République du Sénégal

*****

Un peuple - Un but - Une foi

ANSD

Agence Nationale de la Statistique et de la Démographie

ensae

Ecole Nationale de la Statistique et de l'Analyse Economique

STATISTIQUE NON PARAMETRIQUE

Tests de statistique non paramétrique sous R

Par :

Les élèves de la classe ISE 2

Elèves Ingénieurs Statisticiens Economistes, 2e année

Sous la supervision de :

Dr. Mamadou BALDE

Ingénieur Statisticien Economiste, Chercheur à l'ENSAE

Année académique : 2024-2025

CONTENTS

1 CHAPITRE 1

1.1 Exercice 1 :

Reformulation du problème : générer un échantillon de taille supérieure à 30 suivant différentes lois puis observer la distribution de chacune d'elles.

Conclure.

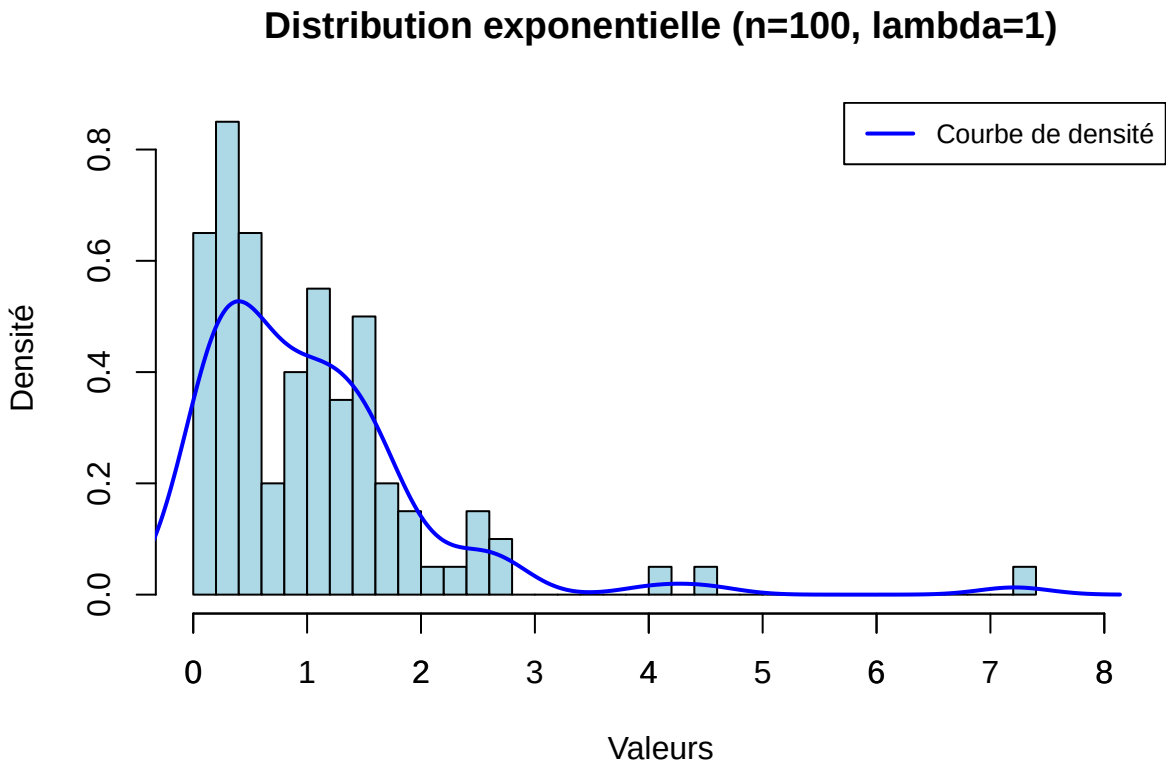
1.1.1 Génération des échantillons

```r
set.seed(123) # Fixer la graine pour assurer la reproductibilité
n <- 100 # Taille des échantillons

#------------- Génération des échantillons ----------------------------

Echant_expo <- rexp(n, rate = 1)# Loi exponentielle (lambda = 1)
Echant_pois <- rpois(n, lambda = 5) # Loi de Poisson (lambda = 5)
Echant_stud <- rt(n, df = 5) # Loi de Student (ddl = 5)
Echant_geo <- rgeom(n, prob = 0.3) # Loi Géométrique (p = 0.3)

# Loi normale
mu <- 0  #moyenne
sigma <- 1  #écart-type
Echant_normal <- rnorm(n, mean = mu, sd = sigma)
```

1.1.2 Distribution d'échantillonnage de chaque loi

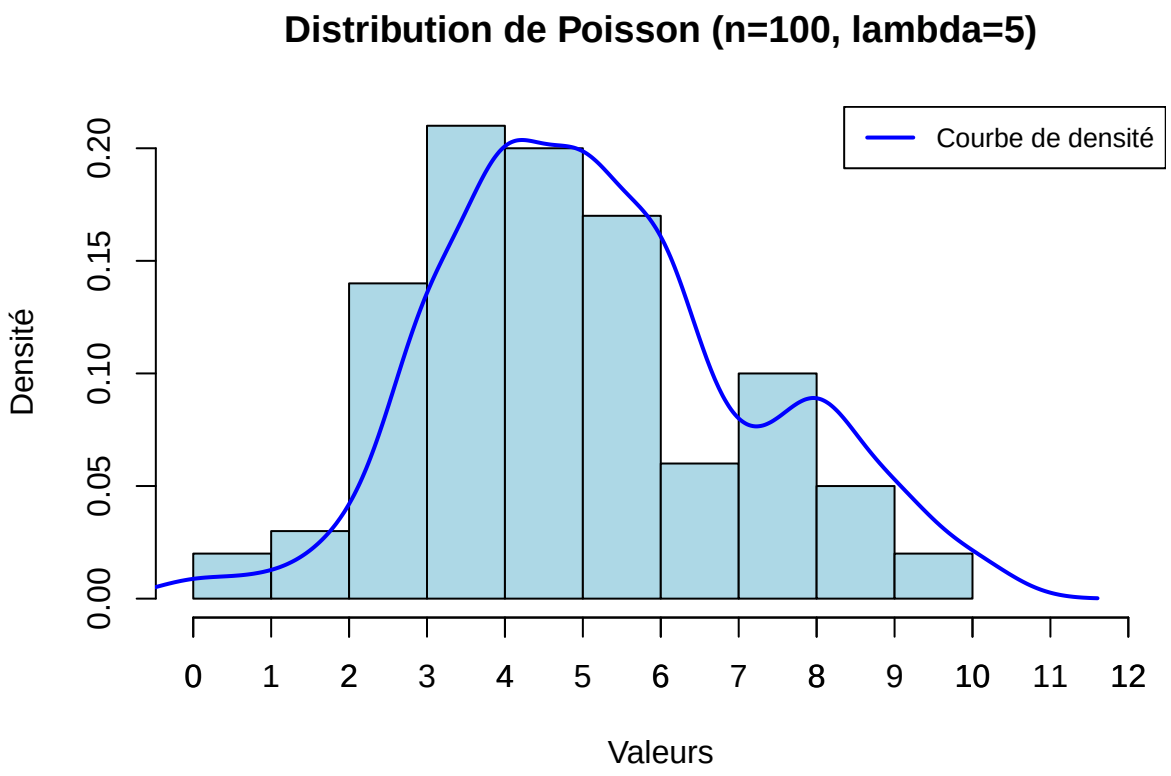
- **Loi exponentielle**

```r
# Histogramme et courbe de densité

hist(Echant_expo, probability = TRUE, col = "lightblue",
     main = "Distribution exponentielle (n=100, lambda=1)",
     xlab = "Valeurs", ylab = "Densité", border = "black",
     xlim = c(0, max(Echant_expo) + 1), breaks = 30)

lines(density(Echant_expo), col = "blue", lwd = 2) # Courbe de densité

# Personnalisation de l'axe des x
axis(1, at = seq(0, max(Echant_expo) + 1, by = 1))  # Graduation tous les 1

# Légende
legend("topright", legend = c("Courbe de densité"),
       col = c("blue"), lwd = 2, lty = c(1), cex = 0.8)
```



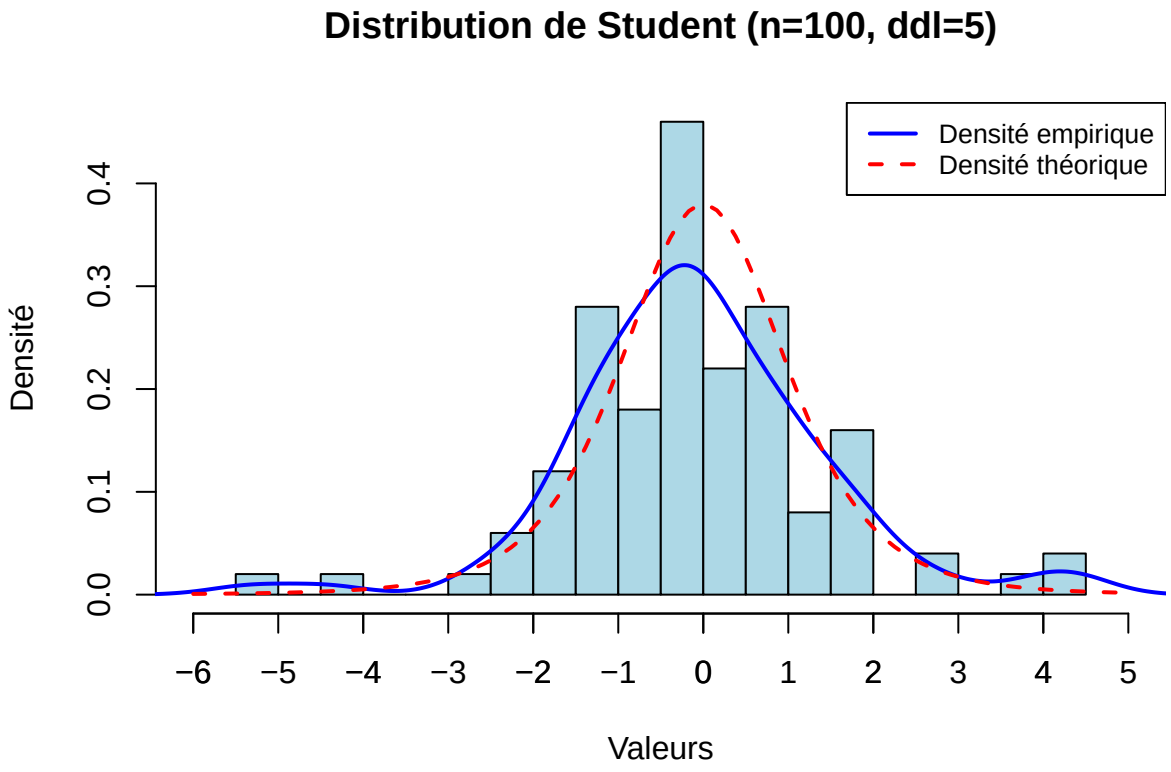
- **Loi de Poisson**

```r
# Histogramme et courbe de densité

hist(Echant_pois, probability = TRUE, col = "lightblue",
     main = "Distribution de Poisson (n=100, lambda=5)",
     xlab = "Valeurs", ylab = "Densité", border = "black",
     xlim = c(0, max(Echant_pois) + 2), breaks = max(Echant_pois) + 1)

lines(density(Echant_pois), col = "blue", lwd = 2) # Courbe de densité

# Personnalisation de l'axe des x
axis(1, at = seq(0, max(Echant_pois) + 2, by = 1))  # Graduation tous les 1

# Légende
legend("topright", legend = c("Courbe de densité"),
       col = c("blue"), lwd = 2, lty = c(1), cex = 0.8)
```



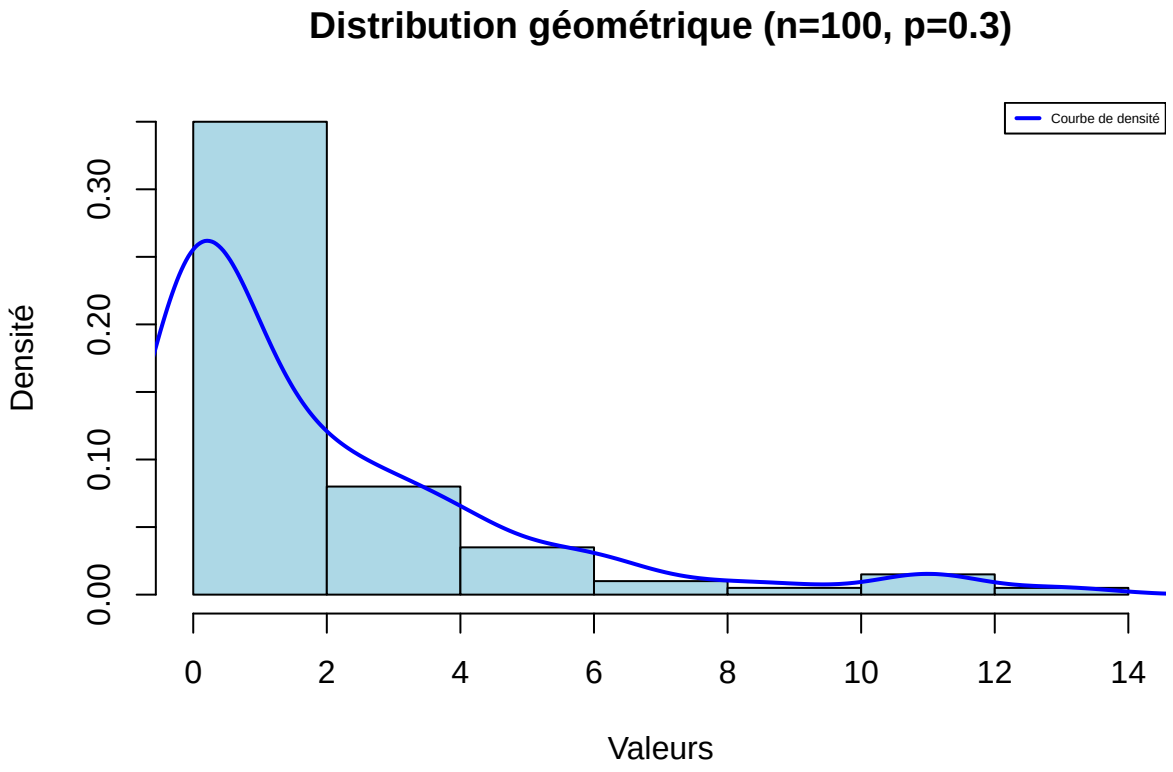
- **Loi de Student**

```r
# Histogramme et courbe de densité

hist(Echant_stud, probability = TRUE, col = "lightblue",
     main = "Distribution de Student (n=100, ddl=5)",
     xlab = "Valeurs", ylab = "Densité", border = "black",
     xlim = c(-6, 5), breaks = 20)

lines(density(Echant_stud), col = "blue", lwd = 2) # Courbe de densité empirique
curve(dt(x, df = 5), col = "red", lwd = 2, add = TRUE, lty = 2) # Densité théorique

# Personnalisation de l'axe des x
axis(1, at = seq(-6, 5, by = 1))  # Ajoute des graduations tous les 1

# Légende
legend("topright", legend = c("Densité empirique", "Densité théorique"),
       col = c("blue", "red"), lwd = 2, lty = c(1,2), cex = 0.8)
```



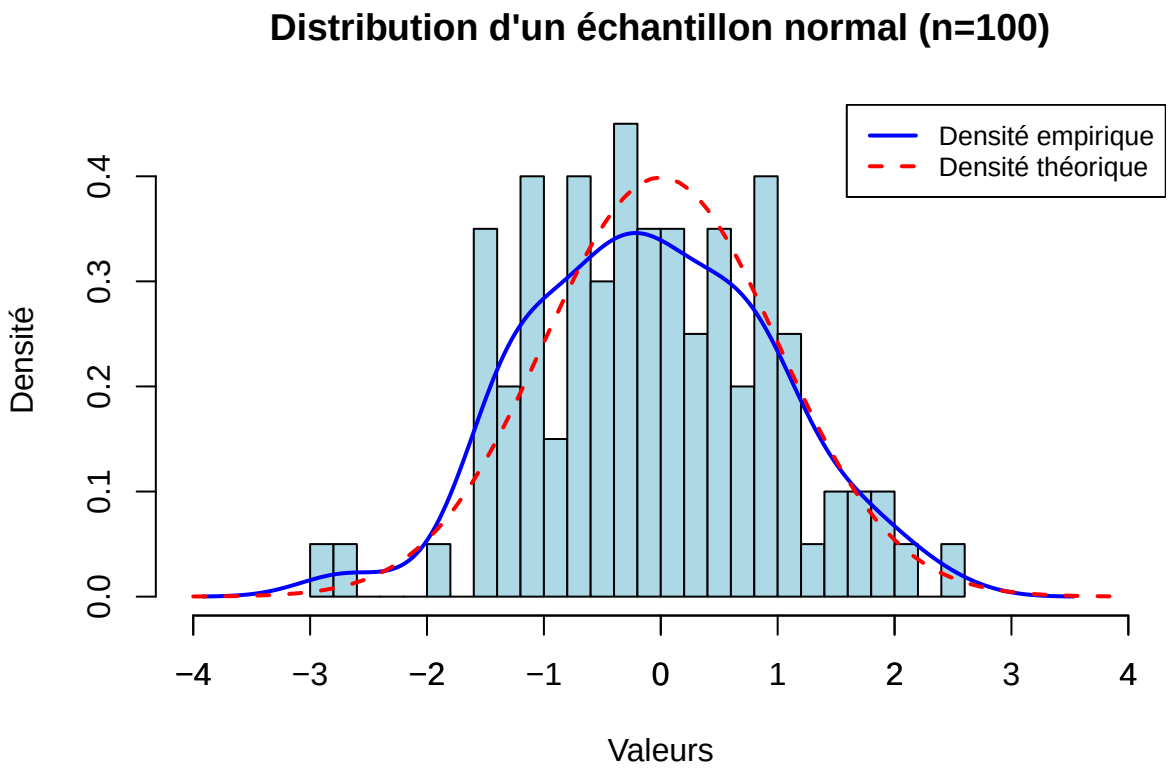
- **Loi géométrique**

```r
#histogramme et courbe de densité

hist(Echant_geo, probability = TRUE, col = "lightblue",
     main = "Distribution géométrique (n=100, p=0.3)",
     xlab = "Valeurs", ylab = "Densité", border = "black")

lines(density(Echant_geo), col = "blue", lwd = 2) # Courbe de densité


legend("topright", legend = c("Courbe de densité"),
       col = c("blue", "red"), lwd = 2, lty = c(1,2), cex = 0.4)
```



- **Loi normale**

```r
# Histogramme et courbe de densité

hist(Echant_normal, probability = TRUE, col = "lightblue",
     main = "Distribution d'un échantillon normal (n=100)",
     xlab = "Valeurs", ylab = "Densité", border = "black",
     xlim = c(-4, 4), breaks = 20)
lines(density(Echant_normal), col = "blue", lwd = 2) # Courbe de densité empirique

# Courbe théorique de la loi normale
curve(dnorm(x, mean = mu, sd = sigma), col = "red", lwd = 2, add = TRUE, lty = 2)

# Personnalisation de l'axe des x
axis(1, at = seq(-4, 4, by = 1))  # Ajoute des graduations tous les 1

# Légende
legend("topright", legend = c("Densité empirique", "Densité théorique"),
       col = c("blue", "red"), lwd = 2, lty = c(1,2), cex = 0.8)
```



Conclusion : La loi de Student a une distribution proche de celle de loi normale, cela est d'autant plus une realité que les degrés de liberté sont élevés. La loi de Poisson tend egalement vers la normalité surtout pour des valeurs de lambda elevés(lambda > 30). Cependant, les lois exponentielle et géométrique sont très éloignées d'une distribution normale, elles nécessitent des transformations pour s'y conformer.

1.2 Exercice 2 : Statistique d'ordre

```
stat_ordre = function(vector){
               if(is.vector(vector)==TRUE) {
                 vect = sort(vector, decreasing = FALSE, na.last = TRUE)
                 return(vect)
               } else {
                 print("Vous devez donner un vecteur")
               }
  }

#Exemple:
vec1 = c(5,2,-1,3)
a =stat_ordre(vec1)
cat("Vecteur (", vec1, ") ", "ordonné :", a , "\n")
```

```
## Vecteur ( 5 2 -1 3 )  ordonné : -1 2 3 5
```

```r
#Exemple sur un échantillon
set.seed(123)
# échantillon de 50 nombres uniques entre 1 et 100
vec2 <- sample(1:100, 50, replace = FALSE)
cat("Echantillon : ", vec2, "\n", "\n")
```

```
## Echantillon :  31 79 51 14 67 42 50 43 97 25 90 69 57 9 72 26 7 95 87 36 78 93 76 15 32 84
##
```

```r
b =stat_ordre(vec2)
cat( "Echantillon ordonné : ", b )
```

```
## Echantillon ordonné :  5 7 8 9 12 13 14 15 18 21 23 25 26 27 29 31 32 33 34 36 38 41 42 43
```

1.3 Exercice 3 : Statistique de rang

```r
#------------------------------Fonction-Rang-------------------------------#

stat_rang <- function(vector) {
  if (!is.vector(vector)) {
    print("Vous devez donner un vecteur")
    return(NULL)
  }

  vecteur <- numeric(length(vector))

  for (i in 1:length(vector)) {
    compteur <- 0
    for (j in 1:length(vector)) {
      if (vector[i] >= vector[j]) {
        compteur <- compteur + 1
      }
    }
    vecteur[i] <- compteur
  }
  return(vecteur)
}
#--------------------------Exemple -----------------------------------#
vecteur = c(5,2,-1,3)
rang=stat_rang(vecteur)
rang
```

```
## [1] 4 2 1 3
```

```r
# Pour un echantillon :

set.seed(123)
 # echantillon de 50 nombres uniques entre 1 et 100
echantillon <- sample(1:100, 50, replace = FALSE)
print(echantillon)
```

```
##  [1] 31 79 51 14 67 42 50 43 97 25 90 69 57  9 72 26  7 95 87 36 78 93 76 15 32
## [26] 84 82 41 23 27 60 53 75 89 71 38 91 34 29  5  8 12 13 18 33 66 64 65 21 77
```

```r
rang_echantillon =stat_ordre(echantillon)
rang_echantillon
```

```
##  [1]  5  7  8  9 12 13 14 15 18 21 23 25 26 27 29 31 32 33 34 36 38 41 42 43 50
## [26] 51 53 57 60 64 65 66 67 69 71 72 75 76 77 78 79 82 84 87 89 90 91 93 95 97
```

```r
#-------------------------------------------------------------------------------#

# Avec la fonction rank  de r

vecteur = c(5,2,-1,3)
rank(vecteur)
```

```
## [1] 4 2 1 3
```

```r
# En cas d'exoequo


vecteur_2 = c(5,5,-1,3)
rank(vecteur_2,)
```

```
## [1] 3.5 3.5 1.0 2.0
```

```r
## Les options : average, first, last, random, max, min

x2 = c(5,5,-1,3,3,2,4,3)

## ranks without averaging
rank(x2, ties.method= "average")
```

```
## [1] 7.5 7.5 1.0 4.0 4.0 2.0 6.0 4.0
```

```r
rank(x2, ties.method= "first")  # first occurrence wins
```

```
## [1] 7 8 1 3 4 2 6 5
```

```r
rank(x2, ties.method= "last")   #  last occurrence wins
```

```
## [1] 8 7 1 5 4 2 6 3
```

```r
rank(x2, ties.method= "random") # ties broken at random
```

```
## [1] 7 8 1 4 5 2 6 3
```

```r
rank(x2, ties.method= "random") # and again
```

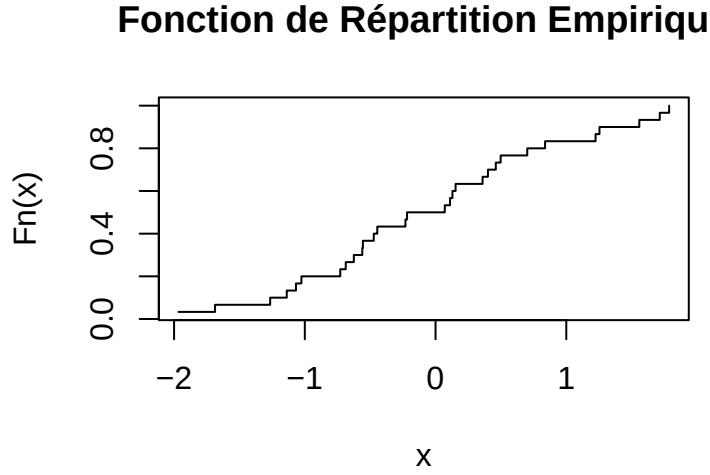
```
## [1] 8 7 1 3 4 2 6 5
```

NB : Pour avoir le code de la fonction rank par exemple, il faut saisir la commande *print(rank)*.

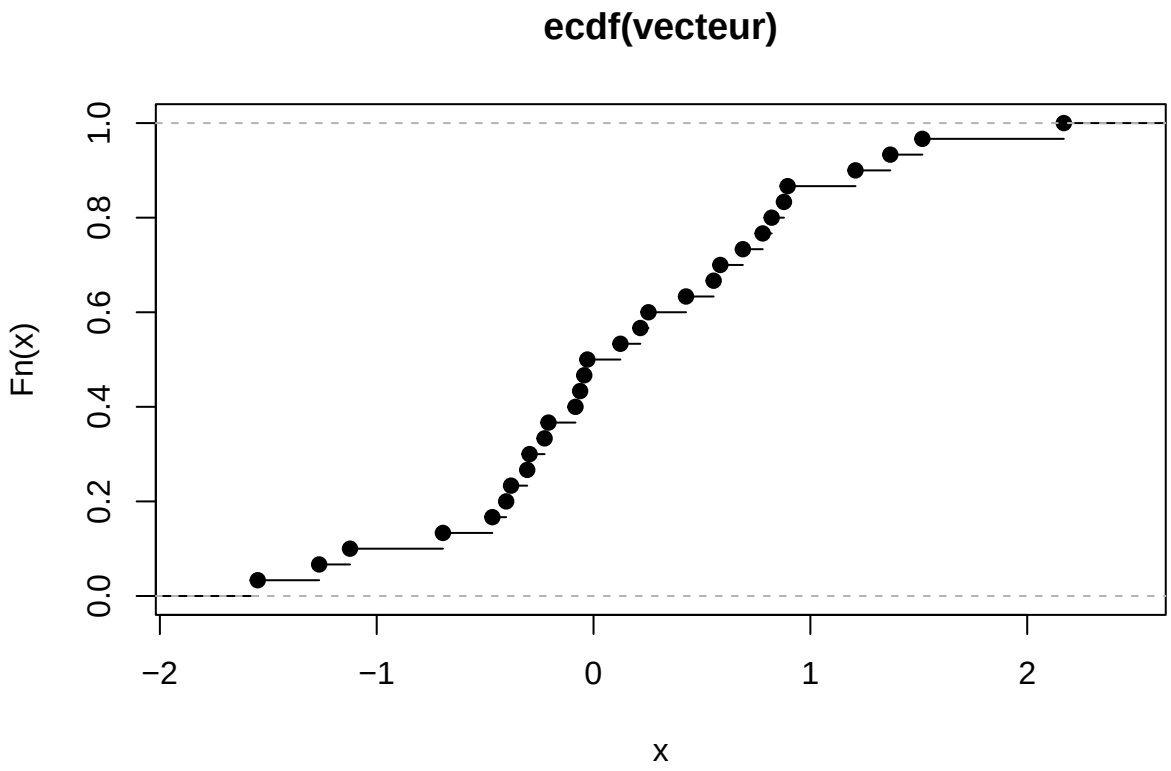
1.4 Exercice 4 : Fonction de répartition empirique

```r
# Fonction pour calculer, retourner les valeurs et tracer la fonction de répartition

calculer_et_tracer_Fn <- function(vecteur) {

  vecteur_trie <- sort(vecteur)

  n <- length(vecteur)

  Fn_values <- sapply(vecteur_trie, function(x) sum(vecteur <= x) / n)

  Fn_data <- data.frame(x = vecteur_trie, Fn = Fn_values)

  print("Valeurs de la fonction de répartition empirique :")
  print(Fn_data)

  plot(Fn_data$x, Fn_data$Fn, type = "s", xlab = "x", ylab = "Fn(x)",
       main = "Fonction de Répartition Empirique")

  return(Fn_data)
}

# Exemple d'utilisation

set.seed(123)
vecteur <- rnorm(30)
resultat <- calculer_et_tracer_Fn(vecteur)
```

```
## [1] "Valeurs de la fonction de répartition empirique :"
##              x         Fn
## 1  -1.96661716 0.03333333
## 2  -1.68669331 0.06666667
## 3  -1.26506123 0.10000000
## 4  -1.13813694 0.13333333
## 5  -1.06782371 0.16666667
## 6  -1.02600445 0.20000000
## 7  -0.72889123 0.23333333
## 8  -0.68685285 0.26666667
## 9  -0.62503927 0.30000000
## 10 -0.56047565 0.33333333
## 11 -0.55584113 0.36666667
## 12 -0.47279141 0.40000000
## 13 -0.44566197 0.43333333
## 14 -0.23017749 0.46666667
## 15 -0.21797491 0.50000000
## 16  0.07050839 0.53333333
## 17  0.11068272 0.56666667
## 18  0.12928774 0.60000000
## 19  0.15337312 0.63333333
## 20  0.35981383 0.66666667
## 21  0.40077145 0.70000000
## 22  0.46091621 0.73333333
## 23  0.49785048 0.76666667
## 24  0.70135590 0.80000000
## 25  0.83778704 0.83333333
## 26  1.22408180 0.86666667
## 27  1.25381492 0.90000000
## 28  1.55870831 0.93333333
## 29  1.71506499 0.96666667
## 30  1.78691314 1.00000000
```



```r
# Autre méthode pour tracer la fonction de répartition  et retourner les Fn

vecteur <- rnorm(30)
plot(ecdf(vecteur))
```




```r
ecdf(vecteur)
```

```
## Empirical CDF
## Call: ecdf(vecteur)
##  x[1:30] = -1.5488, -1.2654, -1.1231,  ..., 1.5165,  2.169
```

```r
fn=ecdf(vecteur)
fn(vecteur)
```

```
##  [1] 0.63333333 0.30000000 0.86666667 0.83333333 0.80000000 0.73333333
##  [7] 0.66666667 0.43333333 0.26666667 0.23333333 0.13333333 0.36666667
## [13] 0.06666667 1.00000000 0.90000000 0.10000000 0.20000000 0.16666667
## [19] 0.76666667 0.40000000 0.60000000 0.50000000 0.46666667 0.93333333
## [25] 0.33333333 0.96666667 0.03333333 0.70000000 0.53333333 0.56666667
```

```r
fn(sort(vecteur)) # Pour ordonner les Fn
```

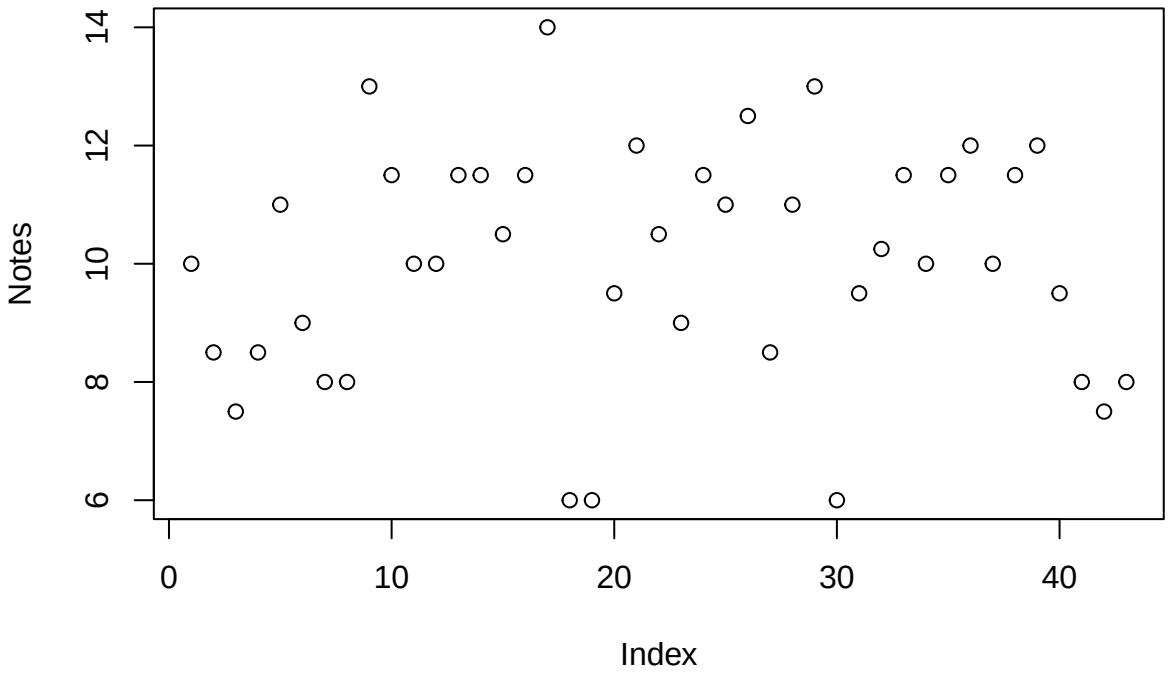
```
##  [1] 0.03333333 0.06666667 0.10000000 0.13333333 0.16666667 0.20000000
##  [7] 0.23333333 0.26666667 0.30000000 0.33333333 0.36666667 0.40000000
## [13] 0.43333333 0.46666667 0.50000000 0.53333333 0.56666667 0.60000000
## [19] 0.63333333 0.66666667 0.70000000 0.73333333 0.76666667 0.80000000
## [25] 0.83333333 0.86666667 0.90000000 0.93333333 0.96666667 1.00000000
```

2 CHAPITRE 2 : Tests non paramétriques pour 1 échantillon

2.1 Tests de corrélation de rang de Spearman

2.1.1 On considère les notes de base de données 2 ci-après :

```
Notes <- c(10,8.5,7.5,8.5,11,9,8,8,13,11.5,10,10,11.5,11.5,10.5,11.5,14,6,6,9.5,
           12,10.5,9,11.5,11,12.5,8.5,11,13,6,9.5,10.25,11.5,10,11.5,12,10,11.5,
           12,9.5,8,7.5,8)
plot(Notes)
```



Commentaires du plot : Les données semblent homogènes (faible écart-type et coefficient de variation) et on n'en dégage pas de tendance a priori. Vérifions cela à l'aide de tests de corrélation de rangs de Spearman.

- **Tendance croissante :**

On veut tester H_0 : les données sont aléatoires et i.i.d contre H_1: on a une tendance croissante dans les données.

```r
# Vecteur rang
rang <- rank(Notes)
# Vecteur d'indices
i= 1:length(Notes)
# Dataframe des indices, notes et rangs
data <- data.frame(i, Notes, rang)
#Test de corrélation des rangs de Spearmann
cor.test(data$i, data$rang,
         alternative="g", #tendance croissante
         method="spearman" )
```

```
##
##  Spearman's rank correlation rho
##
## data:  data$i and data$rang
## S = 12179, p-value = 0.3042
## alternative hypothesis: true rho is greater than 0
## sample estimates:
##        rho
## 0.08038061
```

r_s vaut **0.08** et la p-value est de **0.3**. On ne peut rejeter H_0 au seuil de 5%.

NB : $S = \sum_{i=1}^{n} (R_i - i)^2 = 12179$

- **Tendance décroissante :**

On veut tester H_0 : les données sont aléatoires et i.i.d contre H_1: on a une tendance décroissante dans les données.

```r
cor.test (data$i, data$Notes, alternative = "l", method= "s")
```

```
##
##  Spearman's rank correlation rho
##
## data:  data$i and data$Notes
## S = 12179, p-value = 0.6958
## alternative hypothesis: true rho is less than 0
## sample estimates:
##        rho
## 0.08038061
```

- **Tendance quelconque :**

On veut tester H_0 : les données sont aléatoires et i.i.d contre H_1: on a une tendance quelconque dans les données.

