Groupe 8 : Martin Belan / Luca Le Treust

# DEVOIR D'APPRENTISSAGE STATISTIQUE

## I) Classification non supervisée

Le jeu de données concerne 500 individus pour lesquels 10 variables quantitatives sont mesurées. L'objectif dans cette partie est d'étudier différents algorithmes afin d'obtenir une ou plusieurs possibilités de partitionnement du jeu de données en plusieurs groupes. Il s'agit donc d'un problème de classification non supervisée, où l'objectif est d'envisager les partitions les plus pertinentes.

### 1) Analyse en composantes principales (A.C.P)

Afin de visualiser les données sur le plan, une A.C.P est réalisée. L'objectif est de réduire la dimension par projection sur les composantes principales. Il en résulte une représentation permettant de commencer à approcher la structuration des données (figure 1). Il semblerait globalement que deux ou quatre groupes s'individualisent. Si l'on considère deux groupes, il est possible de différencier un groupe à gauche de l'axe vertical et un autre à droite (ou bien un en haut de l'axe horizontal et un en bas). Si l'on considère quatre groupes, il est possible d'en visualiser un dans chaque partie délimitée par les axes en pointillés. Toutes ces hypothèses ne sont pas exhaustives et l'objectif ensuite sera d'étudier les différentes possibilités de répartition.

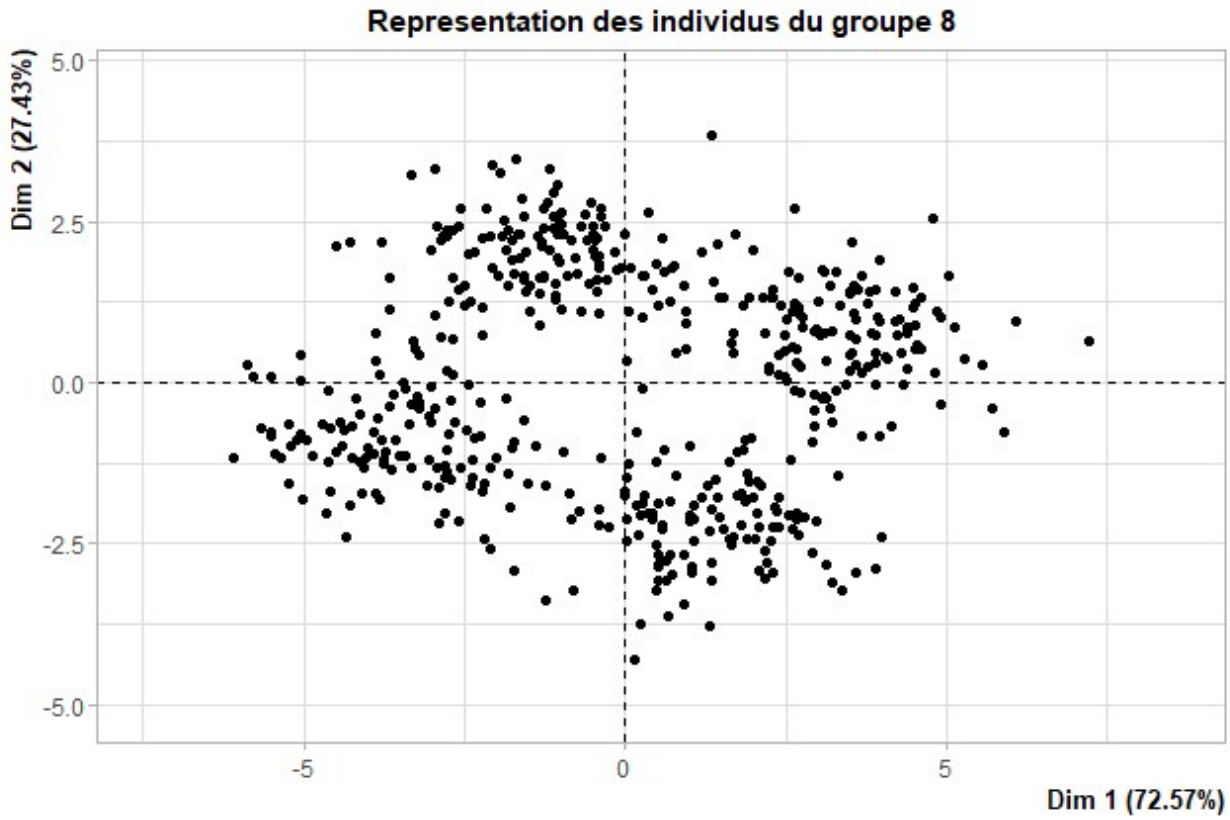


**Figure 1 : Représentation de l'analyse en composantes principales**

## 2) K-means

Une première approche est de considérer l'algorithme des k-moyennes mobiles (k-means).

L'algorithme est répliqué 20 fois pour chaque valeur de groupe envisagée (valeur k), afin d'augmenter la stabilité des résultats. En effet, l'initialisation de l'algorithme peut affecter la convergence aboutissant à des résultats différents. La démarche est donc de répliquer la procédure plusieurs fois, à k fixé, afin d'obtenir un résultat stabilisé. Parmi ces 20 sous-modèles, celui sélectionné est celui dont la variance (ou inertie) intra groupe est minimale. Cette stratégie est répétée pour plusieurs valeurs de k envisagées. L'amplitude des valeurs de k peut être orientée en fonction du nombre de groupes supposés dans ce jeu de données. Selon les hypothèses émises dans la partie ACP, mais par prudence concernant l'incertitude, l'intervalle des valeur entières de k est de 1 à 20 groupes envisagés. La procédure consiste à modéliser 20 sous-modèles à k fixe, de choisir celui qui minimise la variance intra groupe et d'itérer ceci 20 fois, c'est-à-dire pour chaque valeur de k. Ceci conduit à visualiser les variations de variance intra-groupe en fonction de la valeur de k (figure 2).

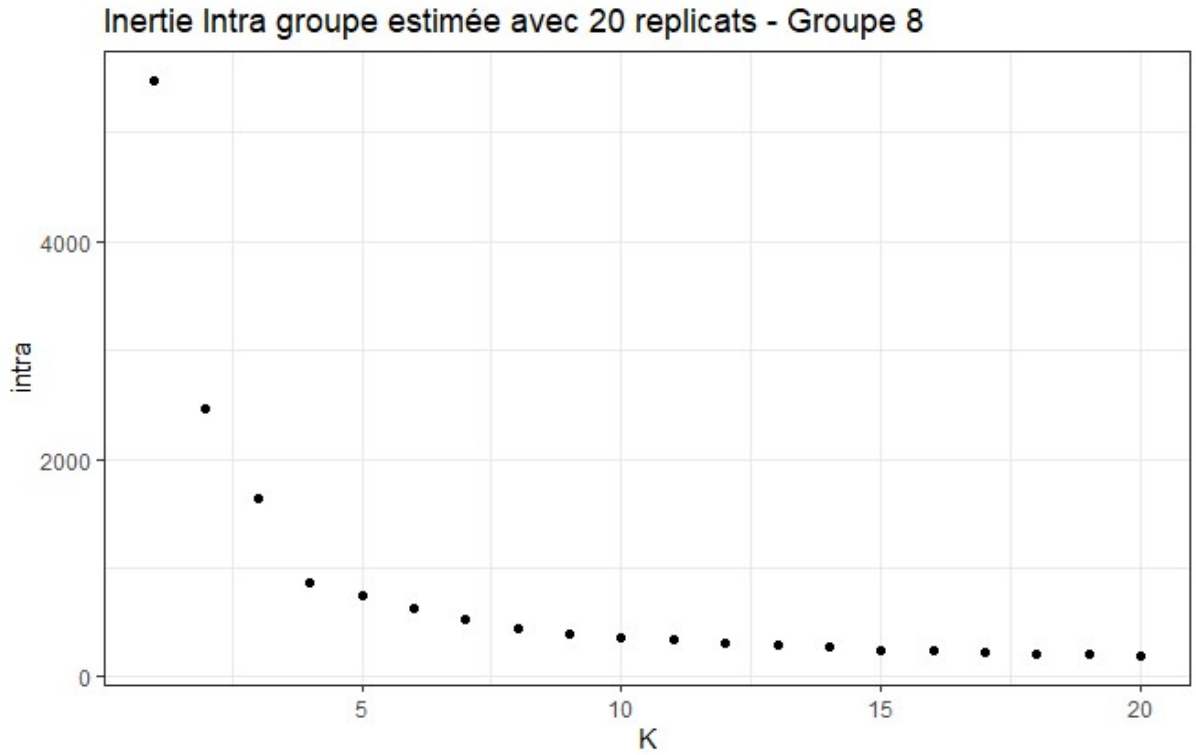


**Figure 2 : Inertie intra-groupe du modèle sélectionné en fonction de la valeur de k**

La courbe montre bien que la variance intra-groupe diminue lorsque k augmente. Il est possible d'observer qu'une très grande diminution de variance est effectuée en passant de un à deux groupes. La valeur de l'inertie peut toutefois être encore améliorée en considérant davantage de groupes. Par exemple, choisir un k de valeur 3 ou 4 permettrait d'obtenir des variances chaque fois plus faible, avec un gain cependant moins important que la variation entre 1 et 2 groupes. Enfin, pour des valeurs de k supérieures à 5, la variance diminue lentement et il n'y a pas de très grande variation comparable aux précédentes.

L'analyse de la courbe de variance suggèrerait par conséquent de sélectionner un nombre de groupes plutôt faible, de l'ordre de 2 à 4, voire 5. L'information apportée par les autres méthodes permettra de renforcer ou d'invalider ces considérations.
Afin de tester la robustesse de cette classification, il est possible de visualiser les A.C.P des résultats des différentes partitions obtenues pour les groupes envisagés (figures 3 et 4).

