Non Parametric Statistic : Estimation de densité avec les données geyser

NIANG Mohamed

6 janvier 2020

Contents


```
library(MASS)
```

1 Description

La base de données geyser de la librairie MASS contient des données d'éruption (temps d'attente et durée) de l'Old Faithful geyser du Yellowstone National Park.

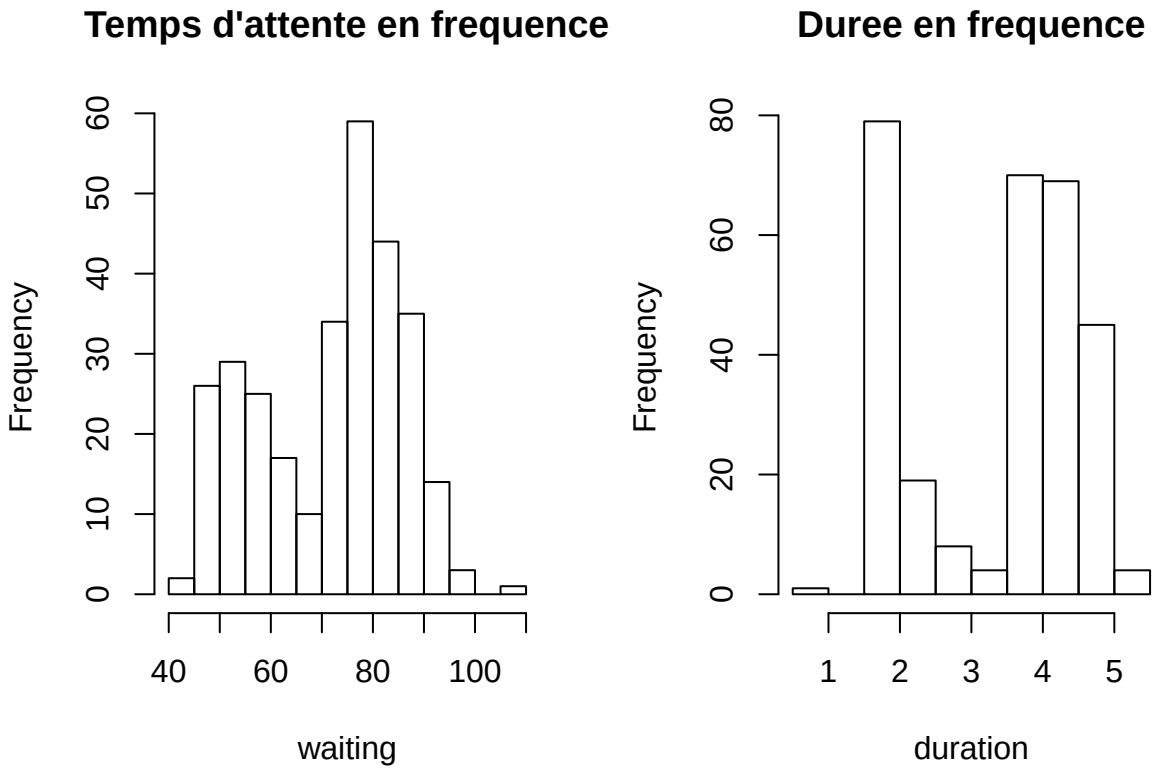
2 Histogramme

La fonction hist permet d'obtenir un histogramme de la distribution des observations.

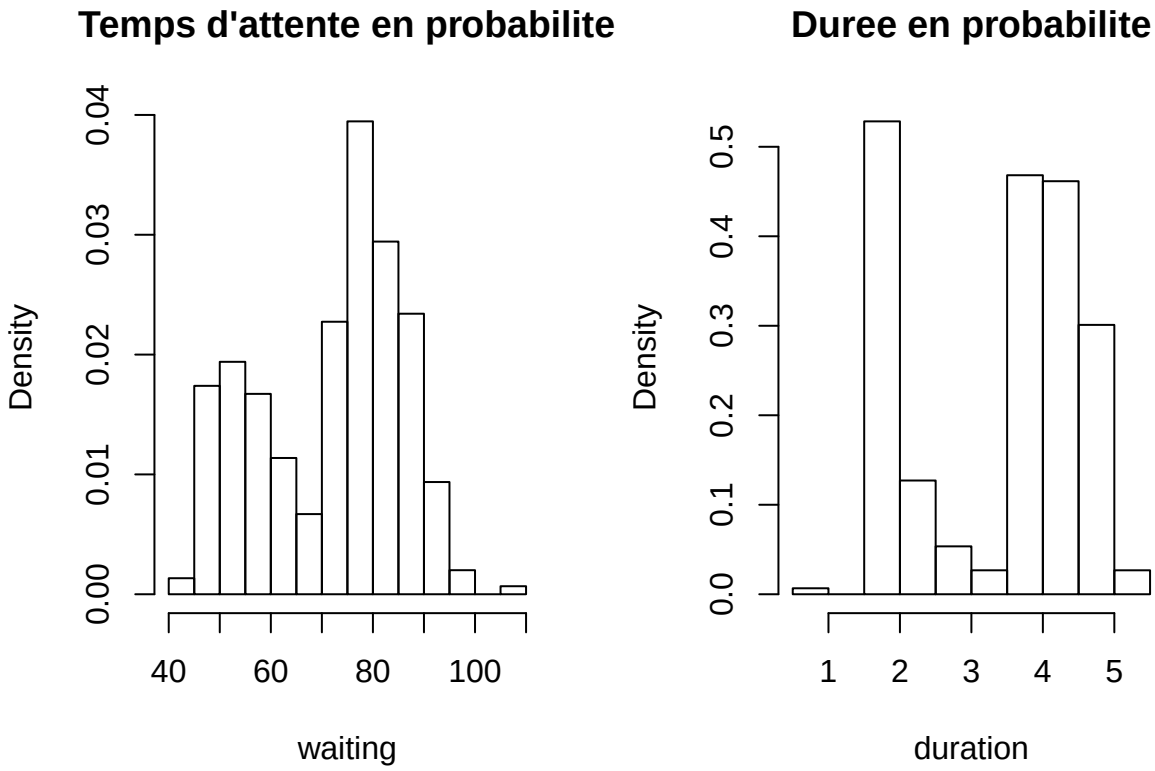
2.1 Histogramme en fréquence puis en proba des observations

```
gdata <- geyser
waiting <- gdata[,1]
duration <- gdata[,2]
```

```
par(mfrow = c(1, 2))
hist(waiting, proba = F, main = "Temps d'attente en frequence")
hist(duration, proba = F, main = "Duree en frequence")
```