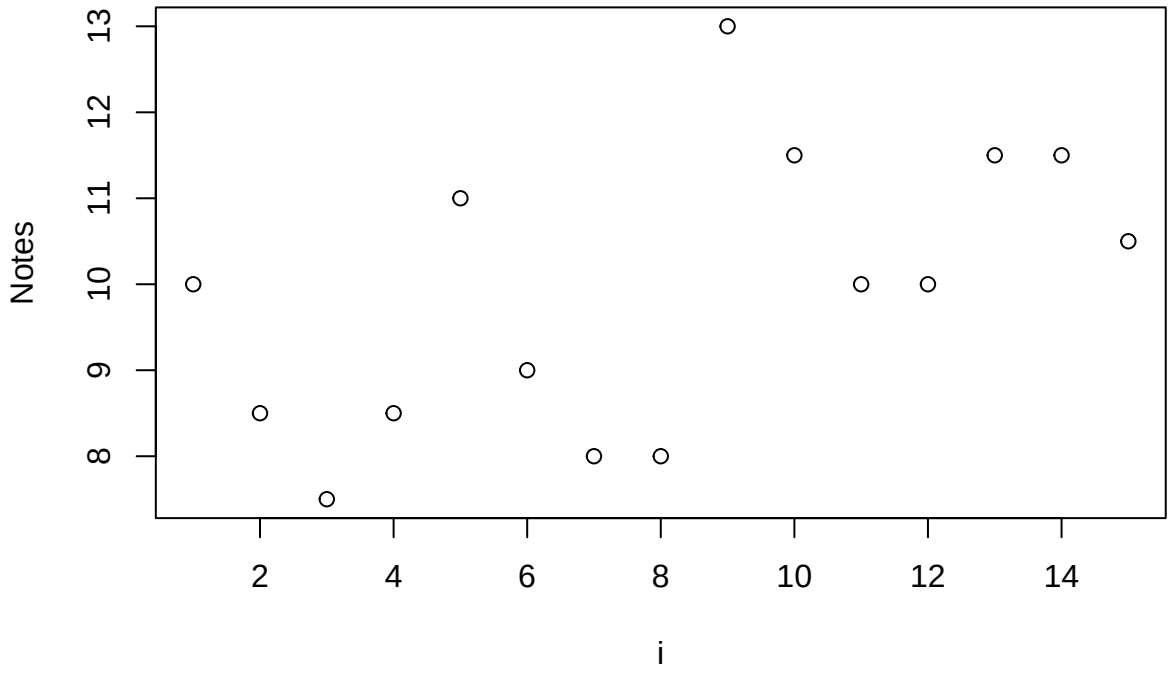
```r
cor.test (data$i, data$Notes, alternative = "two.sided", method= "s")
```

```
## Warning in cor.test.default(data$i, data$Notes, alternative = "two.sided", :
## Impossible de calculer la p-value exacte avec des ex-aequos

##
##  Spearman's rank correlation rho
##
## data:  data$i and data$Notes
## S = 12179, p-value = 0.6084
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.08038061
```

On a les mêmes conclusions dans les 2 derniers cas. On ne peut rejeter H_o au seuil de 5%.

2.1.2 On considère les 15 premières observations.

```r
data1<- data[1:15,1:2]
plot(data1)
```



Il semblerait y avoir une tendance croissante. Par ailleurs, le nombre d'observations est inférieur à 30. On ne peut utiliser le cor.test ici.

```r
data1$rang <- rank(data1$Notes)
n<- length(data1$i)
# calcul de rs
somme <-sum((data1$rang-data1$i)^2)
rs <- 1- 6*somme/(n*(-1 + n^2 ))
# calcul de rs'(que nous notons rs1)
rs1 <- rs*sqrt((n-2)/(1-rs^2))
#t de Student pour rs1
t<- qt(0.95,13)
#Affichage des résultats
cat("rs = ", rs, "\n")
```

```
## rs =  0.5678571
```

```r
cat("rs' = ", rs1, "\n")
```

```
## rs' =  2.48739
```

```r
cat("t (13, 0.95) = ", t, "\n")
```

```
## t (13, 0.95) =  1.770933
```

On obtient que $rs' > t$. Au vu de l'évidence des données, on peut rejeter H_0 au seuil de 5%.

2.2 Test de Kendall

2.2.1 Données

Le test de Kendall permet de repérer une tendance dans un ensemble de données, sans supposer que celles-ci suivent une loi normale. Il se base sur les rangs pour mesurer s'il y a une progression ou une diminution régulière.

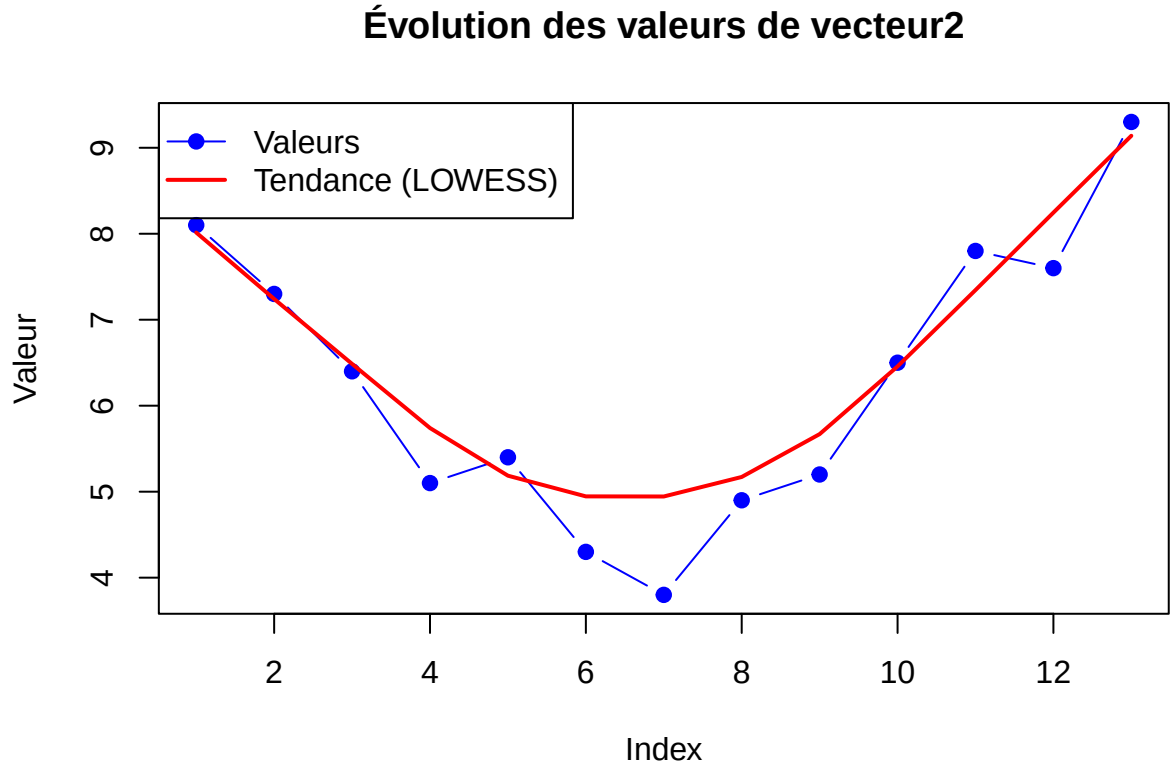
À partir des données fournies dans l'exercice du cours, nous allons appliquer ce test pour voir s'il existe une tendance significative

```r
vecteur2=c(8.1,7.3,6.4,5.1,5.4,4.3,3.8,4.9,5.2,6.5,7.8,7.6,9.3)

inversion <- function(vecteur){
  n=length(vecteur)
  total <-0
  total2 <-0
  for (i in 1:(n-1)) {
    som1 <- 0
    som2 <- 0
    for (j in (i+1):n) {
      if (vecteur[i] > vecteur[j]) {
```

```
        som1 <- som1 + 1
      }
      if (vecteur[i] < vecteur[j]){
        som2 <- som2 + 1
      }
    }
    total <- total + som1
    total2 <- total2 + som2
  }
  S= total2 - total
  return(c(total, S, n))
}

resultat = inversion(vecteur2)
Q <- resultat[1]
S <- resultat[2]
n <- resultat[3]
```

- Visualisation des données

```
plot (1:n, vecteur2, type = "b", col = "blue", pch = 19,
     xlab = "Index", ylab = "Valeur", main = "Évolution des valeurs de vecteur2")
lines(lowess(1:n, vecteur2), col = "red", lwd = 2)  # Tendance lissée
legend("topleft", legend = c("Valeurs", "Tendance (LOWESS)"),
       col = c("blue", "red"), lwd = c(1, 2), pch = c(19, NA), lty = c(1, 1))
```



2.2.2 Utilisation de la fonction cor.test avec Kendall

```r
Ri = rank(vecteur2)
i = 1:n
cor.test(Ri, i, alternative = "greater", method = "kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  Ri and i
## T = 44, p-value = 0.295
## alternative hypothesis: true tau is greater than 0
## sample estimates:
##       tau
## 0.1282051
```

2.2.3 Espérance et variance de Q

- **Formule de calcul de l'espérance :**

$$\mathbb{E}[Q] = \frac{n(n-1)}{4}$$

- **Formule de calcul de la variance :**

$$Var(Q) = \frac{n(n-1)(2n+5)}{72}$$

```r
E_Q <- n*(n-1)/4
V_Q <- n*(n-1)*(2*n+5)/72
E_Q; V_Q
```

```
## [1] 39
```

```
## [1] 67.16667
```

2.2.4 Tau de Kendall

- **Formule de Calcul du tau de Kendall :**

$$\tau = \frac{1-4Q}{n(n-1)}$$

- **Espérance du tau de kendall :**

$$\mathbb{E}[\tau] = 0$$

- **Variance du taux :**

$$Var(\tau) = \frac{2(2n+5)}{9n(n-1)}$$

```
tau = 1 - 4*Q / (n*(n-1))
E_tau = 0
V_tau = 2*(2*n+5) / (9*n*(n-1))
tau; V_tau
```

```
## [1] 0.1282051
```

```
## [1] 0.04415954
```

2.2.5 Test statistique

- **Méthode 1 : test direct manuel**

Statistique de test normalisée :

$$Z = \frac{\tau - E[\tau]}{\sqrt{Var(\tau)}}$$

Cette statistique sera comparée par la loi normale $N(0,1)$ pour déterminer si l'on rejette l'hypothèse nulle.

```
tau_centre = (tau - E_tau) / sqrt(V_tau)
tau_centre
```

```
## [1] 0.6100889
```

```
if (tau_centre > qnorm(0.975) | tau_centre < -qnorm(0.975)) {
  "On rejette H0 (bilatéral) : tendance significative."
} else {
  "On ne rejette pas H0 (bilatéral)."
}
```

```
## [1] "On ne rejette pas H0 (bilatéral)."
```

```
if (tau_centre > qnorm(0.95)) {
  "On rejette H0 (unilatéral à droite) : ordre positif significatif."
} else {
  "On ne rejette pas H0 (unilatéral à droite)."
}
```

```
## [1] "On ne rejette pas H0 (unilatéral à droite)."
```

- **Méthode 2 : test Kendall intégré**

La fonction cor.test() a été utilisée pour estimer la corrélation non paramétrique de Kendall entre les indices (1 à n) et les valeurs de vecteur2.

Si la statistique tau est positive, cela suggère une tendance croissante ; si elle est négative, cela indiquerait une tendance décroissante.

Cependant, si la p-value est supérieure au seuil de 5 %, cette tendance n'est pas statistiquement significative.

Ainsi, si la p-value dépasse ce seuil, on ne rejette pas l'hypothèse nulle d'absence de tendance monotone entre les valeurs et leur ordre.

```
cor.test(1:n, vecteur2, method = "kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  1:n and vecteur2
## T = 44, p-value = 0.59
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## 0.1282051
```

- Calcul de la p-value :

$$p - value = 2 \times P(Z > |z_{obs}|) = 2 \times (1 - \Phi(\mid Z \mid)$$

Si p_value < 0.05 : on rejette l'hypothèse nulle au seuil de 5 % . Il y a une tendance significative dans les données.

Si p_value > 0.05 : on ne rejette pas H_0. Il n'y a pas de preuve suffisante d'une tendance monotone dans les observations.

```
p_value = 2 * pnorm(tau_centre, lower.tail = FALSE)
p_value
```

```
## [1] 0.5418029
```

2.3 Test de signe

2.3.1 Principe

Le principe de ce test est similaire au test de Kendall. Sous l'hypothèse d'indépendance et d'identité de distribution (i.i.d.), i.e. sous H_0, chaque observation a une chance sur deux de dépasser l'observation qui la précède :

$$\mathbb{P}(X_{i+1} > X_i) = \mathbb{P}(X_{i+1} < X_i) = \frac{1}{2}$$

La statistique de test est définie par :

$S = \sum_{i=1}^{n-1} \mathbb{I}(X_i > X_{i+1})$

où $\mathbb{I}(\cdot)$ est la fonction indicatrice. La statistique S représente donc le **nombre de différences positives inversées** dans l'échantillon.

- Si $X_1 < X_2 < \cdots < X_n$ (tendance croissante parfaite), alors $S = 0$
- Si $X_1 > X_2 > \cdots > X_n$ (tendance décroissante parfaite), alors $S = n - 1$

2.3.2 Loi de S sous H_0

Espérance :

$$\mathbb{E}(S) = \mathbb{E}\left(\sum_{i=1}^{n-1} \mathbb{I}(X_i > X_{i+1})\right) = \sum_{i=1}^{n-1} \mathbb{P}(X_i > X_{i+1}) = \frac{n-1}{2}$$

En posant $Z_i = \mathbb{I}(X_i > X_{i+1})$, on a $S = \sum_{i=1}^{n-1} Z_i$.

Variance :

$$\mathbb{V}(S) = \sum_{i=1}^{n-1} \mathbb{V}(Z_i) + \sum_{i \neq j} \mathrm{Cov}(Z_i, Z_j)$$

Il a été démontré que : $\mathbb{V}(S) = \frac{n+1}{12}$

2.3.3 Distribution asymptotique de S

Pour les petits échantillons ($n \leq 12$), on utilise la loi exacte de $S - \mathbb{E}(S)$, tabulée par **Moove et Wallis** (voir aussi table 16 de **Phillips-Tessi**).

- Lorsque $n \to \infty$, on a l'approximation normale suivante :

$$\frac{S - \frac{n-1}{2}}{\sqrt{\frac{n+1}{12}}} \sim \mathcal{N}(0, 1)$$

Avec **correction de continuité** :

$$\left|S - \frac{n-1}{2} - \frac{1}{2}\right| > z_{1-\frac{\alpha}{2}} \cdot \sqrt{\frac{n+1}{12}}$$

où $z_{1-\alpha/2}$ est le quantile de la loi normale tel que :

$$\mathbb{P}(|Z| > z_{1-\alpha/2}) = \alpha \quad \text{avec} \quad Z \sim \mathcal{N}(0,1)$$

Remarque : la correction de $-\frac{1}{2}$ est utilisée pour approximer une loi discrète (ici celle de S) par une loi continue (la loi normale).

$$\mathbb{P}(X_{i+1} > X_i) = \mathbb{P}(X_{i+1} < X_i) = \frac{1}{2}$$

La statistique de test est définie par :

$$S = \sum_{i=1}^{n-1} \mathbb{I}(X_i > X_{i+1})$$

où $\mathbb{I}(\cdot)$ est la fonction indicatrice. La statistique S représente donc le **nombre de différences positives inversées** dans l'échantillon.

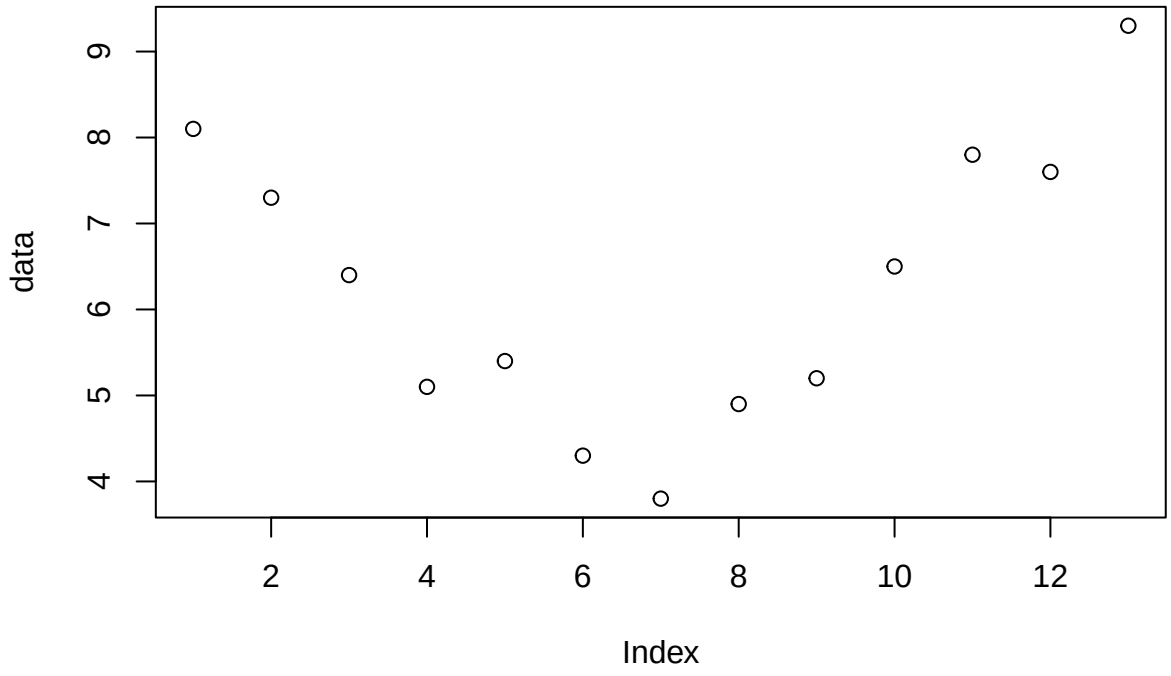
Si $X_1 < X_2 < \cdots < X_n$ (tendance croissante parfaite), la statistique de test compte le nombre de différences positives.

2.3.4 Données et visualisation

```
data=c(8.1,7.3,6.4,5.1,5.4,4.3,3.8,4.9,5.2,6.5,7.8,7.6,9.3)
data
```

```
##  [1] 8.1 7.3 6.4 5.1 5.4 4.3 3.8 4.9 5.2 6.5 7.8 7.6 9.3
```

```
plot(data)
```



2.3.5 Ecriture de la fonction de statistique de signe S

```
Signe <- function(data){
n=length(data)
total <-0
for (i in 1:(n-1)) {
if (data[i] > data[i+1]) {
total <- total + 1
}
}
return(total)
}
```

2.3.6 Application sur les données

```
resultat=Signe(data)
S=resultat
```

2.3.7 Calcul de l'Espérance et de la variance de S

```r
n=length(data)
E<- (n-1)/2
E
```

```
## [1] 6
```

```r
V <- (n+1)/12
V
```

```
## [1] 1.166667
```

On a n =13 > 12 donc on approxime la loi de S par la loi normale centrée et réduite

2.3.8 Approximation par la loi normale

```r
Z =(S - E- (1/2))/ sqrt(V)
Z
```

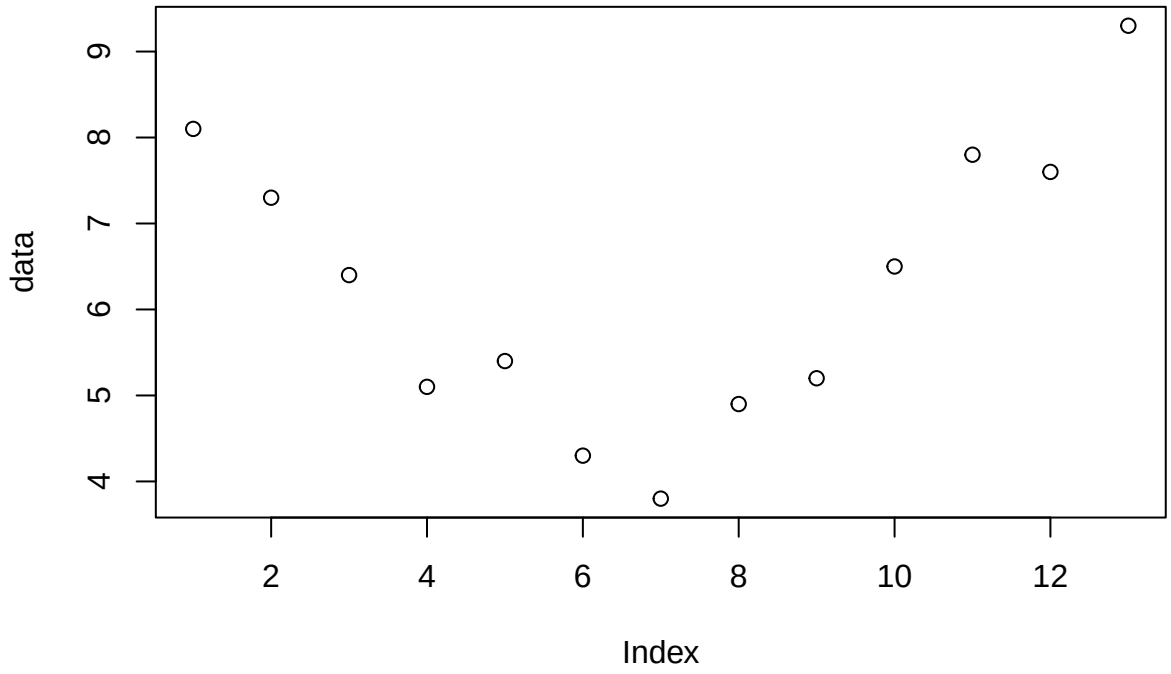
```
## [1] -0.46291
```

- **Pour un test bilateral**

```r
if (Z>qnorm(0.975, 0,1) | Z < -qnorm(0.975, 0,1)) {
"On rejette l'hypothese nulle : il y a une tendance."
} else {
"On ne peut pas rejetter l'hypothese nulle : pas de tendance ."
}
```

```
## [1] "On ne peut pas rejetter l'hypothese nulle : pas de tendance ."
```

```r
plot(data)
```



- **Pour un test unilatéral à gauche**

```
if (Z< -qnorm(0.95, 0,1) ) {
"On rejette l'hypothese nulle : dépendance négative."
} else {
"On ne peut pas rejetter l'hypothèse nulle"
}
```

```
## [1] "On ne peut pas rejetter l'hypothèse nulle"
```

- **Pour un test unilatéral à droite**

```
if (Z>qnorm(0.95, 0,1) ) {
"On rejette l'hypothese nulle : dépendance positive."
} else {
"On ne peut pas rejetter l'hypothèse nulle"
}
```

```
## [1] "On ne peut pas rejetter l'hypothèse nulle"
```

Commentaire : Dans ces trois cas, Z n'appartient pas à la région critique Donc on ne peut pas rejetter H0 : iid.

- **Avec la p-value pour un test bilatéral par exemple**

```r
p_value = 2*(1-pnorm(Z, 0,1,lower.tail = FALSE ))
p_value
```

```
## [1] 0.6434288
```

Conclusion : Comme la p-value est supérieur à 0.05 , on ne rejette pas Ho.

2.4 Test des séquences homogènes

Une séquence homogène est un groupe consécutif d'éléments identiques dans une série de données.

Exemple : dans la séquence `AAABBAAAB`, il y a trois séquences homogènes : `AAA`, `BB`, et `AAAA`.

Le test des séquences homogènes est utilisé pour déterminer si une série de données présente une tendance monotone (croissante ou décroissante) ou si elle est aléatoire.

Les hypothèses du test des séquences homogènes sont les suivantes :

- H_0 : la série de données est aléatoire, sans tendance monotone.
- Alternative unilatérale à droite : H_1 : la série de données est croissante.
- Alternative unilatérale à gauche : H_1 : la série de données est décroissante.
- Alternative bilatérale : H_1 : la série de données présente une tendance monotone quelconque.

2.4.1 Principe du test

Si une série de données contient une **tendance monotone** (croissante ou décroissante), les valeurs successives auront tendance à rester **au-dessus ou en dessous** d'une **valeur de référence**. On utilise en général la **médiane** de la série, notée λ, comme seuil.

On transforme les données selon :

$$A_i = \begin{cases} A & \text{si } X_i \geq \lambda \\ B & \text{si } X_i < \lambda \end{cases}$$

On dénombre ensuite le **nombre de séquences homogènes** : ce sont les blocs consécutifs de A ou de B.

2.4.2 Données

```r
(x <- c(8.1, 7.3, 6.4, 5.1, 5.4, 4.3, 3.8, 4.9, 5.2, 6.5, 7.8, 7.6, 9.3))
```

```
##  [1] 8.1 7.3 6.4 5.1 5.4 4.3 3.8 4.9 5.2 6.5 7.8 7.6 9.3
```

2.4.3 Calcul de la médiane

```r
(lambda <- median(x))
```

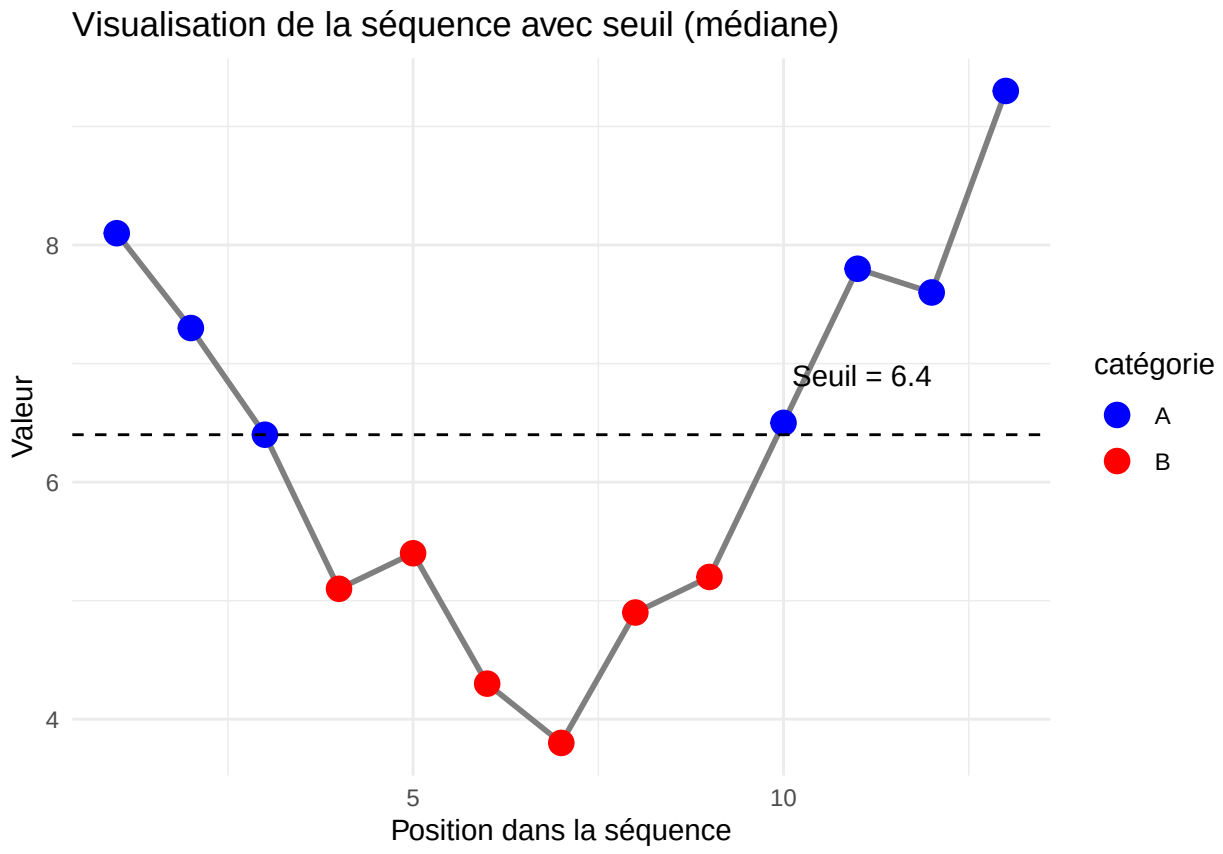
```
## [1] 6.4
```

2.4.4 Création de la suite A/B

```r
(X_bin <- ifelse(x >= lambda, 'A', 'B'))
```

```
##  [1] "A" "A" "A" "B" "B" "B" "B" "B" "B" "A" "A" "A" "A"
```

```r
df <- data.frame(index = 1:length(x), valeur = x, catégorie = X_bin)
# Représentation graphique avec ligne reliant les points
ggplot(df, aes(x = index, y = valeur, color = catégorie, group = 1)) +
  geom_line(color = "gray50", linewidth = 1) +       # Ligne reliant les points
  geom_point(size = 4) +                              # Points individuels
  geom_hline(yintercept = lambda, linetype = "dashed", color = "black") + # Ligne seuil
  annotate("text", x = length(x)-1, y = lambda + 0.5,
           label = paste("Seuil =", lambda), hjust = 1) +
  labs(title = "Visualisation de la séquence avec seuil (médiane)",
       x = "Position dans la séquence",
       y = "Valeur") +
  scale_color_manual(values = c("blue", "red")) +
  theme_minimal()
```



D'après la figure on observe de deux séquences homgènes de A et deux sequences homogènes de B donc notre statistique de test devrait être égale à 4.

Cependant , formellement , elle est construite de cette manière : On considère une série $(X_1, X_2, \dots, X_n)$ et une valeur seuil X_0.

On définit la variable Z_i par :

$$Z_i = \begin{cases} 1 & \text{si } X_i \text{ et } X_{i+1} \text{ sont différentes (AB ou BA)} \\ 0 & \text{sinon (AA ou BB)} \end{cases}$$

La statistique de test est donnée par :

$$S = \sum_{i=1}^{n-1} Z_i + 1$$

avec p le nombre d'éléments de A, m le nombre d'éléments de B et n=m+p .

2.4.5 Calcul de la Statistique de test : nombre de séquences homogènes (S)

Ici on compte les changements de groupe (AB ou BA) :

```
# Nombre de séquences homogènes
# p : nombre de A
# m : nombre de B
# n : taille de la séquence

p <- sum(X_bin == 'A')
m <- sum(X_bin == 'B')
n <- length(X_bin)
p
```

2.4.5.1 Commençons par calculer p, m, et n

```
## [1] 7
```

```
m
```

```
## [1] 6
```

```
n
```

```
## [1] 13
```

```r
S <- 1 + sum(as.numeric(X_bin[-1] != X_bin[-n]))
S
```

2.4.5.2 Valeur de la Statistique de test

```
## [1] 3
```

Pour de petits échantillons (p et m inferieur à 20), on utilise la table de **Swed et Eisenhart** et pour de grands échantillons, on utilise l 'approximation avec la loi normale. Cependant même pour de petits échantillons, on peut faire l'approximation par la loi normale.

Manuellement dans notre cas ou on a m=p=7 et Sur la table de **Swed et Eisenhart** on trouve que la region de rejet est S<=4 et S >=13 donc on rejette l'hypothèse nulle. Cela signifie que l'échantillon n'est pas iid.

Nous passons ensuite à l'approximation de la loi normale.

2.4.6 Approximation avec la loi normale

- **Espérance et variance sous Ho**

$$\mathbb{E}(S) = \frac{2pm}{n} + 1 \quad , \quad \mathrm{Var}(S) = \frac{2pm(2pm-n)}{n^2(n-1)}$$

```r
E_S <- 2 * p * m / n + 1
Var_S <- (2 * p * m * (2 * p * m - n)) / (n^2 * (n - 1))
E_S
```

```
## [1] 7.461538
```

```r
Var_S
```

```
## [1] 2.940828
```

- **Statistique de test (approximation normale)**

$$Z = \frac{S - \mathbb{E}(S)}{\sqrt{\mathrm{Var}(S)}} \sim N(0,1)$$

```r
Z <- (S - E_S) / sqrt(Var_S)
Z
```

```
## [1] -2.601656
```

- **p-value bilatérale**

```r
2 * (1 - pnorm(abs(Z)))
```

```
## [1] 0.009277498
```

On rejette H_0 pour alpha= 0.05.

2.4.7 Test automatisé avec R

- **Utilisation du package `randtests`**

```r
library(randtests)
```

- **Mise en oeuvre du test**

Lorsque la moyenne figure parmi les observations, les résultats du test peuvent différer de ceux de la fonction `runs.test`, laquelle scinde les données en comparant strictement chaque valeur à la médiane.

```r
randtests::runs.test(x, threshold = median(x),
                     alternative = "two.sided", pvalue = "normal")
```

```
##
##  Runs Test
##
## data:  x
## statistic = -2.4221, runs = 3, n1 = 6, n2 = 6, n = 12, p-value =
## 0.01543
## alternative hypothesis: nonrandomness
```

Interprétation : Si la **p-value est petite** (< 0.05), on **rejette H** : présence probable d'une **tendance**. Sinon, on **ne rejette pas H** : l'échantillon est iid.

Comme la p-value est inférieure à 0.05 alors on conclut qu'il y a la présence d'une tendance monotone.

Table de Swed et Eisenhart

TABLE G

Critical values of r in the runs test*

Given in the tables are various critical values of r for values of m and n less than or equal to 20. For the one-sample runs test, any observed value of r which is less than or equal to the smaller value, or is greater than or equal to the larger value in a pair is significant at the $\alpha = .05$ level.

m \ n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2											2	2	2	2	2	2	2	2	2
											–	–	–	–	–	–	–	–	–
3					2	2	2	2	2	2	2	2	2	3	3	3	3	3	3
					–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
4				2	2	2	3	3	3	3	3	3	3	3	4	4	4	4	4
				9	9	–	–	–	–	–	–	–	–	–	–	–	–	–	–
5			2	2	3	3	3	3	3	4	4	4	4	4	4	4	5	5	5
			9	10	10	11	11	–	–	–	–	–	–	–	–	–	–	–	–
6		2	2	3	3	3	3	4	4	4	4	5	5	5	5	5	5	6	6
		–	9	10	11	12	12	13	13	13	13	–	–	–	–	–	–	–	–
7		2	2	3	3	3	4	4	5	5	5	5	5	6	6	6	6	6	6
		–	–	11	12	13	13	14	14	14	14	15	15	15	–	–	–	–	–
8		2	3	3	3	4	4	5	5	5	6	6	6	6	6	7	7	7	7
		–	–	11	12	13	14	14	15	15	16	16	16	16	17	17	17	17	17
9		2	3	3	4	4	5	5	5	6	6	6	7	7	7	7	8	8	8
		–	–	–	13	14	14	15	16	16	16	17	17	18	18	18	18	18	18
10		2	3	3	4	5	5	5	6	6	7	7	7	7	8	8	8	8	9
		–	–	–	13	14	15	16	16	17	17	18	18	18	19	19	19	20	20
11		2	3	4	4	5	5	6	6	7	7	7	8	8	8	9	9	9	9
		–	–	–	13	14	15	16	17	17	18	19	19	19	20	20	20	21	21
12	2	2	3	4	4	5	6	6	7	7	7	8	8	8	9	9	9	10	10
	–	–	–	–	13	14	16	16	17	18	19	19	20	20	21	21	21	22	22
13	2	2	3	4	5	5	6	6	7	7	8	8	9	9	9	10	10	10	10
	–	–	–	–	–	15	16	17	18	19	19	20	20	21	21	22	22	23	23
14	2	2	3	4	5	5	6	7	7	8	8	9	9	9	10	10	10	11	11
	–	–	–	–	–	15	16	17	18	19	20	20	21	22	22	23	23	23	24
15	2	3	3	4	5	6	6	7	7	8	8	9	9	10	10	11	11	11	12
	–	–	–	–	–	15	16	18	18	19	20	21	22	22	23	23	24	24	25
16	2	3	4	4	5	6	6	7	8	8	9	9	10	10	11	11	11	12	12
	–	–	–	–	–	–	17	18	19	20	21	21	22	23	23	24	25	25	25
17	2	3	4	4	5	6	7	7	8	9	9	10	10	11	11	11	12	12	13
	–	–	–	–	–	–	17	18	19	20	21	22	23	23	24	25	25	26	26
18	2	3	4	5	5	6	7	8	8	9	9	10	10	11	11	12	12	13	13
	–	–	–	–	–	–	17	18	19	20	21	22	23	24	25	25	26	26	27
19	2	3	4	5	6	6	7	8	8	9	10	10	11	11	12	12	13	13	13
	–	–	–	–	–	–	17	18	20	21	22	23	23	24	25	26	26	27	27
20	2	3	4	5	6	6	7	8	9	9	10	10	11	12	12	13	13	13	14
	–	–	–	–	–	–	17	18	20	21	22	23	24	25	25	26	27	27	28

* Adapted from Swed, and Eisenhart, C. (1943). Tables for testing randomness of grouping in a sequence of alternatives. *Annals of Mathematical Statistics*, **14**, 83–86, with the kind permission of the authors and publisher.

