BERGUIGA Oussama

Master Statistique & Big Data

DM Régression Non Paramétrique

Importation des données

```
getwd()
```

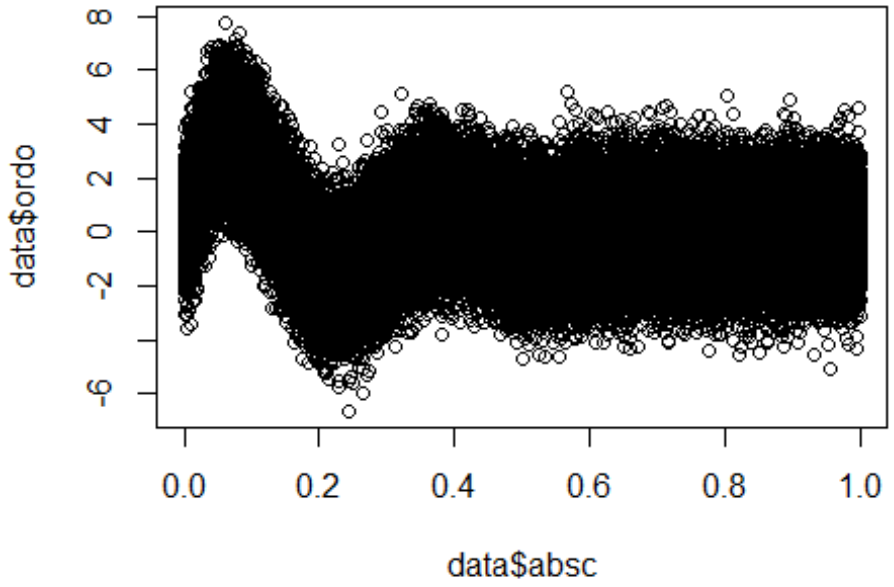
```
## [1] "C:/Users/oussa/Downloads/Data Science/Master Statistique Big Data
Dauphine/Module 2/Regression non-paramétrique"
```

```
setwd("C:/Users/oussa/Downloads/Data Science/Master Statistique Big Data
Dauphine/Module 2/Regression non-paramétrique")
list.files()
```

```
data=read.table('DataReg')
summary(data)
```

```
##       absc                ordo
##  Min.   :0.0000046   Min.   :-6.6500
##  1st Qu.:0.2498369   1st Qu.:-0.8347
##  Median :0.5014559   Median : 0.1267
##  Mean   :0.5009858   Mean   : 0.2378
##  3rd Qu.:0.7526427   3rd Qu.: 1.1674
##  Max.   :0.9999953   Max.   : 7.7522
```

```
plot(data$absc,data$ordo)
```



I) Exploration des propriete de g(x)

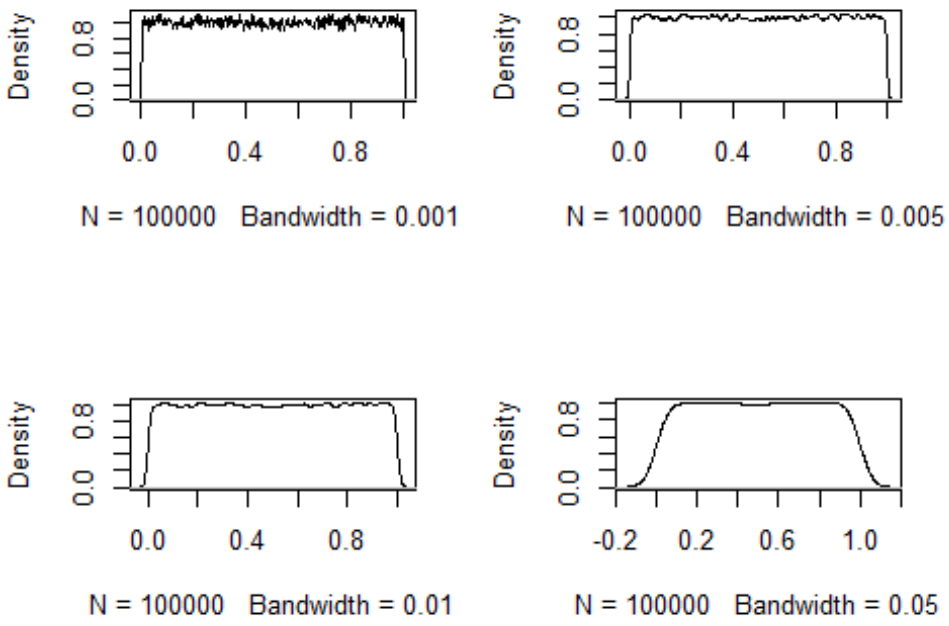
Estimateur non paramétrique de g

D'après le cours, un estimateur de g est donné par $\hat{g}_{n,h}(x) = L_n\big(K_h(x-.)\big) =$

$\frac{1}{n} * \sum_{i=0}^{n}(K_h(x_0 - X_i))$ et $K_h(x_0 - x) = \frac{1}{h} * K(\frac{x_0 - x}{h})$ et K un noyau ie une fonction de **R** dans **R** telle que $\int_{\mathbf{R}} K(x)\,dx = 1$. Le noyau est d'ordre k si pour tout l entier entre 1 et k:

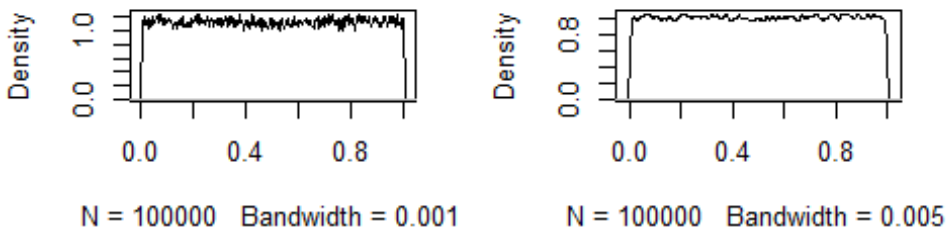
$$\int_{\mathbf{R}} K(x) * x^l\,dx = 0$$

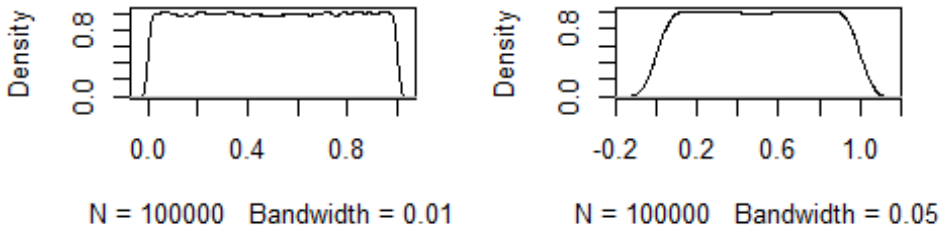
```
par(mfrow=c(2,2))
v=c(0.001,0.005,0.01,0.05)
for (i in 1:length(v))

{plot(density(data$absc,bw=v[i],kernel="gaussian"),main="")}
```



