TP Clustering

2024-11-27

TP– Clustering et classification

Réalisé par:

- EL MOUDDEN Aya
- IMAD Aymen
- ZMIR Yassamine
- MOUTAOUAFFIQ Meryam

Problème 2

Analyse exploratoire des données iris

On commence par charger les données:

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
dim(iris)
```

```
## [1] 150   5
```

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

La base de données `iris` est composée de 150 observations réparties sur 5 colonnes : 4 variables quantitatives (longueur et largeur des sépales et des pétales) et une variable qualitative représentant les espèces des fleurs (`Species`). Ces dernières incluent trois classes : *setosa (50)*, *versicolor(50)* et *virginica(50)*.

```r
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'
```

```
## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag
```

```
## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

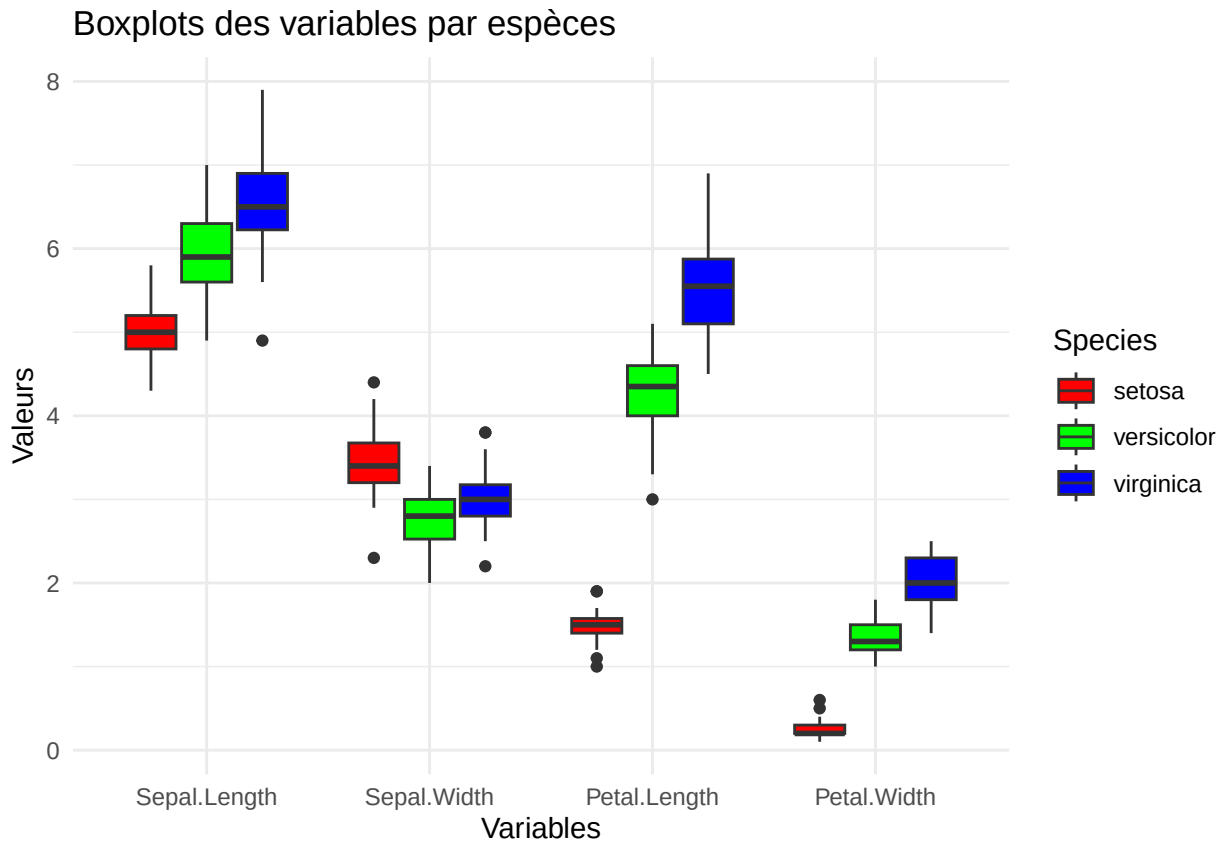
```r
iris %>% group_by(Species) %>% summarise(across(.cols = where(is.numeric), .fns = list(mean = mean, sd =
```

```
## # A tibble: 3 x 9
##   Species    Sepal.Length_mean Sepal.Length_sd Sepal.Width_mean Sepal.Width_sd
##   <fct>                  <dbl>           <dbl>            <dbl>          <dbl>
## 1 setosa                  5.01           0.352             3.43          0.379
## 2 versicolor              5.94           0.516             2.77          0.314
## 3 virginica               6.59           0.636             2.97          0.322
## # i 4 more variables: Petal.Length_mean <dbl>, Petal.Length_sd <dbl>,
## #   Petal.Width_mean <dbl>, Petal.Width_sd <dbl>
```

Les différences de moyenne entre les espèces sont particulièrement marquées pour la longueur et la largeur des pétales, ce qui les rend des caractéristiques discriminantes importantes.

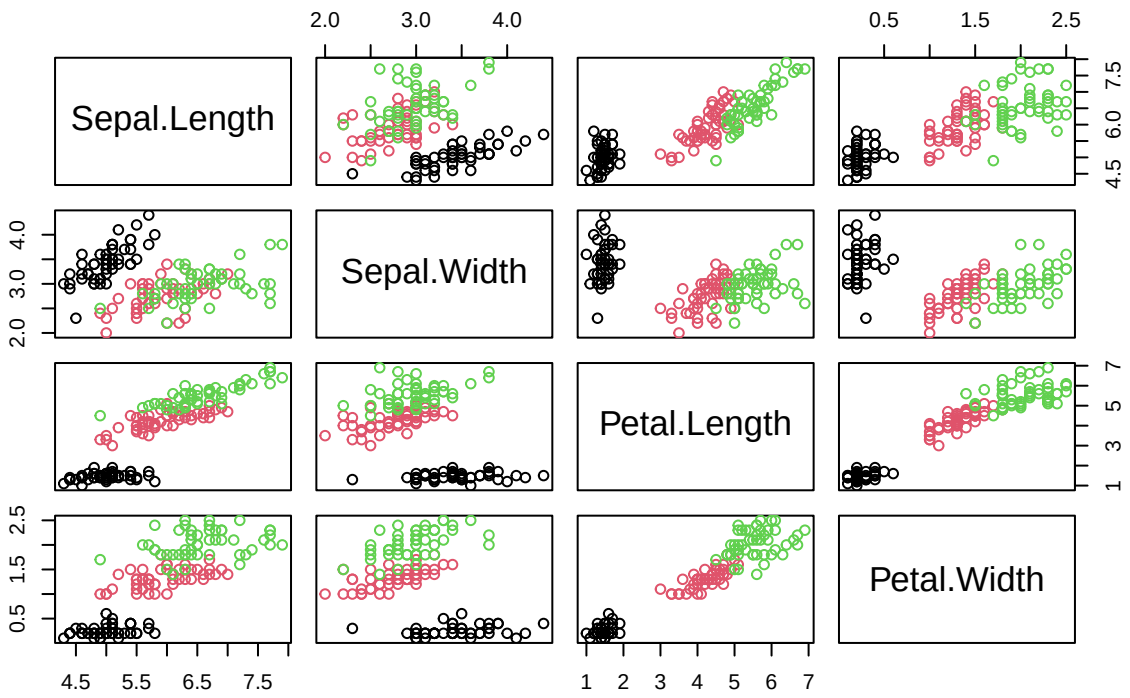
L'écart-type plus élevé pour *Versicolor* et *Virginica* dans certaines variables suggère une plus grande variabilité dans ces groupes, alors que *Setosa* semble plus homogène.

```r
# Réorganiser les données pour inclure toutes les variables
iris_long <- reshape2::melt(iris, id.vars = "Species",
                            measure.vars = c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width"))
library(ggplot2)
ggplot(iris_long, aes(x = variable, y = value, fill = Species)) +
  geom_boxplot() +
  labs(title = "Boxplots des variables par espèces",
       x = "Variables", y = "Valeurs") +
  scale_fill_manual(values = c("red", "green", "blue")) +
  theme_minimal()
```