2.5 Test de localisation : test du signe

```r
vecteur <- seq(-5, 5, length.out = 100)
```

2.5.1 Statistique de test

```r
sta_test <- function(x) {
  # Vérifier que c'est un vecteur numérique
  if (!is.numeric(x) || !is.vector(x)) {
    stop("L'entrée doit être un vecteur numérique.")
  }

  # Calcul des rangs avec moyenne en cas d'égalité
  rangs <- rank(x, ties.method = "average")

  # Filtrer les indices des éléments positifs
  indices_positifs <- which(x > 0)

  # Extraire les rangs correspondants aux valeurs positives
  rangs_positifs <- rangs[indices_positifs]

  # Retourner la somme des valeurs absolues de ces rangs
  return(sum(abs(rangs_positifs)))
}
```

2.5.2 Test avec la loi normale

```r
# Fonction principale de test
test_normal <- function(x, alpha = 0.05, type = "bilateral") {
  if (!type %in% c("bilateral", "gauche", "droite")) {
    stop("Le type de test doit être 'bilateral', 'gauche' ou 'droite'.")
  }

  # Statistique brute
  stat <- sta_test(x)

  # Moyenne et variance de x
  moyenne <- mean(x)
  variance <- var(x)

  if (variance == 0) {
    stop("La variance du vecteur est nulle. Le test n'est pas applicable.")
  }
```

```r
  # Statistique normalisée
  T_stat <- (stat - moyenne) / sqrt(variance)

  # Comparaison selon le type de test
  decision <- ""
  if (type == "bilateral") {
    seuil <- qnorm(1 - alpha / 2)
    decision <- ifelse(abs(T_stat) > seuil, "Rejeter H0", "Ne pas rejeter H0")
  } else if (type == "droite") {
    seuil <- qnorm(1 - alpha)
    decision <- ifelse(T_stat > seuil, "Rejeter H0", "Ne pas rejeter H0")
  } else if (type == "gauche") {
    seuil <- qnorm(alpha)
    decision <- ifelse(T_stat < seuil, "Rejeter H0", "Ne pas rejeter H0")
  }

  # Résultat
  return(list(
    statistique = T_stat,
    seuil = seuil,
    decision = decision
  ))
}
```

```r
test_normal(vecteur)
```

```
## $statistique
## [1] 1288.196
##
## $seuil
## [1] 1.959964
##
## $decision
## [1] "Rejeter H0"
```

2.6 Test de Wilcoxon : avec la fonction de R

```r
# Fonction de test utilisant le test de Wilcoxon
test_wilcoxon <- function(x, alpha = 0.05, type = "bilateral") {
  if (!is.numeric(x) || !is.vector(x)) {
    stop("L'entrée doit être un vecteur numérique.")
  }

  if (!type %in% c("bilateral", "gauche", "droite")) {
    stop("Le type de test doit être 'bilateral', 'gauche' ou 'droite'.")
  }
```

```r
  # Traduction du type de test pour wilcox.test()
  alternative <- switch(type,
                        "bilateral" = "two.sided",
                        "gauche" = "less",
                        "droite" = "greater")

  # Test de Wilcoxon signé
  test <- wilcox.test(x, mu = 0, alternative = alternative, exact = FALSE,
                      correct = FALSE)

  # Décision
  decision <- ifelse(test$p.value < alpha, "Rejeter H0", "Ne pas rejeter H0")

  # Résultat
  return(list(
    statistique = test$statistic,
    p_value = test$p.value,
    decision = decision
  ))
}
```

```r
test_wilcoxon(vecteur)
```

```
## $statistique
##      V
## 2519.5
##
## $p_value
## [1] 0.984912
##
## $decision
## [1] "Ne pas rejeter H0"
```

3 Chapitre 3 : Tests non paramétriques pour 2 échantillons

3.1 Tests d'indépendance

3.1.1 Test de Spearman

On dispose de deux échantillons X et Y appariées. On veut tester H0: X et Y sont indépendants contre H1: X et Y sont dépendants.

La statistique de test considérée est donnée par:

$$T = \sum_{i=1}^{n} R_i S_i$$

Avec R_i et S_i qui sont les rangs de l'observation i dans X et Y.

- **Fonction manuelle**

```r
spearman_test_manual <- function(X, Y, alternative = "two.sided", alpha = 0.05) {
  n <- length(X)
  if (length(Y) != n) stop("Les variables doivent avoir la même longueur")

  # 1. Vérification du paramètre alternative
  alternative <- match.arg(alternative, c("two.sided", "greater", "less"))

  # 2. Calcul des rangs avec gestion des ex-aequo
  rank_X <- rank(X, ties.method = "average")
  rank_Y <- rank(Y, ties.method = "average")

  # 3. Calcul de la statistique T = Σ(Ri * Si)
  T_stat <- sum(rank_X * rank_Y)

  # 4. Correction pour les ex-aequo
  correct_ties <- function(ranks) {
    tied_groups <- table(ranks)
    sum(tied_groups^3 - tied_groups) / 12
  }
  TX <- correct_ties(rank_X)
  TY <- correct_ties(rank_Y)

  # 5. Calcul de la variance
  mean_rank <- (n + 1)/2
  SS_R <- sum(rank_X^2) - n * mean_rank^2
  SS_S <- sum(rank_Y^2) - n * mean_rank^2
  var_T <- (SS_R*SS_S)/(n-1) - (TX*TY)/(n-1) + (TX*SS_S + TY*SS_R)/(n*(n^2 - 1))
  # Note: Sans ex-aequo, Var(T) = (SS_R * SS_S)/(n-1) = n²(n+1)²(n-1)/144

  # 6. Statistique de test
  mu_T <- n * mean_rank^2
```

```r
Z <- (T_stat - mu_T)/sqrt(var_T)

# 7. Calcul du coefficient de Spearman
rho <- (12 * (T_stat - mu_T))/(n^3 - n - 12*(TX + TY) + 12*(TX*TY)/(n-1))

# 8. Valeurs critiques et p-value selon l'alternative
if (alternative == "two.sided") {
  Z_critical <- qnorm(1 - alpha/2)
  p_value <- 2 * pnorm(-abs(Z))
  decision_rule <- paste("|Z| >", round(Z_critical, 4))
} else if (alternative == "greater") {
  Z_critical <- qnorm(1 - alpha)
  p_value <- pnorm(Z, lower.tail = FALSE)
  decision_rule <- paste("Z >", round(Z_critical, 4))
} else { # "less"
  Z_critical <- qnorm(alpha)
  p_value <- pnorm(Z)
  decision_rule <- paste("Z <", round(Z_critical, 4))
}

# 9. Décision
decision <- ifelse(
  (alternative == "two.sided" & abs(Z) > Z_critical) |
    (alternative == "greater" & Z > Z_critical) |
    (alternative == "less" & Z < Z_critical),
  paste("Rejeter H0 au seuil", alpha),
  paste("Ne pas rejeter H0 au seuil", alpha)
)

# Affichage des résultats
cat("Test de Spearman Manuel\n")
cat("-----------------------\n")
cat("Type de test    :", switch(alternative, "two.sided" = "Bilatéral (dépendance quelconqu
cat("Statistique T   :", round(T_stat, 4), "\n")
cat("Statistique Z   :", round(Z, 4), "\n")
cat("Valeur critique :", round(Z_critical, 4), "\n")
cat("Règle de décision:", decision_rule, "\n")
cat("p-value         :", sprintf("%.6f", p_value), "\n")
cat("Coefficient     :", round(rho, 4), "\n")
cat("Décision        :", decision, "\n")
cat("-----------------------\n")

return(list(
  alternative = alternative,
  T_stat = T_stat,
  Z = Z,
  Z_critical = Z_critical,
  p_value = p_value,
```

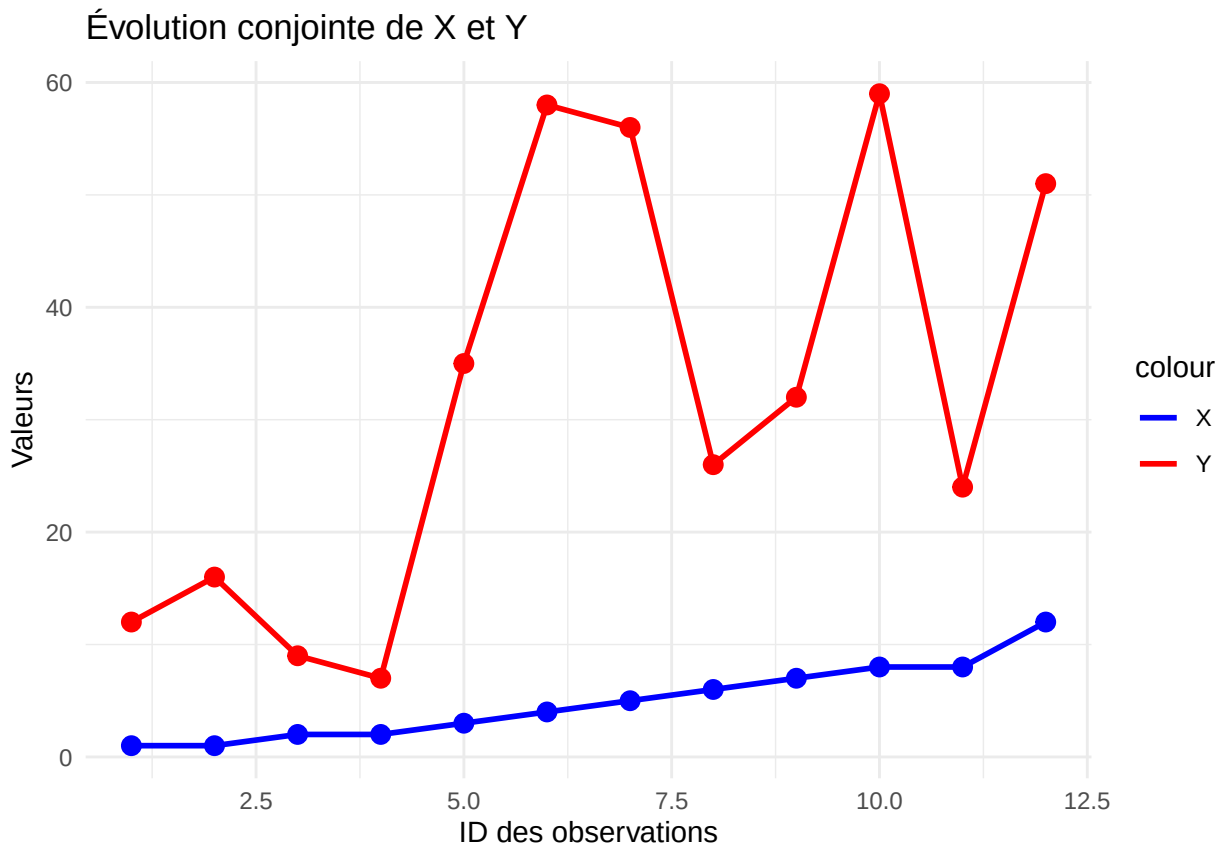
```
    rho = rho,
    decision = decision
  ))
}
```

- **Exemple d'application et comparaison avec la fonction intégrée de R**

Ce test doit normalement être effectué avec au moins 30 observations. mais nous utilisons juste les données de l'exemple d'application du cours ici.

- **Données et visualisation**

```
# ----------------------------
# 1. Données de l'application du cours
# ----------------------------

data <- data.frame(
  ID= c(1,2,3,4,5,6,7,8,9,10,11,12),
  X= c(1,1,2,2,3,4,5,6,7,8,8,12),
  Y= c(12,16,9,7,35,58,56,26,32,59,24,51))

# ----------------------------
# 2. REPRÉSENTATION GRAPHIQUE
# ----------------------------
library(ggplot2)

ggplot(data, aes(x = ID)) +
  geom_line(aes(y = X, color = "X"), linewidth = 1) +
  geom_line(aes(y = Y, color = "Y"), linewidth = 1) +
  geom_point(aes(y = X), color = "blue", size = 3) +
  geom_point(aes(y = Y), color = "red", size = 3) +
  scale_color_manual(values = c("X" = "blue", "Y" = "red")) +
  labs(title = "Évolution conjointe de X et Y",
       x = "ID des observations",
       y = "Valeurs") +
  theme_minimal()
```



On a l'impression que les deux echantillons sont indépendants car X a une tendance plutôt croissante tandis que Y croit puis décroit de temps à autre. toutefois, les tendances des deux échantillons sont globalement croissantes. Voyons les résultats fournis par les tests.

- **Tests**

```
# ----------------------------
#  TEST INTÉGRÉ DE SPEARMAN ET TEST MANUEL
# ----------------------------
# Choisir l'alternative en fonction du graphique: "two.sided", "greater", ou "less"

alternative_choice <- "two.sided"

result_manual <- spearman_test_manual(data$X, data$Y, alternative = alternative_choice)
```

```
## Test de Spearman Manuel
## ----------------------
## Type de test    : Bilatéral (dépendance quelconque)
## Statistique T   : 594.5
## Statistique Z   : 2.0401
## Valeur critique : 1.96
## Règle de décision: |Z| > 1.96
## p-value         : 0.041344
## Coefficient     : 0.6184
## Décision        : Rejeter H0 au seuil 0.05
## ----------------------
```

```r
result_builtin <- cor.test(data$X, data$Y, method = "spearman", exact = FALSE,
                           alternative = alternative_choice)


# ----------------------------
#  COMPARAISON DES RÉSULTATS
# ----------------------------

cat("\n=== Résultats du test intégré ===\n")
```

```
##
## === Résultats du test intégré ===
```

```r
print(data.frame(
  Coefficient_rho = result_builtin$estimate,
  P_value = result_builtin$p.value,
  Méthode = result_builtin$method
))
```

```
##     Coefficient_rho    P_value                          Méthode
## rho       0.6151228 0.03326601 Spearman's rank correlation rho
```

Les deux tests rejettent H0 au seuil 0.05. donc il y a dépendance entre les deux échantillons au seuil 0.05. Mais si on prend un faible risque comme 1%, on ne va plus rejeter H0, donc les 2 échantillons seront indépendants dans ce cas là, ce qui est en ligne avec l'observation faite sur le graphique.

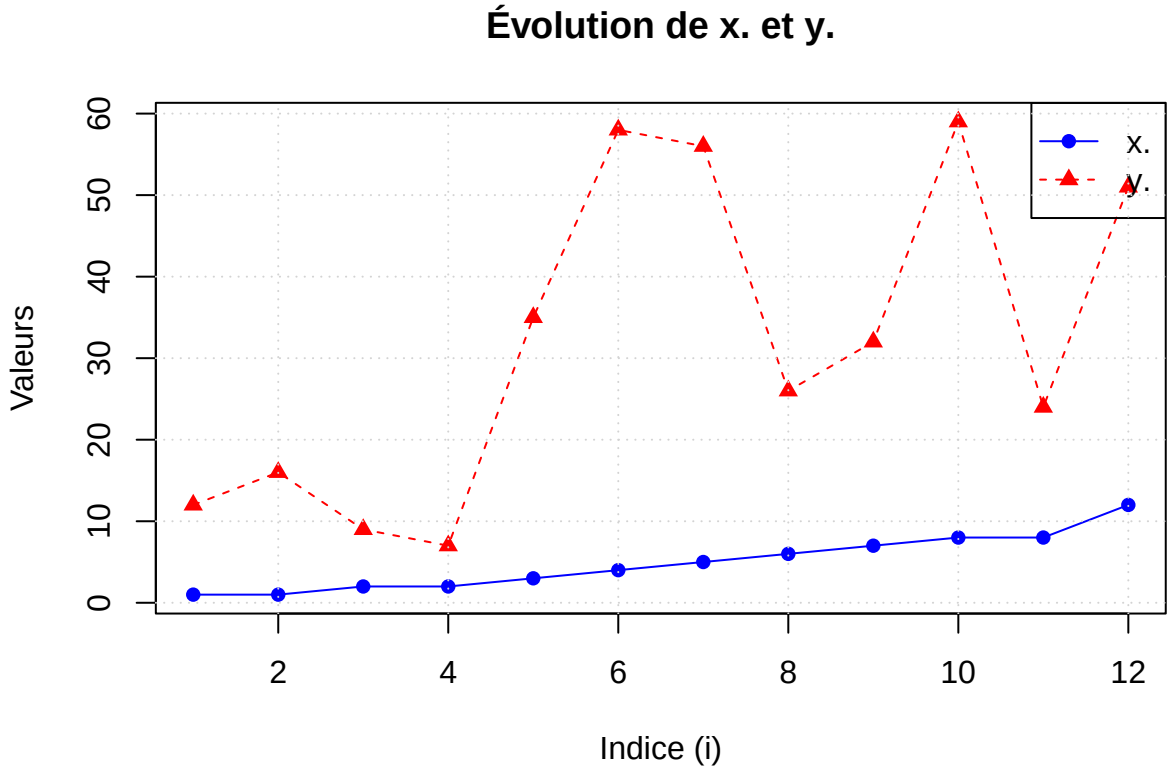
3.1.2 Test de Kendall: cas apparié

- **Données**

```r
# Données du cours
x <- c(1,1,2, 2, 3, 4, 5, 6, 7, 8, 8, 12)
y <- c(12, 16, 9, 7, 35, 58, 56, 26, 32, 59, 24, 51)
```

- **Visualisation**

```r
# Génération de l'indice i
i <- 1:length(x)

# Tracé des deux courbes
plot(i, x, type = "o", col = "blue", pch = 16, ylim = range(c(x, y)),
     xlab = "Indice (i)", ylab = "Valeurs", main = "Évolution de x  et y ")
lines(i, y, type = "o", col = "red", pch = 17, lty = 2)

# Légende
legend("topright", legend = c("x ", "y "),
       col = c("blue", "red"), pch = c(16, 17), lty = c(1, 2))
grid()
```



Le principe est le même que dans le cas d'un échantillon. On regarde le nombre de paires concordantes entre (Xi,Yi) et (Xj,Yj).

- **Hypothèses :**

$$\begin{cases} H_0 : \text{Les données sont iid} \\ H_1 : \text{les données ne sont pas iid} \end{cases}$$

- **La statistique de Kendal**
  - **Calcul de Q**

Il y'a concordante entre ces deux couples si :

$(Xi - Xj)(Yj - Yi) > 0$ i.e $Xj > Xi$ et $Yj > Yi$ ou $Xj < Xi$ et $Yj < Yi$

Donc on a:

$$Q = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{1}\left[(x_j - x_i)(y_j - y_i) > 0\right]$$

```r
QKendall <- function(x, y) {
  n <- length(x)
  count <- 0
  for (i in 1:(n-1)) {
    for (j in (i+1):n) {
      if ((x[j] - x[i]) * (y[j] - y[i]) > 0) {
        count <- count + 1
      }
    }
  }
  return(count)
}
print("le Q Kendall est : ")
```

```
## [1] "le Q Kendall est : "
```

```r
QKendall(x,y)
```

```
## [1] 44
```

- **Calcul du τ**

On pose: $\tau = 1 - \frac{4Q}{n(n-1)}$

```r
StatTest <- function(x, y) {
  n <- length(x)
  Q <- QKendall(x, y)
  tau <- 1 - (4 * Q) / (n * (n - 1))
  return(tau)
}

tau <- StatTest(x, y)
tau
```

```
## [1] -0.3333333
```

Pour n assez grand, la valeur Z observée se calcule comme suit :

$$Z_{obs} = \frac{\tau - \mathbb{E}(\tau)}{\sqrt{Var(\tau)}}$$

Avec:

$$E(\tau) = 1 - \frac{4n(n-1)}{4n(n-1)} = 0$$

$$V(\tau) = \frac{2(2n+5)}{9n(n-1)}$$

```r
Zobs <- function(x, y) {
  n <- length(x)
  tau <- StatTest(x, y)
  variance_tau <- (2 * (2 * n + 5)) / (9 * n * (n - 1))
  z <- tau / sqrt(variance_tau)
  return(z)
}

z_obs <- Zobs(x, y)
z_obs
```

```
## [1] -1.508596
```

- **Région critique et prise de décision**

Soit un seuil de signification α donné. La statistique de test Z_{obs} suit approximativement une loi normale $\mathcal{N}(0,1)$ sous l'hypothèse nulle (H_0).

3.1.2.1 Test bilatéral

$$\text{Rejeter } H_0 \text{ si } |Z_{\text{obs}}| > z_{1-\alpha/2}$$

```r
Kendall_Test <- function(risque, x, y) {
  zobs <- Zobs(x, y)
  seuil <- qnorm(1 - risque / 2)  # Bilatéral

  cat("Z observé =", round(zobs, 3), "\n")
  cat("Seuil critique (bilatéral) =", round(seuil, 3), "\n")

  if (abs(zobs) > seuil) {
    print("On rejette H0 : Il existe une dépendance significative entre X et Y.")
  } else {
    print("On ne rejette pas H0 : Pas de dépendance significative entre X et Y.")
  }
}
Kendall_Test(0.1, x, y)
```

```
## Z observé = -1.509
## Seuil critique (bilatéral) = 1.645
## [1] "On ne rejette pas H0 : Pas de dépendance significative entre X et Y."
```

Pour n faible alors :

- **Définition de la statistique S**

On a : - Q : nombre de paires **concordantes** - Q' : nombre de paires **discordantes**

```r
# Fonction qui retourne les valeurs de Q (concordant), Q' (discordant), et S
SKendall <- function(x, y) {
  n <- length(x)
  Q_concordant <- 0
  Q_discordant <- 0

  for (i in 1:(n - 1)) {
    for (j in (i + 1):n) {
      produit <- (x[j] - x[i]) * (y[j] - y[i])

      if (produit > 0) {
        Q_concordant <- Q_concordant + 1
      } else if (produit < 0) {
        Q_discordant <- Q_discordant + 1
      }
      # si produit == 0 → égalité → on ignore
    }
  }

  S <- Q_discordant - Q_concordant

  return(list(
    Q_concordant = Q_concordant,
    Q_discordant = Q_discordant,
    S = S
  ))
}
result=SKendall(x,y)

print(paste("Q concordant =", result$Q_concordant))
```

```
## [1] "Q concordant = 44"
```

```r
print(paste("Q discordant =", result$Q_discordant))
```

```
## [1] "Q discordant = 19"
```

```r
print(paste("S =", result$S))
```

```
## [1] "S = -25"
```

- **Test bilatéral** : $Rejeter H_0 si |S| > s $

D'après la table de Kendall S= 28 donc on ne rejette pas H_0.

```r
cor.test(x, y, method = "kendall", exact = TRUE)
```

3.1.2.2 Avec la fonction intégrée R

```
##
##  Kendall's rank correlation tau
##
## data:  x and y
## z = 1.7265, p-value = 0.08425
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## 0.3877018
```

3.2 Tests d'alternative de position

3.2.1 Test de Wilcoxon

Le test de Wilcoxon permet de comparer la position (typiquement la médiane) de deux échantillons pour évaluer s'ils proviennent de populations ayant le même paramètre de position.

Hypothèse nulle (H_0) : Les deux groupes ont la même distribution, donc le même paramètre de position (médiane).

On considère l'échantillon fusionné $(X_1, X_2, \dots, X_n, Y_1, Y_2, \dots, Y_m) = (X_1, X_2, \dots, X_{n+m})$ et on note par R_i le rang de X_i dans l'échantillon fusionné.

La statistique de Wilcoxon est : $W_n = \sum_{i=1}^{n} R_i$

$$\textbf{La région critique est :} \quad W = \begin{cases} W_n < c & \text{pour } H_1 : \theta > 0 \quad \text{alors } Y \gg X \\ W_n > c & \text{pour } H_1 : \theta < 0 \quad \text{alors } X \gg Y \\ W_n < c \text{ ou } W_n > c & \text{pour } H_1 : \theta \neq 0 \end{cases}$$

Sous H_0, W_n est symétrique autour de sa moyenne.

$$\mathbb{E}(W_n) = \sum \mathbb{E}(R_i) = n \cdot \mathbb{E}(R_i) = n \cdot \frac{n+m+1}{2}$$

On remarque également : $\mathbb{V}(W_n) = \frac{nm(n+m+1)}{12}$

Les valeurs extrêmes de W_n sont :

a. Si tous les $X_i < Y_i \Rightarrow W_n = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$

b. Si tous les $X_i > Y_i \Rightarrow W_n = (m+1) + (m+2) + \cdots + (m+n)$

$$\Rightarrow W_n = nm + \frac{n(n+1)}{2}$$

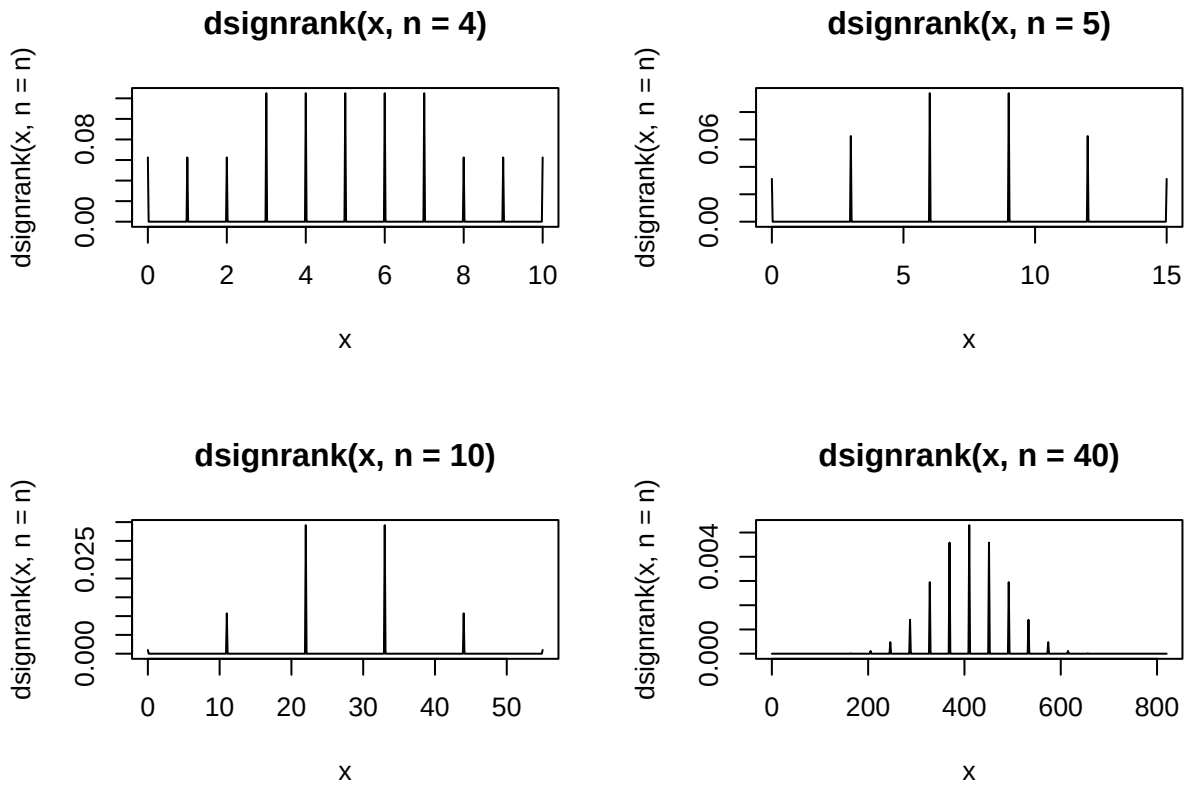
Pour des échantillons de petite taille, on utilise la table de Wilcoxon.

Pour des grands échantillons, on utilise l'approximation :

$$\frac{W_n - \mathbb{E}(W_n)}{\sqrt{\mathbb{V}(W_n)}} \longrightarrow \mathcal{N}(0,1)$$

- **Simulation**

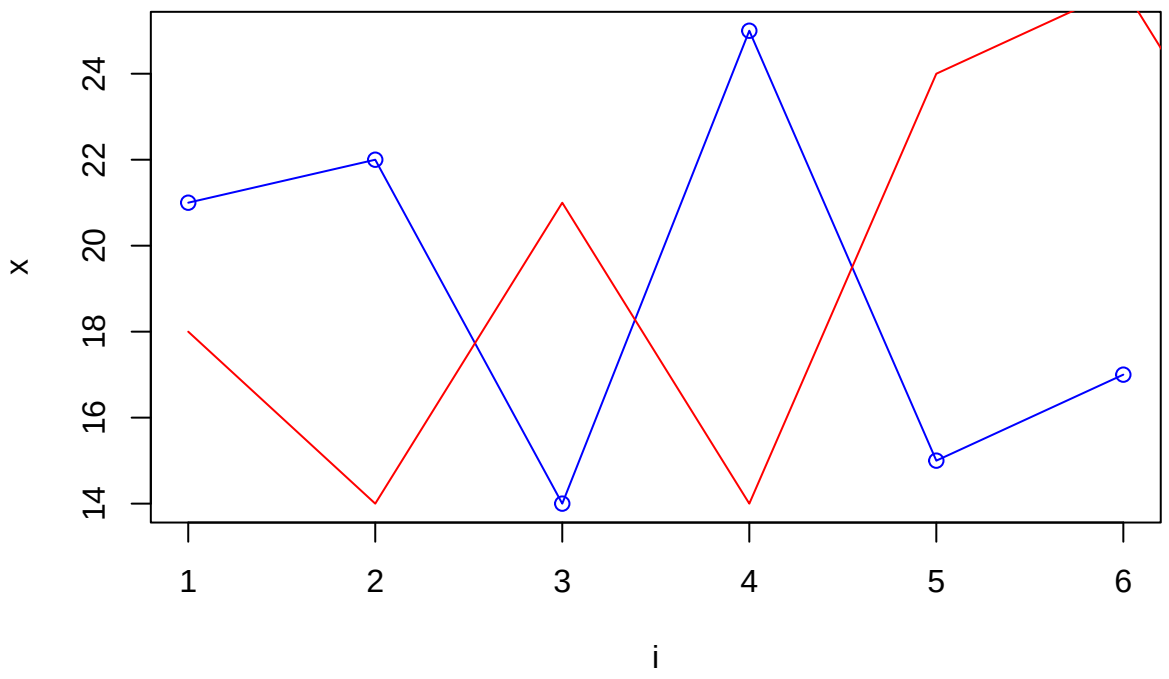
```r
require(graphics)
par(mfrow = c(2,2))
for(n in c(4:5,10,40)) {
  x <- seq(0, n*(n+1)/2, length.out = 501)
  plot(x, dsignrank(x, n = n), type = "l",
       main = paste0("dsignrank(x, n = ", n, ")"))
}
```



- **Données**

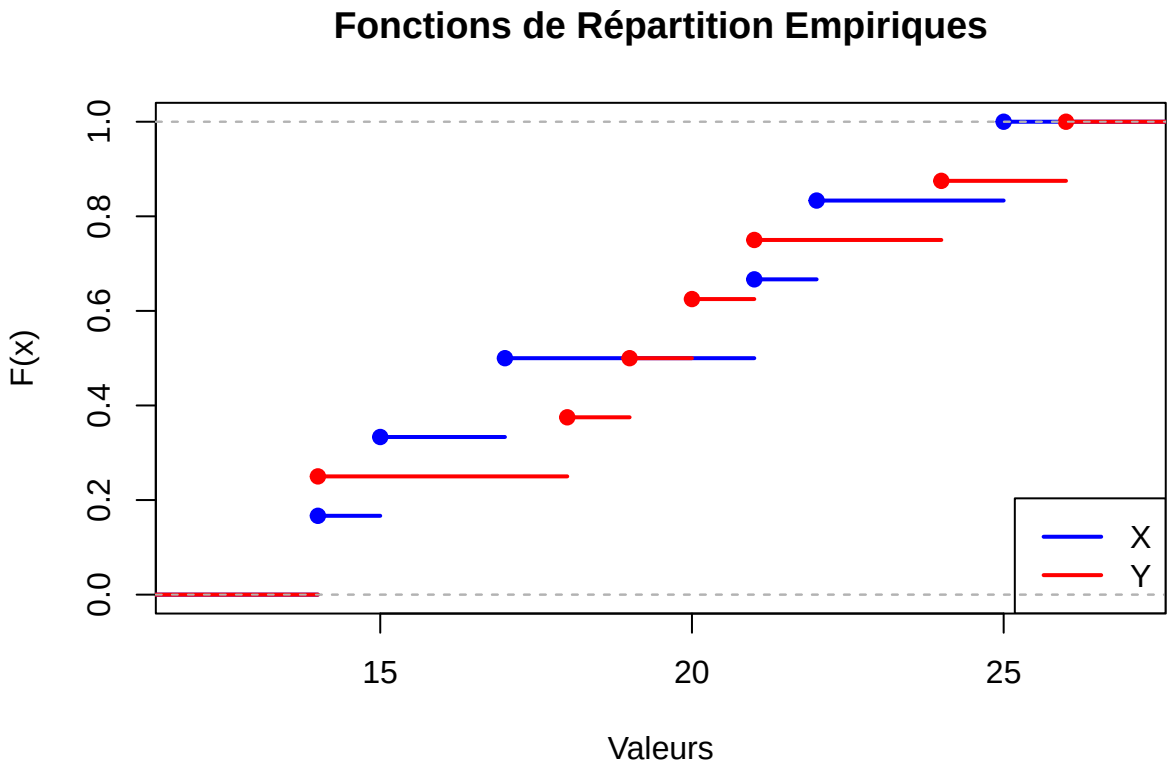
```
# Simulation de deux échantillons indépendants
x <- c(21,22, 14, 25,15,17)   # Échantillon X
y <- c(18, 14, 21,14,24,26,19,20)   # Échantillon Y
n <- length(x)
m <- length(y)
N <- n+m
```

```
i <- 1:n
plot(i, x, col = "blue" )
lines(x, col = "blue" )
lines(y, col = "red")
```

