Régression non-paramétrique

Philippe Real

19/09/2019

#rmarkdown::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")
```

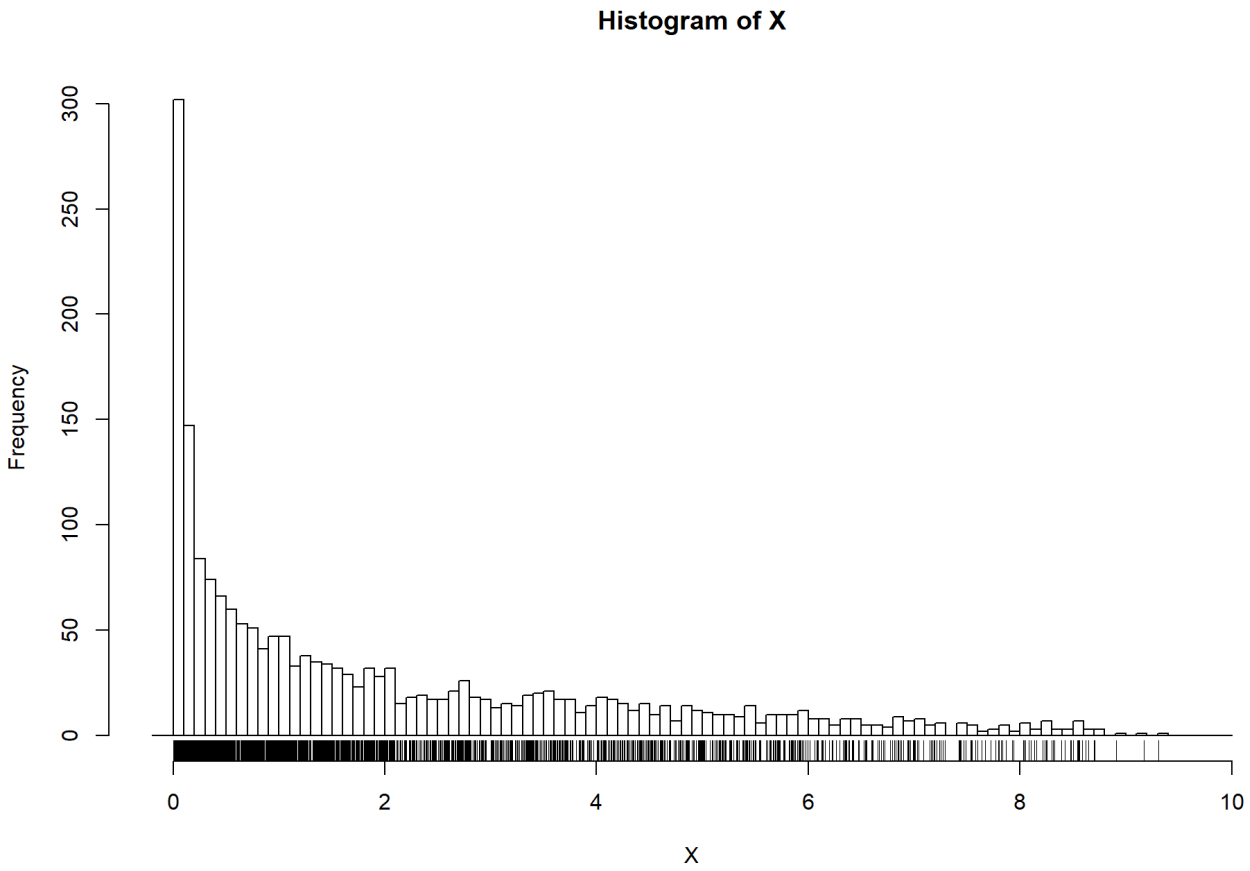
1. Etude de la densitée g des X

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.

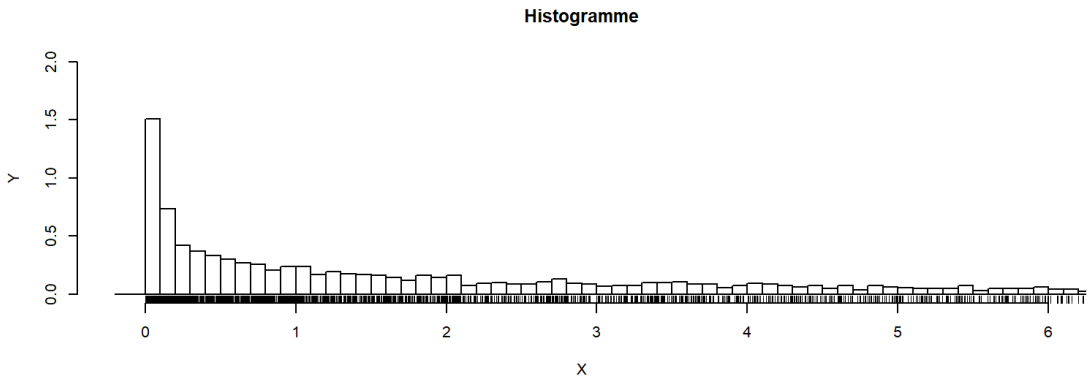


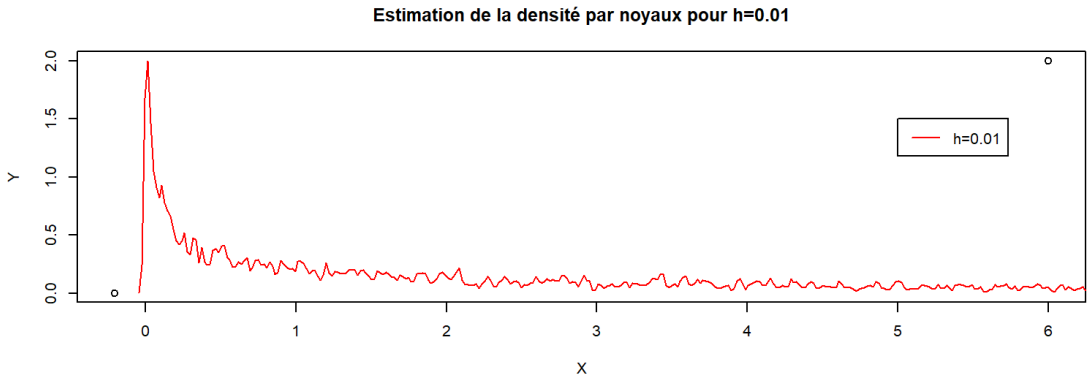
1.1 Estimateur non-paramétrique de g(x)

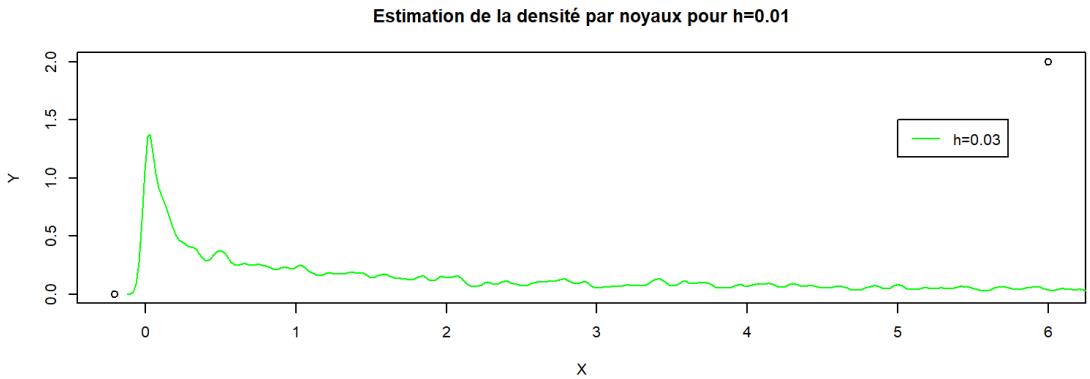
On peut utiliser la fonction bkde du package KernSmooth qui estime la densité par la méthode des noyaux. On prend comme noyau le noyau normal. Ce choix peut sembler arbitraire, mais on a vu que ce n'est pas le choix du noyau qui est le plus important dans l'estimation de la densité. On calcul cette estimateur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

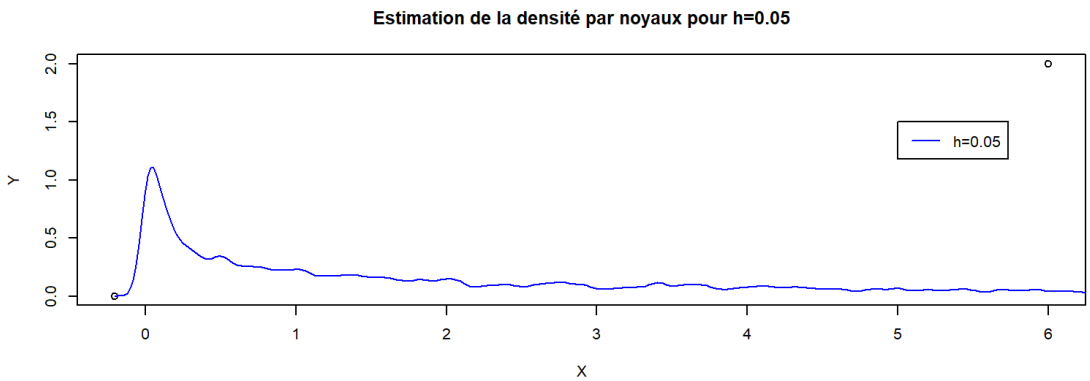
Petites valeurs de la fenêtre h: 0.01 / 0.03 / 0.05

bandwidth h	error quadratic	R2	Biais	MAE	RMSE
0.01	8.4851915	-0.7836155	-1.9231488	1.9540129	2.9129352
0.03	8.4818664	-0.7829165	-1.92493	1.9555547	2.9123644
0.05	8.4958757	-0.7858613	-1.9266524	1.9587723	2.9147686





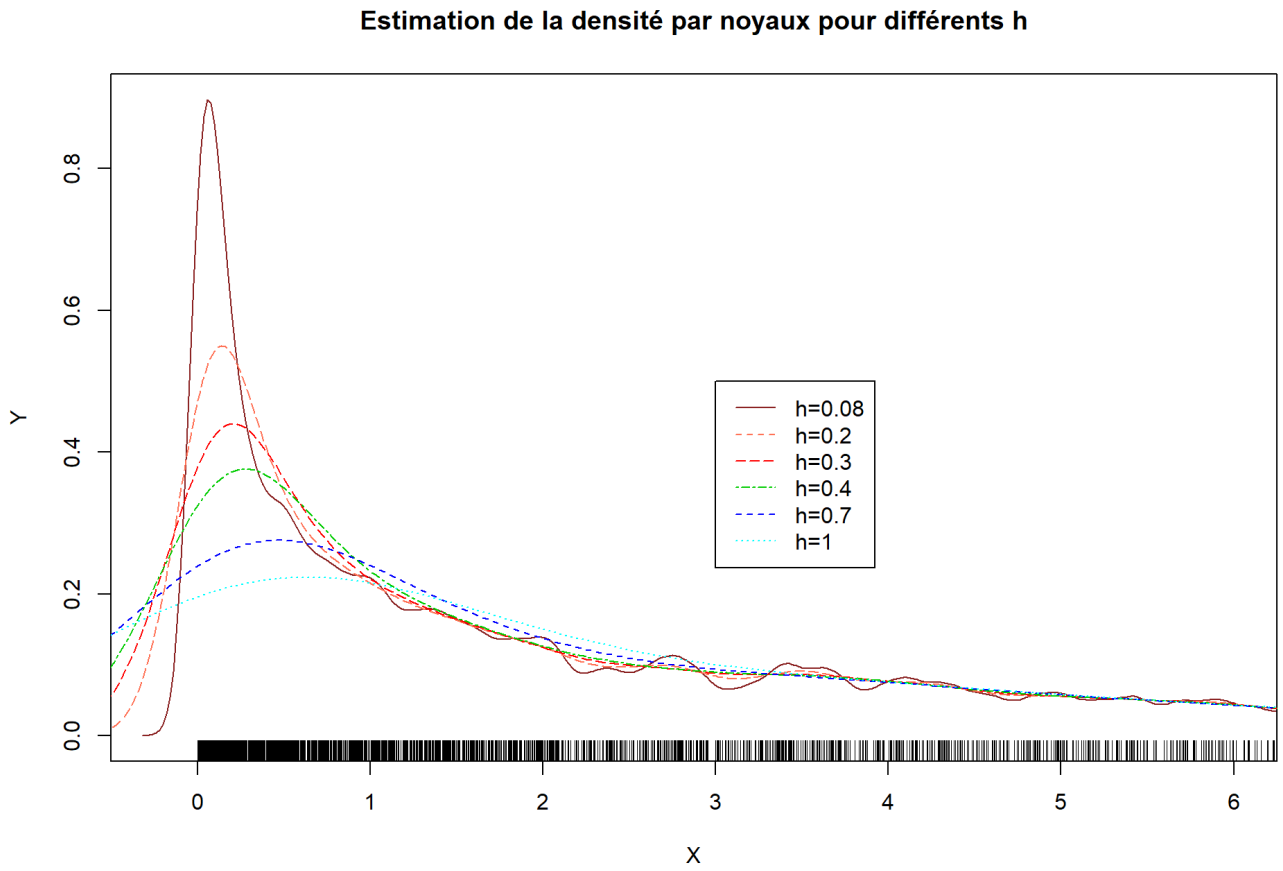




On remarque de fortes oscilations pour des h entre 0.01 et 0.05.

Estimateurs pour h variant de 0.08 à 1

bandwidth h	error quadratic	R2	Biais	MAE	RMSE
0.08	8.4973667	-0.7861747	-1.9291322	1.9589151	2.9150243
0.20	8.5346547	-0.7940128	-1.9379603	1.9654464	2.9214131
0.40	8.5686374	-0.8011561	-1.9496624	1.9715207	2.9272235
0.70	8.6026503	-0.8083057	-1.9625062	1.9786641	2.9330275
1	8.6378039	-0.8156951	-1.9717881	1.9849216	2.9390141



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

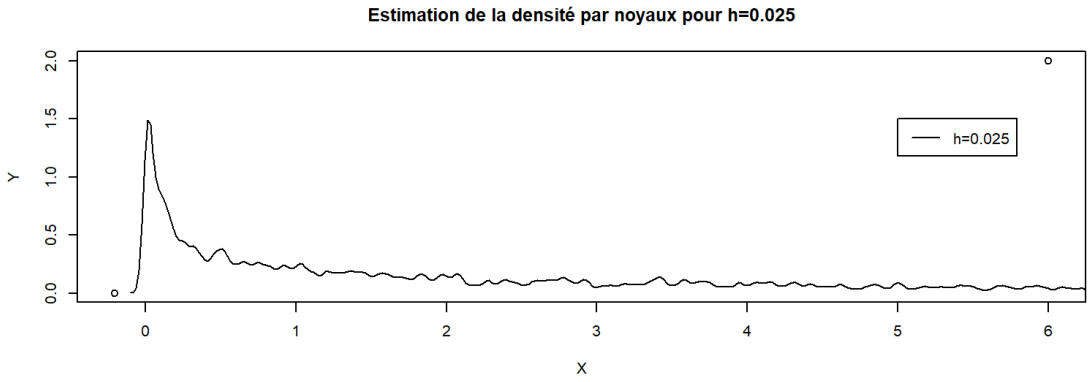
Raison pour laquelle ce choix est important et ce qui se produit si h est mal choisi

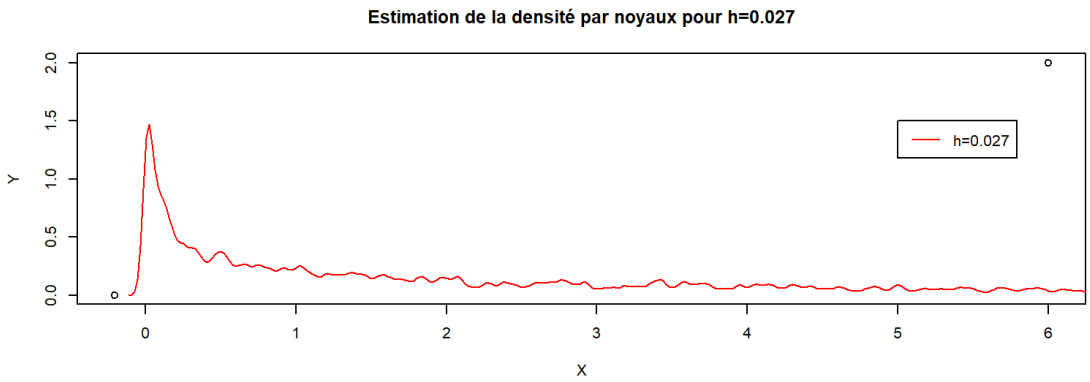
- **Si h est trop grand (courbes bleues) noyau trop régularisant : trop de biais**
- **Si h est trop petit (courbes marron du graphique ci-dessus) : trop oscillant, trop de variance.**

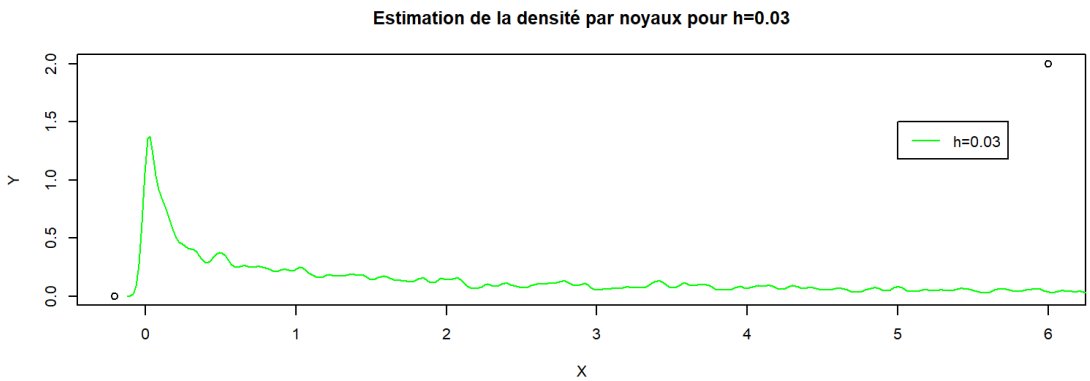
Si on regarde le critère de l'erreur quadratique un h adapté serait compris entre 0.4 et 0.5 Par la suite on va utiliser d'autres critères pour définir un h optimal.

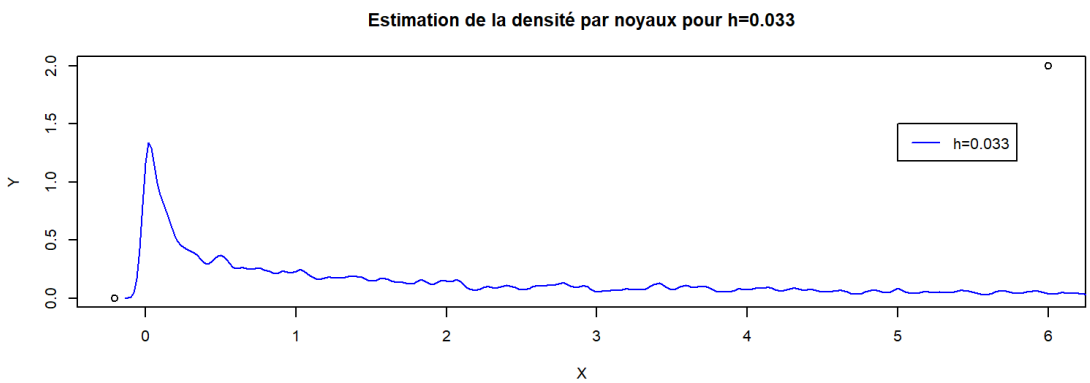
Raffinement pour h entre 0.025 et 0.035

bandwidth h	error quadratic	R2	Biais	MAE	RMSE
0.01	8.4851915	-0.7836155	-1.9231488	1.9540129	2.9129352
0.025	8.4811236	-0.7827604	-1.9244903	1.9541078	2.9122369
0.027	8.4810626	-0.7827475	-1.9246666	1.9546911	2.9122264
0.03	8.4818664	-0.7829165	-1.92493	1.9555547	2.9123644
0.033	8.4836088	-0.7832828	-1.925192	1.9564214	2.9126635
0.035	8.4850721	-0.7835904	-1.925366	1.9569291	2.9129147









1.2 Détermination d'un h optimal

Représentation graphique de l'estimation par noyau de la densité ed X: g(x), où h.n est la fenêtre donnee par validation croisee ou par une autre methode que l'on precisera.

1.2.1 Validation croisée pour la densité,

Utilisation de la fonction bw.ucv library stats

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

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

1.2.2 Règle Silverman

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

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

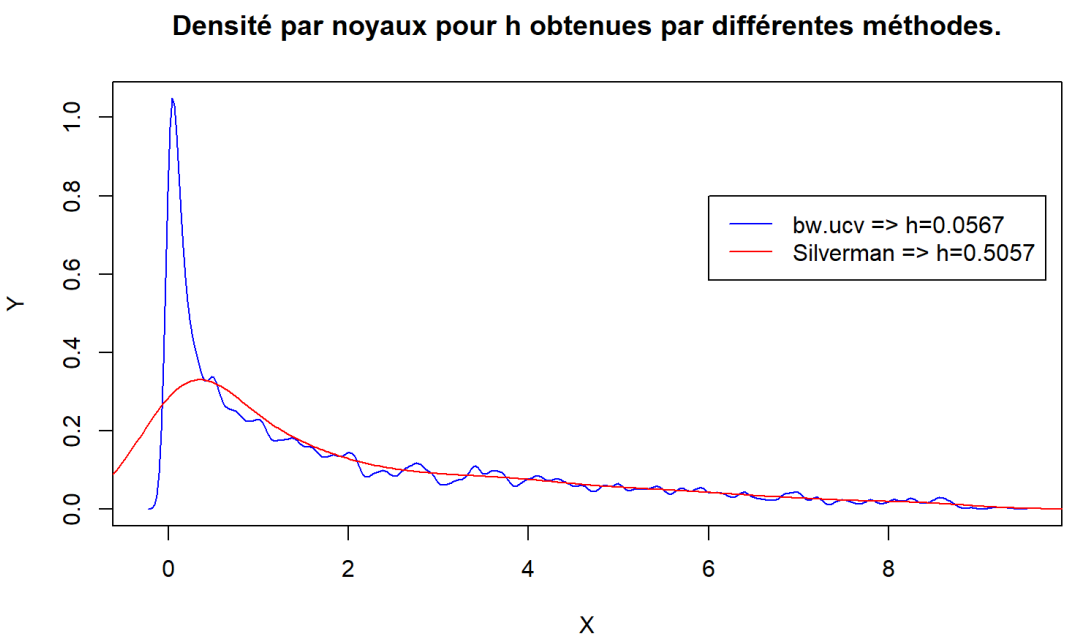
```
## [1] 0.505695
```

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

Fonction bw.nrd

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

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



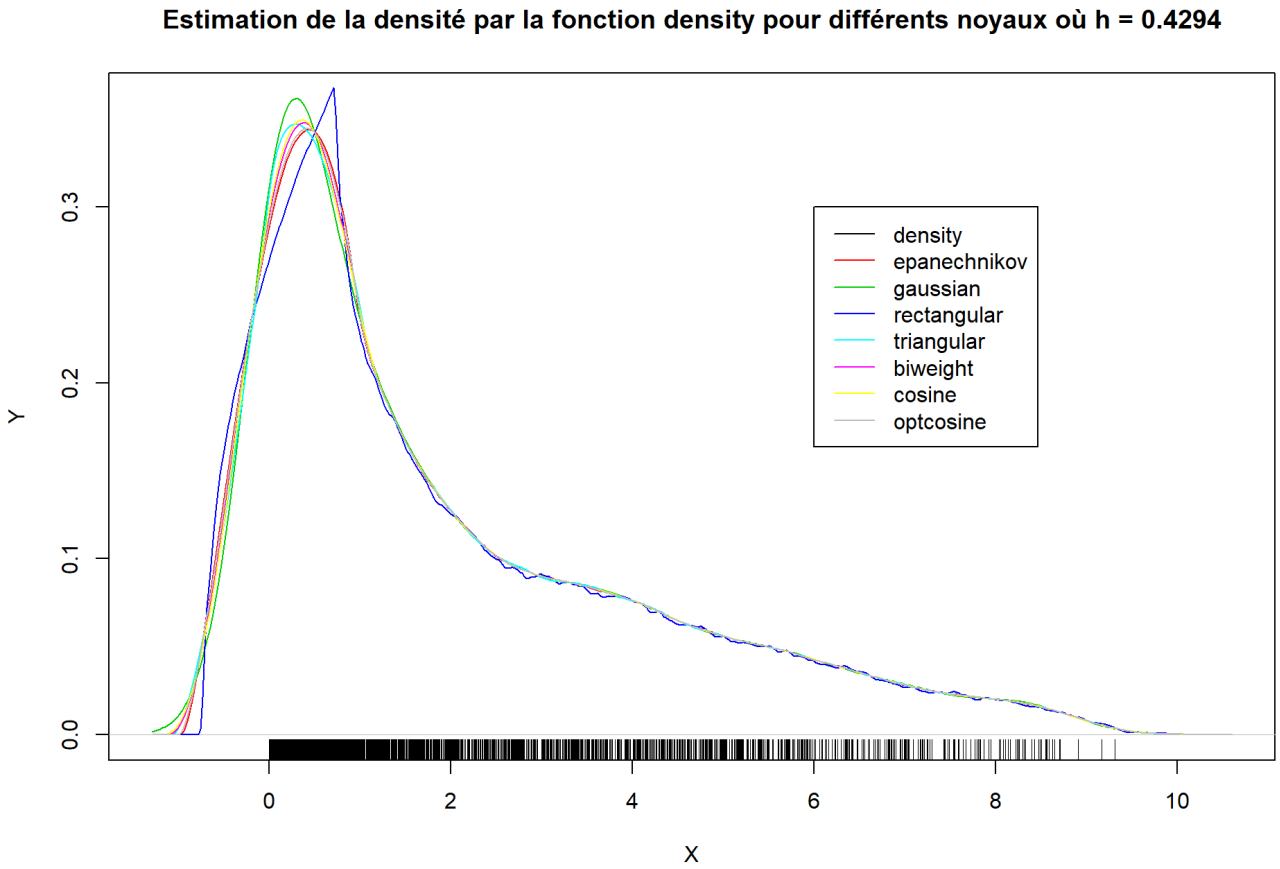
1.2.3 Méthodes alternatives

Fonction density

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

