

# TD3-Kougoum-Kouadio

Nom, Prenom (1) : KOUADIO Cheryl

Nom, Prenom (2) : KOUGOUM MOKO MANI Marilène

## Spectral clustering

### 1. Génération des données selon une loi de mélange gaussienne.

```
rm(list=ls())
# Génération des données
library(tidyverse)
library(mvtnorm)
library(ggplot2)
set.seed(123)

# générer données selon distrib multinomiale
# n = nbre d'échantillon
# size = taille d'échantillon
# prob = proba pour chaque groupe

p1 <- 0.3
p2 <- 0.6
p3 <- 0.1

n <- 500
Z <- sample(c(1,2,3),size=n,prob=c(p1,p2,p3),replace=TRUE)
Z<-factor(Z)
Z <- sort(Z)

n1 <- sum(Z==1)
n2 <- sum(Z==2)
n3 <- sum(Z==3)

# Génération de X/Z=z selon une loi normale

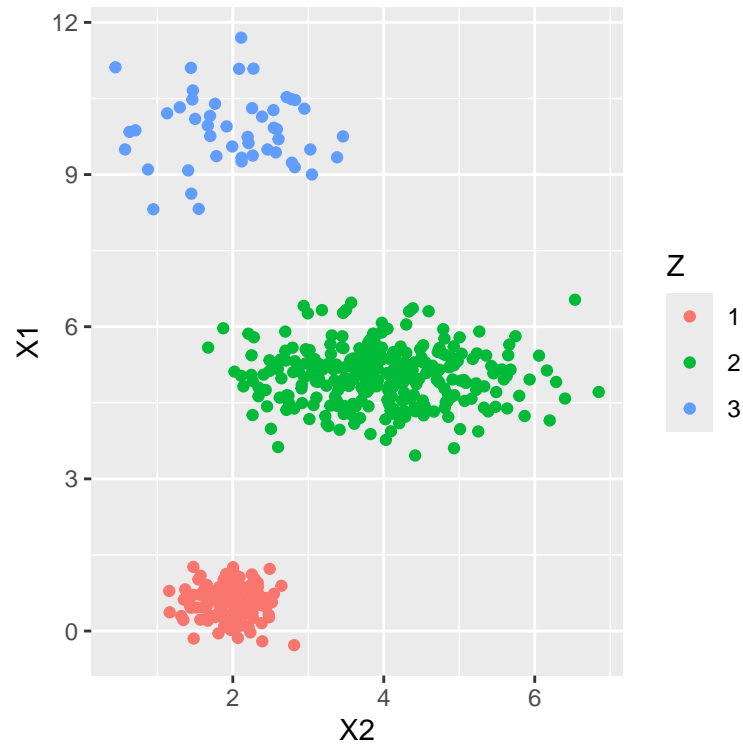
# Lois des 3 groupes
# Lois des 3 groupes
mu1 <- c(0.5,2)
mu2 <- c(5,4)
mu3 <- c(10,2)

sigma1 <- diag(c(0.1,0.1))
sigma2 <- diag(c(0.3,0.8))
sigma3 <- diag(c(0.6, 0.5))
```