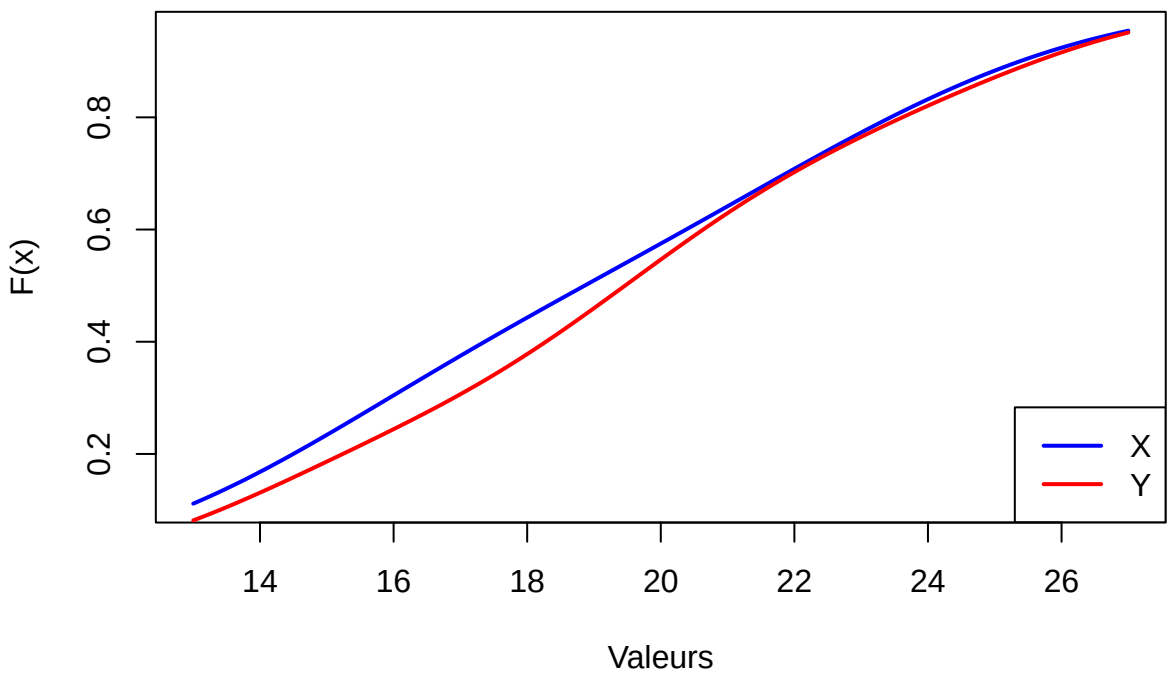


- **Visualisation : Fonctions de répartition empiriques**

```
plot(ecdf(x), main = "Fonctions de Répartition Empiriques", xlab = "Valeurs",
     ylab = "F(x)", col = "blue", lwd = 2)
lines(ecdf(y), col = "red", lwd = 2)
legend("bottomright", legend = c("X", "Y"), col = c("blue", "red"), lwd = 2)
```




```r
# Estimation de la densité avec densité par noyau
d_x <- density(x)
d_y <- density(y)

# Fonction de répartition obtenue par intégration numérique
Fx <- approxfun(d_x$x, cumsum(d_x$y) / sum(d_x$y))
Fy <- approxfun(d_y$x, cumsum(d_y$y) / sum(d_y$y))

# Séquence de valeurs communes pour comparer
x_vals <- seq(min(c(x, y)) - 1, max(c(x, y)) + 1, length.out = 500)

# Tracer les courbes lissées
plot(x_vals, Fx(x_vals), type = "l", col = "blue", lwd = 2,
     xlab = "Valeurs", ylab = "F(x)", main = "Fonctions de Répartition Lissées")
lines(x_vals, Fy(x_vals), col = "red", lwd = 2)
legend("bottomright", legend = c("X", "Y"), col = c("blue", "red"), lwd = 2)
```



- **Calcul de la statistique Wn**

```
# Échantillon combiné
data_combined <- c(x, y)

rangs <- rank(data_combined, ties.method ="average")
rangs
```

```
##  [1]  9.5 11.0  2.0 13.0  4.0  5.0  6.0  2.0  9.5  2.0 12.0 14.0  7.0  8.0
```

```
print("")
```

```
## [1] ""
```

```
# Attribution des rangs à X
rangs_x <- rangs[1:n]
rangs_x
```

```
## [1]  9.5 11.0  2.0 13.0  4.0  5.0
```

```r
print("")
```

```
## [1] ""
```

```r
# Attribution des rangs à X
k=n+1
rangs_y <- rangs[k:N]
rangs_y
```

```
## [1]  6.0  2.0  9.5  2.0 12.0 14.0  7.0  8.0
```

```r
# Statistique de Wilcoxon Wn
Wn <- sum(rangs_x)
Wn
```

```
## [1] 44.5
```

```r
# somme des rangs de y
Wy <- sum(rangs_y)
Wy
```

```
## [1] 60.5
```

- **Valeur critique au seuil de 5**

La fonction *qwilcox* est développé pour la statistique de test de Wilcoxon Mann-Withney définie par :

$$U_n = W_n - n(n+1)/2$$

Donc pour avoir le seuil critique correspondant à la statistique Wn, il faut ajouter le terme n(n+1)/2

```r
seuil <- qwilcox(p = 0.05, n, m, lower.tail =F) + n*(n+1)/2
seuil
```

```
## [1] 58
```

- **Prise de décision**

```r
# Test pour H1: X >> Y
if (Wn > seuil) {
  cat("→ Décision : H0 est rejetée au seuil de 5% - alors X  domine Y.\n")
} else {
  cat("→ Décision : H0 n'est pas rejetée - aucune dominance significative de X sur Y.\n")
}
```

```
## → Décision : H0 n'est pas rejetée - aucune dominance significative de X sur Y.
```

3.2.2 Test de wilcoxon disponible sur R

- **Comparaison**

```r
# Test de Wilcoxon pour H1: X >> Y
wilcox.test(x, y, alternative = "great", exact = FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  x and y
## W = 23.5, p-value = 0.5516
## alternative hypothesis: true location shift is greater than 0
```

- **Fonction**

```r
test_wilcoxon <- function(x, y, alternative = "droite", alpha=0.05) {
  n <- length(x)
  m <- length(y)

  # Fusion des données et calcul des rangs
  data_combinee <- c(x, y)
  rangs <- rank(data_combinee)
  W <- sum(rangs[1:n])
  cat("Statistique de Wilcoxon W =", W, "\n")

  # Définir type d'alternative
  alt <- match.arg(alternative, choices = c("gauche", "droite", "bilateral"))

  # Test exact si petit (< 30)
  if (n <= 25 | m <= 25  ) {
    cat("Méthode : test exact basé sur la table de Wilcoxon\n")

    if (alt == "droite") {
      seuil <- qwilcox(1-alpha, n, m, lower.tail = TRUE)+ n*(n+1)/2
      cat("Seuil critique (droite, 5%) :", seuil, "\n")
      if (W > seuil) {
        cat("→ H  rejeté, alors X domine Y.\n")
      } else {
        cat("→ H  non rejetée\n")
      }

    } else if (alt == "gauche") {
      seuil <- qwilcox(alpha, n, m, lower.tail = TRUE)+ n*(n+1)/2
      cat("Seuil critique (gauche, 5%) :", seuil, "\n")
      if (W < seuil) {
        cat("→ H  rejeté alors Y domine X.\n")
```

```
    } else {
      cat("→ H  non rejetée\n")
    }

  } else if (alt == "bilateral") {
    seuil_inf <- qwilcox(alpha, n, m, lower.tail = TRUE)+ n*(n+1)/2
    seuil_sup <- qwilcox(1-alpha, n, m, lower.tail = TRUE)+ n*(n+1)/2
    cat("Seuils critiques (bilatéral 5%) : <", seuil_inf, " ou >", seuil_sup, "\n")
    if (W < seuil_inf || W > seuil_sup) {
      cat("→ H  rejetée : différence significative entre X et Y.\n")
    } else {
      cat("→ H  non rejetée : pas de différence significative.\n")
    }
  }

} else {
  # Test approximatif (normale)
  cat("Méthode : approximation normale\n")
  EW <- n * (n + m + 1) / 2
  VW <- n * m * (n + m + 1) / 12
  Z <- (W - EW) / sqrt(VW)
  cat("Statistique normalisée Z =", round(Z, 3), "\n")

  if (alt == "droite") {
    z_crit <- qnorm(0.95)
    cat("Seuil critique z (droite, 5%) :", round(z_crit, 3), "\n")
    if (Z > z_crit) {
      cat("→ H  rejetée : X domine Y.\n")
    } else {
      cat("→ H  non rejetée \n")
    }

  } else if (alt == "gauche") {
    z_crit <- qnorm(0.05)
    cat("Seuil critique z (gauche, 5%) :", round(z_crit, 3), "\n")
    if (Z < z_crit) {
      cat("→ H  rejetée : Y domine X.\n")
    } else {
      cat("→ H  non rejetée \n")
    }

  } else if (alt == "bilateral") {
    z_crit <- qnorm(0.975)
    cat("Seuils critiques z (bilatéral 5%) : ±", round(z_crit, 3), "\n")
    if (abs(Z) > z_crit) {
      cat("→ H  rejetée : différence significative entre X et Y.\n")
    } else {
      cat("→ H  non rejetée : pas de différence significative.\n")
```

```
      }
    }
  }
}
```

- **Application**

```
test_wilcoxon (x,y, alternative = "droite")
```

```
## Statistique de Wilcoxon W = 44.5
## Méthode : test exact basé sur la table de Wilcoxon
## Seuil critique (droite, 5%) : 58
## → H  non rejetée
```

```
test_wilcoxon (x,y, alternative = "gauche")
```

```
## Statistique de Wilcoxon W = 44.5
## Méthode : test exact basé sur la table de Wilcoxon
## Seuil critique (gauche, 5%) : 32
## → H  non rejetée
```

```
test_wilcoxon (x,y, alternative = "bilateral")
```

```
## Statistique de Wilcoxon W = 44.5
## Méthode : test exact basé sur la table de Wilcoxon
## Seuils critiques (bilatéral 5%) : < 32  ou > 58
## → H  non rejetée : pas de différence significative.
```

3.2.3 Test de Mann Whitney Wilcoxon

- **Fonction de rang :** Le test de Mann-Whitney Wilcoxon est une méthode non paramétrique utilisée pour déterminer si deux échantillons indépendants proviennent de la même population ou si l'une des deux populations tend à avoir des valeurs plus grandes ou plus petites que l'autre.
- **Hypothèses :**
- $H_0 : F_X = F_Y$
- $H_1 : F_X > F_Y$
- **Statisiques de test :** La statistique de test est $W_n = \sum_{i=1}^{n} R_i - \frac{n(n+1)}{2}$, ou R_i désigne le rang de X_i dans l'échantillon fusionné.

On a :

$$E(W_n) = n \times \frac{n+m+1}{2} - \frac{n(n+1)}{2}$$

$$V(W_n) = nm \times \frac{n+m+1}{12}$$

Avec n, m les tailles respectives des 2 échantillons.

- **Loi de la statistique de test**

Nous pouvons utiliser la table de Wilcoxon ou utiliser l'approximation par la loi normale.

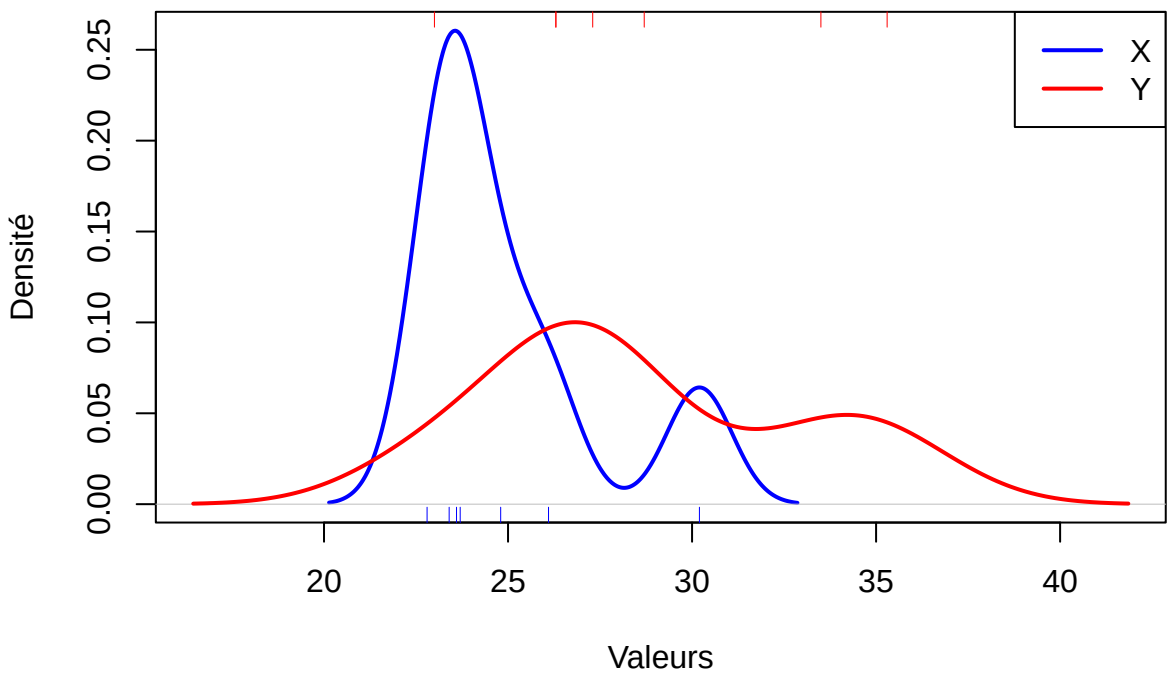
$$\frac{W_n - E(W_n)}{V(W_n)} \to \mathcal{N}(0,1)$$

- **Mise en pratique**

```r
X <- c(22.8, 23.4, 23.6, 23.7, 24.8, 26.1, 30.2)
Y <- c(23, 26.3, 26.3, 27.3, 28.7, 33.5, 35.3)
X_Y <- c(X,Y)
n = length(X)
m = length(Y)
```

- **Visualisation**

```r
# Estimation de la densité
dens_X <- density(X)
dens_Y <- density(Y)

# Détermination des limites communes pour les axes
x_lim <- range(c(dens_X$x, dens_Y$x))
y_lim <- range(c(dens_X$y, dens_Y$y))

# Plot des densités
plot(dens_X, col = "blue", lwd = 2, main = "Densités de X et Y",
     xlab = "Valeurs", ylab = "Densité", xlim = x_lim, ylim = y_lim)
lines(dens_Y, col = "red", lwd = 2)

# Légende
legend("topright", legend = c("X", "Y"), col = c("blue", "red"), lwd = 2)

# Optionnel: Ajouter des rug plots
rug(X, col = "blue", side = 1)
rug(Y, col = "red", side = 3)
```




```
 # -Calcul le rang des éléments de l'échantillon global

 vecteur_rang<-rank(X_Y, ties.method = "average")
 vecteur_rang
```

```
##  [1]  1.0  3.0  4.0  5.0  6.0  7.0 12.0  2.0  8.5  8.5 10.0 11.0 13.0 14.0
```

```
 # -Calcul de la statistique observée

 W_obs = -0.5*length(X)*(length(X)+1)
 for(i in 1: length(X)){
 W_obs=W_obs+vecteur_rang[i]
 }
 W_obs
```

```
## [1] 10
```

```
 # -Décision :

 alpha = 0.05
quantile_droite <- qwilcox(1 - alpha,n,m,log.p= FALSE)

if(W_obs >= quantile_droite){
"On rejette l'hypothese nulle  "
```

```
}else{
"On ne rejette pas l'hypothèse nulle"
}
```

```
## [1] "On ne rejette pas l'hypothèse nulle"
```

Test unilatéral à gauche

```
alpha = 0.05
quantile_gauche <- qwilcox(alpha,n,m,lower.tail =TRUE, log.p= FALSE)
# P(W q) 0.05m c'est d'ailleurs par defaut.

if(W_obs <= quantile_gauche){
"On rejette l'hypothese nulle  "
}else{
"On ne rejette pas l'hypothèse nulle"
}
```

```
## [1] "On rejette l'hypothese nulle  "
```

Test bilatéral

```
alpha = 0.05
q_low <- qwilcox(alpha / 2, n,m)
q_high <- qwilcox(1 - alpha / 2, n,m)

if(W_obs <= q_low || W_obs >= q_high){
"On rejette l'hypothese nulle  "
}else{
"On ne rejette pas l'hypothèse nulle"
}
```

```
## [1] "On ne rejette pas l'hypothèse nulle"
```

Fonction recapitulative

```
wilcox_decision <- function(W_obs, n, m, alpha = 0.05,
                            alternative = c("two.sided", "less", "greater")) {
  alternative <- match.arg(alternative)

if (alternative == "less") {
  # Test unilatéral à gauche
  q_left <- qwilcox(alpha,  n, m)

  if (W_obs <= q_left) {
    return(" Rejet de H0 : le groupe 1 tend à avoir des valeurs plus PETITES que le groupe 2"
```

```
  } else {
    return("  On ne rejette pas H0")
  }
}

   # Test unilatéral à droite
  else if (alternative == "greater") {
    q_right <- qwilcox(1 - alpha,  n, m)
    if (W_obs >= q_right) {
      return("  Rejet de H0 : le groupe 1 tend à avoir des valeurs plus GRANDES que le groupe
    } else {
      return("  On ne rejette pas H0")
    }
    }

  # Test bilatéral
  else {
    q_low <- qwilcox(alpha / 2,  n, m)
    q_high <- qwilcox(1 - alpha / 2,  n, m)

    if (W_obs <= q_low || W_obs >= q_high) {
      return("  Rejet de H0 : les distributions des deux groupes sont significativement diffé
    } else {
      return("  On ne rejette pas H0")
    }
  }
}
```

Application

```
wilcox_decision(W_obs,  n, m, alpha = 0.05, alternative = "two.sided")
```

```
## [1] "  On ne rejette pas H0"
```

```
wilcox_decision(W_obs,  n, m, alternative = "less")
```

```
## [1] " Rejet de H0 : le groupe 1 tend à avoir des valeurs plus PETITES que le groupe 2"
```

```
wilcox_decision(W_obs,  n, m, alternative = "greater")
```

```
## [1] "  On ne rejette pas H0"
```

Test avec la loi normale

```r
## Approximation avec la loi normale
E <- n*(n+m+1)/2 - n*(n+1)/2  # Correction de la formule d'espérance
Var <- (n*m*(n+m+1))/12        # Correction de la formule de variance

Z <- (W_obs - E)/sqrt(Var)     # Statistique standardisée

# Calcul des p-values selon le type de test
p_value_bilateral <- 2*pnorm(-abs(Z))
p_value_gauche <- pnorm(Z)
p_value_droite <- 1 - pnorm(Z)

# Fonction de décision normalisée
wilcox_normal_decision <- function(Z, alpha = 0.05, alternative = c("two.sided", "less", "gre
  alternative <- match.arg(alternative)

  if (alternative == "less") {
    p_value <- pnorm(Z)
    if (p_value < alpha) {
      return(list(
        decision = "  Rejet de H0 (approximation normale) : X tend à avoir des valeurs plus p
        p_value = p_value,
        Z = Z
      ))
    } else {
      return(list(
        decision = "  On ne rejette pas H0 (approximation normale)",
        p_value = p_value,
        Z = Z
      ))
    }
  }
  else if (alternative == "greater") {
    p_value <- 1 - pnorm(Z)
    if (p_value < alpha) {
      return(list(
        decision = "  Rejet de H0 (approximation normale) : X tend à avoir des valeurs plus g
        p_value = p_value,
        Z = Z
      ))
    } else {
      return(list(
        decision = "  On ne rejette pas H0 (approximation normale)",
        p_value = p_value,
        Z = Z
      ))
    }
  }
  else {
```

```
    p_value <- 2*pnorm(-abs(Z))
    if (p_value < alpha) {
      return(list(
        decision = "  Rejet de H0 (approximation normale) : Distributions significativement d
        p_value = p_value,
        Z = Z
      ))
    } else {
      return(list(
        decision = "  On ne rejette pas H0 (approximation normale)",
        p_value = p_value,
        Z = Z
      ))
    }
  }
}

## Application
result_bilateral <- wilcox_normal_decision(Z, alternative = "two.sided")
result_gauche <- wilcox_normal_decision(Z, alternative = "less")
result_droite <- wilcox_normal_decision(Z, alternative = "greater")

# Affichage des résultats
cat("Test bilatéral:\n")
```

```
## Test bilatéral:
```

```
cat("Z =", round(result_bilateral$Z, 3), "| p-value =", result_bilateral$p_value, "\n")
```

```
## Z = -1.853 | p-value = 0.06391934
```

```
cat(result_bilateral$decision, "\n\n")
```

```
##   On ne rejette pas H0 (approximation normale)
```

```
cat("Test unilatéral à gauche:\n")
```

```
## Test unilatéral à gauche:
```

```
cat("Z =", round(result_gauche$Z, 3), "| p-value =", result_gauche$p_value, "\n")
```

```
## Z = -1.853 | p-value = 0.03195967
```

```r
cat(result_gauche$decision, "\n\n")
```

```
##   Rejet de H0 (approximation normale) : X tend à avoir des valeurs plus petites que Y
```

```r
cat("Test unilatéral à droite:\n")
```

```
## Test unilatéral à droite:
```

```r
cat("Z =", round(result_droite$Z, 3), "| p-value =", result_droite$p_value, "\n")
```

```
## Z = -1.853 | p-value = 0.9680403
```

```r
cat(result_droite$decision, "\n")
```

```
##   On ne rejette pas H0 (approximation normale)
```

Avec la fonction disponible dans R

```r
wilcox.test(X,Y,alternative="two.side")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  X and Y
## W = 10, p-value = 0.07332
## alternative hypothesis: true location shift is not equal to 0
```

3.3 Tests d'alternative d'échelle

3.3.1 Test de Savage

Présentation du Test de Savage :

Soient X une variable continue et Y une variable catégorielle de modalités : $y_1, y_2, \dots, y_K$. Le test de savage permet de tester si les sous échantillons de X en fonction de Y : $X|_{Y=y_1}, \dots, X|_{Y=y_K}$ ont la même fonction de répartition.

Pour le test, on construit une fonction de score répartissant la distribution des rangs de X autour de leur position centrale moyenne.
la formule de la fonction de score est la suivante :

$$f(r_i) = \sum_{j=r_i}^{n} \frac{1}{j} = \sum_{j=1}^{n-r_i+1} \frac{1}{j} - 1$$

où r_i est le rang de l'observation i et n la taille totale de l'échantillon.

Statistique de Test :

La statistique de test a alors pour formule :

$$S = \frac{\sum_{k=1}^{K} \frac{(T_k - n_k \bar{f})^2}{n_k}}{\frac{\sum_{i=1}^{n} (f_i - \bar{f})^2}{n-1}}$$

Avec :

- T_k = somme des scores de Savage pour le groupe k
- n_k = effectif du groupe k
- $\bar{f}$ = moyenne des scores de Savage
- K = nombre de groupes

3.3.1.1 Hypothèses du Test La statistique suit une loi du χ^2 à $(K-1)$ degrés de liberté et l'hypothèse H_0 est :

$$H_0 : F_1(x) = F_2(x) = ... = F_K(x)$$

Avec la valeur seuil $\chi^2_{1-\alpha,K-1}$ de la distribution de la statistique de test du χ^2 pour une confiance $(1-\alpha)$ et pour $(K-1)$ degrés de liberté, l'hypothèse alternative est alors :

$H_1 : \exists i, j \in \{1, 2, ..., K\} : F_i(x) \neq F_j(x)$

- tel que $S > \chi^2_{1-\alpha,k-1}$, soit rejet de H_0, pour un test bilatéral
- Le test du χ^2 ne propose pas de forme unilatérale à droite ou à gauche pour S

La loi à laquelle reporter la statistique de test de Savage est celle du χ^2 à $(k-1)$ degrés de liberté. La formule de la p-valeur exacte est alors :

$$p = P(\chi^2_{K-1} > S) = 1 - F_{\chi^2_{K-1}}(S)$$

Tendance pour le rejet de l'hypothèse nulle

Plus la statistique de test de Savage est grande et plus on a de chance de rejeter H_0, ce qui revient à dire que :

- **On rejette** H_0 si $S > \chi^2_{1-\alpha,K-1}$
- Soit que la somme des rangs pour l'un des groupes est nettement plus grande que la moyenne des rangs pondérés, impliquant que l'une des distributions est nettement différente des autres.
- **Tendance lorsque n tend vers l'infini**

Le test de Savage est influencé par la taille de l'échantillon. Pour de très grands échantillons, le test peut rejeter H_0 à tort même lorsque les distributions sont similaires. Cette sensibilité est due à la formule de la statistique de test qui fait intervenir les effectifs des différents groupes.

3.3.1.2 Implémentation

- **Importation des données**

```
donnees <- read_dta("BASES/savage_dataset.dta")
donnees$milieu <- as_factor(donnees$milieu)
```

- **Graphique des courbes de densité par classe**

```
plot_densites <- function(data, var_continue, var_groupe,
                          titre = "Distribution par groupe") {

  # Création du graphique de densité
  p <- ggplot(data, aes_string(x = var_continue, color = var_groupe)) +
    geom_density(size = 1.2, alpha = 0.8) +
    theme_minimal() +
    labs(
      title = titre,
      x = var_continue,
      y = "Densité",
      color = var_groupe
    ) +
    theme(
      legend.position = "bottom",
      plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
      axis.title = element_text(size = 12),
      legend.title = element_text(size = 12)
    )

  return(p)
}
```

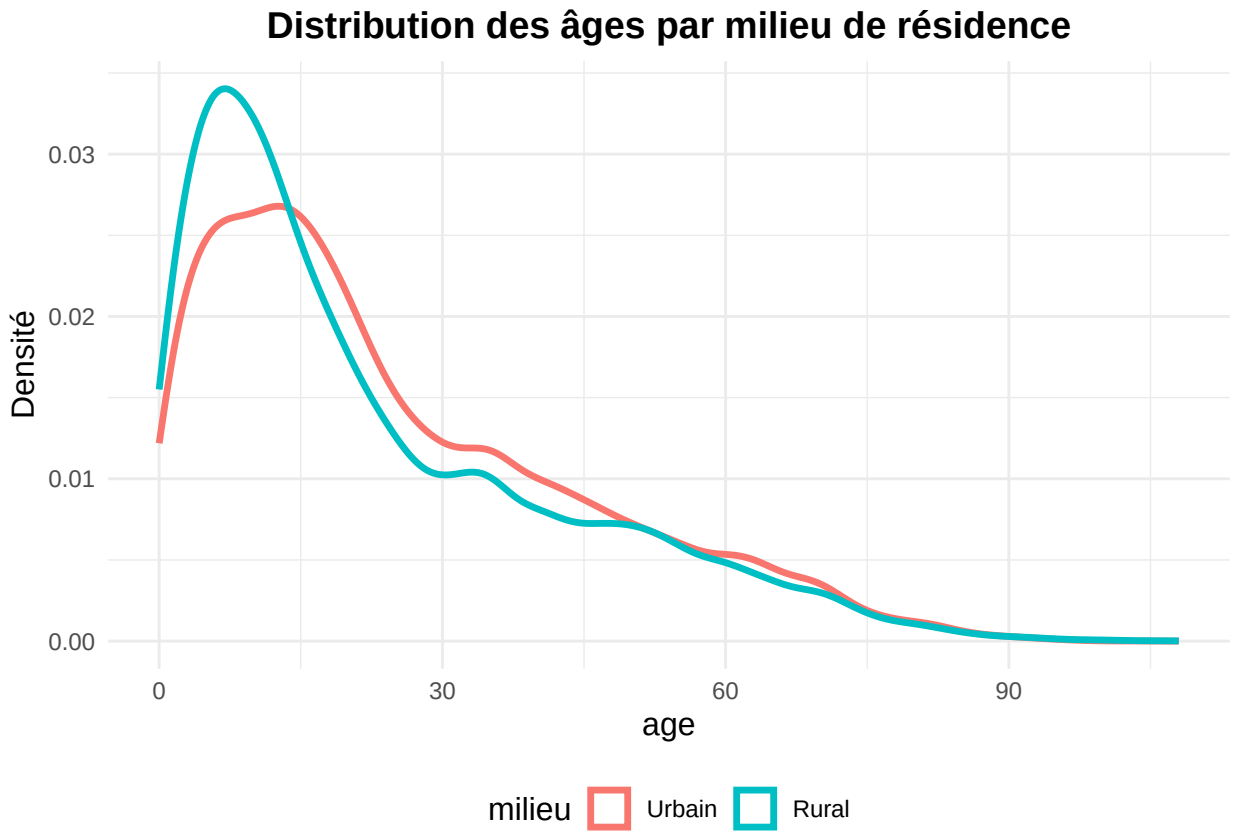
- **Boxplot par classe**

```
plot_boxplots <- function(data, var_continue, var_groupe,
                          titre = "Boxplot par groupe") {

  # Création du boxplot
  p <- ggplot(data, aes_string(x = var_groupe, y = var_continue, fill = var_groupe)) +
    geom_boxplot(alpha = 0.7, outlier.color = "red", outlier.size = 2) +
    theme_minimal() +
    labs(
      title = titre,
      x = var_groupe,
      y = var_continue,
      fill = var_groupe
    ) +
```

```
    theme(
      legend.position = "none",
      plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
      axis.title = element_text(size = 12),
      axis.text.x = element_text(angle = 45, hjust = 1)
    )

  return(p)
}
```

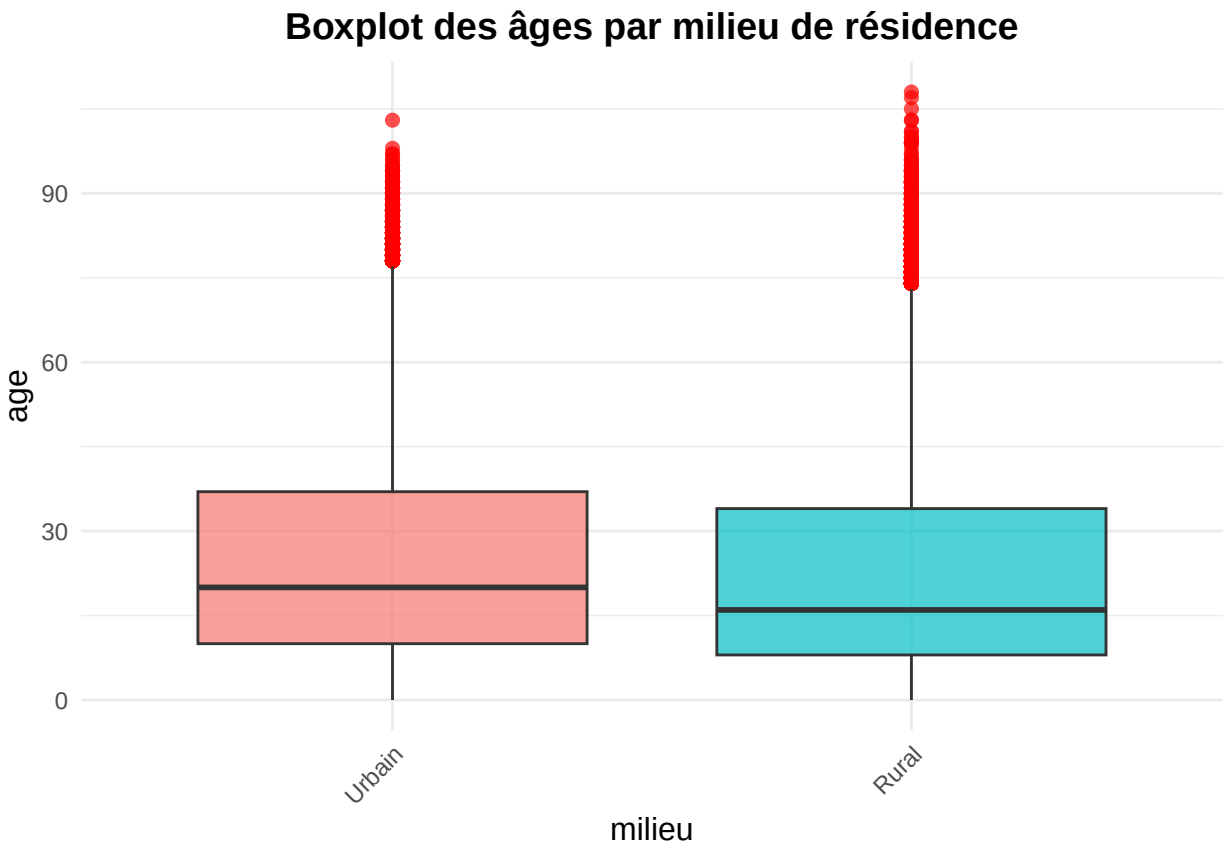
- **Affichage des courbes de densité**

```
plot_densites(donnees, "age", "milieu", "Distribution des âges par milieu de résidence")
```



- **Affichage des boxplots**

```
plot_boxplots(donnees, "age", "milieu", "Boxplot des âges par milieu de résidence")
```



- **Fonction de test de Savage**

```r
SavageTest <- function(X, Y) {

  # Vérifications préliminaires
  if(length(X) != length(Y)) {
    stop("Les vecteurs X et Y doivent avoir la même longueur")
  }

  if(any(is.na(X)) || any(is.na(Y))) {
    stop("Les données ne doivent pas contenir de valeurs manquantes")
  }

  # Transformation de X en variable de rang
  R <- rank(X)

  # Récupération des différents éléments
  n <- length(R)
  biblio <- summary(factor(Y)) #effectif de chaque groupe
  nbClass <- length(biblio)

  # Vérification du nombre de groupes
  if(nbClass < 2) {
    stop("Il faut au moins 2 groupes pour effectuer le test")
  }
```

```r
  print(paste("Nombre d'observations:", n))
  print(paste("Nombre de groupes:", nbClass))
  print("Effectifs par groupe:")
  print(biblio)

  # Initialisation des vecteurs f
  f <- 0 # score de savage pour chaque observation
  ff <- 0 # score de savage par groupe

  # Calcul de f(X restreint aux différents groupes de Y)
  for (c in 1:nbClass) {
    # Focus sur la classe en cours
    Rcalc <- R[which(Y == names(biblio)[c])]
    nb <- length(Rcalc)
    f_Rcalc <- 0
    aa <- 0

    # Pour chaque observation de la classe en cours
    for (i in 1:nb) {
      # Calcul du score de Savage : somme de 1/(n-k+1) pour k allant de rang à n
      a <- sum(1/(n - 1:Rcalc[i] + 1)) - 1
      f_Rcalc <- c(f_Rcalc, a)
      aa <- c(aa, a)
    }

    # Remplissage des vecteurs
    f <- c(f, f_Rcalc[-1])
    ff <- c(ff, sum(aa[-1]))
  }

  # Calcul du dénominateur et du numérateur
  f_barre <- mean(f[-1])
  Denom <- sum((f[-1] - f_barre)^2)/(n - 1)
  Num <- sum((ff[-1] - biblio*f_barre)^2/biblio)

  # Calcul de la statistique de test
  S <- Num/Denom
  df <- nbClass - 1
  p_value <- 1 - pchisq(S, df)

  # Préparation des résultats
  Result <- c(biblio, N = n, Statistique_test = S, df = df, p_value = p_value)

  return(Result)
}
```

- **Fonction d'interprétation complète**

```r
interpretation_savage <- function(X, Y, alpha = 0.05, nom_var_X = "X", nom_var_Y = "Y") {
  #' Fonction complète d'interprétation du test de Savage
  #'
  #' @param X Variable continue
  #' @param Y Variable qualitative (groupes)
  #' @param alpha Seuil de significativité (défaut: 0.05)
  #' @param nom_var_X Nom de la variable continue pour l'affichage
  #' @param nom_var_Y Nom de la variable qualitative pour l'affichage

  cat("========================================\n")
  cat("           TEST DE SAVAGE\n")
  cat("========================================\n\n")

  cat("Variables analysées:\n")
  cat(paste("- Variable continue:", nom_var_X, "\n"))
  cat(paste("- Variable qualitative:", nom_var_Y, "\n\n"))

  cat("Hypothèses:\n")
  cat("H0: Les distributions sont identiques entre les groupes\n")
  cat("H1: Au moins une distribution diffère des autres\n\n")

  # Application du test
  resultats <- SavageTest(X, Y)

  # Extraction des paramètres
  S <- as.numeric(resultats["Statistique_test"])
  df <- as.numeric(resultats["df"])
  p_val <- as.numeric(resultats["p_value"])
  n_total <- as.numeric(resultats["N"])

  # Valeur critique
  valeur_critique <- qchisq(1 - alpha, df)

  cat("Résultats du test:\n")
  cat(paste("- Statistique de test S =", round(S, 4), "\n"))
  cat(paste("- Degrés de liberté =", df, "\n"))
  cat(paste("- P-valeur =", round(p_val, 4), "\n"))
  cat(paste("- Valeur critique (  =", alpha, ") =", round(valeur_critique, 4), "\n\n"))

  # Décision
  cat("Décision:\n")
  if (p_val < alpha) {
    cat(paste("  Rejet de H0 (p =", round(p_val, 4), "<   =", alpha, ")\n"))
    cat("  CONCLUSION: Les distributions diffèrent significativement entre groupes.\n\n")
  } else {
    cat(paste("  Non rejet de H0 (p =", round(p_val, 4), "    =", alpha, ")\n"))
    cat("  CONCLUSION: Pas de différence significative entre les distributions.\n\n")
  }
```

```
  # Interprétation de l'intensité de l'effet
  cat("Intensité de l'effet:\n")
  if (S > 2 * valeur_critique) {
    cat("→ Effet très fort (S >> valeur critique)\n")
  } else if (S > 1.5 * valeur_critique) {
    cat("→ Effet fort (S > 1.5 × valeur critique)\n")
  } else if (S > valeur_critique) {
    cat("→ Effet modéré (S > valeur critique)\n")
  } else {
    cat("→ Effet faible ou absent (S   valeur critique)\n")
  }

  cat("\n========================================\n")

  return(invisible(resultats))
}
```