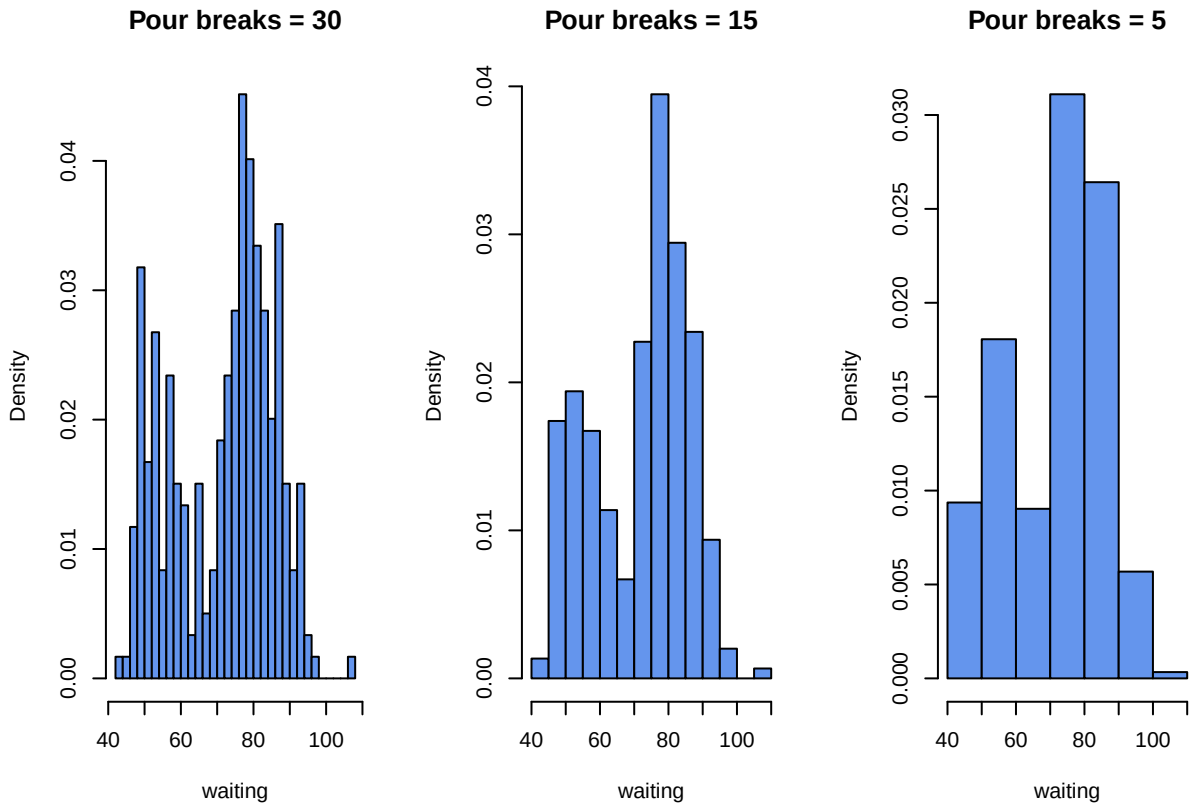



```
par(mfrow = c(1, 2))
hist(waiting, proba = T, main = "Temps d'attente en probabilite")
hist(duration, proba = T, main = "Duree en probabilite")
```