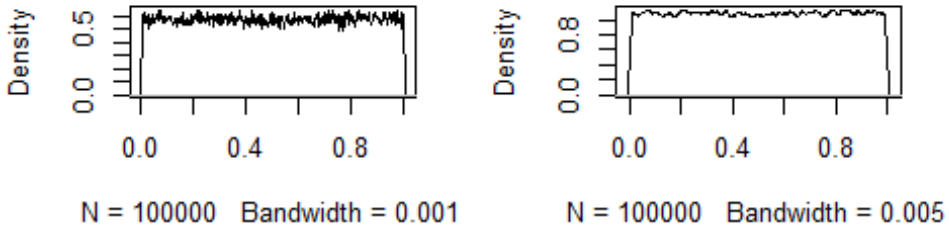
Choix de différents noyaux et discussion de g=1, notamment aux bords de [0,1]

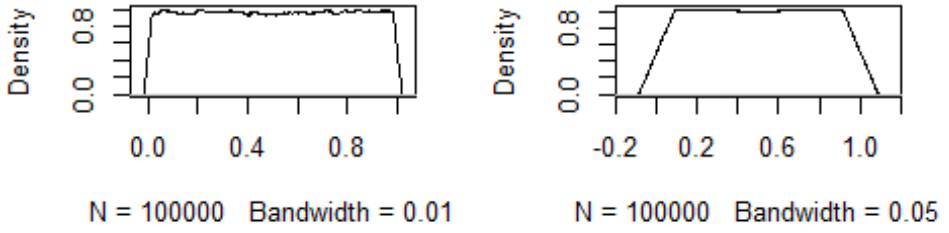
```
par(mfrow=c(2,2))
v=c(0.001,0.005,0.01,0.05)
for (i in 1:length(v))

{plot(density(data$absc,bw=v[i],kernel="triangular"),main="")}
```






```
par(mfrow=c(2,2))
v=c(0.001,0.005,0.01,0.05)
for (i in 1:length(v))

{plot(density(data$absc,bw=v[i],kernel="rectangular"),main="")}
```





A première vue il est plausible de penser que g est constante (égale à 1) sur [0,1], ce qui signifierait que la variable X suit une loi uniforme sur [0,1]. Cependant, si on choisit un noyau "rectangular", on peut penser que la densité de g est une fonction affine sur les bords de [0,1].

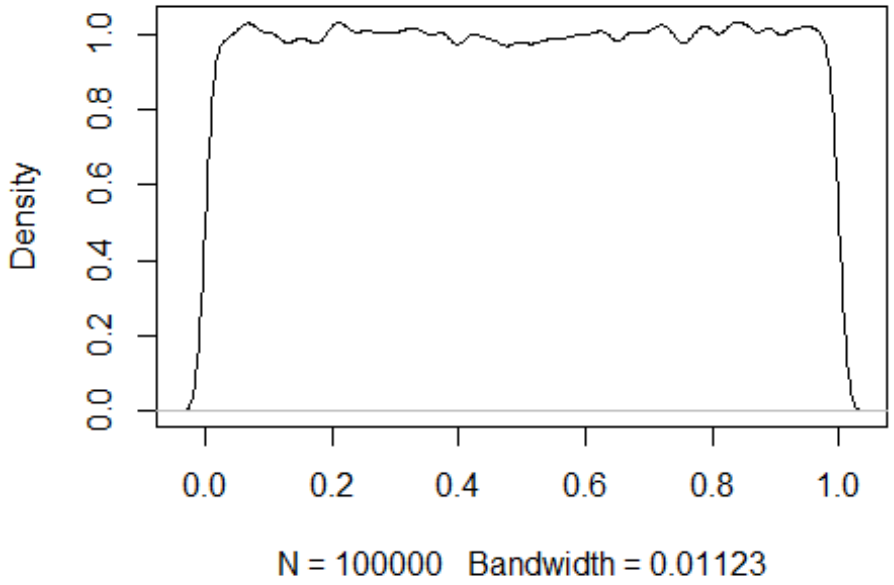
Choix de la fenêtre optimale (noyau gaussien)

La fenêtre optimale est obtenue en minimisant un estimateur obtenu par validation croisée :

$$\hat{h} = \min_h \hat{J}_h \text{ avec } \hat{J}_h = \int_{\mathbf{R}} \hat{g}_{n,h}(x)\,\mathrm{d}x - 2 * \frac{1}{n}\sum_{i=1}^{n} \hat{g}_{(-i),n,h}(X_i)$$

```
#on peut utiliser la fonction bw.ucv ou dpik pour trouver la fenêtre minimale

library(KernSmooth)

## Warning: package 'KernSmooth' was built under R version 3.3.3

## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009

dpik(data$absc)

## [1] 0.01123104

#library(stats)
#bw.ucv(data$absc)
```

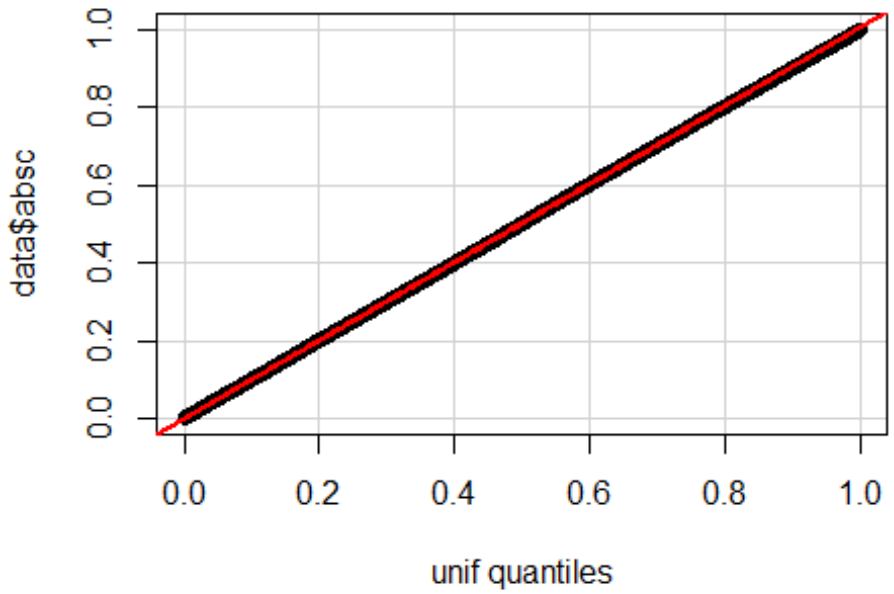
```
#Remarque : la fonction bw.ucv du package "stats" donne un tout autre
résultat

plot(density(data$absc,bw=dpik(data$absc),kernel="gaussian"),main="g density
gaussian Kernel best bendwith choosen with cross validation")
```



