

Régression non-paramétrique

Philippe Real

6/8/2019

#markdown::render("MNP_DMFinal.Rmd") # html_document: default # pdf_document: default

```
#for manipulate data (transform to dataframe)
install.packages("tidyverse")
install.packages("tibble")
install.packages("sm")

install.packages("KernSmooth")
install.packages("np")
install.packages("stats")
install.packages("ggplot2")
```

1. Etude de la densité g des X

1.0 Lecture des données et premières analyses

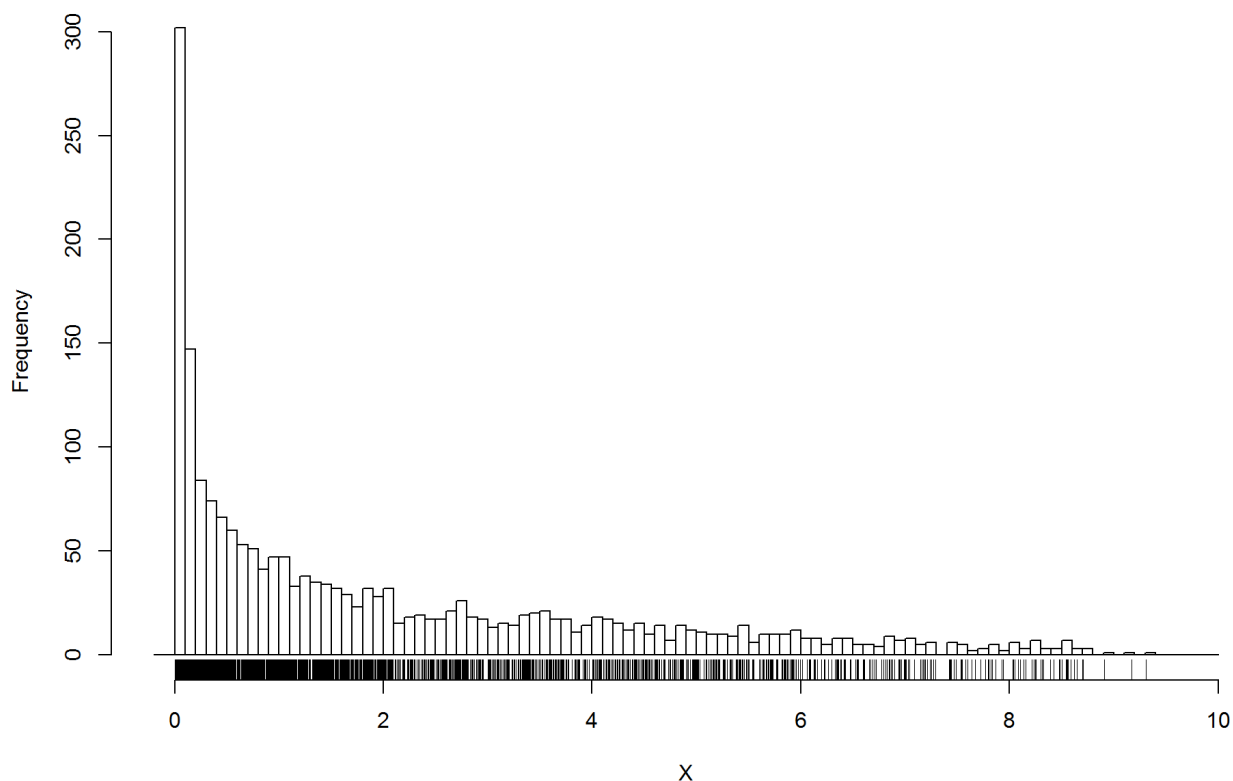
```
d1 = read.csv("Data1.csv")
summary(d1)
```

```
##      X.1          X          Y1
## Min.   : 1.0    Min.   :0.000005 Min.   : -0.2244
## 1st Qu.: 500.8  1st Qu.:0.258074 1st Qu.: 0.2619
## Median :1000.5  Median :1.192414 Median : 0.4303
## Mean   :1000.5  Mean   :2.029447 Mean   : 0.5112
## 3rd Qu.:1500.2  3rd Qu.:3.318174 3rd Qu.: 0.6735
## Max.   :2000.0  Max.   :9.308684 Max.   : 3.1263
```

Pour avoir une idée de la densité de X on peut tracer son histogramme.

```
Bk <- seq(-0.2,10, by = 0.1)
hist(X, breaks = Bk)
rug(X)
```

Histogram of X

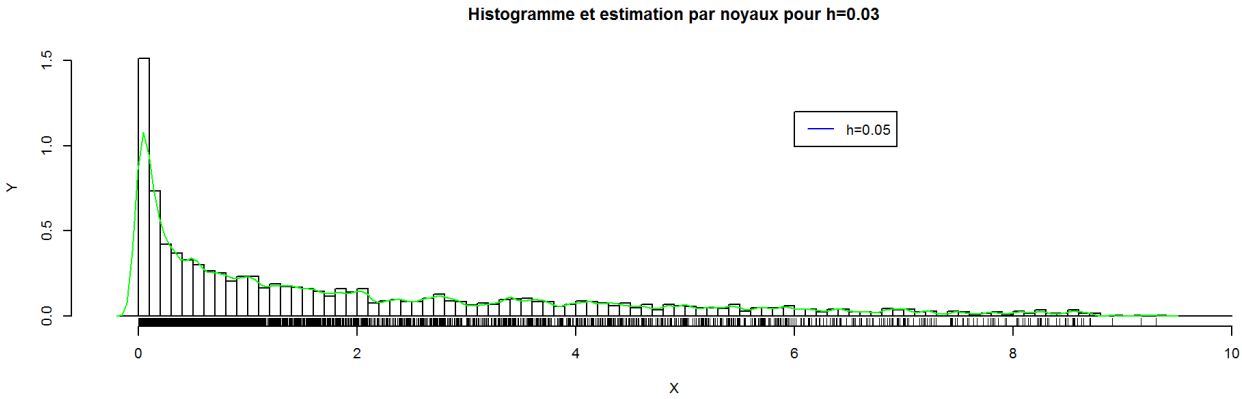
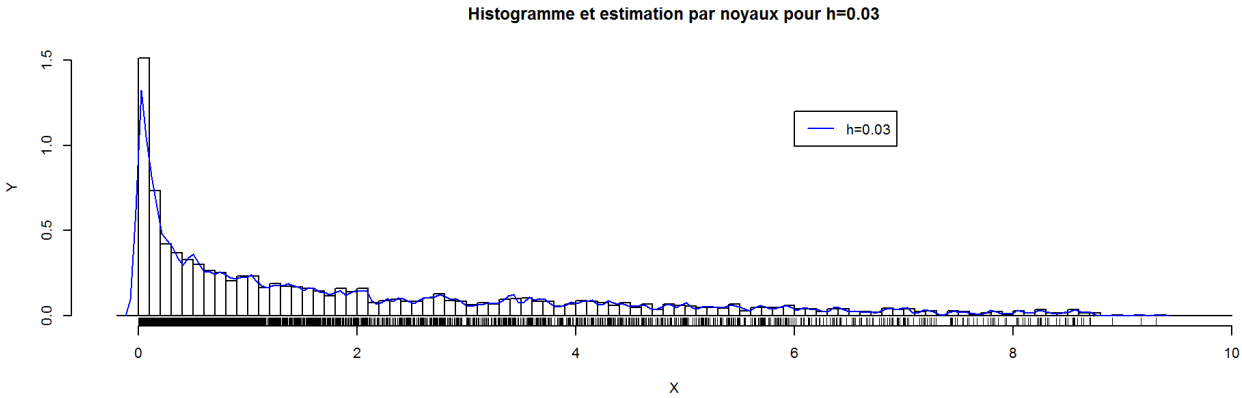
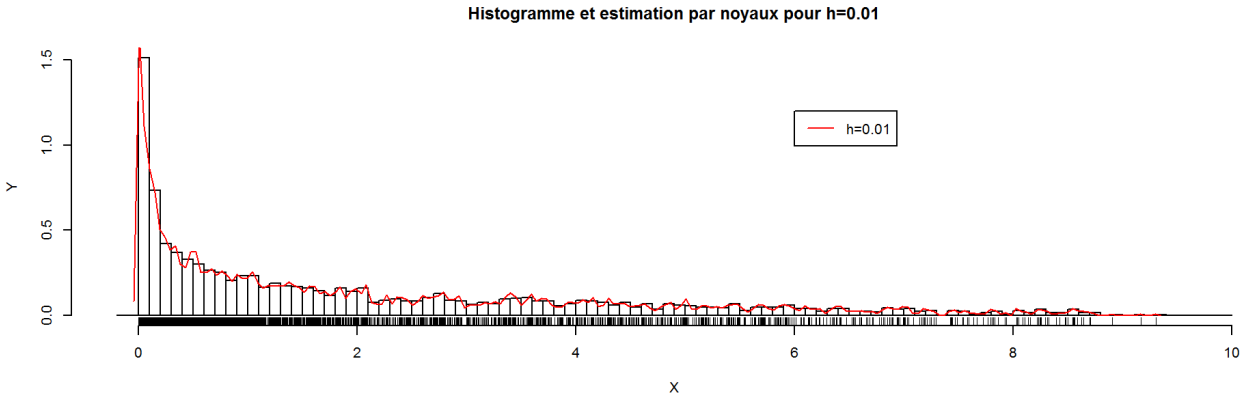


1.1 Estimateur non-paramétrique de g(x)

On peut utiliser la fonction `bkde` du package `KernSmooth` qui estime la densité par la méthode des noyaux. On prend comme noyau le noyau normal. Ce choix peut sembler arbitraire, mais on a vu que ce n'est pas le choix du noyau qui est le plus important dans l'estimation de la densité. On calcule cet estimateur de la densité pour différentes largeurs de fenêtre: h (bandwidth) Et on va déterminer de manière empirique une valeur de h qui semble adapté.

Petites valeurs de la fenêtre h : 0.01 / 0.03

bandwidth h	error quadratic
0.01	8.4734061
0.03	8.4759164
0.05	8.4850157

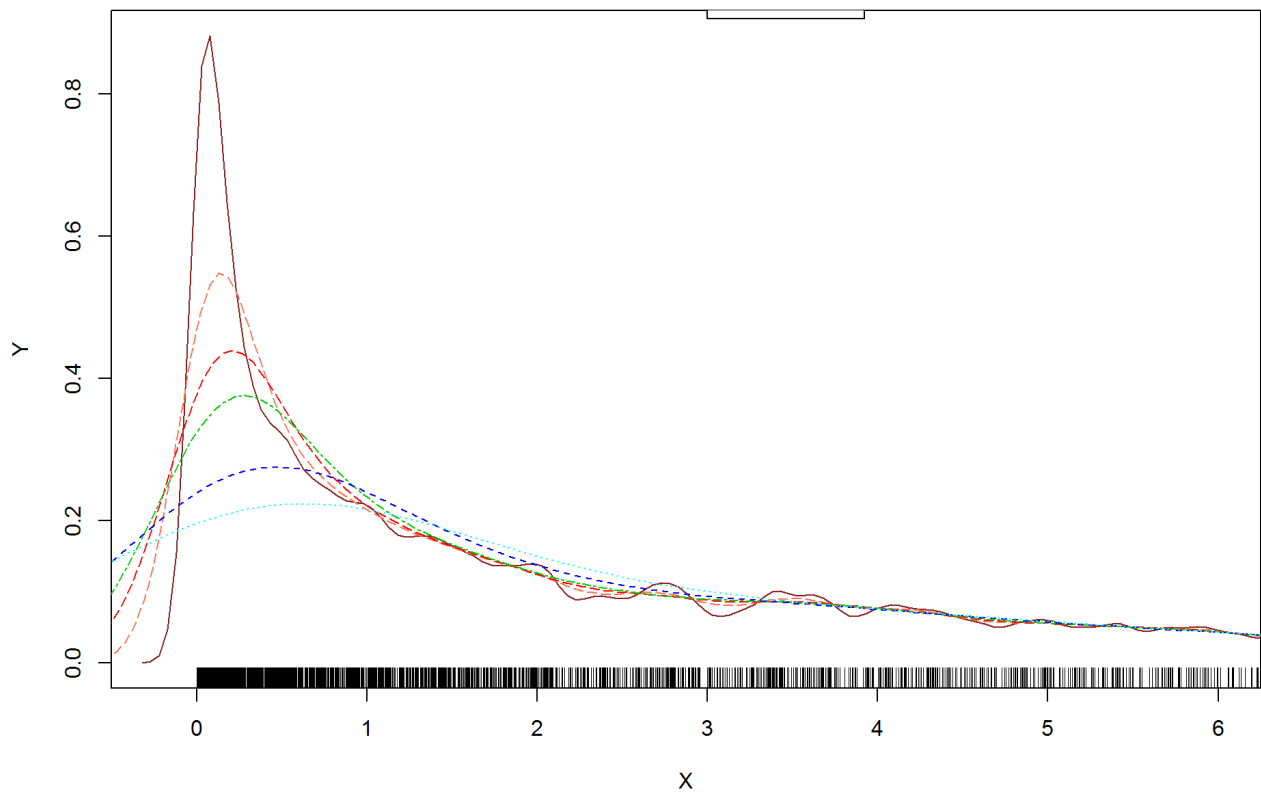


On remarque de fortes oscillations pour des h entre 0.01 et 0.05.

h : 0.08 à 1

bandwidth h	error quadratic
0.08	8.5070249
0.2	8.5329652
0.4	8.5692634
0.7	8.617531
1	8.6528775

Estimation de la densité par noyaux pour différents h



A partir de $h = 0.2$ l'approximation est plus régulière.

Raison pour laquelle ce choix est important et ce qui se produit si h est mal choisi

- Si h est trop grand (courbes bleues) noyau trop régularisant : trop de biais
- Si h est trop petit (courbes marron du graphique ci-dessus) : trop oscillant, trop de variance.