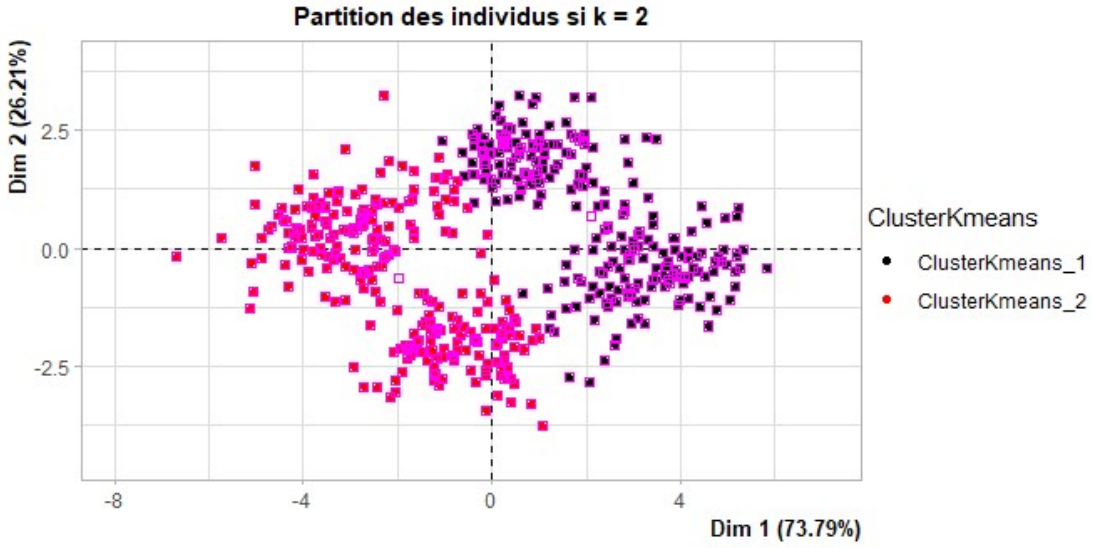


**Figure 3 : Représentation de l'analyse en composantes principales pour une classification par k-means à 2 classes**

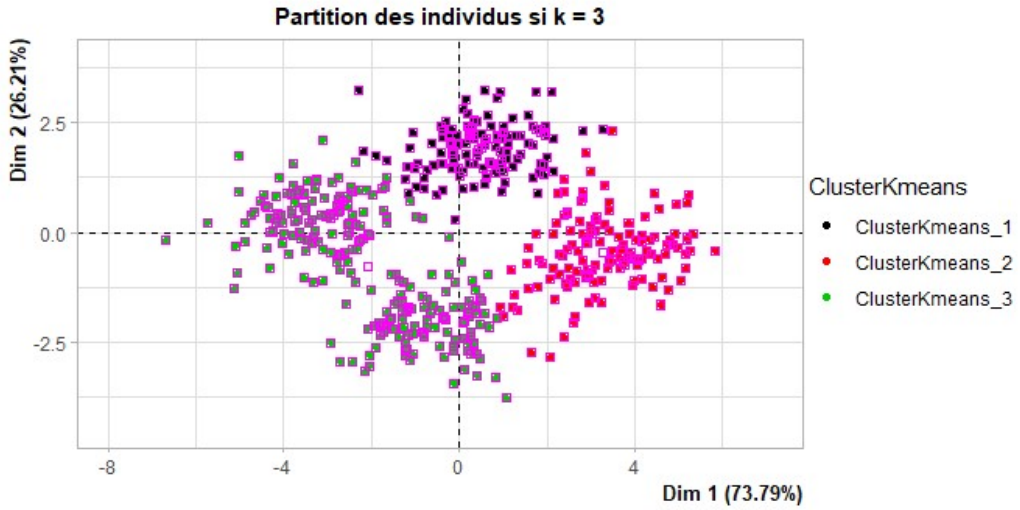


**Figure 4 : Représentation de l'analyse en composantes principales pour une classification par k-means à 3 classes**

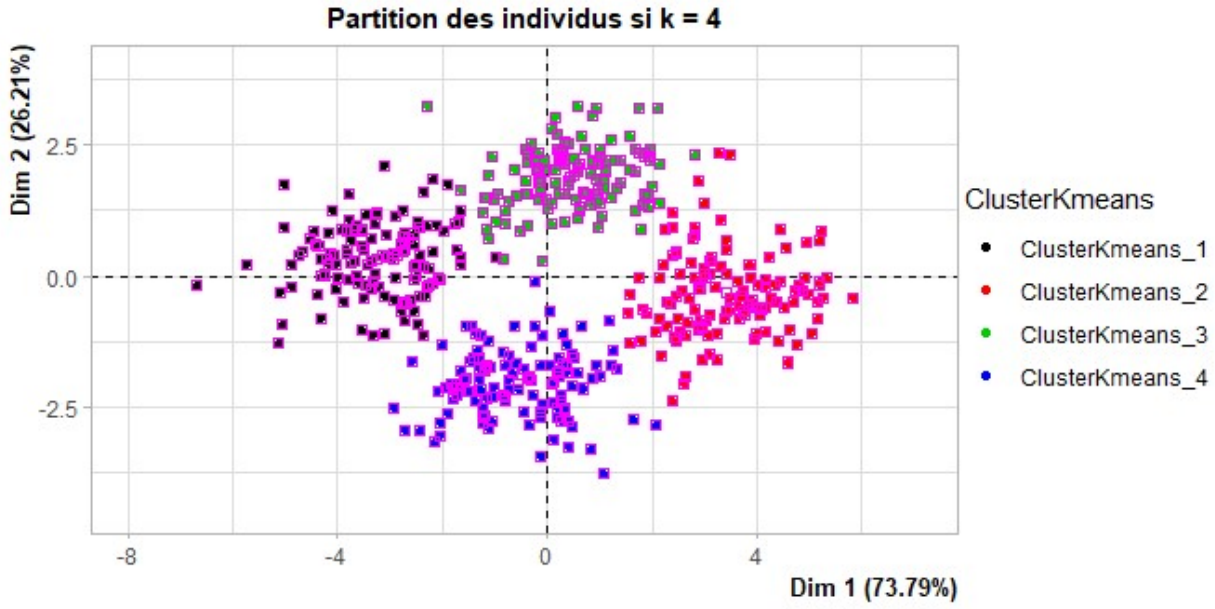


**Figure 5 : Représentation de l'analyse en composantes principales pour une classification par k-means à 4 classes**

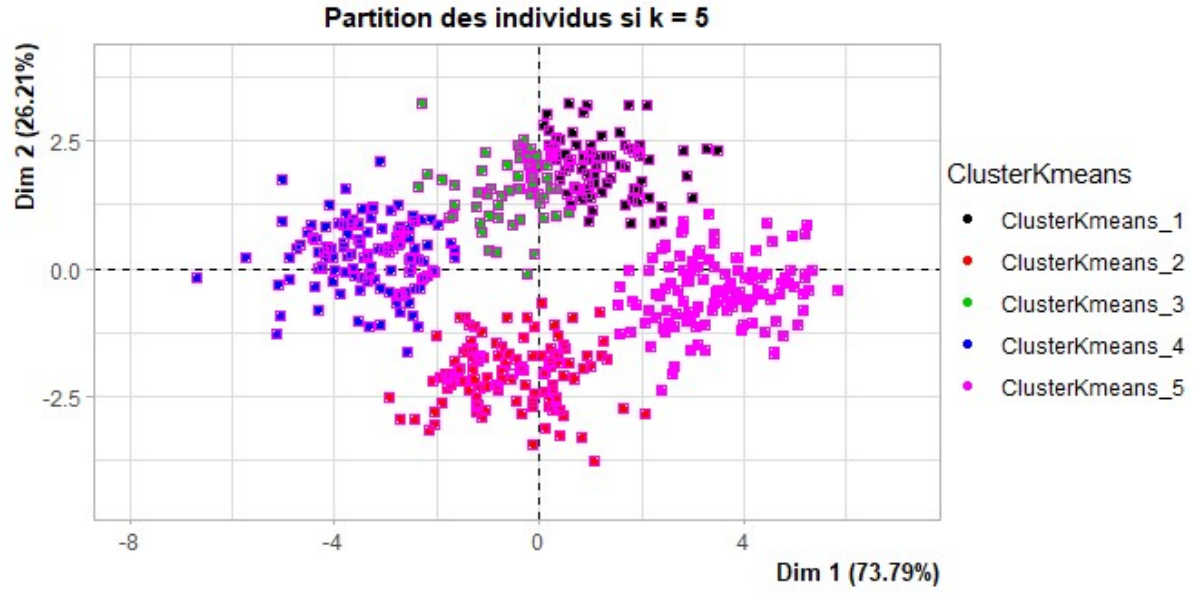


**Figure 6 : Représentation de l'analyse en composantes principales pour une classification par k-means à 5 classes**

L'observation de ces partitions montre que les répartitions sont homogènes malgré la variation du nombre de groupes imposée. En effet, en partant de la partition en deux groupes, et en augmentant la valeur de k, on retrouve que les individus d'un même groupe sont fragmentés en petits sous-groupes qui sont souvent inclus dans l'ensemble du groupe initial plus grand. Par exemple, si l'on regroupe les classes 2 et 3 et les classes 1 et 4 dans la répartition à 4 classes, on retrouve approximativement la répartition en 2 classes. Ce phénomène semble stable pour tout k.

Il en résulte qu'il n'existe pas de variation importante de répartition des sujets en fonction de la valeur de k. L'augmentation du nombre de groupes implique plutôt une sous fragmentation homogène du groupe plus grand associé.

Finalement l'algorithme des k-means semble robuste à la considération du nombre de groupes. Il semble que deux grandes classes soient dissociées et que chacune de ces deux classes puisse éventuellement se fragmenter en petits sous-groupes homogènes. Ces sous-ensembles étant inclus dans les deux grands groupes initiaux, ils ne perturbent pas la dissociation des individus en ces deux classes.

### 3) Classification ascendante hiérarchique

L'algorithme de classification ascendante hiérarchique est utilisé afin d'apporter une information supplémentaire sur le choix du nombre de classes.

La paramètre à faire varier ici est le type de mesure des distances inter-groupes, à métrique d'espace constante par défaut euclidienne. Les mesures envisagées sont, la dissimilarité moyenne (distance moyenne entre les groupes ou « average method »), la dissimilarité maximale (distance entre les points les plus éloignés de chaque groupe ou « complete method »), ou encore la métrique de Ward considérant la distance moyenne avec pondération par les poids des groupes (« Ward method »)[1].

Les résultats de la classification pour chaque type de mesure sont présentés dans les figures 7, 8 et 9.