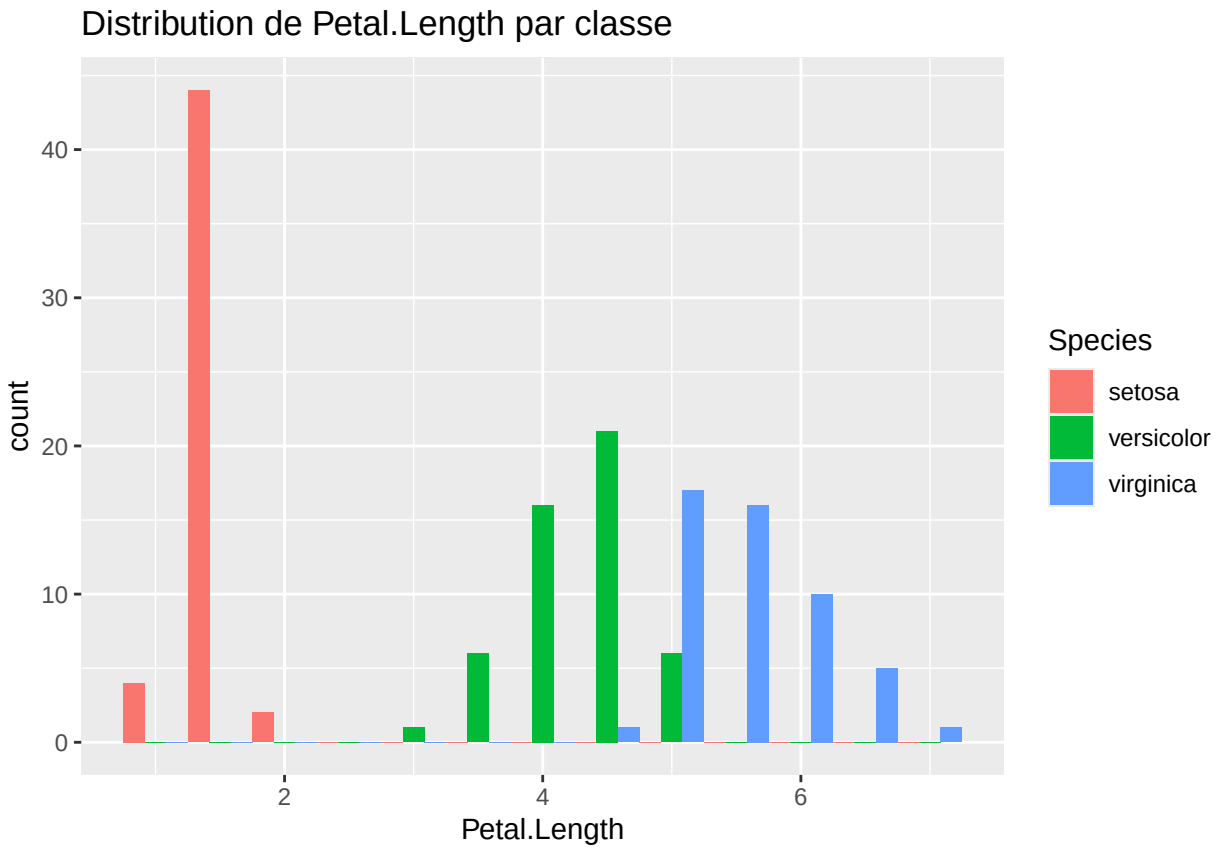
Les boxplots des variables mesurées (`Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`) montrent clairement les distributions et les écarts entre les différentes espèces. Par exemple, pour *Petal.Length*, les boxplots indiquent une grande variation pour *versicolor* et *virginica*, mais une distribution beaucoup plus serrée pour *setosa*.

```r
pairs(iris[, 1:4], col = iris$Species, main = "Scatterplot Matrix by Species")
```



Les espèces *setosa*, *versicolor* et *virginica* sont relativement séparées, bien que certaines superpositions existent entre *versicolor* et *virginica* pour certaines variables, comme la largeur et la longueur des sépales. Cela peut indiquer qu'une séparation complète des espèces en utilisant uniquement ces deux variables pourrait ne pas être optimale. Tandis que la largeur et la longueur des pétales sont des variables discriminantes.

```
library(ggplot2)
ggplot(iris, aes(x = Petal.Length, fill = Species)) +
  geom_histogram(binwidth = 0.5, position = "dodge") +
  labs(title = "Distribution de Petal.Length par classe")
```



La distribution de la longueur des pétales par espèce montre des différences significatives entre les espèces. *Setosa* a une longueur de pétales relativement plus petite, tandis que *virginica* présente une longueur de pétales beaucoup plus longue, avec *versicolor* se situant entre les deux. Ces observations sont cohérents avec les résultats déjà retouvées (la longueur des pétales est une caractéristique discriminante, distribution serrée pour *setosa*).

Cependant, on remarque que les distributions de *Versicolor* et *Virginica* sont relativement proches, notamment pour la variable *Petal.Length*, qui est l'une des caractéristiques les plus discriminantes.

Algorithmes de clustering

K-means *Implémentation manuelle de K-Means*:

Le clustering K-means est une méthode utilisée pour regrouper des données non étiquetées en **k** clusters. Dans ce cas, bien que les données de l'iris soient étiquetées avec les espèces, nous devons ignorer la colonne `Species` afin que l'algorithme effectue un regroupement basé uniquement sur les caractéristiques numériques. Après avoir appliqué l'algorithme K-means, nous pourrons comparer les clusters obtenus avec les espèces réelles pour évaluer la précision du modèle.

```
k_means_manual <- function(data, k, max_iter = 100) {
  # Initialiser les centres de clusters
  set.seed(123)
  centers <- data[sample(1:nrow(data), k), ]

  # Initialisation des variables
  clusters <- rep(0, nrow(data)) # Affectation des clusters
```

```
  old_centers <- centers
  iter <- 0

  # Boucle de convergence
  repeat {
    iter <- iter + 1
    # Etape 1: Affecter chaque point au cluster le plus proche
    for (i in 1:nrow(data)) {
      distances <- apply(centers, 1, function(center) sum((data[i,] - center)^2))
      clusters[i] <- which.min(distances)
    }

    # Etape 2: Recalculer les centres des clusters
    for (j in 1:k) {
      centers[j, ] <- colMeans(data[clusters == j, , drop = FALSE])
    }

    # Vérification de la convergence
    if (all(centers == old_centers) || iter >= max_iter) break
    old_centers <- centers
  }

  return(list(clusters = clusters, centers = centers))
}
```

Application de l'algorithme K-Means manuel:

```
data_iris <- iris[, 1:4]
k <- 3
kmeans_manual_result <- k_means_manual(data_iris, k)
iris$Cluster_Manual <- as.factor(kmeans_manual_result$clusters)

# Affichage du tableau de comparaison
table(iris$Cluster_Manual, iris$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         47        14
##   3      0          3        36
```

Les résultats montrent que l'algorithme K-means a bien séparé **Setosa** dans un cluster distinct (Cluster 1), mais a regroupé **Versicolor** et **Virginica** dans deux clusters partiellement mélangés surtout le Cluster 2. Cela suggère que ces deux espèces partagent des caractéristiques similaires, rendant leur séparation plus difficile.

Calcul de la précision K-Means manuel:

```
kmeans_manual_conf_matrix <- table(iris$Cluster_Manual, iris$Species)
kmeans_manual_accuracy <- sum(diag(kmeans_manual_conf_matrix)) / sum(kmeans_manual_conf_matrix)
cat("Précision K-Means manuel :", kmeans_manual_accuracy, "\n")
```

```
## Précision K-Means manuel : 0.8866667
```

La précision du modèle K-Means manuel est de 88,67 %, ce qui indique que 88,67 % des observations ont été correctement regroupées dans les clusters prédits. Un tel score indique que le modèle K-Means manuel a bien fonctionné pour cette tâche de clustering.

Validons ceci par le modèle k-means prédéfini:

```
set.seed(123)
kmeans_result <- kmeans(iris[, 1:4], centers = 3, nstart = 20)
iris$Cluster <- as.factor(kmeans_result$cluster)

# Afficher les résultats du clustering
kmeans_conf_matrix <-table(iris$Cluster, iris$Species)
print(kmeans_conf_matrix)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         48        14
##   3      0          2        36
```

