M2 - Probabilités et statistiques des nouvelles données

Projet Statistiques en Grande Dimension

Auteur :

Boukhari Yassine

Octobre 2021

Table des matières

Chapitre 1

Partie I

Simulation de nos modèles

Dans cette partie on utilisera les fonctions *hist* et *density* afin de simuler nos variables aléatoires $(X_1, ..., X_n) \sim \frac{1}{2}f_1(x) + \frac{1}{2}f_2(x)$ selon les trois modèles suivants :

1- $f_1 \sim \mathcal{N}(2,1)$ et $f_2 \sim \mathcal{N}(-1, 0.5)$

2- $f_1 \sim \mathcal{U}([0,1])$ et $f_2 \sim \mathcal{N}(-1, 0.5)$

3- $f_1 \sim \Gamma(2,4)$ et $f_2 \sim \Gamma(2,1)$

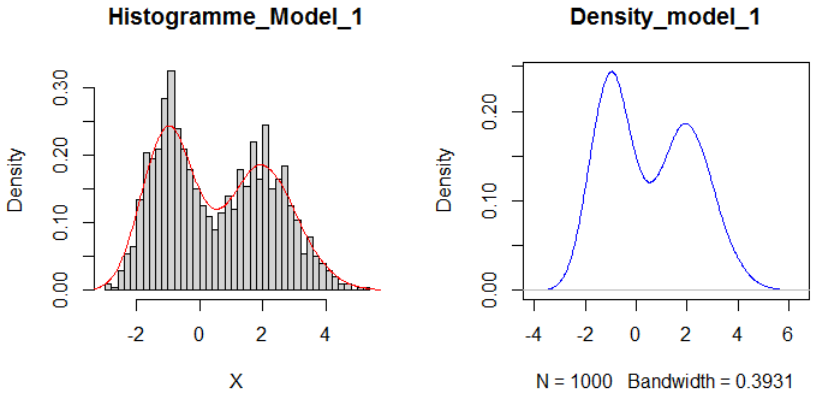
Modèle 1

On commencera par simuler le modèle où $f_1 \sim \mathcal{N}(2,1)$ et $f_2 \sim \mathcal{N}(-1, 0.5)$, on code sur R de la façon qui suit :

```
#Model 1
g1<-function(n){
 b<-rbinom(n,1,0.5)
 u<-rnorm(n,2,1)
 v<-rnorm(n,-1,sqrt(0.5))
 X=(b)*u+(1-b)*v
 return(X)
}
```

Après cela on simule nos variables aléatoire en prenant $n = 1000$ et on affiche l'histogramme.

```r
#Plot du Hist et density pour le model 1
par(mfrow=c(1,2))
n=1000
X<-g1(n)
hist(X,prob=T,breaks = 40,main="Histogramme_Model_1")
lines(density(X,kernel="gaussian",bw="nrd0"),col="red")
plot(density(X),col="blue",main = "Density_model_1")
```



On verra par la suite comment se fera le choix de la meilleure fenêtre h à partir des plots.

Modèle 2

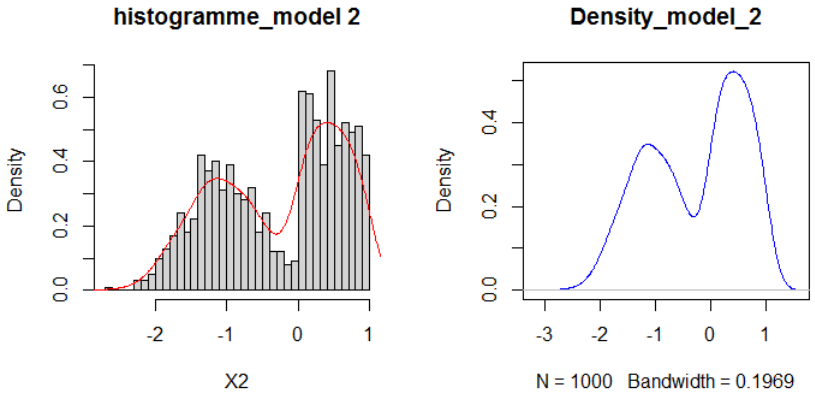
Dans ce cas ci présent on s'intéresse à la simulation de variables aléatoires suivant le cas où notre modèle vérifie $f_1 \sim \mathcal{U}([0,1])$ et $f_2 \sim \mathcal{N}(-1, 0.5)$, on code cela de la façon qui suit :

```r
g2=function(n){
  u<-runif(n,0,1)
  v<-rnorm(n,-1,sqrt(0.5))
  b<-(runif(n,0,1)<1/2)
  X=(b)*u+(1-b)*v
  return(X)
}
```

