

# Big Data

## TD/TP 5 : Théorie et Pratique

### BUT 3

Guillaume Metzler et Antoine Rolland  
Institut de Communication (ICOM)  
Université de Lyon, Université Lumière Lyon 2  
Laboratoire ERIC UR 3083, Lyon, France  
[guillaume.metzler@univ-lyon2.fr](mailto:guillaume.metzler@univ-lyon2.fr); [antoine.rolland@univ-lyon2.fr](mailto:antoine.rolland@univ-lyon2.fr)

## Exercice 1 : Volumétrie et tests statistiques

Dans cet exercice, on se propose d'étudier, de façon empirique, la souplesse des tests statistiques dans un contexte où nous disposons d'une grande volumétrie de données. Dans la cas présent, d'un grand nombre d'exemples.

Plus précisément, nous cherchons à étudier un test la significativité d'un test de comparaison de moyennes sur deux échantillons :

- un premier échantillon de taille  $n$  qui suit une loi normale centrée et réduite.
- un deuxième échantillon suivant une loi normale de moyenne  $\mu = 0.005$  et de variance égale à 1.

On va maintenant étudier ces deux échantillons.

1. Quel test statistique doit-on effectuer pour comparer les moyennes de ces deux échantillons ? Pourquoi ? Quelle est la distribution de probabilités associée à ce test ?


On doit effectuer un test de comparaison de moyenne pour des échantillons **indépendants** où les variances sont égales. En effet, il n'y a pas de dépendance dans la définition de ces deux échantillons et les variances de ces deux derniers sont supposées égales.

Notre test s'écrit donc

$$H_0 : \mu_1 = \mu_2 \quad \text{v.s.} \quad \mu_1 \neq \mu_2$$

et il repose sur la loi de Student.

2. On cherche maintenant à étudier le nombre de fois où le test de comparaison de ces deux moyennes est significatif, en fonction de la taille  $n$  de l'échantillon. Pour cela, vous devrez :

- (a) simuler un échantillon de taille  $n$  dont les données  $x_i \underset{i.i.d.}{\sim} \mathcal{N}(0, 1)$
  - (b) simuler un échantillon de taille  $n$  dont les données  $x_i \underset{i.i.d.}{\sim} \mathcal{N}(0.005, 1)$
  - (c) Pour une valeur donnée de  $n$ , effectuer le test de comparaisons de moyennes sous  et déterminer si le test est significatif ou non avec un seuil  $\alpha = 0.05$ .
3. Que représente la valeur  $\alpha$  dans la question précédente ?

Cette valeur représente le risque de première espèce  $\alpha$ , *i.e.* le risque que nous sommes prêt à prendre en rejetant  $H_0$  à tort.

4. Répéter les questions 2 cent fois pour une valeur de  $n$  fixée et regarder, en pourcentage, le nombre de fois où le test est significatif.
5. Répéter la question 3 et regarder comment évolue ce pourcentage en fonction de la valeur de  $n$ .

```
# Installation du package pour accéder à la fonction "manipulate"

install.packages("manipulate")
library(manipulate)

manipulate::manipulate({

  count = 0
  for (i in 1:100){
    x1 <- rnorm(n, 0.005, 1)
    x2 <- rnorm(n)
    test = t.test(x1, x2, var.equal = TRUE)
    if (test$p.value < 0.05){
      count = count + 1
    }
  }
})

p1 <- hist(x1, probability = TRUE, col = rgb(0, 0, 1, 1/4), ylim = c(0, 0.5), xlim = c(-3, 3),
  xlab = "", main = "Histogrammes",
  sub = paste("Taux de significativité du test : ", count/100))
lines(density(x1), col = rgb(0, 0, 1, 1/2), lwd = 3)
p2 <- hist(x2, probability = TRUE, col = rgb(1, 0, 0, 1/4), add = TRUE, ylim = c(0, 0.5),
  xlim = c(-3, 3), xlab = "")
lines(density(x2), col = rgb(1, 0, 1, 1/2), lwd = 3)
}, n = manipulate::slider(min = 1000, max = 500000, initial = 1000, step = 1000))
```