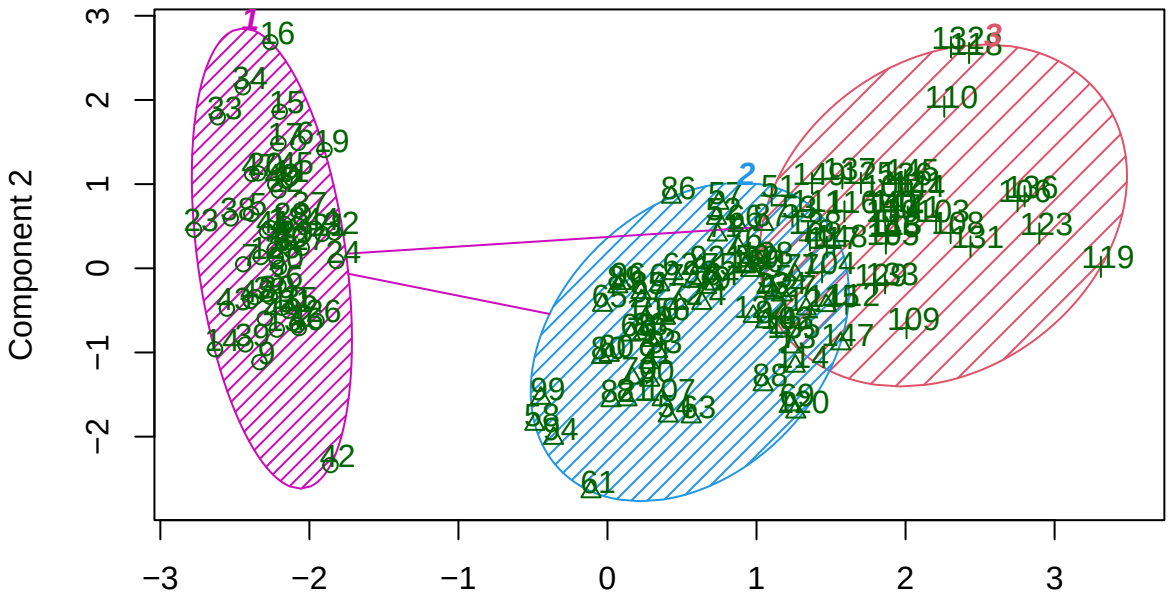
```
# Précision k-means
kmeans_accuracy <- sum(diag(kmeans_conf_matrix)) / sum(kmeans_conf_matrix)
cat("Précision K-Means :", kmeans_accuracy, "\n")
```

```
## Précision K-Means : 0.8933333
```

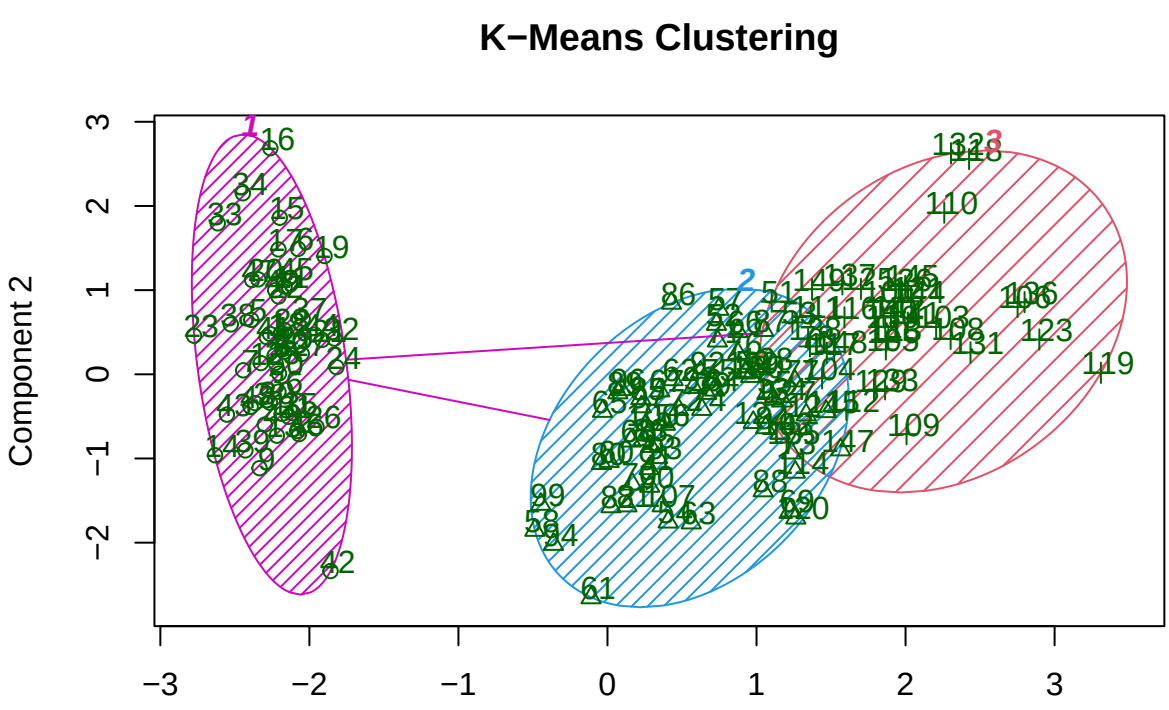
En comparant les deux méthodes, on constate que les résultats sont très proches, mais l'approche prédéfinie semble légèrement plus performante en termes de précision. On valide ainsi, la méthode manuelle.

Visualisation des clusters:

```
library(cluster)
clusplot(data_iris, kmeans_manual_result$clusters, color = TRUE, shade = TRUE, labels = 2, main = "K-Mea
```



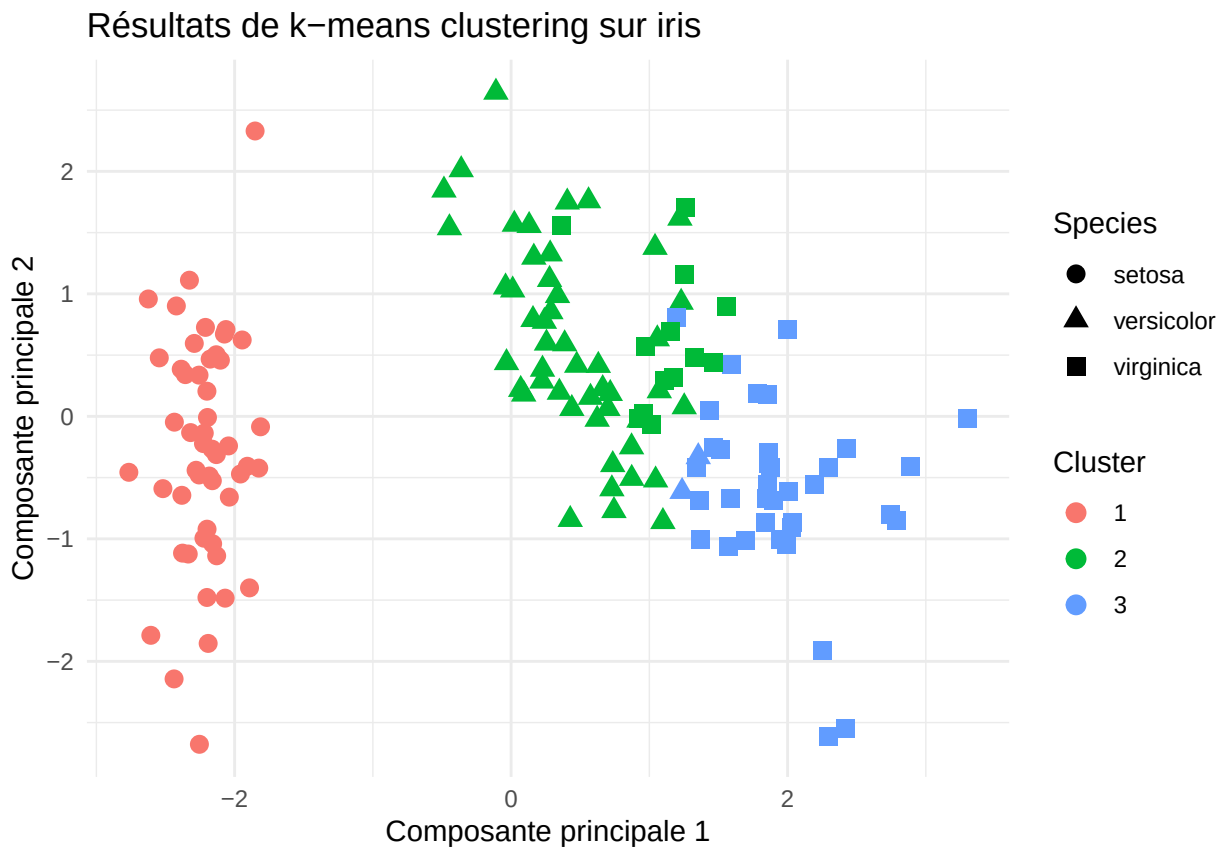

```
clusplot(iris[, 1:4], kmeans_manual_result$cluster, color = TRUE, shade = TRUE, labels = 2, main = "K-M
```



These two components explain 95.81 % of the point variability.

```r
# Réduction de dimensions avec ACP
library(ggplot2)
iris_pca <- prcomp(iris[, 1:4], scale. = TRUE)
iris$PC1 <- iris_pca$x[, 1]
iris$PC2 <- iris_pca$x[, 2]

# Graphe des clusters
ggplot(iris, aes(x = PC1, y = PC2, color = Cluster, shape = Species)) +
  geom_point(size = 3) +
  labs(title = "Résultats de k-means clustering sur iris",
       x = "Composante principale 1",
       y = "Composante principale 2") +
  theme_minimal()
```