De même on s'intéresse au plot de l'histogramme et la fonction de densité de notre vecteur aléatoire $X = (X_1, ..., X_n)$ suivant ce modèle qu'on obtient à partir de ce qui suit :

```r
par(mfrow=c(1,2))
n=1000
```

```
X2<-g2(n)
hist(X2,breaks = 40,freq=FALSE,main = 'histogramme_model 2')
lines(density(X2,bw="nrd0"),col='red')
plot(density(X2),col="blue",main = "Density_model_2")
```



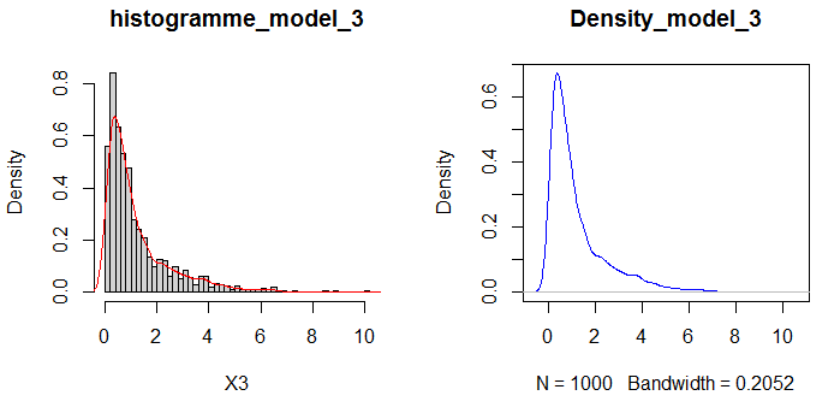
Modèle 3

En dernier on présente le cas où nos variables aléatoires sont simulées à l'aide du modèle vérifiant $f_1 \sim \Gamma(2,4)$ et $f_2 \sim \Gamma(2,1)$:

```
g3=function(n){
  u<-rgamma(n,2,4)
  v<-rgamma(n,2,1)
  b<-(runif(n,0,1)<1/2)
  X=(b)*u+(1-b)*v
  return(X)
}
```

Comme précédemment on simule notre vecteur aléatoire X suivant ce modèle et on affiche son histogramme et sa densité comme suit :

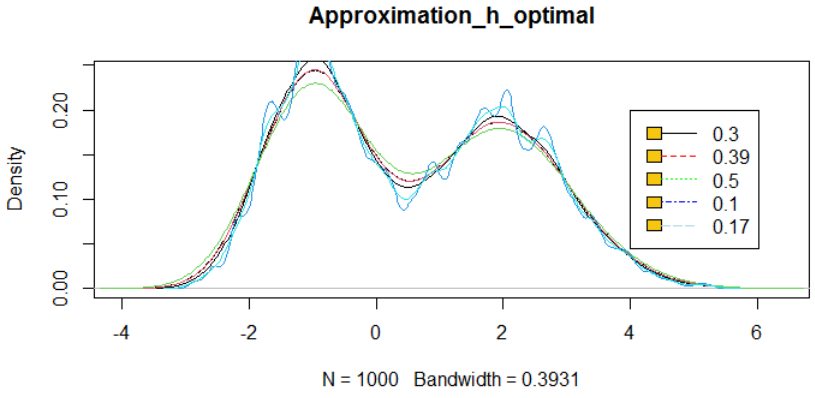
```
par(mfrow=c(1,2))
n=1000
X3<-g3(n)
hist(X3,breaks = 40,freq=FALSE,main = 'histogramme_model_3')
lines(density(X3,bw="nrd0"),col='red')
plot(density(X3),col="blue",main = "Density_model_3")
```