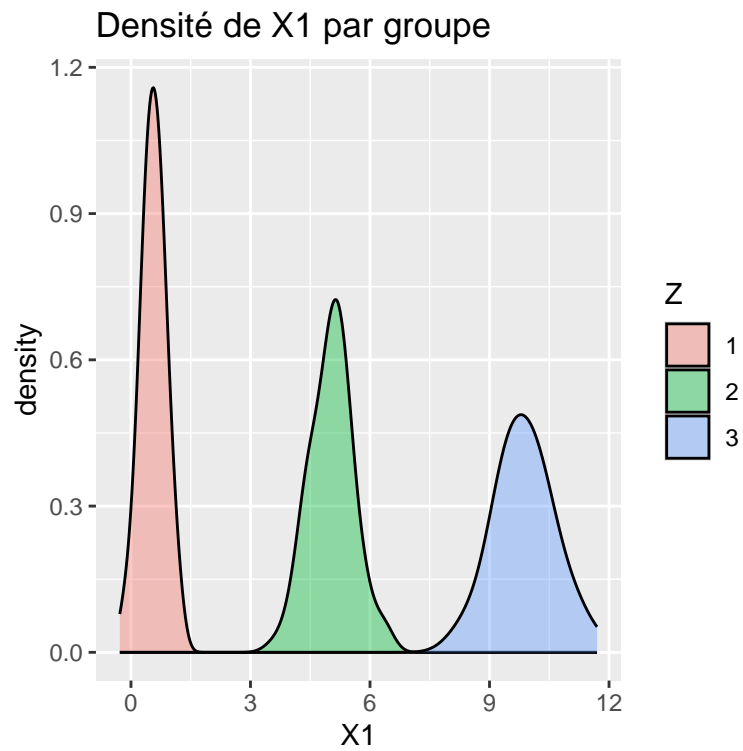
```
X <- rbind(rmvnorm(n1, mean=mu1, sigma=sigma1),
           rmvnorm(n2, mean=mu2, sigma=sigma2),
           rmvnorm(n3, mean=mu3, sigma=sigma3))
X <- data.frame(X)

data <- data.frame(X1=X$X1, X2=X$X2, Z=Z)

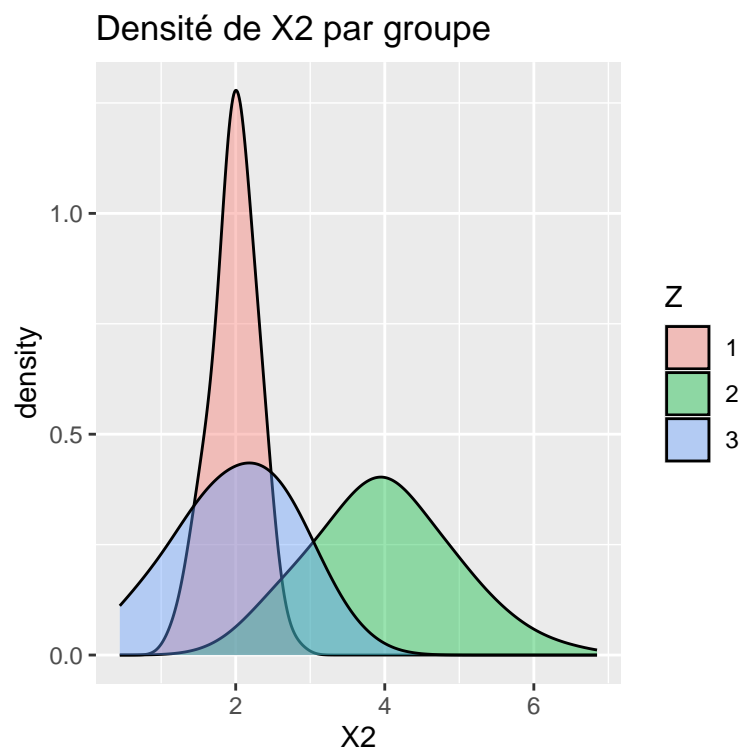
# Représentation bivariable des données
ggplot(data=data, aes(x=X2, y=X1, color=Z, fill=Z)) +
  geom_point()
```



```
# Représentation univariée des données
ggplot(data=data, aes(x=X1, group=Z, fill=Z)) +
  geom_density(adjust=1.5, alpha=.4)+labs(title="Densité de X1 par groupe")
```



```
ggplot(data=data, aes(x=X2, group=Z, fill=Z)) +  
  geom_density(adjust=1.5, alpha=.4) + labs(title="Densité de X2 par groupe")
```



## 2. Choix du critère de similarité : Similarité gaussienne

Le critère de similarité gaussien est :

$$s(X_i, X_j) = \exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right).$$

```
gaussian_similarity <- function(xi,xj,sigma){
  sim <- exp(- (norm(as.matrix(xi)-as.matrix(xj),type="2")^2) / (2*sigma^2))
  return(sim)
}

sigmas <- c(1/3, 2/3, 1)
```