Clustering hiérarchique Implémentation manuelle du clustering hiérarchique:

```r
hclust_manual <- function(data, method = "complete") {
  # Calculer la matrice des distances
  dist_matrix <- as.matrix(dist(data))

  # Initialiser chaque point comme un cluster (chaque point est un cluster individuel)
  clusters <- as.list(1:nrow(data))

  # Répéter jusqu'à ce qu'il ne reste qu'un seul cluster
  while (length(clusters) > 1) {
    min_dist <- Inf
    merge_clusters <- c()

    # Chercher les deux clusters les plus proches
    for (i in 1:(length(clusters) - 1)) {
      for (j in (i + 1):length(clusters)) {
        cluster_i <- clusters[[i]]
        cluster_j <- clusters[[j]]

        # Calculer la distance entre les deux clusters
        if (method == "complete") {
          # Méthode de liaison complète : la distance entre les clusters est la plus grande distance
          dist_ij <- max(dist_matrix[cluster_i, cluster_j])
        } else if (method == "average") {
```

```
          # Méthode de liaison moyenne : la distance est la moyenne des distances entre tous les points
          dist_ij <- mean(dist_matrix[cluster_i, cluster_j])
        } else {
          # Méthode de liaison simple : la distance entre les clusters est la plus petite distance
          dist_ij <- min(dist_matrix[cluster_i, cluster_j])
        }

        if (dist_ij < min_dist) {
          min_dist <- dist_ij
          merge_clusters <- c(i, j)
        }
      }
    }

    # Fusionner les deux clusters les plus proches
    new_cluster <- c(clusters[[merge_clusters[1]]], clusters[[merge_clusters[2]]])
    clusters <- clusters[-merge_clusters]
    clusters[[length(clusters) + 1]] <- new_cluster
  }

  return(clusters[[1]])
}
```

Application du clustering hiérarchique manuel:

```
data_iris <- iris[, 1:4]
hclust_result_manual <- hclust_manual(data_iris)
hclust_manual_clusters <- cutree(hclust(as.dist(dist(data_iris)), method = "ward.D2"), k = 3)
iris$HCluster_Manual <- as.factor(hclust_manual_clusters)
table(iris$HCluster_Manual, iris$Species)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         49        15
##   3      0          1        35
```

Les résultats du clustering hiérarchique montrent que Setosa est parfaitement séparée dans un cluster distinct (Cluster 1), tandis que Versicolor et Virginica sont partiellement mélangées dans les Clusters 2 et 3. En particulier, le Cluster 2 contient à la fois des observations de Versicolor et de Virginica, mettant l'accent sur le fait que ces deux espèces partagent des caractéristiques similaires.

Calcul de la précision du clustering hiérarchique manuel:

```
hclust_manual_conf_matrix <- table(iris$HCluster_Manual, iris$Species)
hclust_manual_accuracy <- sum(diag(hclust_manual_conf_matrix)) / sum(hclust_manual_conf_matrix)
cat("Précision Clustering Hiérarchique manuel :", hclust_manual_accuracy, "\n")
```

```
## Précision Clustering Hiérarchique manuel : 0.8933333
```

La précision est élevée indiquant que le modèle est bon. Toutefois, appliquons la fonction prédéfinie.

```r
dist_iris <- dist(iris[, 1:4])
# Clustering hiérarchique
hclust_result <- hclust(dist_iris, method = "ward.D2")
# Découpage en 3 clusters
iris$HCluster <- as.factor(cutree(hclust_result, k = 3))
hclust_conf_matrix <- table(iris$HCluster, iris$Species)
print(hclust_conf_matrix)
```

```
##
##     setosa versicolor virginica
##   1     50          0         0
##   2      0         49        15
##   3      0          1        35
```

Précision clustering hiérarchique:

```r
hclust_accuracy <- sum(diag(hclust_conf_matrix)) / sum(hclust_conf_matrix)
cat("Précision Clustering Hiérarchique :", hclust_accuracy, "\n")
```

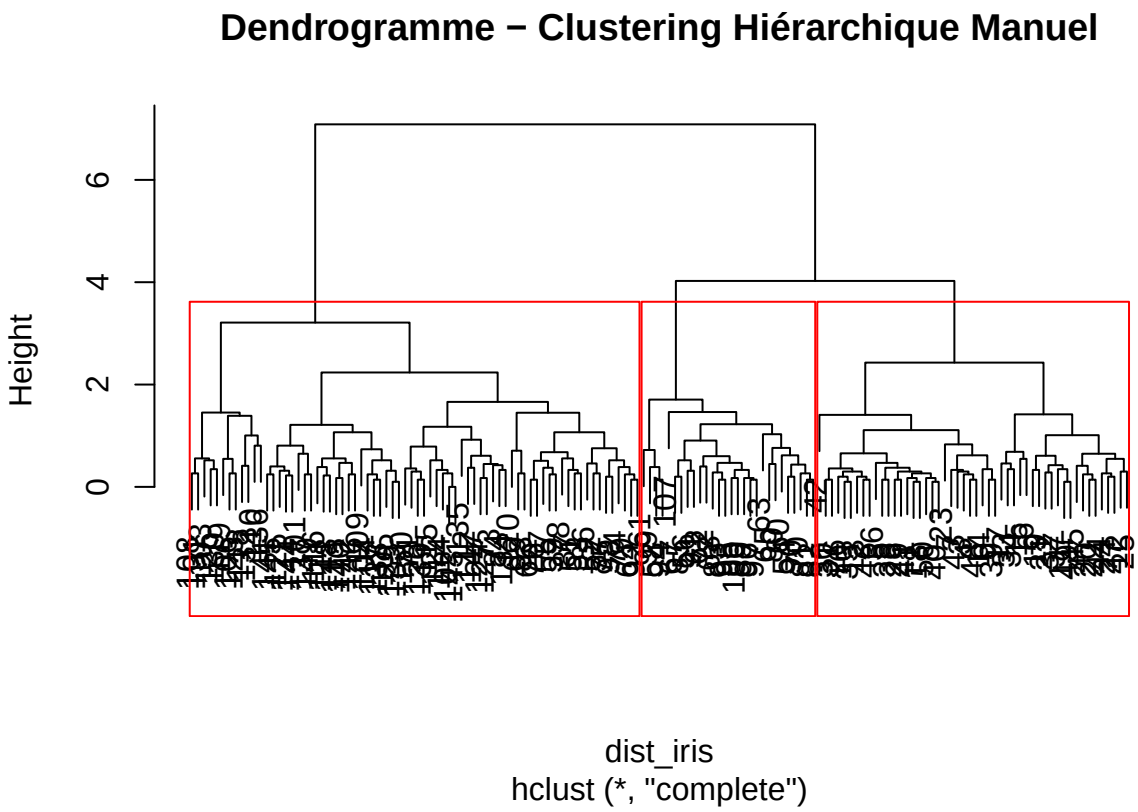
```
## Précision Clustering Hiérarchique : 0.8933333
```

La précision Clustering Hiérarchiquemanuelle est égale à la précision de la fonction prédéfinie. On val

Visualisation du dendrogramme:

```r
dendrogram_plot <- function(hclust_result) {
  dist_iris <- dist(data_iris)
  hc_result <- hclust(dist_iris, method = "complete")
  plot(hc_result, main = "Dendrogramme - Clustering Hiérarchique Manuel")
  rect.hclust(hc_result, k = 3, border = "red")
}

# Tracer le dendrogramme
dendrogram_plot(hclust_result_manual)
```




```
set.seed(123)
library(class)
library(caret)
```

Implémentation de la méthode KNN

```
## Warning: le package 'caret' a été compilé avec la version R 4.4.2
```

```
## Le chargement a nécessité le package : lattice
```

```
train_index <- createDataPartition(iris$Species, p = 0.7, list = FALSE)
train_data <- iris[train_index, ]
test_data <- iris[-train_index, ]

train_x <- train_data[, 1:4]
train_y <- train_data$Species
test_x <- test_data[, 1:4]
test_y <- test_data$Species

knn_result <- knn(train = train_x, test = test_x, cl = train_y, k = 3)

# Matrice de confusion
```

```r
conf_matrix <- table(knn_result, test_y)
print(conf_matrix)
```

```
##             test_y
## knn_result   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         15         2
##   virginica       0          0        13
```

La matrice de confusion donne les informations suivantes :

- **Setosa** : Le modèle a correctement classé toutes les observations de Setosa dans la classe “setosa” (15 correctes, aucune erreur).
- **Versicolor** : Le modèle a bien classé la majorité des observations de Versicolor (15 correctes), mais a fait 2 erreurs, classant ces observations comme “virginica”.
- **Virginica** : Le modèle a bien classé toutes les observations de Virginica (13 correctes), sans erreur.