## Exercice 2 : Imputations valeurs manquantes

On se propose d'étudier l'impact sur la moyenne de différentes méthodes de complétions des valeurs manquantes à l'aide de quatre méthodes distinctes :

- imputation par la moyenne
- imputation par la valeur 0

- 
- imputation par l'algorithme  $k$ -NN (les  $k$  plus proches voisins)

Pour cela, on va considérer le jeu de données suivant :

```
# On fixe la graine

set.seed(1)

# On génère un jeu de données à plusieurs dimensions,
# mais avec un nombre d'exemples limité.

p = 2
n = 500

X = matrix(rnorm(n*p,3,2),ncol = 2)
Y_full = -1 + X[,1] - 2*X[,2]

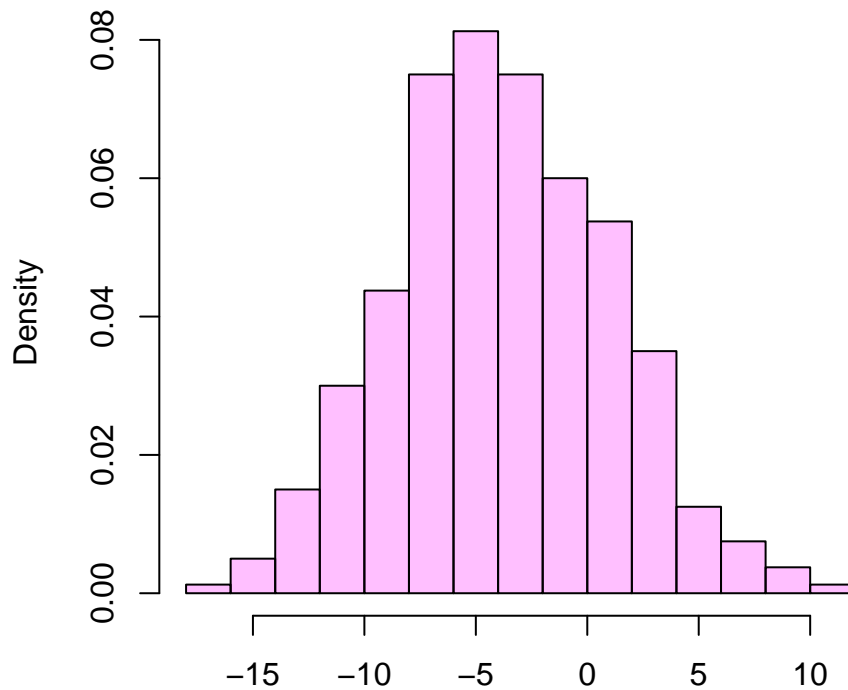
# On va supprimer les 100 premières valeurs de ce vecteur

Y_miss= Y_full
Y_miss[1:100] = NA

# On peut étudier l'histogramme des valeurs de Y

hist(Y_miss,probability = TRUE, col=rgb(1,0,1,1/4),
xlab="", main = "Histogrammes")
```