Après avoir établi les différentes représentations de notre modèle on revient au premier et on essaie d'observer la fenêtre optimale et cela grâce aux différents plots selon différentes valeur de h on verra de là celle qui approche au mieux notre fonction densité.

```
#Approximation graphique de la fenetre h par tatonnement
bws=c(0.3,0.39,0.5,0.1,0.17)

plot(density(X),main = "Approximation_h_optimal")
for (i in 1:length(bws)){
lines(density(X,kernel="gaussian",bw=bws[i]),col=i)
}
legend(4,0.2,15,legend=c('bw'=0.3,'bw'=0.39,
'bw'=0.5,'bw'=0.1,'bw'=0.17),
       col=c("black","red","green",
       "blue","lightblue","magenta")
       ,lty=1:5)
```



A partir des différents plot la valeur de la fenêtre h qui arrive à bien calibrer notre densité est celle qui admet pour valeur $h \simeq 0.39$ est cela est vérifié par le *Bandwidth=0.3931* qui donne la valeur précise de notre fenêtre.

1.1 Partie II

Approximation de la fenêtre Par méthode du MISE

Dans cette partie on s'intéresse à l'approximation de la fenêtre du premier modèle en s'appuyant sur la méthode du *MISE*. Cette méthode consiste à trouver h *optimal* tel que le risque intégré *(MISE)*$\rightarrow 0$ sous les conditions que $h_n \rightarrow 0$ et $nh \rightarrow 0$.
L'idée sera d'approximer notre densité f grâce à une fonction $\hat{f}_n$ pour cela notre h devra vérifier la condition

$$h \in argmin_{h>0}\ MISE(\hat{f}_{n,h}) \text{ où } MISE(\hat{f}_{n,h}) = \mathbb{E}(\int (\hat{f}_{n,h}(x) - f(x))^2 dx)$$

tel que :

$$\begin{aligned} \hat{f}_{n,h}(x) &= \frac{1}{nh}\sum_{i=1}^{n} K(\frac{X_i - x}{h}) \\ K(x) &= \frac{1}{\sqrt{2\pi}} \exp^{\frac{-1}{2}x^2} \end{aligned}$$

où on sélectionne *K()* comme noyau gaussien et $\hat{f}_{n,h}()$ est l'estimateur gaussien de notre densité $f(x) = \frac{1}{2}f_1(x) + \frac{1}{2}f_2(x)$ avec $f_1 \sim \mathcal{N}(-1, 0.5)$ et $f_2 \sim \mathcal{N}(2, 1)$. On commence par vérifier que la trajectoire de notre $\hat{f}_{n,h}()$ suit bien celle de notre densité.

• Étape 1 Simuler X selon notre le modèle 1 et fixer nos variables d'entrée

```
n=1000

g<-function(x){
  b<-rbinom(n,1,0.5)
  u<-rnorm(n,2,1)
  v<-rnorm(n,-1,sqrt(0.5))
  X=(b)*u+(1-b)*v
  return(X)
}
X<-g(n) #Simuler X selon le modele 1
h=seq(0.001,2,0.01)# n=length(h)
```

```
a=min(X)#Valeur de bords inf de l'integrale
b=max(X)#Valeur de bords sup  de l'integrale
```

• Étape 2 Créer les fonctions pour le noyau gaussien $K()$ et estimateur gaussien $\hat{f}_{n,h}$

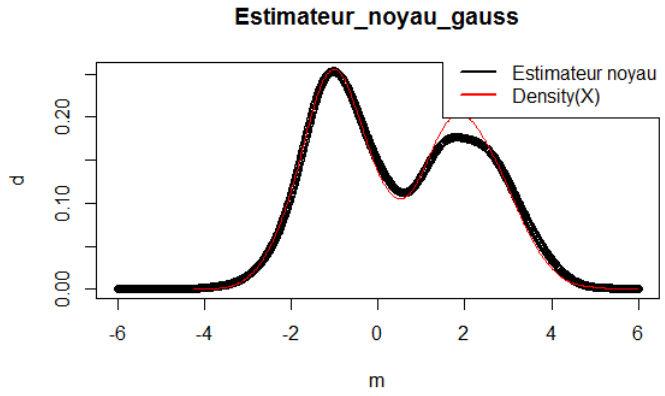
```
#Construction Estimateur_noyau_gaussien

#Noyau gaussien K()
noyau_gauss=function(x){
  return(1/sqrt(2*pi)*exp(-x**2/2))
}

estimateur_noyau_gauss=function(X,x,h){
  # ou h vecteur, X vecteur, n scalaire
  res=0
  n=length(X)
  # res_final=c()
  # for (i in 1:length(h))
  for(j in 1:n){
    res=res+noyau_gauss((x-X[j])/h)
  }
  return(1/(n*h)*res)}
```