Globalement, le modèle est performant pour la classification des espèces, avec quelques erreurs dans la distinction entre Versicolor et Virginica vues leurs caractéristiques communes.

Calcul de la précision de KNN:

```r
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
cat("Précision de KNN :", accuracy, "\n")
```

```
## Précision de KNN : 0.9555556
```

KNN offre une **précision élevée** par rapport aux autres algorithmes non supervisés (K-means et clustering hiérarchique), car il est spécifiquement conçu pour la classification supervisée. Il surpasse ces algorithmes en utilisant les étiquettes d’espèce dans son apprentissage, permettant une meilleure identification des différentes classes.

Problème 3

Nous allons travailler avec les données mammal.dentition.

```r
library(ggplot2)
library(class)
library(cluster.datasets)
library(cluster)
```

```r
data(mammal.dentition)
head(mammal.dentition)
```

```
##              name top.i bottom.i top.c bottom.c top.pm bottom.pm top.m bottom.m
## 1         Opossum     5        4     1        1      3         3     4        4
## 2 Hairy tail mole     3        3     1        1      4         4     3        3
## 3     Common mole     3        2     1        0      3         3     3        3
## 4  Star nose mole     3        3     1        1      4         4     3        3
## 5       Brown bat     2        3     1        1      3         3     3        3
## 6 Silver hair bat     2        3     1        1      2         3     3        3
```

Nous allons supprimer la 1ère colonne.

```
mammal_data <- mammal.dentition[, -1]
```

Clustering (classification non-supervisée)

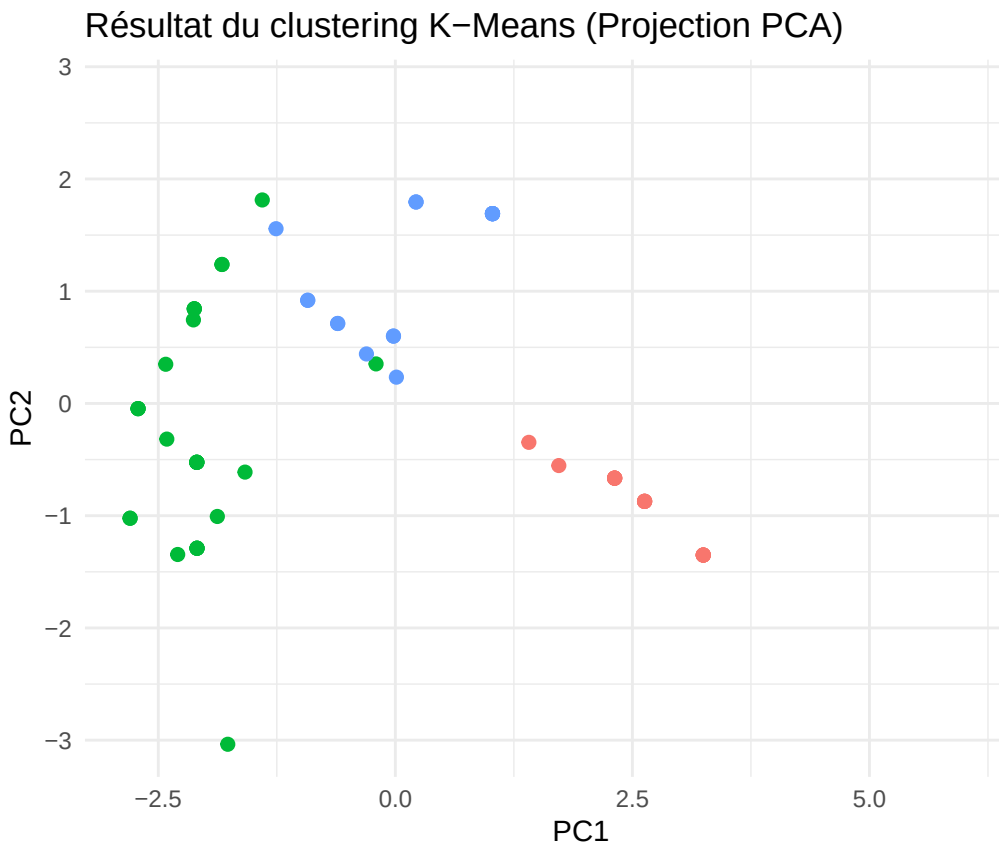
```
kmeans_result <- kmeans(mammal_data, centers = 3, nstart = 20)

mammal.dentition$Cluster_kmeans <- as.factor(kmeans_result$cluster)
```

kmeans

```
pca_result <- prcomp(mammal_data, center = TRUE, scale. = TRUE)
pca_data <- data.frame(pca_result$x, Cluster = as.factor(kmeans_result$cluster))

ggplot(pca_data, aes(PC1, PC2, color = Cluster)) +
  geom_point(size = 2) +
  ggtitle("Résultat du clustering K-Means (Projection PCA)") +
  theme_minimal()
```



PCA pour réduire la dimension

```r
set.seed(123)  # Pour reproductibilité
k <- 3
max_iter <- 100
n <- nrow(mammal_data)
p <- ncol(mammal_data)

centres <- mammal_data[sample(1:n, k), ]

# 4. Implémenter l'algorithme K-means manuellement
for (iter in 1:max_iter) {

  # 4.1. Étape 1 : Assigner chaque point au cluster le plus proche
  distances <- matrix(NA, nrow = n, ncol = k)
  for (i in 1:n) {
    for (j in 1:k) {
      distances[i, j] <- sum((mammal_data[i, ] - centres[j, ])^2)  # Calcul de la distance euclidienne
    }
  }

  # Affecter chaque point au cluster ayant la distance minimale
  clusters <- apply(distances, 1, which.min)

  # 4.2. Étape 2 : Calculer les nouveaux centres de clusters
  new_centres <- matrix(NA, nrow = k, ncol = p)
  for (j in 1:k) {
    new_centres[j, ] <- colMeans(mammal_data[clusters == j, , drop = FALSE])
  }

  # 4.3. Vérifier si les centres ont changé
  if (all(new_centres == centres)) {
    cat("Convergence atteinte après", iter, "itérations.\n")
    break
  }

  centres <- new_centres  # Mettre à jour les centres
}
```

k-means à la main

```
## Convergence atteinte après 6 itérations.
```

```r
table(clusters)
```

```
## clusters
##  1  2  3
## 20 28 18
```

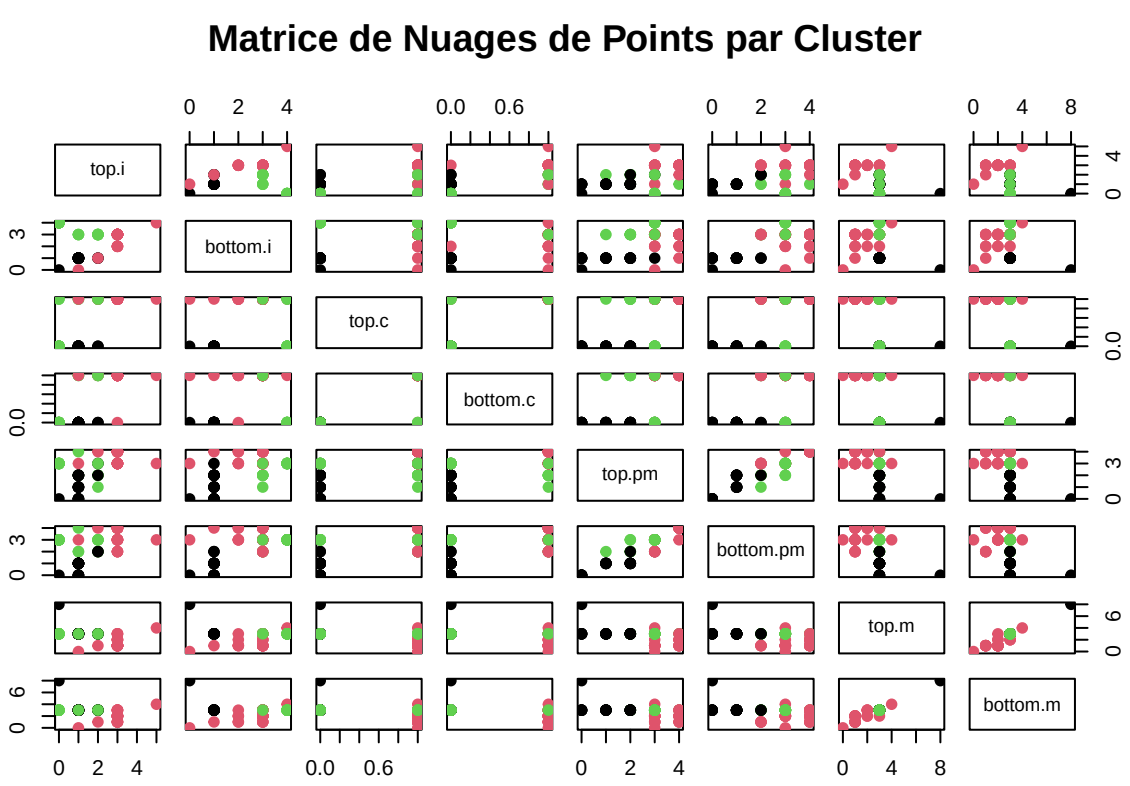
```r
print("Centres finaux des clusters :")
```

```
## [1] "Centres finaux des clusters :"
```