## Histogrammes



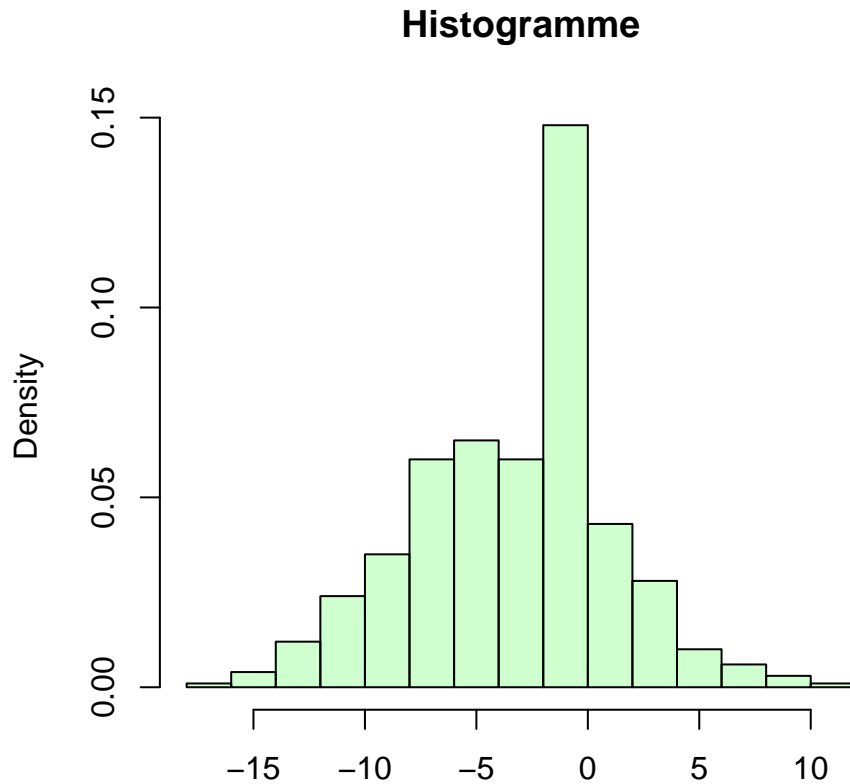
On représentera et commentera l'histogramme des valeurs du vecteur  $Y$  complété par les méthodes suivantes :

1. en faisant une imputation par la valeur 0

```
# Imputation par la valeur 0

Y_comp_zero = Y_miss
Y_comp_zero[is.na(Y_comp_zero)]=0

hist(Y_comp_zero,probability = TRUE, col=rgb(1/4,1,1/4,1/4),
xlab="", main = "Histogramme")
```



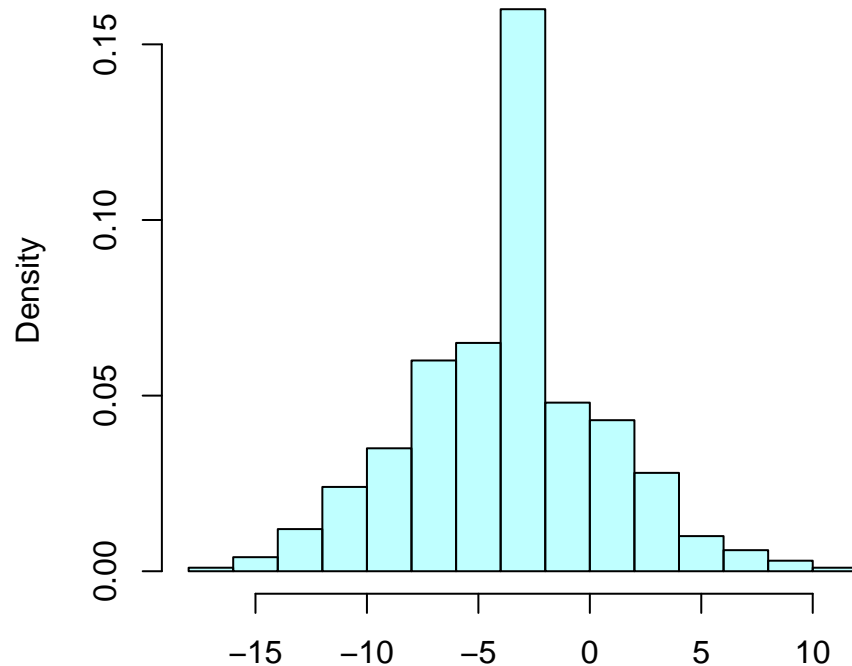
2. en effectuant une imputation par la moyenne des valeurs de  $Y$  observées,

```
# Imputation par la valeur moyenne des observations

Y_comp_mean = Y_miss
Y_comp_mean[is.na(Y_comp_mean)] = mean(Y_miss, na.rm = TRUE)

hist(Y_comp_mean, probability = TRUE, col = rgb(0, 1, 1, 1/4),
     xlab = "", main = "Histogramme")
```