```
## [1] 0.4293637
```

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



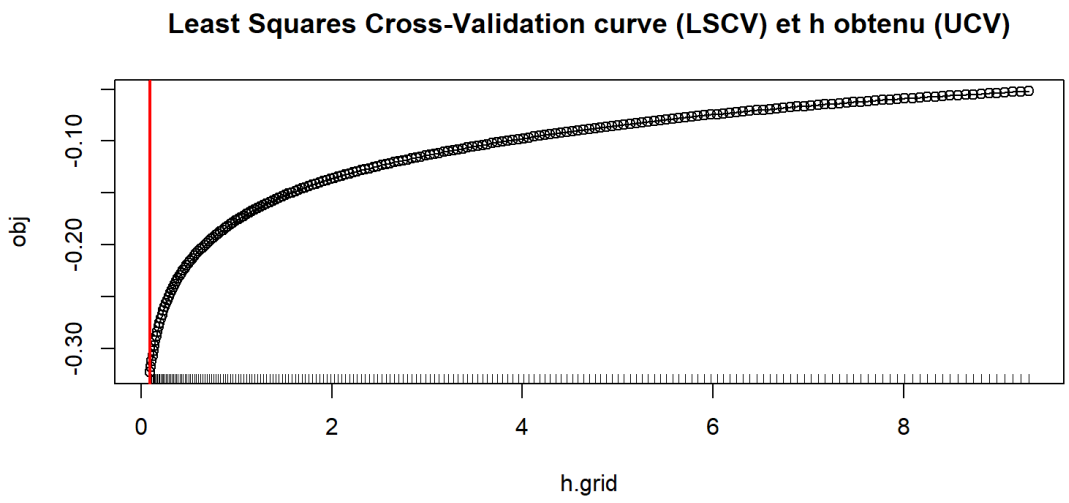
Fonction dpik

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

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

Fonction ucv

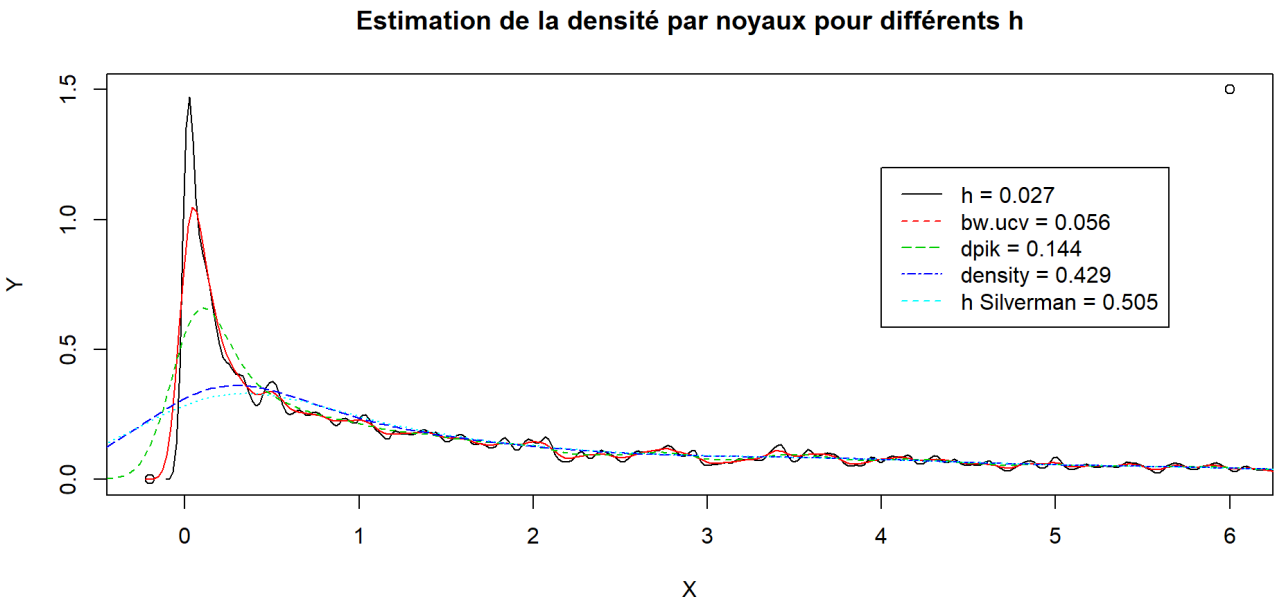
La fenêtre h est obtenu par (UCV) de Least Squares Cross-Validation curve (LSCV) et h obtenu (UCV)




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

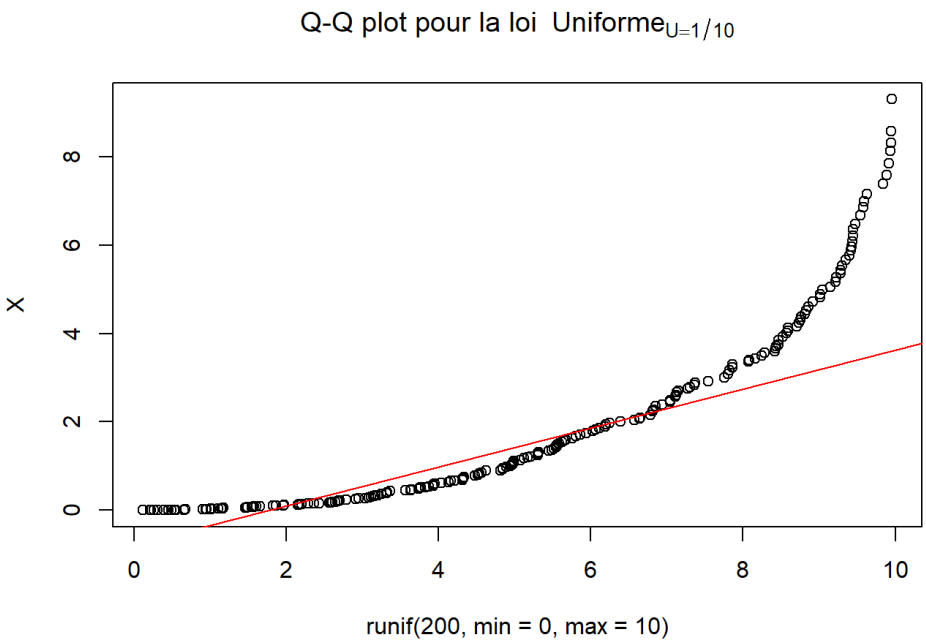
Résumé des résultats

Méthode	valeur de h
bw.ucv	0.0567514
dpik	0.1439489
density	0.4294
ESM	autour de 0.027
Regle Silverman	0.505695



1.3 QQPlot - g(x) densité Uniforme ?

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



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

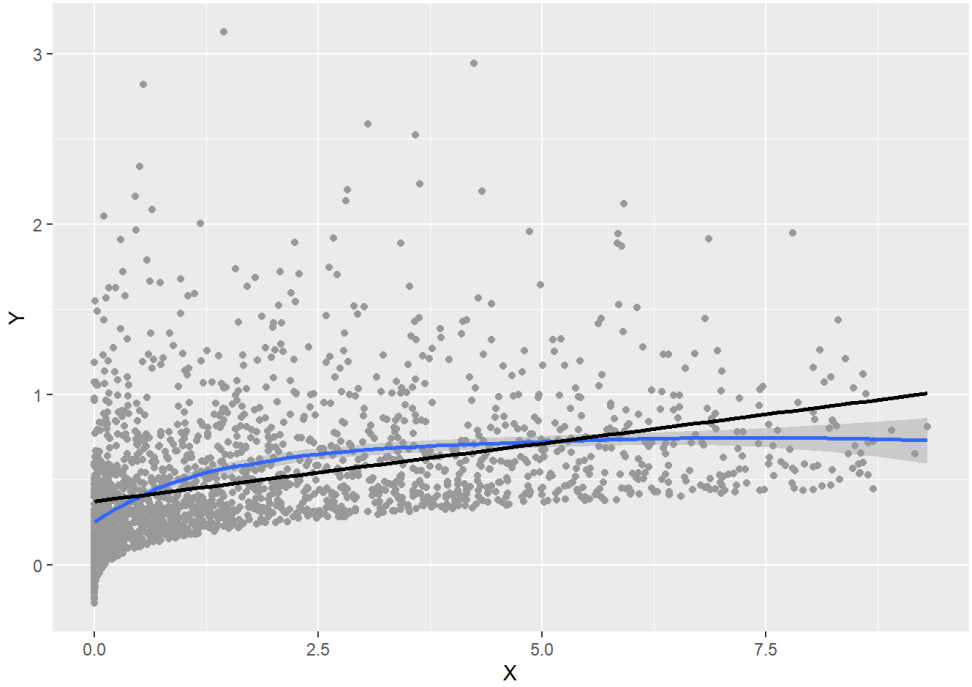
Plus de précision où les données ne sont pas trop dispersées et ou la densité est importante. Donc plus de précision dans l'intervalle [0+dela , 5] La précision diminue à droite… Dans un voisinage de 0 l'estimation n'est pas précise non plus: difficulté d'estimer où g(x) = 0

2. Reconstruction de r(x)

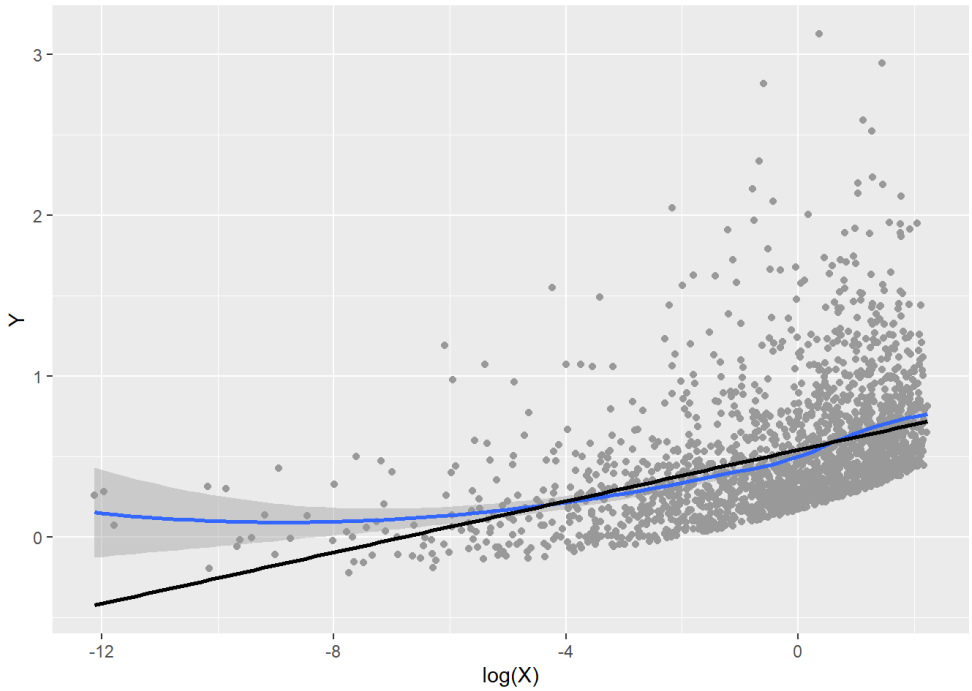
On est dans le cadre de l'estimation non paramétrique, on reprend les hypothèses classiques On utilise les données de Data1, (X,Y)

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 retouve la quasi totalité de l'échantillon, la transfoarmtion (log(x)) a permis de bien linéariser.

2.2. Construction d'un estimateur non-parametrique 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**

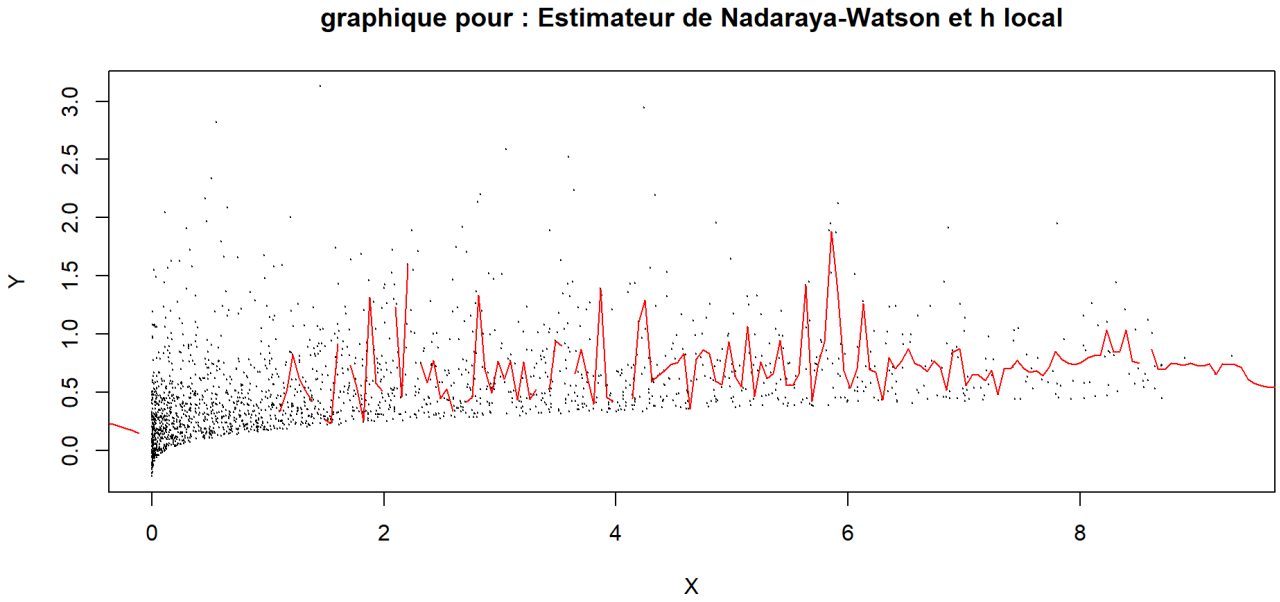
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.005019 0.014504 0.036759 0.059364 0.078701 0.173141
```