3.3.1.3 Application

- **base complète**

```
print("\n=== TEST DE SAVAGE ===")
```

```
## [1] "\n=== TEST DE SAVAGE ==="
```

```
resultats <- interpretation_savage(donnees$age, donnees$milieu,
                                   alpha = 0.05,
                                   nom_var_X = "Âge",
                                   nom_var_Y = "Milieu de résidence")
```

```
## ========================================
##            TEST DE SAVAGE
## ========================================
##
## Variables analysées:
## - Variable continue: Âge
## - Variable qualitative: Milieu de résidence
##
## Hypothèses:
## H0: Les distributions sont identiques entre les groupes
## H1: Au moins une distribution diffère des autres
##
## [1] "Nombre d'observations: 63530"
## [1] "Nombre de groupes: 2"
## [1] "Effectifs par groupe:"
## Urbain  Rural
```

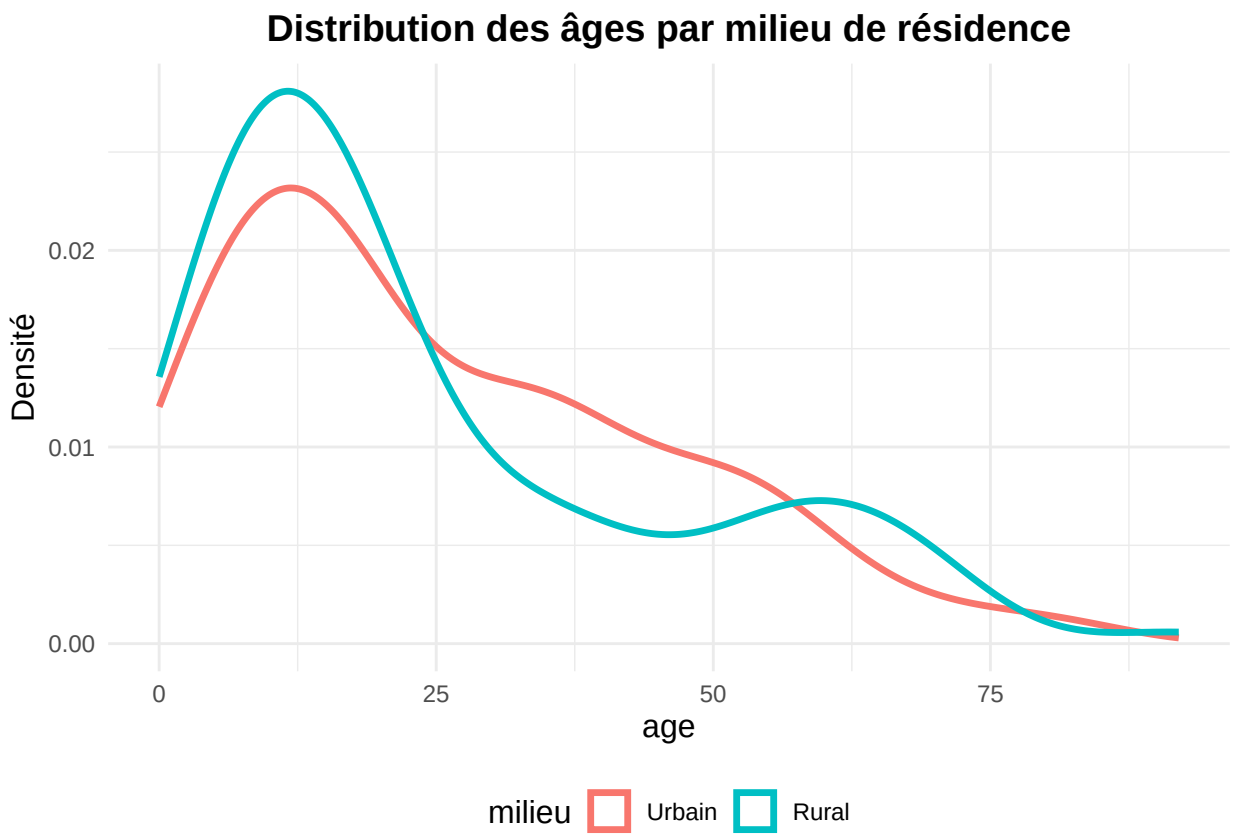
```
##  32155  31375
## Résultats du test:
## - Statistique de test S = 165.5153
## - Degrés de liberté = 1
## - P-valeur = 0
## - Valeur critique (  = 0.05 ) = 3.8415
##
## Décision:
##   Rejet de H0 (p = 0 <   = 0.05 )
##   CONCLUSION: Les distributions diffèrent significativement entre groupes.
##
## Intensité de l'effet:
## → Effet très fort (S >> valeur critique)
##
## ========================================
```

- **sous échantillon**

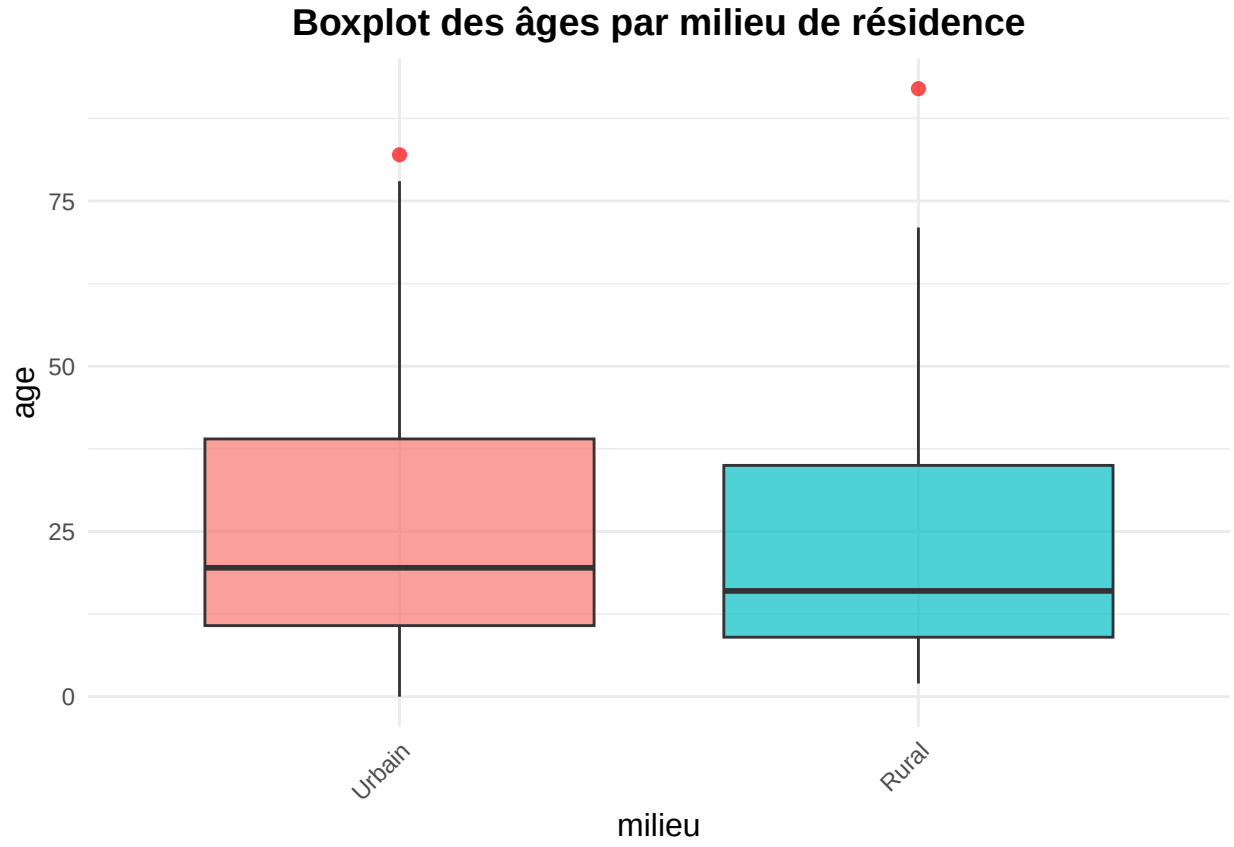
```
# Création du sous-échantillon stratifié : 1000 individus par milieu
set.seed(42)  # Pour reproductibilité

sous_echantillon <- donnees |>
  split(donnees$milieu) |>  # Divise les données selon le milieu
  lapply(function(df) df[sample(nrow(df), size = min(100, nrow(df))), ]) |>  # Tire 1000 alé
  do.call(what = rbind)  # Recombine les strates

# Vérification des effectifs par milieu
table(sous_echantillon$milieu)
```

```
##
## Urbain  Rural
##    100    100
```

```
plot_densites(sous_echantillon, "age", "milieu", "Distribution des âges par milieu de réside
```




```
plot_boxplots(sous_echantillon, "age", "milieu", "Boxplot des âges par milieu de résidence")
```



```r
print("\n=== TEST DE SAVAGE ===")
```

```
## [1] "\n=== TEST DE SAVAGE ==="
```

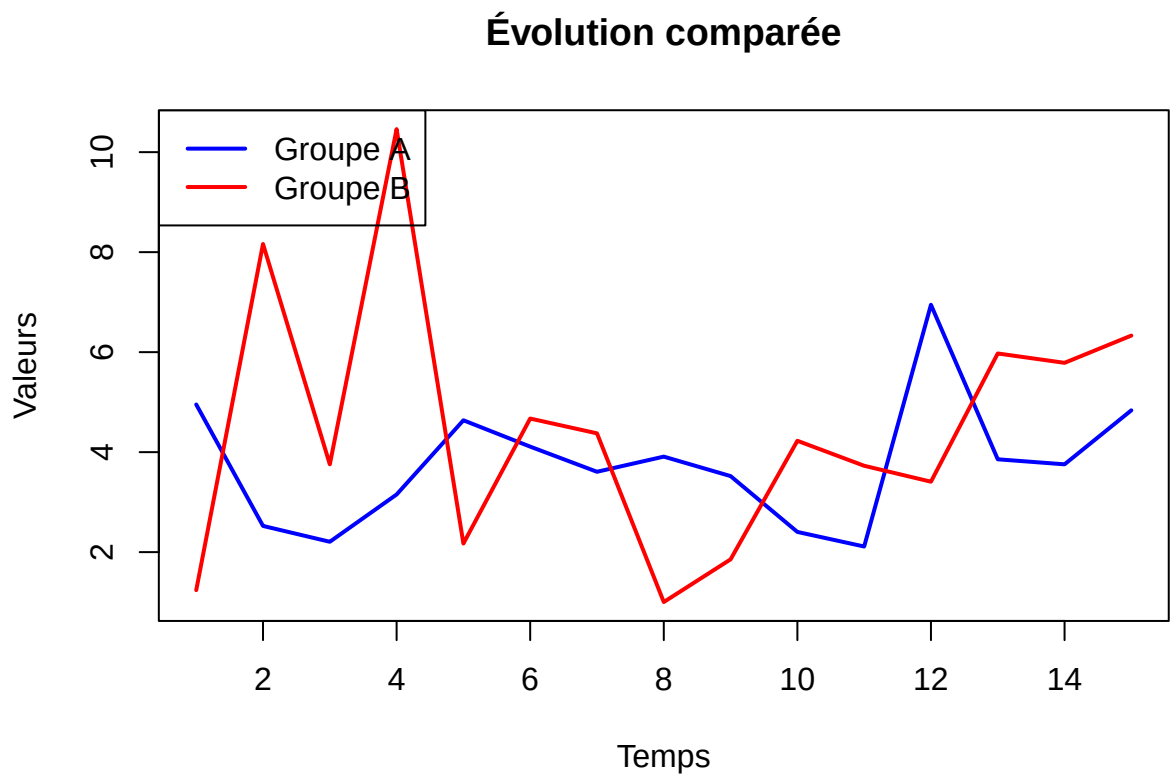
```r
resultats <- interpretation_savage(sous_echantillon$age, sous_echantillon$milieu,
                                   alpha = 0.05,
                                   nom_var_X = "Âge",
                                   nom_var_Y = "Milieu de résidence")
```

```
## ========================================
##            TEST DE SAVAGE
## ========================================
##
## Variables analysées:
## - Variable continue: Âge
## - Variable qualitative: Milieu de résidence
##
## Hypothèses:
## H0: Les distributions sont identiques entre les groupes
## H1: Au moins une distribution diffère des autres
##
## [1] "Nombre d'observations: 200"
## [1] "Nombre de groupes: 2"
## [1] "Effectifs par groupe:"
## Urbain  Rural
##    100    100
## Résultats du test:
## - Statistique de test S = 0.0093
## - Degrés de liberté = 1
## - P-valeur = 0.9231
## - Valeur critique (  = 0.05 ) = 3.8415
##
## Décision:
##   Non rejet de H0 (p = 0.9231    = 0.05 )
##   CONCLUSION: Pas de différence significative entre les distributions.
##
## Intensité de l'effet:
## → Effet faible ou absent (S   valeur critique)
##
## ========================================
```

3.3.2 Test de Mood

```r
# Données d'exemple
x3 <- 1:15  # Les périodes (il y a 15 valeurs dans chaque vecteur)
```

```r
x1 <- c(4.953, 2.524, 2.207, 3.153, 4.637, 4.11, 3.607, 3.91, 3.521,
        2.404, 2.112, 6.947, 3.857, 3.756, 4.836)  # Groupe A

y1 <- c(1.24, 8.163, 3.756, 10.46, 2.172, 4.672, 4.376, 1.002, 1.855,
        4.227, 3.727, 3.409, 5.973, 5.786, 6.331)  # Groupe B

# Tracer la première courbe (x1 en bleu)
plot(x3, x1, type = "l", col = "blue", lwd = 2, ylim = range(c(x1, y1)),
     xlab = "Temps", ylab = "Valeurs", main = "Évolution comparée")

# Ajouter la deuxième courbe (y1 en rouge)
lines(x3, y1, col = "red", lwd = 2)

# Ajouter une légende
legend("topleft", legend = c("Groupe A", "Groupe B"),
       col = c("blue", "red"), lty = 1, lwd = 2)
```



Création de la base de donneés

```r
x <- c(4.953, 2.524, 2.207, 3.153, 4.637, 4.11, 3.607, 3.91, 3.521,  2.404,
       2.112, 6.947, 3.857, 3.756, 4.836)  # Groupe A
y <- c(1.24, 8.163, 3,756, 10,46, 2.172, 4.672, 4.376, 1.002, 1.855, 4.227,
       3.727, 3.409, 5.973, 5.786, 6.331) # Groupe B
```

Appliquer le test de Mood

```r
mood.test(x, y)
```

```
##
##  Mood two-sample test of scale
##
## data:  x and y
## Z = -2.3767, p-value = 0.01747
## alternative hypothesis: two.sided
```

Notre p_value est inferieur a 0,05 alors on rejet Ho c'est-à-dire que les dispersions ne sont pas les mêmes. Autrement dit, les dispersions sont differentes.

A présent, nous allons refaire le test manuellement et comparer aux résultats précédents, avant d'élaborer une fonction.

```r
n <- length(x) # taille de X
m <- length(y) # taille de Y
N <- n + m     # taille totale
```

Rang, score et tableau

On combine les deux groupes et on calcule les rangs et les scores

- **Score de chaque observation**

$$\text{score}_i = \left(\text{rang}_i - \frac{N+1}{2}\right)^2$$

```r
# Combinaison
z <- c(x, y)
groupe <- c(rep("X", n), rep("Y", m))
rang <- rank(z)
score <- (rang - (N + 1) / 2)^2

# Tableau
table_mood <- data.frame(Observation = z, Groupe = groupe, Rang = rang, Score = score)
knitr::kable(table_mood, digits = 3)
```

Observation	Groupe	Rang	Score
4.953	X	24	56.25
2.524	X	8	72.25
2.207	X	6	110.25
3.153	X	10	42.25
4.637	X	21	20.25
4.110	X	18	2.25
3.607	X	13	12.25

Observation	Groupe	Rang	Score
3.910	X	17	0.25
3.521	X	12	20.25
2.404	X	7	90.25
2.112	X	4	156.25
6.947	X	28	132.25
3.857	X	16	0.25
3.756	X	15	2.25
4.836	X	23	42.25
1.240	Y	2	210.25
8.163	Y	29	156.25
3.000	Y	9	56.25
756.000	Y	32	240.25
10.000	Y	30	182.25
46.000	Y	31	210.25
2.172	Y	5	132.25
4.672	Y	22	30.25
4.376	Y	20	12.25
1.002	Y	1	240.25
1.855	Y	3	182.25
4.227	Y	19	6.25
3.727	Y	14	6.25
3.409	Y	11	30.25
5.973	Y	26	90.25
5.786	Y	25	72.25
6.331	Y	27	110.25

- **On calcule la somme des scores pour le groupe X (appelée M_n)**

$$M_n = \sum_{i=1}^{n} \left(\text{rang}_i - \frac{N+1}{2} \right)^2$$

```
Mn <- sum(score[groupe == "X"])
Mn
```

```
## [1] 759.75
```

Ensuite, on calcule l'espérance et la variance.

- *Espérance de M_n*

$$\mathbb{E}[M_n] = \frac{n(N^2-1)}{12}$$

- *Variance de M_n*

$$\text{Var}(M_n) = \frac{nm(N+1)(N^2-1)}{180}$$

```
esp <- n * (N^2 - 1) / 12
var <- n * m * (N + 1) * (N^2 - 1) / 180
esp
```

```
## [1] 1278.75
```

```
var
```

```
## [1] 47825.25
```

- **Normalisation :**

Statistique normalisée : $Z = \frac{M_n - \mathbb{E}[M_n]}{\sqrt{\text{Var}(M_n)}}$

```
Z <- (Mn - esp) / sqrt(var)
Z
```

```
## [1] -2.373224
```

- **Règle de décision :**

Si $|Z| > z_{1-\alpha/2}$, alors on rejette H_0

```
# Seuil critique
  z_crit <- qnorm(1 - 0.05 / 2)
z_crit
```

```
## [1] 1.959964
```

```
# calcul du valeur absolue de Z
valeur_Z<- abs(Z)
valeur_Z
```

```
## [1] 2.373224
```

On voit que la valeur absolue de Z est superieur a 1,96 alors on rejette Ho c'est a dire les dispersions ne sont pas les mêmes

Fonctions

```r
mood_test_manuel <- function(x, y, alpha) {
  # Étapes préliminaires
  n <- length(x)
  m <- length(y)
  N <- n + m
  z <- c(x, y)
  groupe <- c(rep("X", n), rep("Y", m))

  # Rangs et scores
  rang <- rank(z)
  score <- (rang - (N + 1)/2)^2

  # Mn : somme des scores pour le groupe X
  Mn <- sum(score[groupe == "X"])

  # Espérance et variance de Mn
  esperance <- n * (N^2 - 1) / 12
  variance <- n * m * (N + 1) * (N^2 - 1) / 180

  # Statistique Z
  Z <- (Mn - esperance) / sqrt(variance)
  abs_Z <- abs(Z)

  # Seuil critique
  z_crit <- qnorm(1 - alpha / 2)

  # Décision
  decision <- ifelse(abs_Z > z_crit,
                     "Rejet de H0 : dispersions différentes",
                     "On ne rejette pas H0 : dispersions identiques")

  # Retourner les résultats sous forme de liste
  resultats <- list(
    Mn = Mn,
    Esperance = esperance,
    Variance = variance,
    Z = Z,
    abs_Z = abs_Z,
    z_critique = z_crit,
    alpha = alpha,
    Decision = decision
  )

  return(resultats)
}
```

Renseignons à présent les valeurs.

```r
# Tes données
x <- c(4.953, 2.524, 2.207, 3.153, 4.637, 4.11, 3.607, 3.91, 3.521,
       2.404, 2.112, 6.947, 3.857, 3.756, 4.836)

y <- c(1.24, 8.163, 3.756, 10.46, 2.172, 4.672, 4.376, 1.002, 1.855,
       4.227, 3.727, 3.409, 5.973, 5.786, 6.331)

# Appel avec alpha = 0.01 (ou tout autre seuil)
resultat <- mood_test_manuel(x, y, alpha = 0.05)

# Affichage
print(resultat)
```

```
## $Mn
## [1] 750.5
##
## $Esperance
## [1] 1123.75
##
## $Variance
## [1] 34836.25
##
## $Z
## [1] -1.999789
##
## $abs_Z
## [1] 1.999789
##
## $z_critique
## [1] 1.959964
##
## $alpha
## [1] 0.05
##
## $Decision
## [1] "Rejet de H0 : dispersions différentes"
```

3.4 Tests d'alternative générale

3.4.1 Test de Kolmogorov

```r
x <- c(0.9, 1.8, 2.2, 2.6, 2.7)
y <- c(0.6, 1.1, 1.4, 2.3, 3.0, 3.1)
```

3.4.1.1 Exemple

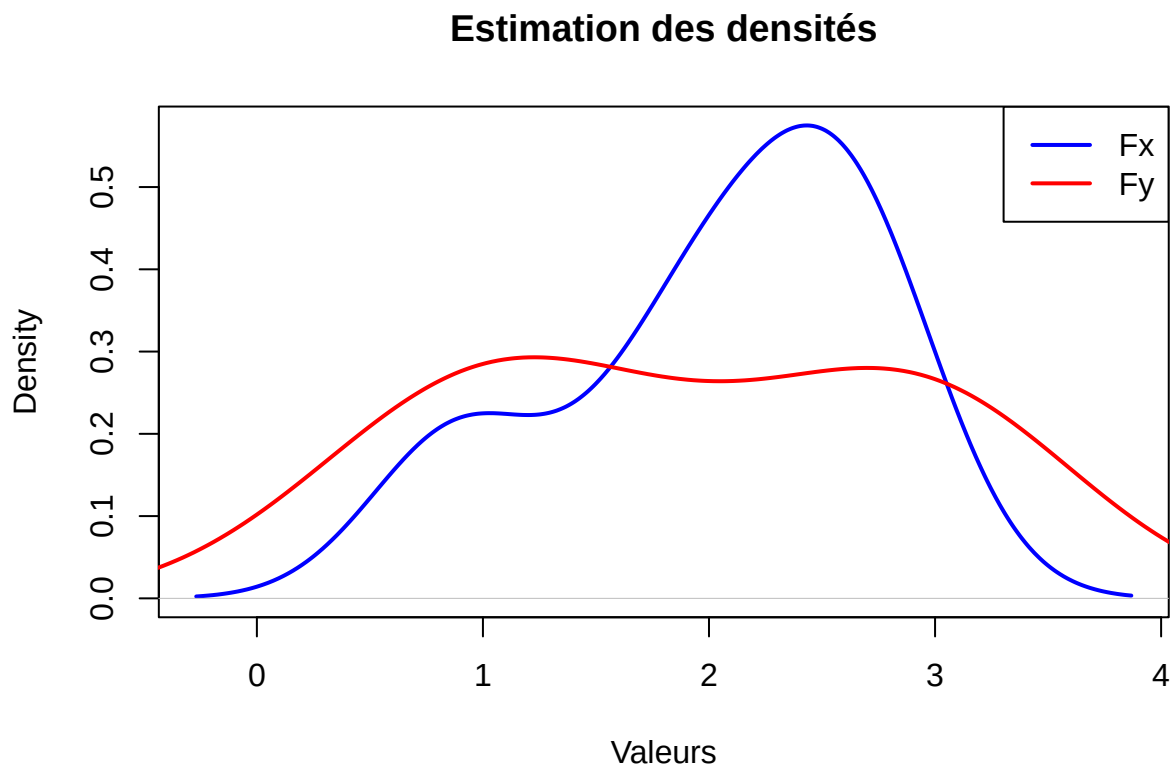
- **Fusion des vecteurs x et y**

```
x_y <- c(x,y)
x_yr <- sort(x_y)
```

- **Visualisation des données**

  - **Densité**

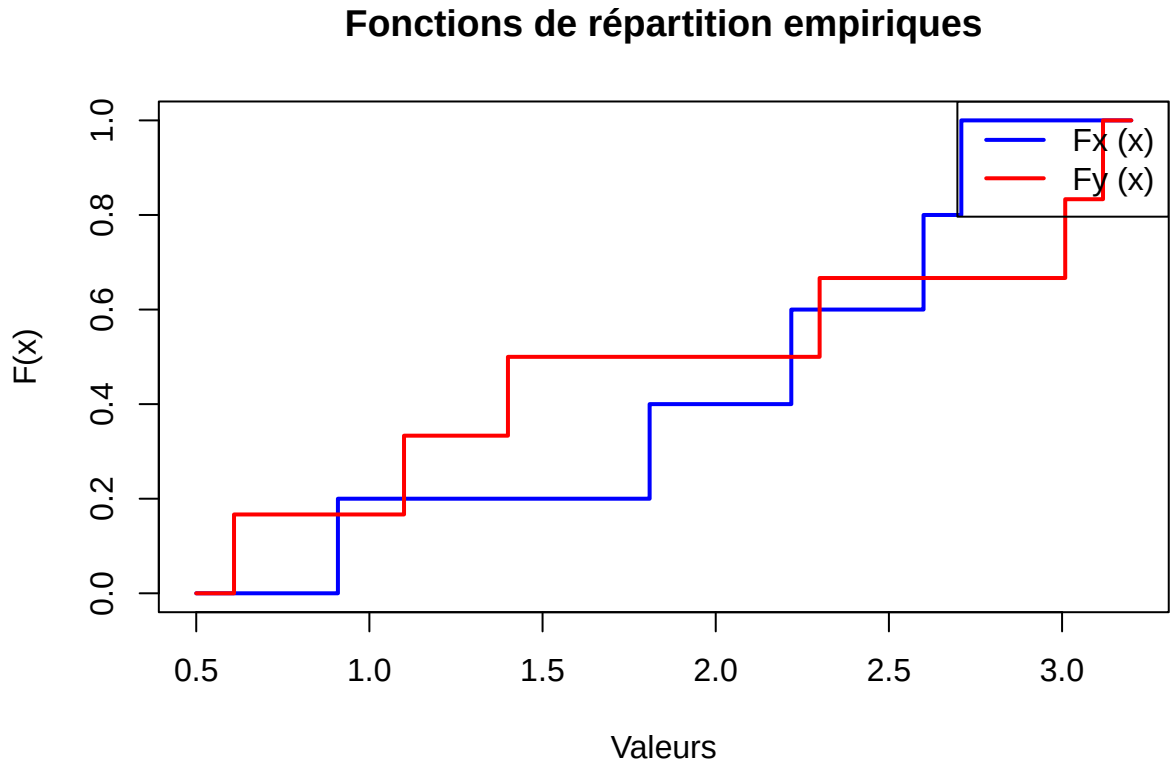
```
dx <- density(x)
dy <- density(y)

plot(dx, col = "blue", lwd = 2, main = "Estimation des densités", xlab = "Valeurs", ylim = ra
lines(dy, col = "red", lwd = 2)

legend("topright", legend = c("Fx", "Fy"), col = c("blue", "red"), lwd = 2)
```



- **Fonction de répartition empirique**

```
Fx <- ecdf(x)
Fy <- ecdf(y)

range_x <- seq(min(x_yr) - 0.1, max(x_yr) + 0.1, length.out = 100)
```

```r
plot(range_x, Fx(range_x), type = "s", col = "blue", lwd = 2,
     xlab = "Valeurs", ylab = "F(x)", main = "Fonctions de répartition empiriques")
lines(range_x, Fy(range_x), type = "s", col = "red", lwd = 2)

legend("topright", legend = c("Fx (x)", "Fy (x)"), col = c("blue", "red"), lwd = 2)
```



- **Tableau format long**

```r
N <- length(x_yr)
x_long <- c(x, rep(NA, N - length(x)))
y_long <- c(y, rep(NA, N - length(y)))
df <- data.frame(x = x_long, y = y_long)

df$x_y <- x_yr

# Serie source de chaque valeur
df$xn <- ifelse(df$x_y %in% x, 1, 0)
df$yn <- ifelse(df$x_y %in% y, 1, 0)

# Effectifs cumulés
df$ECC1 <- sapply(df$x_y, function(val) {
  sum(x <= val)
})
df$ECC2 <- sapply(df$x_y, function(val) {
```

```
  sum(y <= val)
})

# Fonction de répartition
df$Fx <- df$ECC1/length(x)
df$Fy <- df$ECC2/length(y)

df["Fx-Fy"] <- df$Fx-df$Fy
df["Fy-Fx"] <- df$Fy-df$Fx

print(df)
```

```
##       x   y x_y xn yn ECC1 ECC2  Fx        Fy       Fx-Fy       Fy-Fx
## 1   0.9 0.6 0.6  0  1    0    1 0.0 0.1666667 -0.16666667  0.16666667
## 2   1.8 1.1 0.9  1  0    1    1 0.2 0.1666667  0.03333333 -0.03333333
## 3   2.2 1.4 1.1  0  1    1    2 0.2 0.3333333 -0.13333333  0.13333333
## 4   2.6 2.3 1.4  0  1    1    3 0.2 0.5000000 -0.30000000  0.30000000
## 5   2.7 3.0 1.8  1  0    2    3 0.4 0.5000000 -0.10000000  0.10000000
## 6    NA 3.1 2.2  1  0    3    3 0.6 0.5000000  0.10000000 -0.10000000
## 7    NA  NA 2.3  0  1    3    4 0.6 0.6666667 -0.06666667  0.06666667
## 8    NA  NA 2.6  1  0    4    4 0.8 0.6666667  0.13333333 -0.13333333
## 9    NA  NA 2.7  1  0    5    4 1.0 0.6666667  0.33333333 -0.33333333
## 10   NA  NA 3.0  0  1    5    5 1.0 0.8333333  0.16666667 -0.16666667
## 11   NA  NA 3.1  0  1    5    6 1.0 1.0000000  0.00000000  0.00000000
```

- **Statistique de test**

La table de Kolmogrov n'est pas tabulé sur R.

```
Kmn <- max(df["Fx-Fy"],df["Fy-Fx"])
paste("Kmn = ", Kmn)
```

```
## [1] "Kmn =  0.333333333333333"
```

```
paste("Valeur de c avec la loi normale standard", pnorm(0.975, mean = 0, sd = 1)
)
```

```
## [1] "Valeur de c avec la loi normale standard 0.83521987001969"
```

```
paste("Test de Kolmogrov sur R")
```

```
## [1] "Test de Kolmogrov sur R"
```

```r
print(ks.test(x, y))
```

```
##
##  Exact two-sample Kolmogorov-Smirnov test
##
## data:  x and y
## D = 0.33333, p-value = 0.8182
## alternative hypothesis: two-sided
```

```r
ks.stat <- function(x, y) {
  x_yr <- sort(c(x, y))
  N <- length(x_yr)

  x_long <- c(x, rep(NA, N - length(x)))
  y_long <- c(y, rep(NA, N - length(y)))

  # Créer le tableau initial
  df <- data.frame(
    x = x_long,
    y = y_long,
    x_y = x_yr
  )

  df$xn <- ifelse(df$x_y %in% x, 1, 0)
  df$yn <- ifelse(df$x_y %in% y, 1, 0)

  # Effectifs cumulés pour chaque valeur de x_y
  df$ECC1 <- sapply(df$x_y, function(val) sum(x <= val))
  df$ECC2 <- sapply(df$x_y, function(val) sum(y <= val))

  # Fonctions de répartition empiriques
  df$Fx <- df$ECC1 / length(x)
  df$Fy <- df$ECC2 / length(y)

  df["Fx-Fy"] <- df$Fx - df$Fy
  df["Fy-Fx"] <- df$Fy - df$Fx

  # Statistique de test
  Kmn <- max(df["Fx-Fy"],df["Fy-Fx"])

  return(list(
    tableau = df,
    Kmn = Kmn
  ))
}
```

3.4.1.2 Fonctions

```
df <- read_excel("BASES\\notes_kolmogorov.xlsx")

x <- df[["Anthropo/Moyenne"]]
y <- df[["Estimation/Moyenne"]]

ks.stat(x,y)
```