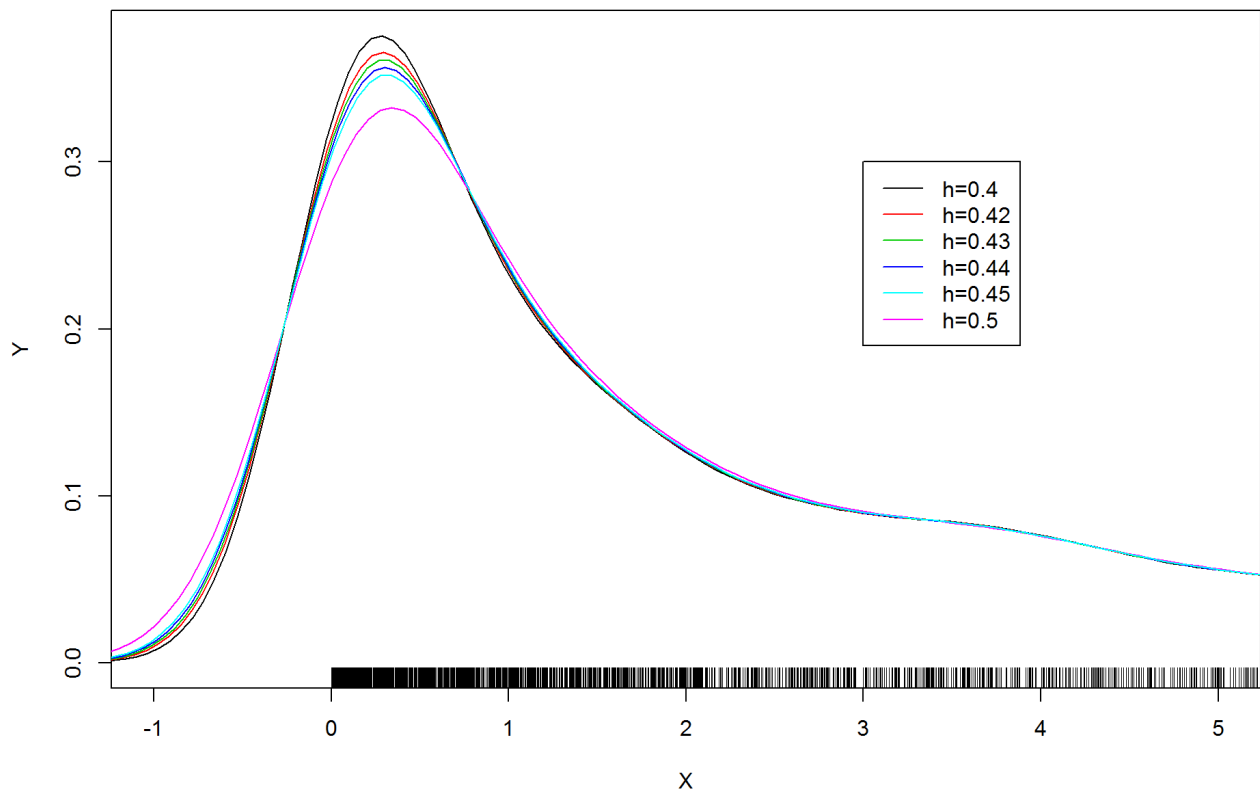
Si on regarde le critère de l'erreur quadratique un h adapté serait compris entre 0.4 et 0.5 Par la suite on va utiliser d'autres critères pour définir un h optimal.

Raffinement pour h : 0.4 - 0.5

bandwidth h	error quadratic
0.4	8.5692634
0.42	8.5731981
0.43	8.5751122
0.44	8.5769881
0.45	8.5788314
0.5	8.5875782

```
## Warning in rug(X): some values will be clipped
```

Estimation de la densité par noyaux pour différents h entre: 0.4 et 0.5



Les estimations de la densité $g(x)$ de X pour des valeurs de h comprise entre 0.4 et 0.5 sont très proches. A ce niveau il est difficile de déterminer empiriquement le h optimal.

1.2 Détermination d'un h optimal

Représentation graphique de l'estimation par noyau de la densité $g(x)$, où $h.n$ est la fenêtre donnée par validation croisée ou par une autre méthode que l'on précisera.

1.2.1 Validation croisée pour la densité,

Utilisation de la fonction `bw.ucv` library `stats`

```
h<-bw.ucv(X)
h
```

```
## [1] 0.05675136
```

1.2.2 Règle Silverman

En appliquant la règle de Silverman, on obtient le h suivant:

```
n<-length(X)
hh<-1.06*sqrt(var(X))*n**(-1/5)
hh
```

```
## [1] 0.505695
```

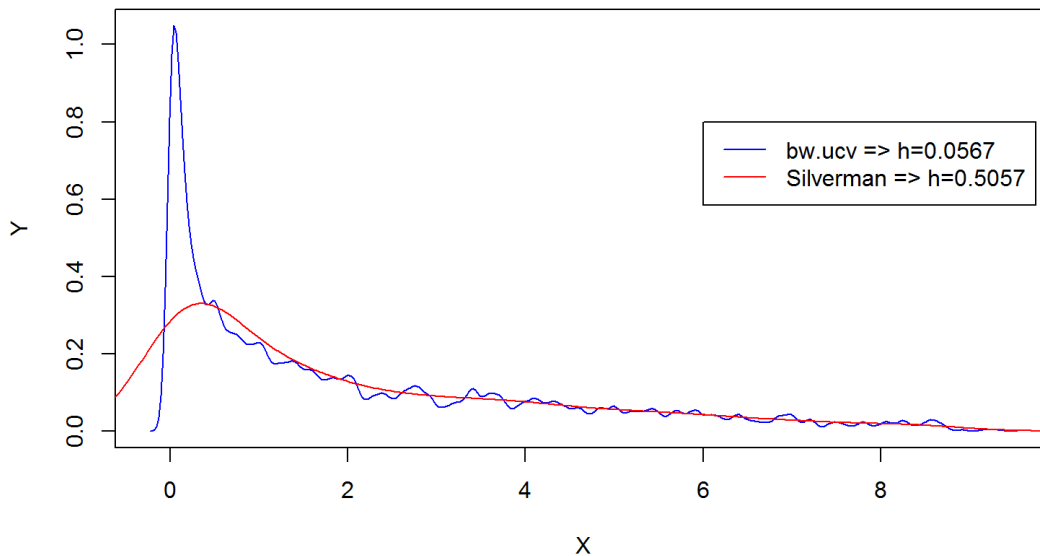
On obtient le même résultat avec la fonction: `bw.nrd`

Fonction `bw.nrd`

```
nrd <- bw.nrd(x = X)
nrd
```

```
## [1] 0.505695
```

Densité par noyaux pour h obtenues par différentes méthodes.



1.2.3 Méthodes alternatives

Fonction density

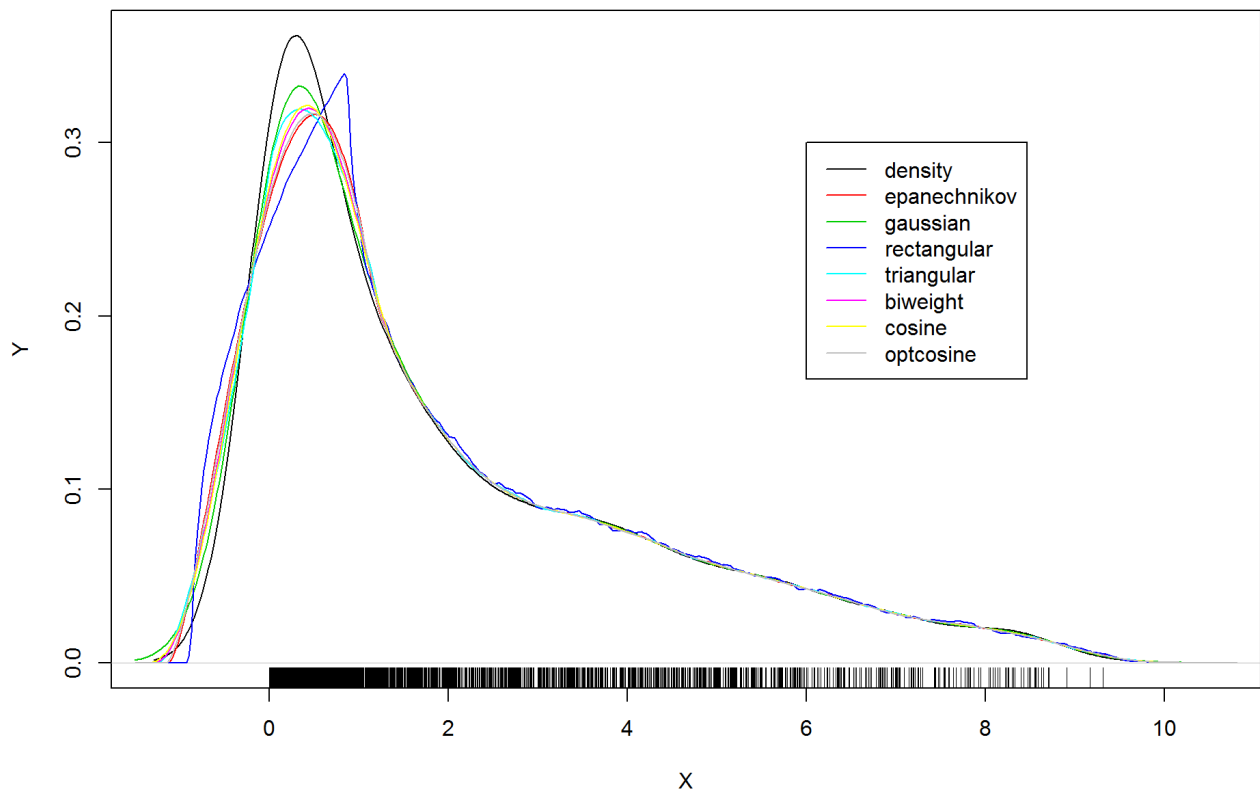
On peut regarder le résultat de la fonction density du package RSmooth qui teste différents noyaux et renvoie un h optimal

```
density(X)
```

```
##  
## Call:  
## density.default(x = X)  
##  
## Data: X (2000 obs.); Bandwidth 'bw' = 0.4294  
##  
##      x          y  
## Min.   :-1.288   Min.   :0.0000075  
## 1st Qu.: 1.683   1st Qu.:0.0199479  
## Median : 4.654   Median :0.0526611  
## Mean   : 4.654   Mean   :0.0840423  
## 3rd Qu.: 7.626   3rd Qu.:0.1028372  
## Max.   :10.597   Max.   :0.3618178
```

Remarque: La fonction density calcul automatiquement le h optimal. On peut regarder les différentes estimations de la densité issues de noyaux différents. Comme énoncé plus haut les estimations sont proches (excepté pour le noyau rectangulaire)

Estimation de la densité par la fonction density pour différents noyaux où $h = 0.4294$



Fonction dpik

méthode du package Kersmooth: Select a Bandwidth for Kernel Density Estimation

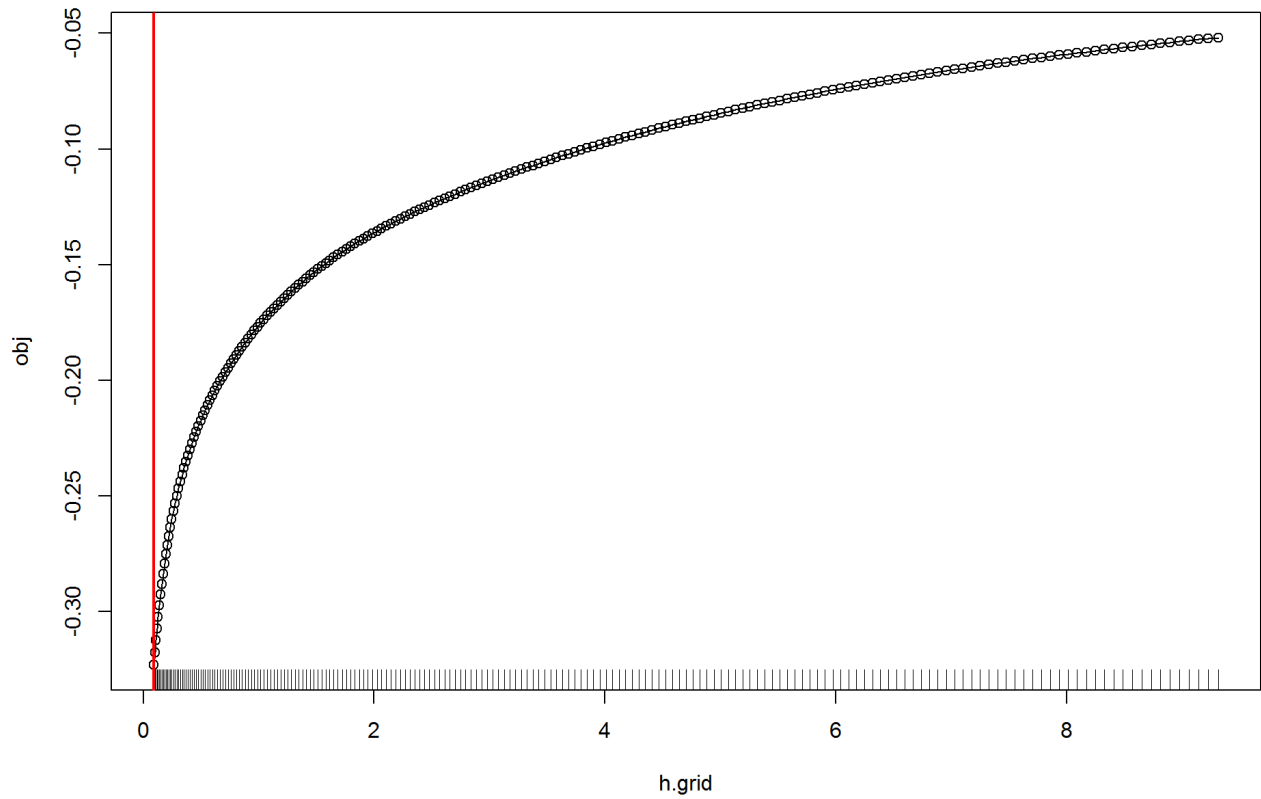
```
hdpik <- dpik(X)
hdpik
```

```
## [1] 0.1439489
```

Fonction ucv

La fenêtre h est obtenu par (UCV) de Least Squares Cross-Validation curve (LSCV) et h obtenu (UCV)

Least Squares Cross-Validation curve (LSCV) et h obtenu (UCV)

