DM

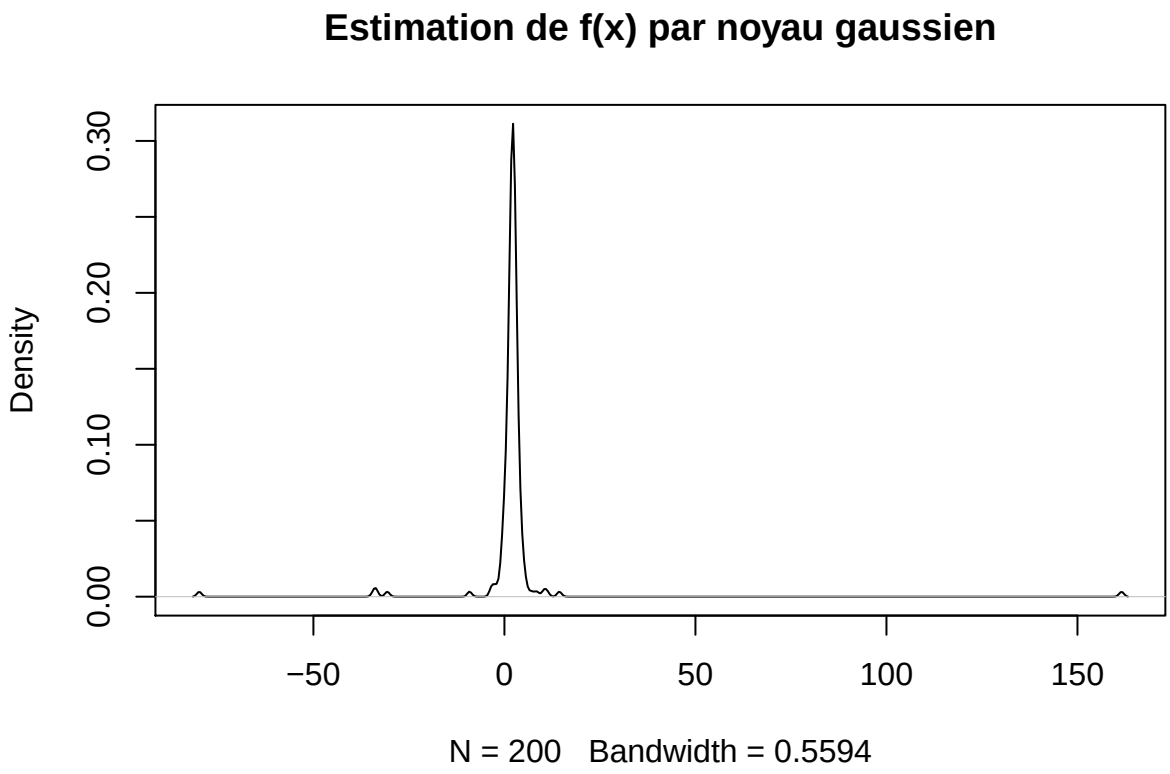
Vivien Dupont et Louis Petat-Lenoir

01/03/2021

Question 1

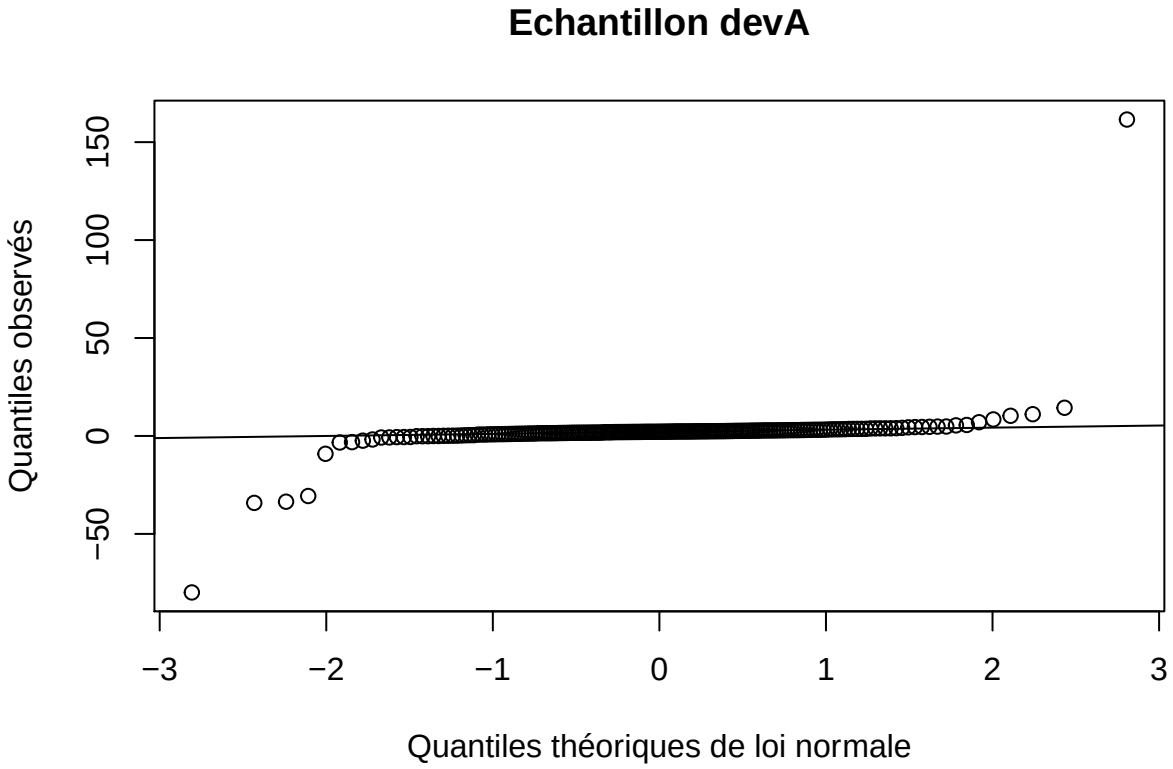
Le meilleur estimateur de $f(x)$ au sens de l'erreur quadratique moyenne intégrée est l'estimateur de Parzen-Rosenblatt $\hat{f}(x) = \frac{1}{nh_{cv}} \sum_{j=1}^{n} K(\frac{x-X_j}{h_{cv}})$ ou K est un noyau choisi et h_{cv} petit obtenue par validation croisé *leave-one-out*.

```
fn_chapeau <- density(devA$X1, bw="ucv", adjust=1, kernel="gaussian")
plot(density(devA$X1, bw="ucv", adjust=1, kernel="gaussian"), main = "Estimation de f(x) par noyau gauss
```