Avec un noyau gaussien et une fenêtre optimale, on peut penser que g est la densité de la loi uniforme sur [0,1]

Test de l'hypothèse g=1,ie loi uniforme

```
library(car)

## Warning: package 'car' was built under R version 3.3.3

qqPlot(data$absc,distribution="unif")
```



D'après le QQplot, g colle parfaitement à la densité de la loi uniforme sur [0,1]

Question facultative

$c^{(k)}$ est le moment d'ordre k des X_i.D'après la loi des grands nombres, un estimateur $c_n^{(k)}$ de $c^{(k)}$ est : $\frac{1}{n} * \sum_{i=0}^{n}(K_h(x_0 - X_i))$

$$\hat{c}_{n,k}\,(x) = \frac{1}{n}\sum_{i=1}^{n} X_i^k$$

Construction d'un intervalle de confiance

Posons H_0 : g est la densité de la loi uniforme D'après la loi des grands nombres, si l'hypothese de loi uniforme est juste,l'estimateur devrait converger vers $\frac{1}{k+1}$

D'après le cours, $\sqrt{n}\big(L_n(\phi) - L(\phi)\big)$ converge en loi vers une loi normale d' espérance nulle et de variance($Var\big(\phi(x)\big)$ Ici $\phi(X)$=X^k donc comme sous H_0 X suit la loi uniforme,on utilise la formule de l'esperance d'une transformée de variable pour trouver : $Var\big(\phi(x)\big) =$

$$\mathbb{E}(X^{2k}) - \mathbb{E}(X^k)^2 = \frac{1}{(2*k+1)} - \frac{1}{(k+1)^2} = \frac{k^2}{(2*k+1)*(k+1)^2)}$$

Intervalle de confiance pour une loi gaussienne :

Soient μ l'espérance et σ^2 la variance, μ appartient à l'intervalle $[X_n - q_\alpha * \frac{\sigma}{\sqrt{n}}, X_n + q_\alpha * \frac{\sigma}{\sqrt{n}}]$ au risque α

```
#on calcul l'estimateur sur les 10^5 valeurs de X

n=1e5

c_k=function(k){
return (1/n*sum((data$absc)^k))

}

c_k(4)
```

```
## [1] 0.2015518
```

```
#Application


intervalle_confiance=function(n,k){
v=c(0,0)


#quantile 99% d'une gaussienne
q_99=qnorm(0.995)

sigma_k=sqrt(k^2/(((2*k+1)*(k+1)^2)))

mu_k=1/(k+1)

v[1]=mu_k-q_99*sigma_k/sqrt(n)
v[2]=mu_k+q_99*sigma_k/sqrt(n)
  return(v)
}




#on genere un vecteur v de p entiers uniformement tires sur [1,m], et on
teste pour tout element k apparenant à v si c_k,n appartient à l'intervalle
de confiance
test=function(m,p){
t=0
```

```
v=round(runif(p,1,m))

for (j in v)

    {


if (c_k(j)>=(intervalle_confiance(n,j)[1]) &
c_k(j)<=(intervalle_confiance(n,j)[2]))

    {t=t+1}

  }
return(t/p)

  }

test(1e10,1e4)

## [1] 1
```

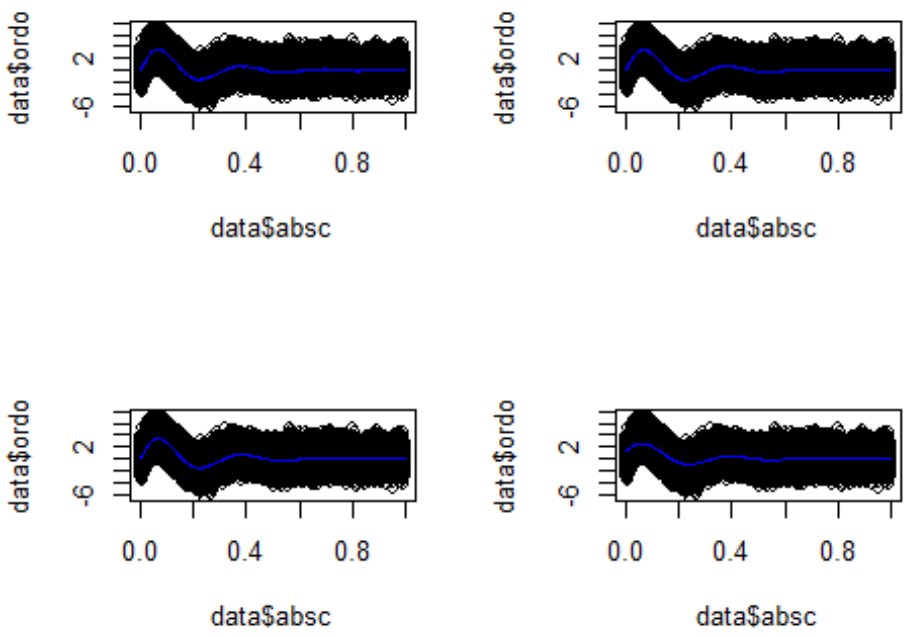
Conclusion : on a de fortes présomptions de penser que g est la densité de la loi uniforme sur [0,1]

II) Recondstruction de r(x)

Construction de l'estimateur de Nadariya-Watson

Estimateur de Nadariya-Watson est donné par : $\hat{r}_{n,h}(x_0) = \frac{\sum_{i=1}^{n} K_h(x_0 - x_i) y_i}{\sum_{i=1}^{n} K_h(x_0 - x_i)}$

```
library(KernSmooth)
par(mfrow=c(2,2))
v=c(0.001,0.005,0.01,0.05)
for (i in 1:length(v))


{plot(data$absc,data$ordo)

lines(locpoly(data$absc,data$ord,bandwidth=v[i]),main="",type='l',col='blue')
}
```



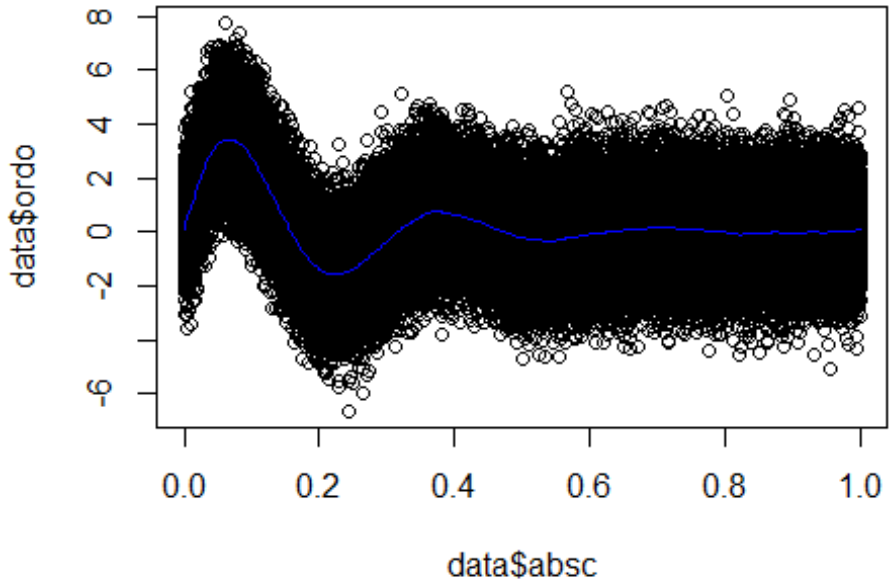
Inconvénients sur h

h trop petit : sous-lissage, estimateur est très irrégulier reproduit simplement les observations

h trop grand : sur-lissage, on se rapproche de l'espérance des Yi.