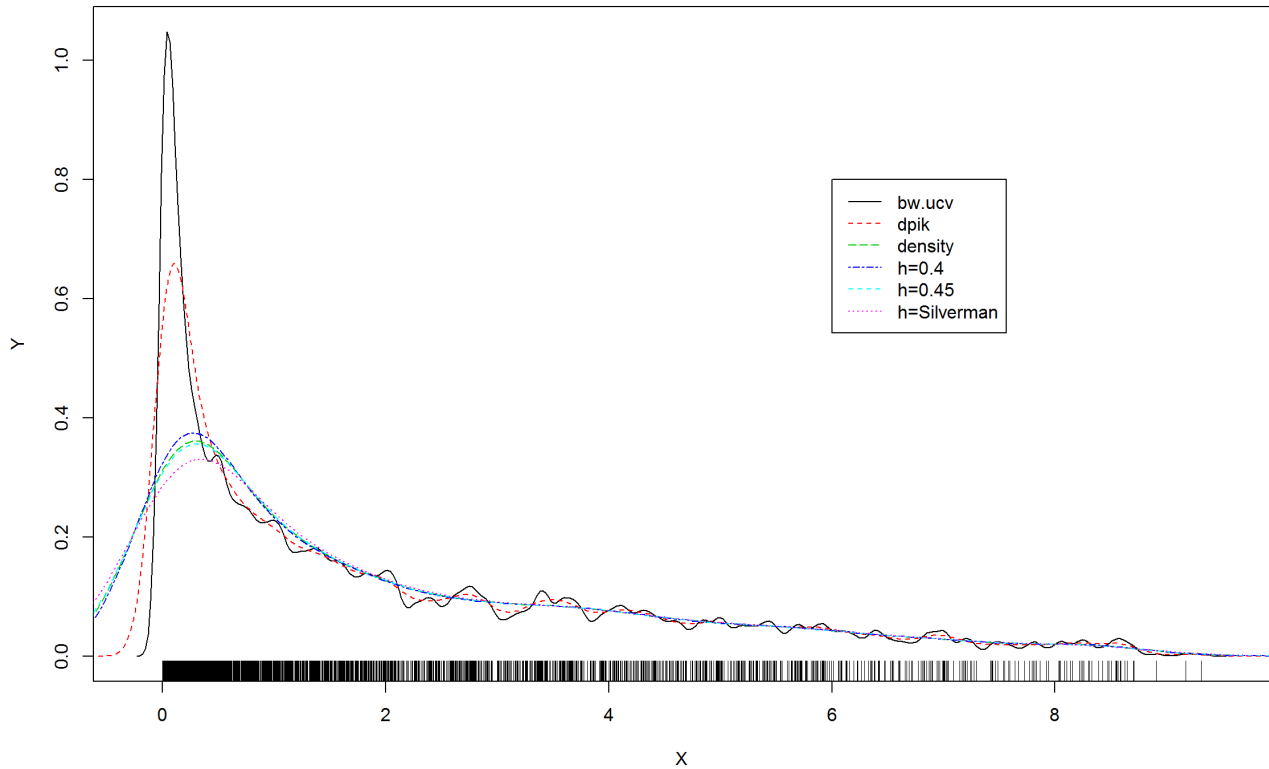


```
## [1] 0.09308678
```

Résumé des résultats

Méthode	valeur de h
bw.ucv	0.0567514
dpik	0.1439489
density	0.4294
ESM	entre 0.4 et 0.5
Regle Silverman	0.505695

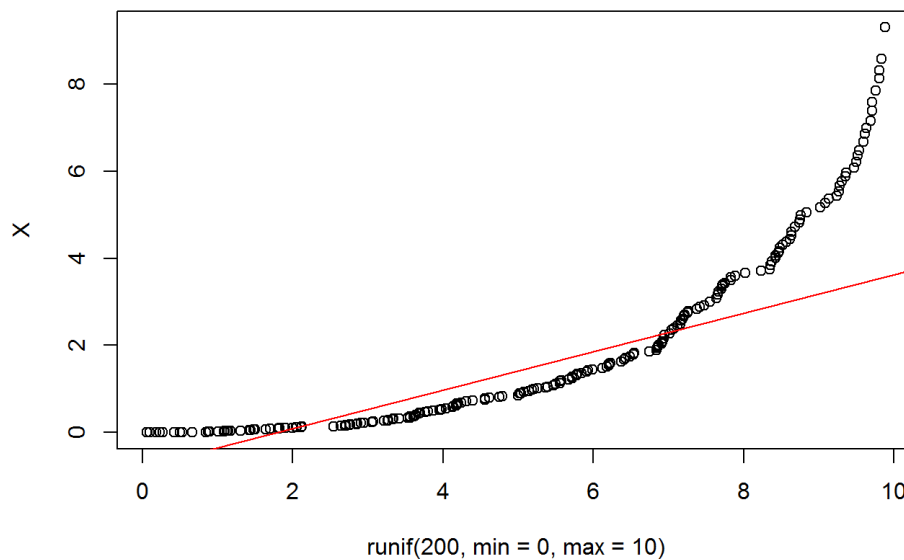
Estimation de la densité par noyaux pour différents h



1.3 QQPlot - g(x) densité Uniforme ?

Implementation d'un QQ-plot pour vérifier empiriquement l'hypothèse g(x) suit (ou pas) une densité Uniforme $U=1/10$ sur $[0,10]$

Q-Q plot pour la loi Uniforme $_{U=1/10}$



A la vue du graphique QQPlot par rapport à la loi uniforme ($U=1/10$) l'hypothèse selon laquelle g est uniforme n'est pas raisonnable. Cependant entre les valeurs en abscisse $[2,6]$ l'hypothèse semble plus crédible. On pourrait diviser l'espace en 3 parties: $[0, 2]$ $[2, 6]$ $[6, 10]$

1.4 Zone de l'espace où l'estimation de r sera plus précise

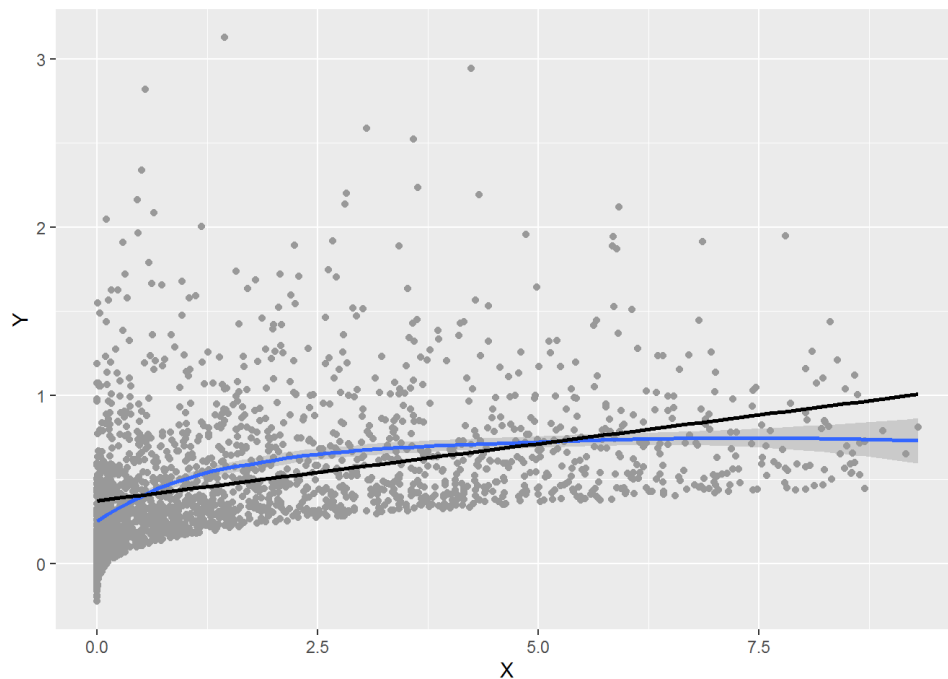
Plus de précision où les données ne sont pas trop dispersées et où la densité est importante. Donc plus de précision dans l'intervalle $[0+\text{dela}, 5]$ La précision diminue à droite... Dans un voisinage de 0 l'estimation n'est pas précise non plus: difficulté d'estimer où $g(x) = 0$

2. Reconstruction de r(x)

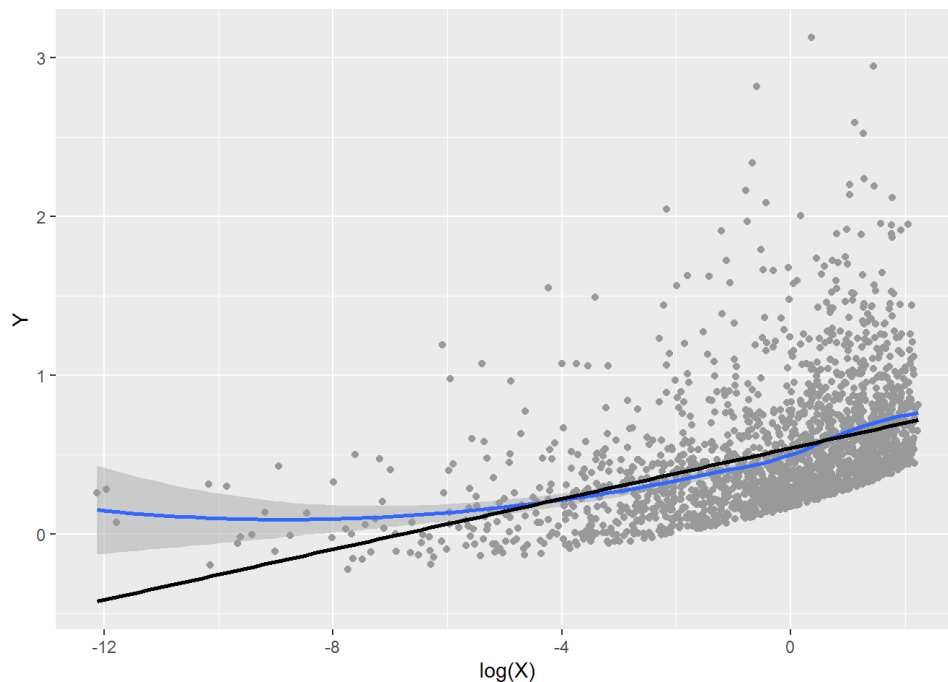
On est dans le cadre de l'estimation non paramétrique, on reprend les hypothèses classiques On utilise les données de Data1, (X,Y)

2.1 La fonction r peut elle être linéaire ?

On trace Y1 en fonction de X Sans transformation, à la vue du graphe, r ne peut être linéaire.



Maintenant On trace Y1 en fonction de $\log(X)$



Dans la région $[-5, 1]$ où l'on retrouve la quasi totalité de l'échantillon, la transformation $(\log(x))$ a permis de bien linéariser.

2.2. Construction d'un estimateur non-paramétrique de $r(x)$

Détermination de la fenêtre h

Validation croisée: visualisation de h en fonction des points de grid

h compris entre 0.125 et 0.135

Find optimum CV bandwidth, with sensible grid

On trouve un h optimum $h = 0.1200601$

- Test: choix de h local

```
h_CVopt <- h_CVloc(Xo,Yo,10)
if(length(which(is.na(h_CVopt$h_locG)))==0){h=h_CVopt$h_locG}else{h=h_CVopt$h_locG[-which(is.na(h_CVopt$h_locG))]}
summary(h)
```

```
##      Min.   1st Qu.   Median     Mean   3rd Qu.    Max.
## 0.005019 0.014504 0.036759 0.059364 0.078701 0.173141
```

- Test: choix de h_{global} , à l'aide de la fonction `dpill`

```
hDpill <- dpill(Xo,Yo)
hDpill
```

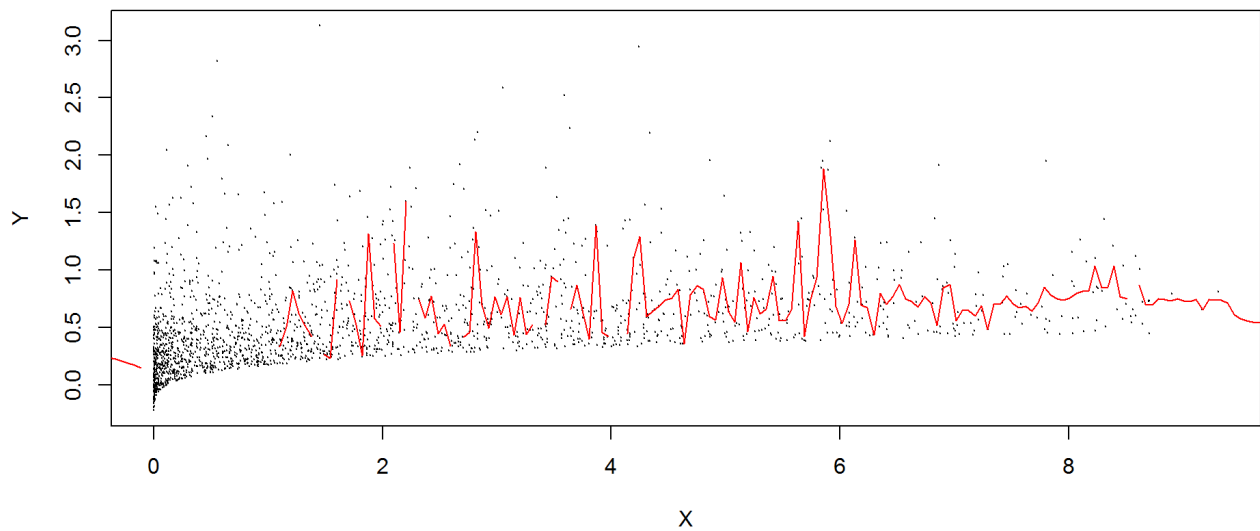
```
## [1] 0.2186849
```

```
hCV<-0.1200601
```

Estimateur de Nadaraya-Watson : avec un h_{local}

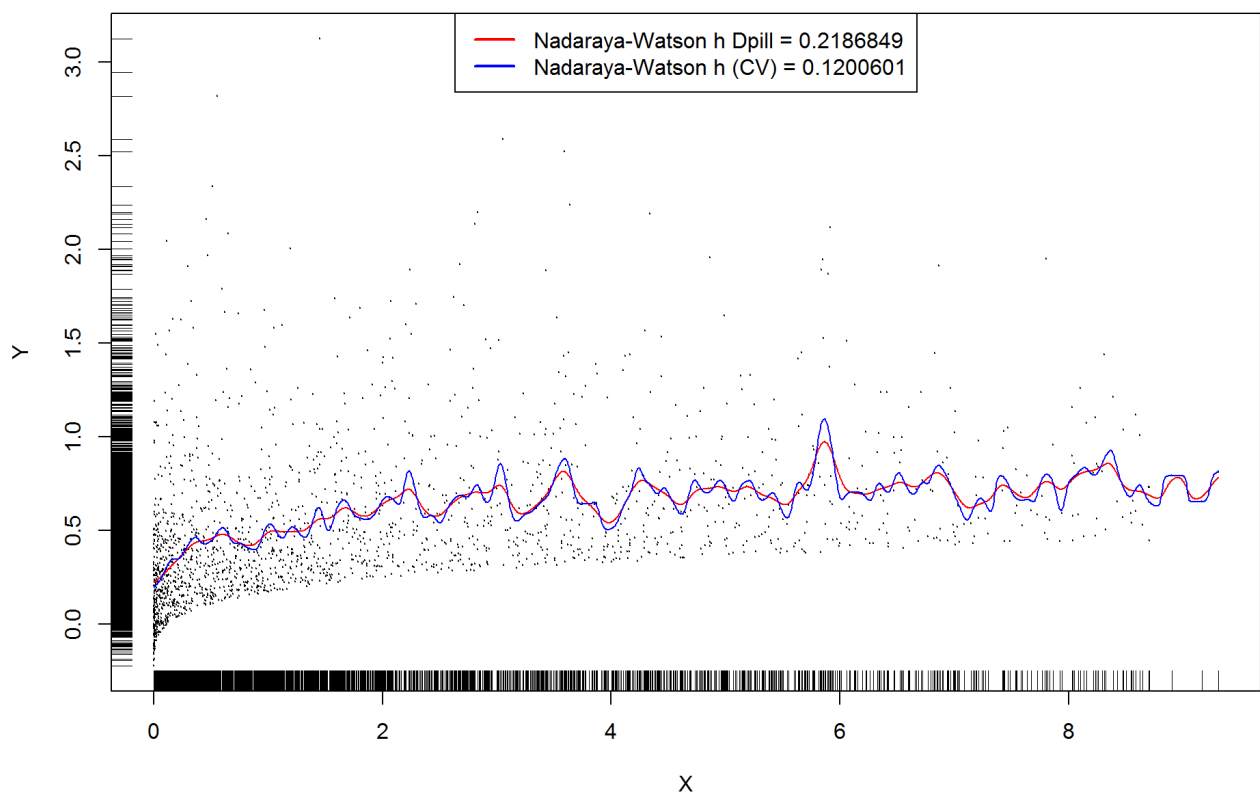
h_{local} calculé aux différents points de grid.