Question 2

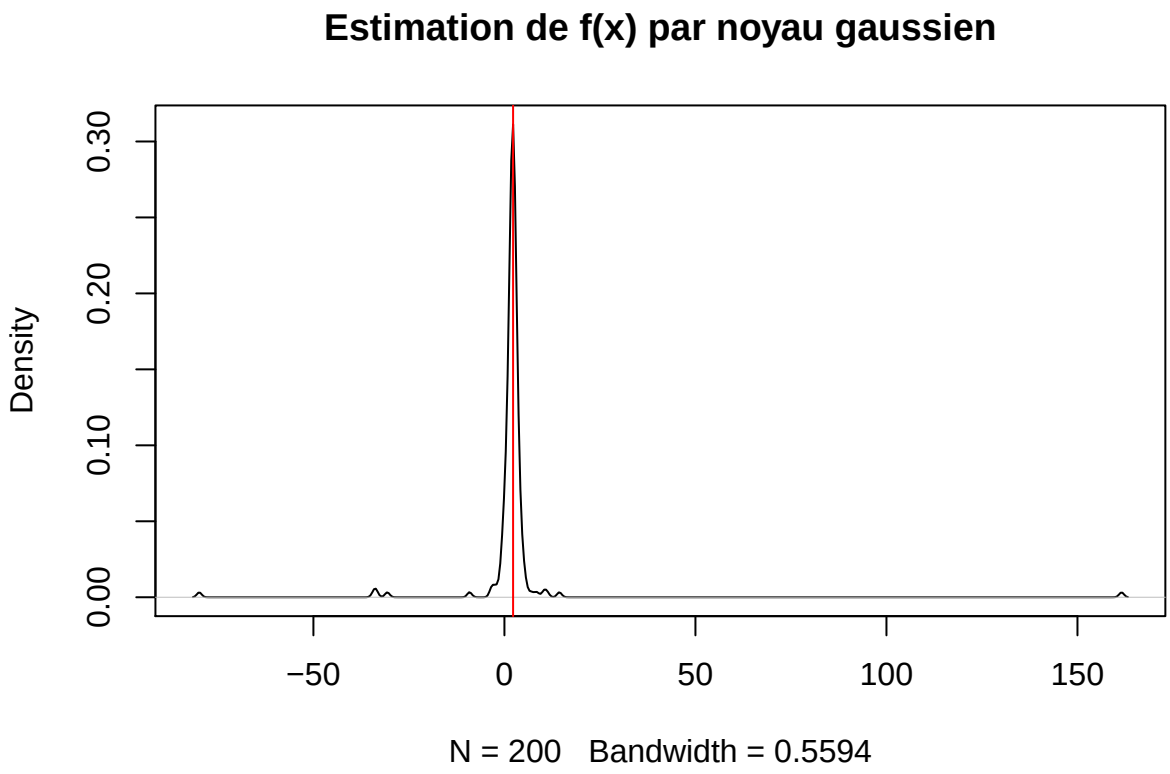
- Puisque l'on estime une loi de densité (semblant continue de par les nombreux chiffres après la virgule des observations) on souhaite que notre estimation ait les bonnes propriétés associées aux lois de densité. Ainsi il vient naturellement que le noyau K doit être une densité de probabilité, lisse, continue et différentiable. Par défaut et sans information supplémentaire j'ai décidé de retenir le noyau gaussien. De plus, en faisant exception des valeurs extrêmes de notre echantillon, la partie centrale de la distribution de notre echantillon semble suivre une loi normale. Néanmoins le choix du noyau importe peu puisque c'est avant tout le choix du paramètre de lissage qui intervient dans l'arbitrage biais-variance. Plus h est choisi petit et plus le biais sera faible et la viariance grande, si h est grand ce sera l'inverse.



- Le choix de la fenêtre h est réalisé par validation croisée *leave-one-out* puisque $h_{cv} = argmin_h[\int \hat{f}_n^2(x)dx - \frac{2}{n}\sum_{i=1}^{n}\hat{f}_{-i}(X_i)]$.
h_{cv} est alors égal à 0.5593725.

Question 3

$f(x)$ est une loi symétrique par rapport à θ_0. Par conséquent, on peut estimer graphiquement θ_0 par l'abscisse du point de maximum de la courbe. *i.e.*



On approxime alors θ_0 par $\theta_0^{approx} = 2.2656625$.

Question 4

On peut faire mieux que ces approximations :

On choisit le noyau gaussien : $K(x) = \frac{1}{\sqrt{2\pi}} exp(-x^2/2)$
notre fonction de densité est estimée par l'estimateur de Nadaraya-Watson : $\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x-X_i}{h})$
la largeur de fenêtre obtenue par validation croisée : $h_{cv} = 0.5593725$

Or :
$\tilde{\theta} = argmax_\theta \sum_{i=1}^{n} ln \hat{f}_n(2\theta - X_i)$
$\tilde{\theta} = argmax_\theta \sum_{i=1}^{n} ln[\frac{1}{nh} \sum_{j=1}^{n} K(\frac{2\theta - X_i - X_j}{h})]$
$\tilde{\theta} = argmax_\theta \sum_{i=1}^{n} ln \sum_{j=1}^{n} K(\frac{2\theta - X_i - X_j}{h})$