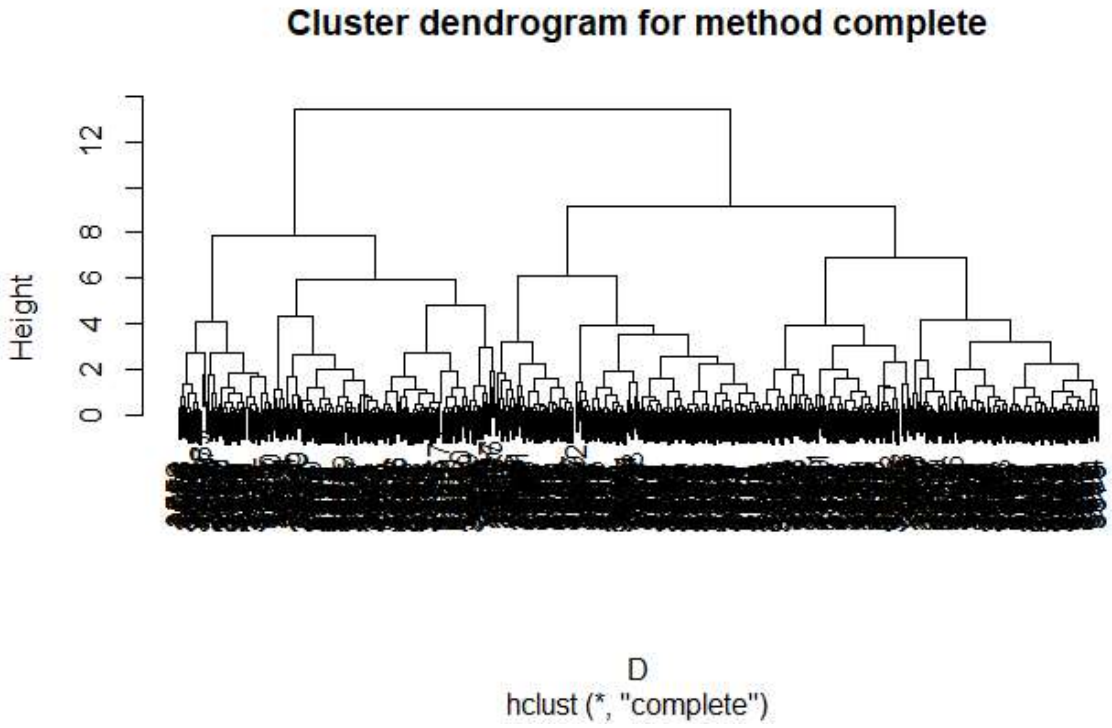


**Figure 7 : Dendrogramme pour la méthode « complete »**

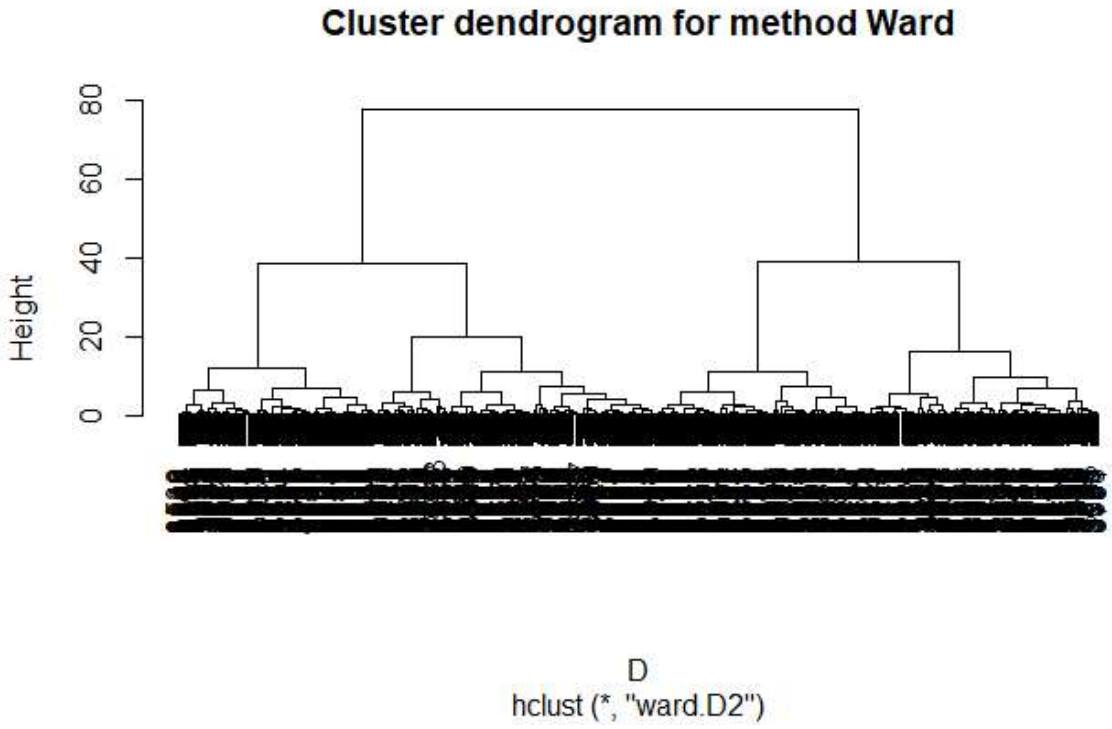


**Figure 8 : Dendrogramme pour la méthode de Ward**

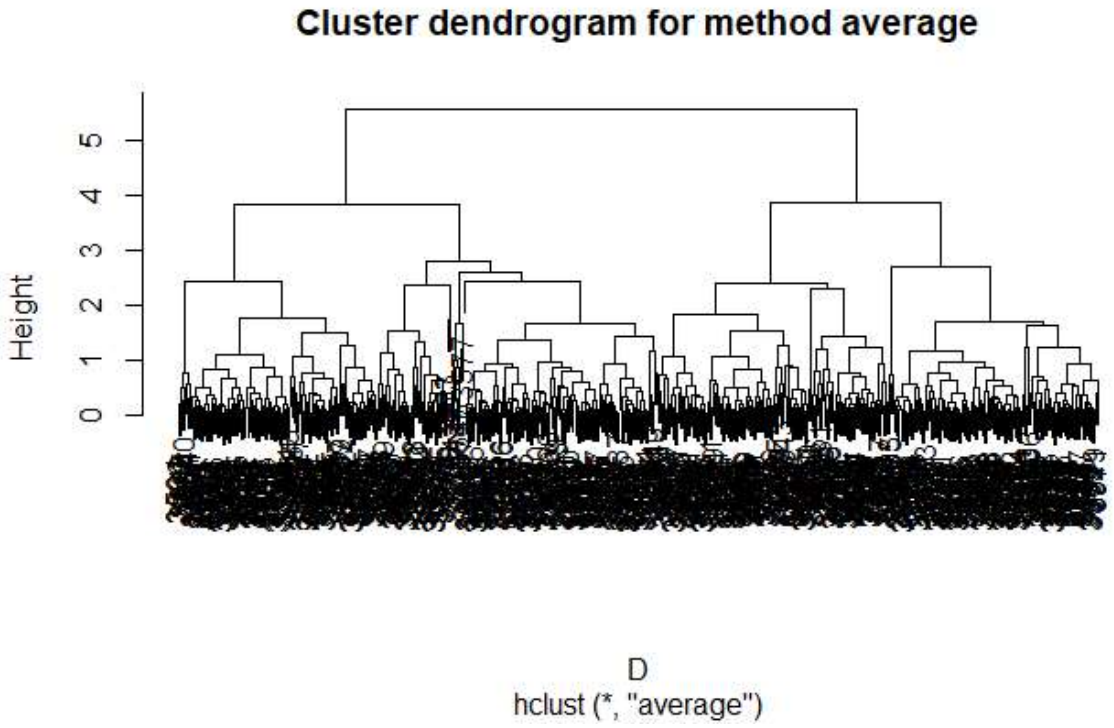


**Figure 9 : Dendrogramme pour la méthode « average »**

Globalement, les plus grandes variations de variance intra-groupe figurent dans les parties supérieures des dendrogrammes. En effet, les hauteurs des branches sont maximales dans ces zones et en particulier pour des coupures correspondant à une partition en deux groupes. Les dendrogrammes suggèrent également qu'une variation importante d'inertie est effectuée en passant à 4 groupes, notamment celui utilisant la métrique de Ward. Enfin, les partitions en 3 ou 5 groupes pourraient être écartée puisqu'elles sont souvent associées à de petites variations d'inertie sur ces dendrogrammes.

On confirme ces variations d'inertie importantes pour 2 ou 4 groupes, plus marquées pour les méthodes « ward.D2 » et « average » que pour la méthode « complete » par le graphique des variations d'inertie en fonction du nombre de classes données par la classification hiérarchique ascendante (figure 10, 11 et 12).