• Partie 3 Simulation du plot $\hat{f}_{n,h}$

```
#Plot de l'estimateur noyau gaussien
n=1000
m<-seq(-6,6,0.01)
p<-g(n)
d=c()
for(i in 1:length(m)){
  d[i]=estimateur_noyau_gauss(p,m[i],0.3957)
}
plot(m,d,main = "Estimateur_noyau_gauss")
lines(density(X,kernel="gaussian",bw="nrd0"),col="red")
legend(x=1.5,y=0.28,legend = c("Estimateur␣noyau","Density(X)"),col= c('black','
       ,lwd = 2)
densite = function(x){
  return(0.5*(dnorm(x,2,1))+0.5*(dnorm(x,-1,sqrt(0.5))))
}
```



La courbe de $\hat{f}_{n,h}$ suit bien celle de la densité.
Passons maintenant à l'estimation de notre h (fenêtre) optimal de notre modèle, pour cela on construit la fonction MISE qui fera intervenir la fonction **integrate()** sur R. Et on termine comme le montrera le code ci dessous par application de Monte-Carlo pour l'étape du calcul de l'espérance

```
#Construction du MISE

#densite du modele
densite = function(x){
  return(0.5*(dnorm(x,2,1))+0.5*(dnorm(x,-1,sqrt(0.5))))
}
fct_int=function(X,h){
int_mc=c()
a=min(X)
b=max(X)
for (i in (1:length(h))){
f_int<-function(x) (estimateur_noyau_gauss(X,x,h=h[i])-densite(x))**2
int_mc[i]=integrate(f_int,lower=a,upper=b,subdivisions =2000)$value
}
return(int_mc)}
#Calcul de la matrice Monte Carlo
#ou chaque colonne represente un MC applique sur h[i]

Mat_mc<-function(nb_iteration,n,h){
M=matrix(nrow = nb_iteration,ncol=length(h))
for (i in 1:nb_iteration){
  print(i)
```

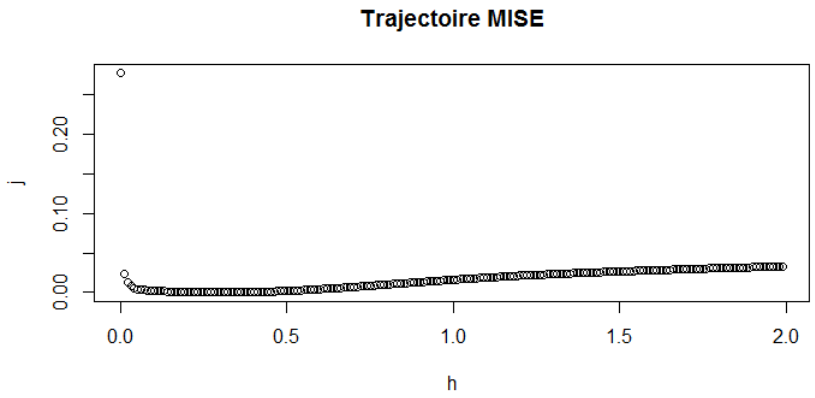
```
  x=g(nb_iteration)
  M[i,]=fct_int(X,h)
}
MISE_final=c()
for (i in 1:length(h)){
  MISE_final[i]=mean(M[,i])
}
return(MISE_final)
}
```

Pour trouver le h minimal du MISE on utilise le code qui suit :

```
j=Mat_mc(30,n,h)

plot(h,j,main = 'Trajectoire MISE')
m=which.min(j)
h_hat=h[m]
h_hat
```