2.2 Variation de l'argument breaks et constat

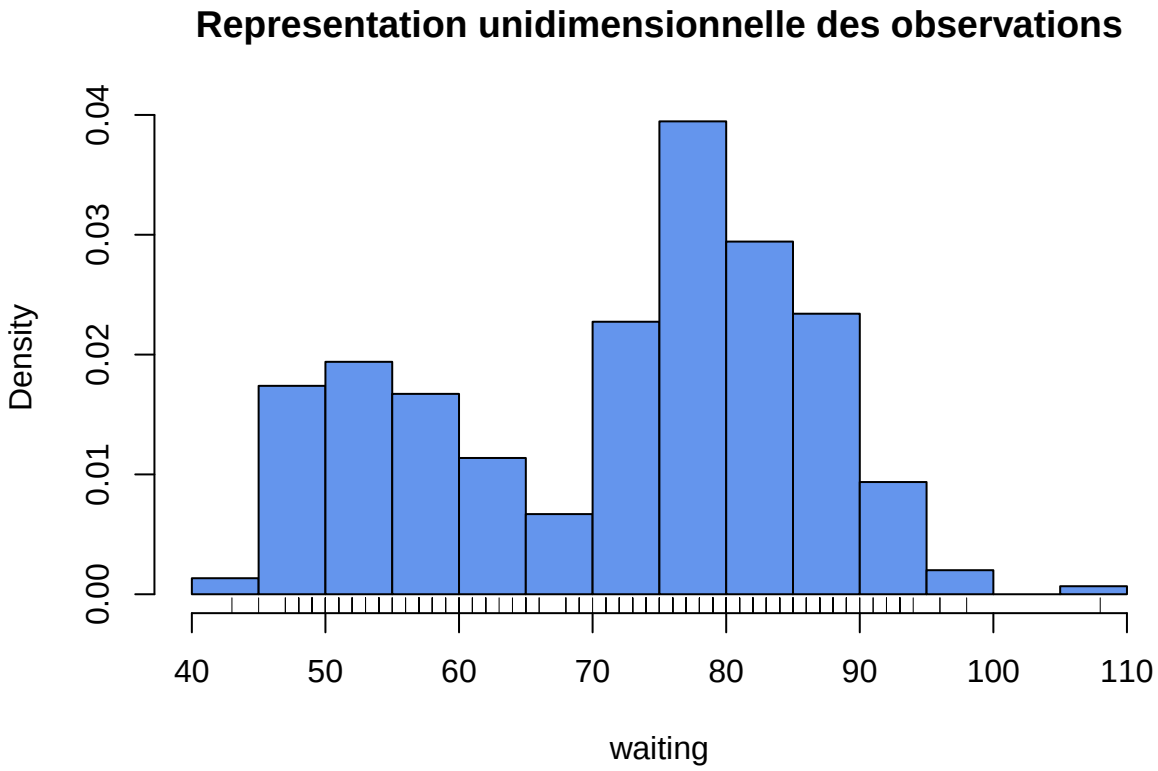
```
par(mfrow = c(1,3))
hist(waiting, col = "cornflowerblue", proba = T, breaks = 30, main = "Pour breaks = 30")
hist(waiting, col = "cornflowerblue", proba = T, breaks = 15, main = "Pour breaks = 15")
hist(waiting, col = "cornflowerblue", proba = T, breaks = 5, main = "Pour breaks = 5")
```



Commentaires : Une grande breaks entraine une mauvaise estimation et une petite beaks fait perdre de l'informations. Ainsi pour avoir une bonne estimation, il faut prendre le juste milieu.

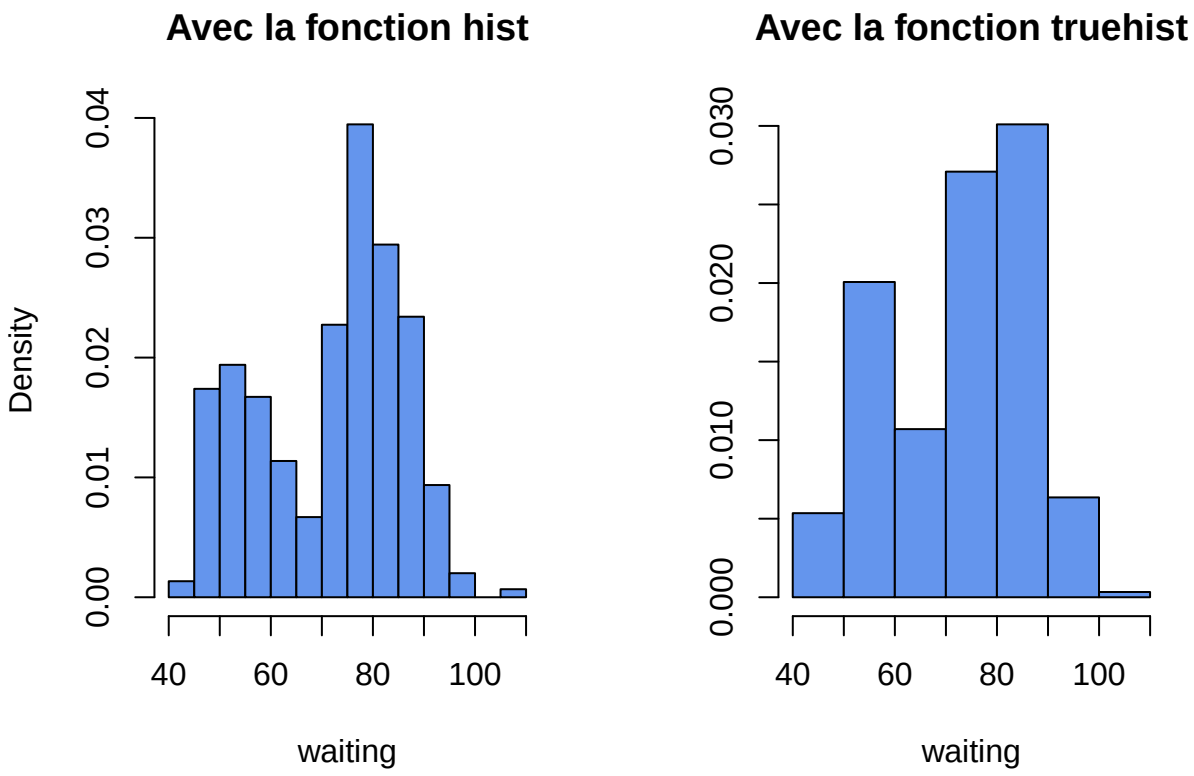
2.3 Représentation unidimensionnelle des observations

```
hist(waiting, col = "cornflowerblue", proba = T, breaks = 10, main = "Representation unidimensionnelle
rug(waiting)
```