## 3. Création de la matrice d'adjacence

La matrice d'adjacence sert à représenter les similarités entre les points de données. Chaque élément de la matrice indique la similarité entre deux points de données. Elle est définie comme suit :

$$W_{ij} = s(X_i, X_j).$$

```
adjacency_matrix <- function(X, sim_function, sigma){
  n <- nrow(X)
  W <- matrix(0, n, n)
  for(i in 1:n){
    for(j in 1:n){
      W[i,j] <- sim_function(X[i,], X[j,], sigma)
    }
  }
  return(W)
}

W1 <- adjacency_matrix(X, gaussian_similarity, sigma=sigmas[1])
W2 <- adjacency_matrix(X, gaussian_similarity, sigma=sigmas[2])
W3 <- adjacency_matrix(X, gaussian_similarity, sigma=sigmas[3])
```

## 4. Création de la matrice des degrés

La matrice des degrés est une matrice diagonale où chaque élément diagonal représente le degré d'un nœud dans le graphe. Le degré d'un nœud est la somme des poids des arêtes connectées à ce nœud. Elle est définie comme suit :

$$D_{ii} = \sum_j W_{ij}.$$

```
degree_matrix <- function(W){
  D <- diag(rowSums(W))
  return(D)
}

D1 <- degree_matrix(W1)
D2 <- degree_matrix(W2)
D3 <- degree_matrix(W3)
```

## 5. Création de la matrice laplacienne

### a. Matrice laplacienne non normalisée

La matrice laplacienne non normalisée est définie comme la différence entre la matrice des degrés et la matrice d'adjacence. Elle est utilisée pour capturer la structure du graphe. Elle est définie comme suit :

$$L = D - W.$$

```
unnormalized_laplacian_matrix <- function(D, W){  
  L <- D - W  
  return(L)  
}  
  
L1_unnormalized <- unnormalized_laplacian_matrix(D1, W1)  
L2_unnormalized <- unnormalized_laplacian_matrix(D2, W2)  
L3_unnormalized <- unnormalized_laplacian_matrix(D3, W3)
```

### b. Matrice laplacienne normalisée symétrique

La matrice laplacienne normalisée symétrique est une version normalisée de la matrice laplacienne non normalisée. Elle est définie comme suit :

$$L_{sym} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}.$$

```
sym_normalized_laplacian_matrix <- function(D, W){  
  
  L <- unnormalized_laplacian_matrix(D, W)  
  D_inv_sqrt <- diag(1/sqrt(diag(D)))  
  
  L_sym <- D_inv_sqrt %*% L %*% D_inv_sqrt  
  return(L_sym)  
}  
  
L_sym1 <- sym_normalized_laplacian_matrix(D1, W1)  
L_sym2 <- sym_normalized_laplacian_matrix(D2, W2)  
L_sym3 <- sym_normalized_laplacian_matrix(D3, W3)
```

### c. Matrice laplacienne non symétrique

La matrice laplacienne non symétrique est une autre version de la matrice laplacienne. Elle est définie comme suit :

$$L_{rw} = D^{-1} L = I - D^{-1} W.$$

```
rw_normalized_laplacian_matrix <- function(D, W){  
  
  L <- unnormalized_laplacian_matrix(D, W)  
  D_inv <- diag(1/diag(D))  
  
  L_rw <- D_inv %*% L  
  return(L_rw)  
}  
  
L_rw1 <- rw_normalized_laplacian_matrix(D1, W1)  
L_rw2 <- rw_normalized_laplacian_matrix(D2, W2)  
L_rw3 <- rw_normalized_laplacian_matrix(D3, W3)
```

## 6. Calcul des valeurs et des vecteurs propres

### a. Détermination de la multiplicité de la valeur propre 0

On constate qu'avec un sigma grand pour la similarité gaussienne, les points de données sont plus connectés, ce qui peut réduire le nombre de composantes connexes dans le graphe. Cela conduit à une multiplicité plus faible de la valeur propre 0 dans la matrice laplacienne.

```
k <- function(L){  
  eig <- eigen(L)  
  eig_values <- eig$values  
  multiplicity <- sum(abs(eig_values) < 0.1) # Tolérance pour la comparaison avec zéro  
  return(multiplicity)  
}
```

```
(k1 <- k(L1_unnormalized))
```

```
## [1] 5
```

```
(k2 <- k(L2_unnormalized))
```