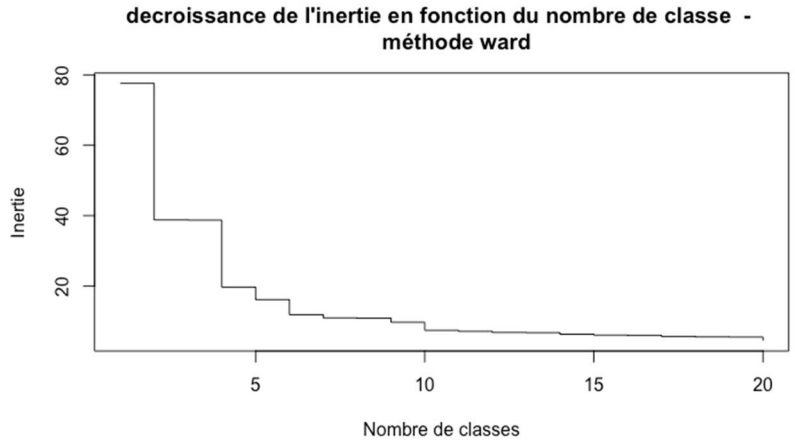


**Figure 10. Evolution de l'inertie en fonction du nombre de classe pour la méthode « ward.D2 »**

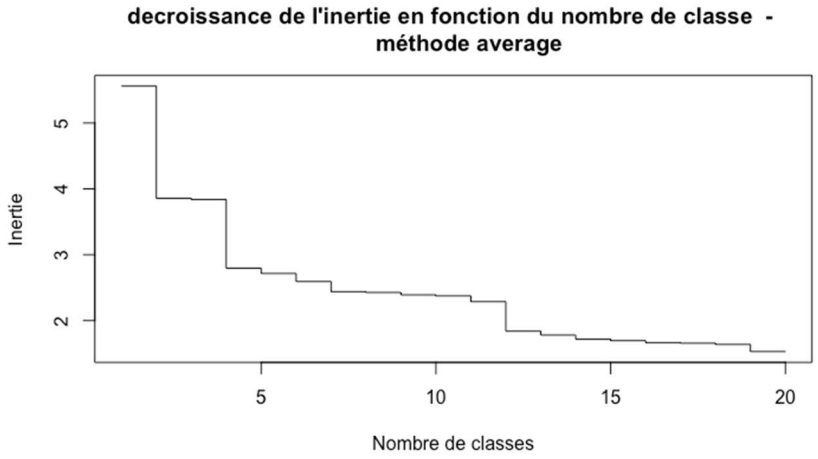


**Figure 11. Evolution de l'inertie en fonction du nombre de classe pour la méthode « average»**

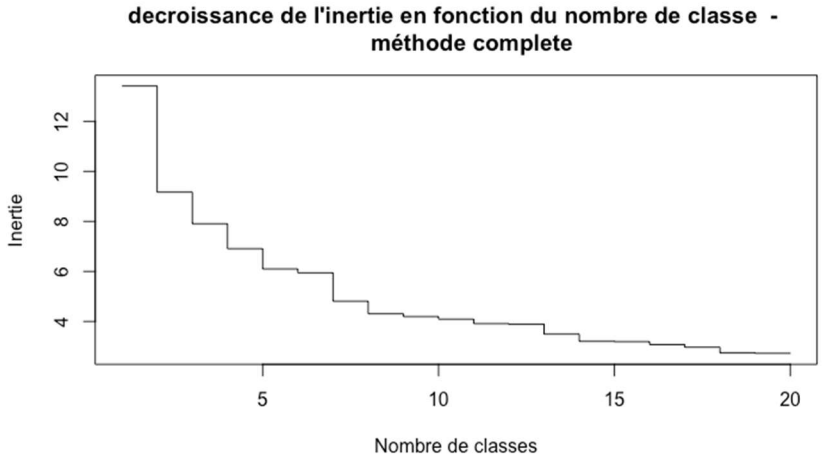


**Figure 12. Evolution de l'inertie en fonction du nombre de classe pour la méthode « complete »**

Ces visualisations graphiques de variation d'inertie confirment nos hypothèses qu'une plus grande homogénéité intra-classe est obtenue pour un nombre de classe de 2 ou 4. Le gain sur l'inertie par augmentation du nombre de classe ensuite est par ailleurs plus faible.

### 4) Modèles de mélanges gaussiens

Les modèles de mélange gaussiens sont une méthode alternative des deux premières présentées pour réaliser de la classification non supervisée basée sur une approche d'itération en 2 étapes « Expectation-Maximization ». La sélection de modèle pour ce type d'approche est basée sur les matrices de covariances en cas de variables multiples et la qualité du modèle est évaluée par le score BIC (Bayesian Information Criterion) dont le but est d'être maximisé.

Afin de choisir le modèle de structure de covariance le plus adapté pour définir le nombre de classes, on compare les modèles en fonction du BIC fourni selon le nombre de classes et selon le type de modèle de covariance utilisé. Pour cela, on utilise la fonction « mclustBIC » puis on représente graphiquement l'évolution du BIC en fonction du nombre de groupe et du modèle de covariance (figure 13).

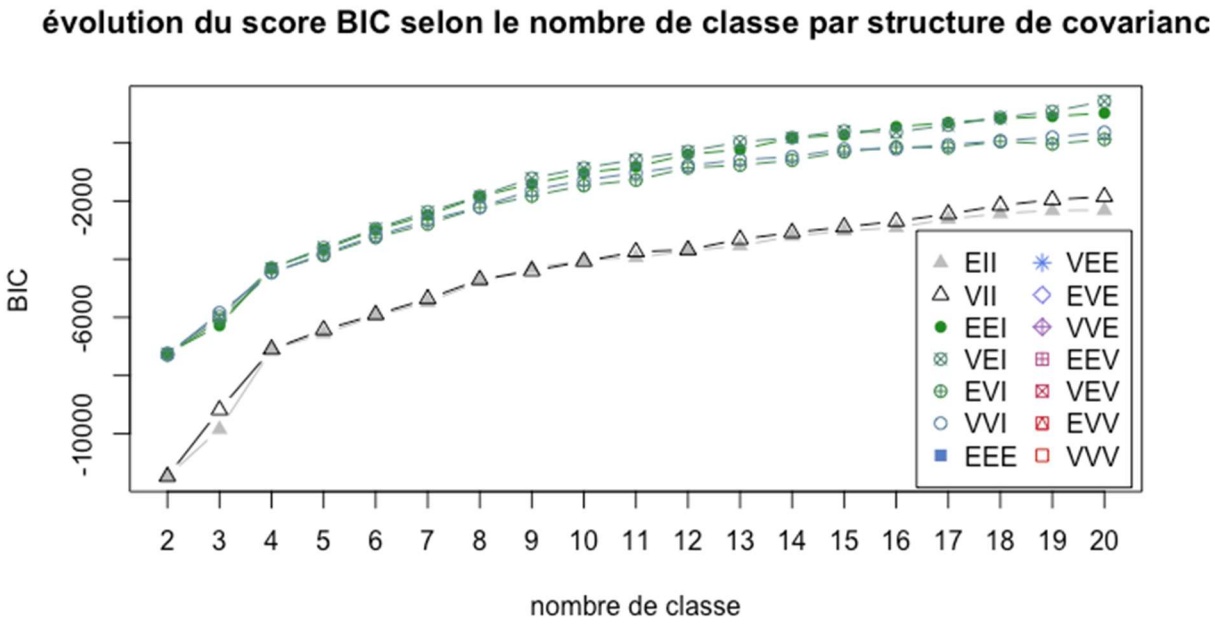


**Figure 13 : evolution du score BIC en fonction du nombre de classe et de la structure de covariance appliquée au modèle**

Seuls 6 modèles ont pu être présentés pour 2 classes ou plus (EII, VII, EEI, VEI, EVI et VVI) et le modèle pour lequel le BIC est maximisé est le modèle « VEI » (volumes variables mais forme identique et sans orientation). On note une variation importante du BIC pour le passage de k = 1 à 2 et de k = 3 à 4 quel que soit le modèle, et notamment vérifié avec le

modèle « VEI » (figure 14). A partir de k=5, le BIC continue d'augmenter mais de manière moins rapide, ne contribuant plus à faire varier le BIC de façon importante. Ces observations sont cohérentes avec les méthodes précédentes utilisées pour déterminer le nombre de classes.

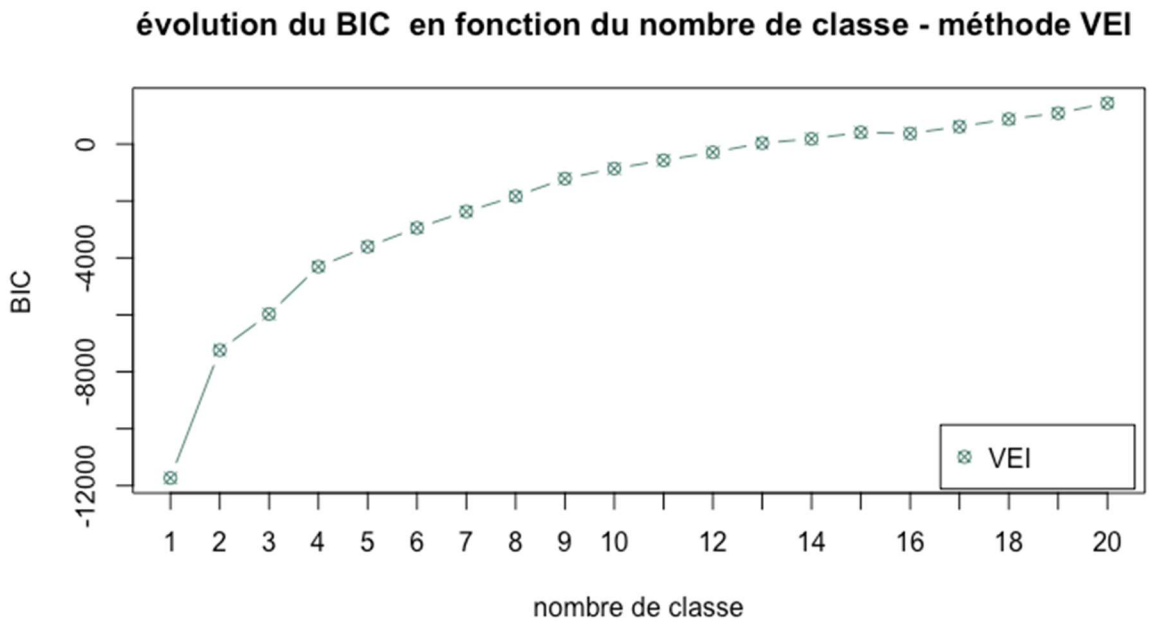


**Figure 14 : évolution du score BIC en fonction du nombre de classe pour la modèle de mélange gaussien (structure de covariance VEI)**

### 5) Comparaison des partitions obtenues en fonctions des modèles

L'objectif est de fixer un nombre de clusters a priori, et sélectionné selon les résultats précédents, afin de comparer les répartitions des individus en fonction des méthodes.

Une visualisation de la classification peut être générée par une ACP sur les données classées par les différentes méthodes. Nous comparerons les partitions entre un modèle de k-means et un modèle de mélange gaussien (avec une structure de covariance type « VEI »), pour une classification en 2 puis en 4 groupes (figure 15A et B, 16A et B ) .