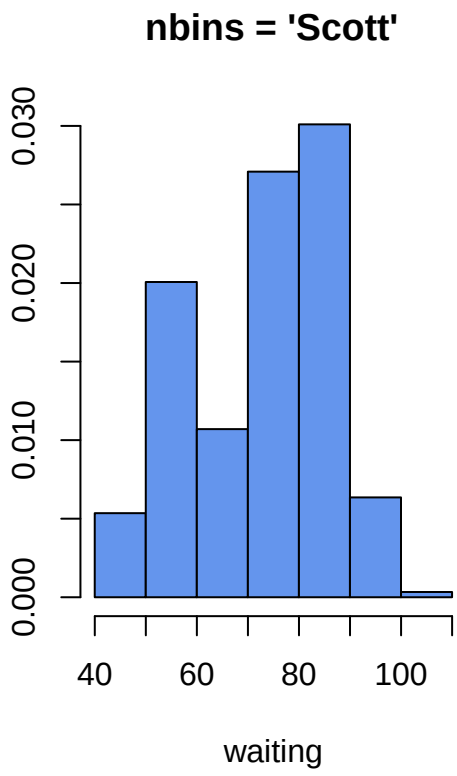
2.4 Histogramme et truehist

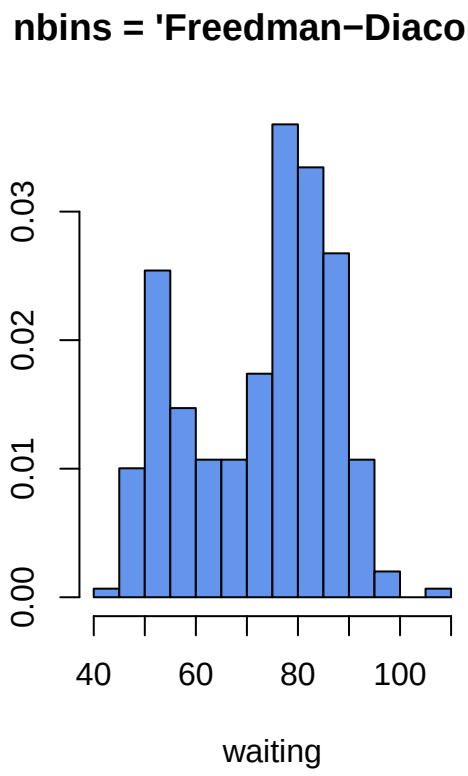
```
par(mfrow= c(1,2))
hist(waiting, col = "cornflowerblue", proba = T, main = "Avec la fonction hist")
truehist(waiting, col = "cornflowerblue", main = "Avec la fonction truehist")
```



Commentaires : La différence se trouve sur le nombre de barre utilisée et la hauteur des barres.

```
# Differente option proposee par la fonction truehist
par(mfrow= c(1,2))
truehist(waiting, col = "cornflowerblue", main = "nbins = 'Scott'")
truehist(waiting, col = "cornflowerblue", nbins = 'Freedman-Diaconis', main = "nbins = 'Freedman-Diacon
```





3 Estimateur à noyau en dimension

3.1 Observation de la structure du density

```
dens <- density(waiting)
dens$n
```

```
## [1] 299
```

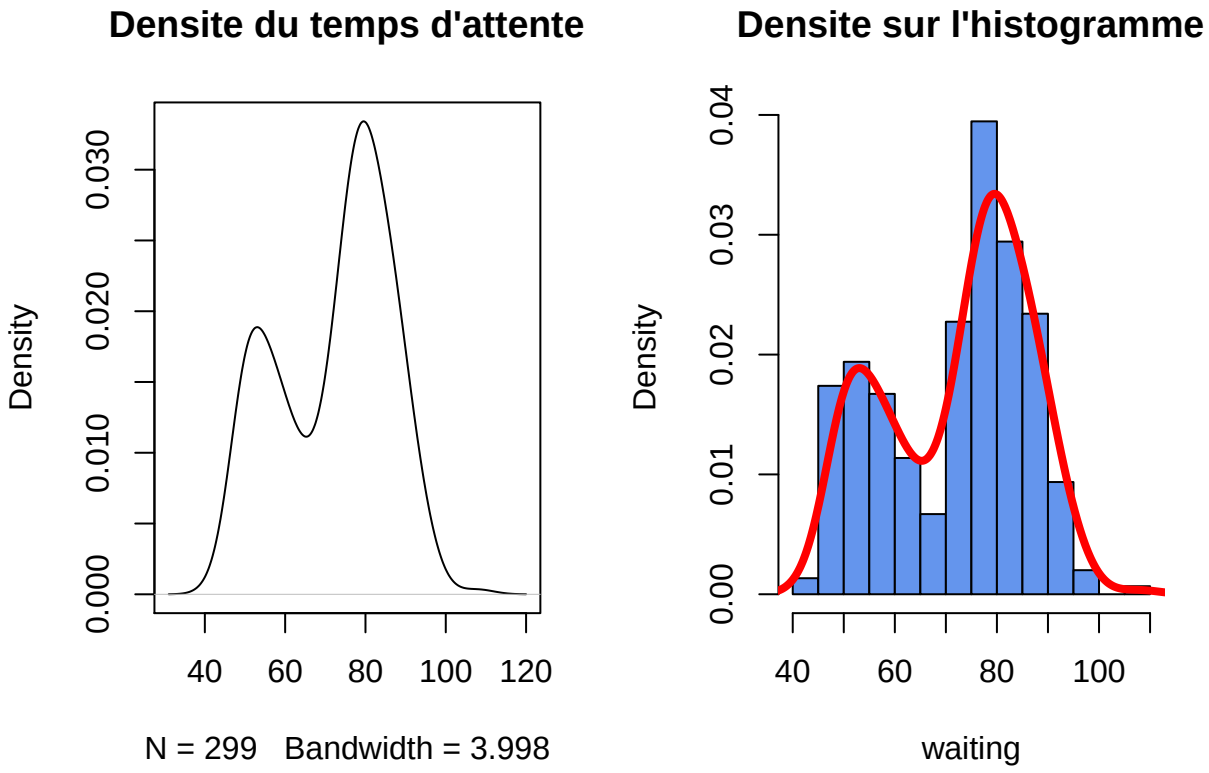
```
length(dens$x)
```

```
## [1] 512
```