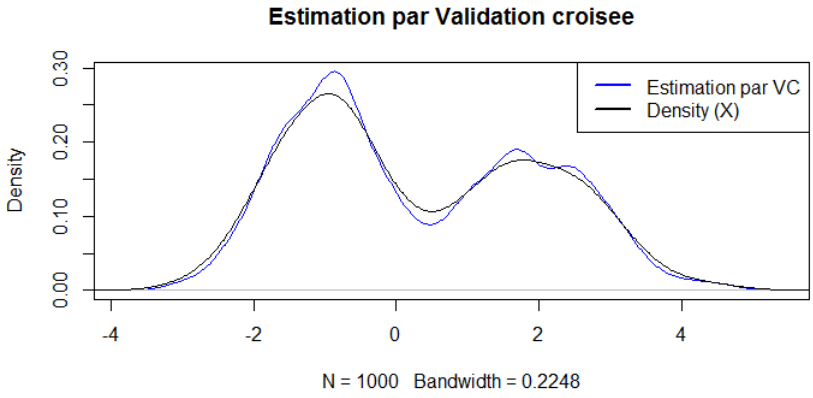
On tombe sur une valeur $h_{hat} \simeq 0.33$ justifié par le plot qui suit



Voyons maintenant si avec la méthode par validation croisée la valeur de notre fenêtre h sera impacté pour cela on fera appel à la fonction de R density() vu précédemment

```
#Plot par validation croisee
plot(density(X,bw="ucv"),col='blue',main = 'Estimation par Validation croisee')
lines(density(X,bw=h_hat),col='green')
lines(density(X))
legend(x="topright",legend = c("Estimation par VC",
"Estimation par h_hat",'Density (X)'),
```

```
lty=c(1,1,1),col= c('blue',"green",1)
        ,lwd = 3)
```



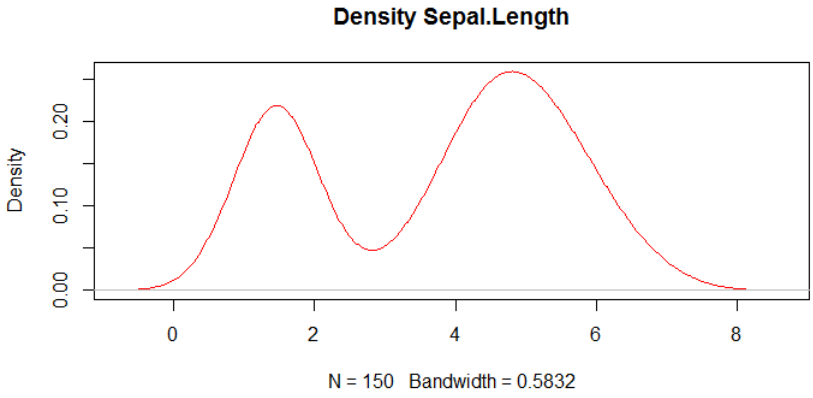
Dans ce cas on pourra constater que le h optimal obtenu par Validation croisée est égale $h_{cv} \simeq 0.22$. Il en résulte une moins bonne approximation de la densité qu'avec $h_{hat} \simeq 0.33$. Chose qui conclue la partie 2 consacré au *MISE* maintenant nous allons passé à la partie qui s'intéresse à l'utilisation de la validation croisée sur des données réelles.

