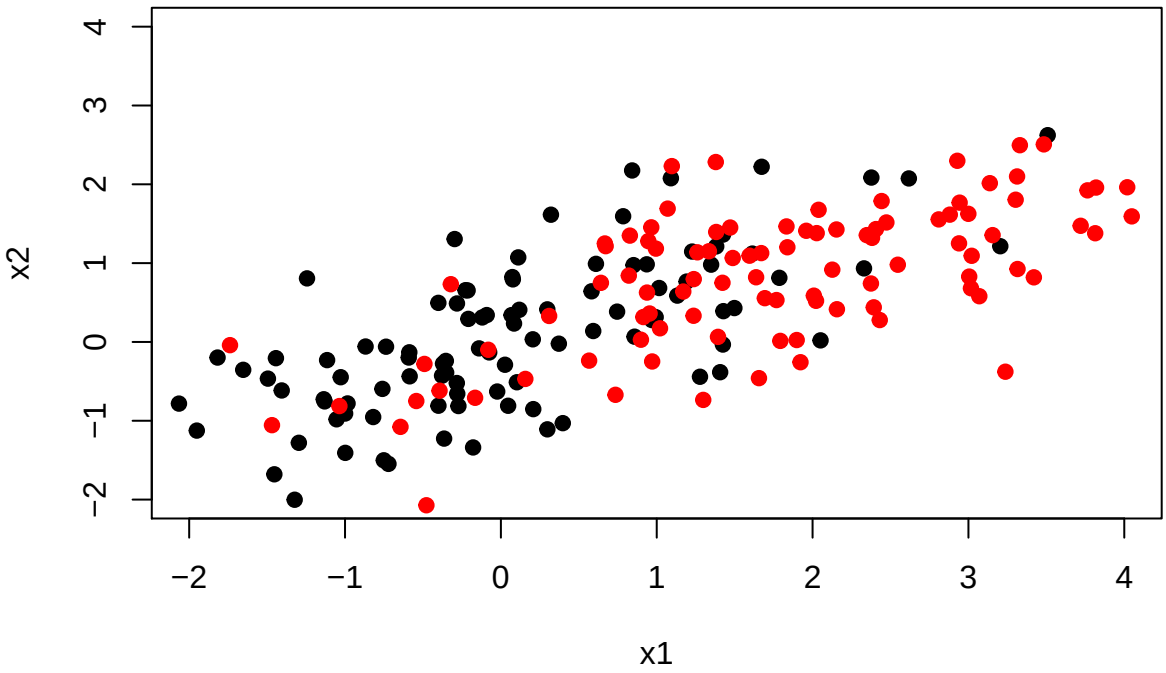
```
plot(1:240,pca$rotation[,"PC1"],type="l",col="blue",lty=1,xlab="spectral channel",
     ylab="intensity",panel.first = grid())
lines(1:240,pca$rotation[,"PC2"],type="l",col="red",lty=1)
legend("bottomright",legend=c("PC1","PC2"),col = c("blue","red"),lty=1)
for(b in 1:B){
  ind = sample(1:n,size=n,replace=TRUE)
  pca_bootstrap = prcomp(X[ind,])
lines(1:240,pca_bootstrap$rotation[,"PC1"],xlab="spectral channel",ylab="intensity",
      panel.first = grid(),col="blue")
lines(1:240,pca_bootstrap$rotation[,"PC2"],xlab="spectral channel",ylab="intensity",
      panel.first = grid(),col="red")
}
```



On remarque que les variabilités des intensités des spectres représentés sur les deux composantes principales décrivent chacune deux allures symétriques. Pour chaque composante, l'une des symétries suit la même allure que celle représentée dans la question 2 (figure 2).

Exercice 4 : tests par permutation

```
#load data
load("permutation-dataset.Rdata")
ind = 1:100
plot(X[ind,],pch=19,xlab="x1",ylab = "x2",xlim=c(-2,4),ylim = c(-2,4))
points(X[-ind,],col="red",pch=19)
```



On souhaite tester si on est capable de détecter une diffèrence entre ces deux populations, i.e., si on est capable de détecter une différence significative entre les deux nuages de points, stockés dans les 100 premièrs et les 100 dernières lignes de la matrice. On considére pour cela une statistique basée sur une information de voisinage :

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

1. Que mesure cette statistique ?

Cette statistique mesure la probabilité de l'ensemble des points qui sont voisins parmis les n points (n=200), en distinguant ceux qui sont noirs et rouges.