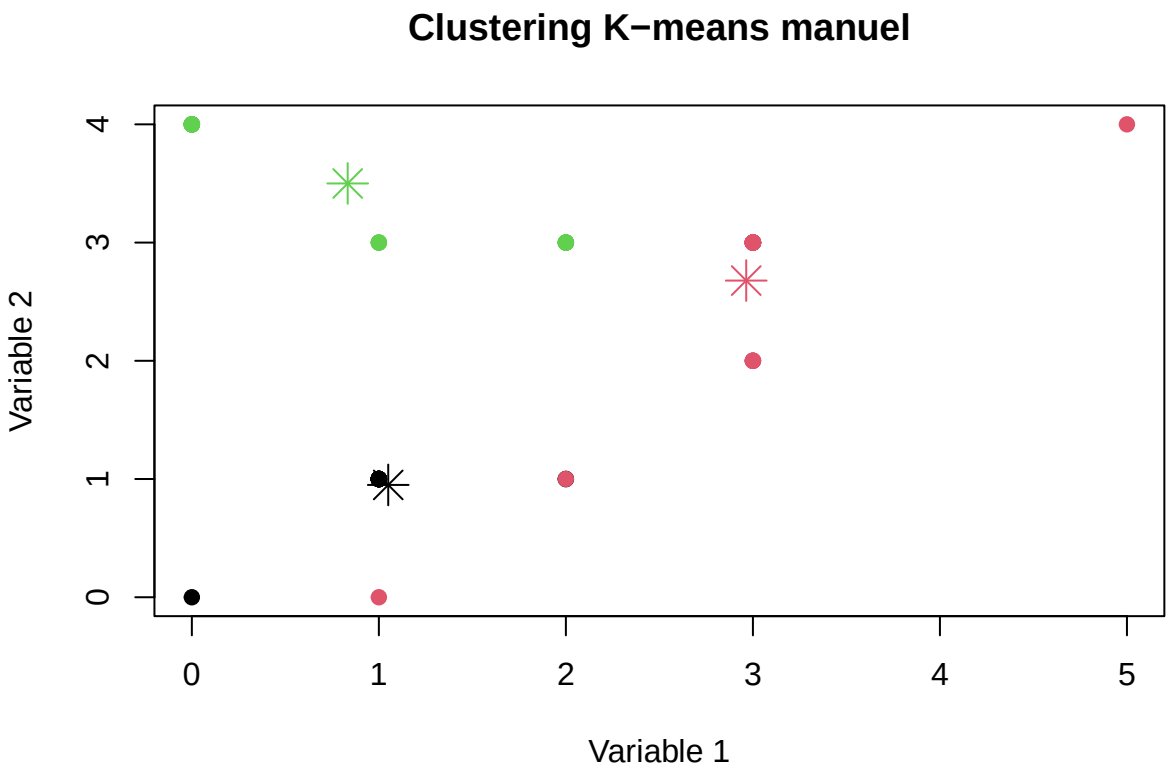
```r
print(centres)
```

```
##           [,1]     [,2]      [,3]      [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 1.0500000 0.950000 0.0000000 0.0000000 1.150000 0.800000 3.250000 3.250000
## [2,] 2.9642857 2.678571 1.0000000 0.9642857 3.500000 3.321429 1.535714 1.964286
## [3,] 0.8333333 3.500000 0.6111111 0.5000000 2.666667 2.833333 3.000000 3.000000
```

```r
#  Plot des clusters
pairs(mammal_data, col = clusters, pch = 19, main = "Matrice de Nuages de Points par Cluster")
```

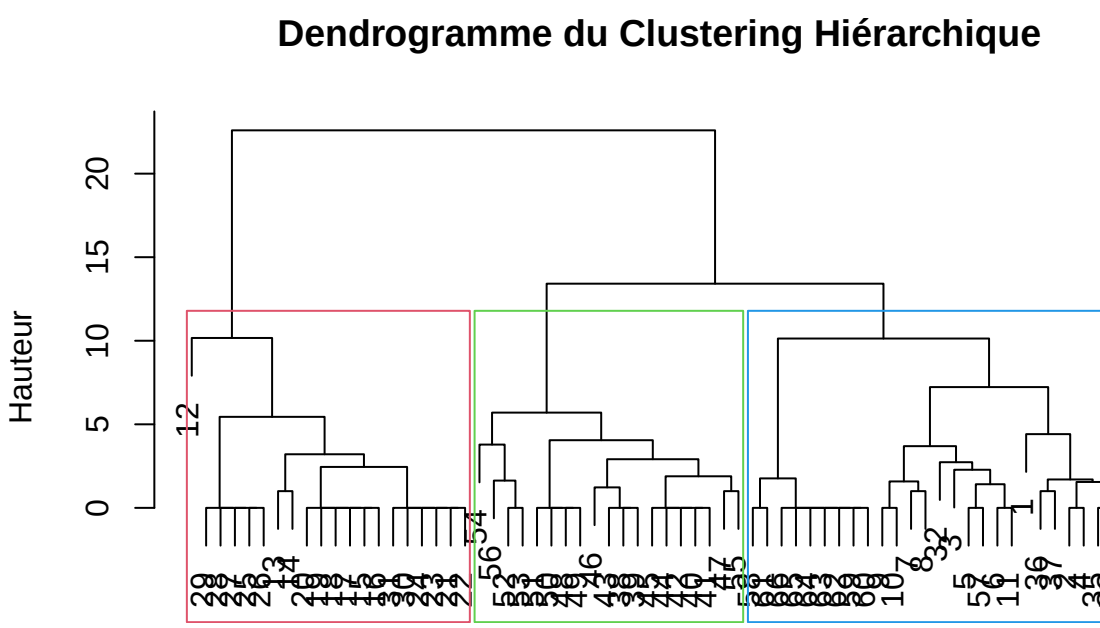



```r
# Choisir deux variables pour le plot (par exemple, les deux premières colonnes)
plot(mammal_data[, 1], mammal_data[, 2], col = clusters, pch = 19,
     xlab = "Variable 1", ylab = "Variable 2", main = "Clustering K-means manuel")
# Ajouter les centres des clusters
points(centres[, 1], centres[, 2], col = 1:k, pch = 8, cex = 2)
```




```r
dist_matrix <- dist(mammal_data)
hclust_result <- hclust(dist_matrix, method = "ward.D2")
cutree_hclust <- cutree(hclust_result, k = 3)


plot(hclust_result, main = "Dendrogramme du Clustering Hiérarchique", sub = "", xlab = "", ylab = "Haut
rect.hclust(hclust_result, k = 3, border = 2:4)  # Ajouter des rectangles autour des 3 clusters
```



Clustering hiérarchique

```r
hclust_manual <- function(mammal_data, method = "ward.D2")

#----------hierarchique manu
# Implémentation manuelle du clustering hiérarchique (HAC)
hclust_manual <- function(data, method = "complete") {
  # Calculer la matrice des distances
  dist_matrix <- as.matrix(dist(data))

  # Initialiser chaque point comme un cluster (chaque point est un cluster individuel)
  clusters <- as.list(1:nrow(data))

  # Répéter jusqu'à ce qu'il ne reste qu'un seul cluster
  while (length(clusters) > 1) {
    min_dist <- Inf
    merge_clusters <- c()

    # Chercher les deux clusters les plus proches
    for (i in 1:(length(clusters) - 1)) {
      for (j in (i + 1):length(clusters)) {
        cluster_i <- clusters[[i]]
        cluster_j <- clusters[[j]]

        # Calculer la distance entre les deux clusters
        if (method == "complete") {
          # Méthode de liaison complète : la distance entre les clusters est la plus grande distance
          dist_ij <- max(dist_matrix[cluster_i, cluster_j])
```

```
        } else if (method == "average") {
          # Méthode de liaison moyenne : la distance est la moyenne des distances entre tous les points
          dist_ij <- mean(dist_matrix[cluster_i, cluster_j])
        } else {
          # Méthode de liaison simple : la distance entre les clusters est la plus petite distance
          dist_ij <- min(dist_matrix[cluster_i, cluster_j])
        }

        if (dist_ij < min_dist) {
          min_dist <- dist_ij
          merge_clusters <- c(i, j)
        }
      }
    }

    # Fusionner les deux clusters les plus proches
    new_cluster <- c(clusters[[merge_clusters[1]]], clusters[[merge_clusters[2]]])
    clusters <- clusters[-merge_clusters]
    clusters[[length(clusters) + 1]] <- new_cluster
  }

  return(clusters[[1]])
}
```