Cependant, on aura tendance à préférer un h petit (puisque dans la théorie h doit tendre vers 0)

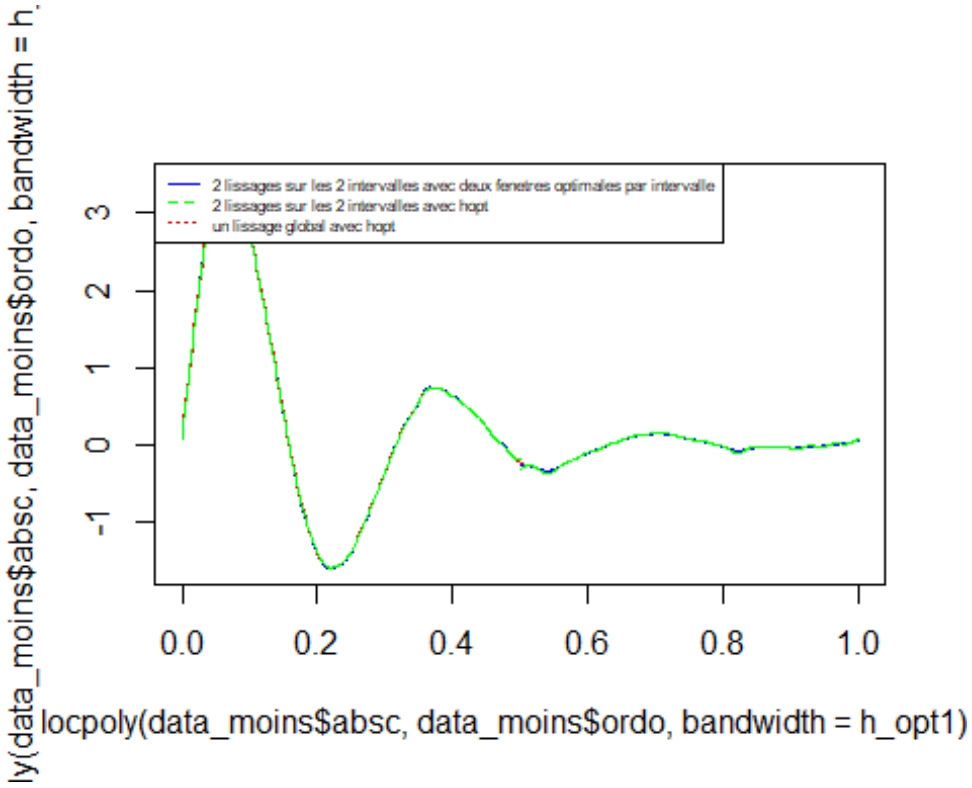
```
h_opt=dpill(data$absc,data$ordo)
print(h_opt)

## [1] 0.008072254

plot(data$absc,data$ordo)
lines(locpoly(data$absc,data$ord,bandwidth=h_opt),main="",type='l',col='blue'
)
```



Decoupage du jeu de données

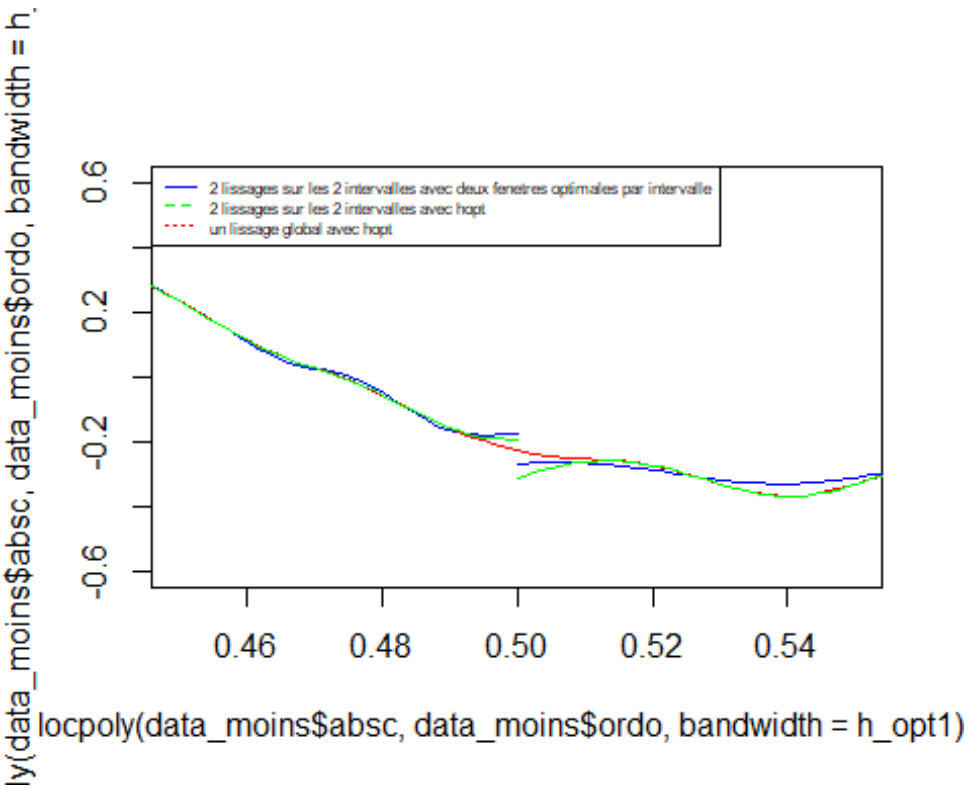
```
data$test=data$absc>=(1/2)

data_plus=data[data$test==TRUE,1:2]

data_moins=data[data$test==FALSE,1:2]


h_opt1=dpill(data_moins$absc,data_moins$ordo)
h_opt2=dpill(data_plus$absc,data_plus$ordo)

print(c(h_opt1,h_opt2))

## [1] 0.006171008 0.014692359

#en bleu : 2 lissages sur les 2 intervalles avec deux fenetres optimales par
intervalle

#en vert 2 lissages sur les 2 intervalles avec hopt

#en rouge un lissage global avec hopt

plot(locpoly(data_moins$absc,data_moins$ordo,bandwidth =
h_opt1),col="blue",type='l',xlim=c(0,1))
lines(locpoly(data_plus$absc,data_plus$ordo,bandwidth =
h_opt2),col="blue",type='l')
```

```
lines(locpoly(data$absc,data$ordo,bandwidth=h_opt),main="",col="red",type='l'
)

lines(locpoly(data_moins$absc,data_moins$ordo,bandwidth =
h_opt),col="green",type='l',xlim=c(0,1))
lines(locpoly(data_plus$absc,data_plus$ordo,bandwidth =
h_opt),col="green",type='l')
legend("topleft", c("2 lissages sur les 2 intervalles avec deux fenetres
optimales par intervalle","2 lissages sur les 2 intervalles avec hopt","un
lissage global avec hopt"), col = c("blue","green","red"), lty =
1:3,cex=0.50)
```




```
plot(locpoly(data_moins$absc,data_moins$ordo,bandwidth =
h_opt1),col="blue",type='l',xlim=c(0.45,0.55),ylim=c(-0.6,0.6))
lines(locpoly(data_plus$absc,data_plus$ordo,bandwidth =
h_opt2),col="blue",type='l')
lines(locpoly(data$absc,data$ordo,bandwidth=h_opt),main="",col="red",type='l'
)

lines(locpoly(data_moins$absc,data_moins$ordo,bandwidth =
h_opt),col="green",type='l')
lines(locpoly(data_plus$absc,data_plus$ordo,bandwidth =
h_opt),col="green",type='l')

legend("topleft", c("2 lissages sur les 2 intervalles avec deux fenetres
optimales par intervalle","2 lissages sur les 2 intervalles avec hopt","un
```

```
lissage global avec hopt"), col = c("blue","green","red"), lty =
1:3,cex=0.50)
```