graphique pour : Estimateur de Nadaraya-Watson et h local



Estimateur de Nadaraya-Watson avec la librairie stat - fonction : ksmooth - $h = 0.2186849$ et $h = 0.1200601$

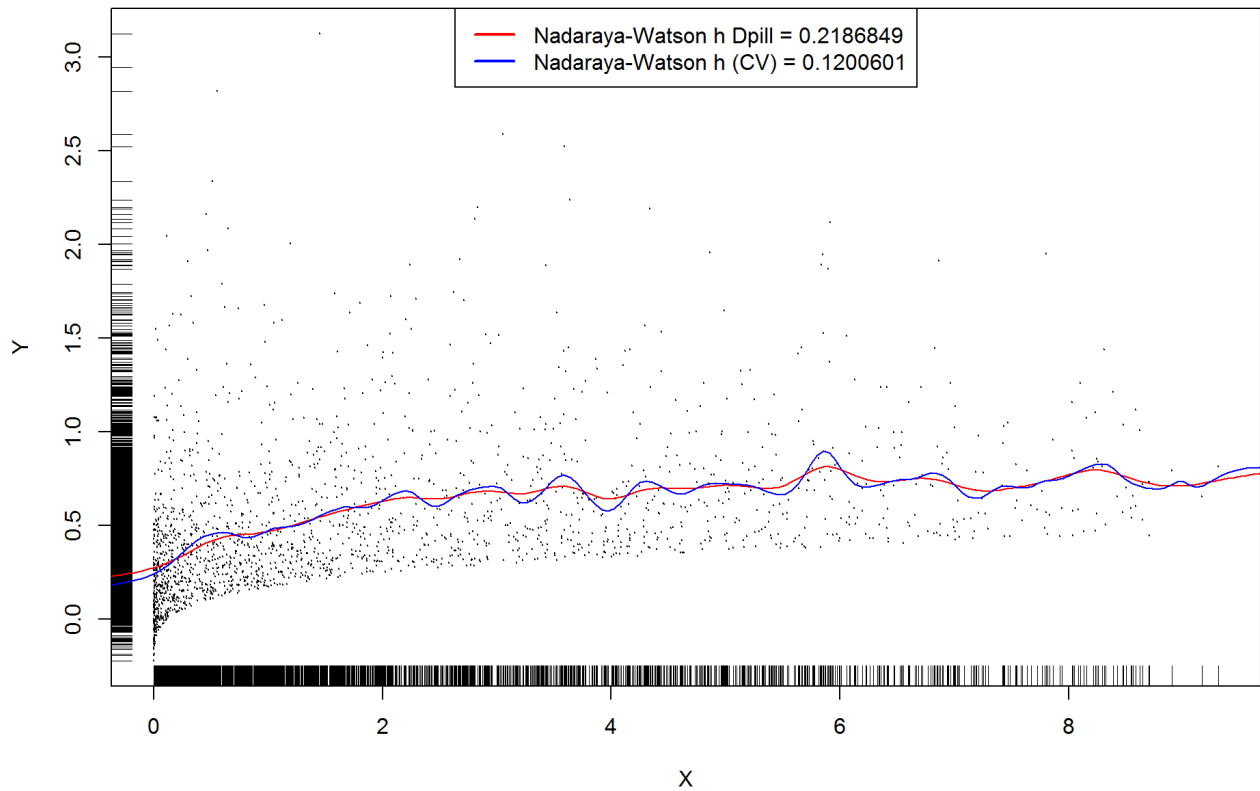
Nadaraya-Watson avec ksmooth et différents h



L'estimateur est sensible au choix de h . L'estimateur de Nadaraya-Watson est très oscillant par construction.

Estimateur de Nadaraya-Watson à partir de la fonction recodé NW - $h = 0.2186849$ et $h = \text{r hCV}$

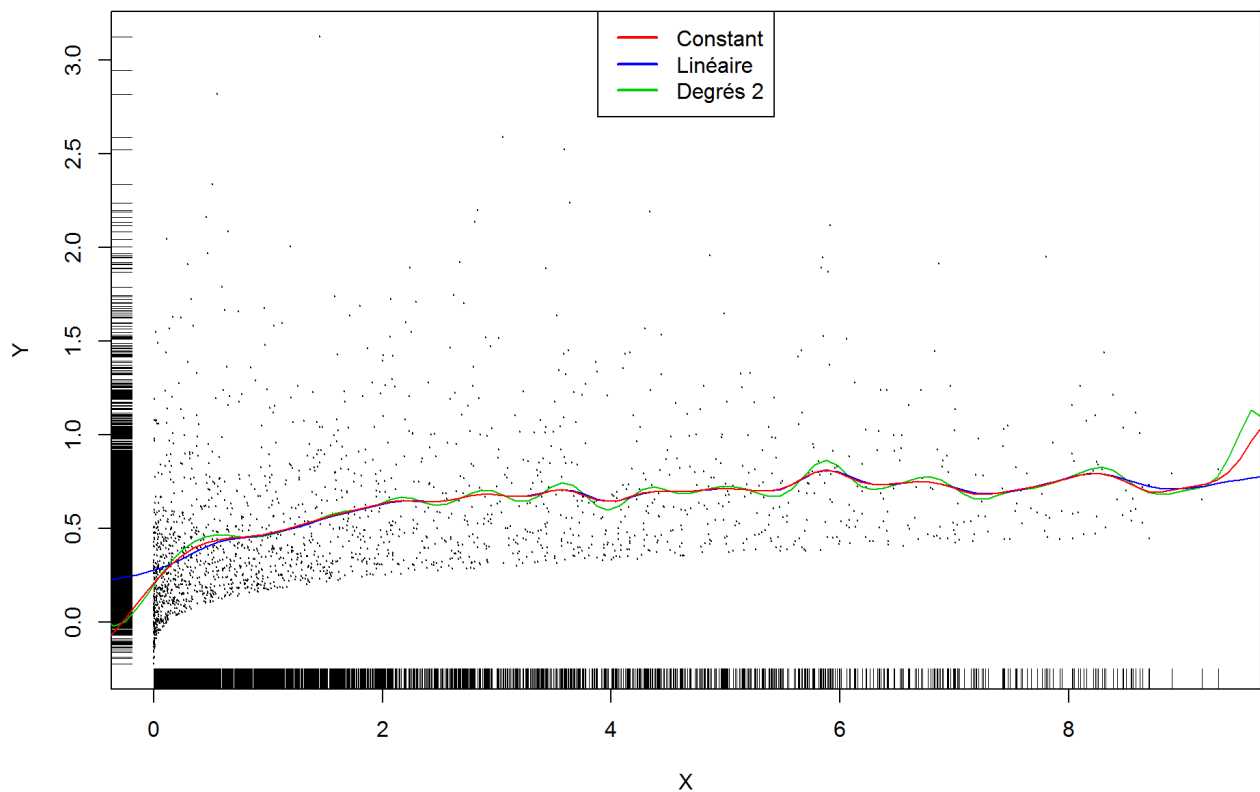
Nadaraya-Watson avec la fonction mNW et différents h



Estimateur par polynômes locaux

Utilisation de la fonction `locpoly` du package `Kernsmooth` avec $h = 0.2186849$ Attention à ordonner l'échantillon (en X) pour garder la cohérence des couples (X,Y), car sinon `KernSmooth` va réordonner les X sans tenir compte de la composante Y.

Regression par polynômes locaux avec locpoly



2.3. estimation de r en regressant Y1 sur log(X)

```
## [1] 0.649558
```

Find optimum CV bandwidth, with sensible grid

Estimation de la régression de Log(X) avec Nadaraya-Watson avec la librairie stat - fonction : ksmooth - h = 0.649558 et h = 0.4046381

```
ln_hCV<-0.4046381
d1o <- order(d1[, "X"])
Xo <- d1[d1o, "X"]
Yo <- d1[d1o, "Y1"]
xGrid <- seq(-12, 3, l = 200)

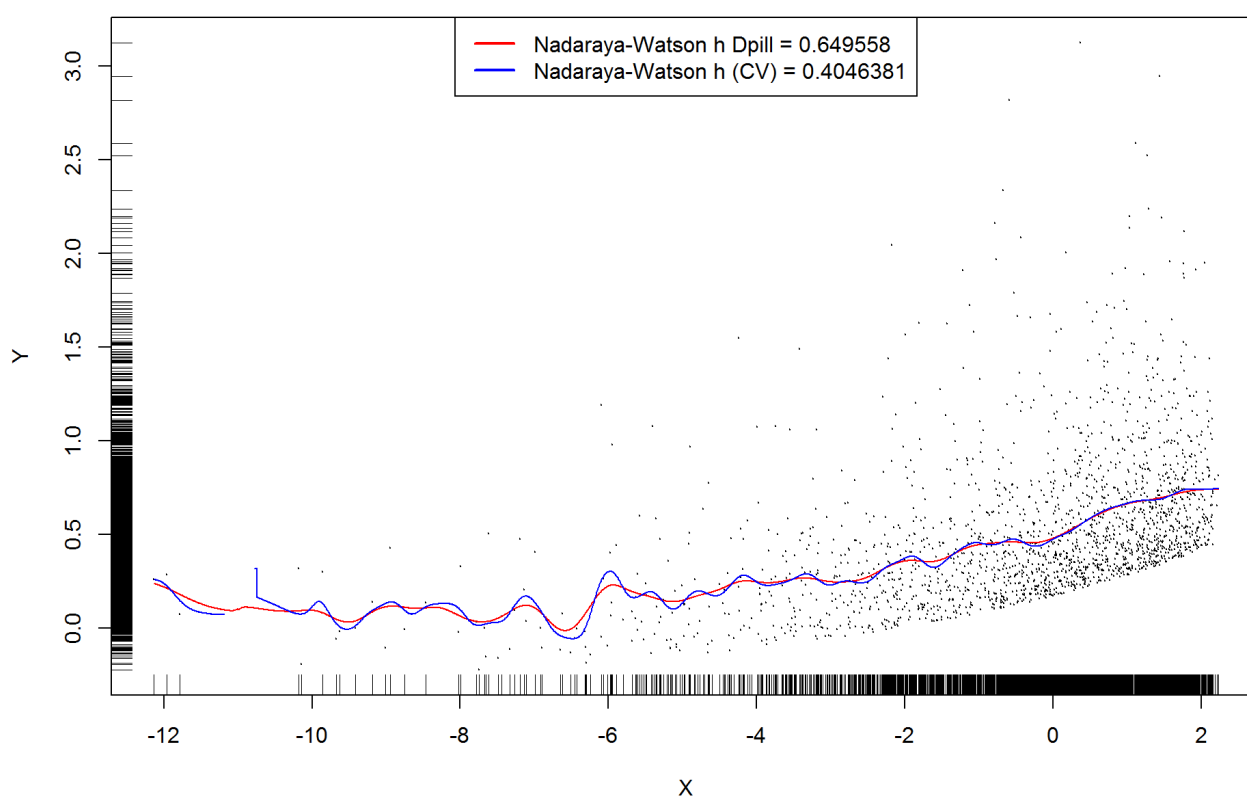
log_ksmoothDpill <- ksmooth(log(Xo), Yo, "normal", bandwidth = ln_hDpill)
log_ksmoothCV <- ksmooth(log(Xo), Yo, "normal", bandwidth = ln_hCV)

# Plot data
plot(log(Xo), Yo, pch=20, cex=0.01, xlab="X", ylab="Y", main="Estimateur de la regression de Log(X) avec Nadaraya-Watson: ksmooth
et différents h")
rug(log(Xo), side = 1); rug(Yo, side = 2)

lines(log_ksmoothDpill, col = "red")
lines(log_ksmoothCV, col = "blue")

legend("top", legend = c("Nadaraya-Watson h Dpill = 0.649558", "Nadaraya-Watson h (CV) = 0.4046381"), lwd = 2, col = c("red",
"blue"))
```

Estimateur de la regression de Log(X) avec Nadaraya-Watson: ksmooth et différents h



Estimateur de la régression de Log(X) avec Nadaraya-Watson à partir de la fonction recodé NW - h = 0.649558 et h = 0.4046381