- **Test: choix de hglobal, à l'aide de la fonction dpill**

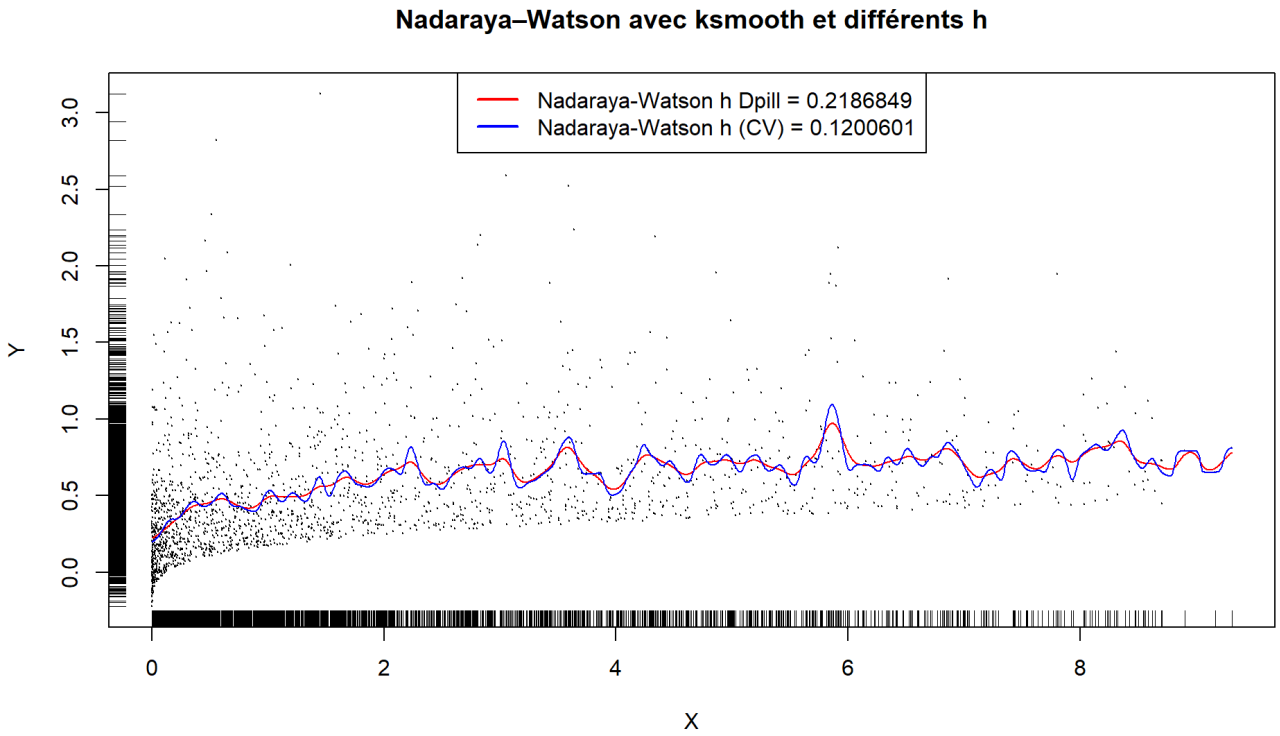
```
## [1] 0.2186849
```

Estimateur de Nadaraya-Watson : avec un h_local

h_local calculé aux différents points de grid.

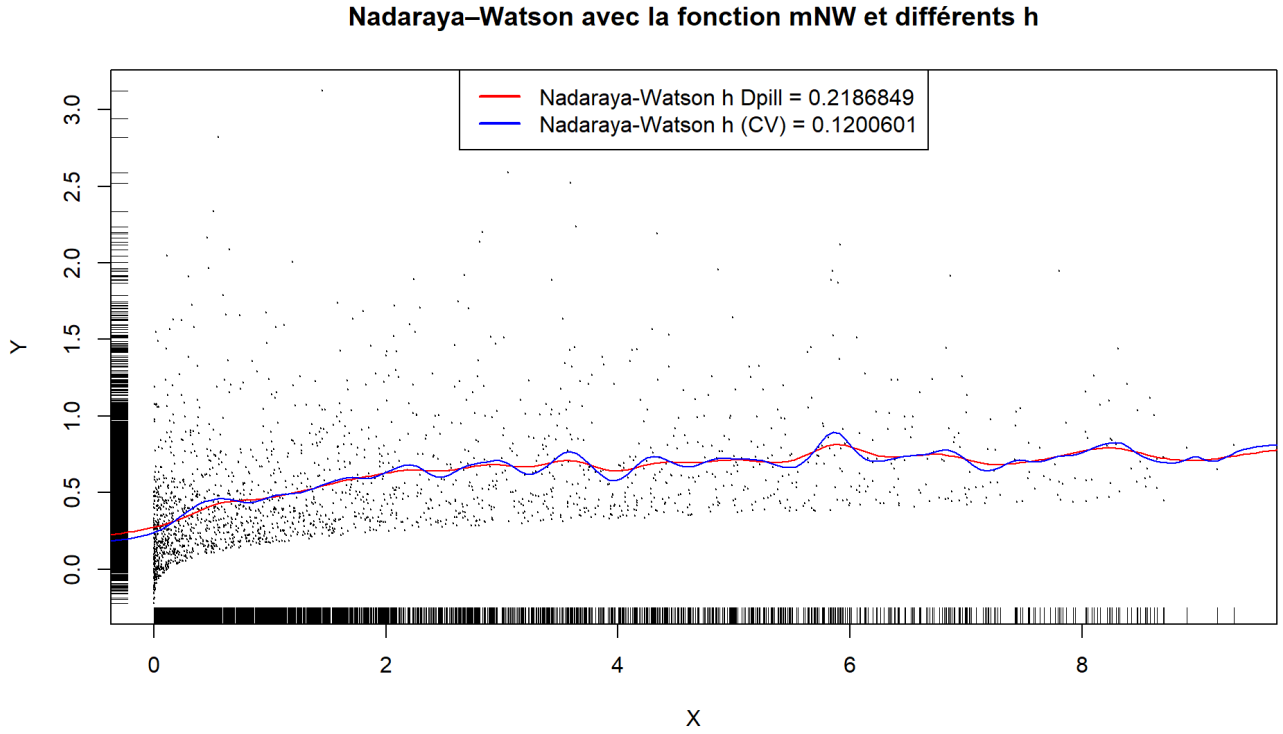


Estimateur de Nadaraya-Watson avec la librairie stat - fonction : ksmooth - h = 0.2186849 et h = 0.1200601