**Figure 15 : Comparaison des représentation de l'analyse en composantes principales pour une classification à 2 classes : A) méthode k-means ; B)mélange gaussien**

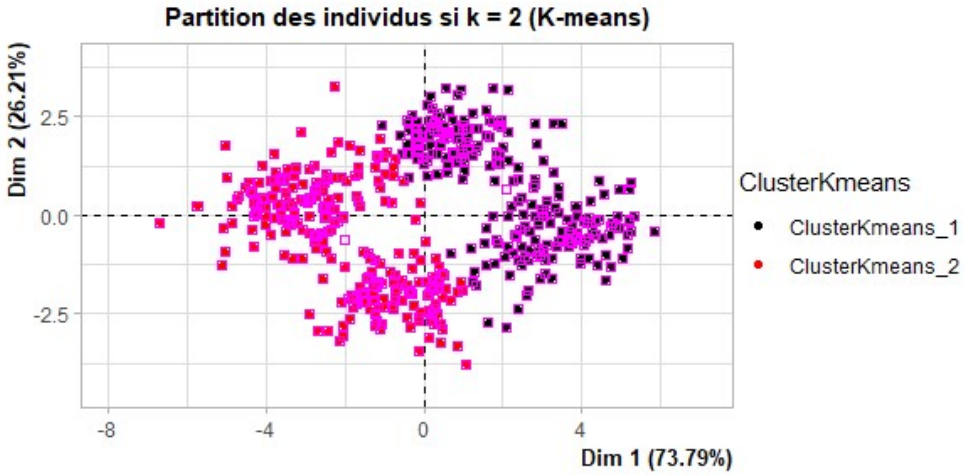


**Figure 15.A**

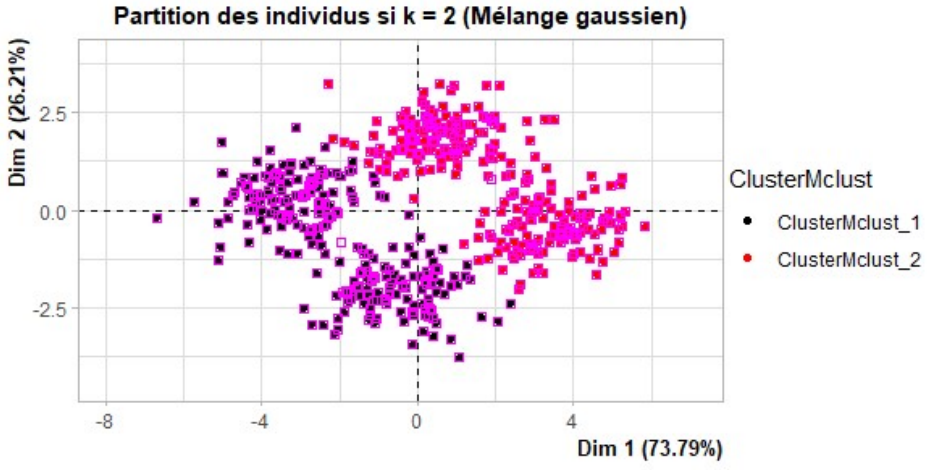


**Figure 15.B**

**Figure 16 : Comparaison des représentation de l'analyse en composantes principales pour une classification à 4 classes : A) méthode k-means ; B)mélange gaussien**

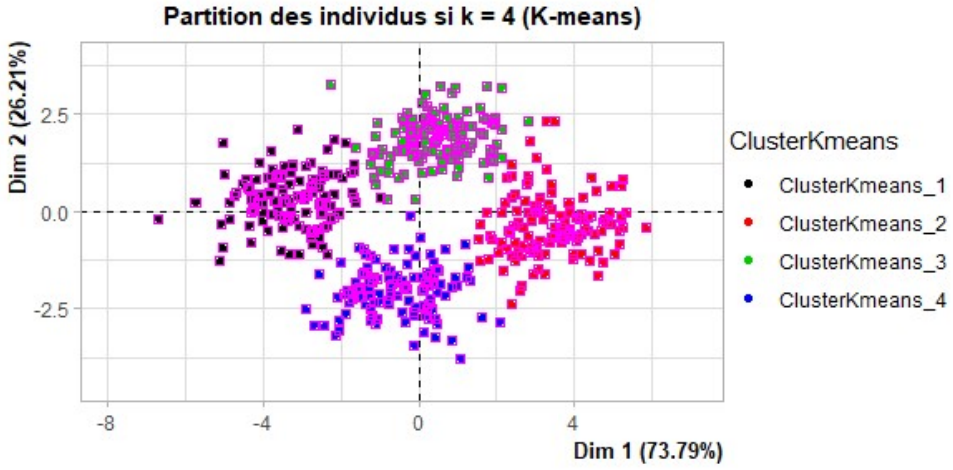


**Figure 16.A**

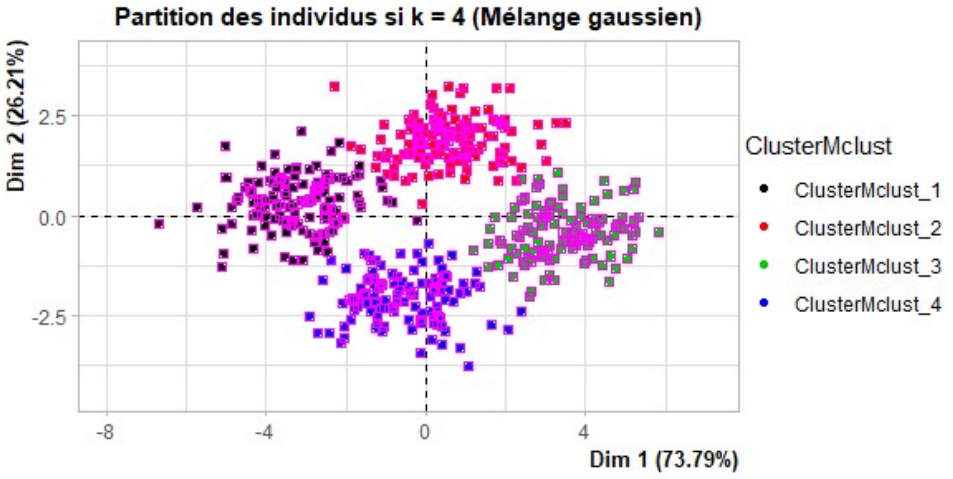


**Figure 16. B**

Finalement, qu'il s'agisse d'une classification en deux ou quatre groupes, les deux modèles convergent vers un résultat homogène de répartition. Il est possible de noter certaines différences, notamment aux frontières de classification, où certains individus peuvent se situer dans l'un ou l'autre groupe en fonction du modèle. Toutefois, globalement la très grande majorité des individus se retrouvent regroupés dans les mêmes classes quel que soit la méthode utilisée entre les k-means ou le modèle de mélange gaussien (type « VEI »). Cette conclusion semble stable selon que l'on considère deux ou quatre classes a priori.

# II) Classification supervisée

Le jeu de données concerne 1000 individus pour lesquels 11 variables quantitatives ont été mesurées, ainsi que qu'une variable binaire « stabf » qui concerne la classe à prédire (stable ou instable). Dans cette partie, l'objectif sera d'étudier les performances de classification des différents modèles et de les comparer entre eux. Les critères de performance seront estimés de manière non biaisée par l'utilisation d'un échantillon de test (70% de l'échantillon total) ou par validation croisée. Il s'agira dans tous les cas d'une mesure du risque d'erreur par le taux de mauvaise classification, estimé donc sur des échantillons de test.

La variable à prédire étant codée de façon binaire conditionnellement à une variable quantitative existante, nous conclurons l'analyse en comparant des modèles de régression sur cette variable à prédire codée de manière continue.

## 1) Analyse en composantes principales

Afin d'étudier la structure des données dans le plan, une analyse en composante principale est réalisée (figure 17). Cela permet de résumer l'information en passant de 11 à 2 dimensions.

Visuellement, il ne semble pas exister d'orientation des structures selon un axe différent que les axes des coordonnées. Les deux classes semblent se superposer, et il est possible de constater que les variances ne semblent pas excessivement hétérogènes selon les deux dimensions, pour chacune des classes. Les volumes et les formes semblent assez similaires.

Ce constat pourra servir d'hypothèse initiale pour appréhender le problème du choix de structure de variance dans les modèles suivants. Toutefois, il ne s'agit que d'une représentation simplifiée en deux dimensions et les structures pourraient en réalité différer. Ces hypothèses seront testées par la suite.

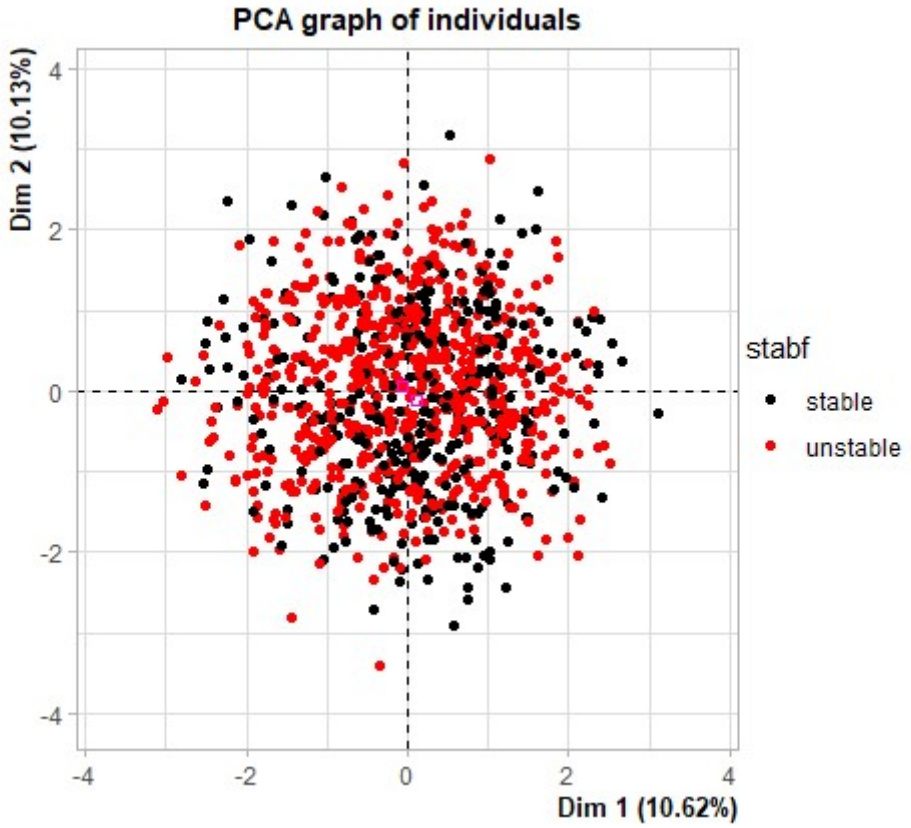


**Figure 17 : Analyse en composantes principales des données selon les classes**

## 2) Analyse factorielle discriminante

D'après les informations provenant de l'ACP, les structures semblent donc assez proches en volumes, formes et orientation entre les deux groupes. Ceci nous permet d'effectuer une analyse discriminante en considérant que la matrice de covariance ne dépend pas du groupe. Nous considèrerons trois modèles, variant par le type de matrice de covariance commune aux deux classes associée : modèle EII (famille sphérique avec variances identiques dans chaque dimension), modèle EEI (pour considérer la différence de variance selon les axes) et le modèle EEE (pour considérer également une différence de variance ainsi qu'une existence de covariance). Ces trois modèles ajustés sont de complexité croissante en termes de paramètres. Ceci implique que l'on va vérifier si la réduction du biais du modèle peut conduire à réduire l'erreur malgré la hausse de l'ajustement. Les erreurs de classification sont évaluées directement par validation croisée à 5 plis (figure 18).