Évaluer les résultats avec l'indice de silhouette

```
silhouette_kmeans <- silhouette(kmeans_result$cluster, dist_matrix)
silhouette_hclust <- silhouette(cutree_hclust, dist_matrix)

# Moyenne des coefficients de silhouette
silhouette_kmeans_mean <- mean(silhouette_kmeans[, 3])
silhouette_hclust_mean <- mean(silhouette_hclust[, 3])

cat("Silhouette moyenne (k-means):", silhouette_kmeans_mean, "\n")
```

```
## Silhouette moyenne (k-means): 0.4665756
```

```
cat("Silhouette moyenne (hiérarchique):", silhouette_hclust_mean, "\n")
```

```
## Silhouette moyenne (hiérarchique): 0.421197
```

Identifier la meilleure méthode

```
if (silhouette_kmeans_mean > silhouette_hclust_mean) {
  cat("La méthode k-means produit les meilleurs résultats.\n")
} else {
  cat("La méthode hiérarchique produit les meilleurs résultats.\n")
}
```

```
## La méthode k-means produit les meilleurs résultats.
```

Classification supervisée avec k-NN

```r
knn_manuel <- function(train, test, labels, k) {
  # Vérification des dimensions
  if (nrow(train_data) == 0 || nrow(test_data) == 0) {
    stop("Les ensembles d'entraînement ou de test sont vides.")
  }
  if (k < 1) {
    stop("k doit être supérieur ou égal à 1.")
  }

  # Initialisation du vecteur des prédictions
  predictions <- vector("character", nrow(test_data))

  # Calcul des distances et classification pour chaque observation de test
  for (i in 1:nrow(test_data)) {
    # Calculer les distances euclidiennes entre l'observation test et toutes les observations d'entraîn
    distances <- sqrt(rowSums((t(t(train_data) - test_data[i, ]))^2))

    # Trouver les indices des k plus proches voisins
    nearest_neighbors <- order(distances)[1:k]

    # Récupérer les étiquettes des voisins
    neighbor_labels <- train_labels[nearest_neighbors]

    # Identifier la classe majoritaire parmi les voisins
    predictions[i] <- names(which.max(table(neighbor_labels)))
  }

  # Retourner les prédictions
  return(predictions)
}
```

```r
mammal.dentition$classe <- sample(c("Carnivore", "Herbivore", "Omnivore"), nrow(mammal_data), replace =
# Compter les observations par classe
class_counts <- table(mammal.dentition$classe)
```

```r
# Garder uniquement les classes avec plus d'une observation
valid_classes <- names(class_counts[class_counts > 1])
dentition_filtered <- mammal.dentition[mammal.dentition$classe %in% valid_classes, ]
```

Nous allons diviser les données.

```r
train_indices <- sample(1:nrow(dentition_filtered), size = 0.7 * nrow(dentition_filtered))

train_data <- dentition_filtered[train_indices, -c(1, ncol(dentition_filtered))]  # Exclure noms et cla
train_labels <- dentition_filtered[train_indices, "classe"]

test_data <- dentition_filtered[-train_indices, -c(1, ncol(dentition_filtered))]
test_labels <- dentition_filtered[-train_indices, "classe"]
```

```r
k <- 3

knn_ <- knn(train_data, test_data, train_labels, k)

# Matrice de confusion
confusion_matrix <- table(Predicted = knn_, Actual = test_labels)
print(confusion_matrix)
```

```
##            Actual
## Predicted   Carnivore Herbivore Omnivore
##   Carnivore         1         2        1
##   Herbivore         4         1        2
##   Omnivore          3         3        3
```

```r
# Calculer la précision
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Précision globale avec k =", k, ":", accuracy, "\n")
```

```
## Précision globale avec k = 3 : 0.25
```

Nous allons tester différentes valeurs de k pour k-NN.

```r
accuracy_results <- c()
for (k in 2:10) {
  knn_pred <- knn(train_data, test_data,  train_labels,  k)
  conf_matrix <- table(test_labels, knn_pred)
  accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
  accuracy_results <- c(accuracy_results, accuracy)
  cat("k =", k, " - Précision:", round(accuracy * 100, 2), "%\n")
}
```

```
## k = 2  - Précision: 30 %
## k = 3  - Précision: 25 %
## k = 4  - Précision: 20 %
## k = 5  - Précision: 15 %
## k = 6  - Précision: 20 %
## k = 7  - Précision: 20 %
## k = 8  - Précision: 25 %
## k = 9  - Précision: 40 %
## k = 10  - Précision: 25 %
```

```r
# Identifier la meilleure valeur de k
best_k <- which.max(accuracy_results)
cat("La meilleure valeur pour k est:", best_k, "avec une précision de", round(accuracy_results[best_k] *
```

```
## La meilleure valeur pour k est: 8 avec une précision de 40 %
```

Problème 4

Dans cette partie nous allons travailler sur les données “dataset diabetes”.

```r
library(cluster)
diabetes_data <- read.csv("C:/Users/Eleve/Documents/Data/dataset_diabetes.csv")
summary(diabetes_data)
```

```
##     pregnant         glucose        diastolic         triceps
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##    insuline2h          imc           pedigree           age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725   Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00
##     class
##  Length:768
##  Class :character
##  Mode  :character
##
##
##
```

```r
colSums(is.na(diabetes_data))
```

```
##   pregnant    glucose  diastolic    triceps insuline2h        imc   pedigree
##          0          0          0          0          0          0          0
##        age      class
##          0          0
```

Nous remarquons qu'il n'y a pas de valeurs manquantes.

Nous allons ensuite séparer les données (caractéristiques) et les labels (classes).

```r
features <- diabetes_data[, -ncol(diabetes_data)]
labels <- diabetes_data$class  # "class" est la colonne des étiquettes
set.seed(123)
```

Définition des fonctions manuelles pour les méthodes de clustering

Clustering hiérarchique (hclust_manual)

```r
hclust_manual <- function(data, method = "complete") {
  # Calculer la matrice des distances
  dist_matrix <- as.matrix(dist(data))

  # Initialiser chaque point comme un cluster (chaque point est un cluster individuel)
  clusters <- as.list(1:nrow(data))
```

```
  # Répéter jusqu'à ce qu'il ne reste qu'un seul cluster
  while (length(clusters) > 1) {
    min_dist <- Inf
    merge_clusters <- c()

    # Chercher les deux clusters les plus proches
    for (i in 1:(length(clusters) - 1)) {
      for (j in (i + 1):length(clusters)) {
        cluster_i <- clusters[[i]]
        cluster_j <- clusters[[j]]

        # Calculer la distance entre les deux clusters
        if (method == "complete") {
          # Méthode de liaison complète : la distance entre les clusters est la plus grande distance
          dist_ij <- max(dist_matrix[cluster_i, cluster_j])
        } else if (method == "average") {
          # Méthode de liaison moyenne : la distance est la moyenne des distances entre tous les points
          dist_ij <- mean(dist_matrix[cluster_i, cluster_j])
        } else {
          # Méthode de liaison simple : la distance entre les clusters est la plus petite distance
          dist_ij <- min(dist_matrix[cluster_i, cluster_j])
        }

        if (dist_ij < min_dist) {
          min_dist <- dist_ij
          merge_clusters <- c(i, j)
        }
      }
    }

    # Fusionner les deux clusters les plus proches
    new_cluster <- c(clusters[[merge_clusters[1]]], clusters[[merge_clusters[2]]])
    clusters <- clusters[-merge_clusters]
    clusters[[length(clusters) + 1]] <- new_cluster
  }

  return(clusters[[1]])
}
```

k-means (my_kmeans)

```
my_kmeans <- function(data, k, max_iter = 100, tol = 1e-4) {
  # Initialiser les centres de clusters aléatoirement
  set.seed(123)
  centers <- data[sample(1:nrow(data), k), ]

  # Initialiser les clusters
  clusters <- rep(0, nrow(data))
  iter <- 0
  converged <- FALSE

  while (iter < max_iter && !converged) {
    iter <- iter + 1
```

```
    # Étape 1 : Assigner chaque point au cluster le plus proche
    for (i in 1:nrow(data)) {
      distances <- apply(centers, 1, function(center) sum((data[i, ] - center)^2))
      clusters[i] <- which.min(distances)
    }

    # Étape 2 : Recalculer les centres
    new_centers <- sapply(1:k, function(cluster) {
      cluster_points <- data[clusters == cluster, , drop = FALSE]
      if (nrow(cluster_points) > 0) colMeans(cluster_points) else centers[cluster, ]
    })
    new_centers <- t(new_centers)  # Convertir en matrice

    # Vérifier la convergence (variation des centres)
    diff <- sum((new_centers - centers)^2)
    if (diff < tol) {
      converged <- TRUE
    }

    centers <- new_centers
  }

  list(clusters = clusters, centers = centers, iterations = iter)
}
```

Dans une première partie nous allons appliquer les 2 méthodes de clustering sur les données brutes, ensuite sur les données centrées réduites. Nous allons appliquer la fonction manuelle et la fonction prédefinie sur R et ensuite comparer les résultats.