3.4.1.3 Application

```
## $tableau
##       x    y  x_y xn yn ECC1 ECC2         Fx        Fy      Fx-Fy     Fy-Fx
## 1  16.5  8.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 2  14.5 13.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 3  15.5  8.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 4   8.0 10.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 5  15.0 10.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 6  15.0  8.5  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 7  15.5  9.5  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 8  16.0 10.5  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 9  14.5 10.5  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 10 15.5  8.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 11 16.5 10.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 12 14.0 10.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 13 14.5 13.5  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 14 15.5  8.0  8.0  1  1    1   13 0.02380952 0.3095238 -0.2857143 0.2857143
## 15 16.0 11.5  8.5  0  1    1   15 0.02380952 0.3571429 -0.3333333 0.3333333
## 16 16.0  8.0  8.5  0  1    1   15 0.02380952 0.3571429 -0.3333333 0.3333333
## 17 16.5  8.0  9.5  0  1    1   17 0.02380952 0.4047619 -0.3809524 0.3809524
## 18 16.0 10.0  9.5  0  1    1   17 0.02380952 0.4047619 -0.3809524 0.3809524
## 19 15.5 13.5 10.0  0  1    1   25 0.02380952 0.5952381 -0.5714286 0.5714286
## 20 16.0 11.0 10.0  0  1    1   25 0.02380952 0.5952381 -0.5714286 0.5714286
## 21 16.5 12.0 10.0  0  1    1   25 0.02380952 0.5952381 -0.5714286 0.5714286
## 22 14.5  8.0 10.0  0  1    1   25 0.02380952 0.5952381 -0.5714286 0.5714286
## 23 16.5 11.0 10.0  0  1    1   25 0.02380952 0.5952381 -0.5714286 0.5714286
## 24 15.5  8.0 10.0  0  1    1   25 0.02380952 0.5952381 -0.5714286 0.5714286
## 25 15.5  8.0 10.0  0  1    1   25 0.02380952 0.5952381 -0.5714286 0.5714286
## 26 15.0 11.0 10.0  0  1    1   25 0.02380952 0.5952381 -0.5714286 0.5714286
## 27 14.5  8.0 10.5  0  1    1   29 0.02380952 0.6904762 -0.6666667 0.6666667
## 28 14.5 10.0 10.5  0  1    1   29 0.02380952 0.6904762 -0.6666667 0.6666667
## 29 16.0 10.0 10.5  0  1    1   29 0.02380952 0.6904762 -0.6666667 0.6666667
## 30 14.5  8.0 10.5  0  1    1   29 0.02380952 0.6904762 -0.6666667 0.6666667
## 31 15.0 10.5 11.0  0  1    1   32 0.02380952 0.7619048 -0.7380952 0.7380952
## 32 16.0 14.0 11.0  0  1    1   32 0.02380952 0.7619048 -0.7380952 0.7380952
## 33 16.5 10.0 11.0  0  1    1   32 0.02380952 0.7619048 -0.7380952 0.7380952
```

```
## 34 15.5  8.0 11.5  0  1    1   34 0.02380952 0.8095238 -0.7857143 0.7857143
## 35 15.5 10.5 11.5  0  1    1   34 0.02380952 0.8095238 -0.7857143 0.7857143
## 36 16.0 12.5 12.0  0  1    1   35 0.02380952 0.8333333 -0.8095238 0.8095238
## 37 15.0 12.5 12.5  0  1    1   38 0.02380952 0.9047619 -0.8809524 0.8809524
## 38 15.5  8.5 12.5  0  1    1   38 0.02380952 0.9047619 -0.8809524 0.8809524
## 39 16.5 12.5 12.5  0  1    1   38 0.02380952 0.9047619 -0.8809524 0.8809524
## 40 16.5  8.0 13.0  0  1    1   39 0.02380952 0.9285714 -0.9047619 0.9047619
## 41 15.0  9.5 13.5  0  1    1   41 0.02380952 0.9761905 -0.9523810 0.9523810
## 42 16.0 11.5 13.5  0  1    1   41 0.02380952 0.9761905 -0.9523810 0.9523810
## 43   NA   NA 14.0  1  1    2   42 0.04761905 1.0000000 -0.9523810 0.9523810
## 44   NA   NA 14.0  1  1    2   42 0.04761905 1.0000000 -0.9523810 0.9523810
## 45   NA   NA 14.5  1  0    9   42 0.21428571 1.0000000 -0.7857143 0.7857143
## 46   NA   NA 14.5  1  0    9   42 0.21428571 1.0000000 -0.7857143 0.7857143
## 47   NA   NA 14.5  1  0    9   42 0.21428571 1.0000000 -0.7857143 0.7857143
## 48   NA   NA 14.5  1  0    9   42 0.21428571 1.0000000 -0.7857143 0.7857143
## 49   NA   NA 14.5  1  0    9   42 0.21428571 1.0000000 -0.7857143 0.7857143
## 50   NA   NA 14.5  1  0    9   42 0.21428571 1.0000000 -0.7857143 0.7857143
## 51   NA   NA 14.5  1  0    9   42 0.21428571 1.0000000 -0.7857143 0.7857143
## 52   NA   NA 15.0  1  0   15   42 0.35714286 1.0000000 -0.6428571 0.6428571
## 53   NA   NA 15.0  1  0   15   42 0.35714286 1.0000000 -0.6428571 0.6428571
## 54   NA   NA 15.0  1  0   15   42 0.35714286 1.0000000 -0.6428571 0.6428571
## 55   NA   NA 15.0  1  0   15   42 0.35714286 1.0000000 -0.6428571 0.6428571
## 56   NA   NA 15.0  1  0   15   42 0.35714286 1.0000000 -0.6428571 0.6428571
## 57   NA   NA 15.0  1  0   15   42 0.35714286 1.0000000 -0.6428571 0.6428571
## 58   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 59   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 60   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 61   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 62   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 63   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 64   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 65   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 66   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 67   NA   NA 15.5  1  0   25   42 0.59523810 1.0000000 -0.4047619 0.4047619
## 68   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 69   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 70   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 71   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 72   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 73   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 74   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 75   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 76   NA   NA 16.0  1  0   34   42 0.80952381 1.0000000 -0.1904762 0.1904762
## 77   NA   NA 16.5  1  0   42   42 1.00000000 1.0000000  0.0000000 0.0000000
## 78   NA   NA 16.5  1  0   42   42 1.00000000 1.0000000  0.0000000 0.0000000
## 79   NA   NA 16.5  1  0   42   42 1.00000000 1.0000000  0.0000000 0.0000000
## 80   NA   NA 16.5  1  0   42   42 1.00000000 1.0000000  0.0000000 0.0000000
## 81   NA   NA 16.5  1  0   42   42 1.00000000 1.0000000  0.0000000 0.0000000
## 82   NA   NA 16.5  1  0   42   42 1.00000000 1.0000000  0.0000000 0.0000000
```

```
## 83   NA   NA 16.5  1  0   42   42 1.00000000 1.0000000  0.0000000 0.0000000
## 84   NA   NA 16.5  1  0   42   42 1.00000000 1.0000000  0.0000000 0.0000000
##
## $Kmn
## [1] 0.952381
```

```r
ks.test(x, y)
```

```
##
##  Exact two-sample Kolmogorov-Smirnov test
##
## data:  x and y
## D = 0.95238, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Appendix 3

Kolmogorov–Smirnov Tables

Critical values, $d_{alpha};(n)^a$, of the maximum absolute difference between sample $F_n(x)$ and population $F(x)$ cumulative distribution.

Number of trials, n	Level of significance, α			
	0.10	0.05	0.02	0.01
1	0.95000	0.97500	0.99000	0.99500
2	0.77639	0.84189	0.90000	0.92929
3	0.63604	0.70760	0.78456	0.82900
4	0.56522	0.62394	0.68887	0.73424
5	0.50945	0.56328	0.62718	0.66853
6	0.46799	0.51926	0.57741	0.61661
7	0.43607	0.48342	0.53844	0.57581
8	0.40962	0.45427	0.50654	0.54179
9	0.38746	0.43001	0.47960	0.51332
10	0.36866	0.40925	0.45662	0.48893
11	0.35242	0.39122	0.43670	0.46770
12	0.33815	0.37543	0.41918	0.44905
13	0.32549	0.36143	0.40362	0.43247
14	0.31417	0.34890	0.38970	0.41762
15	0.30397	0.33760	0.37713	0.40420
16	0.29472	0.32733	0.36571	0.39201
17	0.28627	0.31796	0.35528	0.38086
18	0.27851	0.30936	0.34569	0.37062
19	0.27136	0.30143	0.33685	0.36117
20	0.26473	0.29408	0.32866	0.35241
21	0.25858	0.28724	0.32104	0.34427
22	0.25283	0.28087	0.31394	0.33666
23	0.24746	0.27490	0.30728	0.32954
24	0.24242	0.26931	0.30104	0.32286

Practical Reliability Engineering, Fifth Edition. Patrick D. T. O'Connor and Andre Kleyner.

Critical values, $d_{alpha};(n)^a$, of the maximum absolute difference between sample $F_n(x)$ and population $F(x)$ cumulative distribution.

Number of trials, n	Level of significance, α			
	0.10	0.05	0.02	0.01
25	0.23768	0.26404	0.29516	0.31657
26	0.23320	0.25907	0.28962	0.31064
27	0.22898	0.25438	0.28438	0.30502
28	0.22497	0.24993	0.27942	0.29971
29	0.22117	0.24571	0.27471	0.29466
30	0.21756	0.24170	0.27023	0.28987
31	0.21412	0.23788	0.26596	0.28530
32	0.21085	0.23424	0.26189	0.28094
33	0.20771	0.23076	0.25801	0.27677
34	0.20472	0.22743	0.25429	0.27279
35	0.20185	0.22425	0.26073	0.26897
36	0.19910	0.22119	0.24732	0.26532
37	0.19646	0.21826	0.24404	0.26180
38	0.19392	0.21544	0.24089	0.25843
39	0.19148	0.21273	0.23786	0.25518
40^b	0.18913	0.21012	0.23494	0.25205

[a] Values of $d_\alpha(n)$ such that $p(\max)|F^n(x) - F(x)|d^\alpha(n) = \alpha$.

[b] $N > 40 \approx \frac{1.22}{N^{1/2}}, \frac{1.36}{N^{1/2}}, \frac{1.51}{N^{1/2}}$ and $\frac{1.63}{N^{1/2}}$ for the four levels of significance.

4 Chapitre 4 : Problème de k > 2 échantillons

4.1 Test de Kruskal - Wallis

```
# Données
salaire <- c(420, 450, 470, 490,   # X1 : économie
             410, 430, 440, 470,   # X2 : éducation
             400, 410, 430, 440)   # X3 : santé

ministere <- factor(rep(c("Economie", "Education", "Sante"), each = 4))

data.frame(salaire, ministere)
```

```
##    salaire ministere
## 1      420  Economie
## 2      450  Economie
## 3      470  Economie
## 4      490  Economie
## 5      410 Education
## 6      430 Education
## 7      440 Education
## 8      470 Education
## 9      400     Sante
## 10     410     Sante
## 11     430     Sante
## 12     440     Sante
```

On veut tester :

H_0 : le salaire a les mêmes distributions contre H_1 : au moins une distribution est différente

4.1.1 Programmation du test de Kruskal - Wallis

```
# Calcul des rangs
rangs <- rank(salaire)
rangs
```

```
##  [1]  4.0  9.0 10.5 12.0  2.5  5.5  7.5 10.5  1.0  2.5  5.5  7.5
```

```
#somme des rangs par groupe
data <- data.frame(salaire, ministere, rangs)
S_i <- tapply(data$rangs, data$ministere, sum)
# Effectifs par groupe
n_i <- tapply(data$salaire, data$ministere, length)

print (S_i)
```

```
##  Economie Education     Sante
##      35.5      26.0      16.5
```

$$H = \frac{12}{n(n+1)} \sum_{i=1}^{K} \frac{S_i^2}{n_i} - 3(n+1)$$

```r
# Calcul de la statistique H

n <- length(salaire)
H <- (12 / (n * (n+ 1))) * sum((S_i^2) / n_i) - 3 * (n+ 1)
print(H)
```

```
## [1] 3.471154
```

Sous H0, H suit une loi de Khi-deux à 3-1 degrés de liberté.

```r
# nombre de degrés de libertés
ddl <- length(unique(ministere)) - 1
# valeur critique de la table de la loi de Khi-deux
stat_theo <- qchisq(0.95, df = 2)
p_value <- 1 - pchisq(H, ddl)

cat("Statistique calculée =", round(H, 3), "\n")
```

```
## Statistique calculée = 3.471
```

```r
cat("valeur critique =", round(stat_theo, 3), "\n")
```

```
## valeur critique = 5.991
```

```r
cat("p-value =", round(p_value,3), "\n")
```

```
## p-value = 0.176
```

3.47 < 5.99. Ainsi, au seuil de $ = $ 5 %, on ne peut rejeter H0. C'est dire que les distributions de salaires sont identiques.

4.1.2 Avec Kruskal.test

```r
kruskal.test(salaire ~ ministere)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  salaire by ministere
## Kruskal-Wallis chi-squared = 3.5204, df = 2, p-value = 0.172
```

p-value > 0.05. Ainsi, au seuil de $\alpha = 5$ %, on ne peut rejeter H0. C'est dire que les distributions de salaires sont identiques.

4.1.3 Correction des ex-aequos

Les rangs moyens utilisés dans le test introduisent une distorsion dans la distribution de H si des ex-aequos existent.

$$H_{corrig} = \frac{H}{1 - \sum_{i=1}^{s} \frac{(t_i^3 - t_i)}{n^3 - n}}$$

```
# Identification des tailles des groupes d'ex-aequos
freq <- table(salaire)          # compte les fréquences des valeurs
ties <- freq[freq > 1]          # garde seulement celles > 1
sum_ties <- sum(ties^3 - ties)  # somme des (t_i^3 - t_i)

# Correction
correction <- 1 - (sum_ties / (n^3 - n))
H_corrige <- H / correction

# 3. Affichage
cat("H =", round(H,3), "\n")
```

```
## H = 3.471
```

```
cat("Facteur de correction =", round(correction,3), "\n")
```

```
## Facteur de correction = 0.986
```

```
cat("H corrigée =", round(H_corrige,3), "\n")
```

```
## H corrigée = 3.52
```

```
cat("Valeur critique =", round(qchisq(0.95, 2),3), "\n")
```

```
## Valeur critique = 5.991
```

$3.52 < 5.991$ donc on ne peut rejeter H0.

NB : La fonction **Kruskal.test** corrige les ex-aequos.

5 CHAPITRE 5 : Estimation d'une densité

5.1 Estimation de la densité par histogramme

5.1.1 Construction de l'estimateur par l'histogramme

Soit $[a; b]$ l'intervalle contenant les observations $x_1, x_2, \ldots, x_n$.

On partitionne l'intervalle $[a; b]$ en J classes de longueur h. Les classes sont notées par :

$$A_j = [a_j; a_{j+1}], \quad j = 1, 2, \ldots, J$$

avec

$$h = a_{j+1} - a_j$$

Si on note n_j le nombre d'observations de l'échantillon appartenant à la classe A_j, alors :

$$n_j = \text{card}\{i : x_i \in A_j\} = \sum_{i=1}^{n} 1_{\{x_i \in A_j\}}$$

L'estimateur de la densité f sur la classe A_j est donné par :

$$\hat{f}_n(x) = \frac{n_j}{n \cdot h}, \quad x \in [a_j, a_{j+1}]$$

On peut aussi écrire :

$$\hat{f}_n(x) = \frac{1}{n} \cdot \frac{\text{card}\{x_i \leq a_{j+1}\} - \text{card}\{x_i \leq a_j\}}{h}$$

Note : Cet estimateur est une fonction en escalier, constante sur chaque classe A_j.

5.1.2 Application

```r
# Données
X <- c(1.1, 2.3, 1.7, 2.8, 3.2, 1.9, 2.5, 3.7, 2.9, 2.1)

# Taille de l'échantillon
n <- length(X)

# Définition des bornes de l'intervalle
a <- min(X)
b <- max(X)

# Nombre de classes
J <- 4
```

```r
# Largeur des classes
h <- (b - a) / J

# Définition des bornes des classes
breaks <- seq(a, b, by = h)
```

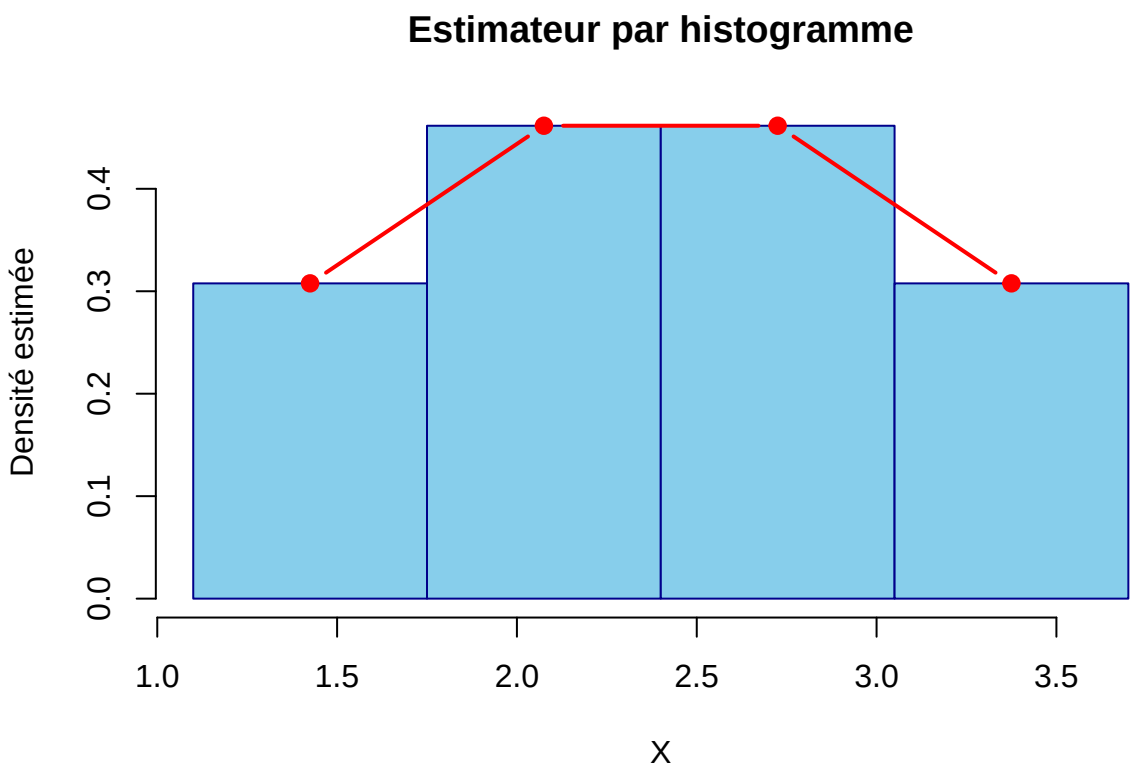
```r
# Construction de l'histogramme
hist_res <- hist(X, breaks = breaks, plot = FALSE)

# Effectifs par classe (n_j)
n_j <- hist_res$counts

# Estimateur de la densité par classe
f_hat <- n_j / (n * h)

# Affichage des résultats
data.frame(
  Classe = paste0("[", round(breaks[-length(breaks)], 2), ", ", round(breaks[-1], 2), "]"),
  Effectif = n_j,
  Densite_estimee = round(f_hat, 3)
)
```

5.1.2.1 Construisons à présent l'estimateur

```
##        Classe Effectif Densite_estimee
## 1 [1.1, 1.75]        2           0.308
## 2 [1.75, 2.4]        3           0.462
## 3 [2.4, 3.05]        3           0.462
## 4 [3.05, 3.7]        2           0.308
```

Traçons l'histogramme

```r
# Tracé de l'histogramme
hist_res <- hist(X, breaks = breaks, freq = FALSE, col = "skyblue",
                 main = "Estimateur par histogramme",
                 xlab = "X", ylab = "Densité estimée", border = "darkblue")

# Coordonnées du polygone des fréquences :
# abscisses = milieux des classes
mids <- hist_res$mids

# ordonnées = densité estimée (f_hat)
densites <- hist_res$density

# Tracé de la ligne du polygone des fréquences
lines(mids, densites, type = "b", col = "red", lwd = 2, pch = 19)
```



5.1.3 Estimation de la MISE et calcul de l'AMISE

- **Définition de l'erreur quadratique moyenne intégrée (MISE)**

L'erreur quadratique moyenne intégrée (MISE) est définie par :

$$\text{MISE} = \int_{-\infty}^{+\infty} E\left[(\hat{f}_n(x) - f(x))^2\right] dx$$

Pour l'estimateur par histogramme, l'approximation asymptotique de la MISE est :

$$\text{MISE} \approx \frac{h^2}{12} \int_{-\infty}^{+\infty} (f'(x))^2 dx + \frac{1}{nh} + o(h^2) + o\left(\frac{1}{nh}\right)$$

Le terme principal du MISE, appelé AMISE (MISE asymptotique), est :

$$\text{AMISE} = \frac{h^2}{12} \int_{-\infty}^{+\infty} (f'(x))^2 dx + \frac{1}{nh}$$

```r
# Données
X <- c(1.1, 2.3, 1.7, 2.8, 3.2, 1.9, 2.5, 3.7, 2.9, 2.1)
n <- length(X)
mu <- mean(X)
sigma <- sd(X)
```

```r
# Paramètres de l'histogramme
J <- 4
h <- (max(X) - min(X)) / J
```

- **Estimation empirique du MISE par simulation Monte Carlo**

Pour estimer empiriquement le MISE, on utilise la simulation Monte Carlo avec la formule :

$$\text{MISE} \approx \frac{1}{B} \sum_{b=1}^{B} \int (\hat{f}_n^{(b)}(x) - f(x))^2 dx$$

où $\hat{f}_n^{(b)}(x)$ est l'estimateur calculé sur le b-ième échantillon simulé et B est le nombre de répétitions Monte Carlo.

- **Estimation de la MISE et calcul de l'AMISE**

```r
# Données
X <- c(1.1, 2.3, 1.7, 2.8, 3.2, 1.9, 2.5, 3.7, 2.9, 2.1)
n <- length(X)
mu <- mean(X)
sigma <- sd(X)

# Paramètres de l'histogramme
J <- 4
h <- (max(X) - min(X)) / J
```

```r
# Estimation empirique du MISE par simulation
B <- 500  # nombre de répétitions Monte Carlo
grid_x <- seq(min(X)-1, max(X)+1, length.out = 500)
true_density <- dnorm(grid_x, mean = mu, sd = sigma)
mise_vals <- numeric(B)

for (b in 1:B) {
  # Génération d'un nouvel échantillon
  sample_b <- rnorm(n, mean = mu, sd = sigma)
  breaks_b <- seq(min(sample_b), max(sample_b), length.out = J + 1)
  hist_b <- hist(sample_b, breaks = breaks_b, plot = FALSE)
  mids_b <- hist_b$mids
  f_hat_b <- hist_b$density

  # Interpolation de la densité estimée
  f_interp <- approx(x = mids_b, y = f_hat_b, xout = grid_x,
                     method = "constant", rule = 2)$y

  # Calcul de l'erreur quadratique intégrée
  mise_vals[b] <- mean((f_interp - true_density)^2)
```

```
}

# Moyenne des erreurs → estimation du MISE
MISE_estimee <- mean(mise_vals)

# Calcul théorique de l'AMISE
# Pour la densité normale N(mu, sigma^2), on connaît :
I_fprime2 <- 1 / (2 * pi * sigma^5)  #   (f')^2 dx
AMISE <- (1 / (n * h)) + (h^2 / 12) * I_fprime2

# Affichage des résultats
cat("Estimation empirique du MISE :", round(MISE_estimee, 6), "\n")
```

```
## Estimation empirique du MISE : 0.180887
```

```
cat("Valeur théorique de l'AMISE   :", round(AMISE, 6), "\n")
```

```
## Valeur théorique de l'AMISE   : 0.175143
```

5.1.3.1 Calcul théorique de l'AMISE

Pour une densité normale $N(\mu, \sigma^2)$, on connaît la valeur analytique de :

$$\int_{-\infty}^{+\infty} (f'(x))^2 dx = \frac{1}{2\pi^{1/2}\sigma^5}$$

D'où l'AMISE théorique pour la densité normale :

$$\text{AMISE} = \frac{h^2}{12} \times \frac{1}{2\pi^{1/2}\sigma^5} + \frac{1}{nh}$$

```
# Calcul théorique de l'AMISE
# Pour la densité normale N(mu, sigma²), on connaît :
I_fprime2 <- 1 / (2 * pi * sigma^5)  #
AMISE <- (1 / (n * h)) + (h^2 / 12) * I_fprime2

# Affichage des résultats
cat("Estimation empirique du MISE :", round(MISE_estimee, 6), "\n")
```

```
## Estimation empirique du MISE : 0.180887
```

```
cat("Valeur théorique de l'AMISE   :", round(AMISE, 6), "\n")
```

```
## Valeur théorique de l'AMISE   : 0.175143
```

5.1.4 Calcul de la fenêtre optimale h^*

La fenêtre optimale théorique qui minimise l'AMISE est obtenue en résolvant :

$$\frac{d}{dh}(\text{AMISE}) = 0$$

Ce qui donne :

$$h^* = \left[\frac{6}{n \times \int_{-\infty}^{+\infty}(f'(x))^2 dx}\right]^{1/3}$$

L'AMISE avec la fenêtre optimale est proportionnelle à $n^{-2/3}$, ce qui donne la vitesse de convergence de l'estimateur par histogramme.

```
# Calcul de la fenêtre optimale
h_optimal <- (6 / (n * I_fprime2))^(1/3)

# AMISE avec la fenêtre optimale
AMISE_optimal <- (h_optimal^2 / 12) * I_fprime2 + 1 / (n * h_optimal)

cat("Fenêtre optimale h*           :", round(h_optimal, 4), "\n")
```

```
## Fenêtre optimale h*           : 0.9973
```

```
cat("AMISE avec h* optimal         :", round(AMISE_optimal, 6), "\n")
```

```
## AMISE avec h* optimal         : 0.150405
```

```
cat("Fenêtre utilisée h            :", round(h, 4), "\n")
```

```
## Fenêtre utilisée h            : 0.65
```

5.1.5 Calcul de h de façon automatique

```
X <- c(1.1, 2.3, 1.7, 2.8, 3.2, 1.9, 2.5, 3.7, 2.9, 2.1)
n <- length(X)

# Choix automatique du nombre de classes selon Sturges
J <- ceiling(log2(n) + 1)

# Bornes des classes
breaks <- seq(min(X), max(X), length.out = J + 1)
```

```r
# Histogramme sans affichage
hist_res <- hist(X, breaks = breaks, plot = FALSE)

# Effectifs
n_j <- hist_res$counts

# Largeur classes (elles sont égales ici)
h <- diff(breaks)[1]

# Estimation densité
f_hat <- n_j / (n * h)

# Tableau résultats
data.frame(
  Classe = paste0("[", round(breaks[-length(breaks)], 2), ", ", round(breaks[-1], 2), "]"),
  Effectif = n_j,
  Densite_estimee = round(f_hat, 3)
)
```

```
##         Classe Effectif Densite_estimee
## 1  [1.1, 1.62]        1           0.192
## 2 [1.62, 2.14]        3           0.577
## 3 [2.14, 2.66]        2           0.385
## 4 [2.66, 3.18]        2           0.385
## 5  [3.18, 3.7]        2           0.385
```

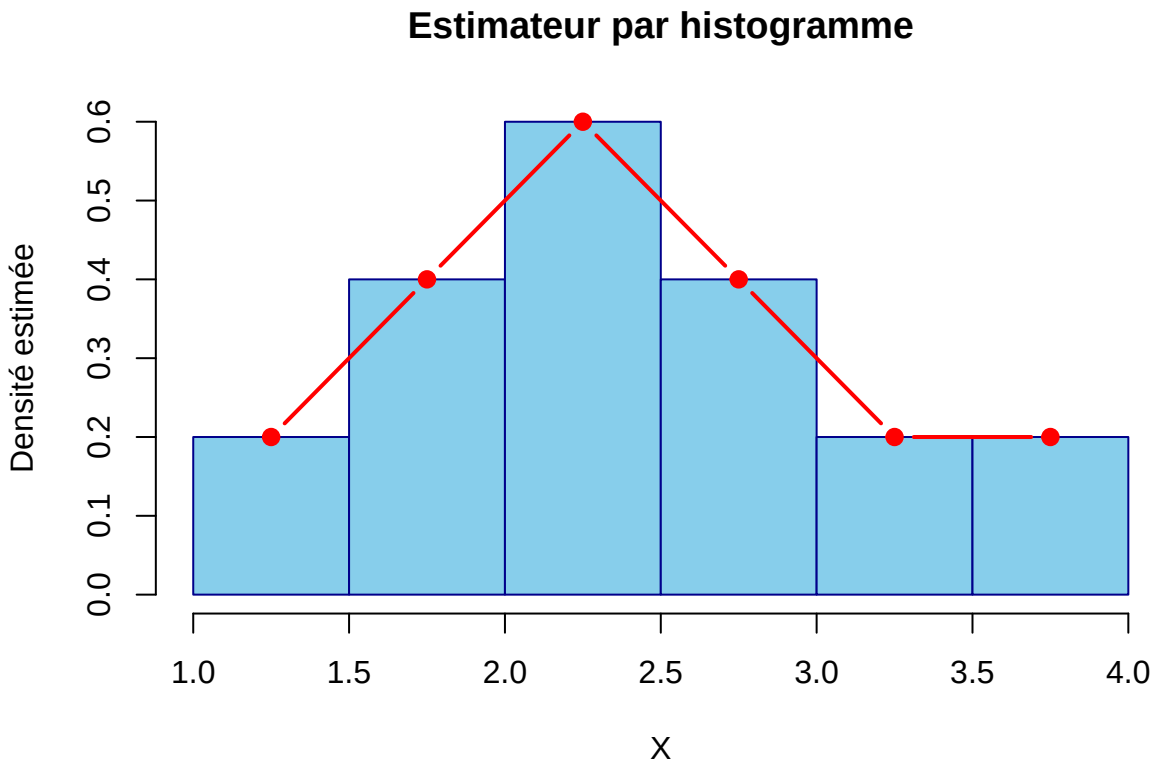
```r
# Données
X <- c(1.1, 2.3, 1.7, 2.8, 3.2, 1.9, 2.5, 3.7, 2.9, 2.1)

# Taille de l'échantillon
n <- length(X)

# Définition des bornes de l'intervalle
a <- min(X)
b <- max(X)

# Nombre de classes
#J <- 4

# Largeur des classes
h <- 0.9973

# Définition des bornes des classes
breaks <- seq(a, b, by = h)
```

5.1.6 Construisons à présent l'estimateur

```r
# Construction de l'histogramme
hist_res <- hist(X, plot = FALSE)

# Effectifs par classe (n_j)
n_j <- hist_res$counts

# Estimateur de la densité par classe
f_hat <- n_j / (n * h)

# Affichage des résultats
data.frame(
  Classe = paste0("[", round(breaks[-length(breaks)], 2), ", ", round(breaks[-1], 2), "]"),
  Effectif = n_j,
  Densite_estimee = round(f_hat, 3)
)
```

```
##        Classe Effectif Densite_estimee
## 1  [1.1, 2.1]        1           0.100
## 2 [2.1, 3.09]        2           0.201
## 3  [1.1, 2.1]        3           0.301
## 4 [2.1, 3.09]        2           0.201
## 5  [1.1, 2.1]        1           0.100
## 6 [2.1, 3.09]        1           0.100
```

```r
# Tracé de l'histogramme
hist_res <- hist(X,  freq = FALSE, col = "skyblue",
                 main = "Estimateur par histogramme",
                 xlab = "X", ylab = "Densité estimée", border = "darkblue")

# Coordonnées du polygone des fréquences :
# abscisses = milieux des classes
mids <- hist_res$mids

# ordonnées = densité estimée (f_hat)
densites <- hist_res$density

# Tracé de la ligne du polygone des fréquences
lines(mids, densites, type = "b", col = "red", lwd = 2, pch = 19)
```



5.2 Estimation par la méthode de l'histogramme : validation croisée

5.2.1 Principe de la validation croisée

La validation croisée est une méthode générique pour sélectionner des paramètres de modèles (comme le nombre de classes dans un histogramme) en évaluant leur performance prédictive. En statistique non paramétrique, elle permet d'optimiser des estimateurs (densité, régression, etc.)

```
set.seed(123)
CV_hist=function(x) # définition de la fonction en fonction de l'échantillon x
{
  n=length(x)
  # valeurs minimale et maximale de x, utilisées pour délimiter les bornes de l'histogramme
  a=min(x)
  b=max(x)
  N=round(n/5); # nombre maximal de classe à tester
  b=b+(b-a)/N
  m_CV=1
  J0=2/(n-1)
  J=1:N;

  # Boucle pour tester différents nombres de classes
  for (m in 2:(N+1)){
    h=(b-a)/m
```

```
    hatp=1:m
    A=(1:m)%*%t((1:n)*0+1)
    xx=((1:m)*0+1)%*%t((x-a)/h)
    hatp=rowSums(((A-1)<=xx)*(xx<A))/n
    J[m-1]=2-(n+1)*sum(hatp^2)
    remove(hatp)
    J[m-1]=J[m-1]/((n-1)*h)

    # Choix du nombre optimal de classes
    if (J[m-1]<J0) {m_CV=m; J0=J[m-1]}
  }
  op=par(mfcol=c(1,2),pty="m",omi=c(0,0,0,0))
  plot(2:(N+1),J,type='l',lwd=2,col='darkred',
       main="La courbe de la fonction de validation croisée',,xlab='nb de classes",ylab='CV')
  h=(b-a)/m_CV
  hatf=1:m_CV
  n=length(x)
  m=m_CV

  # Calcul de l'estimateur de densité optimal
  for (j in 1:m_CV){hatf[j]=sum(((j-1)*h<=x-a)*(x-a<j*h))/(n*h)}
  xleft=a-h+(1:m)*h
  xright=xleft+h
  ybottom=(1:m)*0
  ytop=hatf
  plot(c(a-h/n,xleft,b),c(0,hatf,0),type="n",xlab="Les classes",
       ylab="Estimateur de densité",main="Histogramme avec le nombre de classes optimal")

  rect(xleft, ybottom, xright, ytop, col = "cyan", border = "darkblue", lwd = 1)
  par(op)
  return(m_CV)
}
```

n: Nombre d'observations dans l'échantillon x. a: Minimum des valeurs de x. b: Maximum des valeurs de x. N: Nombre initial de classes, souvent fixé à n/5. m_CV: Variable pour stocker le nombre optimal de classes sélectionné. J0: Initialisation d'une mesure de performance basée sur la validation croisée.

5.2.2 Boucle de validation croisée

La boucle for (m in 2:(N+1)) parcourt différents nombres de classes possibles. Pour chaque nombre de classes m :

- Calcule la largeur de classe h,
- Estime la fonction de densité à l'aide de l'histogramme pour chaque nombre de classes,
- Calcule une mesure d'erreur de l'histogramme à l'aide de la validation croisée (J[m-1]),
- Compare J[m-1] avec J0 (la meilleure performance observée jusqu'à présent) et met à jour m_CV si une meilleure performance est trouvée avec le nombre actuel de classes m,
- Trace la courbe de la fonction de validation croisée (J par rapport à m),

- Identifie le nombre optimal de classes m_CV qui minimise l'erreur de validation croisée,
- Trace l'histogramme final avec le nombre optimal de classes sélectionné.

La fonction CV_hist retourne m_CV, qui est le nombre optimal de classes déterminé par la méthode de validation croisée.

En résumé, la méthode de validation croisée évalue différentes configurations d'histogrammes (en termes de nombre de classes) en utilisant une mesure d'erreur spécifique (J) basée sur la performance de l'histogramme pour estimer la densité des données. Elle sélectionne le nombre de classes qui minimise cette erreur, permettant ainsi de trouver une estimation optimale de la densité des données à partir de l'échantillon x.

5.2.3 Autres méthodes pour trouver le nombre de classes optimales

- **Méthode de Sturges**

La méthode de Sturges est une règle empirique simple pour déterminer le nombre de classes k dans un histogramme. Elle est particulièrement adaptée aux distributions symétriques et unimodales (comme la loi normale).

```r
n <- 8000
x <- runif(n, 0, 5)

Cv_hist_stur <- function(sample) {
  n <- length(sample)
  k_optimal <- floor(1 + log2(n))
  return(k_optimal)
}

Cv_hist_stur(x)
```

Commentaire: On trouve 13 classes: ce qui veut dire que nos donées sont découpées en 13 classes ou intervalles de classes.

- **Méthode de Freedman-Diaconis :** La méthode de Freedman-Diaconis est une approche robuste qui tient compte de la dispersion des données via l'écart interquartile (IQR). Elle est optimale pour les distributions asymétriques ou avec outliers.