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

```
(k3 <- k(L3_unnormalized))
```

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

### b. Détermination des k vecteurs propres correspondants

```
eigen_decomposition <- function(L){  
  
  k <- k(L)  
  eig <- eigen(L)  
  eig_values <- eig$values  
  eig_vectors <- eig$vectors  
  
  # Sélection des k plus petites valeurs propres et leurs vecteurs propres correspondants  
  indices <- order(eig_values)[1:k]  
  selected_vectors <- eig_vectors[, indices]  
  
  return(list(values = eig_values[indices], vectors = selected_vectors))  
}
```

```
eig_L1_unnormalized <- eigen_decomposition(L1_unnormalized)
```

```
eig_L2_unnormalized <- eigen_decomposition(L2_unnormalized)
```

```
eig_L3_unnormalized <- eigen_decomposition(L3_unnormalized)
```

```
U1 <- eig_L1_unnormalized$vectors
```

```
U2 <- eig_L2_unnormalized$vectors
```

```
U3 <- eig_L3_unnormalized$vectors
```

## 7. Clustering

```
spectral_clustering <- function(U, num_clusters){  
  set.seed(123)  
  kmeans_result <- kmeans(U, centers = num_clusters, nstart = 1)
```

```

    return(kmeans_result)
}

clusters1 <- spectral_clustering(U1, k1)

clusters2 <- spectral_clustering(U2, k2)

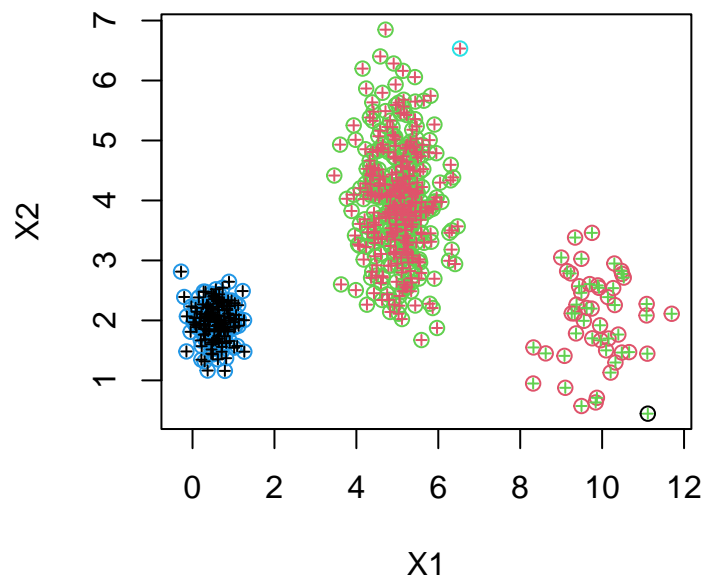
clusters3 <- spectral_clustering(U3, k3)

# Représentation graphique des clusters obtenus vs vrais groupes

plot(data[,1:2], col = clusters1$cluster, main = "Spectral Clustering (sigma=1/3)")
points(data[,1:2], col = data$Z, pch=3, cex=0.5)