On peut tirer 2 enseignements de ces graphiques :

- le choix de la fenêtre optimale est lié à la variation de la fonction sur un intervalle donné : on remarque que les 2 fenêtres sont très différentes (du simple au double). En effet la variation des Y_i (et donc de $r(x)$) est bien plus importante sur l'intervalle [0,0.5] que sur l'intervalle [0.5,1]
- la régression au bord d'un intervalle peut être faussée par manque d'information : cf ici la régression au point Xi=0.5.Le lissage au point 0.5 est plus juste dans le cas du lissage de r sur [0,1] car on dispose de 2 fois plus d'informations sur le comportement des Y_i au voisinage de 0.5 que dans le cas de 2 lissages séparés [0,0.5] et [0.5,1]

III) Etude de la loi des ξ_i

REMARQUE : dans cette partie je considère les Xi NON DETERMINISTES donc d'après la relation Y=r(X)+ ξ,et par independance de x et (donc r(X))et ξ : Var(Y)=Var (r(X))+ σ^2

Implementation

```
sigma_hat=function(y,n)

{

   return (1/(2*(n-1))*sum(diff(y)[1:(n-1)]^2))
}
```

```
n=1e5
sigma_hat(data$ordo,n)
```

```
## [1] 2.589785
```

Justification

```
var(data$ordo)
```

```
## [1] 2.575987
```

```
abs(sigma_hat(data$ordo,n)-var(data$ordo))/var(data$ordo)#0.5% d'erreur relative
```

```
## [1] 0.005356477
```

On pose $Z_i = \frac{(Y_{i+1}-Y_i)^2}{2}$, alors d'après la loi des grands nombres, l'estimateur de Rice converge vers l'espérance de Zi. Or $\mathbb{E}(Z_i)=\frac{1}{2} * \mathbb{E}((Y_i - Y_{i+1})^2)$ =

$\frac{1}{2} * \mathbb{E}(Y_i^2 + Y_{i+1}^2 - 2 * Yi * Y_{i+1})=\frac{1}{2} * \mathbb{E}(Y_i^2) + \mathbb{E}(Y_{i+1}^2) - 2 * \mathbb{E}(Y_i)\mathbb{E}(Y_i + 1))$ car Y_i et Y_{i+1} sont independants or :

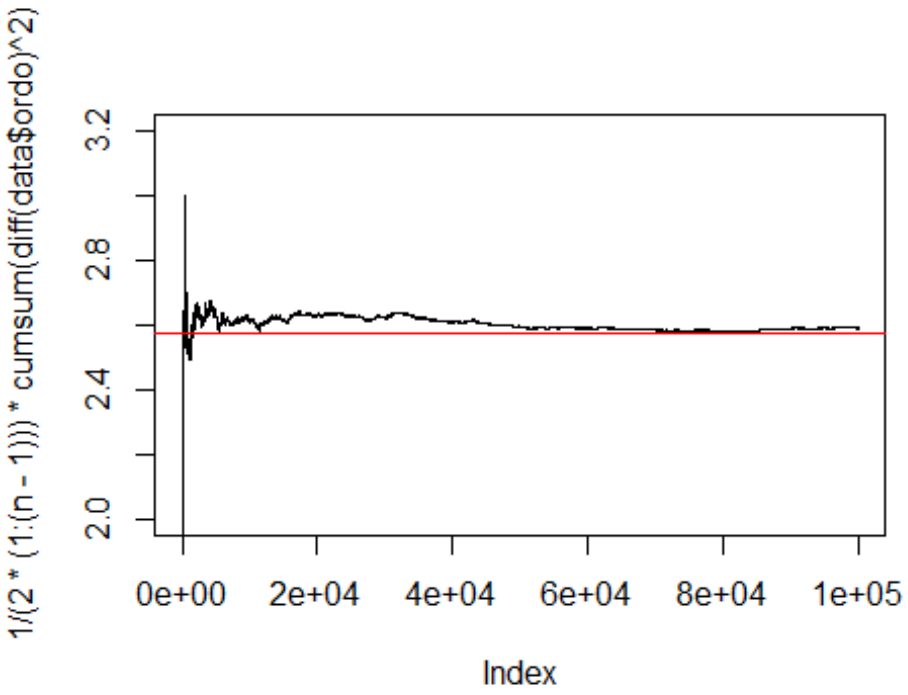
$\mathbb{E}(Yi^2)=\mathbb{E}(Y_{i+1}^2)$ et $\mathbb{E}(Yi)=\mathbb{E}(Y_{i+1})$

donc $\mathbb{E}(Zi)=1/2 * (2 * (\mathbb{E}(Yi^2) - \mathbb{E}(Yi)^2))$ = $Var(Y_i)$

L'estimateur de Rice $\hat{\sigma}_n^2$ est donc un estimateur sans biais de la variance des Y_i. Remarque : lien entre la variance de Y_i et celle de ξ_i : $Var(Y_i)=Var(r(X_i))+\sigma^2$ par indépendance de ξ_i et X_i

Précision de l'estimateur

```
plot(1/(2*(1:(n-
1)))*cumsum(diff(data$ordo)^2),xlim=c(1,1e5),ylim=c(2,3.2),type='l')

abline(h=var(data$ordo),col="red")
```