1.2 Partie III

Estimation de la fenêtre par validation croisée

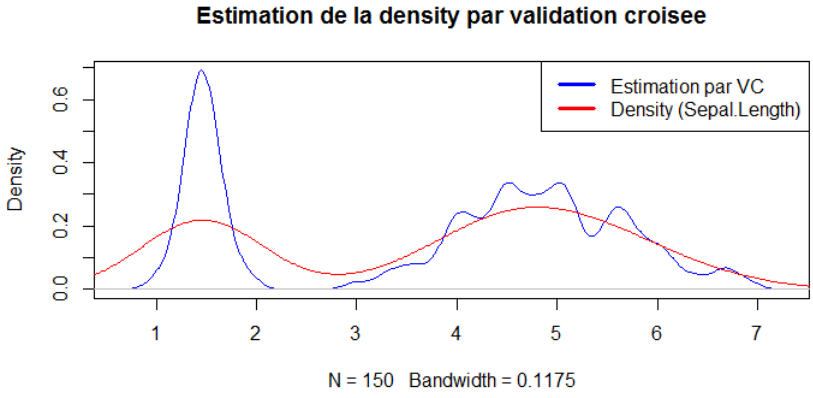
Pour ce cas là on s'intéresse à l'étude de l'une des données les plus utilisées sur R qui sont *"les Iris de fisher"* et en particulier *"Petal.Length"* (Longueur des pétales). Notre base de données sera composée de 150 observations et aura pour density une qui s'approche de notre premier modèle comme le montre le plot qui suit :

```
#Importation des data et Plot de notre density
data("iris")
attach(iris)
data1=Petal.Length
plot(density(data1,bw="nrd0"),col="red",main = "Density␣Sepal.Length")
```



Maintenant passons comme vu précédemment à l'aide de la fonction **density()** à l'estimation d'une densité par méthode de validation-croisée on choisira au vu de la forme de nos data un noyau *gaussien*. On obtient par la suite la figure qui suit :

```
#Comparaison entre la vrai densite et celle approximee par VC
#Plot selon le principe de Validation croisee
plot(density(data1,bw="ucv",kernel = 'gaussian'),
col="blue",main = "Estimation de la density par validation croisee")
lines(density(data1),col="red")
legend(x="topright",
legend = c("Estimation par VC",
'Density (Sepal.Length)'),
lty=c(1,1),col= c('blue',"red")
       ,lwd = 3)
```



1.3 Partie IV

Algorithme d'estimation par Validation croisée

Dans cette partie on présentera un programme qui aura pour objectif les étapes effectuées par les fonction *density()* afin d'estimer une fenêtre h par validation croisée. Pour cela on fera notre étude sur le premier modèle afin de simuler notre échantillon de données X

Commençons d'abords par fixer l'objectif à atteindre contrairement à la méthode classique pour estimer la fenêtre h. Ici nous allons nous appuyer sur le principe du **leave-one-out** qui permet de créer une sorte d'indépendance à $\hat{f}_{n,h}$.
Pour $i = 1, ..., n$ on considère $\hat{f}_{n,h}^{(-i)} = \frac{1}{(n-1)h} \sum_{j \neq i} K(\frac{X_j - x}{h})$ et on a :

$$\mathbb{E}[\frac{1}{n} \sum_{i=1}^{n} f_{n,h}^{(-i)}(X_i)] = \mathbb{E}[\int \hat{f}_{n,h}(x) f(x) dx]$$

On aboutit par la suite au critère à vérifier pour trouver la fenêtre optimal par Cross-validation

$$h_{cv} \in argmin \int \hat{f}_{n,h}^2(x) - \frac{2}{n} \sum_{i=1}^{n} \hat{f}_{n,h}^{(-i)}(X_i)$$

On pose $CV_{gaussian} = \int \hat{f}_{n,h}^2(x) - \frac{2}{n} \sum_{i=1}^{n} \hat{f}_{n,h}^{(-i)}(X_i)$

• Étape 1 : Construction de la formule $CV_{gaussian}$

```
#Taille de l'echantillon & Data
n=1000
X=g(n)
#Programme pour le cross-validation
cv_gaussien = function(h){
  new_data = sort(X)
  #membre de droite :
  k = 0
  for (i in (1:n)){
    for (j in (1:n))
      if(i!=j)
        k=k+noyau_gauss((new_data[j]-new_data[i])/h)
  }
  k=k*(2/(n*h*(n-1)))
```

```
  #estimation
  integrale=0
  for(i in (2:n)){
    print(i)
    integrale = integrale + (new_data[i]-new_data[i-1])*(estimateur_noyau_gauss(
  }
  return(integrale-k)
}
}
```