```r
# méthode simple mais peu adapté à de grands échantillons
Freedman_Diaconis_hist <- function(sample) {
  n <- length(sample)
  IQR_value <- IQR(sample) # calcul de l'écart interquartile
  h <- 2 * IQR_value * n^(-1/3)
  range_x <- max(sample) - min(sample) # etendue des données
  k_optimal <- floor(range_x / h)
  return(k_optimal)
}
```

```r
# méthode plus robuste
Freedman_Diaconis_hist(x)

# validation croisée
CV_hist(x)
```

5.2.4 Analyse des graphiques

Pour le premier graphe, l'axe horizontal représente le nombre de classes (bins) testées : de 2 à N+1.

L'axe vertical représente la valeur de la fonction de validation croisée (CV) associée à chaque nombre de classes. La courbe montre comment évolue la qualité de l'estimation de la densité en fonction du nombre de classes.

Pour le deuxième graphe, il s'agit de l'estimation de densité à l'aide d'un histogramme utilisant le nombre de classes optimal sélectionné via la validation croisée.

Les barres bleues représentent les valeurs estimées de la densité pour chaque intervalle.

Cet histogramme correspond à une estimation par histogramme du support [0, 5] de la variable simulée (loi uniforme).

5.3 Estimation par la méthode du noyau

On suppose qu'on dispose de n observations $x_1, ..., x_n$.

Un estimateur de la densité de la loi qui régie les données $x_1, ..., x_n$ est donnée par :

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

où K est un noyau(définition dans le cours).

5.3.1 Noyau de Parzen-Rosenblatt

Données

```r
# Données
X <- c(1.1, 2.3, 1.7, 2.8, 3.2, 1.9, 2.5, 2.9, 2.1, 3.7)
n <- length(X)

#  h (amplitude / 4)
h <- (max(X) - min(X)) / 4
cat("h =", h, "\n")
```

```
## h = 0.65
```

```r
bornes <- seq(min(X),max(X), h)
```

Estimation par la méthode des histogrammes :

Soient [a,b] l'intervalle contenant les observations $(x_1, x_2, ...x_n)$.On partitionne l'intervalle [a,b] en J classes de longueur h. Les classes sont notées : $A_j = [a_j, a_{j+1}]$. Chaque classe dispose d'un effectif n_j ,j=1,2,...,J et $h_j = a_{j+1} - a_j$. Autrement dit, $Card(A_j) = n_j$. Et,

$$f_n = \frac{n_j}{n \times h}$$

```r
n_ <- numeric(4)
for (i in X){
  if (bornes[1]<= i & i < bornes[2]){
    n_[1] <- n_[1]+1
  }
  if (bornes[2]<= i & i < bornes[3]){
    n_[2] <- n_[2]+1
  }
  if (bornes[3]<= i & i < bornes[4]){
    n_[3] <- n_[3]+1
  }
  if (bornes[4]<= i & i <= bornes[5]){
    n_[4] <- n_[4]+1
  }
}
n_
```

```
## [1] 2 3 3 2
```

```r
f_n_histogramme <- n_/ (n*4)
f_n_histogramme
```

```
## [1] 0.050 0.075 0.075 0.050
```

```r
# Vérification de la somme
sum(f_n_histogramme)*4==1
```

```
## [1] TRUE
```

Estimation par le noyau de ROSENBALTT :

Pour pallier les limites de l'estimateur de la densité par la méthode des histogrammes, plus précisement celle liée à la **continuité** de l'estimateur de la densité, on utilise la méthode d'estimation par le noyau. Cette partie porte sur le noyau de ROSENBALTT.

Formule de la densité

$$\hat{f}_n(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h}K(\frac{x-x_i}{h})$$

Avec

$$K(x) = \frac{1}{2}1_{[-1,1]}(x)$$

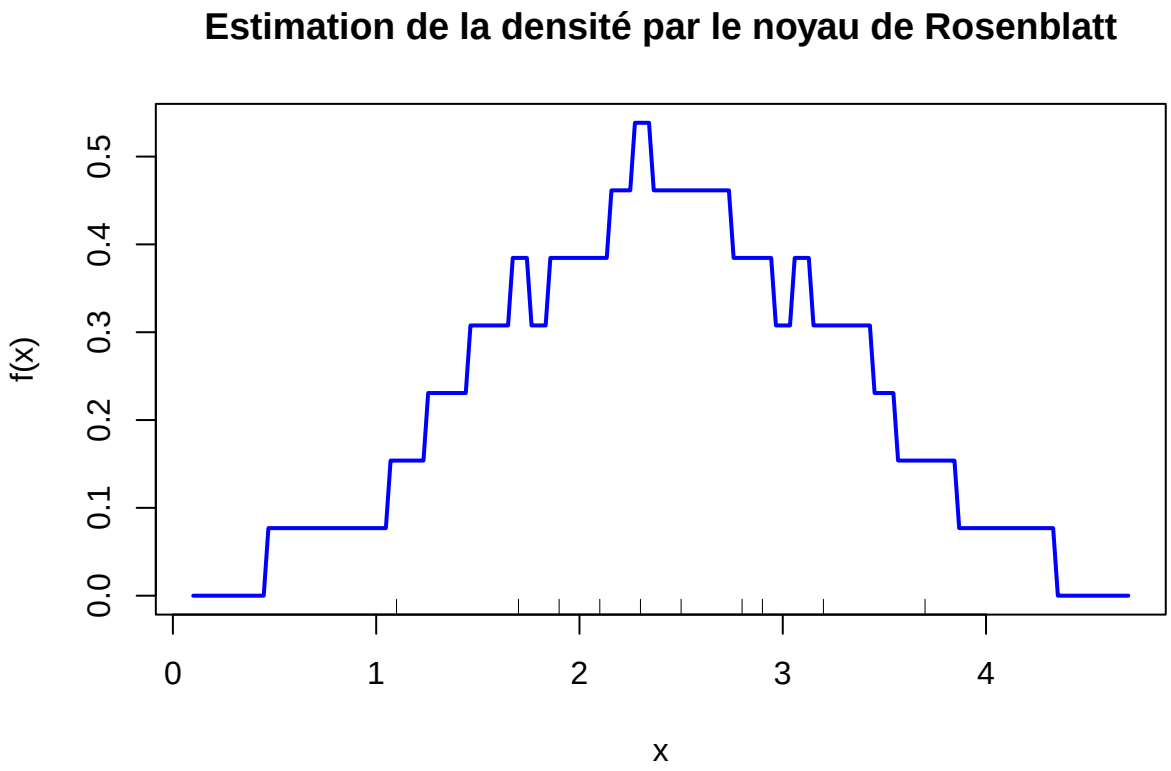
```r
# Fonctions et boucles

# Fonction de Rosenblatt
Rosenblatt <- function(u) {
  ifelse(abs(u) <= 1, 0.5, 0)
}

# Fonction pour estimer f(x) en un point x
fonction_density <- function(x, X, h) {
  n <- length(X)
  u <- (x - X) / h
  fx <- sum(Rosenblatt(u)) / (n * h)
  return(fx)
}
# Boucle pour calculer
densite <- numeric(n)

for (i in 1:n) {
  densite[i] <- fonction_density(X[i], X, h)
}
```

```r
# Afficher les résultats
data.frame(X = X, f_n = round(densite, 4))
```

```
##       X    f_n
## 1   1.1 0.1538
## 2   2.3 0.5385
## 3   1.7 0.3846
## 4   2.8 0.3846
## 5   3.2 0.3077
## 6   1.9 0.3846
## 7   2.5 0.4615
## 8   2.9 0.3846
## 9   2.1 0.3846
## 10  3.7 0.1538
```

```r
# Graphique
x_grid <- seq(min(X) - 1, max(X) + 1, length.out = 200)
dens_grid <- sapply(x_grid, function(x) fonction_density(x, X, h))

plot(x_grid, dens_grid, type = "l", lwd = 2, col = "blue",
```

```
     main = "Estimation de la densité par le noyau de Rosenblatt",
     xlab = "x", ylab = "f(x)")
rug(X)
```



Avec une fonction native de R : Il n'y pas de fonction native de R permettant de calculer la densité avec le noyau de Parzen-Rosenblatt.

5.3.2 Noyau triangulaire

L'objectif ici , c'est d'estimer la densité de la loi de x à partir de la méthode du noyau en utilisant le noyau triangulaire .

La densité estimée au point X est donnée par la forme ci-après :

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

avec K donné par :

$$K(u) = (1 - |u|) \cdot \mathbb{1}_{\{|u| \leq 1\}}$$

- **Fonction du noyau triangulaire**

```r
triangular_kernel <- function(u) {
  ifelse(abs(u) <= 1, 1 - abs(u), 0)
}
```

- **Fonction pour l'estimation de la densité aux points x**

```r
# Estimation de la densité aux points de x
estimate_density_x <- function(x, h) {
  n <- length(x)
  f_hat <- numeric(n)

  for (i in seq_along(x)) {
    u <- (x-x[i]) / h
    f_hat[i] <- sum(triangular_kernel(u)) / (n * h)
  }

  return(f_hat)
}
```

- **Application de la fonction pour estimer la densité aux points de x**

Utilisons des données de l'exercice du cours :

```r
# Données
x <- c(1.1, 2.3, 1.7,2.8, 3.2, 1.9, 2.5, 3.7, 2.9, 2.1)
```

$$h = \frac{\max(x) - \min(x)}{\text{nombre de classes}}$$

```r
# Nombre de classes
k <- 4

# Calcul de la fenêtre h
h <- (max(x) - min(x)) / k

# Affichage
h
```

```
## [1] 0.65
```

```r
densities_x <- estimate_density_x(x, h)

resultats <- data.frame(
  x = x,
  densite_estimee = densities_x
)

# Affichage du tableau
resultats
```

```
##      x densite_estimee
## 1  1.1       0.1656805
## 2  2.3       0.4852071
## 3  1.7       0.3431953
## 4  2.8       0.4615385
## 5  3.2       0.3313609
## 6  1.9       0.4378698
## 7  2.5       0.4733728
## 8  3.7       0.1893491
## 9  2.9       0.4378698
## 10 2.1       0.4852071
```

5.3.3 Noyau Triweight

- **Estimation de densité**

L'estimateur à noyau de la densité est défini par :

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

avec :

- n la taille de l'échantillion
- h la bande passante (paramétre de lissage)
- K la fonction noyau

Le noyau de Triweight est défini par :

$$K(u) = \begin{cases} \dfrac{35}{32}\,(1-u^2)^3, & |u| \le 1, \\ 0, & |u| > 1. \end{cases}$$

- **Définition du noyau de Triweight :**

Syntaxe par defaut de R: bkde(estim_notes, kernel = "triweight", bandwidth = h) et h = bw.nrd0(estim_notes), estimation de h par la methode de Silverman Cette methode suppose que notre dataset suit une loi normale.

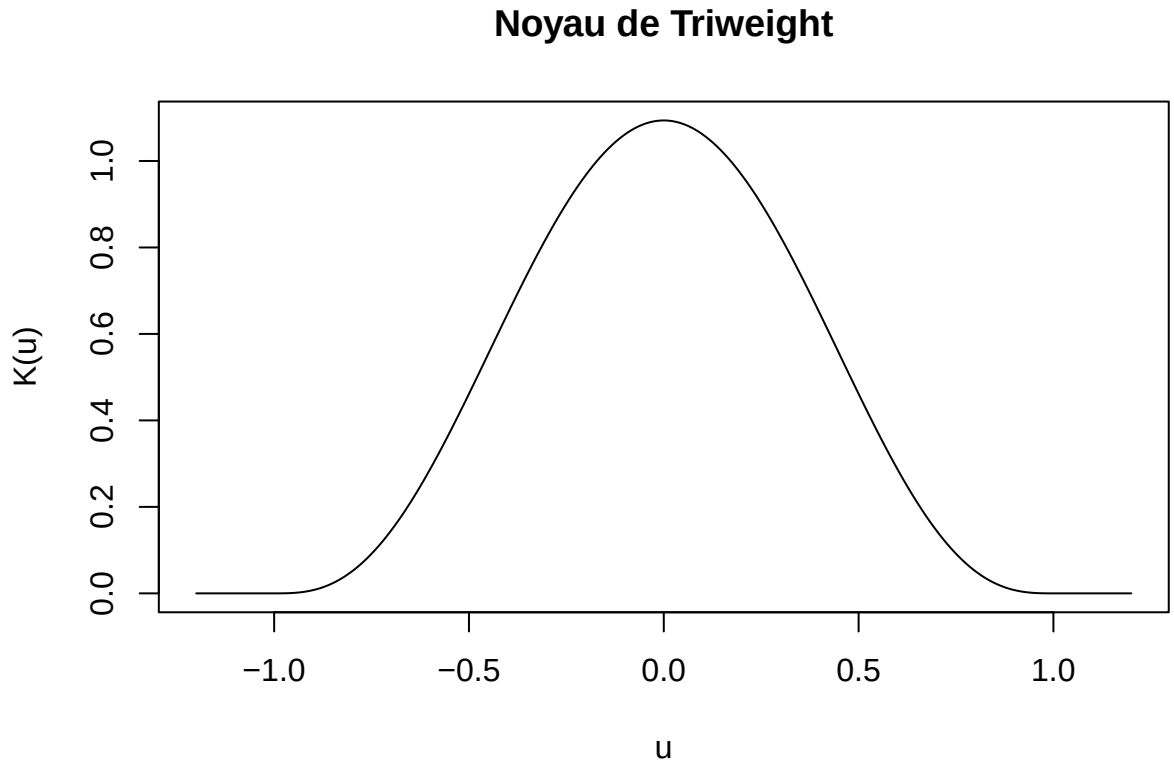
```
triweight_kernel <- function(u) {
  ifelse(abs(u) <= 1, (35/32) * (1 - u^2)^3, 0)
}
```

- **Fonction d'estimation**

```r
density_triweight <- function(x0, X, h) {
  u <- (x0 - X) / h # u est un vecteur
  mean(triweight_kernel(u)) / h
}
```

- **Visualisation de K(u) sur [-1.2,1.2]**

```r
u <- seq(-1.2, 1.2, length.out = 400) # vecteur de 400 vleurs equidistant sur l'intervalle
plot(u, triweight_kernel(u), type = 'l',
     main = 'Noyau de Triweight', xlab = 'u', ylab = 'K(u)')
```




```r
data <- read_excel("BASES/notes_Triweight_kernel.xlsx")
head(data)
```

```
## # A tibble: 6 x 2
##   `Estimation/Moyenne` `Anthropo/Moyenne`
##                  <dbl>              <dbl>
## 1                    8               16.5
## 2                   13               14.5
## 3                    8               15.5
## 4                   10                8
## 5                   10               15
## 6                    8.5             15
```

```r
estim_notes <- data[[1]]
anthrop_notes <- data[[2]]
```

- **Estimation de la densité**

ref: https://fastercapital.com/fr/contenu/La-regle-empirique-de-Silverman---une-regle-a-modeliser-par---les-connaissances-de-Silverman-sur-la-regression-du-noyau.html

$$h = 0.9 \cdot \min\left(\sigma, \frac{\text{IQR}}{1.34}\right) \cdot n^{-1/5}$$

```r
# Grille d'evaluaton
grid <- seq(min(estim_notes) - 1, max(estim_notes) + 1, length.out = 600)

#print(grid)

# Calcul de h par la régle empirique de Silverman'.
# Le principe est de minimiser l'erreur quadratique moyenne (pour n grand, alors AMISE)

h_theorique <- 1.06 * sd(estim_notes) * length(estim_notes)^(-1/5)
# Ce h est la version theorique , elle est sensible aux outliers


# En pratique on se sert d'une version plus robuste (c'est elle qui s'obtient avec bw.nrd0(
# car les données ne sont pas necessairement parfaitement normale donc possibilite de valeu
h_silverman <- 0.9 * min(sd(estim_notes), IQR(estim_notes)/1.34) * length(estim_notes)^(-1/5

cat("h théorique (1.06 ·   · n^(-1/5)) :", round(h_theorique, 4), "\n")
```

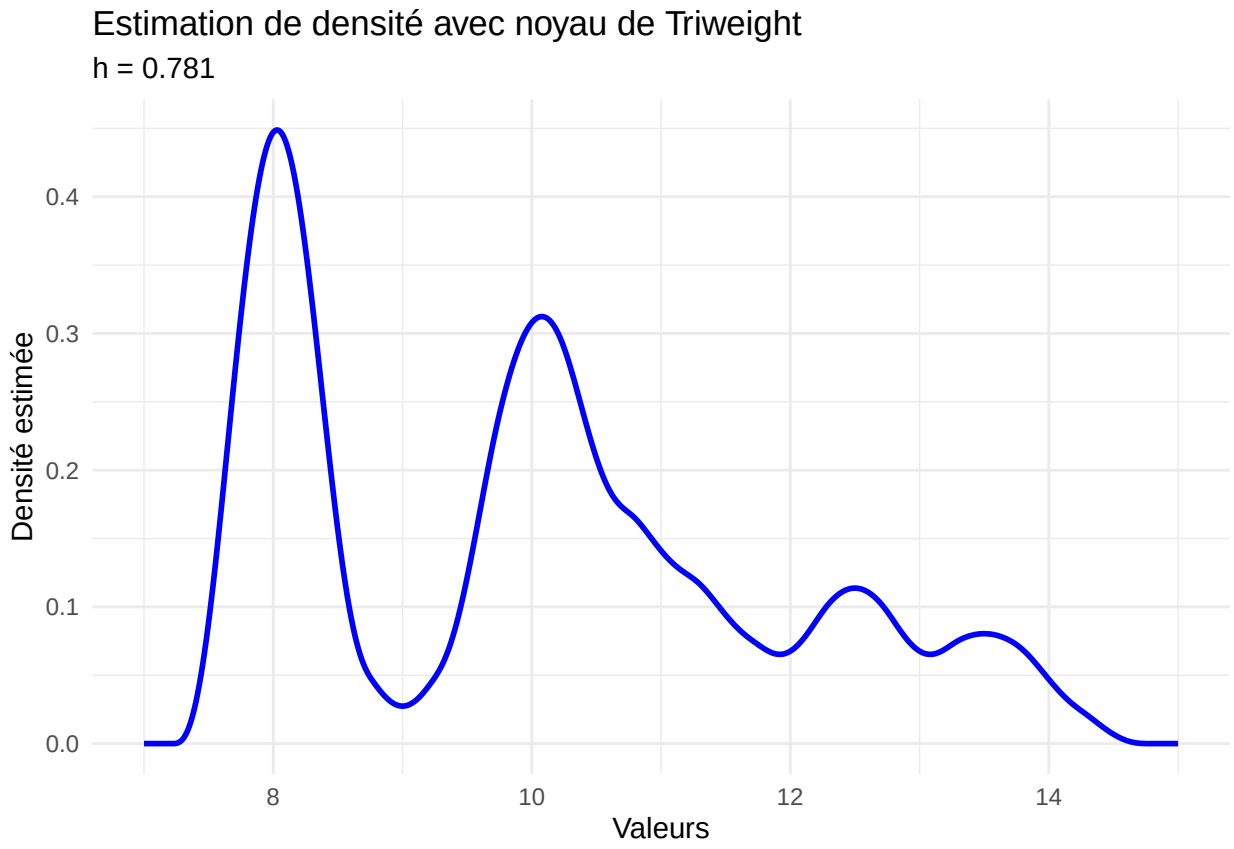
```
## h théorique (1.06 ·   · n^(-1/5)) : 0.9199
```

```r
cat("h robuste (bw.nrd0)              :", round(h_silverman, 4), "\n")
```

```
## h robuste (bw.nrd0)              : 0.781
```

```r
# Calcul de l'estimation de tous les points

dens_tri <- sapply(grid, function(x0) density_triweight(x0, X = estim_notes, h = h_silverman
```

```r
df <- data.frame(x = grid, dens = dens_tri)
ggplot(df, aes(x = x, y = dens)) +
  geom_line(size = 1, color = 'blue') +
  labs(
    title = 'Estimation de densité avec noyau de Triweight',
    subtitle = paste('h =', round(h_silverman, 3)),
    x = 'Valeurs', y = 'Densité estimée'
  ) +
  theme_minimal()
```



- **Avec le package KernSmooth**

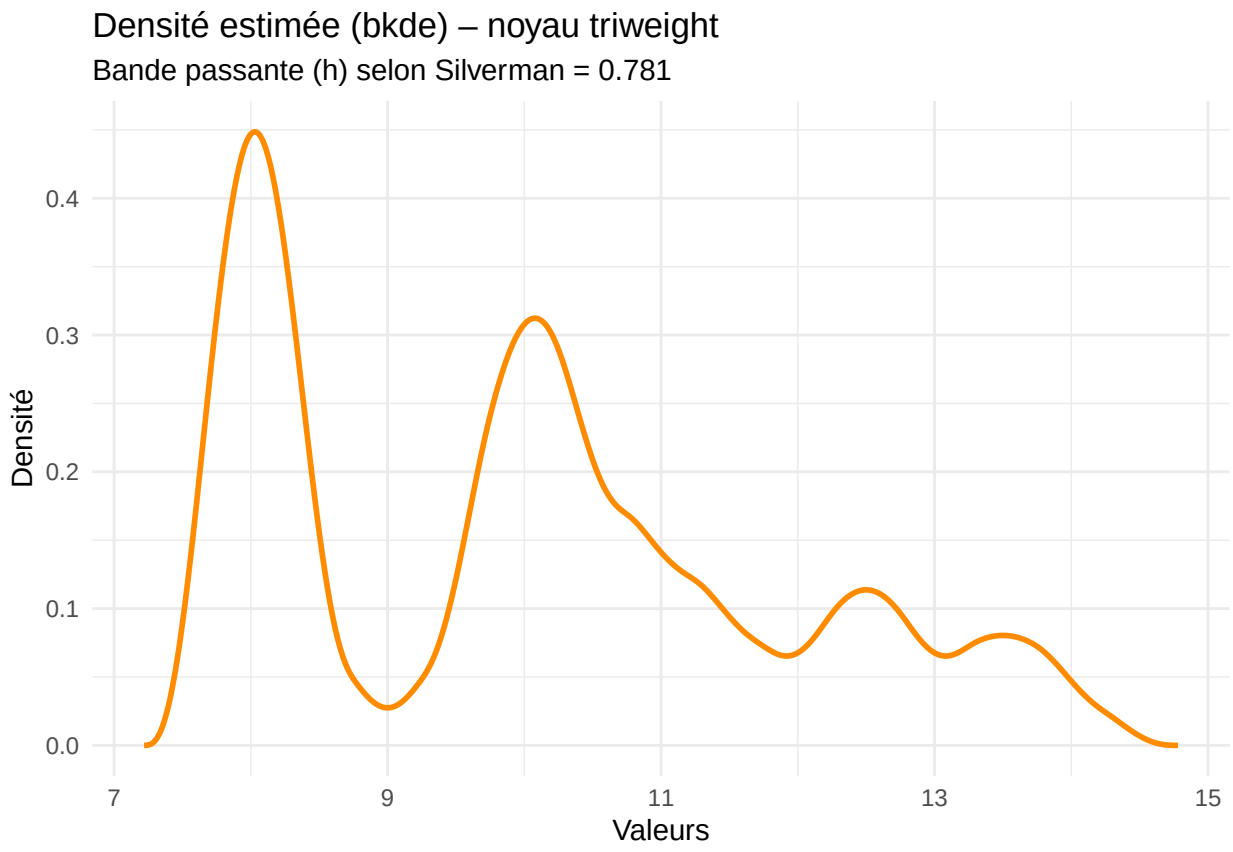
Par defaut, le noyau triweight n'est pas implementé dans la fonction density,on se sert du package Par defaut, le noyau triweight n'est pas implementé dans la fonction density. On se sert du package KernSmooth

```
h <- bw.nrd0(estim_notes)  # methode de Silverman, argument par defaut dans la fonction den
print(h)
```

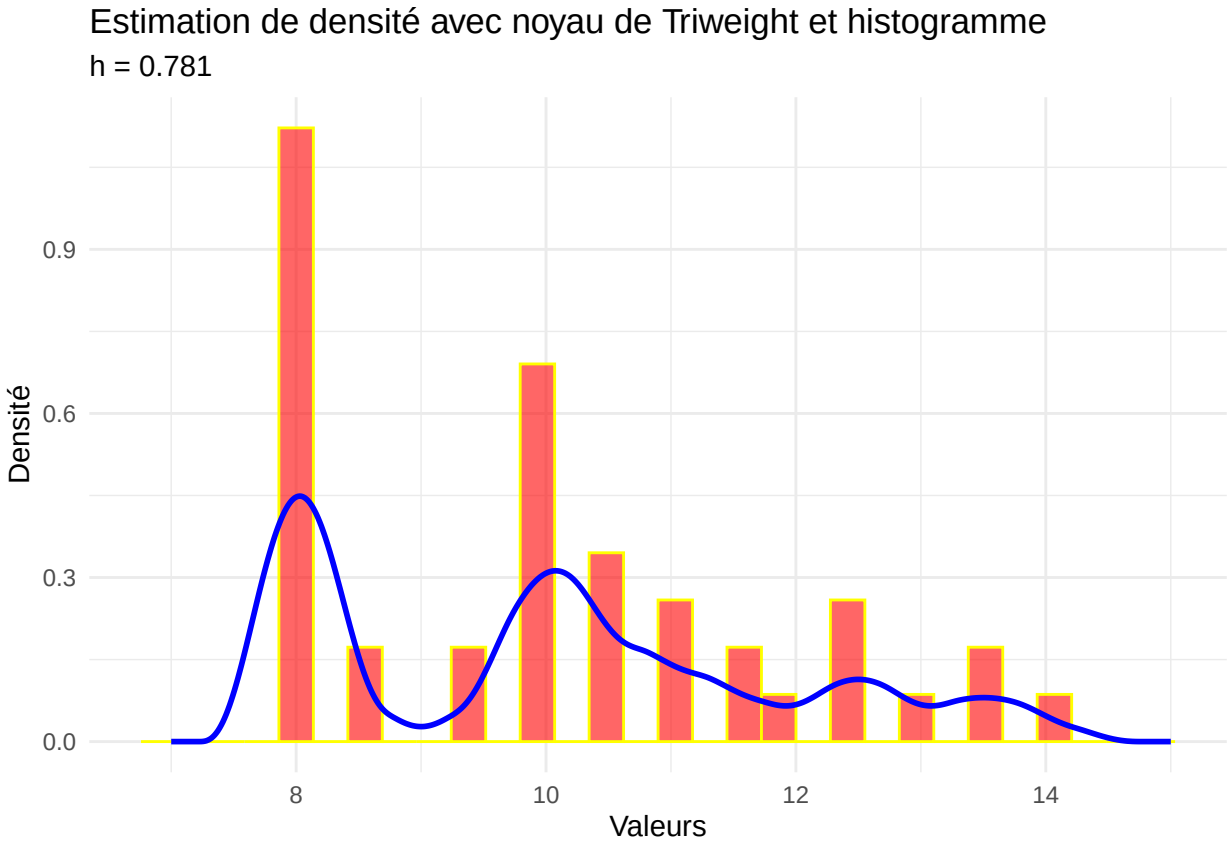
```
## [1] 0.7810198
```

```
res <- bkde(estim_notes, kernel = "triweight", bandwidth = h)


df_kd <- data.frame(x = res$x, y = res$y)


ggplot(df_kd, aes(x = x, y = y)) +
  geom_line(color = "darkorange", size = 1) +
  labs(
    title = "Densité estimée (bkde) - noyau triweight",
    subtitle = paste0("Bande passante (h) selon Silverman = ", round(h, 3)),
    x = "Valeurs",
    y = "Densité"
```

```
  ) +
  theme_minimal()
```

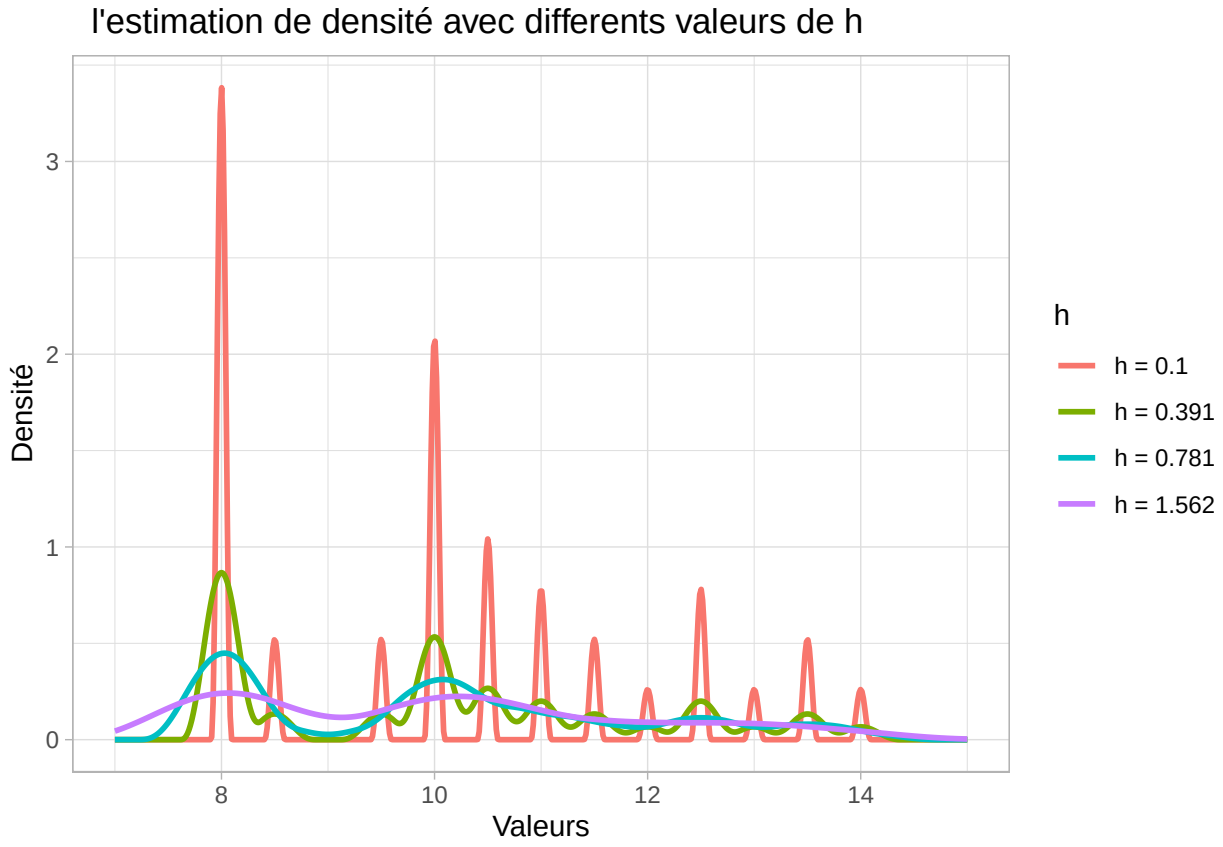



```
# df <- data.frame(x = grid, dens = dens_tri)

ggplot() +
  # Histogramme normalisé (les hauteurs correspondent a une densité)
  geom_histogram(aes(x = estim_notes, y = after_stat(density)),
                 bins = 30, fill = "red", color = "yellow", alpha = 0.6) +

  # Courbe de densité estimée
  geom_line(data = df, aes(x = x, y = dens),
            color = "blue", size = 1) +

  labs(
    title = "Estimation de densité avec noyau de Triweight et histogramme",
    subtitle = paste("h =", round(h_silverman, 3)),
    x = "Valeurs",
    y = "Densité"
  ) +
  theme_minimal()
```



- **Influence de la valeur de h**

```
h_values <- c(0.1, h_silverman / 2, h_silverman, 2 * h_silverman)
# Pour chaque valeur de h on créer un data frame puis on les merge tous.

df_h <- do.call(rbind, lapply(h_values, function(h) {
  dens <- sapply(grid, function(x0) density_triweight(x0, X = estim_notes, h = h))
  data.frame(
    x    = grid,
    dens = dens,
    h    = paste0('h = ', round(h, 3))# arrondir a 3 chiffres
  )
}))

ggplot(df_h, aes(x = x, y = dens, color = h)) +
  geom_line(size = 1) +
  labs(
    title = "  l'estimation de densité avec differents valeurs de h",
    x = "Valeurs", y = "Densité"
  ) +
  theme_light()
```



- **Calcul de h par la methode AMISE**

Source : http://archives.univ-biskra.dz/bitstream/123456789/21522/1/Baia_Ikram.pdf , page 25
La formule asymptotique de l'erreur quadratique moyenne intégrée est :

$$\text{AMISE}(h) \approx \frac{R(K)}{nh} + \frac{1}{4}h^4\mu_2(K)^2R(f'')$$

Où :

- $$R(K) = \int K^2(u)\,du)$$
- $$\mu_2(K) = \int u^2K(u)\,du)$$
- $$R(f'') = \int [f''(x)]^2 dx)$$
- n est la taille de l'échantillon La valeur de h qui minimise cette AMISE est :

$$h_{\text{AMISE}} = \left(\frac{R(K)}{n\mu_2(K)^2R(f'')}\right)^{1/5}$$

Pour le noyau **Triweight** : -

$$(R(K) = \frac{350}{429})$$

-

$$(\mu_2(K) = \frac{1}{9})$$

R(f'') dépend de la vrai densité f qui est inconnue en pratique.

On approxime R(f'') sous l'hypothèse que les données suivent une densité normale, auquel cas :

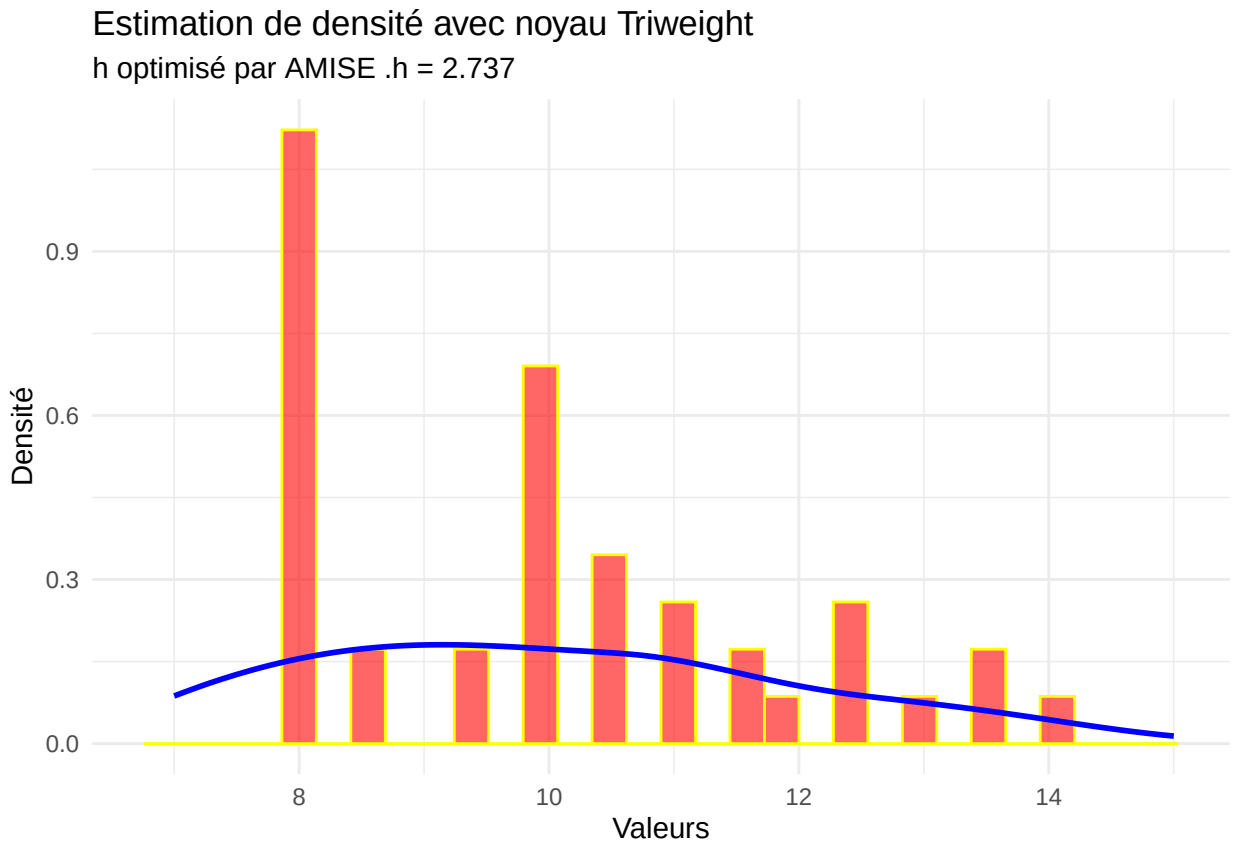
$$R(f'') \approx \frac{3}{8\sqrt{\pi}\sigma^5}$$

où sigma est l'écart-type de l'échantillon.