Le biais de l'estimateur est nul. Concernant la variance : d'après le TCL on peut construire un intervalle de confiance pour $\mathbb{E}(Z_i)= Var(Y_i)$ de la forme (au risque α) :

$[\hat{\sigma}_n^2 - q_\alpha * \frac{s}{\sqrt{n}}, \hat{\sigma}_n^2 + q_\alpha * \frac{s}{\sqrt{n}}]$ avec s=$\sqrt{Var(Z_i)}$

```
min(data$ordo)
```

```
## [1] -6.650033
```

```
max(data$ordo)
```

```
## [1] 7.752179
```

or $Var(Zi)$=$Var(\frac{1}{2} * (Y_{i+1} - Y_i)^2)$=$\frac{1}{4} * Var((Yi + 1 - Yi)^2)$ <= $\frac{1}{4}\mathbb{E}((Y_{i+1} - Y_i)^2)$ or comme Yi appartient à [-10,10] (cf ci-dessus), $|Y_{i+1} - Y_i|$ inférieur à 20 donc $Var(Z_i)$ inferieur

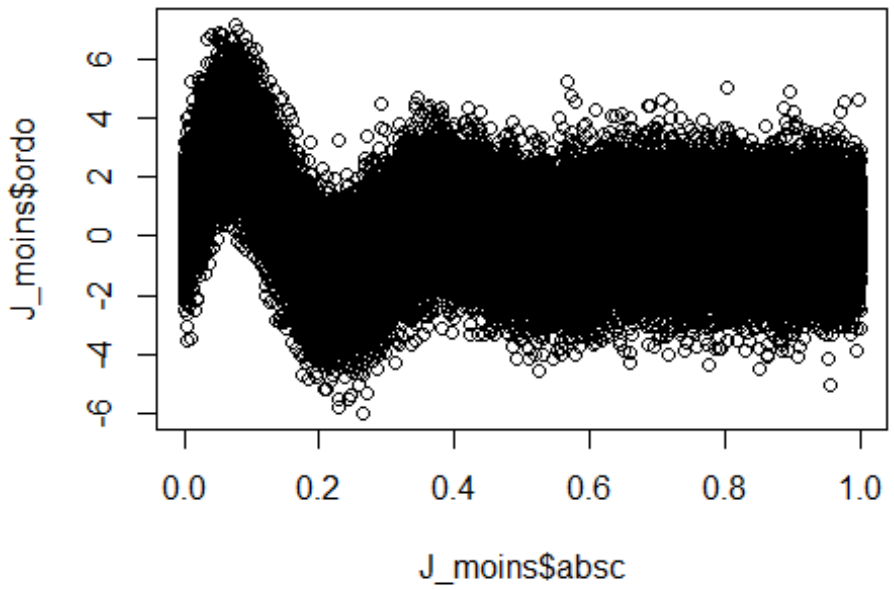
à $\frac{1}{4} * 20^2 = 100$ donc s inférieur à 10.

```
#application


q_99=qnorm(0.995)
majorant_s=10


intervalle=c(sigma_hat(data$ordo,n)-
```

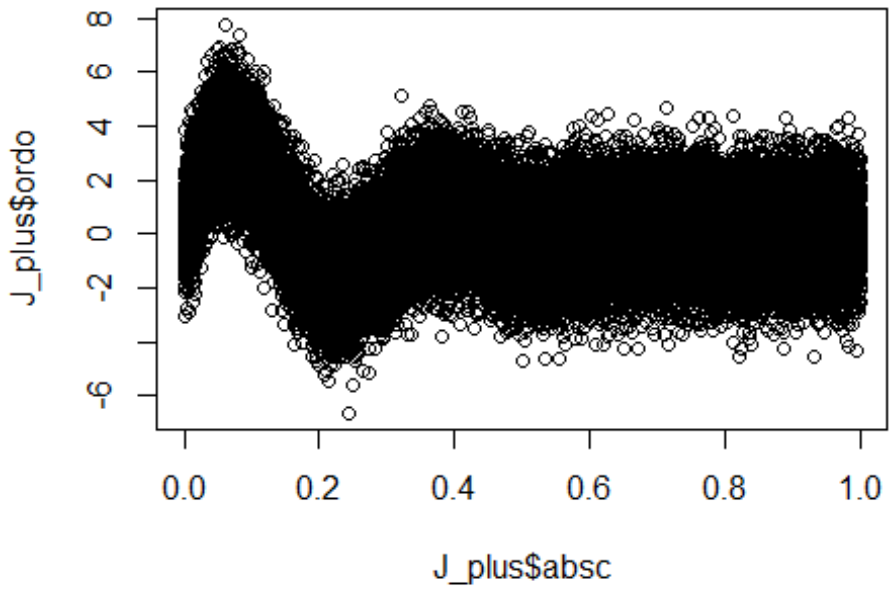
```
q_99*majorant_s/sqrt(n),sigma_hat(data$ordo,n)+q_99*majorant_s/sqrt(n))
print(min(intervalle))

## [1] 2.50833

print(max(intervalle))

## [1] 2.67124

print(var(data$ordo))

## [1] 2.575987

print(var(data$ordo)>=min(intervalle) & var(data$ordo)<=max(intervalle))

## [1] TRUE
```

Distribution de $\widetilde{Y}_i$

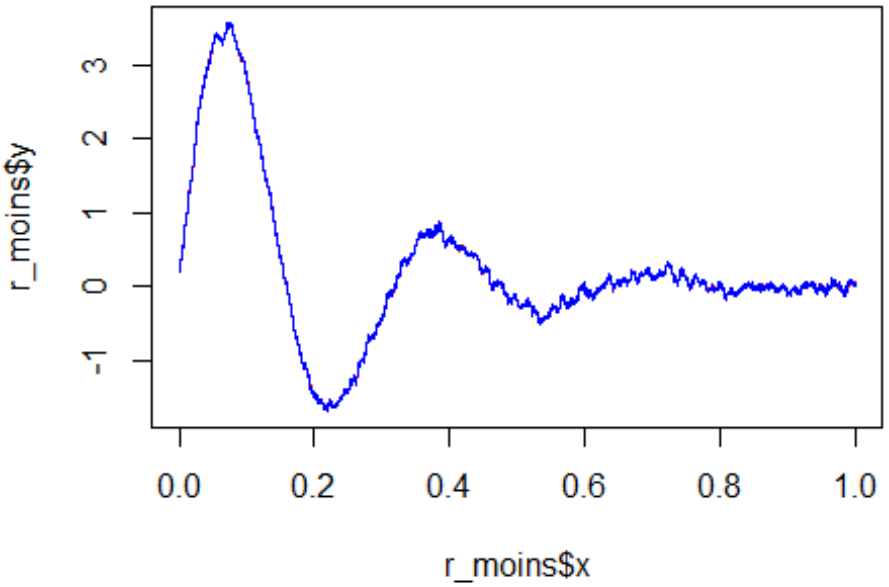
```
#par(mfrow=c(2,1))

J_moins=data[1:5e4,]

J_plus=data[(5e4+1):1e5,]


plot(J_moins$absc,J_moins$ordo)
```