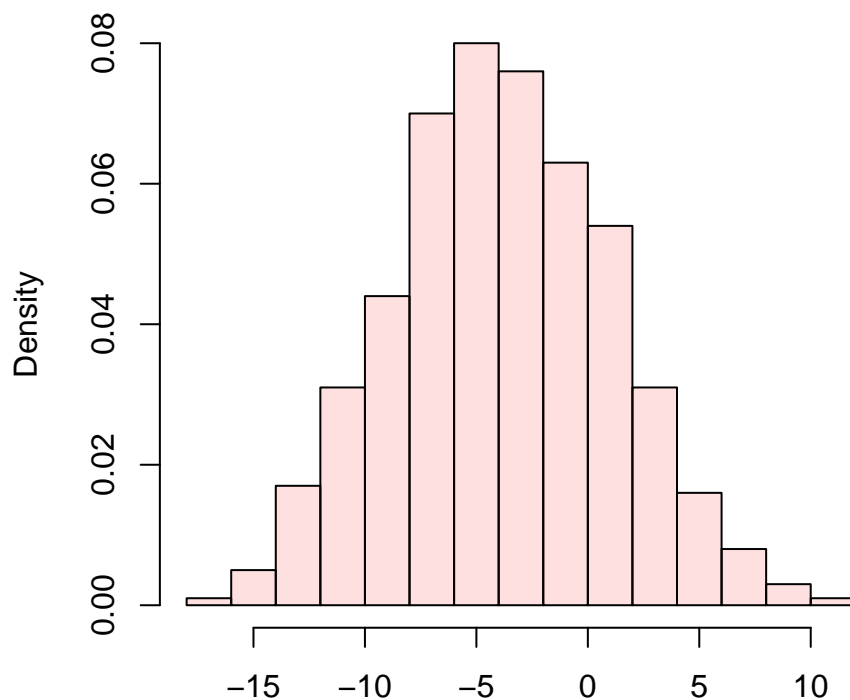
## Histogramme



3. en complétant avec une valeur au hasard tirée suivant une loi normale de moyenne la moyenne empirique observée et d'écart-type l'écart-type observé),

```
# Imputation par échantillonnage.  
  
moy<-mean(Y_miss, na.rm=T)  
ec<-sd(Y_miss, na.rm=T)  
Y_imput_norm<-Y_miss  
Y_imput_norm[is.na(Y_miss)]<-rnorm(100, mean=moy, sd=ec)  
  
hist(Y_imput_norm,probability = TRUE, col=rgb(1,0,0,1/8),  
      xlab="", main = "Histogrammes")
```