```

### Spectral Clustering (sigma=1/3)

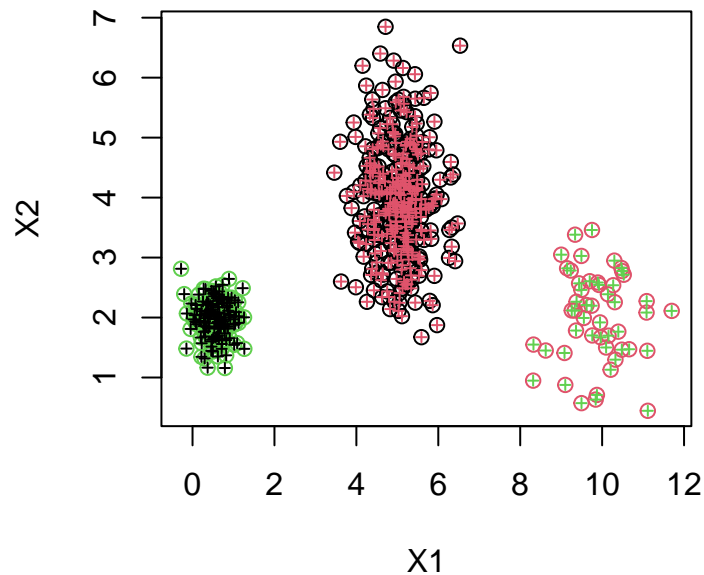


```

plot(data[,1:2], col = clusters2$cluster, main = "Spectral Clustering (sigma=2/3)")
points(data[,1:2], col = data$Z, pch=3, cex=0.5)

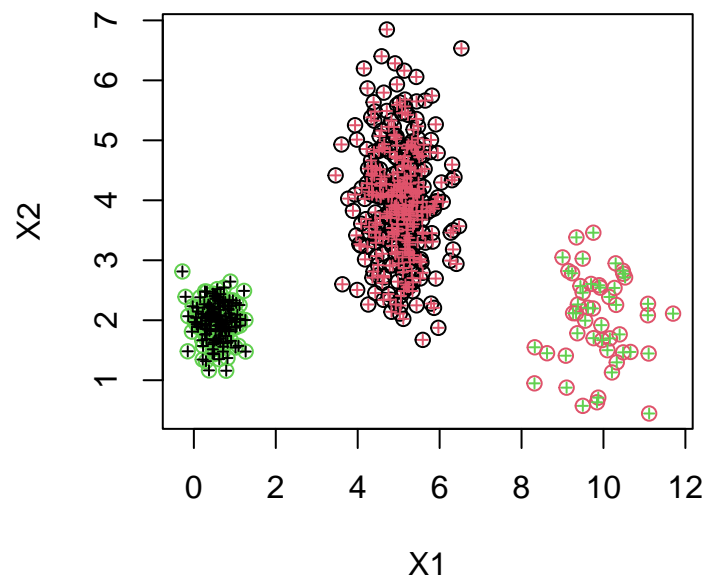
```

## Spectral Clustering (sigma=2/3)



```
plot(data[,1:2], col = clusters3$cluster, main = "Spectral Clustering (sigma=1)")
points(data[,1:2], col = data$Z, pch=3, cex=0.5)
```