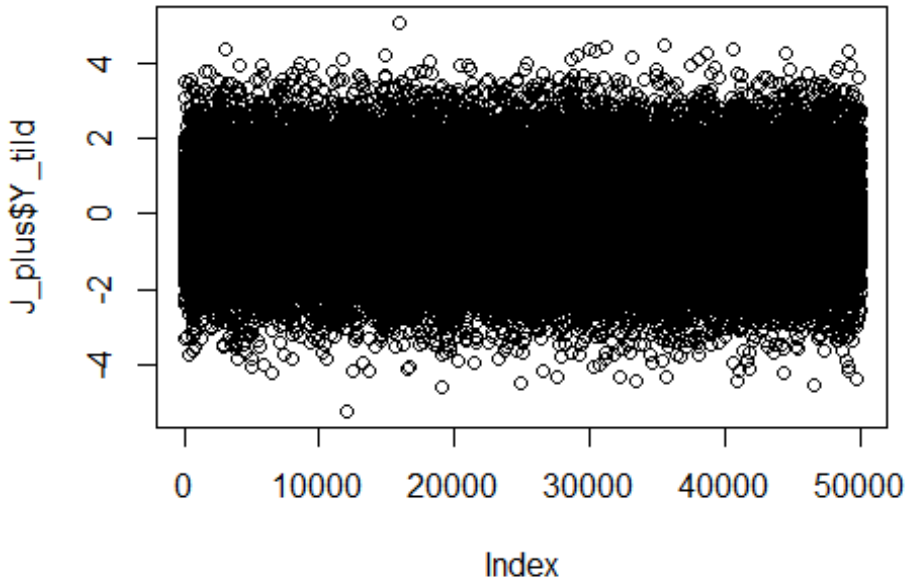
```r
plot(J_plus$absc,J_plus$ordo)
```



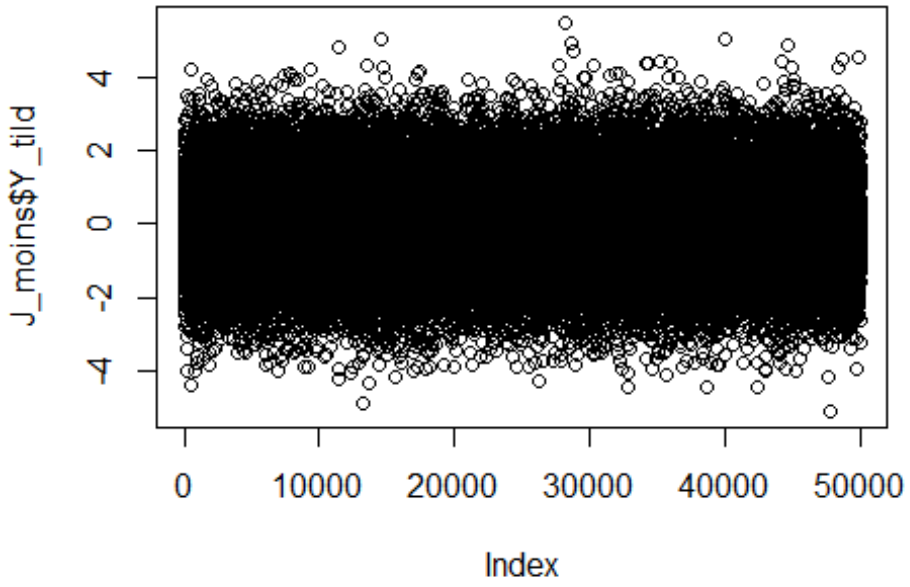

```r
?ksmooth

## starting httpd help server ...

##  done

#on utilise la fonction ksmooth qui permet de faire des predictions
r_moins=ksmooth(x=J_moins$absc,y=J_moins$ordo,bandwidth=h_opt)
predict_r_moins=ksmooth(x=J_moins$absc,y=J_moins$ordo,bandwidth=h_opt,n.points=length(J_plus$absc)
                        ,x.points=J_plus$absc)


plot(r_moins,col="red",type="l",xlim=c(0,1))
lines(predict_r_moins,col="blue",type="l")
```




```
#les deux distributions semblent être extrememement proches
#r_moins$y[1:100]-predict_r_moins$y[1:100]


#normalement la distribution de Y_tild devrait être celle de Xsi_i_plus
#Attention dans l'operation a ne pas oublier de trier les valeurs par X
croissant dans J_plus car ksmooth retourne des valeurs de (x,y) avec x
croissant or les Xi sont aléatoires et uniformes donc desordonnes !

J_plus$Y_tild=J_plus[order(J_plus$absc),'ordo']-predict_r_moins$y

#str(predict_r_moins$y)
plot(J_plus$Y_tild)
```




```
mean(J_plus$Y_tild)

## [1] -0.003873694

var(J_plus$Y_tild)

## [1] 1.43351

#generons egalement les Xsi_moins
r_plus=ksmooth(x=J_plus$absc,y=J_plus$ordo,bandwidth=h_opt)
predict_r_plus=ksmooth(x=J_plus$absc,y=J_plus$ordo,bandwidth=h_opt,n.points=l
ength(J_moins$absc),x.points=J_moins$absc)

J_moins$Y_tild=J_moins[order(J_moins$absc),"ordo"]-predict_r_plus$y
plot(J_moins$Y_tild)
```




```
mean(J_moins$Y_tild)
```

```
## [1] 0.003518048
```

```
var(J_moins$Y_tild)
```

```
## [1] 1.4492
```

```
mean(c(J_moins$Y_tild,J_plus$Y_tild))
```

```
## [1] -0.0001778229
```

```
var(c(J_moins$Y_tild,J_plus$Y_tild))
```

```
## [1] 1.441354
```

Les distributions de $\widetilde{Y}_i$ devrait approximativement être celle de ξ_i

Estimation de $\mu(x)$

pour reconstituer $\mu(x)$, on utilise le lissage par noyau pour une densite comme pour la densite g(x)

```
#ajoutons les Xsi_plus et Xsi_moins pour avoir plus de points
Xsi_i=c(J_moins$Y_tild,J_plus$Y_tild)
mean(Xsi_i)#environ 0
```

```
## [1] -0.0001778229
```