## Histogrammes



4. en complétant les valeurs manquantes à l'aide d'un algorithme du plus proche voisin, *i.e.* en prenant la valeur  $k = 1$  (on pourrait aussi tester des valeurs de  $k$  différentes si on le souhaite),

```
# Imputation par la méthode du plus proche voisin.

Y_comp_knn = Y_miss

my_knn_comp = function(k = 1, X_miss, X_other, Y_other){
  # Calcul de la distance et stockage des résultats
  dist <- matrix(0, nrow = nrow(X_miss), ncol = nrow(X_other))
  for (i in 1:nrow(X_miss)){
    for (j in 1:nrow(X_other)){
      dist[i,j] <- sum((X_miss[i,] - X_other[j,])^2)
    }
  }

  # On ordonne maintenant les distances par ordre croissant, mais ce qui va
  # nous intéresser, ce sont les indices pour récupérer les valeurs associées de Y.

  nearest_neighbor <-
  t(apply (dist ,1, function(x) {sort(x,index.return = TRUE)$ix}))
  val_neighbor <- t(apply (nearest_neighbor ,1, function(x) {Y_other[x]}))

  # On a récupéré une matrice avec les labels des plus proches voisins
  # par distances croissantes. Il reste à faire la classification,
  # à l'aide de la valeur de $k$
```

```

predicted_value <- apply(val_neighbor, 1, function(x) {x[1:k]})
return(predicted_value)
}

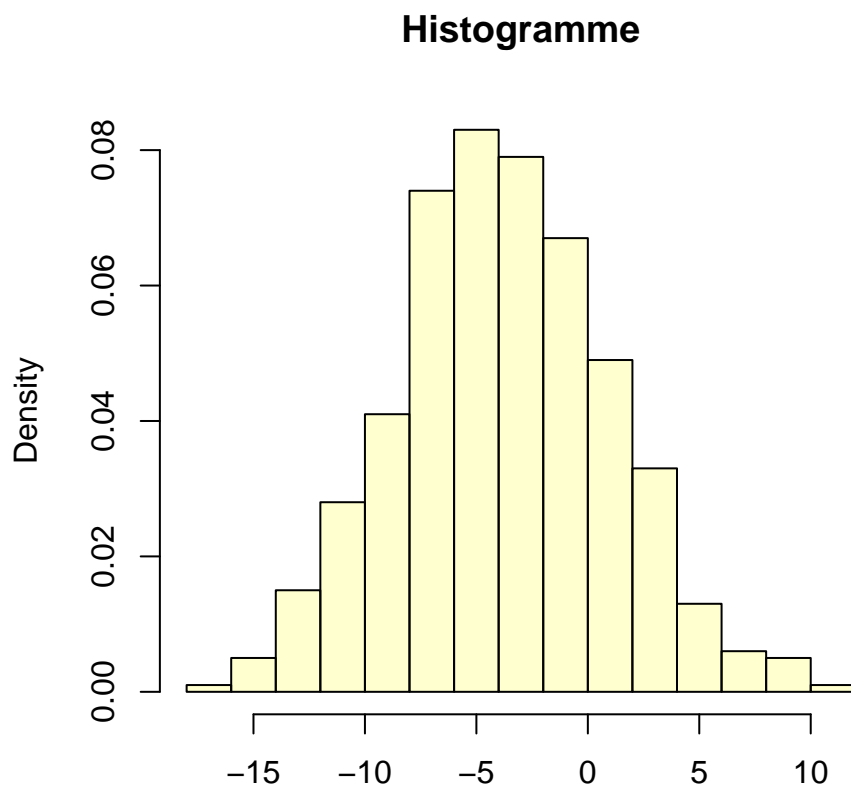
# On effectue la complétion


X_miss = X[1:100,]
X_other = X[101:500,]
Y_other = Y_full[101:500]

Y_comp_knn[is.na(Y_comp_knn)] =
my_knn_comp(k = 1, X_miss, X_other, Y_other)

hist(Y_comp_knn, probability = TRUE, col=rgb(1,1,1/4,1/4),
xlab="", main = "Histogramme")

```



Nous pourrions également prendre une valeur de  $k$  plus grande que 1, il faudrait modifier la fonction de façon à prendre la moyenne des valeurs de  $Y$  au lieu que de prendre la valeur associée au plus proche voisin, dans la définition de la variable `val_neighbor`. Nous aurions également plus le faire avec le package **CLASS** de .

```

library(class)
Y_imput_knn<-Y_miss

entrainement<-X[!is.na(Y_miss),]

```



```
test<-X[is.na(Y_miss),]  
valeurs_entrainement<-as.factor(Y_miss[!is.na(Y_miss)])  
valeurs_a_predire<-knn(entrainement, test, valeurs_entrainement, k=3)  
  
Y_imput_knn[is.na(Y_miss)]<-as.numeric(as.character(valeurs_a_predire))
```