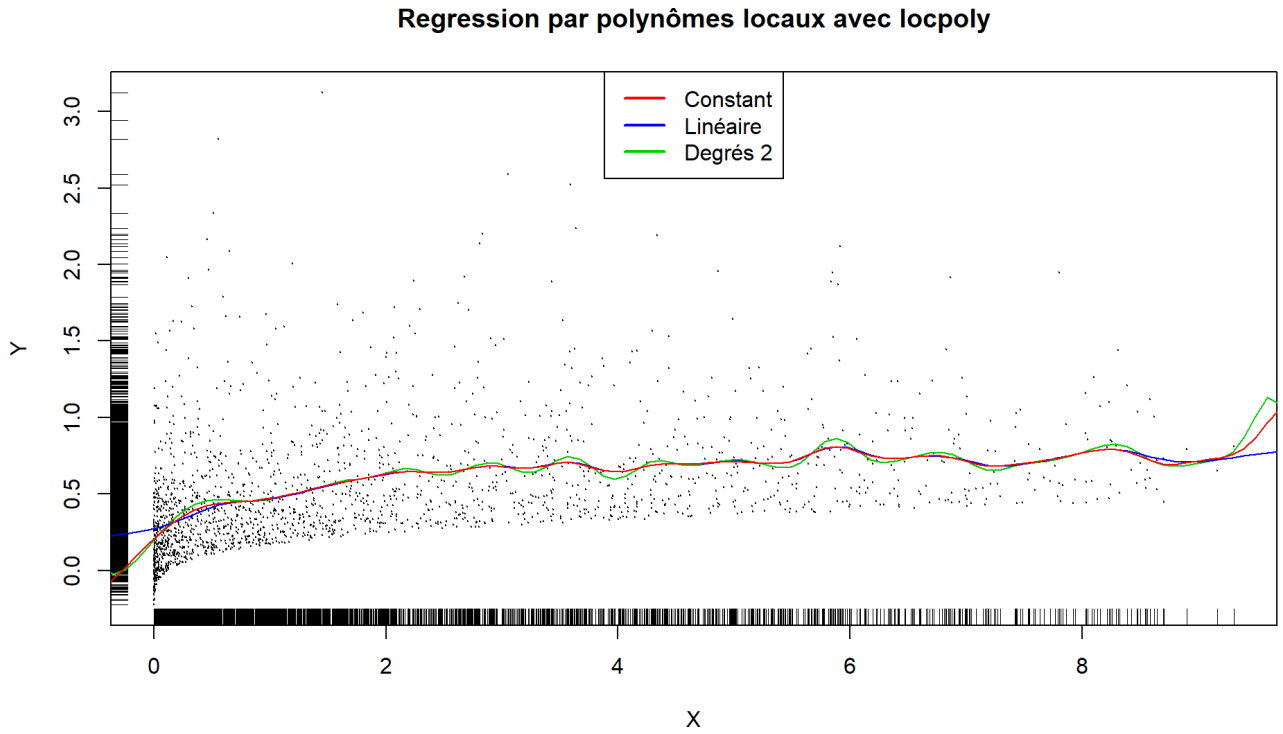
L'estimateur est sensible au choix de h. L'estimateur de Nadaraya-Watson est très oscillant par construction.

Estimateur de Nadaraya-Watson à partir de la fonction recodé NW - h = 0.2186849 et h = 0.1200601

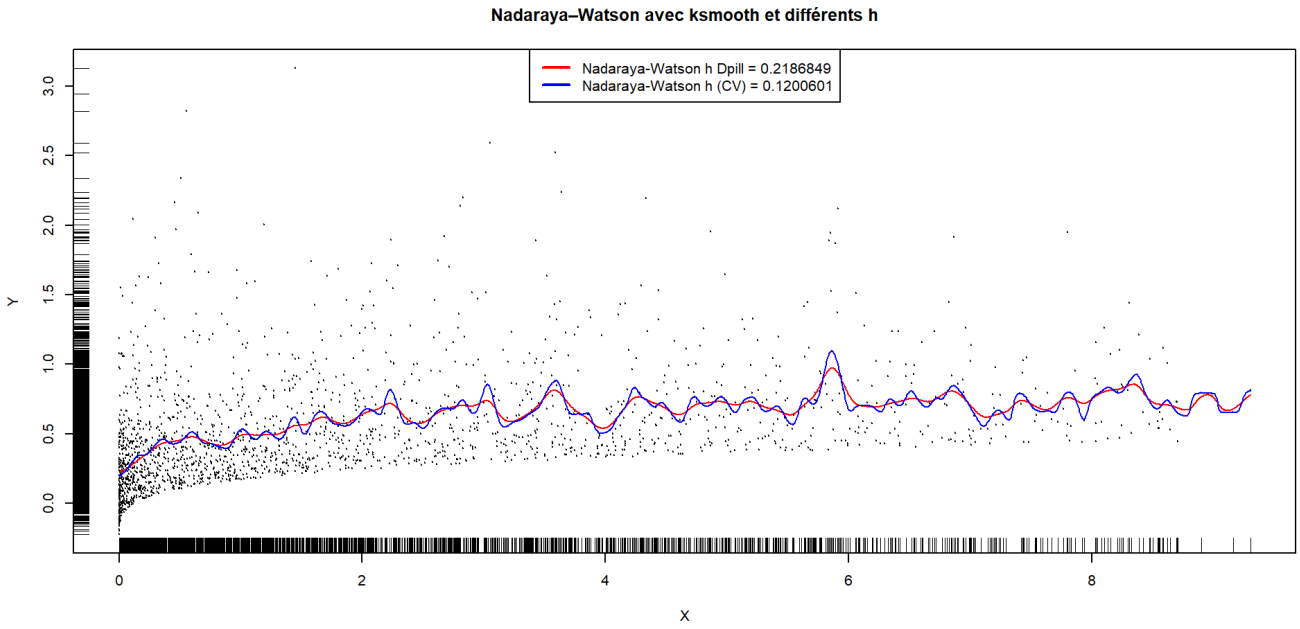


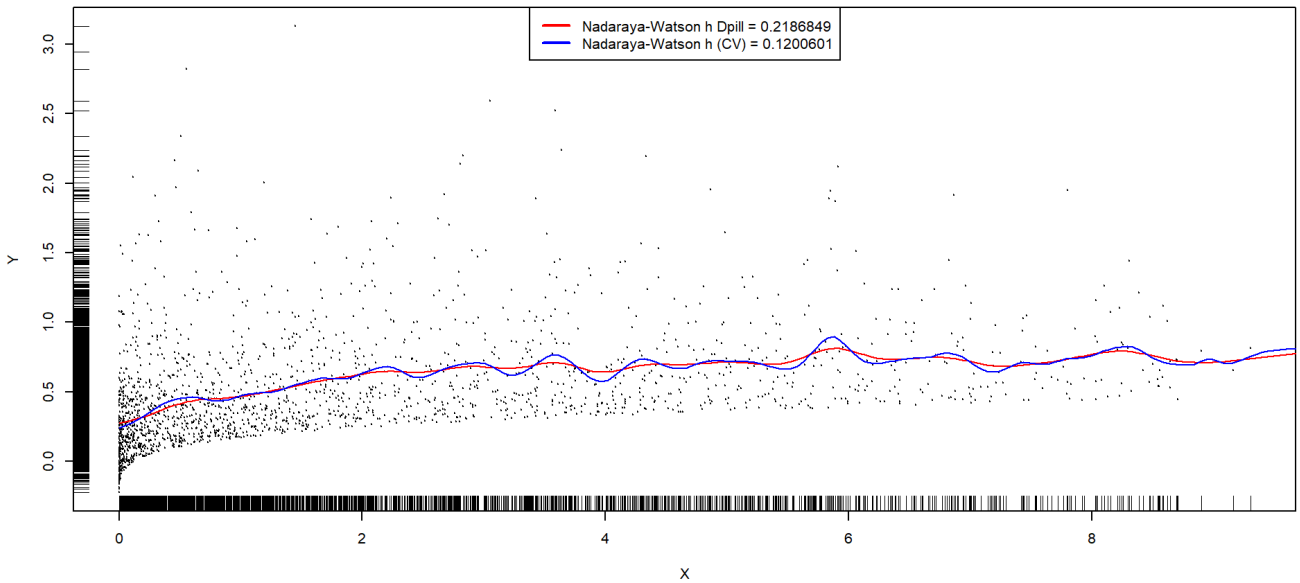
Estimateur par polynômes locaux

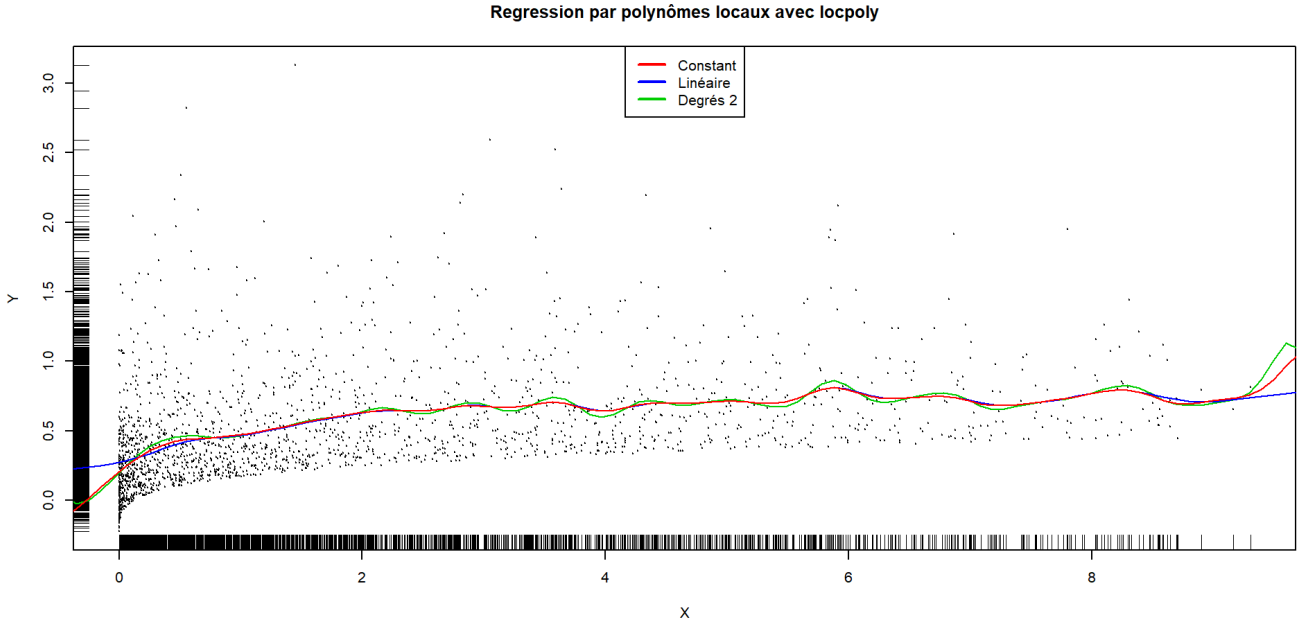
Utilisation de la fonction locpoly du package Kersmooth avec h = 0.2186849 Attention à ordonner l'échantilon (en X) pour garder la cohérence des couples (X,Y), car sinon KernSmooth va réordonner les X sans tenir compte de la composante Y.



Comparaison des différentes méthodes





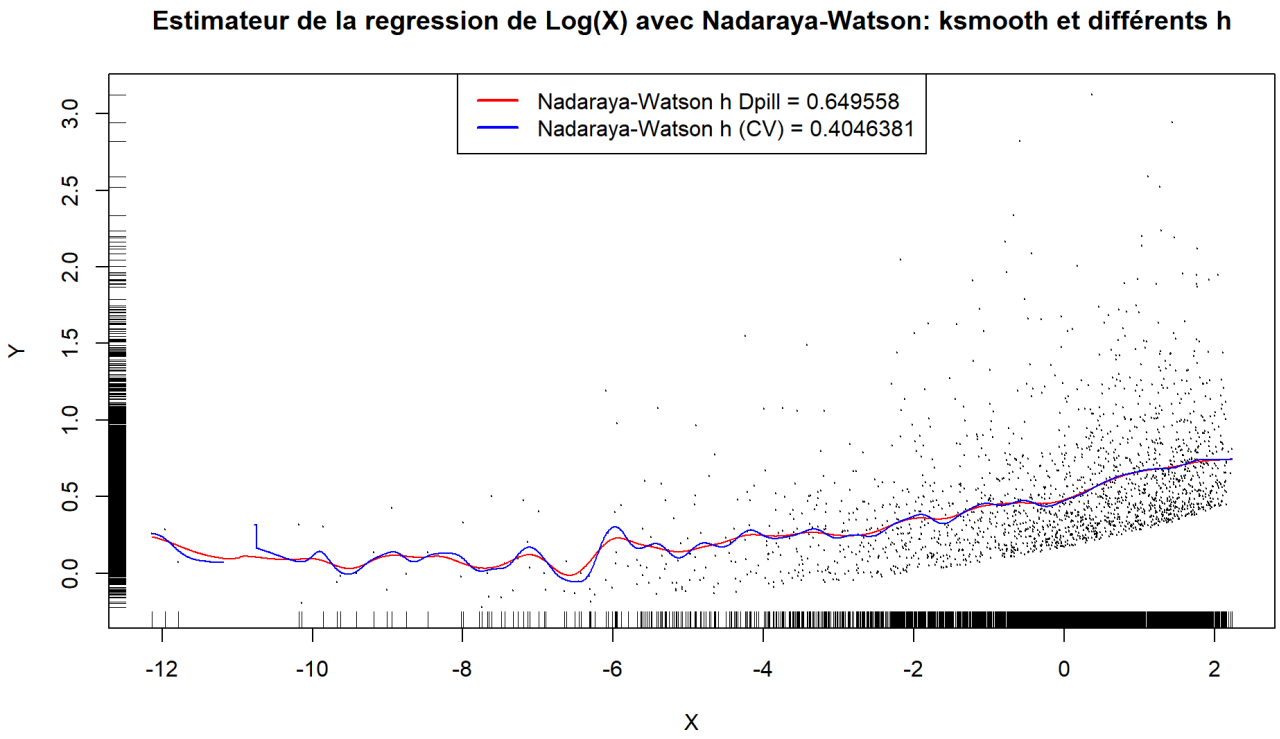


2.3. estimation de r en regressant Y1 sur log(X)