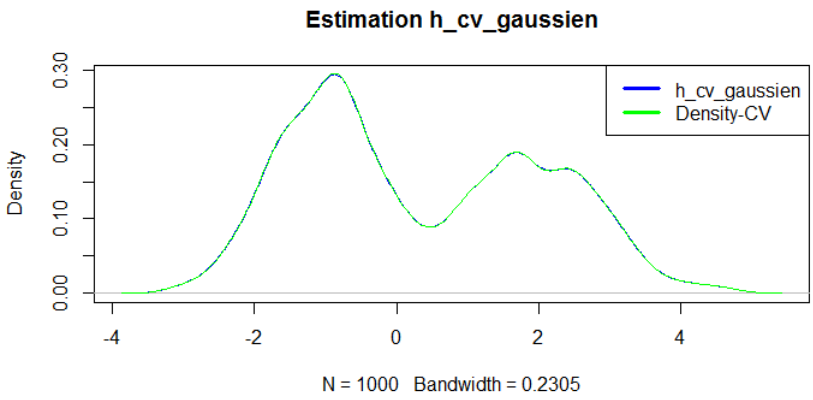
• Étape 2 : Recherche du Minimum à l'aide de la fonction **optimise()**

```
#hmin

h_cv_gaussien = optimise(cv_gaussien,lower=0.001,
upper=2,maximum=FALSE)$minimum
```

La valeur obtenue après simulation est $h_{cv}\_gaussien \simeq 0.23$ qui est une assez bonne approximation de la valeur exacte obtenue par density() pour la cross-validation comme le montre le plot ci-dessous

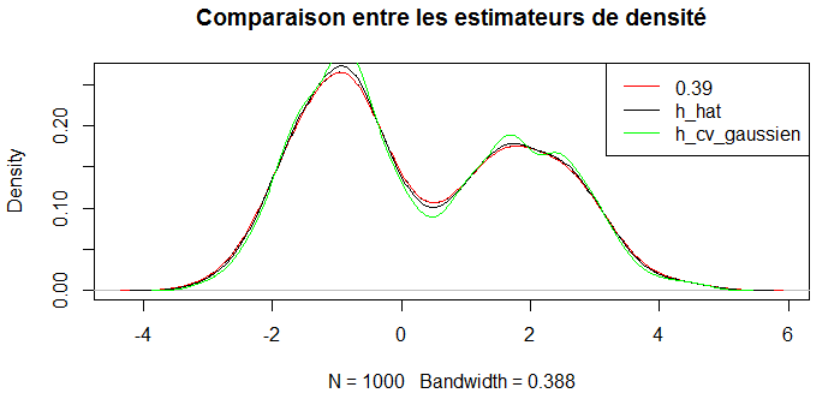
```
#Comparaison entre h_cv_gaussien(algo CV) et density() par Cross-validation
plot(density(X,bw=h_cv_gaussien),col="blue",
main = 'Estimation h_cv_gaussien')
lines(density(X,bw="ucv"),col="green")
legend(x="topright",legend = c("h_cv_gaussien",'Density-CV')
,lty=c(1,1),col= c('blue',"green")
       ,lwd = 3)
```



On conclut ce projet par la superposition des différents estimateurs de la densité obtenu dans chaque parties (MISE,Algo Cross-validation)

```
#Comparaison entre la fenetre h_hat obtenue par MISE
#et h_cv_gaussien par Cross-validation
bws=c(0.39,h_hat,h_cv_gaussien)
col=c('red','black','green')

plot(density(X),main = "Comparaison␣entre␣les␣estimateurs␣de␣densite")
for (i in 1:length(bws)){
  lines(density(X,kernel="gaussian",bw=bws[i]),col=col[i])
  legend(x='topright',legend=c("bw"='0.39',
  'bw'="h_hat",'bw'="h_cv_gaussien"),
         col=c("red","black","green"),
         lty=c(1,1,1))
}
```



Comme l'on peut le constater comme dans le cas où l'on a utiliser la fonction *density()* pour le *cross-validation* notre h_{hat} a donnée une meilleure approximation que $h_{cv}\_gaussian$ pour la fonction de densité du modèle 1.

Bibliographie

[1] C.Denis M.Hebiri, cours Statistiques grande dimension, *Université Gustave Eiffel*(2021)