On dérive la fonction dans le argmax par rapport à θ afin de la maximiser :

$g^{'}(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{K^{'}(\frac{2\theta - X_i - X_j}{h})}{K(\frac{2\theta - X_i - X_j}{h})}$

$g^{'}(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{(\frac{-2\theta}{h^2} + \frac{X_i}{h^2} + \frac{X_j}{h^2}) exp(-(\frac{2\theta - X_i - X_j}{h})^2/2)}{exp(-(\frac{2\theta - X_i - X_j}{h})^2/2)}$

$g^{'}(\theta) = \sum_{i=1}^{n} \sum_{j=1}^{n} (2\theta + X_i + X_j)$

$g^{'}(\theta) = 0$

équivaut à :

$2n^2\theta - n^2\bar{X_n} - n^2\bar{X_n} = 0$

$\tilde{\theta} = \bar{X_n}$

Ainsi en prenant le noyau gaussien et un paramètre de lissage optimal par validation croisée on trouve que l'estimateur asymptotiquement efficace pour θ est $\bar{X_n} = 2.0501181$
Or, $E[\bar{X_n}] = \frac{1}{n}\sum_{i=1}^{n} E[X1]$ car les X_i sont indépendants deux à deux, qu'ils sont issus de la même loi de densité $f(x)$ et que X_i semble converger dans L^1. Ainsi par la loi des grands nombres : $\tilde{\theta}$ converge vers la vraie valeur de θ asymptotiquement.

Partie B

Question 1

Sachant que les données observées sont issues d'une régression, que nos valeurs manquantes sont en fait des réponses et que l'hypothèse des observations MAR est satisfaite on a :

$Y = m(X) + \epsilon$, $E(\epsilon|X) = 0$ et $Y \perp\!\!\!\perp D, X$.

On souhaite estimer l'esperance de Y avec l'estimateur de Nadaraya-Watson: $\hat{r}(u) = \frac{\sum Z_i K(\frac{u-U_i}{h})}{\sum K(\frac{u-U_i}{h})}$ avec K le noyau et h la fenêtre.

On sait que $P(X) = P(D = 1|X) = E(D|X)$ donc on a :

$m(X) = E(Y|X) = \frac{E(DY|X)}{E(D|X)}$. On applique l'estimateur de Nadaraya-Watson sur E(DY|X) et E(D|X) et on obtient

$E(DY|X) = \frac{\sum D_i Y_i K(\frac{x-X_i}{h})}{\sum K(\frac{x-X_i}{h})}$ et $E(D|X) = \hat{p}(X) = \frac{\sum D_i K(\frac{x-X_i}{h})}{\sum K(\frac{x-X_i}{h})}$

ce qui nous donne l'estimateur $\hat{m}_n$ de m en simplifiant:

$\hat{m}_n(x) = \frac{\sum D_i Y_i K(\frac{x-X_i}{h})}{\sum D_i K(\frac{x-X_i}{h})}$