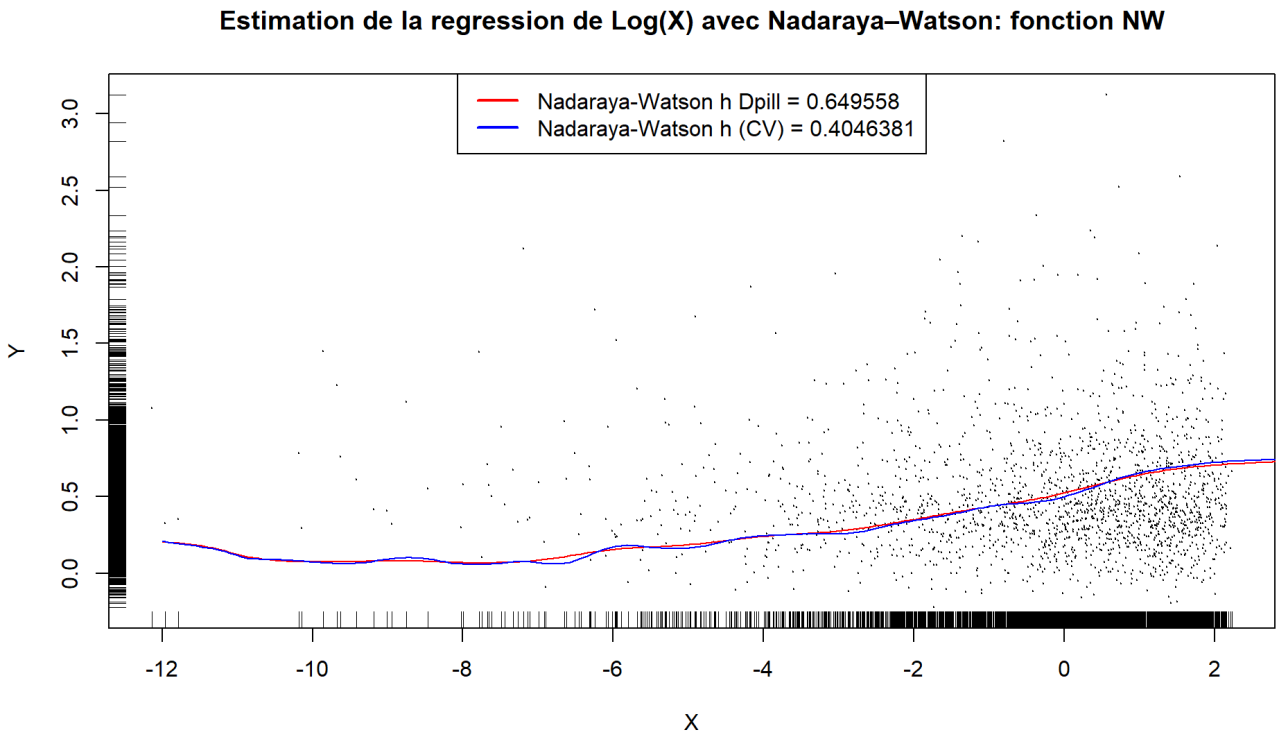
```
## [1] 0.649558
```

Find optimum CV bandwidth, with sensible grid

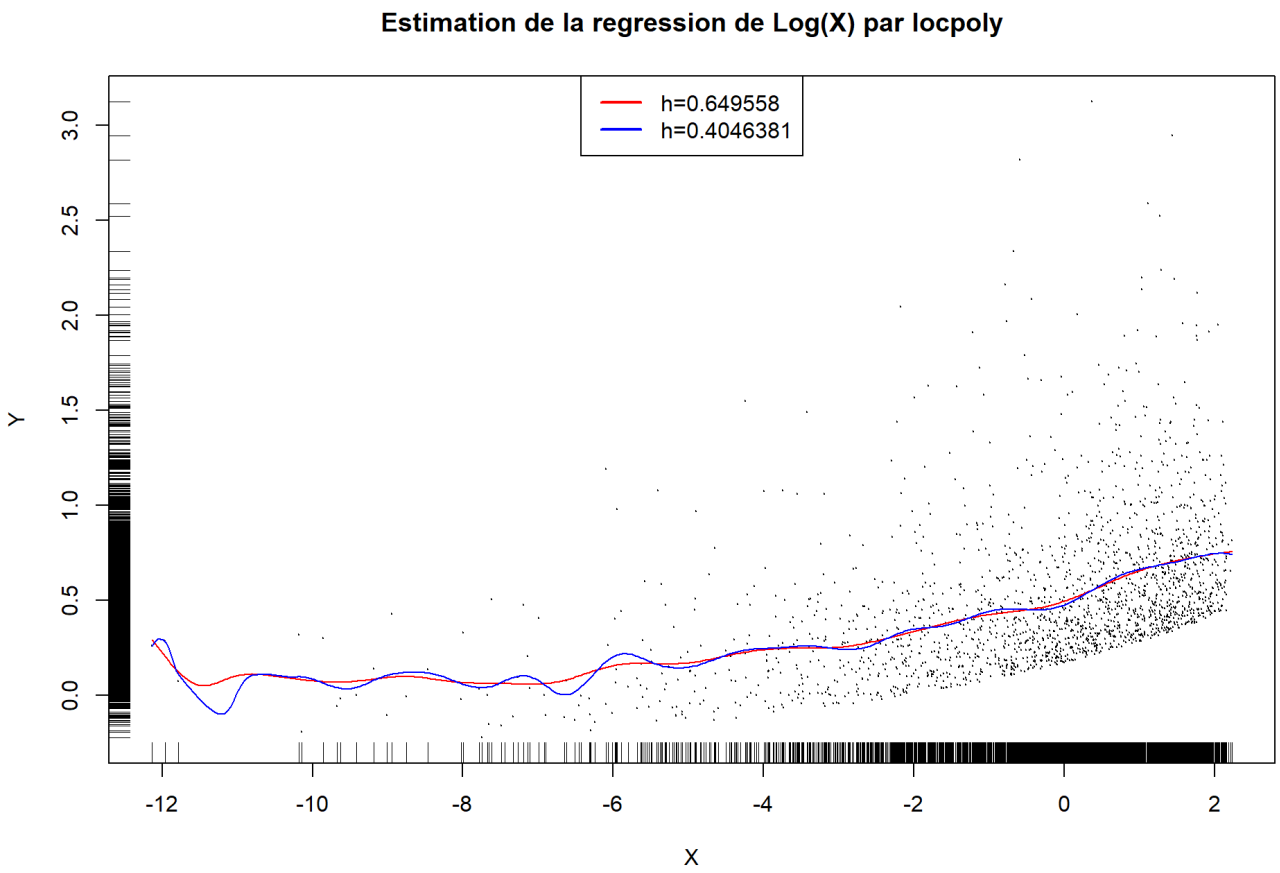
Estimation de la régression de Log(X) avec Nadaraya-Watson avec la librairie stat - fonction : ksmooth - h = 0.649558 et h = 0.4046381



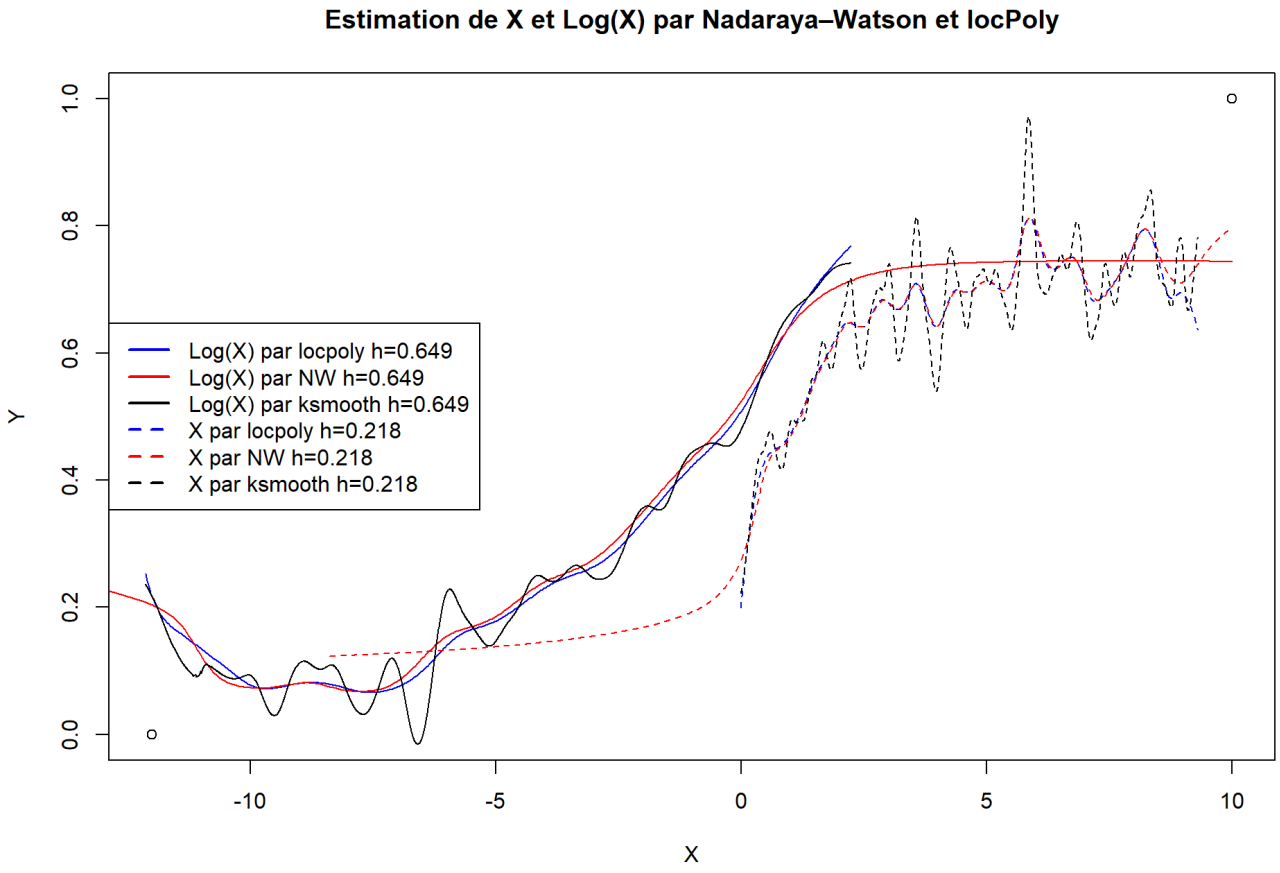
Estimateur de la régression de Log(X) avec Nadaraya-Watson à partir de la fonction recodé NW - h = 0.649558 et h = 0.4046381



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



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



2.4. Remarques? Explications?

Régularité proportionalité?

3. Etude de la densitée μ(x) des ξi

3.1 A partir du jeu de donnees Data1

3.1.1.

La distribution approximative de ~ξi est celle de ξi

3.1.2 Représentation de la densité μ(x) des ξi

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

```
## [1] 0.004196189
```