Commentaires : A partir de 299 individus, la fonction density donne 512 points qui ont une abscisse et une ordonnée, l'ordonnée étant une densité.

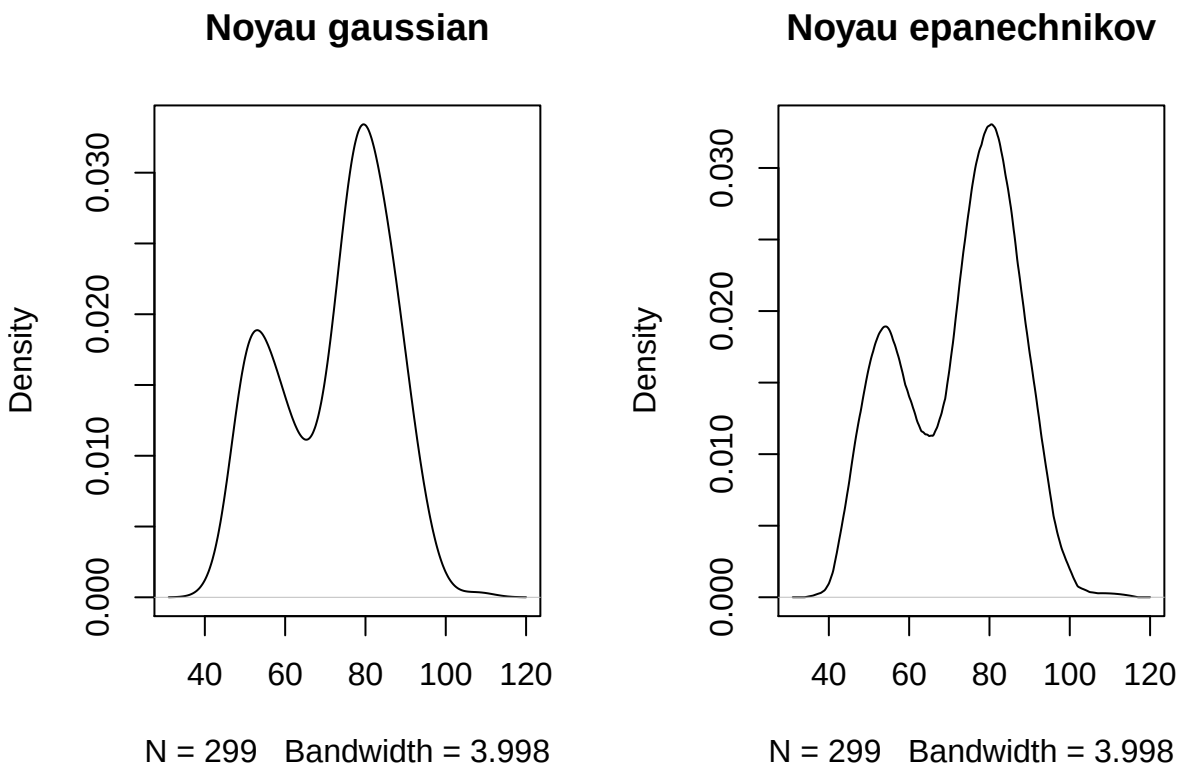
3.2 Courbe de l'estimateur à noyau

```
par(mfrow = c(1,2))
plot(dens, main = "Densite du temps d'attente")
hist(waiting, col = "cornflowerblue", proba = T, breaks = 10, main = "Densite sur l'histogramme")
lines(dens, col = "red", lwd = 4)
```