On cherche à écrire un programme permettant de calculer l'estimateur $\hat{\alpha}_n$ de $\alpha_0 = E(Y)$. On a calculé deux estimateurs de α_0 tels que $\hat{\alpha}_{n,1} = \sum \hat{m}(X_i)/n$ et $\hat{\alpha}_{n,2} = \frac{1}{n}\sum[\frac{D_i Y_i}{\hat{p}(X_i)} + (1 - \frac{D_i}{\hat{p}(X_i)})\hat{m}(X_i)]$

```
NW<-function(x,h=5){
  a<-c()
  kernel<-c()

  for (i in 1:200){
    if (is.na(dataB$V1[i])){
      kernel[i]<-0
      a[i]<-0}

    else{
      kernel[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*3.14)
      a[i]<-dataB$V1[i]*kernel[i]}


  }

  numerateur= sum(a)
  denominateur= sum(kernel)
```

```
  NaWa=numerateur/denominateur

  return(NaWa)
}


p_hat<-function(x,h=5){
  kernel<-c()
  kernel2<-c()

  for (i in 1:200){
    if (is.na(dataB$V1[i])){
      kernel[i]<-0
      }

    else{
      kernel[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*3.14)

      }

    kernel2[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*3.14)

  }

  numerateur= sum(kernel)
  denominateur= sum(kernel2)
  NaWa=numerateur/denominateur

  return(NaWa)
}

NW<-function(x,h=5){
  a<-c()
  kernel<-c()

  for (i in 1:200){
    if (is.na(dataB$V1[i])){
      kernel[i]<-0
      a[i]<-0}

    else{
      kernel[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*3.14)
      a[i]<-dataB$V1[i]*kernel[i]}


  }

  numerateur= sum(a)
  denominateur= sum(kernel)
  NaWa=numerateur/denominateur

  return(NaWa)
}
```

```
p_hat<-function(x,h=5){
  kernel<-c()
  kernel2<-c()

  for (i in 1:200){
    if (is.na(dataB$V1[i])){
      kernel[i]<-0
      }

    else{
      kernel[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*3.14)

      }

    kernel2[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*3.14)

  }

  numerateur= sum(kernel)
  denominateur= sum(kernel2)
  NaWa=numerateur/denominateur

  return(NaWa)
}
```

Pour ceci, nous avons construit les fonctions retournant respectivement $\hat{p}$ et $\hat{m}$ pour un certain x et un paramètre de lissage h en appliquant la contrainte D=1.Nous avons choisi tout d'abord h=5 d'après un critère graphique mais nous calculerons sa valeur optimale par une validation croisée ultérieurement.

```
phat<-sapply(dataB$V2,p_hat)
Nadi<-sapply(dataB$V2, NW)

alpha1_n<-sum(Nadi)/200


alpha2_n<-c()
for (i in 1:200){
  if (is.na(dataB$V1[i])){
    alpha2_n[i]<-Nadi[i]
  }
  else{
    alpha2_n[i]<-dataB$V1[i]/phat[i]+(1-1/phat[i])*Nadi[i]
  }

}
alpha2_n<-sum(alpha2_n)/200
```

Puis, il a fallu appliquer ces fonctions à l'ensemble des X_i avec la fonction sapply() retournant ainsi un vecteur de valeurs. Il suffit ensuite de construire les estimateurs $\hat{\alpha}_{n,1}$ et $\hat{\alpha}_{n,2}$ avec ces valeurs

Avec h=5 on obtient $\hat{\alpha}_{n,1}$ =0.9173743 et $\hat{\alpha}_{n,2}$= 0.9173858

Nous calculons ensuite le paramètre de lissage h optimal par validation croisée. Puisque $\hat{p}$ et $\hat{m}$ sont des estimateurs de regression différents, il faut trouver un h par validation croisée pour chacun d'entre eux.

Calculons tout d'abord l'estimateur $h_{p_{CV}}$ le paramètre de lissage de $\hat{p}$. On fait une validation croisée en leave-1-out en prenant:

$h_{p_{CV}} = \arg\min_h \frac{1}{n} \sum [D_i - \hat{p}_{n,h}^{(-i)}(X_i)]^2$

avec $\hat{p}_{n,h}^{(-i)} = \frac{\sum_{j \neq i}^{n} D_j K(.)}{\sum_{j \neq i}^{n} K(.)}$

```
P_moinsI<-function(x,h=1,k){
  D<-c()
  p<-c()
  kernel<-c()
  for (i in 1:200){
    kernel[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*pi)
    if (is.na(dataB$V1[i])){
      D[i]<-0
      p[i]<-0}
    else{
      D[i]<-1
      p[i]<-D[i]*kernel[i]}
  }
  p<-p[-k]

  numerateur= sum(p)
  denominateur= sum(kernel)
  NaWa=numerateur/denominateur

  return(NaWa)

}



h_p_cv<-function(h){
  summum<-c()
  for (k in 1:200){
    if (is.na(dataB$V1[k])){
      summum[k]<-P_moinsI(dataB$V2[k],h,k)**2
    }
    else{
      summum[k]<-(1-P_moinsI(dataB$V2[k],h,k))**2
    }}
  return(sum(summum)/length(summum))
}
```