## Spectral Clustering (sigma=1)

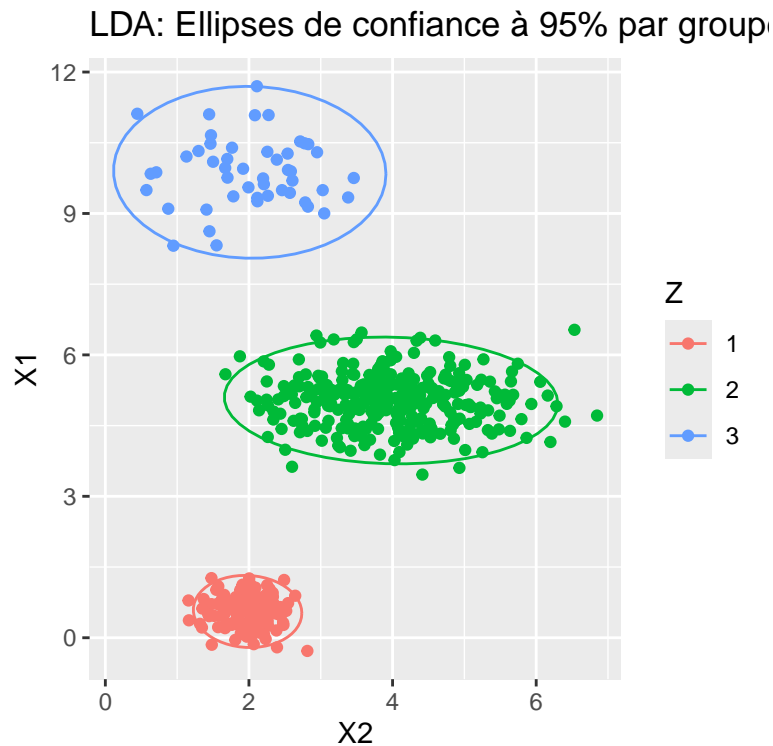


## II. Analyse discriminante

On réutilise le même jeu de données généré précédemment. Par la suite, nous allons appliquer l'analyse discriminante linéaire (LDA) et l'analyse discriminante quadratique (QDA) pour classifier les données.

```
library(MASS)
set.seed(123)
lda_model <- lda(Z ~ X1 + X2, data = data)
```

```
ggplot(data=data, aes(x=X2, y=X1, color=Z, fill=Z)) +
  geom_point() +
  stat_ellipse(type="norm", level=0.95) +
  labs(title="LDA: Ellipses de confiance à 95% par groupe")
```



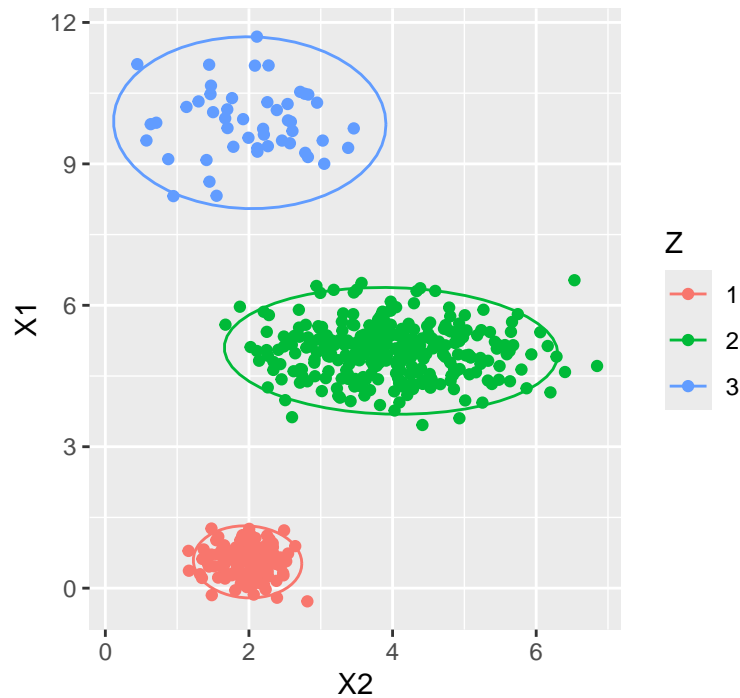
```
qda_model <- qda(Z ~ X1 + X2, data = data)
```

```
qda_model
```

```
## Call:
## qda(Z ~ X1 + X2, data = data)
##
## Prior probabilities of groups:
##      1      2      3
## 0.294 0.610 0.096
##
## Group means:
##      X1      X2
## 1 0.557259 1.980894
## 2 5.033402 3.979225
## 3 9.873017 2.011073
```

```
ggplot(data=data, aes(x=X2, y=X1, color=Z, fill=Z)) +
  geom_point() +
  stat_ellipse(type="norm", level=0.95) +
  labs(title="QDA: Ellipses de confiance à 95% par groupe")
```

QDA: Ellipses de confiance à 95% par group



## Conclusion

Dans ce TD, nous avons exploré deux techniques de clustering et de classification : le spectral clustering et l'analyse discriminante (LDA et QDA).

Le spectral clustering s'est avéré assez efficace pour identifier des groupes dans des données (simulées selon une loi de mélange gaussien), en utilisant des similarités basées sur une fonction gaussienne. Ceci peut s'expliquer par le fait que les données générées étaient très bien séparées, ce qui a facilité la tâche du clustering. De plus, nous avons exploré plusieurs valeurs de paramètres sigma ( $1/3$ ,  $2/3$ ,  $1$ ), ce qui a influencé la connectivité des points de données et, par conséquent, la performance du clustering.

L'analyse discriminante, quant à elle, a permis de classer les données en fonction de leurs caractéristiques, avec LDA supposant des covariances égales entre les classes et QDA permettant des covariances distinctes. Ces méthodes offrent des approches complémentaires pour l'analyse de données multivariées.