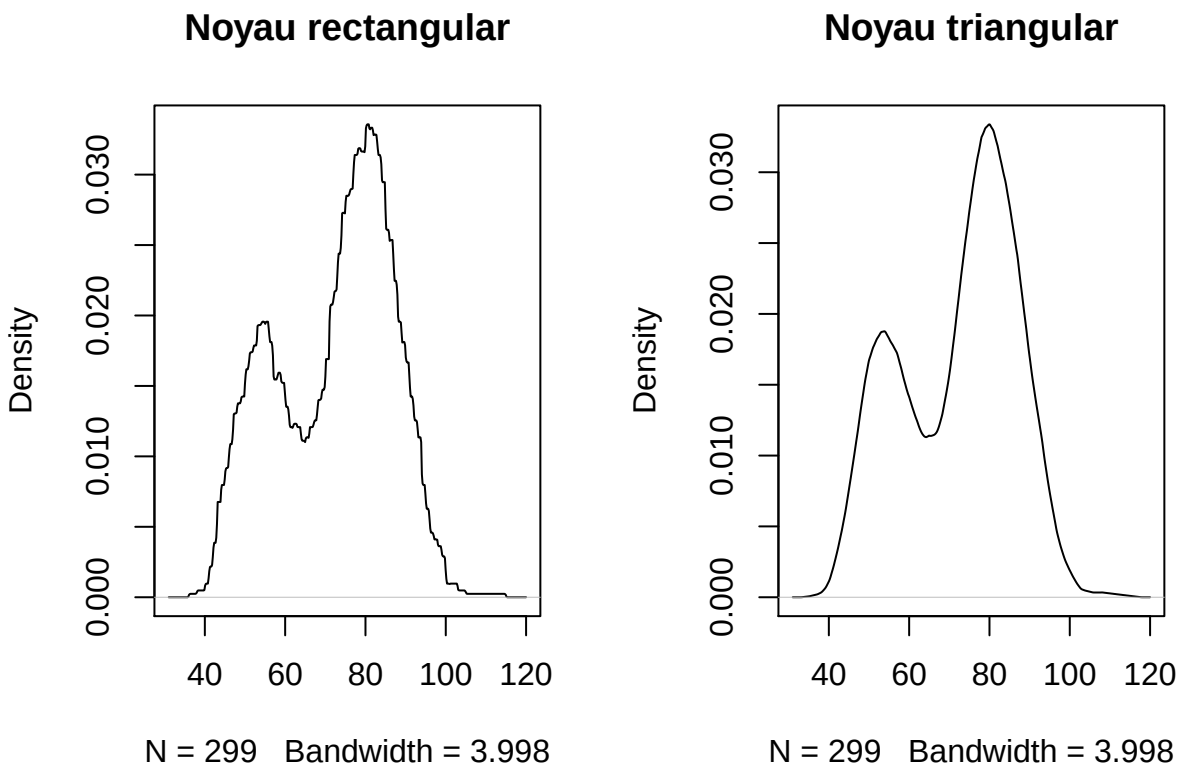


3.3 Variation du noyau avec la fonction kernel

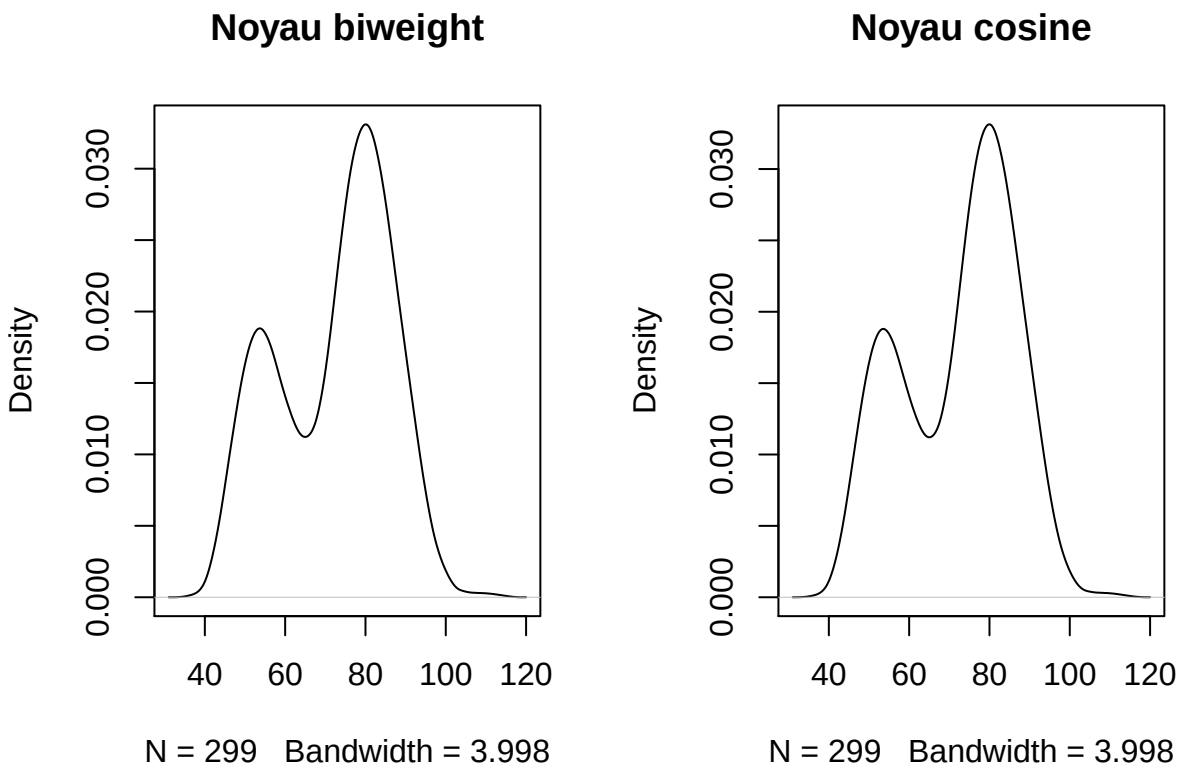
```
par(mfrow = c(1,2))
plot(density(waiting, kernel = "gaussian"), main = "Noyau gaussian")
plot(density(waiting, kernel = "epanechnikov"), main = "Noyau epanechnikov")
```




```
par(mfrow = c(1,2))
plot(density(waiting, kernel = "rectangular"), main = "Noyau rectangular")
plot(density(waiting, kernel = "triangular"), main = "Noyau triangular")
```




```
par(mfrow = c(1,2))
plot(density(waiting, kernel = "biweight"), main = "Noyau biweight")
plot(density(waiting, kernel = "cosine"), main = "Noyau cosine")
```