On construit un premier algorythme pour déterminer les $\hat{p}_{n,h}^{(-i)}$ avec une fonction prenant en argument une observation x, un paramètre de lissage h qui variera à l'avenir et un paramètre k permettant de retirer l'observation i. Cette fonction retourne la valeur de $\hat{p}_{n,h}^{(-i)}$ pour un x_i fixé en vérifiant si Y_i est observé.

Une deuxième fonction prenant en argument h pour faire varier le paramètre de lissage et retournant la valeur $h_{p_{CV}}$ applique la fonction ci-dessus en différençiant les 200 Y_i observés ou non.

```
sequence<-seq(1,1.2,0.001)
estim_hp_cv<-sapply(sequence,h_p_cv)


min_hp_cv<-estim_hp_cv[1]
h_p_min<-sequence[1]
for (i in 1:length(estim_hp_cv)){

  if (estim_hp_cv[i]<min_hp_cv){
    min_hp_cv<-estim_hp_cv[i]
    h_p_min<-sequence[i]
  }
}
print(h_p_min)
```

```
## [1] 1.069
```

En appliquant à la dernière fonction plusieurs séquences de h différents, on essaye de determiner une approximation de h optimale c'est à dire la valeur de h pour laquelle $h_{p_{CV}}$ est $\arg\min_h \frac{1}{n}\sum[D_i - \hat{p}_{n,h}^{(-i)}(X_i)]^2$. On trouve donc pour $h_{p_{CV}}$ la valeur 1.069

Maintenant qu'on a trouvé le $h_{p_{CV}}$ par validation croisée pour $\hat{p}$ on va faire une validation croisée pour $\hat{m}$. l'estimation de h pour l'estimateur $\hat{m}$ par la validation croisée qui va

vérifier $h_{CV} = \arg\min_h \frac{1}{n}\sum[\frac{D_i Y_i}{\hat{p}_{n,h}^{(-i)}} - \hat{m}_{n,h}^{(-i)}(X_i)]^2$

avec $\hat{m}_{n,h}^{(-i)} = \dfrac{\sum_{j\neq i}^{n} D_j Y_j K(.)}{\sum_{j\neq i}^{n} D_j K(.)}$

Pour construire un algorithme permettant de determiner une approximation de h, il est néccessaire de construire $\hat{m}_{n,h}^{(-i)}$

```
NW_moinsI<-function(x,h=1,k){
  a<-c()
  kernel<-c()

  for (i in 1:200){
    if (is.na(dataB$V1[i])){
      kernel[i]<-0
      a[i]<-0}

    else{
      kernel[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*pi)
      a[i]<-dataB$V1[i]*kernel[i]}
    }

  a<-a[-k]
  kernel<-kernel[-k]
  numerateur= sum(a)
  denominateur= sum(kernel)
  NaWa=numerateur/denominateur
```

```
  return(NaWa)
}

h_cv<-function(h){
  summum<-c()
  for (k in 1:200){
    if (is.na(dataB$V1[k])){
      summum[k]<-NW_moinsI(dataB$V2[k],h,k)**2
    }
    else{
      summum[k]<-(dataB$V1[k]/P_moinsI(dataB$V2[k],h=1.069,k=k)-NW_moinsI(dataB$V2[k],h,k))**2
    }}
  summum=summum[summum!=0]
  return(sum(summum)/length(summum))
}
```

Dans la même logique que pour $h_{p_{CV}}$, on construit un premier algorythme pour déterminer les $\hat{m}_{n,h}^{(-i)}$ avec une fonction prenant en argument une observation x, un paramètre de lissage h qui variera à l'avenir et un paramètre k permettant de retirer l'observation i. Cette fonction retourne la valeur de $\hat{m}_{n,h}^{(-i)}(X)$