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

```
# calcul des distances
n_points = 200
d = dist(X)
dmat = as.matrix(d,nrow=n_points,ncol=n_points,bycol=T)
nxi = c()
for(i in 1:n_points){
  # nombre de voisins
  nxi = c(nxi,which.min(as.vector(dmat[,i][-i]))+1)
}

procedure_permutation = function(n,B){
  # n : nombre de points tirés par hasard parmis les 200 points
  # B : nombre de permutation
  # calcul de t
  ind = 1:n
  nxi1 = nxi[ind]
  nxi2 = nxi[-ind]
  xnoir = nxi1[nxi1<n]
```

```
  xrouge = nxi2[nxi2>n]
  t = (length(xnoir) + length(xrouge)) /n_points
  t.perm=numeric(B)
  for(b in 1:B){
    ind = sample(n_points,n)
    nx1 = nxi[ind]
    nx2 = nxi[-ind]
    xnoir = nx1[nx1<n]
    xrouge = nx2[nx2>n]
    t.perm[b] = (length(xnoir) + length(xrouge)) / n_points
  }
  t.perm = c(t.perm,t)
  pval = mean(t.perm >=t)
  return(pval)
}
n = 100
B=1000

cat("La p-valeur de la statistique t est : ",procedure_permutation(n,B))
```

```
## La p-valeur de la statistique t est :  0.000999001
```

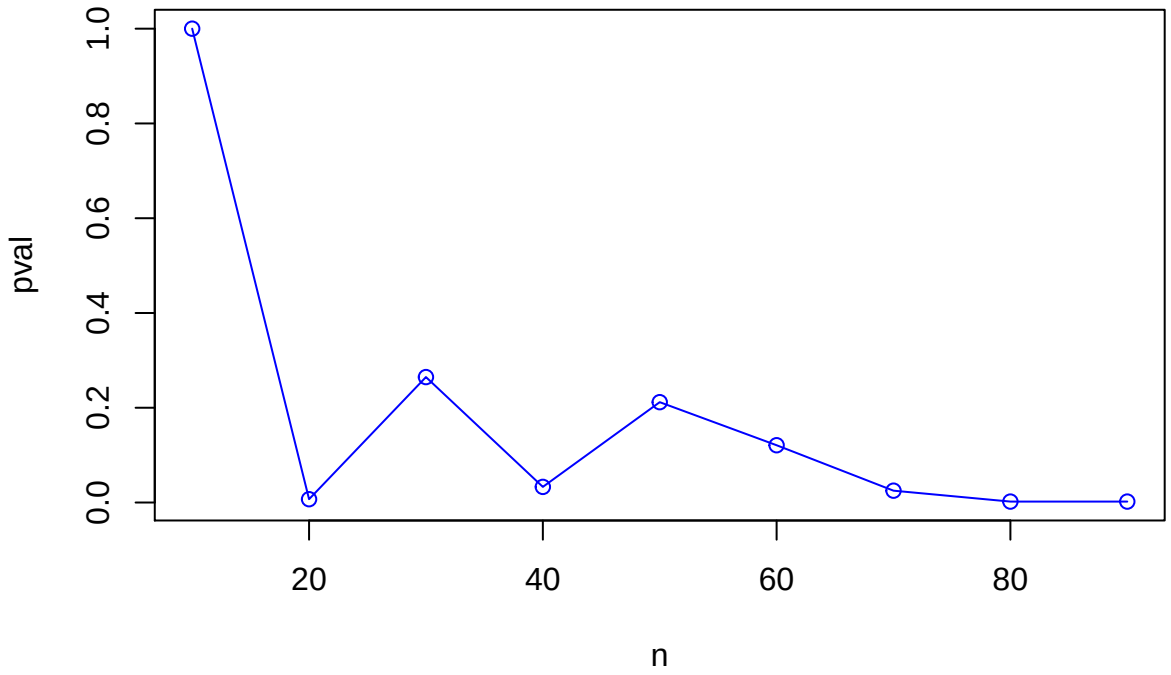
Soient les deux hypoteses suivantes:
H_0 : Il n'y a pas de difference significative entre les nuages de points
H_1 : Il existe une difference significative entre les nuages de points

Decision :

La p-valeur trouvée est inferieur à la valeur de alpha fixée ($\alpha = 0.05$), donc on rejette H_0 et on accepte H_1. Ce qui montre qu'il y a une difference significative entre les points rouges et noirs.

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

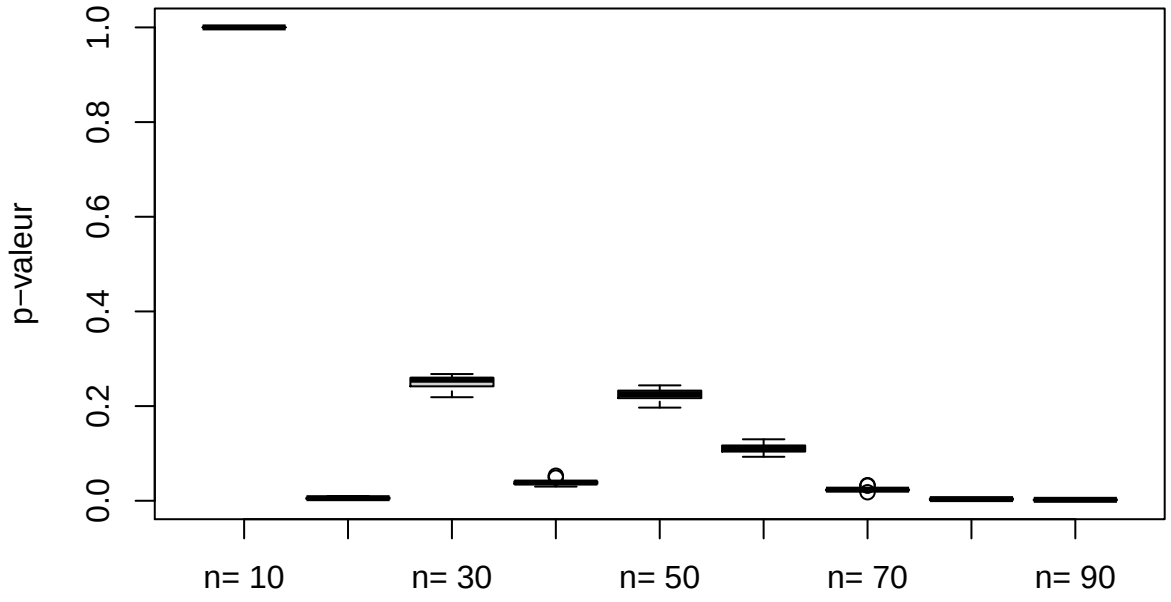
```
n = seq(10,90,by=10)
sizen = length(n)
pval = numeric(sizen)
for(i in 1 : sizen){
  pval[i] = procedure_permutation(n[i],B)
}
plot(n,pval,col="blue")
lines(n,pval,col="blue")
```