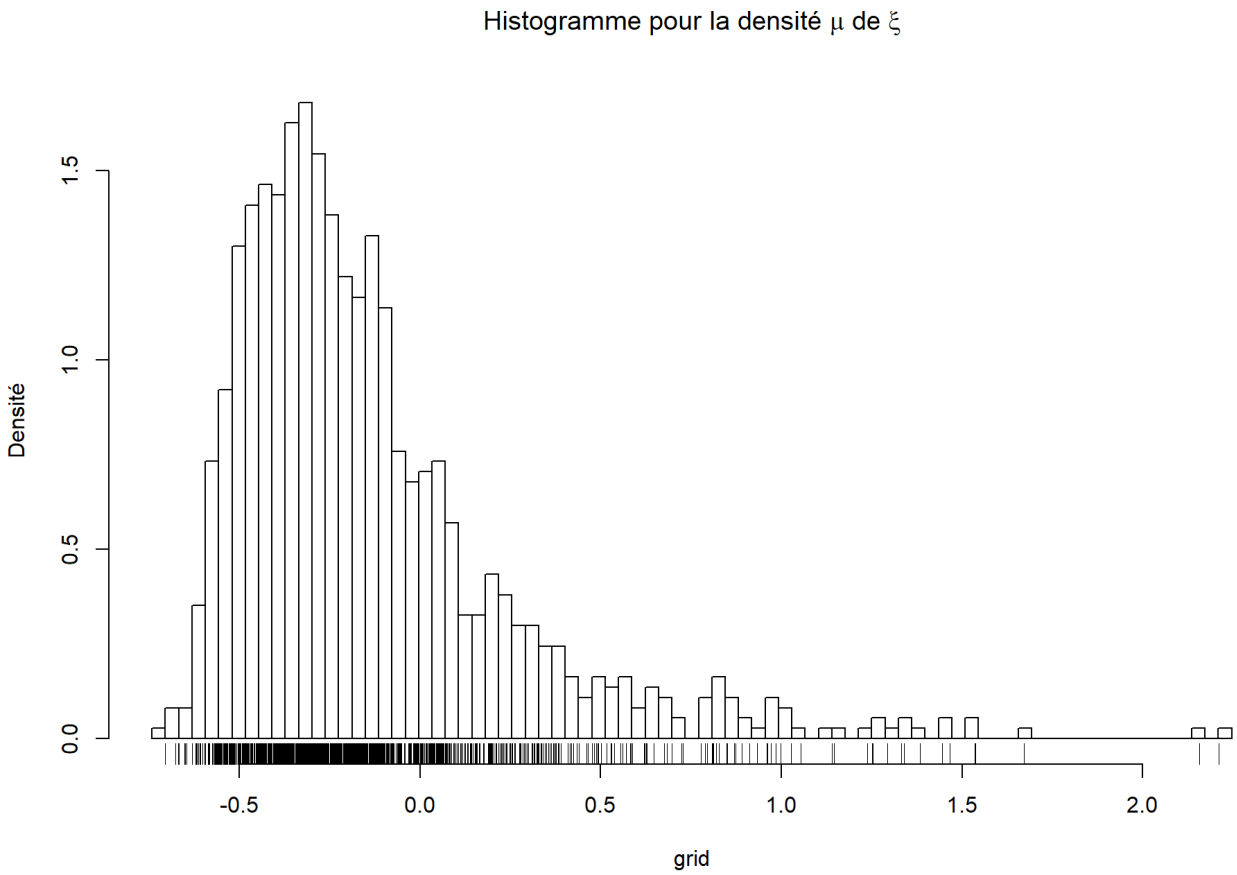
```
## [1] 0.03026436
```

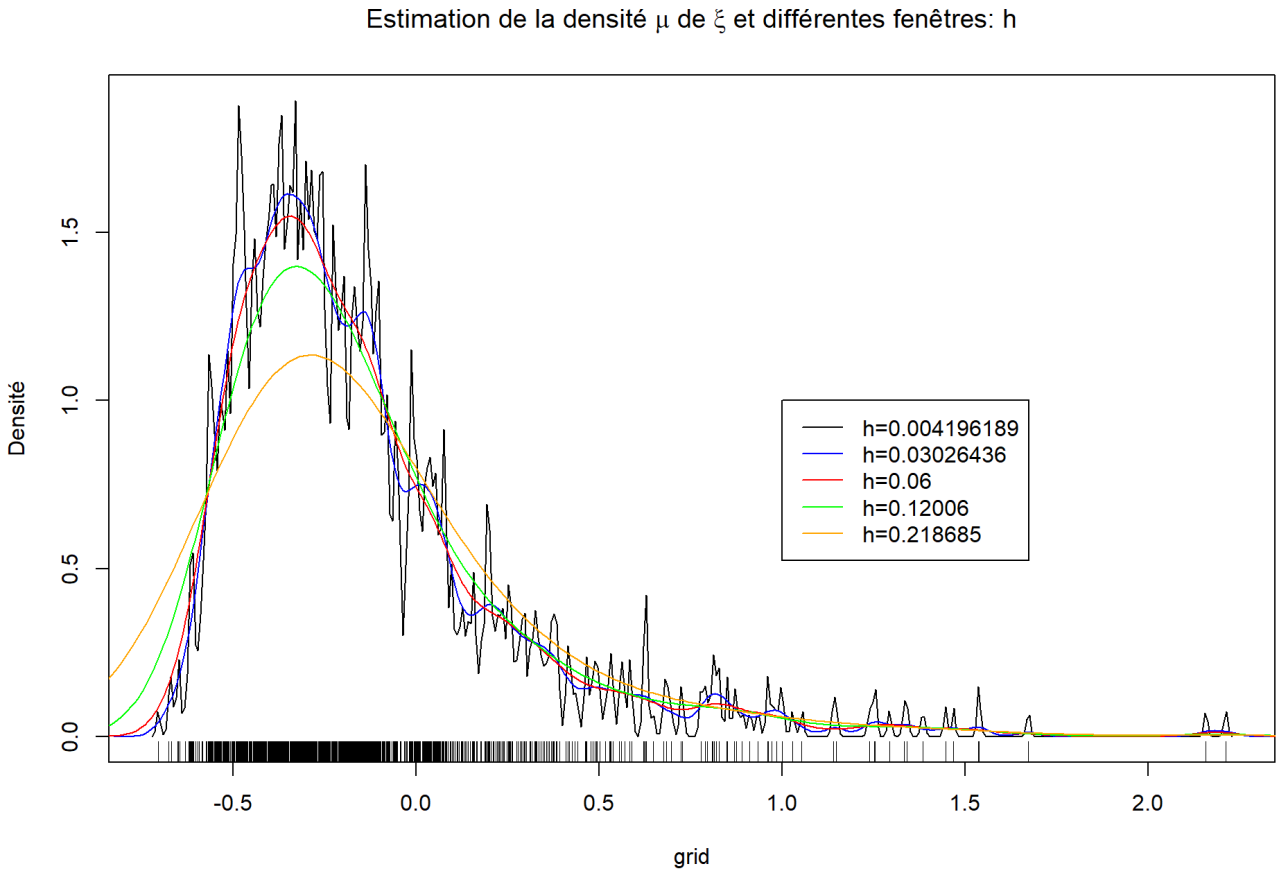
```
h_ucv
```

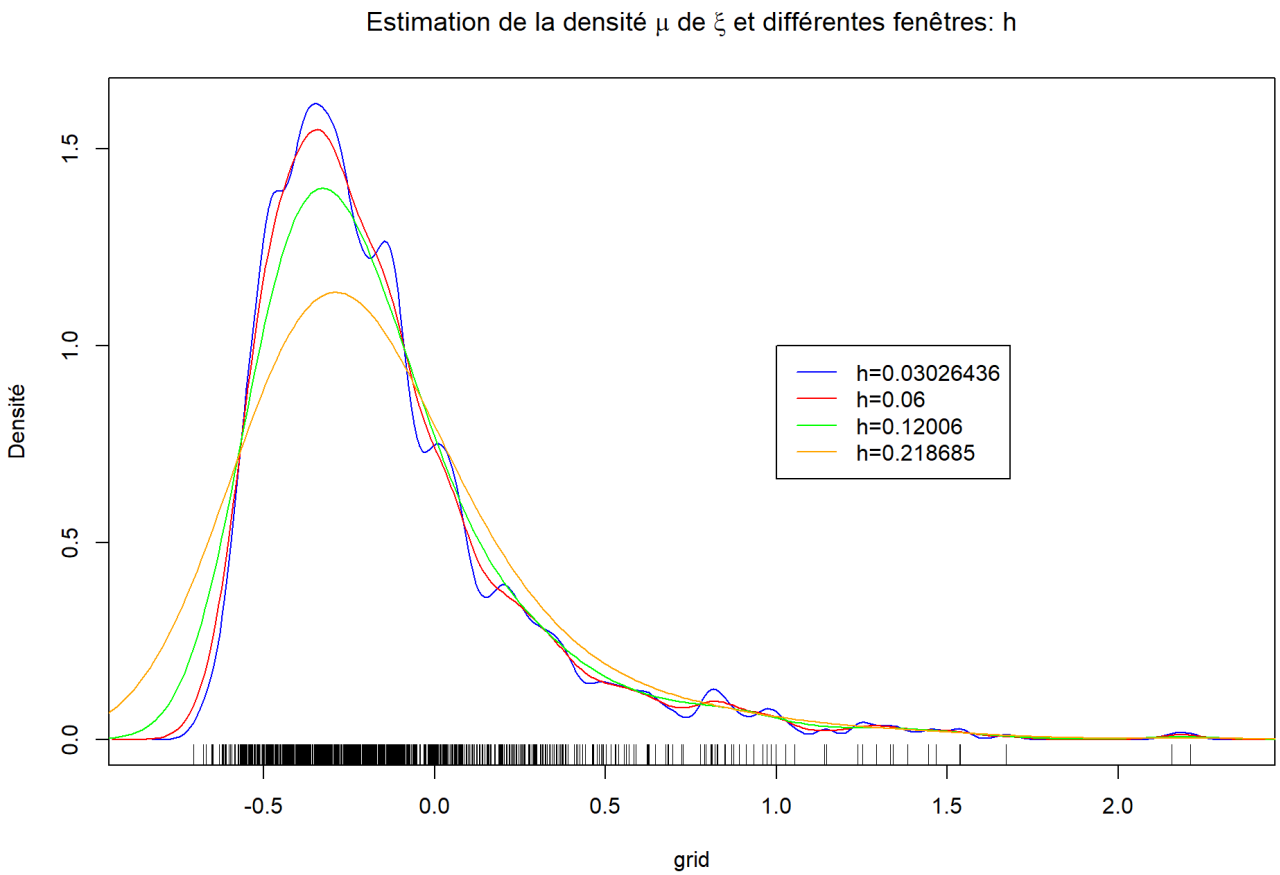
```
## [1] 0.004196189
```

```
h_sil
```

```
## [1] 0.03026436
```







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

3.1.3. (Facultatif.) Quel est l'interêt d'avoir decoupé le jeu de donnees selon J+ et J- ?

Données d'apprentissage et données de test. On peut utilser le jeux de données d'apprentissage pour estimer notre lois, et le jeux de test

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

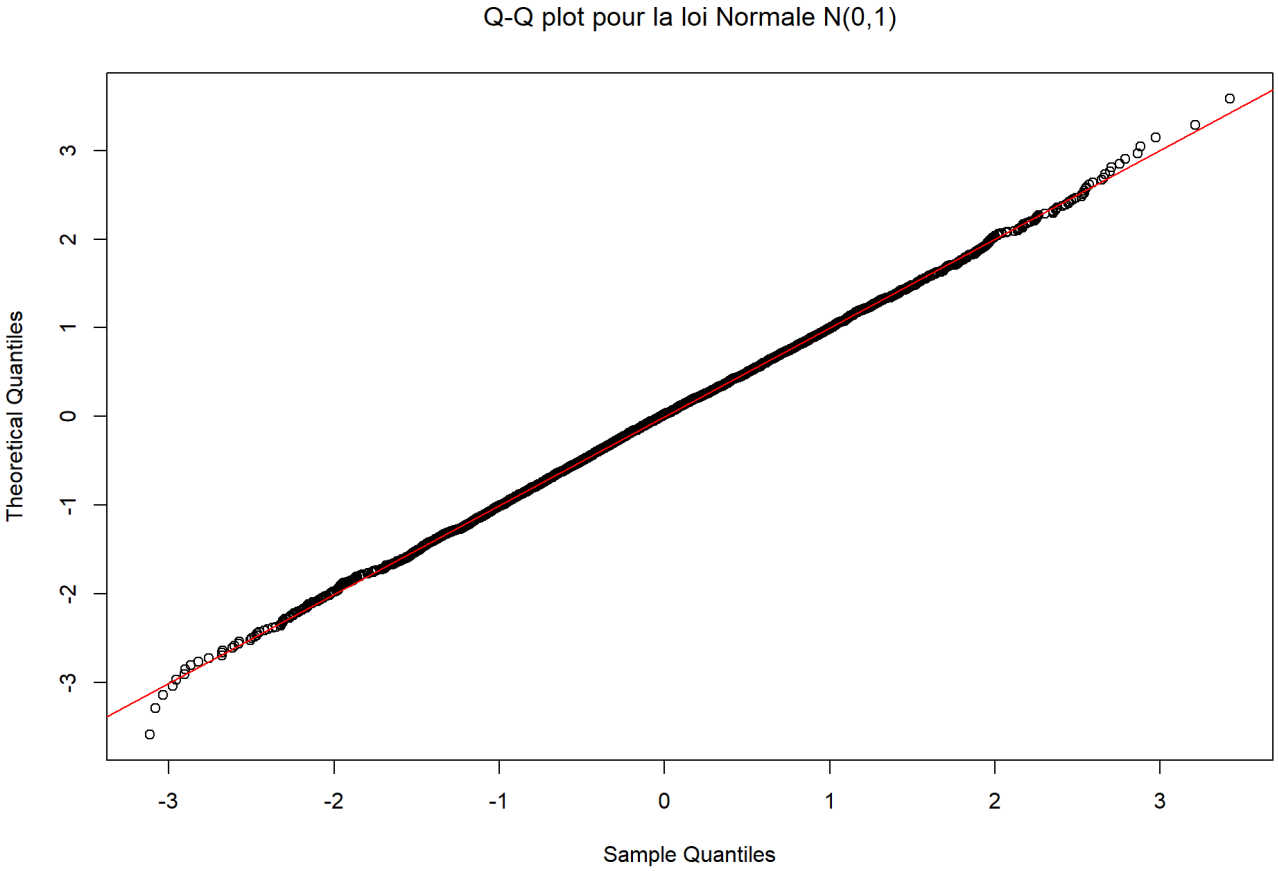
```
## [1] 0.3622163
```

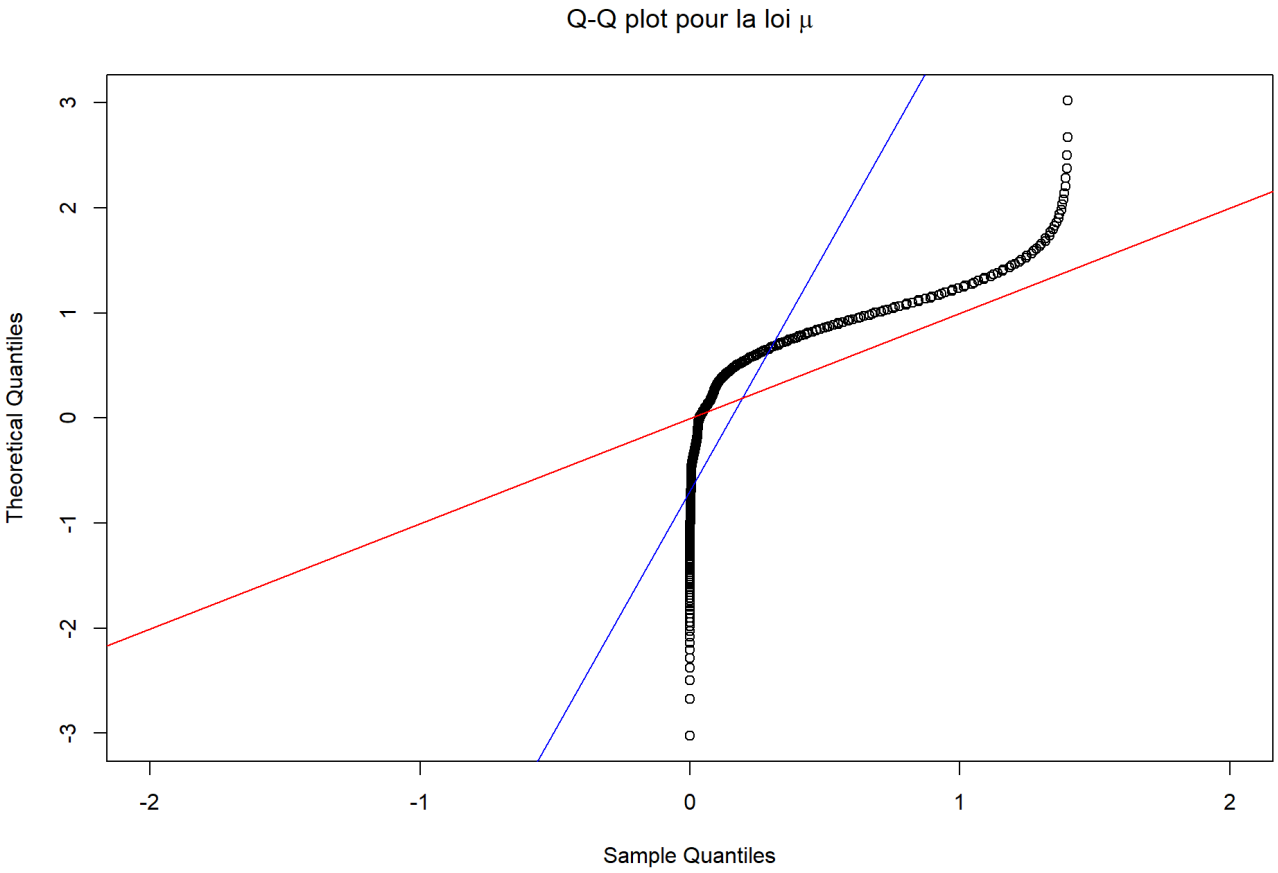
```
## [1] 0.3622163
```

3.1.4. La densitée μ peut-elle être gaussienne ?

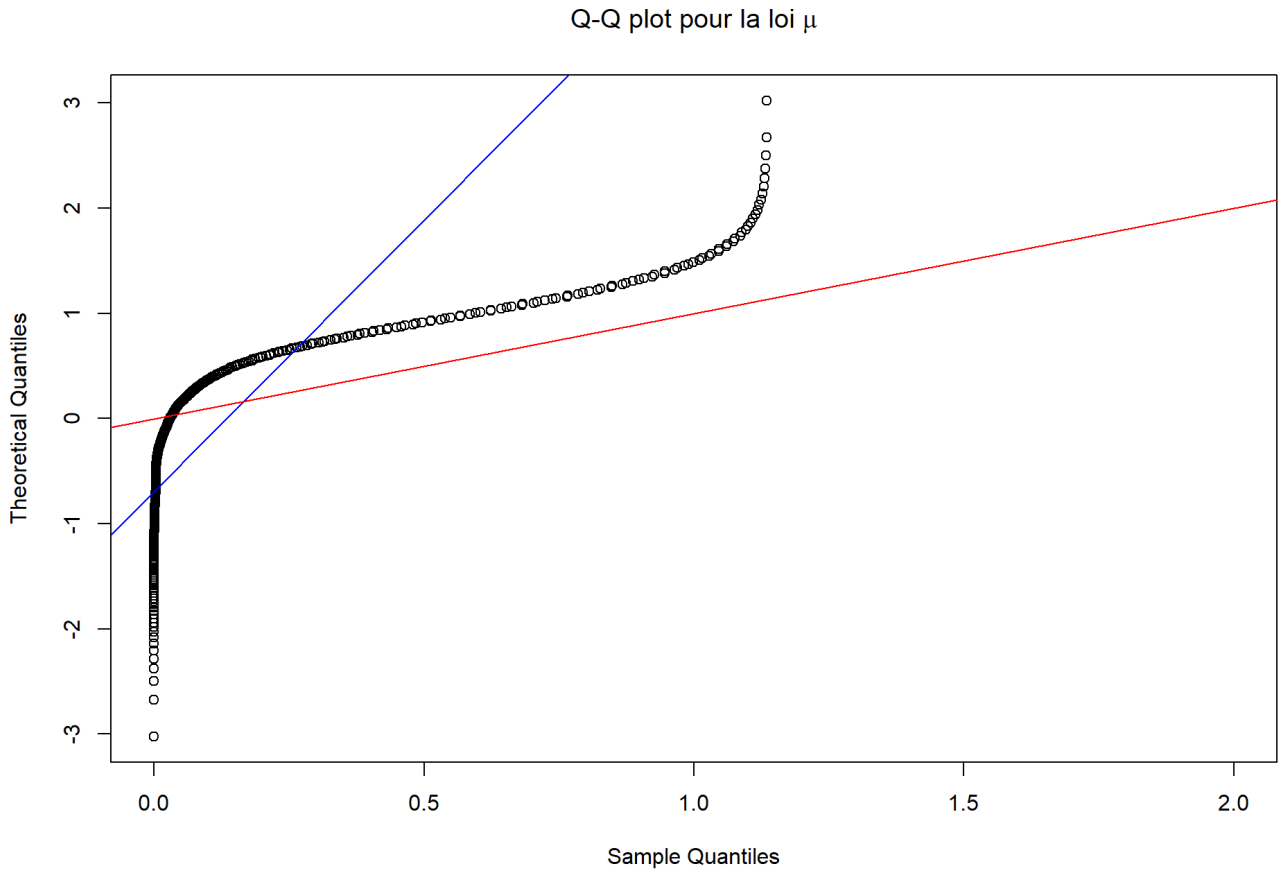
Protocole empirique de verification http://www.biostat.ulg.ac.be/pages/Site_r/Normalite.html (http://www.biostat.ulg.ac.be/pages/Site_r/Normalite.html)