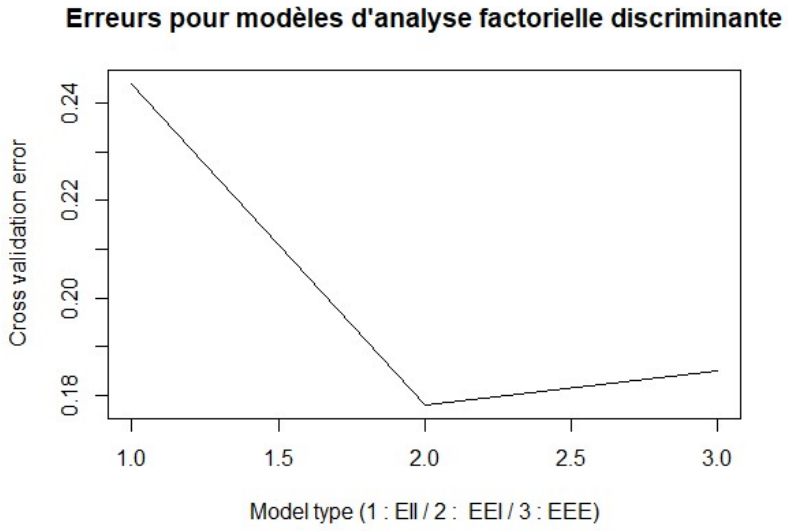


**Figure 18 : Erreurs en validation croisée pour les modèles à matrice de covariance commune**

Le risque d'erreur est minimisé pour le modèle EEI (17,8%) mais l'erreur pour le modèle EEE est très proche (18,5%). Le modèle EII offre quant à lui une erreur bien plus élevée de 24,4%. L'estimation des erreurs par validation croisée permet de sélectionner un modèle EEI conduisant à une erreur estimée à 17.8%.

La méthode de validation par échantillon de test, permet d'obtenir des résultats proches. Les risque d'erreurs sont de 16,3% pour la méthode EEI, 18,3% pour EEE et 22,7% pour EII (figure 19). Quelle que soit la méthode de validation utilisée, le modèle EEI est toujours celui permettant d'obtenir la meilleure classification pour l'analyse factorielle discriminante. Il s'agit des modèles à matrices de covariance identiques pour les classes, sans covariance mais à variance variable selon l'axe.

Nous aurions aussi pu tester les autres modèles, c'est-à-dire ceux qui n'imposent pas une matrice de covariance commune pour les classes (cas non linéaire), avec la même démarche de sélection de modèles.

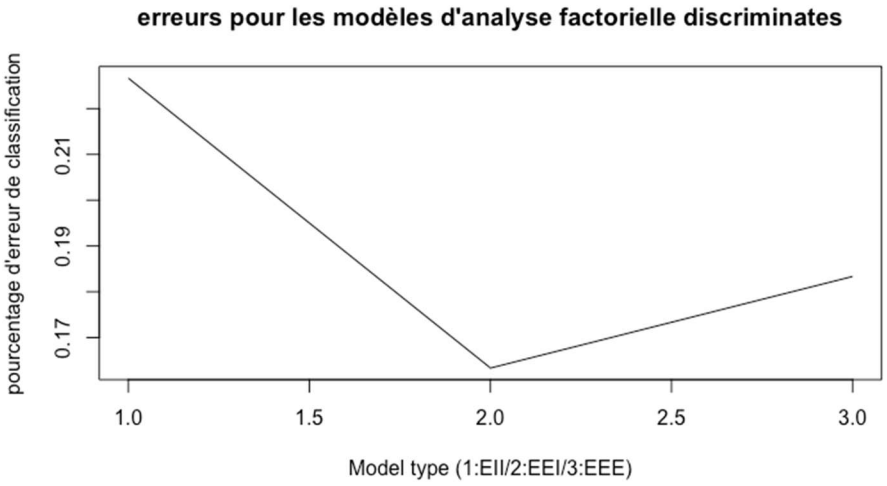


**Figure 19 : Erreurs de classification sur échantillon de validation pour les modèles à matrice de covariance commune**

### 3) K nearest neighbors

Des modèles de k-plus-proche-voisins sont ajustés. L'erreur est estimée par l'échantillon de validation (figure 20) ou par validation croisée (figure 21). Le paramètre à faire varier ici est k, soit le nombre de voisins considérés pour la classification par vote majoritaire. L'intervalle de variation de k est de 1 à 40 par pas entiers. L'erreur est estimée par le taux de mauvaises classifications.

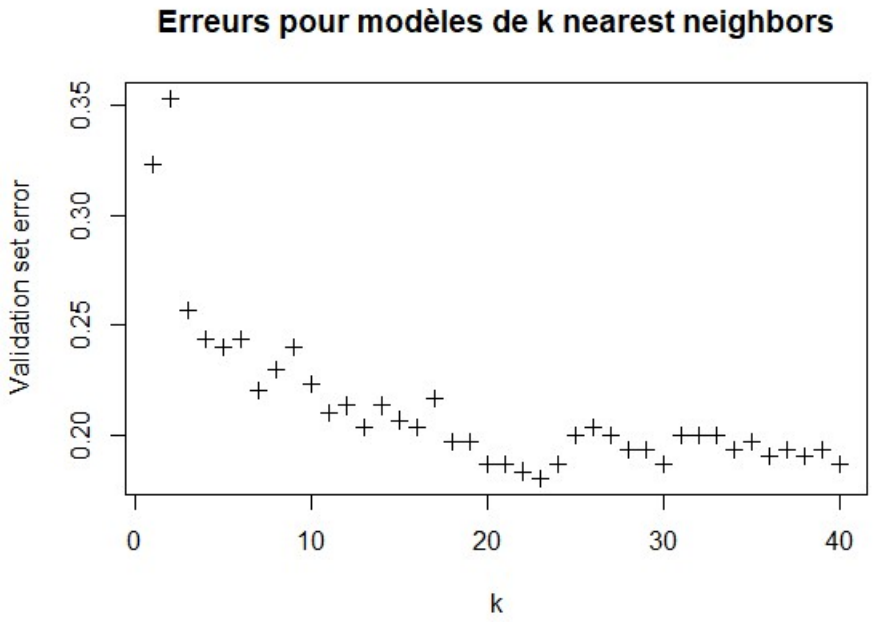


**Figure 20 : Estimation de l'erreur de classification par échantillon de validation selon la méthode KNN**

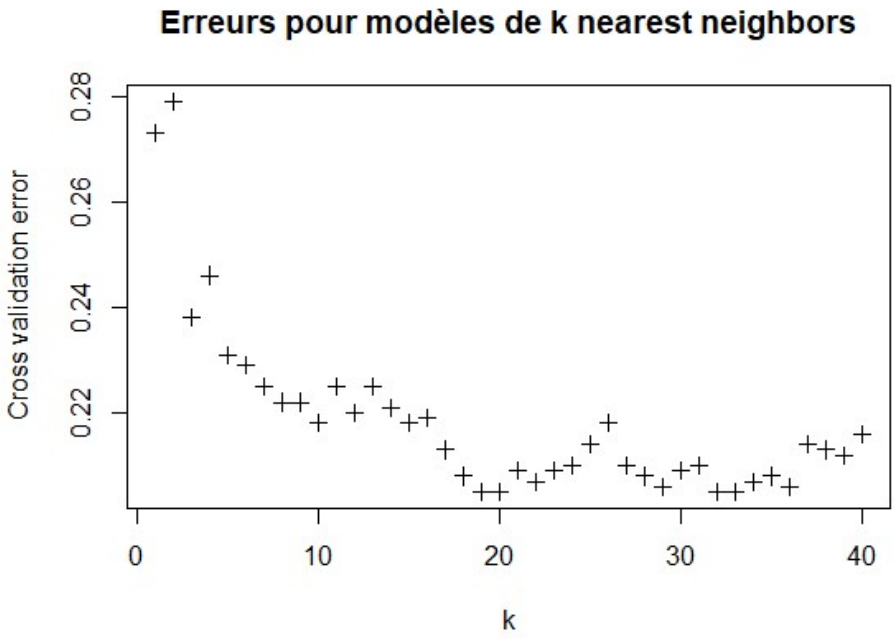


**Figure 21 : Estimation de l'erreur de classification par validation croisée selon la méthode KNN**

Quelle que soit la méthode, l'estimation montre que l'erreur est plutôt élevée pour des valeurs de k faibles (de l'ordre de 30%). Le risque décroit lorsque k augmente. Le taux minimum d'erreur est atteint pour un paramètre proche de 20 voisins et correspond à une erreur de 20,5 % (k = 19 en validation croisée). Des modèles plus parcimonieux conduisent à une erreur assez proche. Pour une valeur de 10, l'erreur est par exemple de 22% et le gain de performance n'est très important par la suite. Finalement, le gain de performance s'observe même à partir d'une valeur de k proche de 5 ou 6 (erreur d'environ 23%).

Nous retiendrons finalement un modèle à 19 plus proches voisins conduisant à une erreur estimée en validation croisée de 20.5%.

### 4) Arbres de classification

Concernant les arbres de classification, un premier paramètre à faire varier est le nombre de feuilles terminales. En effet, un arbre très ajusté donc moins biaisé, peut conduire éventuellement à une erreur de test plus élevée. Tout d'abord, l'arbre le plus ajusté est modélisé. Il contient 31 feuilles terminales et son erreur estimée sur l'échantillon de test est de 24%. L'objectif est de déterminer la valeur du paramètre de taille de l'arbre qui minimise l'erreur de classification. Les erreurs sont estimées par validation croisée à 10 plis avec la librairie « tree » de R mesurant l'erreur par calcul de la déviance (figure 22).