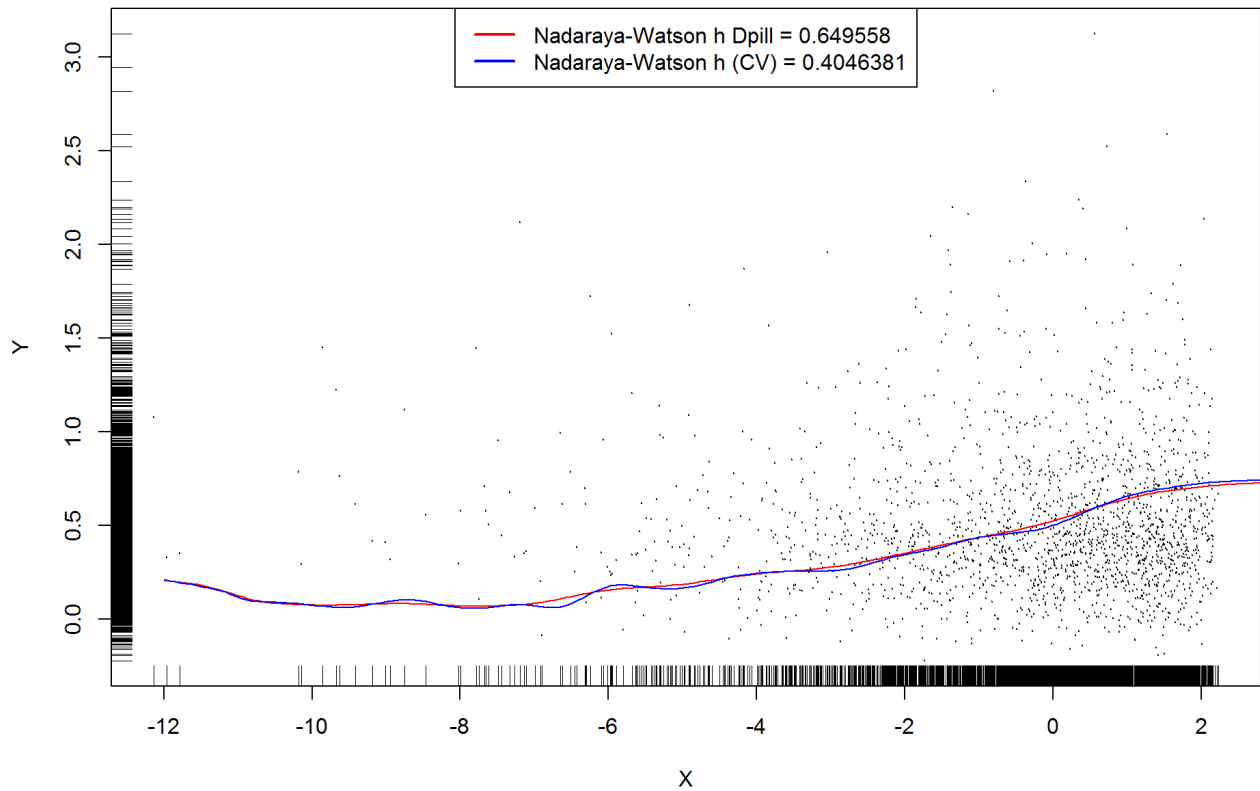
```
d1o <- order(d1[, "X"])
Xo <- d1[d1o, "X"]
Yo <- d1[d1o, "Y1"]
xGrid <- seq(-12, 3, l = 200)

plot(log(Xo), Y, pch=20, cex=0.01, xlab="X", ylab="Y", main="Estimation de la regression de Log(X) avec Nadaraya-Watson: fonction
NW")
rug(log(Xo), side = 1); rug(Yo, side = 2)

lines(xGrid, mNW(x = xGrid, X = log(Xo), Y = Yo, h = ln_hDpill), col = "red")
lines(xGrid, mNW(x = xGrid, X = log(Xo), Y = Yo, h = ln_hCV), col = "blue")

legend("top", legend = c("Nadaraya-Watson h Dpill = 0.649558", "Nadaraya-Watson h (CV) = 0.4046381"), lwd = 2, col = c("red",
"blue"))
```

Estimation de la regression de Log(X) avec Nadaraya-Watson: fonction NW



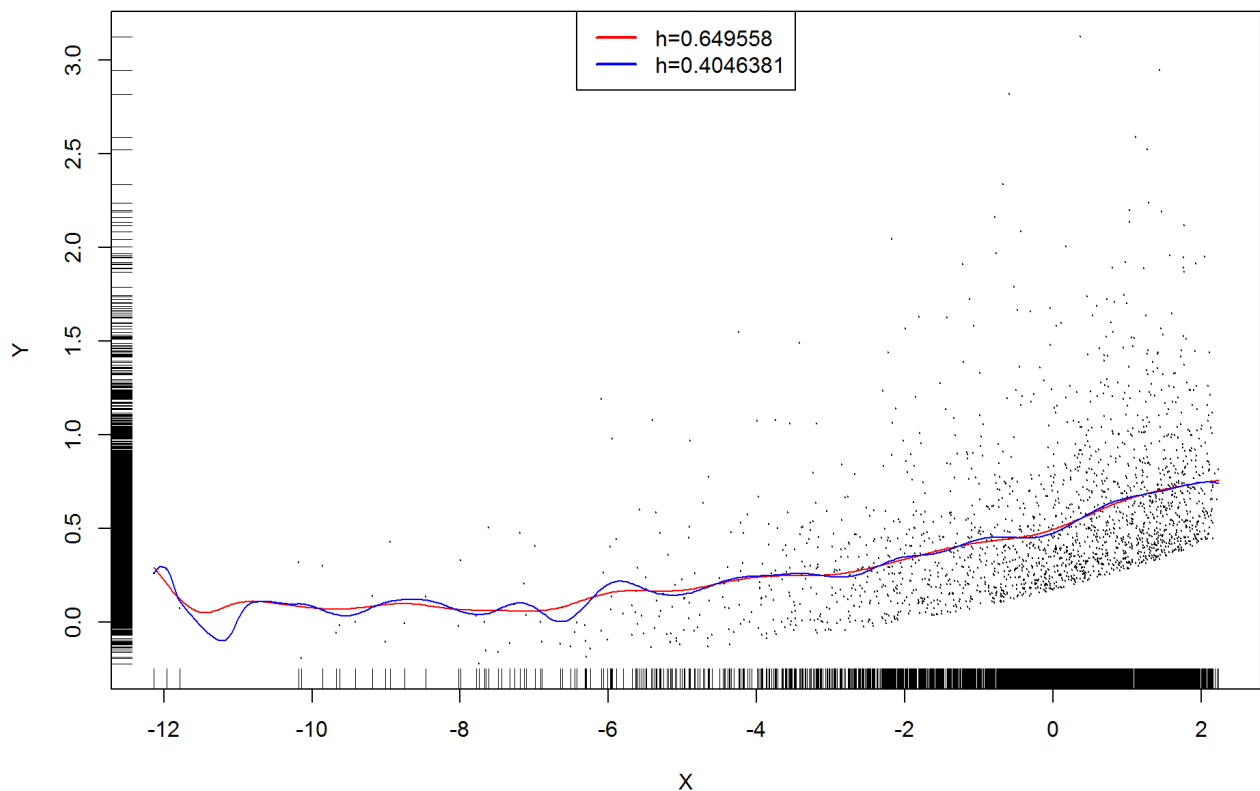
Estimation de la régression de Log(X) avec par polynômes locaux

```
plot(log(Xo),Yo,pch=20,cex=0.01,xlab="X",ylab="Y",main="Estimation de la regression de Log(X) par locpoly")
rug(log(Xo), side = 1); rug(Yo, side = 2)

lines(locpoly(log(Xo), Yo, degree = 2, bandwidth = ln_hDpill), col="red")
lines(locpoly(log(Xo), Yo, degree = 2, bandwidth = ln_hCV), col="blue")

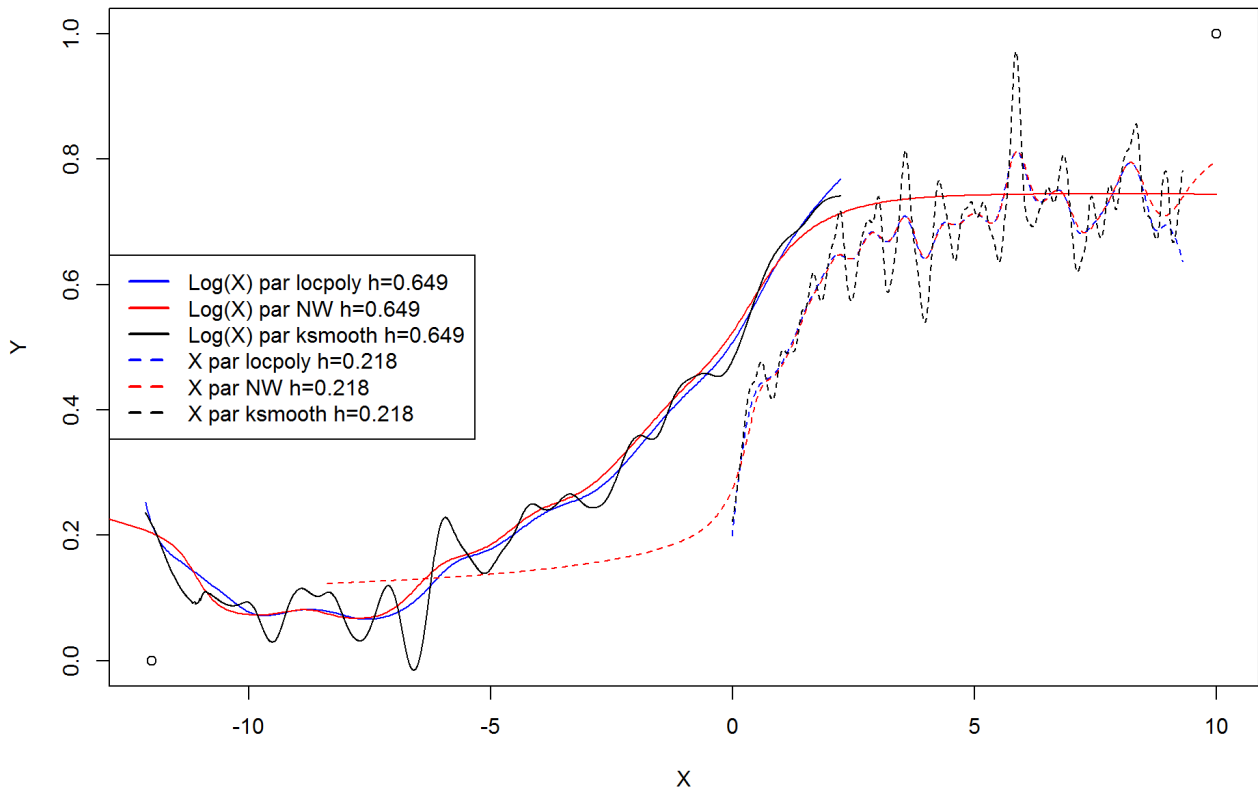
legend("top", legend = c("h=0.649558","h=0.4046381"), lwd = 2, col = c("red","blue"), lty = 1)
```

Estimation de la regression de Log(X) par locpoly



Dans la zone à forte densité où est concentrée l'essentiel de l'information $[-5,1]$ les 2 estimateurs sont très proches, quasiment confondus.

Estimation de X et Log(X) par Nadaraya–Watson et locPoly



2.4. Remarques? Explications?

Régularité proportionnalité?

3. Etude de la densité $\mu(x)$ des ξ_i

3.1 A partir du jeu de données Data1

3.1.1.