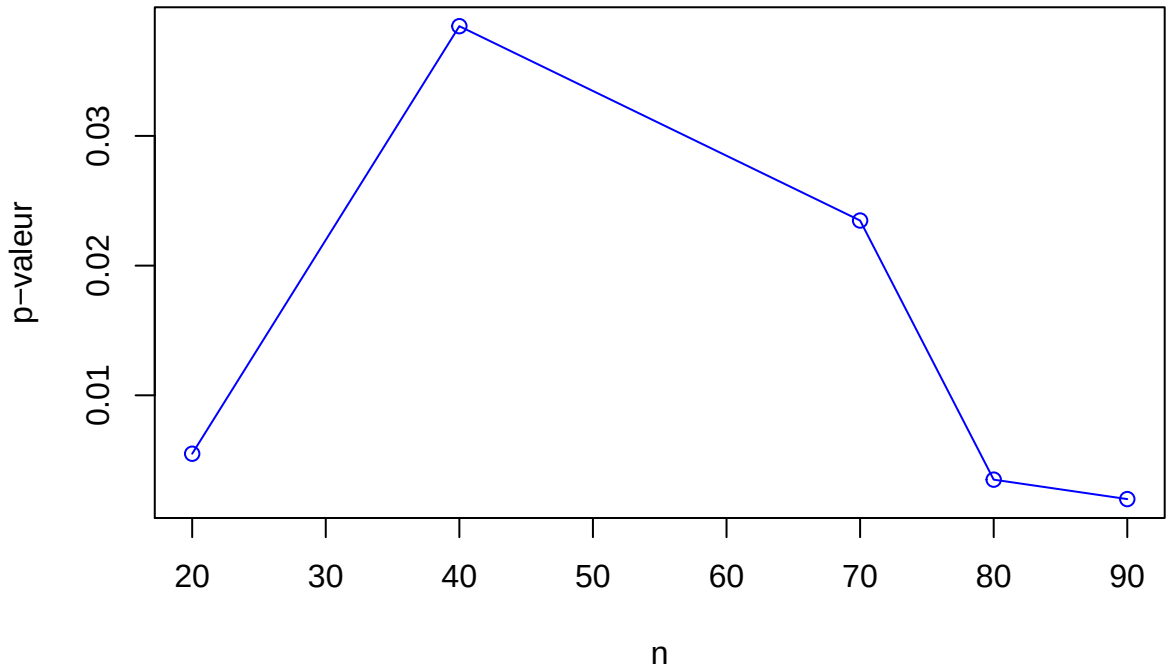
On remarque que les p-valeurs décroîts de façon générale quand n devient grand.

4. Enfin, reproduisons cette seconde analyse 10 fois et représentons la variabilité dans les p valeurs 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 a 0.05 ?

```
pval = matrix(0,10,sizen)
for(i in 1 : 10){
  for(j in 1:sizen){
    pval[i,j] = procedure_permutation(n[j],B)
  }
}
boxplot(pval,names=paste("n=",n),ylab="p-valeur")
```



```r
# calcul de la mediane
mediane_pval = apply(pval,2,median)
ind = which(mediane_pval<0.05)
plot(n[ind],mediane_pval[ind],col="blue",xlab="n",ylab="p-valeur")
lines(n[ind],mediane_pval[ind],col="blue")
```



D'après la figure de la question 2 (représentation des p-valeurs), on constate que la différence entre les points rouges et noirs est significative mais cela ne nous permet pas d'affirmer une significativité à partir de n = 20 car n = 30 ne l'est pas.
Pour mieux illustrer la significativité de la différence des points noirs et rouges nous avons pris les valeurs médianes des p-valeurs de chaque n qui sont supérieur à 0.05. On remarque qu'il y a une très grande variabilité pour les valeurs de n comprise entre 20 et 70 mais devient faible à partir de n = 80. Cela nous permet d'affirmer qu'on peut détecter une différence significative à partir de n = 80.