```
R_K <- 350 / 429
mu2_K <- 1 / 9

# Taille de l'échantillon et écart-type

n <- length(estim_notes)
sd_X <- sd(estim_notes)

# Approximation de R(f'') sous hypothèse normale

Rf2 <- 3 / (8 * sqrt(pi) * sd_X^5)


h_amise <- (R_K / (mu2_K^2 * Rf2 * n))^(1/5)
print(h_amise)
```

```
## [1] 2.737458
```

```
dens_tri <- sapply(grid, function(x0) density_triweight(x0, X = estim_notes, h = h_amise))
df <- data.frame(x = grid, dens = dens_tri)
ggplot() +
  geom_histogram(aes(x = estim_notes, y = after_stat(density)),
                 bins = 30, fill = "red", color = "yellow", alpha = 0.6) +
  geom_line(data = df, aes(x = x, y = dens),
            color = "blue", size = 1) +
  labs(
    title = "Estimation de densité avec noyau Triweight",
    subtitle = paste("h optimisé par AMISE .h =", round(h_amise, 3)),
    x = "Valeurs", y = "Densité"
  ) +
  theme_minimal()
```



Le choix de h relativement grand a pour effet de produire une courbe très lisse, qui atténue les pics de l'histogramme.

- **Estimation de h par cross-validation par maximum de vraisemblance**

Son incovénient nous le verrons c'est le temps de calcul.

Source : https://cran.opencpu.org/web/packages/kedd/vignettes/kedd.pdf , Page 9

La vraisemblance des données sous KDE est :

$$L(h) = \prod_{i=1}^{n} \hat{f}_h(X_i)$$

En prenant le logarithme, on obtient :

$$\log L(h) = \sum_{i=1}^{n} \log \hat{f}_h(X_i)$$

où $\hat{f}_h(X_i)$ est l'estimation de la densité en X_i avec le paramètre de lissage h.

- **Problème de surapprentissage**

Lorsque $h \to 0$, on a :

$$\hat{f}_h(X_i) \to \infty$$

Pour u>1, K(u) est nulle donc pour i différent de j, K((X_i - X_j)/h) ==>0 mais comme K(0)>0 alors 1/(nh)*K(0)==> + infini chaque observation contribue fortement à sa propre estimation (graphiquement on obtient des pics sur les points connus). Cela conduit à un sur-apprentissage .

Cela pousserait l'algorithme à choisir un h arbitrairement petit, conduisant à un sur-apprentissage extrême où la densité estimée serait une série de pics infinitésimaux à chaque point de donnée.

Pour éviter ce problème, on utilise la cross-validation par maximum de vraisemblance.

On estime la densité en excluant successivement chaque observation X_i.

L'estimateur de la densité en X_i excluant X_i est donné par :

$$\hat{f_{h,-i}}(X_i) = \frac{1}{(n-1)h} \sum_{j \neq i} K\left(\frac{X_i - X_j}{h}\right)$$

La log-vraisemblance devient :

$$CV(h) = \frac{1}{n} \sum_{i=1}^{n} \log \hat{f_{h,-i}}(X_i)$$

ce critère de CV converge en probabilité vers la MISE: https://www.researchgate.net/publication/280609040_Bootstrap_dans_l%27estimation_de_la_densite_par_la_methode_du_noyau

$$h_{CV} = \arg\max_{h>0} CV(h)$$

Cette méthode est non paramétrique et ne suppose aucune hypothèse forte sur la forme de la densité, tout en évitant le sur-apprentissage.

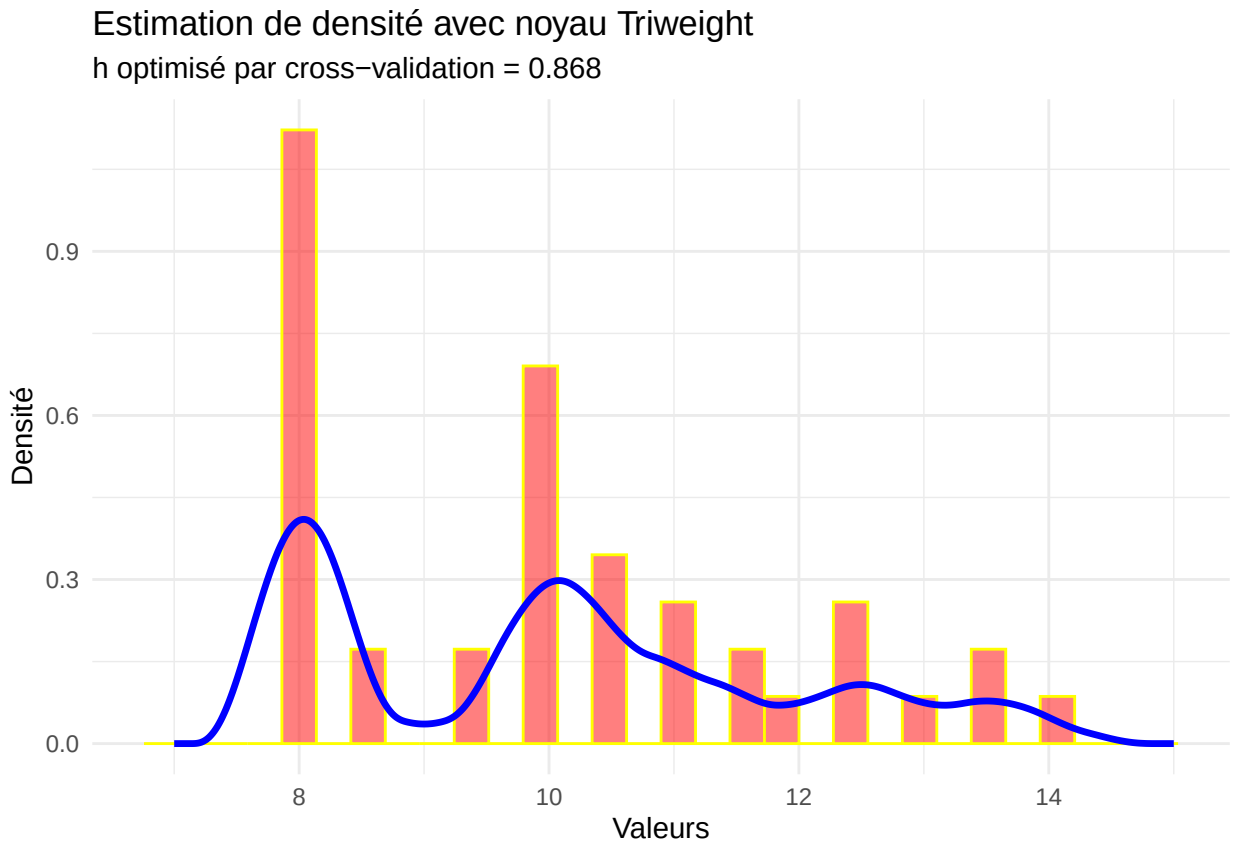
```r
# Fonction de cross-validation

cv_log_likelihood <- function(h, X) {
  n <- length(X)
  log_dens <- sapply(1:n, function(i) {
    xi <- X[i]
    x_others <- X[-i]
    u <- (xi - x_others) / h
    k_vals <- triweight_kernel(u)
    f_hat <- mean(k_vals) / h
    log(f_hat)
  })
  -mean(log_dens) # Le signe - car en pratique on prend l'oppose de CV ( On minimise)
}
```

A present pour trouver le h optimal il nous faut des candidats

```r
# Séquence de valeurs pour h à tester 100 valeurs
h_seq <- seq(0.1, 2, length.out = 100)

# Application de la fonction (output list)
cv_vals <- sapply(h_seq, function(h) cv_log_likelihood(h, estim_notes))

# Recherche de la valeur optimale de h

h_cv <- h_seq[which.min(cv_vals)]
cat("h optimal est:", round(h_cv, 4), "\n")
```

```
## h optimal est: 0.8677
```

```r
# Densité estimée avec h_cv
dens_cv <- sapply(grid, function(x0) density_triweight(x0, X = estim_notes, h = h_cv))
df_cv <- data.frame(x = grid, dens = dens_cv)

ggplot() +
  geom_histogram(aes(x = estim_notes, y = after_stat(density)),
                 bins = 30, fill = "red", color = "yellow", alpha = 0.5) +
  geom_line(data = df_cv, aes(x = x, y = dens),
            color = "blue", size = 1.2) +
  labs(
    title = "Estimation de densité avec noyau Triweight",
    subtitle = paste("h optimisé par cross-validation =", round(h_cv, 3)),
    x = "Valeurs", y = "Densité"
  ) +
  theme_minimal()
```



Lien avec la MISE : Il a été montré que la minimisation de ce critère de validation croisée est asymptotiquement équivalente à la minimisation de la MISE.

5.3.4 Noyau du cosinus

Le noyau cosinus est donné par :

$$K(t) = \frac{\pi}{4} \cos\left(\frac{\pi}{2} t\right) \mathbf{1}_{[-1,1]}(t)$$

Le code R pour la fonction est le suivant :

```
K <- function(x)
  {
  K <- 0
  if (-1 <= x && x <= 1) {
    K <- (pi/4)*cos(x*pi/2)
  }
  return (K)
}
```

- **Test**

```
print(K(0))
```

```
## [1] 0.7853982
```

- **L'estimateur de densité**

L'estimateur de densité moyennant le noyau cosinus est donné par :

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} \frac{\pi}{4} \cos\left(\frac{\pi}{2} \frac{x - x_i}{h}\right) \mathbf{1}_{[-1,1]}\left(\frac{x - x_i}{h}\right)$$

Le code R est le suivant :

```
fn <- function(data,x,nb_cla){
  n <- length(data)
  h <- (max(data)-min(data))/nb_cla
  fn <- 0
  for (i in 1:n){
    fn <- fn + K((x-data[i])/h)
  }
  fn <- fn/(n*h)
  return (fn)
}
```

- **Test-**

```
data <- c(1.1,2.3,1.7,2.8,3.2,1.9,2.5,3.7,2.9,2.1)

# Test de la fonction au point x = 1.1
print(fn(data = data,x = 1.1,nb_cla = 4))
```

```
## [1] 0.135395
```

- **Application sur des données réelles de l'EHCVM**

Nous allons maintenant donner une application concrète sur les données EHCVM.

```
ehcvm_welfare_sen2018 <- read_dta("BASES/ehcvm_welfare_sen2018.dta")
head(ehcvm_welfare_sen2018)
```

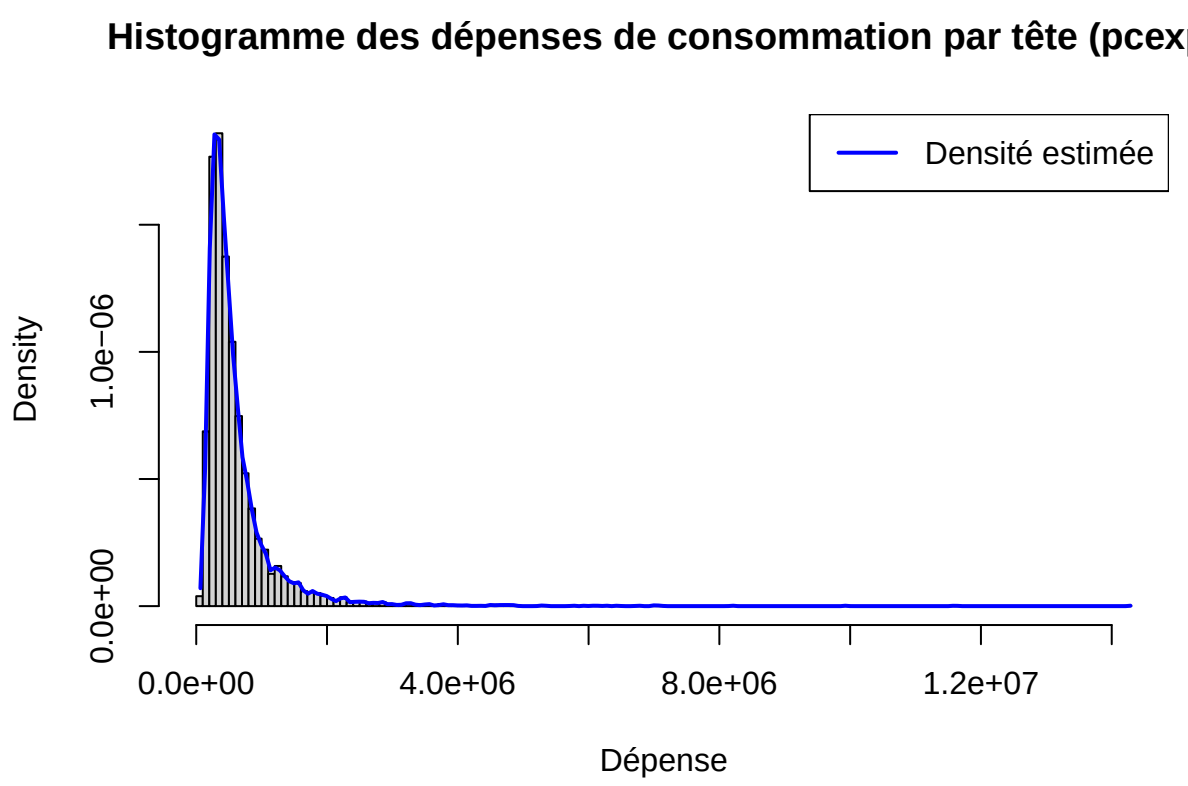
```
## # A tibble: 6 x 35
##   country  year  hhid grappe menage vague   zae region   milieu  hhweight hhsize
##   <chr>   <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl+lb> <dbl+l>    <dbl>  <dbl>
## 1 SEN      2018  1001      1      1     1     1 1 [daka~ 1 [Urb~    1750.      2
## 2 SEN      2018  1002      1      2     1     1 1 [daka~ 1 [Urb~    1750.      2
## 3 SEN      2018  1003      1      3     1     1 1 [daka~ 1 [Urb~    1750.      1
## 4 SEN      2018  2001      2      1     2     1 1 [daka~ 1 [Urb~     266.     10
## 5 SEN      2018  2002      2      2     2     1 1 [daka~ 1 [Urb~     266.      6
## 6 SEN      2018  2003      2      3     2     1 1 [daka~ 1 [Urb~     266.      4
## # i 24 more variables: eqadu1 <dbl>, eqadu2 <dbl>, hgender <dbl+lbl>,
```

```
## #   hage <dbl>, hmstat <dbl+lbl>, hreligion <dbl+lbl>, hnation <dbl+lbl>,
## #   halfab <dbl+lbl>, heduc <dbl+lbl>, hdiploma <dbl+lbl>, hhandig <dbl+lbl>,
## #   hactiv7j <dbl+lbl>, hactiv12m <dbl+lbl>, hbranch <dbl+lbl>,
## #   hsectins <dbl+lbl>, hcsp <dbl+lbl>, dali <dbl>, dnal <dbl>, dtot <dbl>,
## #   pcexp <dbl>, zzae <dbl>, zref <dbl>, def_spa <dbl>, def_temp <dbl>
```

```
data <- as.vector(ehcvm_welfare_sen2018$pcexp)
```

- **Représentation de l'histogramme des données**

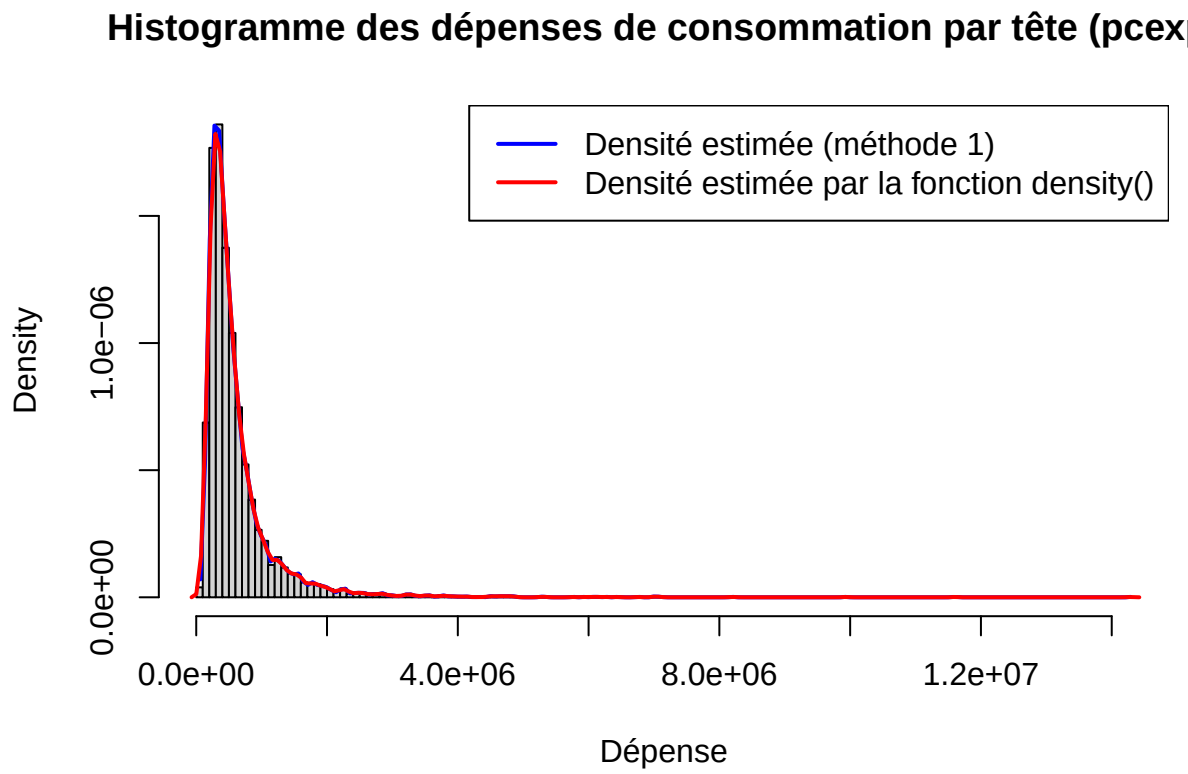
```
hist(data, breaks = 200,freq = FALSE, main = "Histogramme des dépenses de consommation par tê

x_vals <- seq(min(data), max(data), length.out = 200)
f_vals <- sapply(x_vals, function(x) fn(data, x , nb_cla = 200))
lines(x_vals, f_vals, col = "blue", lwd = 2)
legend("topright", legend = c("Densité estimée"), col = c("blue"), lwd = 2, lty = c(1, 2))
```



- **Estimation de la densité avec la fonction density()**

Nous utilisons maintenant la fonction prédéfinie dans R pour comparer à la notre.

```r
# Créer l'histogramme de base
hist(data, breaks = 200, freq = FALSE,
     main = "Histogramme des dépenses de consommation par tête (pcexp)",
     xlab = "Dépense", col = "lightgray")

# Première courbe
x_vals <- seq(min(data), max(data), length.out = 200)
f_vals <- sapply(x_vals, function(x) fn(data, x, nb_cla = 200))
lines(x_vals, f_vals, col = "blue", lwd = 2)

# Deuxième courbe
dens <- density(data, kernel = "cosine", n = 200)
lines(dens$x, dens$y, col = "red", lwd = 2)

# Légende
legend("topright",
       legend = c("Densité estimée (méthode 1)", "Densité estimée par la fonction density()"),
       col = c("blue", "red"), lwd = 2, lty = 1)
```



On constate qu'on a pratiquement les mêmes résultats.

- **Cas de la mesure de pauvreté**

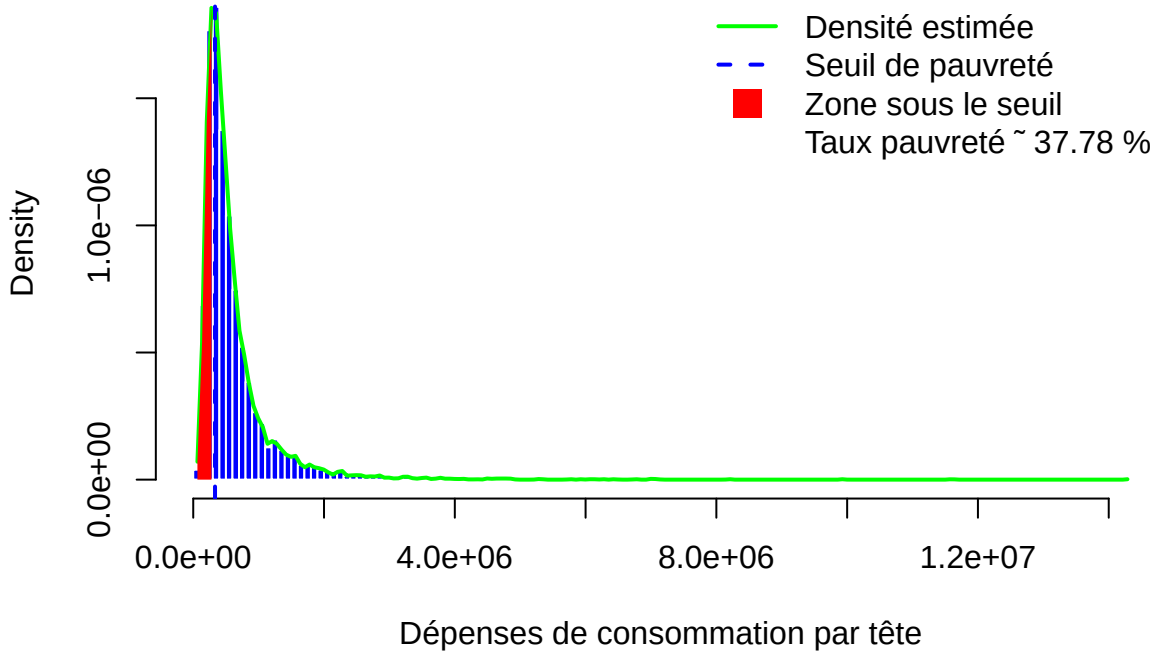
L'objectif ici est d'approximer la densité de la loi des dépenses de consommation par tête (variable pcexp) et de déterminer le taux de pauvreté.

- **Taux de pauvreté (aire sous la densité à gauche du seuil z)**

```
z <- mean(ehcvm_welfare_sen2018$zref)
dens_vals <- sapply(data, function(xi) as.numeric(xi < z))
```

- **Construction**

```
hist(data, breaks = 200, freq = FALSE,
     main = "Analyse de la pauvreté",
     xlab = "Dépenses de consommation par tête",
     col = "blue", border = "white")

# Tracer la densité
lines(x_vals, f_vals, col = "green", lwd = 2)


# Colorier l'aire sous la courbe avant le seuil z
polygon_x <- x_vals[x_vals <= z]
polygon_y <- f_vals[x_vals <= z]
polygon(c(polygon_x, rev(polygon_x)), c(rep(0, length(polygon_x)), rev(polygon_y)),
        col = "red", border = NA)

# Ligne verticale pour le seuil
abline(v = z, col = "blue", lwd = 2, lty = 2)

# Calcul du taux de pauvreté
z <- mean(ehcvm_welfare_sen2018$zref)
poid=ehcvm_welfare_sen2018$hhweight*ehcvm_welfare_sen2018$hhsize/sum(ehcvm_welfare_sen2018$hh
# Calcul du taux de pauvreté
taux_pauvrete <- sum(poid[data < z])

legend("topright",
       legend = c("Densité estimée","Seuil de pauvreté","Zone sous le seuil", paste("Taux pau
       col = c("green","blue","red", NA),
       lty = c(1,2, NA, NA), lwd = c(2,2, NA, NA), pch = c(NA,NA, 15, NA), pt.cex = 2, bty =
```



- **Affichage en console**

```
cat("Taux de pauvreté estimé :", round(taux_pauvrete * 100, 2), "%\n")
```

```
## Taux de pauvreté estimé : 37.78 %
```

5.3.5 Noyau d'Épanchnikov

La fonction du noyau d'Épanchnikov est définie comme suit :

$$K(u) = \begin{cases} \frac{3}{4}(1-u^2) & \text{si } |u| \leq 1 \\ 0 & \text{sinon} \end{cases}$$

```
epanechnikov_kernel <- function(u) {
  k <- 0.75 * (1 - u^2) * (abs(u) <= 1)
  return(k)
}
```

- **Fonction d'estimation de noyau**

L'estimateur de la densité à noyau basé sur un échantillon $X_1, X_2, \ldots, X_n$ est défini par :

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

- **Estimateur avec le noyau d'Épanchnikov**

En remplaçant $K(u)$ par le noyau d'Épanchnikov, on obtient :

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} \left[\frac{3}{4} \left(1 - \left(\frac{x - X_i}{h} \right)^2 \right) \cdot \mathbf{1}_{|\frac{x-X_i}{h}| \leq 1} \right]$$

- **Optimisation de h**

Largeur de bande optimale h^* :

En minimisant l'AMISE (erreur quadratique intégrée moyenne asymptotique), on obtient la **largeur de bande optimale** :

$$h^* = \left(\frac{R(K)}{\mu_2^2(K) \cdot R(f'') \cdot n} \right)^{1/5}$$

Pour le noyau d'Épanchnikov :

$$R(K) = \int_{-1}^{1} K^2(u)\, du = \frac{3}{5}, \quad \mu_2(K) = \int_{-1}^{1} u^2 K(u)\, du = \frac{1}{5}$$

En remplaçant dans la formule de h^*, on obtient :

$$h^* = \left(\frac{\frac{3}{5}}{\left(\frac{1}{5}\right)^2 \cdot R(f'') \cdot n} \right)^{1/5} = \left(\frac{3}{5} \cdot \frac{25}{1} \cdot \frac{1}{R(f'') \cdot n} \right)^{1/5} = \left(\frac{15}{R(f'') \cdot n} \right)^{1/5}$$

Nous ne connaissons pas la valeur de R(f' '). La règle de Silverman fournit une approximation pratique de la largeur de bande h pour l'estimateur de densité à noyau, lorsque la densité f est supposée proche d'une loi normale.

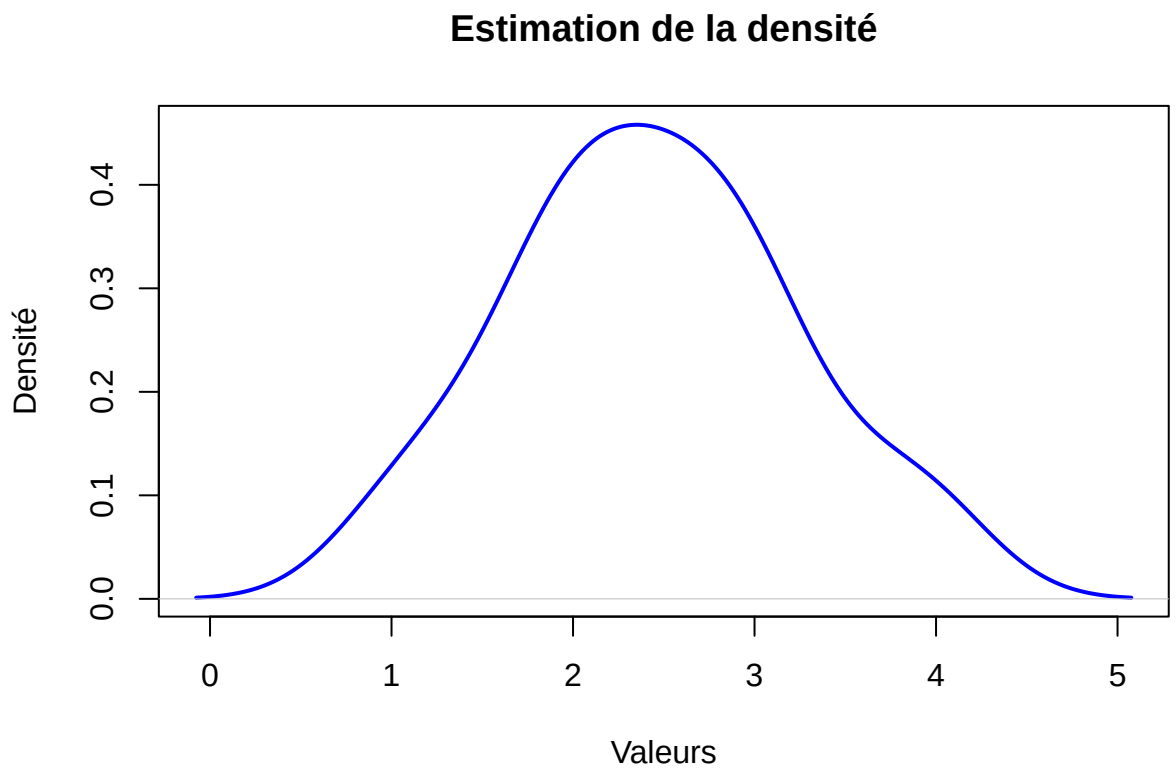
La formule est donnée par :

$$h_{\text{Silverman}} = 0{,}9 \cdot \min\left(\sigma, \frac{\text{IQR}}{1{,}34} \right) \cdot n^{-1/5}$$

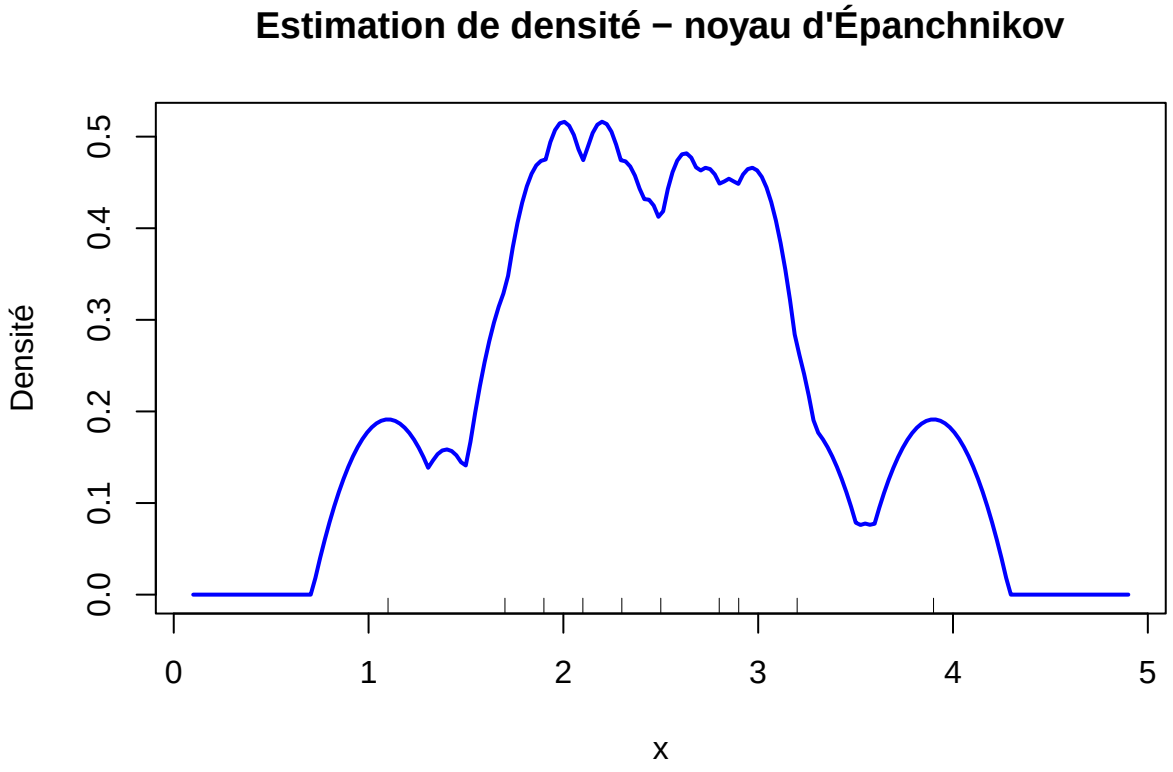
```r
h_optimal <- function(data) {
  n <- length(data)
  sigma <- sd(data)
  iqr <- IQR(data)
  s <- min(sigma, iqr / 1.34)
  h <- 0.9 * s * n^(-1/5)
  return(h)
}
```

- **Pratique**

```r
data <- c(2.1, 2.3, 1.9, 2.5, 1.7, 2.8, 1.1, 3.2, 3.9, 2.9)
# Estimation de la densité
dens <- density(data)

# Tracé de la densité
plot(dens, main = "Estimation de la densité", xlab = "Valeurs", ylab = "Densité", col = "blue
```




```r
# Génération de données simulées


# Points où on évalue la densité
x <- seq(min(data) - 1, max(data) + 1, length.out = 200)

# Largeur de bande
h <- h_optimal(data)

# Calcul de la densité
density_values <- epanechnikov_density(x, data, h)

# Affichage du résultat
plot(x, density_values, type = "l", lwd = 2, col = "blue",
     main = "Estimation de densité - noyau d'Épanchnikov",
     xlab = "x", ylab = "Densité")
rug(data)  # Ajoute les observations sur l'axe x
```



5.3.6 Noyau Biweight

Partie théorique

- **Échantillon utilisé**

$$X = (1.1,\ 2.3,\ 1.7,\ 2.8,\ 3.2,\ 1.9,\ 2.5,\ 3.7,\ 2.9,\ 2.1)$$

- **Noyau Biweight**

$$K(x) = \begin{cases} \frac{15}{16} \times (1 - x^2)^2 & \text{si } |x| < 1 \\ 0 & \text{sinon} \end{cases}$$

- **Estimateur de la densité par noyau Biweight**

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

où h est la fenêtre de lissage.

Application

- **Données**

```r
### Chargement des données
X <- c(1.1, 2.3, 1.7, 2.8, 3.2, 1.9, 2.5, 3.7, 2.9, 2.1)
```

- **Définition du noyau Biweight**

```r
# Fonction noyau Biweight
biweight_kernel <- function(x) {
  ifelse(abs(x) < 1, (15/16) * (1 - x^2)^2, 0)
}
```

- **Estimation de la densité**

```r
# Fonction d'estimation de la densité
density_estimate <- function(x, sample, h) {
  n <- length(sample)
  sum <- 0
  for (i in 1:n) {
    sum <- sum + biweight_kernel((x - sample[i]) / h)
  }
  return(sum / (n * h))
}
```

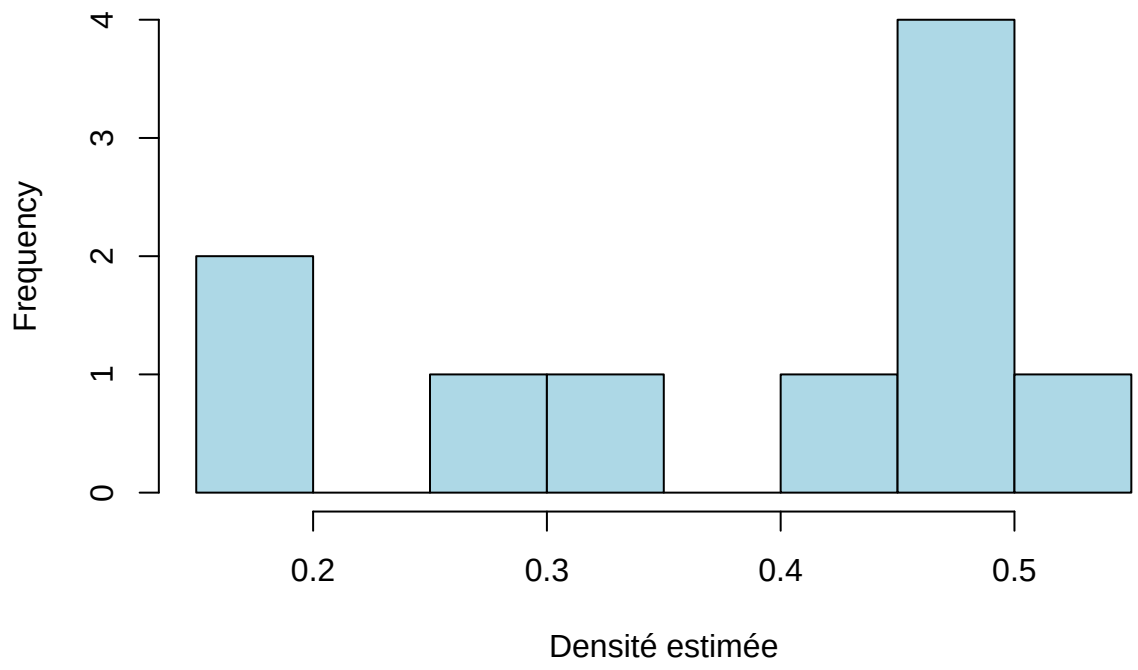
- **Calcul des densités estimées**

```r
# Définition de la fenêtre de lissage
h <- 0.5

# Calcul des densités estimées pour chaque observation
density_estimates <- sapply(X, density_estimate, sample = X, h = h)

# Affichage des densités estimées
density_estimates
```

```
##  [1] 0.1875 0.4764 0.3441 0.4614 0.2886 0.4764 0.4452 0.1875 0.4614 0.5007
```

- **Visualisation : Histogramme des densités estimées**

```r
hist(density_estimates,
     main = "Histogramme des densités estimées",
     xlab = "Densité estimée",
     col = "lightblue",
     border = "black")
```



Utilisation du test de Wald pour tester l'égalité de la densité estimée avec la densité uniforme

- **Hypothèse du test**

$$\begin{cases} H_0 : \text{La densité de } X \text{ est uniforme} \\ H_1 : \text{La densité de } X \text{ n'est pas uniforme} \end{cases}$$

- **Région de rejet**

Pour un test bilatéral de Wald au seuil $\alpha = 5\%$, les valeurs critiques c_1 et c_2 sont déterminées telles que :

$$P\left(\hat{f}_n(x) < c_1\right) = \frac{\alpha}{2} \quad \text{et} \quad P\left(\hat{f}_n(x) > c_2\right) = \frac{\alpha}{2}$$

- **Staistique du test**

$$W = \frac{(\hat{f}(x) - f_0(x))^2}{\text{Var}(\hat{f}(x))}$$

- **Calcul des statistiques : Moyenne et Variance**

```r
# Calcul de la moyenne
mean_density <- mean(density_estimates)

# Calcul de la variance
var_density <- var(density_estimates)

# Affichage des résultats
cat("Moyenne des densités estimées :", mean_density, "\n")
```

```
## Moyenne des densités estimées : 0.38292
```

```r
cat("Variance des densités estimées :", var_density, "\n")
```

```
## Variance des densités estimées : 0.01492526
```

- **Test statistique et décision**

```r
# Calcul de la statistique de test
test_statistic <- (mean_density - 1) / sqrt(var_density / length(density_estimates))

# Calcul de la valeur critique pour un test bilatéral à 5%
alpha <- 0.05
critical_value <- qnorm(1 - alpha / 2)

# Affichage de la statistique de test et de la valeur critique
cat("Statistique de test :", test_statistic, "\n")
```

```
## Statistique de test : -15.97278
```

```r
cat("Valeur critique :", critical_value, "\n")
```

```
## Valeur critique : 1.959964
```

```r
# Prise de décision
if (abs(test_statistic) > critical_value) {
  cat("Nous rejetons l'hypothèse nulle (H0).\n")
} else {
  cat("Nous ne rejetons pas l'hypothèse nulle (H0).\n")
}
```

```
## Nous rejetons l'hypothèse nulle (H0).
```