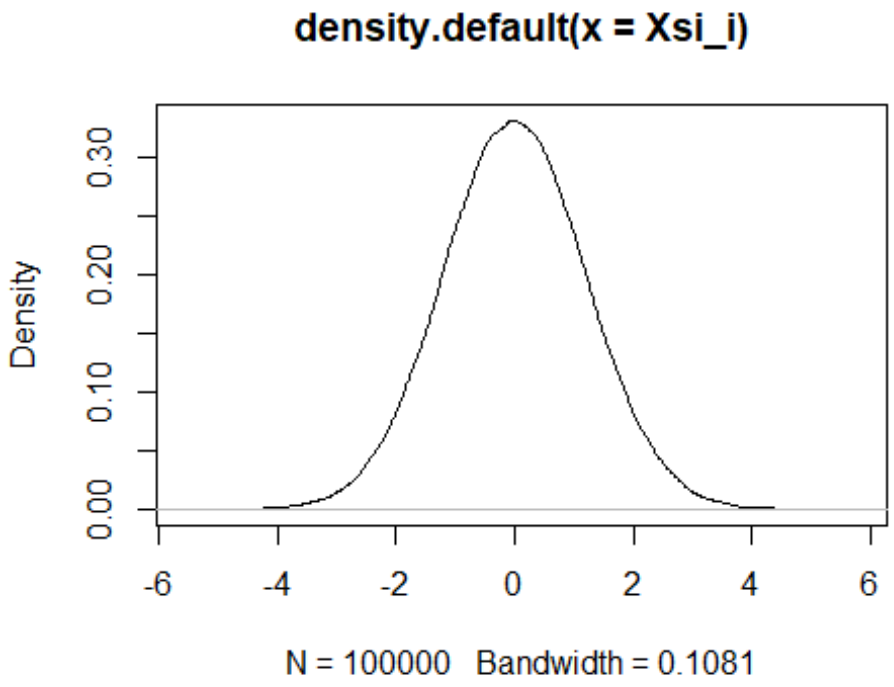
```r
var(Xsi_i)#1.44
```

```
## [1] 1.441354
```

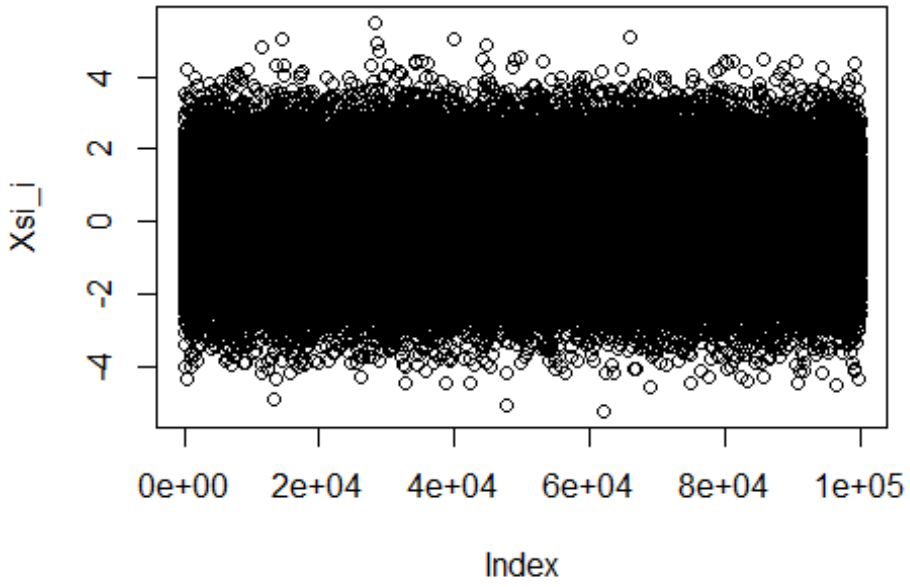
```r
#on peut utiliser l'estimateur de Rice pour estimer sigma² :
sigma_hat(J_moins$Y_tild,length(J_moins$Y_tild))
```

```
## [1] 1.436067
```

```r
plot(density(Xsi_i),type='l')
```




```r
plot(Xsi_i)
```



A première vue la densité ressemble à une gaussienne centrée en 0.

Remarque : σ^2 vaudrait donc 1.44. Or si on considère les Xi non déterministes, d'après la relation Var(Y)=Var (r(X))+ σ^2donc

Var(r(X))=Var(Y) - σ^2

```
Var_r_X=var(data$ordo)-var(Xsi_i)
Var_r_X#1.13
```

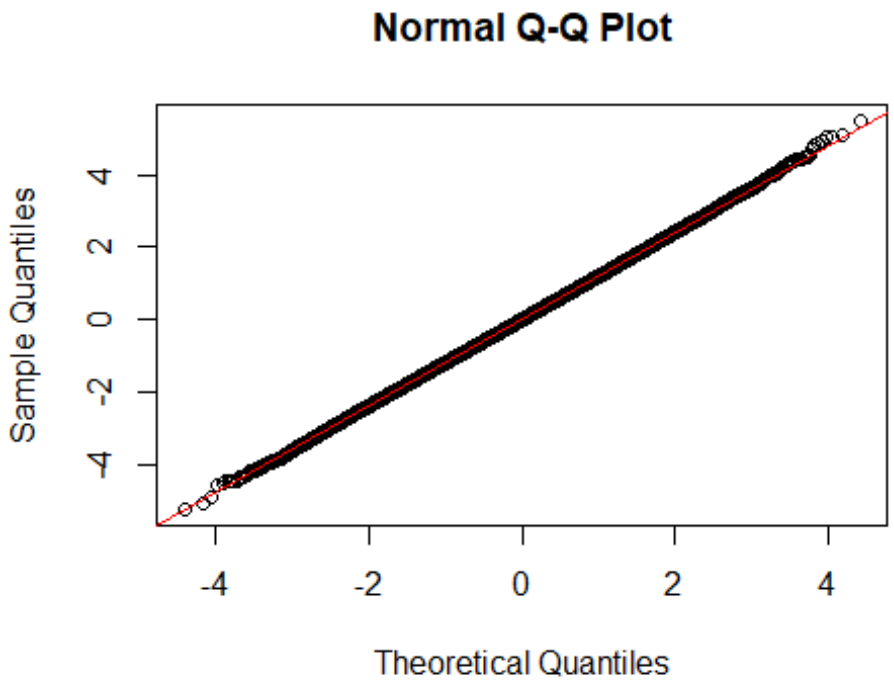
```
## [1] 1.134633
```

question sur la gaussianité

qqplot

test de gaussianité

```
#la densite dessinee a la question precedente

qqnorm(Xsi_i)
qqline(Xsi_i,col=2)
```




```
#Test de normalite pour les Xsi_i : on peut  utiliser un test de  Shapiro-
Wulk
#Shapiro-Wulk test :
#Ho= la série statistique suit une distribution normale
#H1=la serie statistique ne suit pas une distribution gaussienne

?shapiro.test
shapiro.test(sample(Xsi_i,5000))

##
##  Shapiro-Wilk normality test
##
## data:  sample(Xsi_i, 5000)
## W = 0.99973, p-value = 0.7856

#p-value >> 5%, alors le test de  Shapiro  montre que les Xsi sont  gaussiens

#Remarque :  test de Kolmogorov Smirnov :
#Ho : la distribution est Gaussienne
#ks.test(Xsi_i,"pnorm")

#p-value << 5 % on rejette Ho

#Il semblerait donc qu'un test  rejette la gaussianité distribution
...bizzare...
```