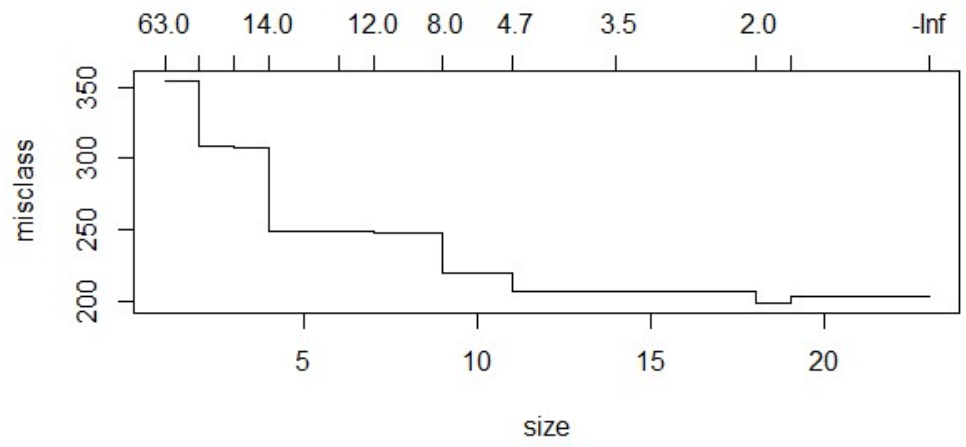


**Figure 22 : Erreur de classification par la mesure de la déviance et en validation croisée en fonction de la taille de l'arbre de classification**

On constate que l'erreur est minimisée pour une taille d'arbre correspondant à 18 feuilles terminales. L'erreur de classification correspondante estimée sur l'échantillon de test est passée à 22.3%. Ce modèle comporte donc moins de feuilles terminales, donc un biais légèrement plus élevé que l'arbre complet mais permet de faire réduire l'erreur de test.

Il est également possible d'utiliser la technique du Boosting, qui consiste à tirer au sort des échantillons d'apprentissage par bootstrapping avec une probabilité plus élevée, au fur et à mesure qu'augmente le nombre de boost, de tirer les observations mal prédites. Ceci permet donc d'améliorer la pertinence du modèle sur les erreurs de classification. On constate , sur la figure 23, que dès 5 boosts, la probabilité d'erreur de classification est améliorée, et qu'elle se stabilise à partir de 20 boosts (11.7% vs 24% pour l'arbre seul non élagué).

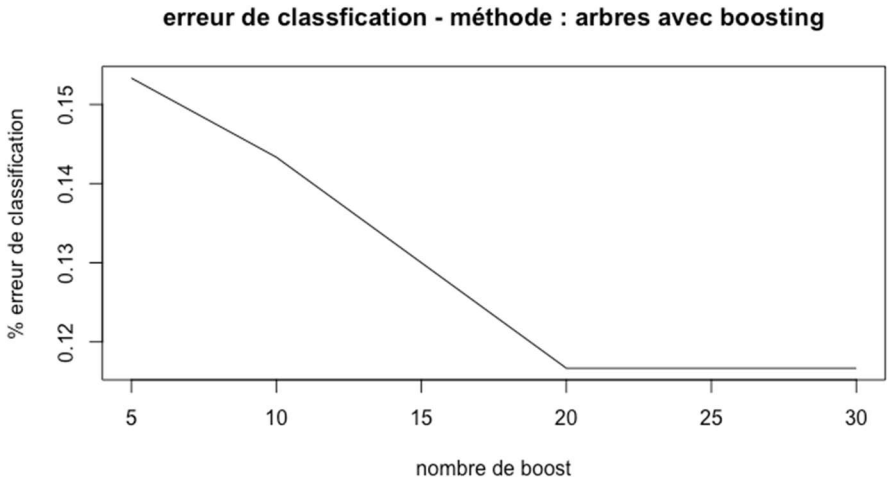


**Figure 23 : Erreur de classification par la méthode de l'arbre de classification en fonction du nombre de boost**

## 5) Random forest

Finalement, le dernier algorithme utilisé est une forêt aléatoire. Un des principaux paramètres à faire varier pour sélectionner un modèle est le nombre d'arbres. Nous avons fixé une amplitude de 100 à 3000 arbres par pas de 200 arbres, conduisant à ajuster 15 modèles différents. Dans le cas particulier de cet algorithme, toutes les données sont utilisées dans l'ajustement du modèle puisque par définition, la procédure conduira à re-échantillonner les observations pour chaque arbre construit. Les erreurs sont donc directement estimées sur les individus « out of bag » (OOB), non concernés par l'ajustement de chaque arbre ajusté.

Par défaut l'algorithme est fixé à 3 prédicteurs tirés au sort (sur les 11) pour la division à chaque nœud ce qui est théoriquement optimal. Seul le paramètre du nombre d'arbres a varié et l'erreur est estimée par les OOB (figure24)

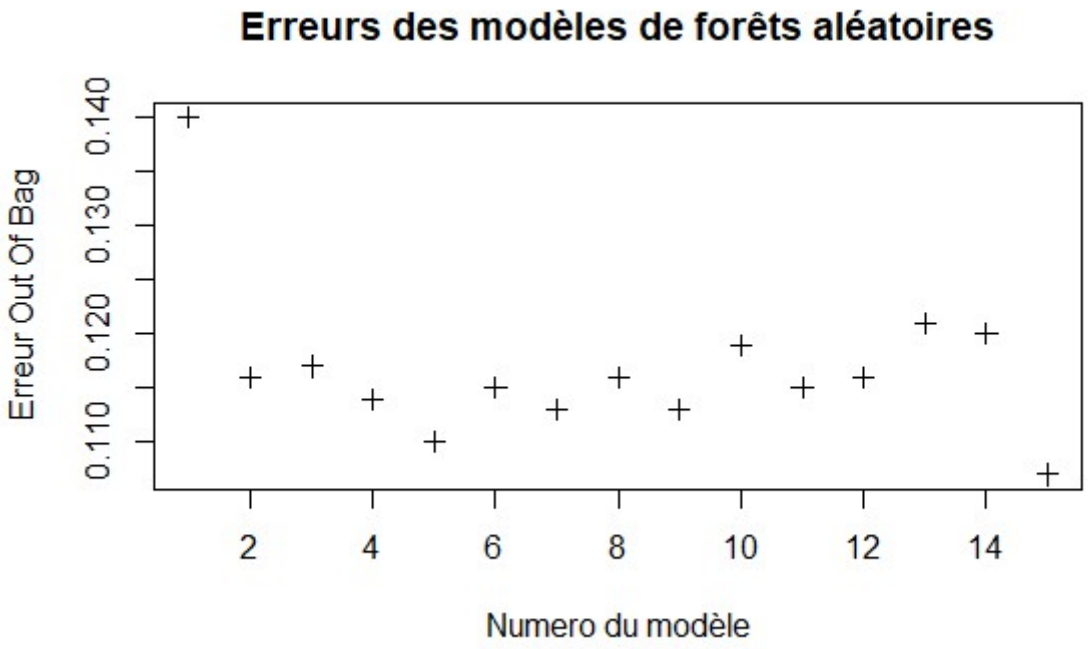


**Figure 24. : Estimation des erreurs « out of bag » des modèles de forêt aléatoires**

Tout d'abord, le premier modèle à 100 arbres conduit à une erreur de 14%. Ce risque diminue ensuite pour le modèle de 300 arbres à environ 11.5%. L'ajout du nombre d'arbre au-delà de 300 n'apporte pas forcément de très nette amélioration de performance. Le modèle retenu est celui minimisant l'erreur, c'est-à-dire celui correspondant au modèle 5 soit 900 arbres et conduisant à une erreur de 11%. Notons que cette erreur de 11% constitue la valeur de l'erreur la plus faible parmi tous les modèles considérés jusqu'à présent.

L'ajustement du modèle optimal, c'est-à-dire pour un nombre d'arbres à 900 et un nombre de prédicteur de 3, apporte une information quant à l'importance de l'utilisation des variables dans la prédiction. Les variables sont classées par ordre d'importance : Tau4

(70.0%), Tau1 (59.4%), Tau2 (55.9%), Tau3 (54.6%), G3 (43.2%), G2 (43.18%), G4 (39.4%), G1 (38.0%), P2 (17.9%), P3 (17.8%) et enfin P4 (16.7%). Ces proportions représentent les moyenne de réduction de l'indice de Gini, c'est-à-dire la contribution au gain d'homogénéité.

## 6) Comparaison des performances de deux modèles de régression

Pour terminer, une autre approche pourrait consister à envisager la variable à prédire, non plus comme un critère binaire mais comme un critère continu et établir des modèles de régression. En effet, la variable « stabf » a été convertie en facteur binaire sur la base d'un critère continu. Nous nous proposons donc d'établir deux modèles de régression pour les comparer entre eux. L'erreur sera estimée par validation croisée ou échantillon de test via la mesure de la moyenne de l'écart au carré entre prédiction et observation (« mean squared error » MSE ou « mean squared residuals » MSR).

- ***Forêts aléatoires***

10 modèles de forêts aléatoires sont entrainés, pour un nombre d'arbre variant de 100 à 2000 par pas de 200. L'estimation directe de l'erreur « out of bag » est effectuée pour chaque modèle, ce qui conduit à sélectionner le modèle optimal (figure 25)

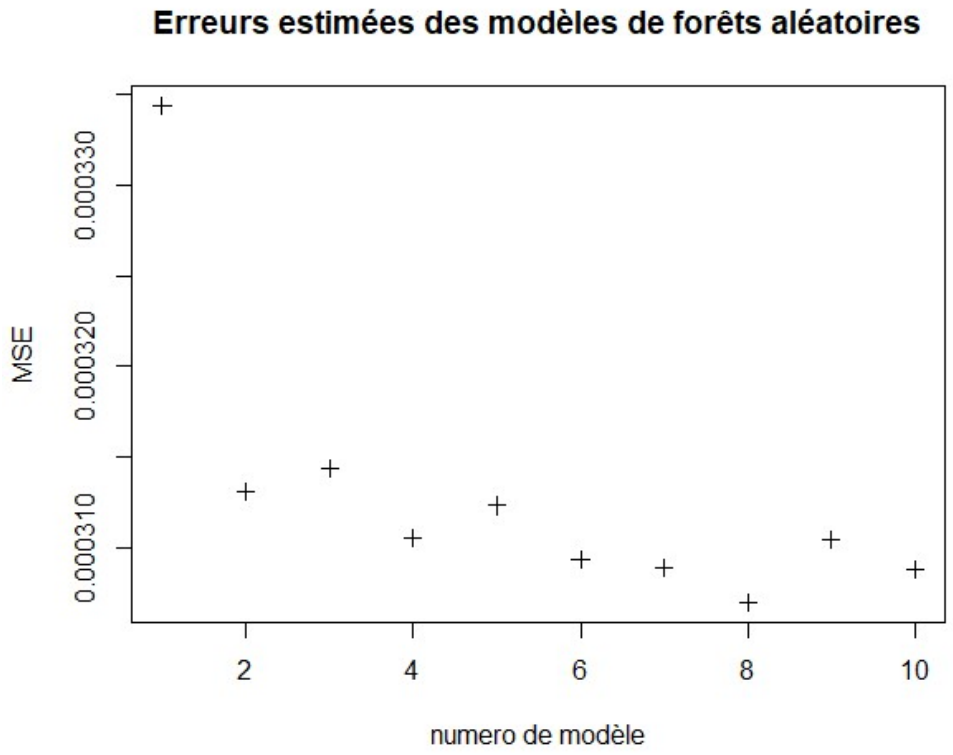


**Figure 25 : erreurs mesurées sur MSE par modèle de regression de forêts aléatoires**

Le modèle candidat contient donc 1500 arbres (MSE = 0.00031). Notons que même si cette erreur parait faible, elle n'est pas nulle. Les valeurs de la variable « stab » sont en effet très proches de zéro (figure 26).