- **1. test du QQPlot**





=> non ou bien gaussien par morceau



- **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 = 0.88092, p-value < 2.2e-16
```

Le test de Shapiro-Wilk donne une probabilité de dépassement de p-value < 2.2e-16, nettement < à 0.05. L'hypothèse de normalité est rejettéee.

- **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 rejetté

3.1.5. homoscedasticité du modèle

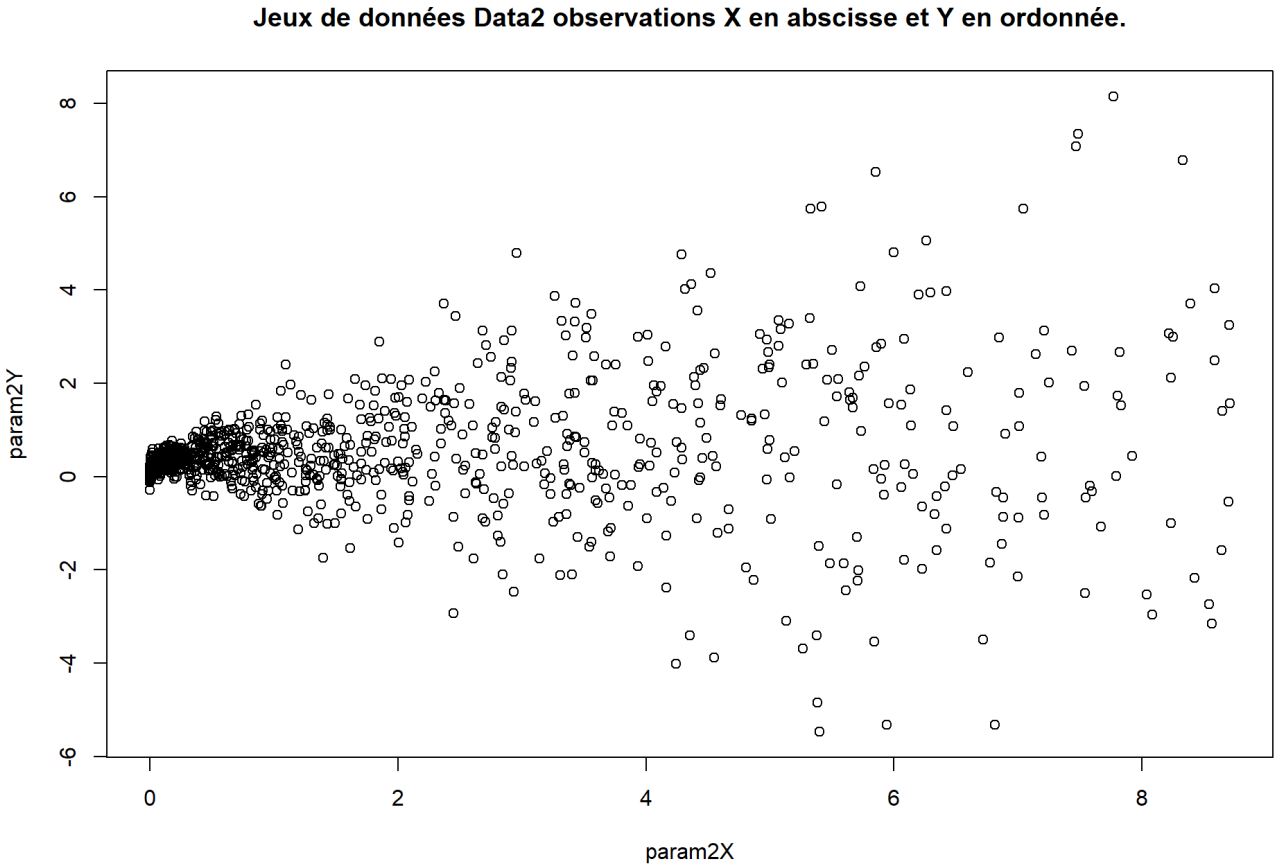
(Facultatif.) Comment peut-on tester si le modele est bien homoscedastique ?

3.2 A partir du jeu de donnees Data2

On cherche a estimer μ(x) et σ(x)^2$. Pour cela, on coupe a nouveau l'echantillon en deux et on considere a 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
```



3.2.1. Justifier qu'en regressant ξi sur Xi on obtient un estimateur de σ(x)^2$.

Implémentation et le visualisation graphique

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

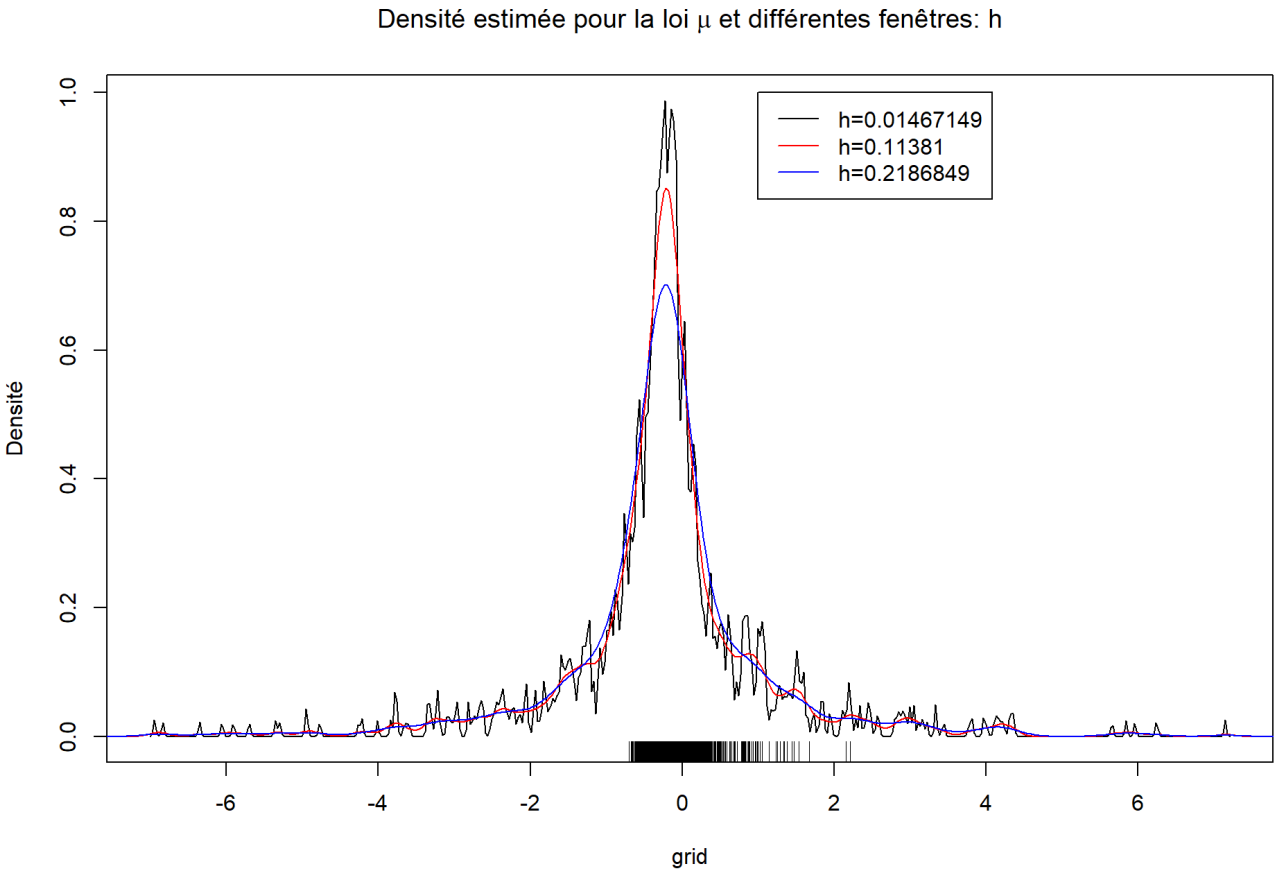
```
 #hist(fit22$y, breaks=bins32,freq = FALSE,xlab="grid",ylab="Densité", main=expression("Histogramme et estimateurs de "*sigm
a*"^2"))

 #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)
```