Clustering sur les données brutes

Kmeans

```
kmeans_result <- kmeans(features, centers = length(unique(labels)))  #
diabetes_data$Cluster_kmeans <- as.factor(kmeans_result$cluster)

# Matrice de confusion
cm1<- table(Predicted = diabetes_data$Cluster_kmeans, Actual = labels)
accuracy_Kmeans <- sum(diag(cm1)) / sum(cm1)
print(accuracy_Kmeans)
```

```
## [1] 0.3398438
```

K-means manuelle

```
kmeans_man <- my_kmeans(features,2)  # Nombre de clusters = classes uniques

# Ajouter les clusters aux données
diabetes_data$Cluster_kmeans_man <- as.factor(kmeans_man$cluster)

# Matrice de confusion pour comparer avec les classes réelles
```

```r
cm_k<- table(Predicted = diabetes_data$Cluster_kmeans_man, Actual = labels)
accuracy_K_man <- sum(diag(cm_k)) / sum(cm_k)
print(accuracy_K_man)
```

```
## [1] 0.3398438
```

Nous remarquons que les resultats sont similaires, cependant l'accuracy est relativement faible. Ceci pourrait etre du au fait que les données ne sont pas équilibrés, il y a plus de tested_negative que de tested_positive.

Clustering hiérarchique

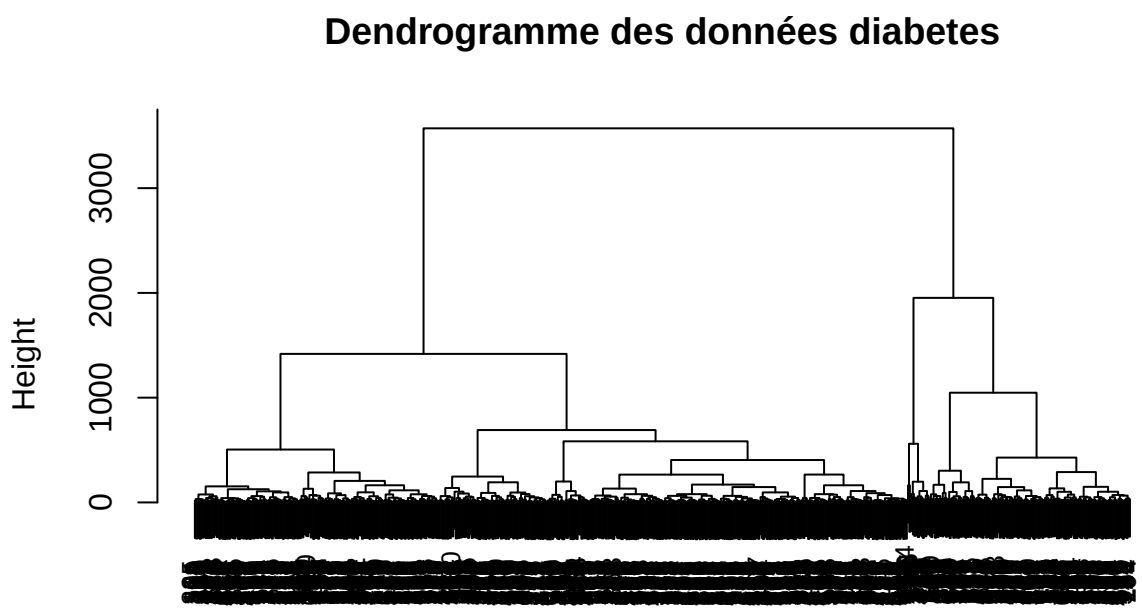
```r
# Calculer la matrice de distance
dist_matrix <- dist(features)

hc <- hclust(dist_matrix, method = "ward.D2")
diabetes_data$Cluster_hierarchical <- as.factor(cutree(hc, k = length(unique(labels))))

# Matrice de confusion
cm12<- table(Predicted = diabetes_data$Cluster_hierarchical, Actual = labels)
accuracy_hierach <- sum(diag(cm12)) / sum(cm12)
print(accuracy_hierach)
```

```
## [1] 0.6757812
```

```r
# Visualiser le dendrogramme
plot(hc, main = "Dendrogramme des données diabetes", sub = "", xlab = "", cex = 0.8)
```



Clustering hiérarchique manuel

```
# Appliquer le clustering hiérarchique
#hc <- hclust_manual(features,method = "complete")

#diabetes_data$Cluster_hierarchical <- as.factor(cutree(hc, k = length(unique(labels))))

# Matrice de confusion
#chc<-table(Predicted = diabetes_data$Cluster_hierarchical, Actual = labels)

#acc<-sum(diag(chc))/sum(chc)
```

L'accuracy du clustering hiérarchique est plus élevée, ce qui démontre sa robustesse.

Clustering sur les données centrées réduites

```
scaled_features <- scale(features)
```

K-means

```
scaled_kmeans <- kmeans(scaled_features, centers = length(unique(labels)))
diabetes_data$Cluster_kmeans2 <- as.factor(scaled_kmeans$cluster)
cm2<-table(Predicted = diabetes_data$Cluster_kmeans2, Actual = labels)
accuracy_kmeans_sc <- sum(diag(cm2)) / sum(cm2)
print(accuracy_kmeans_sc)
```

```
## [1] 0.2942708
```

Nous remarquons que l'accuracy du K-means sur les données scales et centrées est un peu plus faible que celle sur les données brutes, ceci peut être du au fait que la standardisation elimine les relations naturelles entre les individus.

Clustering hiérarchique

```
dist_matrix2 <- dist(scaled_features)
hc2<- hclust(dist_matrix2, method = "ward.D2")

diabetes_data$Cluster_hierarchical2 <- as.factor(cutree(hc2, k = length(unique(labels))))

#Matrice de confusion
cm22<-table(Predicted = diabetes_data$Cluster_hierarchical2, Actual = labels)
accuracy_hier_sc <- sum(diag(cm22)) / sum(cm22)
print(accuracy_hier_sc)
```

```
## [1] 0.3385417
```

De même pour le clustering hiérarchique.

```r
#plot(hc2, main = "Dendrogramme des données diabetes scaled", sub = "", xlab = "", cex = 0.8)
```

Classification avec k-NN

Nous allons d'abord diviser la data en données d'entrainement et données test.

```r
library(class)

set.seed(123)
train_indices <- sample(1:nrow(features), size = 0.7 * nrow(features))
train_data <- scaled_features[train_indices, ]
train_labels <- labels[train_indices]

test_data <- scaled_features[-train_indices, ]
test_labels <- labels[-train_indices]
```

```r
# Choisir une valeur initiale de k
k <- 5
knn_result <- knn(train = train_data, test = test_data, cl = train_labels, k = k)

# Matrice de confusion
confusion_matrix <- table(Predicted = knn_result, Actual = test_labels)
print(confusion_matrix)
```

```
##                 Actual
## Predicted         tested_negative tested_positive
##   tested_negative             131              40
##   tested_positive              19              41
```

La matrice de confusion est plus ou moins précise par rapport aux méthodes précedentes.

```r
# Calculer la précision
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Précision avec k =", k, ":", accuracy, "\n")
```

```
## Précision avec k = 5 : 0.7445887
```

```r
accuracies <- sapply(1:10, function(k) {
  knn_result <- knn(train = train_data, test = test_data, cl = train_labels, k = k)
  confusion_matrix <- table(Predicted = knn_result, Actual = test_labels)
  sum(diag(confusion_matrix)) / sum(confusion_matrix)
})
```

La précision est également très élevée.

```r
# Meilleure valeur de k
best_k <- which.max(accuracies)
cat("Meilleure valeur de k :", best_k, "avec une précision de :", max(accuracies), "\n")
```

```
## Meilleure valeur de k : 4 avec une précision de : 0.7748918
```

Une meilleure accuracy a été detectée avec 4 clusters ce qui est un meilleur résultat.