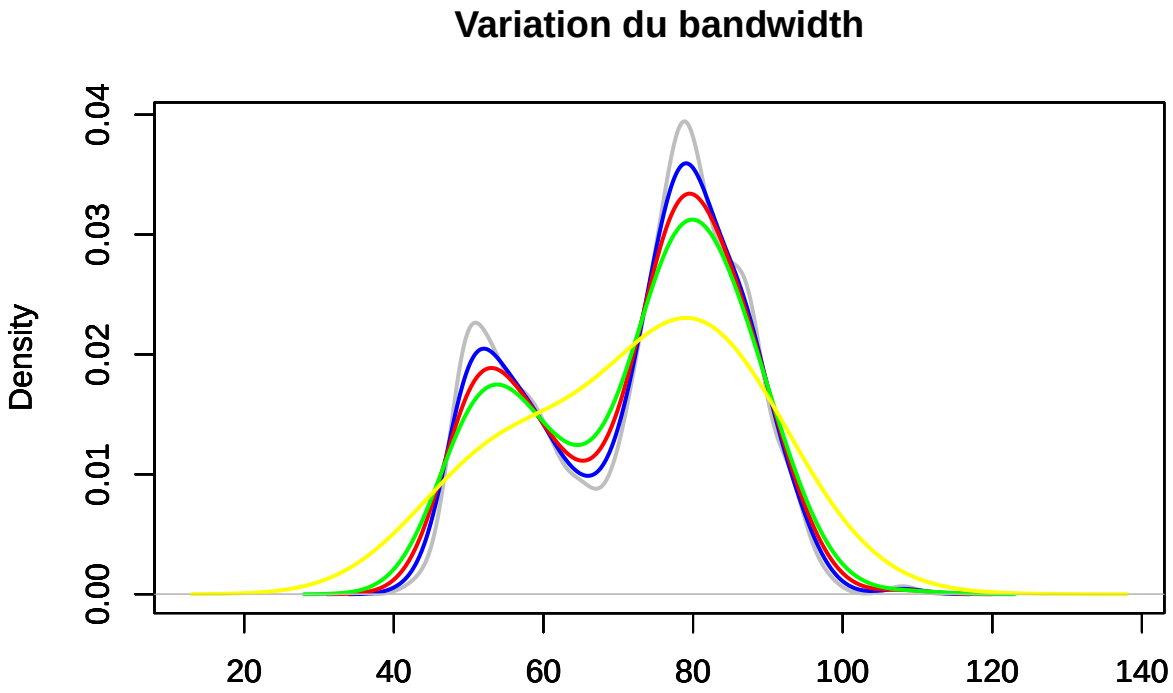
Commentaires : Il n'y a pas de différence significative sur la desnité estimée en remplacant le noyau.

3.4 Variation du paramètre de fenetre

```
dens_2 <- density(waiting, bw = 2)
dens_3 <- density(waiting, bw = 3)
dens_4 <- density(waiting, bw = 4)
dens_5 <- density(waiting, bw = 5)
dens_10 <- density(waiting, bw = 10)
```

```
x_mini <- min(dens_2$x, dens_3$x, dens_4$x, dens_5$x, dens_10$x)
x_maxi <- max(dens_2$x, dens_3$x, dens_4$x, dens_5$x, dens_10$x)
xlimi = c(x_mini, x_maxi)
y_mini <- min(dens_2$y, dens_3$y, dens_4$y, dens_5$y, dens_10$y)
y_maxi <- max(dens_2$y, dens_3$y, dens_4$y, dens_5$y, dens_10$y)
ylimi = c(y_mini, y_maxi)
```

```
plot(dens_2, xlim = xlimi, ylim = ylimi, main = "Variation du bandwidth", xlab = "", col = "gray", lwd =
par(new=T)
plot(dens_3, xlim = xlimi, ylim = ylimi, main = "", xlab = "", col = "blue", lwd = 2)
par(new=T)
plot(dens_4, xlim = xlimi, ylim = ylimi, main = "", xlab = "", col = "red", lwd = 2)
par(new=T)
plot(dens_5, xlim = xlimi, ylim = ylimi, main = "", xlab = "", col = "green", lwd = 2)
par(new=T)
plot(dens_10, xlim = xlimi, ylim = ylimi, main = "", xlab = "", col = "yellow", lwd = 2)
```



Commentaires : Une grande bandwidth entraine de petite variation.