La distribution approximative de $\sim \xi_i$ est celle de ξ_i

3.1.2 Représentation de la densité $\mu(x)$ des ξ_i

choix de h établi a la question 2.2: $h=0.2186849$

h_ucv

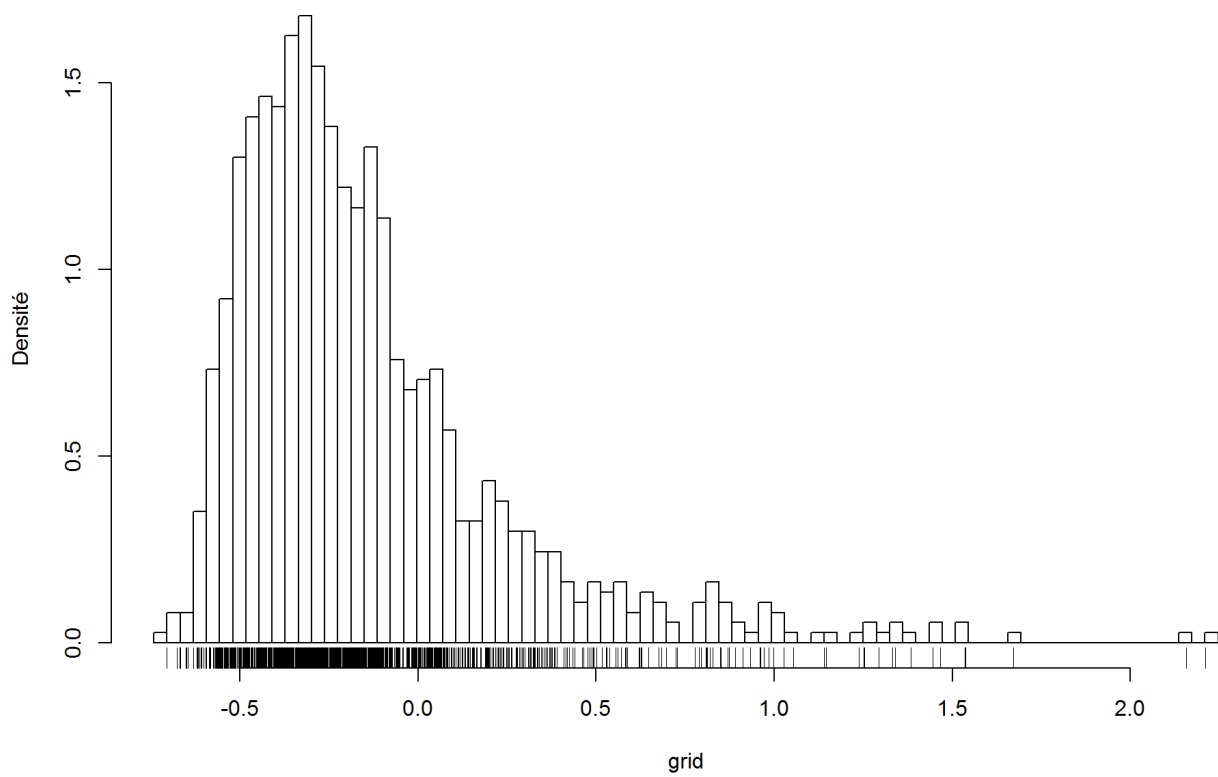
[1] 0.004196189

h_sil

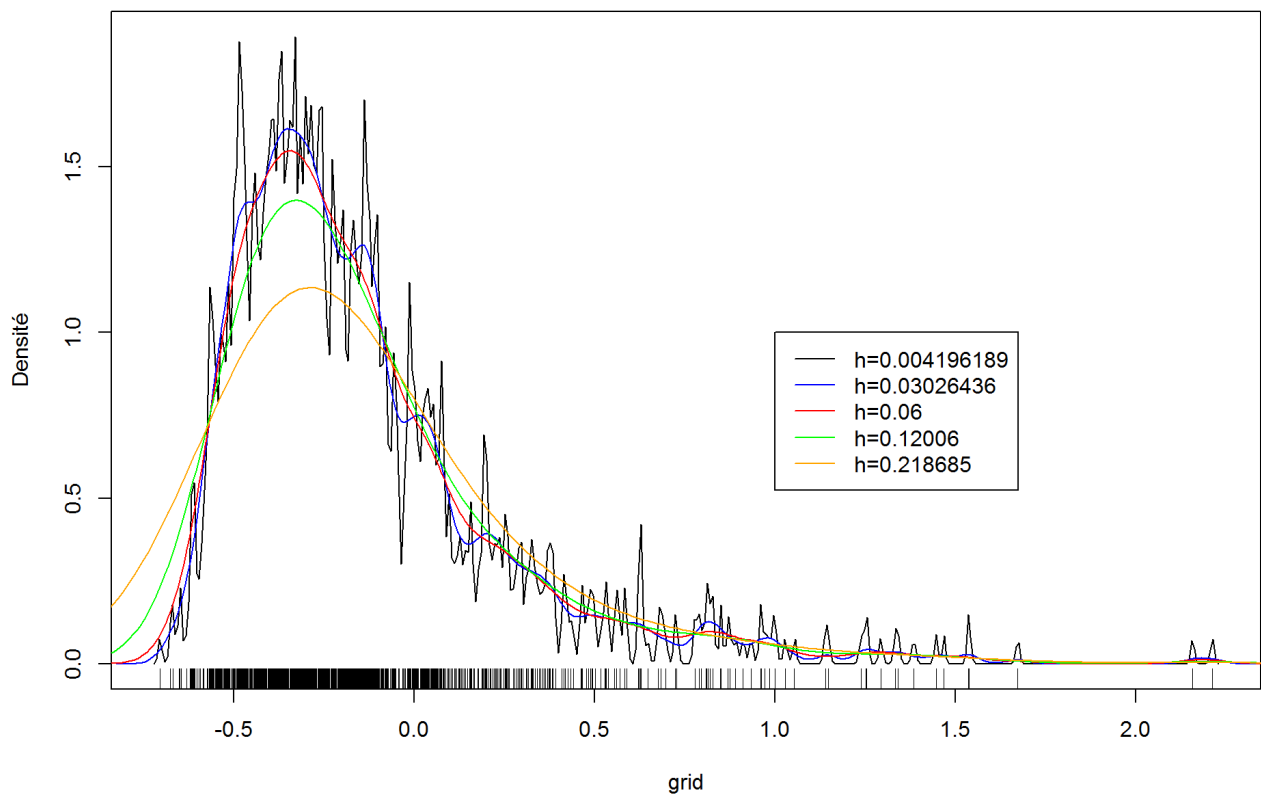
[1] 0.03026436

```
# hist(locpoly_hdpill$y, breaks=grid_dpih,freq = FALSE,xlab="grid",ylab="Densité", main="Histogramme et densité" )
#
# lines(r_ucv$x,r_ucv$y,type="l",col="blue")
# lines(r_sil$x,r_sil$y,type="l",col="red")
#
# legend(6,3,c("h=0.003554934", "h=0.02757643"), col=c("blue", "red"),lty=1:1)
```

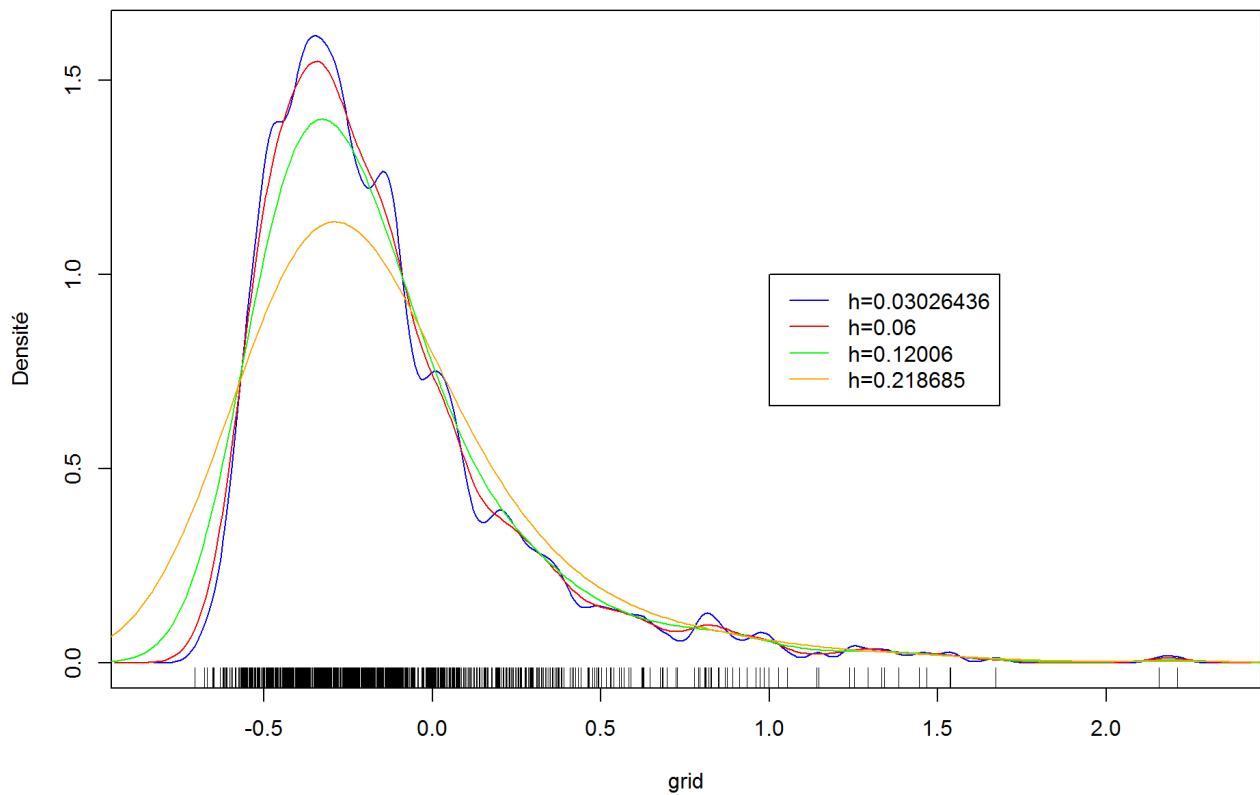
Histogramme pour la densité μ de ξ



Estimation de la densité μ de ξ et différentes fenêtres: h



Estimation de la densité μ de ξ et différentes fenêtres: h



On retrouve la forme du graphe de la densité d'une loi gamma

3.1.3. (Facultatif.) Quel est l'intérêt d'avoir decoupé le jeu de données selon J_+ et J_- ?

Données d'apprentissage et données de test. On peut utiliser le jeu de données d'apprentissage pour estimer notre loi, et le jeu de test

Pas de fonction "predict" On programme la fonction qui définit l'estimateur.

```
## [1] 0.3622163
```

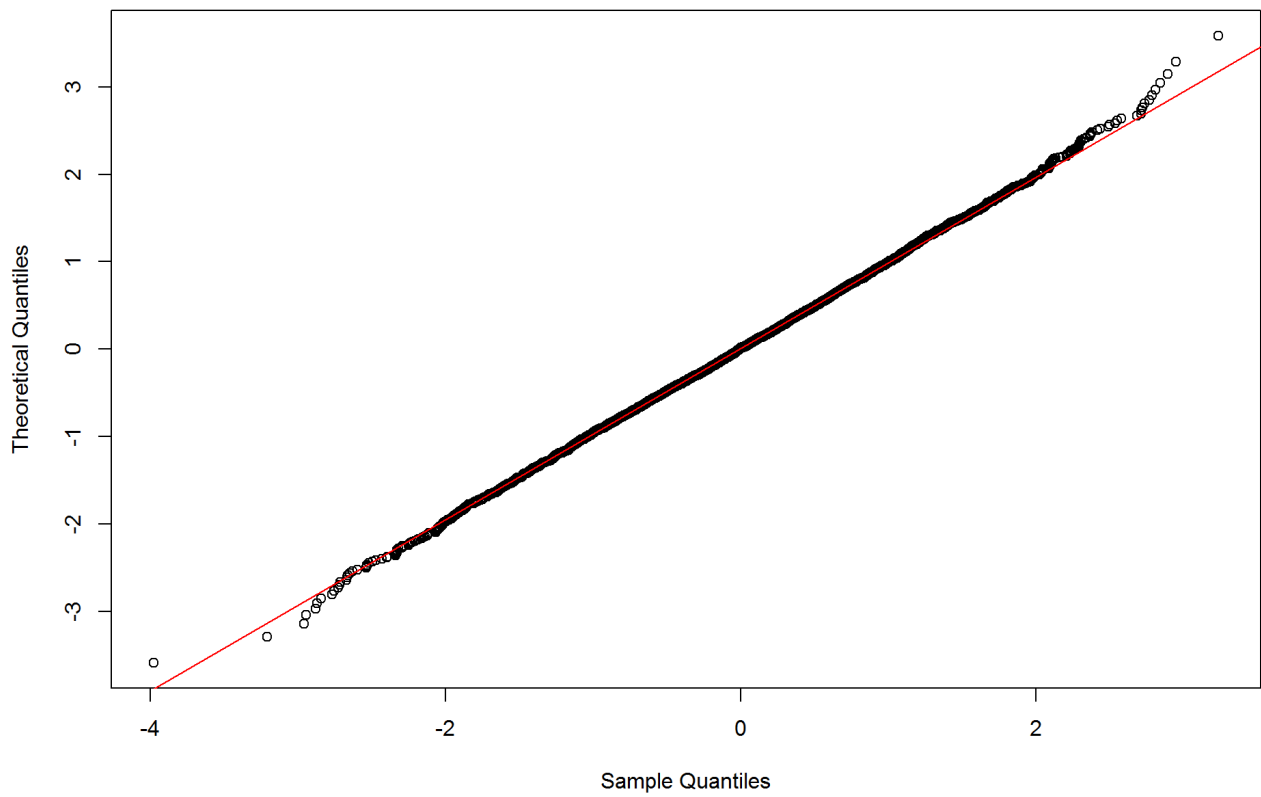
```
## [1] 0.3622163
```

3.1.4. La densité μ peut-elle être gaussienne ?

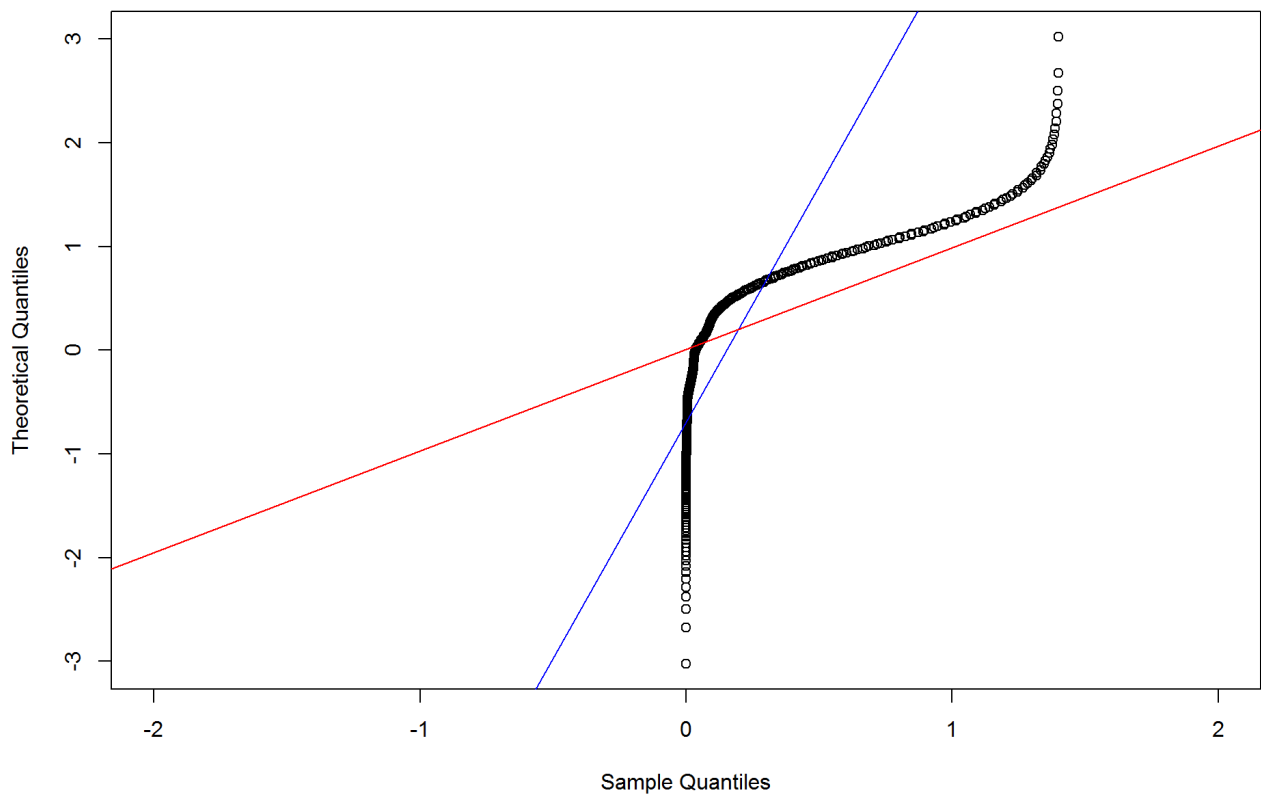
Protocole empirique de verification

- 1. test du QQPlot

Q-Q plot pour la loi Normale $N(0,1)$

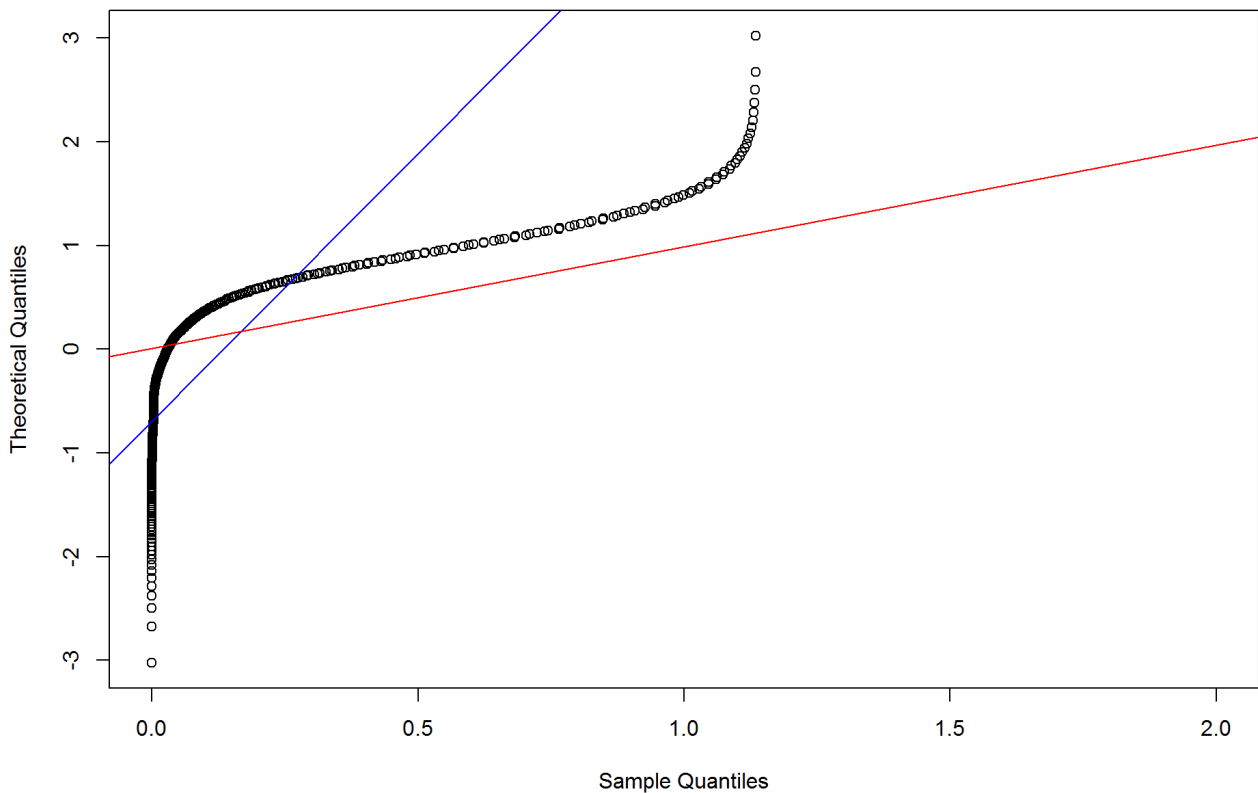


Q-Q plot pour la loi μ



=> non ou bien gaussien par morceau

Q-Q plot pour la loi μ



- 2. test de shapiro

```
shapiro.test(nu_hcv$y)
```

```
##
## Shapiro-Wilk normality test
##
## data:  nu_hcv$y
## W = 0.6624, p-value < 2.2e-16
```

```
shapiro.test(norm)
```

```
##
## Shapiro-Wilk normality test
##
## data:  norm
## W = 0.88545, p-value < 2.2e-16
```

Le test de Shapiro-Wilk donne une probabilité de dépassement de $p\text{-value} < 2.2e-16$, nettement < 0.05 . L'hypothèse de normalité est rejetée.