Une deuxième fonction prenant en argument h pour faire varier le paramètre de lissage et retournant la valeur h_{CV} applique la fonction ci-dessus en différençiant les 200 Y_i observés ou non. Dans cette fonction, on applique la valeur optimale de $h_{p_{CV}}$ pour la valeur de $\hat{p}_{n,h}^{(-i)}$ calculée prédédemment.

```
sequence<-seq(0.5,1.5,0.01)
estim_hcv<-sapply(sequence,h_cv)


min_hcv<-estim_hcv[1]
h_min<-sequence[1]
for (i in 1:length(estim_hcv)){

  if (estim_hcv[i]<min_hcv){
    min_hcv<-estim_hcv[i]
    h_min<-sequence[i]
  }
}
print(h_min)
```

```
## [1] 0.6
```

En appliquant à la dernière fonction plusieurs séquences de h différents, on essaye de determiner une approximation de h optimale c'est à dire la valeur de h pour laquelle h_{CV} est minimal. On trouve donc pour h_{CV} la valeur 0.6

On remplace ainsi ensuite la valeur h=5 par h=1.069 pour $\hat{p}$ et h=0.6 pour $\hat{m}$ dans les fonctions permettant de calculer ces estimateurs

```
h_validcroisee=1.069
NW<-function(x,h=0.6){
  a<-c()
  kernel<-c()
```

```
  for (i in 1:200){
    if (is.na(dataB$V1[i])){
      kernel[i]<-0
      a[i]<-0}

    else{
      kernel[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*pi)
      a[i]<-dataB$V1[i]*kernel[i]}


  }

  numerateur= sum(a)
  denominateur= sum(kernel)
  NaWa=numerateur/denominateur

  return(NaWa)
}


p_hat<-function(x,h=1.069){
  kernel<-c()
  kernel2<-c()

  for (i in 1:200){
    if (is.na(dataB$V1[i])){
      kernel[i]<-0
    }

    else{
      kernel[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*pi)

    }

    kernel2[i]<-exp(-(((x-dataB$V2[i])/h)**2)/2)/sqrt(2*pi)

  }

  numerateur= sum(kernel)
  denominateur= sum(kernel2)
  NaWa=numerateur/denominateur

  return(NaWa)
}
phat<-sapply(dataB$V2,p_hat)
Nadi<-sapply(dataB$V2,NW)

theta_tild1<-sum(Nadi)/200
print(theta_tild1)
```

```
## [1] 0.8078243
```

```r
theta_tild2<-c()
for (i in 1:200){
  if (is.na(dataB$V1[i])){
    theta_tild2[i]<-Nadi[i]
  }
  else{
    theta_tild2[i]<-dataB$V1[i]/phat[i]+(1-1/phat[i])*Nadi[i]
  }

}
theta_tild2<-sum(theta_tild2)/200
print(theta_tild2)
```

```
## [1] 0.9011848
```

On obtient $\hat{\alpha}_{n,1}$ =0.8078243 et $\hat{\alpha}_{n,2}$= 0.9011848 ce qui est proche de la moyenne empirique. Par résultat du cours $\hat{\alpha}_{n,1}$ et $\hat{\alpha}_{n,2}$ sont des estimateurs consistants et asyptotiquement efficaces de $\alpha_o = E(Y)$. On aurait aussi pu appliquer la loi des grands nombres pour le prouver

Question 2

Pour le choix des paramètres de lissage, nous avons choisi de les determiner par validation croisée leave-one-out car cette méthode minimise les erreurs de prédiction. Quant à la méthode, nous avons appliqué l'estimateur de NW à un jeu de données avec valeurs manquantes en prenant un noyau gaussien. Le choix du noyau importe peu, seul le choix du paramètre de lissage permet de réaliser un arbitrage Biais-variance.