4 Validation croisée

4.1 L'expression du risque MISE d'un estimateur à noyau

Le risque MISE est défini sous la forme suivante :

$$\mathbb{E}_f ||\hat{f}_{n,h} - f||_2^2$$

avec

$$\hat{f}_{n,h}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{X_i - x}{h})$$

Ainsi

$$MISE(h) = J(h) = \mathbb{E}_f(\int \hat{f}_{n,h}^2(x)dx) - 2\mathbb{E}_f(\int f(x)\hat{f}_{n,h}(x)dx) + cste$$

L'expression de l'estimateur obtenu par validation croisée :

$$\mathbb{E}_f(\int \hat{f}_{n,h}^2(x)dx)$$

Et peut etre estimée par :

$$\int \hat{f}^2_{n,h}(x)dx$$

L'espérance étant linéaire, ce qui entraine :

$$\int f(x)\mathbb{E}_f(\hat{f}_{n,h}(x))dx$$

Ainsi, on a :

$$\mathbb{E}_f(\hat{f}_{n,h}(x)) = \frac{1}{h}\int K(\frac{u-x}{h})f(u)du$$

Ce qui donne :

$$\mathbb{E}_f(\hat{f}_{n,h}(x)) = \frac{1}{n(n-1)h}\sum_i\sum_{j\neq i} K(\frac{X_i - X_j}{h})$$

D'où :

$$\hat{J}(h) = \int \hat{f}^2_{n,h}(x)dx - \frac{2}{n(n-1)h}\sum_i\sum_{j\neq i} K(\frac{X_i - X_j}{h})$$

Par validation croisée, on obtient :

$$\hat{J}_{n,h}(h) = \frac{1}{V}\sum_{v=1}^{V}(\int (\hat{f}^v_{n,h})^2(x)dx - \frac{2}{|C_v|(|C_v|-1)h}\sum_{i,j\in C_v, j\neq i} K(\frac{X_i - X_j}{h}))$$

4.2 Démonstration

On a :

$$\hat{f}^{C_v}_{n,h}(X_i) = \frac{1}{|C_v|h}\sum_{j\in C_v} K(\frac{X_j - X_i}{h})$$

En séparant les i égaux à j de ceux qui sont différents, on a :

$$\sum_{i\in C_v}\hat{f}^{C_v}_{n,h}(X_i) = \frac{1}{|C_v|h}\sum_{i,j\in C_v, i\neq j} K(\frac{X_j - X_i}{h}) + \frac{1}{|C_v|h}\sum_{i\in C_v} K(0)$$

En multipliant par :

$$\frac{2}{|C_v|-1}$$

On trouve finalement :

$$\frac{2}{(|C_v|-1)|C_v|h}\sum_{i,j\in C_v, i\neq j} K(\frac{X_j - X_i}{h}) = \frac{2}{|C_v|-1}\sum_{i\in C_v}(\hat{f}^{C_v}_{n,h}(X_i) - \frac{K(0)}{|C_v|h})$$

4.3 Utilisation d'une grille sur les valeurs de h

4.3.1 Structure du programme

```
# valeurs de h
vec_h <- seq(0.5,8,0.25)
# Nombre de paquets d'observations
V <- 5
```

4.3.2 Calcul de l'estimateur du risque MISE

```
# Fonction qui va estimer f
estim_f <- function(X,h){
  M <- length(X)
  f_x <- c(1:M)
  for(i in 1:M){
    som <- 0
    for(j in 1:M){
      som <- som + exp(-((X[j]-X[i])/h)**2/2)/sqrt(2*pi)
    }
    f_x[i] <- som/M/h
  }
  return(f_x)
}
```

```
# Fonction qui va estimer J
estim_J <- function(I_estim,I_eval,X,h){
  X_estim <- X[I_estim]
  X_eval <- X[I_eval]
  M_estim <- length(I_estim)
  M_eval <- length(I_eval)
  J_hat <- 0

  # Le vecteur des f(X) pour X dans X_estim
  f_X_estim <- estim_f(X_estim,h)

  # Le vecteur des f(X) pour X dans X_eval
  f_X_eval <- estim_f(X_eval,h)

  # Premier terme de l'estimateur de J
  som <- 0
  for(i in 1:(M_estim-1)){
    som <- som + (X_estim[i+1]-X_estim[i])*f_X_estim[i+1]**2
  }
  J_hat <- som/M_estim

  # Deuxieme terme de l'estimateur de J
  som <- sum(f_X_eval) - 1/(sqrt(2*pi)*h)
  som <- som * 2 / (M_eval-1)
  J_hat <- J_hat - som

  return(J_hat)
}
```

```
# Fonction qui va renvoyer la moyenne par cross-validation de l'estimateur de J pour differentes valeurs
cross_val <- function(vec_h,V,X){
```

```
  H <- length(vec_h)
  n <- length(X)
  size <- floor(n/V)

  J_hat <- matrix(nrow = V,ncol = H)

  # Calcule l'estimateur de J pour chaque h pour chaque iteration de cross-validation
  for(ind_h in 1:H){
    h <- vec_h[ind_h]
    I_estim <- seq(size+1,n,1)
    I_eval <- seq(1,size,1)
    J_hat[1,ind_h] <- estim_J(I_estim,I_eval,X,h)
    for(iter in 2:(V-1)){
      I_estim <- c(seq(1,size*(iter-1),1),seq(1+size*iter,n,1))
      I_eval <- seq(1+size*(iter-1),size*iter,1)
      J_hat[iter,ind_h] <- estim_J(I_estim,I_eval,X,h)
    }
    I_estim <- seq(1,size*(V-1),1)
    I_eval <- seq(1+size*(V-1),n,1)
    J_hat[V,ind_h] <- estim_J(I_estim,I_eval,X,h)
  }

  mean_J_hat <- apply(J_hat,2,mean)
  return(list("ind_min"= which.min(mean_J_hat), "h_min" =vec_h[which.min(mean_J_hat)],
              "J_per_h" = mean_J_hat))
}

# Resultat de la fonction de cross validation
result <- cross_val(vec_h,V,waiting)
```

4.3.3 Selection du meilleur estimateur