- 3. test de Kolmogorov-Smirnoff Dans ce cas-ci également, il existe dans R une commande pour tester l'ajustement de données à une loi normale via le test de Kolmogorov-Smirnov:

```
ks.test(nu_hcv$y, "pnorm", mean(nu_hcv$y), sd(nu_hcv$y))
```

```
##
## One-sample Kolmogorov-Smirnov test
##
## data:  nu_hcv$y
## D = 0.28319, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

p-value faible rejeté

3.1.5. homoscedasticité du modèle

(Facultatif.) Comment peut-on tester si le modèle est bien homoscedastique ?

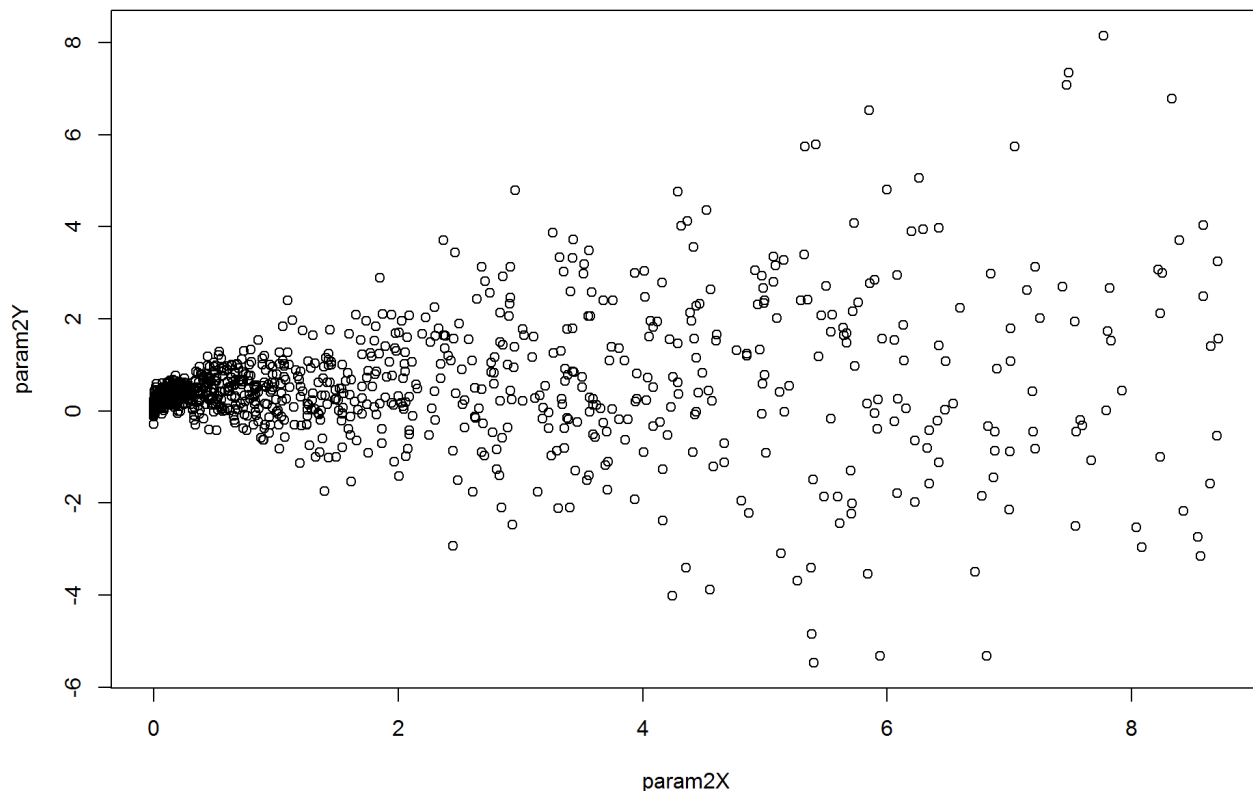
3.2 A partir du jeu de données Data2

On cherche à estimer $\mu(x)$ et $\sigma(x)^2$. Pour cela, on coupe à nouveau l'échantillon en deux et on considère à nouveau $ksi(i)$:

```
summary(d2)
```

##	X.1	X	Y2
## Min. :	1.0	Min. :0.000005	Min. :-5.46738
## 1st Qu.:	500.8	1st Qu.:0.258074	1st Qu.: 0.08333
## Median :	1000.5	Median :1.192414	Median : 0.35947
## Mean :	1000.5	Mean :2.029447	Mean : 0.53951
## 3rd Qu.:	1500.2	3rd Qu.:3.318174	3rd Qu.: 0.87849
## Max. :	2000.0	Max. :9.308684	Max. :10.15297

Jeux de données Data2 observations X en abscisse et Y en ordonnée.



3.2.1. Justifier qu'en regressant ξ_i sur X_i on obtient un estimateur de $\sigma(x)^2$.

Implémentation et le visualisation graphique

choix de h etabli a la question 2.2: $h=0.2186849$

```
#hist(fit22$y, breaks=bins32,freq = FALSE,xlab="grid",ylab="Densité", main=expression("Histogramme et estimateurs de "sigma
a"^2"))

#lines(resH32$x,resH32$y,type="l",col="blue")
#lines(resHH32$x,resHH32$y,type="l",col="red")
#legend(6,3,c("h=0.01079333", "h=0.08372629"), col=c("blue", "red"),lty=1:1)
```

```
## [1] 0.01467149
```

```
## [1] 0.11381
```

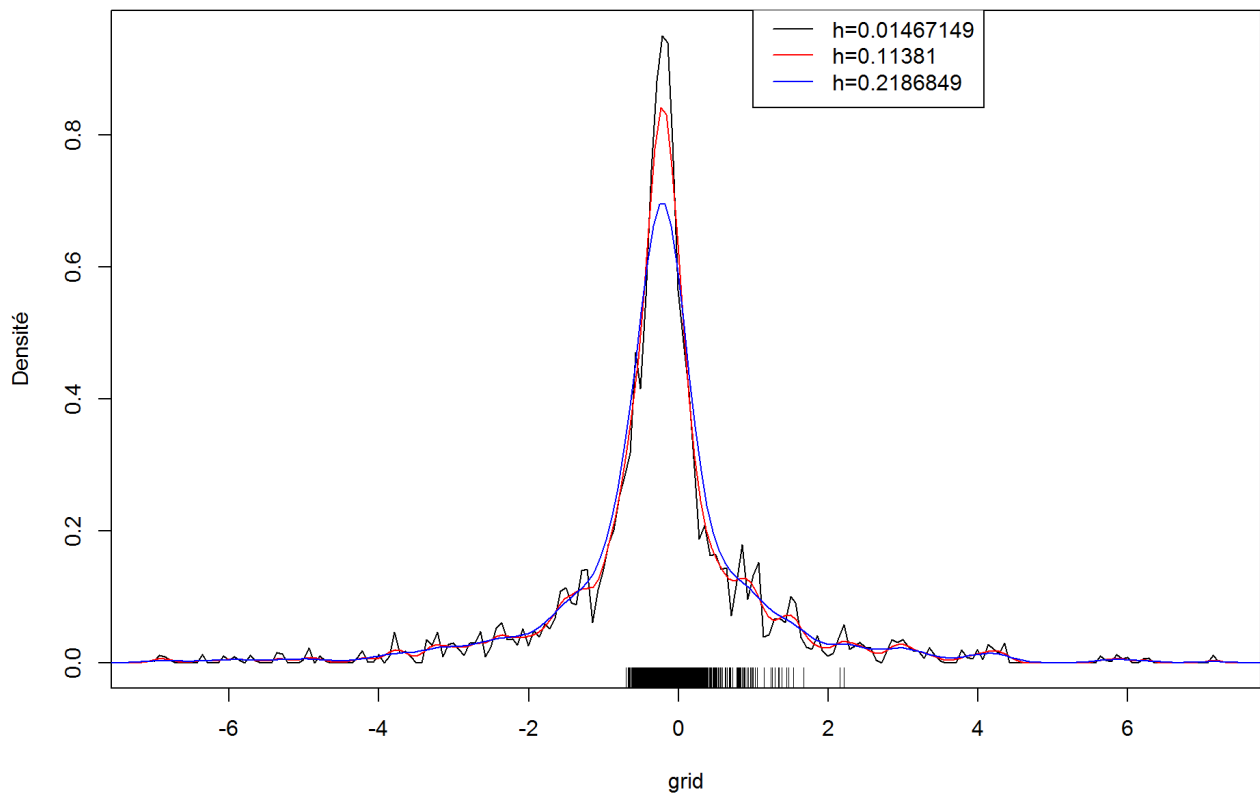
```
## [1] 0.1200601
```

```
## [1] 0.2186849
```

```
plot(mu_ucv2$x,mu_ucv2$y,type="l",col="black",xlab="grid",ylab="Densité",
     main = expression("Densité estimée pour la loi "mu" et différentes fenêtres: h"))
rug(estNu, side = 1);

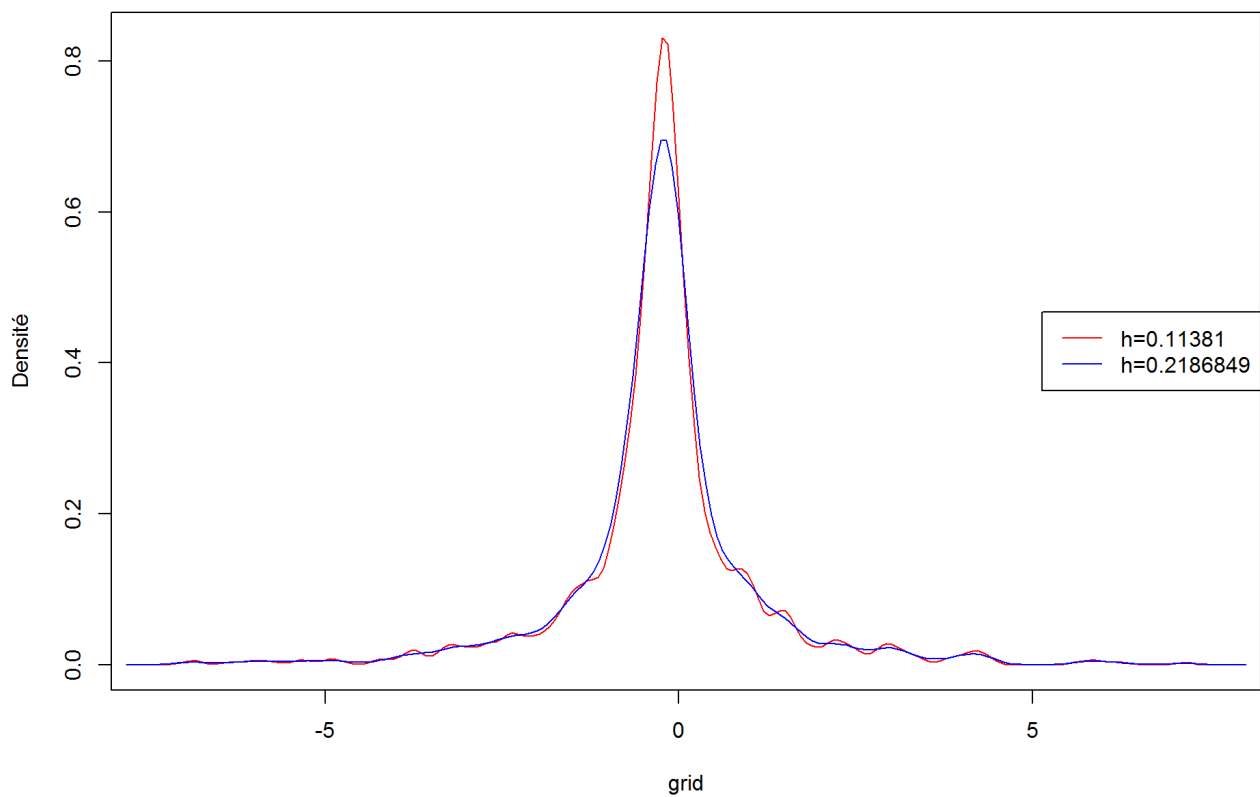
lines(mu_sil2$x,mu_sil2$y,type="l",col="red")
#lines(sigma_hcv$x,sigma_hcv$y,type="l",col="green")
lines(mu_hDpill$x,mu_hDpill$y,type="l",col="blue")
legend(1,1,c("h=0.01467149", "h=0.11381", "h=0.2186849"), col=c("black", "red", "blue"),lty=1:1)
```

Densité estimée pour la loi μ et différentes fenêtres: h

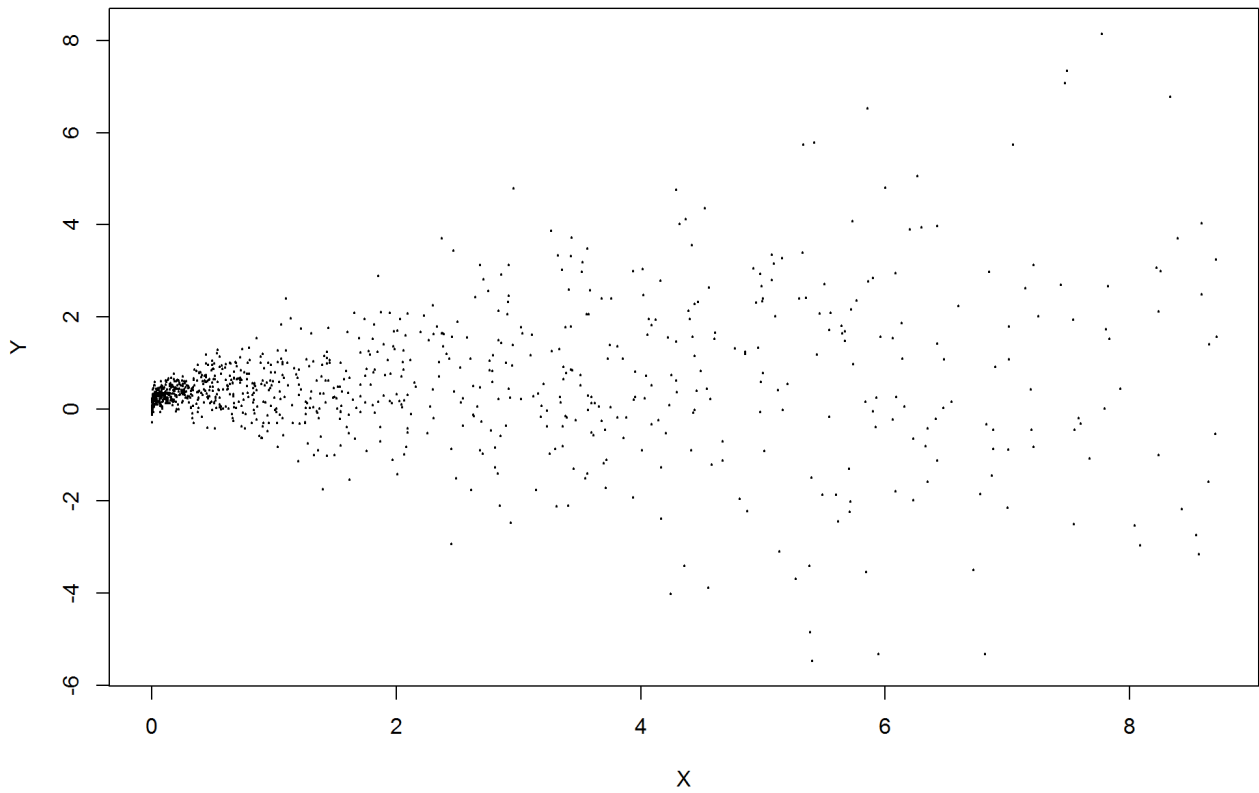