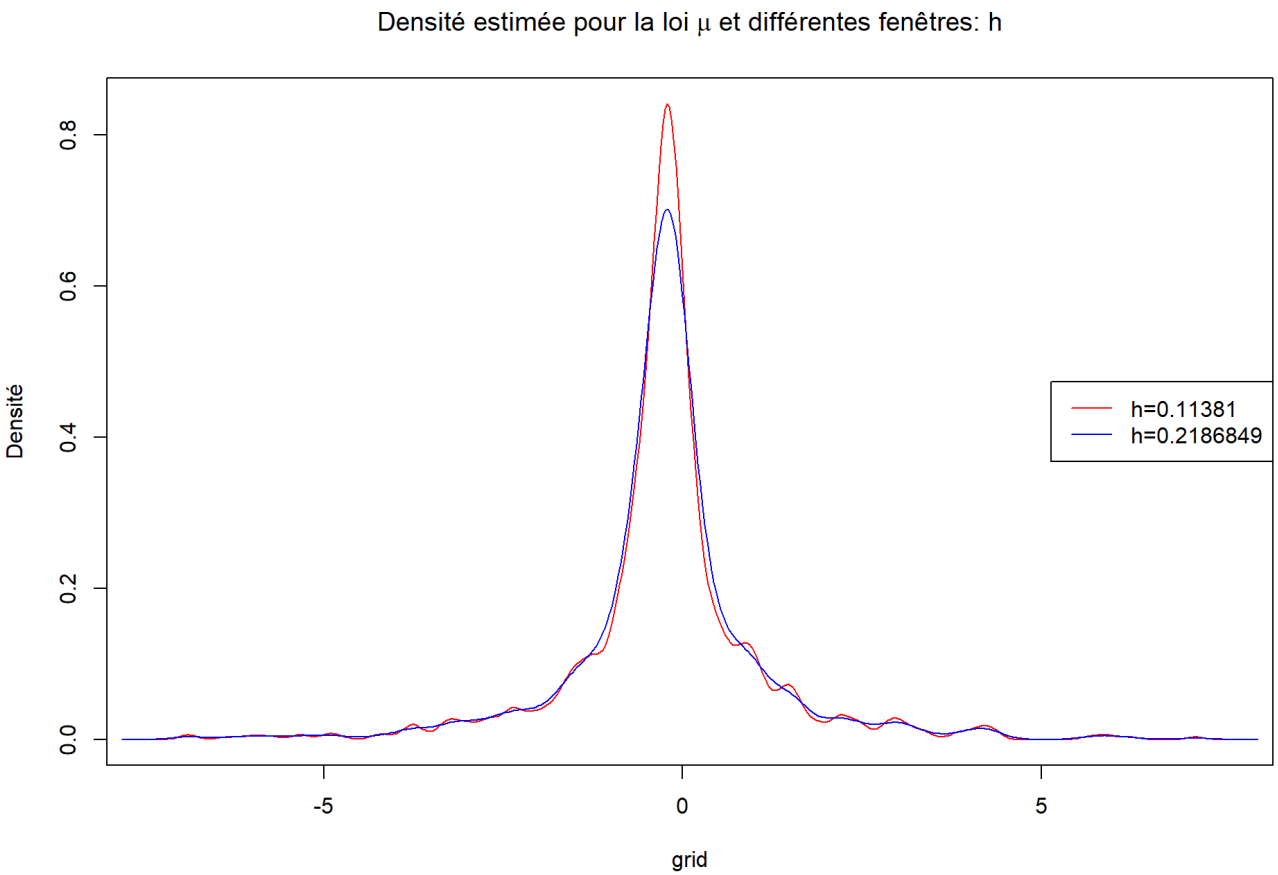
```
## [1] 0.01467149
```

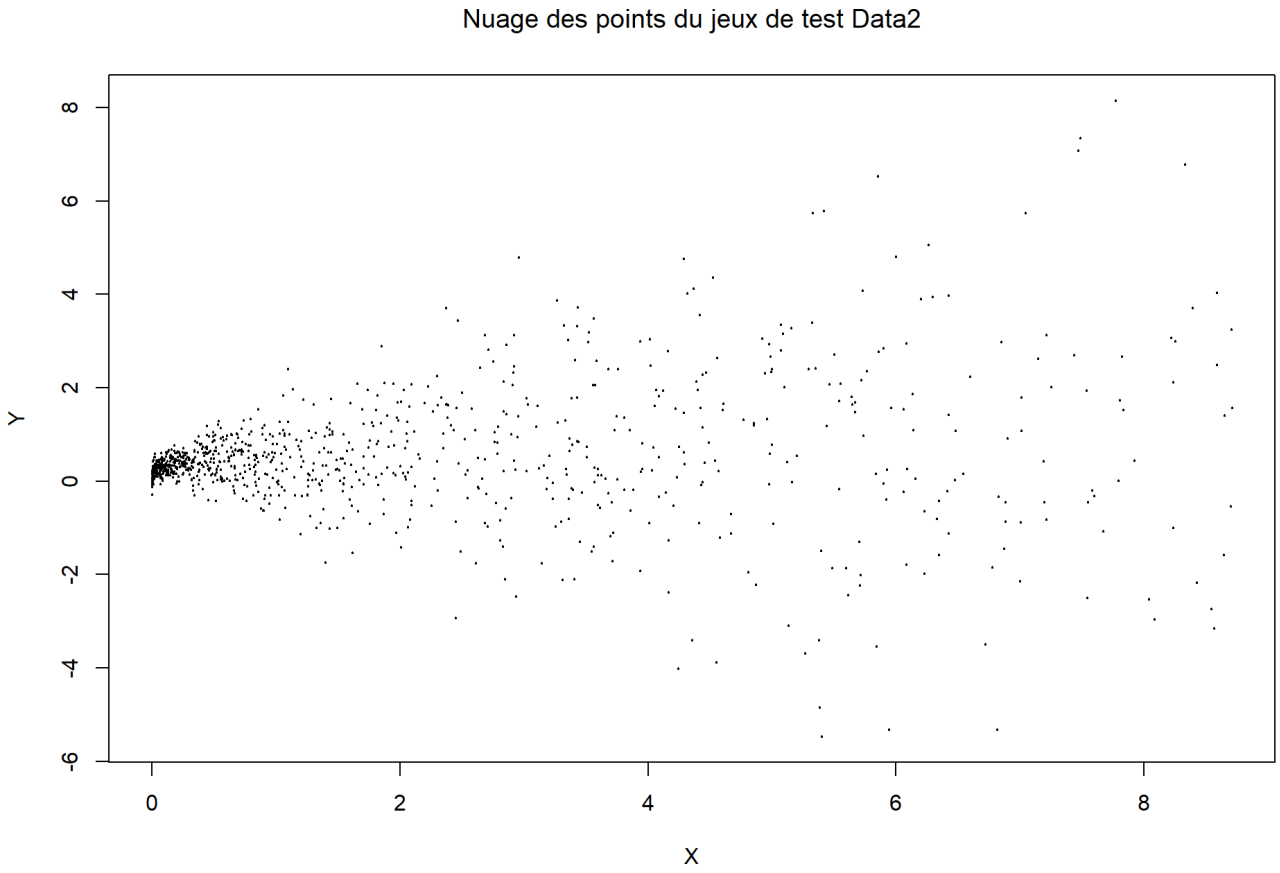
```
## [1] 0.11381
```

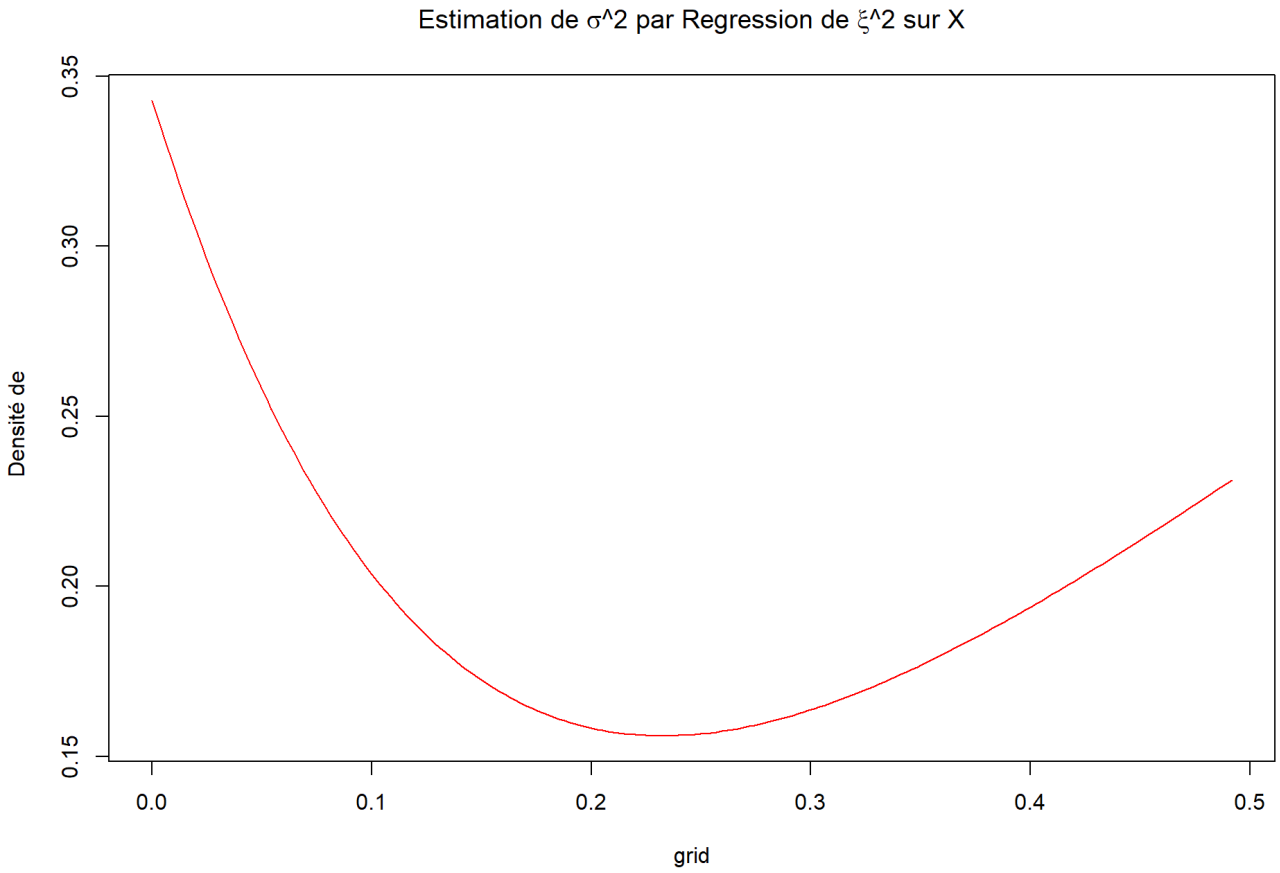
```
## [1] 0.1200601
```

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


Densité estimée pour la loi μ et différentes fenêtres: h
h=0.01467149
h=0.11381
h=0.2186849
Densité
grid
0.0
0.2
0.4
0.6
0.8
1.0
-6
-4
-2
0
2
4
6


Densité estimée pour la loi μ et différentes fenêtres: h
h=0.11381
h=0.2186849
Densité
grid
0.0
0.2
0.4
0.6
0.8
-5
0
5





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

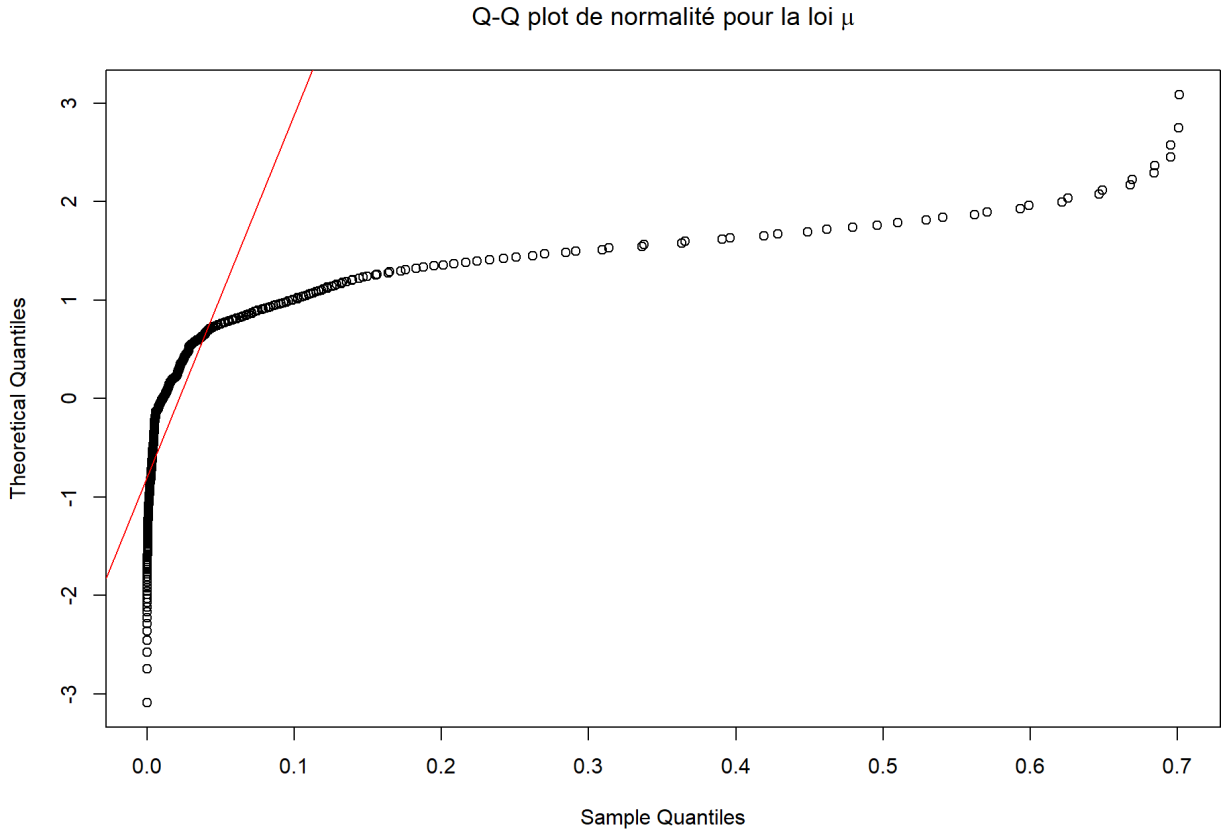
=> symétrie, centrée en 0

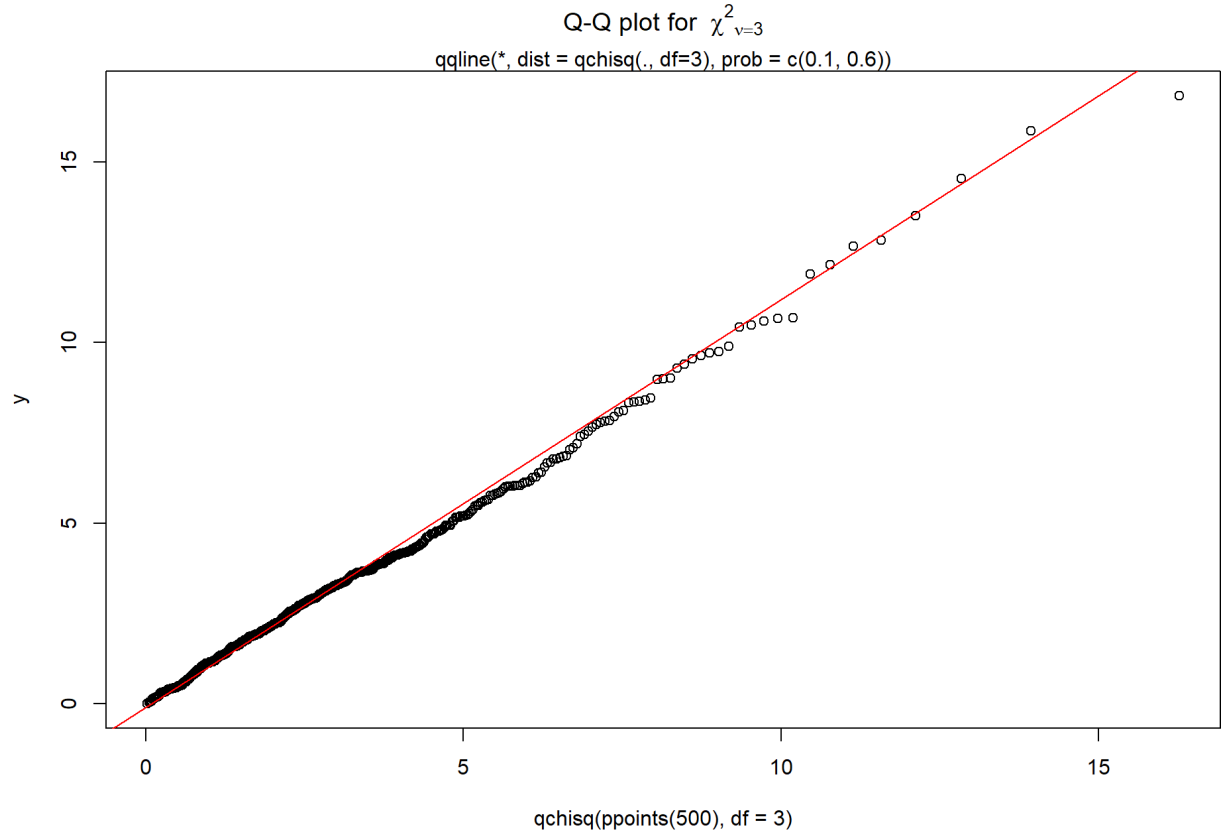
3.2.2. La densitée x->μ(x) peut-elle etre gaussienne ?

Proposer un protocole pour le verifier empiriquement et l'implementer. On pourra penser a renormaliser ksi par la fonction estimee a la question precedente et s'aider des questions de la Section 3.1.

http://www.biostat.ulg.ac.be/pages/Site_r/Normalite.html (http://www.biostat.ulg.ac.be/pages/Site_r/Normalite.html)

- **1- test du QQPlot**





=> non ou bien gaussien par morceau

- **2- test de shapiro**

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

```
##
##  Shapiro-Wilk normality test
##
## data:  mu_hDpill$y
## W = 0.49198, p-value < 2.2e-16
```

Le test de Shapiro-Wilk donne une probabilité de dépassement de p-value < 2.2e-16, nettement < à 0.05. L'hypothèse de normalité est rejettéee.

- **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:**

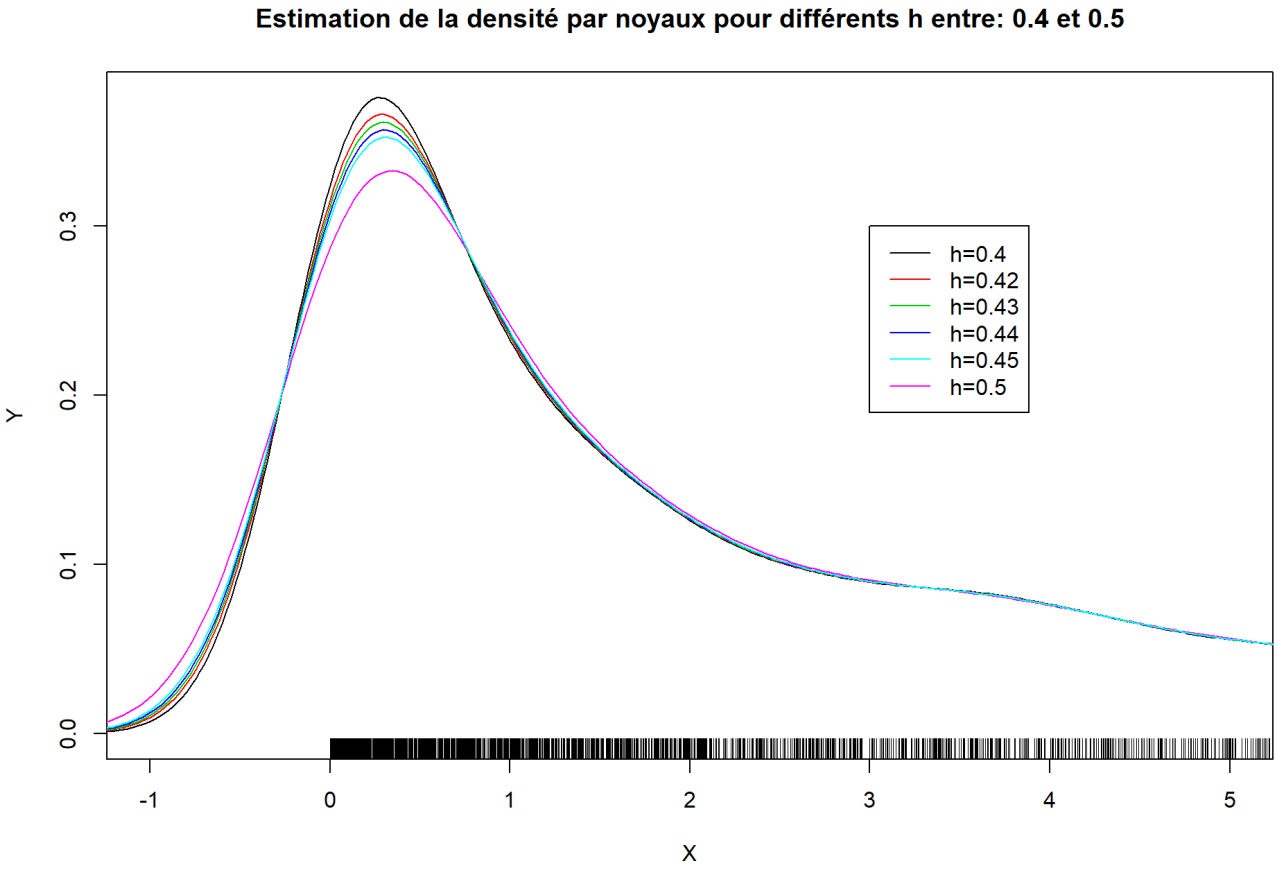
```
ks.test(mu_hDpill$y,"pnorm",mean(mu_hDpill$y),sd(mu_hDpill$y))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  mu_hDpill$y
## D = 0.32364, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

p-value faible rejetté

Raffinement pour h: 0.4 - 0.5

bandwidth h	error quadratic	R2	Biais	MAE	RMSE
0.40	8.5686374	-0.8011561	-1.9496624	1.9715207	2.9272235
0.42	8.5707968	-0.80161	-1.9506701	1.9719415	2.9275923
0.43	8.5718552	-0.8018324	-1.9511644	1.9721651	2.9277731
0.44	8.5729051	-0.8020532	-1.9516526	1.9723901	2.9279524
0.45	8.5739505	-0.8022729	-1.9521347	1.9726206	2.9279524
0.50	8.5792099	-0.8033785	-1.9544584	1.9737791	2.9290288



Les estimations de la tensité 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.

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