```
plot(mu_hcv$x,mu_hcv$y,type="l",col="red" ,xlab="grid",ylab="Densité", main = expression("Densité estimée pour la loi " *mu*
et différentes fenêtres: h"))
lines(mu_hDpill$x,mu_hDpill$y,type="l",col="blue")
#legend(1,1,c("h=0.11381","h=0.2186849"), col=c("red","blue"),lty=1:1)
legend("right", legend = c("h=0.11381","h=0.2186849"), col=c("red","blue"),lty=1:1)
```

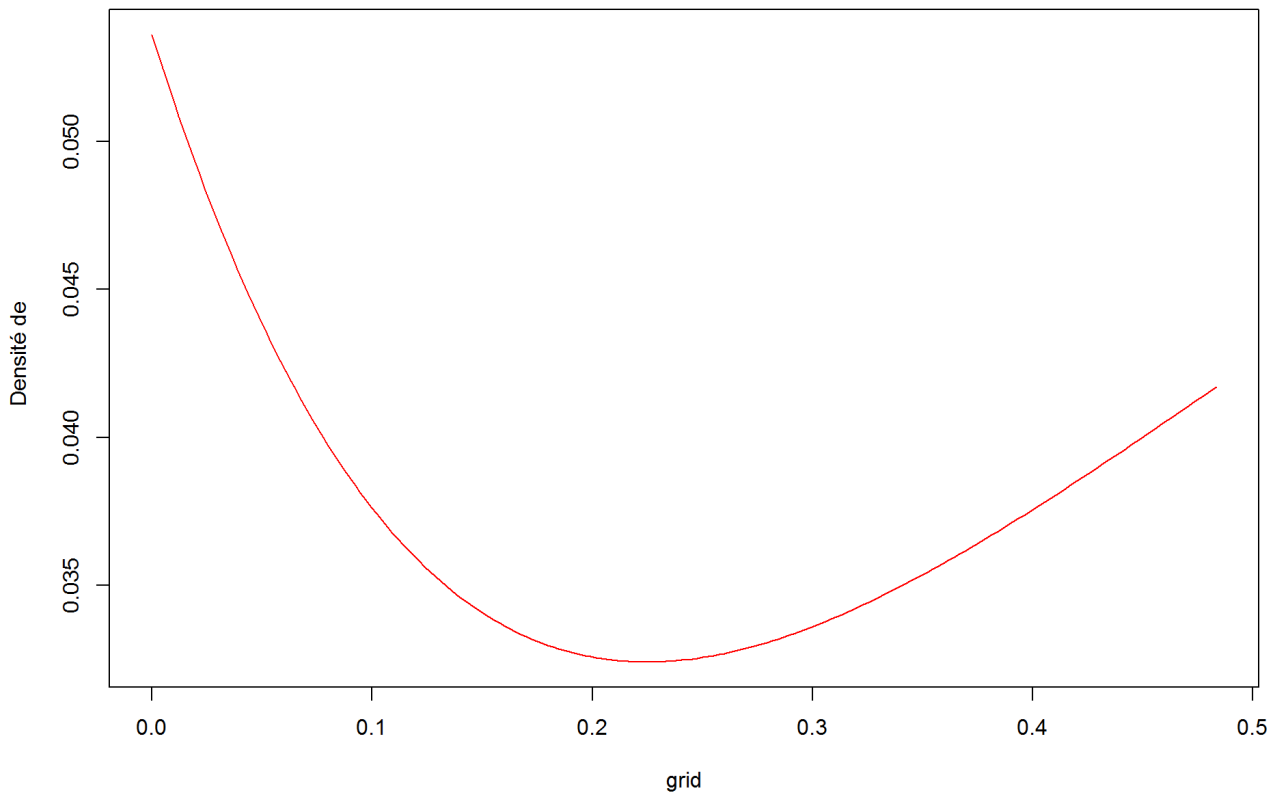
Densité estimée pour la loi μ et différentes fenêtres: h



Nuage des points du jeux de test Data2



Estimation de σ^2 par Regression de ξ^2 sur X



En comparant avec le jeu de données (Figure 1 a droite), retrouve-t-on un résultat attendu:

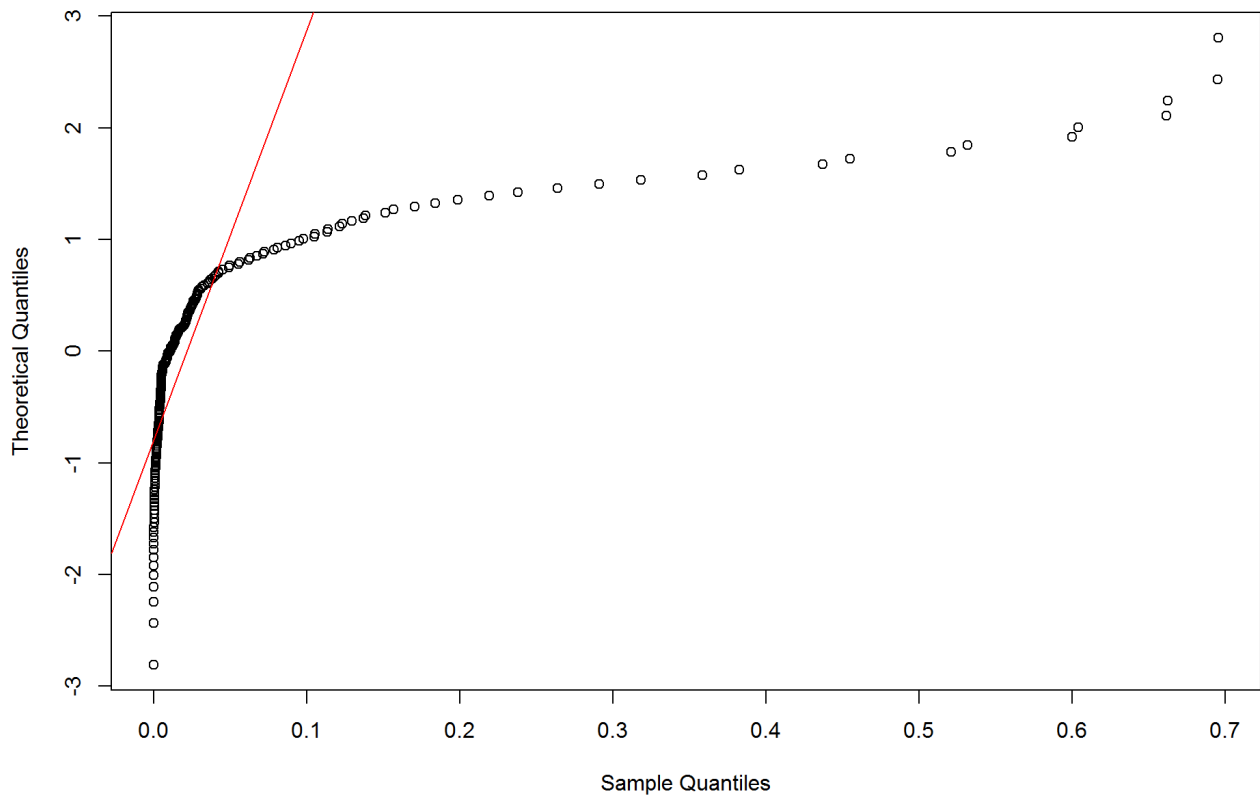
=> symétrie, centrée en 0

3.2.2. La densité $x \rightarrow \mu(x)$ peut-elle être gaussienne ?

Proposer un protocole pour le vérifier empiriquement et l'implémenter. On pourra penser à renormaliser ξ par la fonction estimée à la question précédente et s'aider des questions de la Section 3.1.

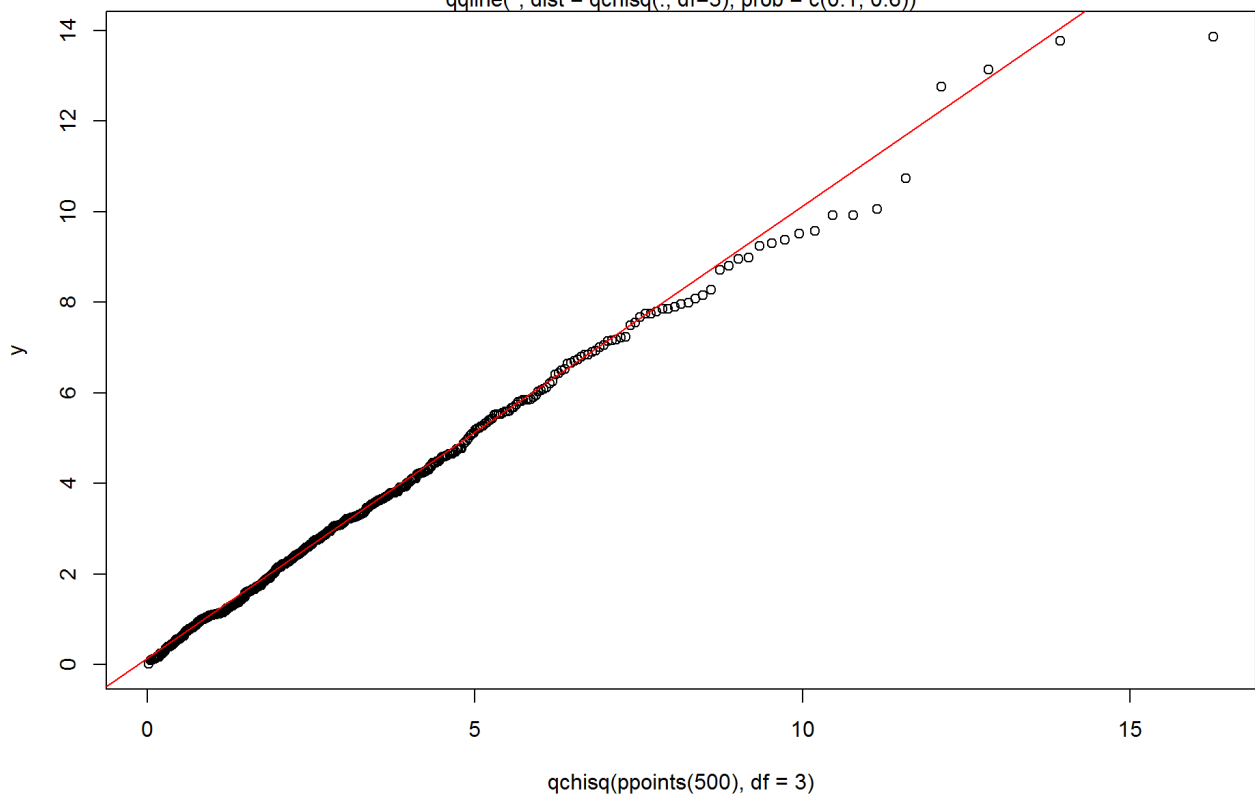
- 1- test du QPlot

Q-Q plot de normalité pour la loi μ



Q-Q plot for $\chi^2_{v=3}$

qqline(*, dist = qchisq(., df=3), prob = c(0.1, 0.6))



=> non ou bien gaussien par morceau

- 2- test de shapiro

```
shapiro.test(mu_hDpill$y)
```

```
##
## Shapiro-Wilk normality test
##
## data: mu_hDpill$y
## W = 0.49213, p-value < 2.2e-16
```

Le test de Shapiro-Wilk donne une probabilité de dépassement de p-value $< 2.2\text{e-}16$, nettement < 0.05 . L'hypothèse de normalité est rejetée.

- 3- test de Kolmogorov-Smirnoff Dans ce cas-ci également, il existe dans R une commande pour tester l'ajustement de données à une loi normale via le test de Kolmogorov-Smirnov:

```
ks.test(mu_hDpill$y, "pnorm", mean(mu_hDpill$y), sd(mu_hDpill$y))
```

```
## Warning in ks.test(mu_hDpill$y, "pnorm", mean(mu_hDpill$y),
## sd(mu_hDpill$y)): ties should not be present for the Kolmogorov-Smirnov
## test
```

```
##
## One-sample Kolmogorov-Smirnov test
##
## data: mu_hDpill$y
## D = 0.32436, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

p-value faible rejeté

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.