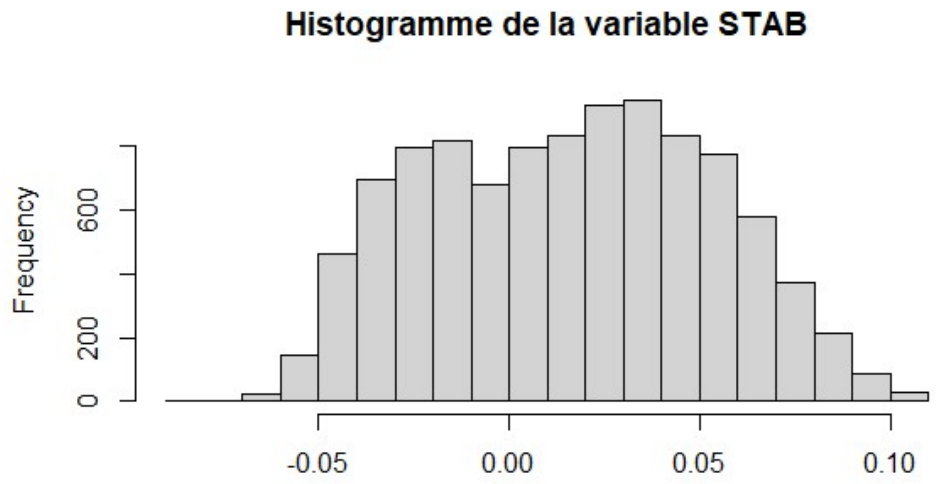


**Figure 26 : distribution de la mesure de la variable STAB**

- ***Régression linéaire***

E (stab | Xj ) ~ Tau1 + Tau2 + Tau3 + Tau4 + P2 + P3 + P4 + GA + G2 + G3 + G4

Le modèle est sélectionné par un algorithme « forward » et par le critère Cp de Mallows. Il s'agit d'un critère pénalisé qui prendre en compte le nombre de prédicteurs, et qui est à minimiser pour sélectionner le nombre de prédicteurs (figure 27).

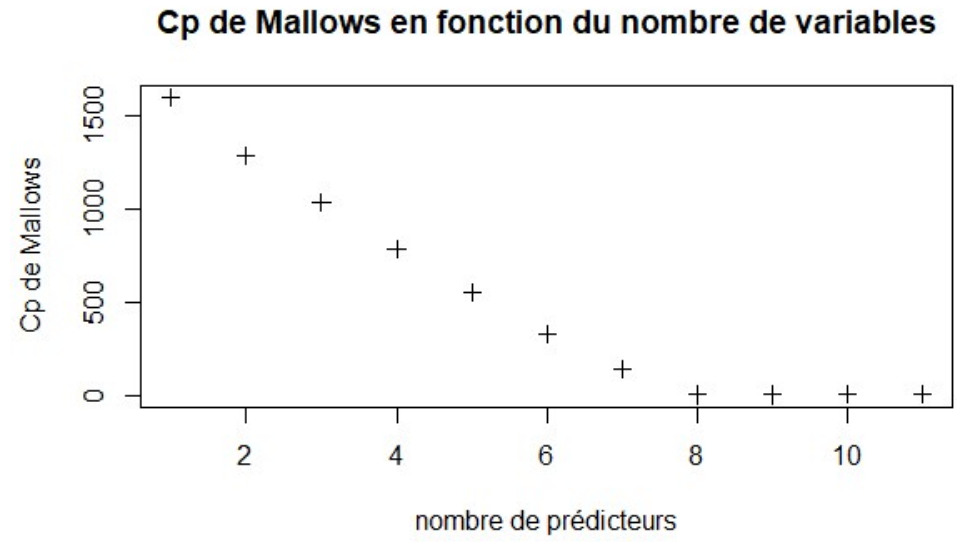


**Figure 27 : Sélection du modèle linéaire par minimisation du Cp de Mallows**

Il est possible de retenir un modèle minimisant le Cp de Mallows et le modèle complet à 11 variables convient. Après ajustement sur les données d'entrainement, l'erreur est évaluée sur l'échantillon test de donne un MSE de 0.00039.

Finalement, le modèle linéaire conduit à une erreur plus élevée d'un facteur 1,25 par rapport au modèle de Random Forest (MSE_lm = 0.00039 et MSE_rf = 0.00031).

## BILAN : Cas supervisé

Finalement, les modèles optimaux pour chaque algorithme ont été sélectionnés. Il est désormais possible de comparer quantitativement les erreurs entre les modèles puisqu'elles ont été évaluées de façon non biaisée. Les résultats sont présentés en figure 28.

| Modèle | Paramètres | Erreur de classification |
|---|---|---|
| Arbre simple de classification | 18 feuilles terminales | 22.3% |
| K plus proches voisins | 19 voisins | 20.5% |
| Analyse factorielle discriminante | Structure de covariance « EEI » | 17.8% |
| Boosting | 5 boosts | 11.7% |
| Forêt aléatoire | 900 arbres | 11.0% |

**Figure 28 : tableau comparatif des performances des modèles optimaux**

Au total, on constate que même en faisant varier les paramètres permettant d'optimiser ces différents modèles, les taux d'erreur de classification pour les méthodes d'arbre de classification, des K plus proches voisins et de l'analyse factorielle discriminante restent élevés (entre 18 et 23%) en comparaison avec la méthode des forêts aléatoires. Cette dernière méthode minimise le risque d'erreur de classification à 11%, permettant de confirmer la robustesse de cette méthode pour la classification supervisée.

# III) Conclusion générale

Finalement, nous avons constaté les avantages à utiliser différents modèles dans les cas de classification supervisée et non supervisée. La combinaison des informations apportées par l'exploration des différentes méthodes non supervisées, permet d'envisager un ou des nombres de classes probables et d'évaluer la robustesse de ces hypothèses. Dans le cas supervisé, elle permet de quantifier les risques d'erreurs non biaisées commises par les modèles, et de sélectionner ainsi des modèles optimaux permettant de minimiser cette erreur, parfois au prix d'une interprétation plus limitée face à la complexité des méthodes.

Notons enfin, que nous n'avons pas choisi d'effectuer de standardisation des données, ce qui aurait pu être pertinent, notamment pour certains algorithmes construits sur des calculs de distances géométriques (k plus proches voisins, k moyennes mobiles, classification hiérarchique par exemple).

## Bibliographie

1 : Trevor Hastie, Robert Tibshirani et Jerome Friedman, *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*, 2009, 2e éd. (1re éd. 2001)

## Code R

Voir fichier joint

```
# ----------- DEVOIR - groupe 8 -----------

set.seed(8)

library(tidyverse)


### CLASSIFICATION NON SUPERVISEE


library(data.table)
DonneesNSGpe <- fread("DonneesNSGpe.txt", data.table = F)
head(DonneesNSGpe, n = 4)

Nbgroupe <- 8
DonneesNS <- DonneesNSGpe[DonneesNSGpe$Groupe == Nbgroupe, -1]
dim(DonneesNS)
dfU <- DonneesNS

  # standardisation

dfU_scaled <- dfU
for (i in 1:10){
  dfU_scaled[,i] <- (dfU_scaled[,i] - mean(dfU_scaled[,i])) /
sd(dfU_scaled[,i])
}


  ###### Analyse en composantes principales : ACP

seed <- 8
library(FactoMineR)
resPCA <- PCA(dfU, scale.unit = FALSE, graph = FALSE)
PPCA <- plot.PCA(resPCA, choix = "ind", label = "none", axes = c(1, 2)) +
  ggtitle(paste("Representation des individus du groupe", seed))
PPCA

resPCA_scaled <- PCA(dfU_scaled, scale.unit = FALSE, graph = FALSE)
PPCA_scaled <- plot.PCA(resPCA_scaled, choix = "ind", label = "none", axes
= c(1, 2)) +
  ggtitle(paste("Representation des individus du groupe", seed))
PPCA_scaled


  ##### K means


intra <- rep(NA, l = 20)

for (k in 1:20) {
  intra[k] <- min(replicate(20,kmeans(dfU,k)$tot.withinss))
}


plot(1:20, intra)

PKmeansK <- ggplot(cbind.data.frame(K = 1:20, Intra = intra), aes(K,
intra)) +
```

```
  geom_point() +
  ggtitle(paste0("Inertie Intra groupe estimée avec 20 replicats - Groupe
", seed)) +
  theme_bw()
PKmeansK


    # K means : PCA si k=2

kmeans.rep <- replicate(20, kmeans(dfU, 2))
kmeans.rep
best <- which.min(unlist(kmeans.rep['tot.withinss', ]))
best
resKmeans <- kmeans.rep[, best]
resKmeans$cluster

resPCACluster <- PCA(cbind.data.frame(ClusterKmeans =
as.factor(resKmeans$cluster),

                                      dfU),
                     scale.unit = FALSE,
                     graph = FALSE,
                     quali.sup = 1:2)

PKmeans <- plot.PCA(resPCACluster, choix = "ind", label = "none", axes =
c(1, 2), habillage = 1) +
  ggtitle(paste("Partition des individus si k = 2"))
PKmeans



  # K means PCA si k=3

kmeans.rep <- replicate(20, kmeans(dfU, 3))
kmeans.rep
best <- which.min(unlist(kmeans.rep['tot.withinss', ]))
best
resKmeans <- kmeans.rep[, best]
resKmeans$cluster

resPCACluster <- PCA(cbind.data.frame(ClusterKmeans =
as.factor(resKmeans$cluster),

                                      dfU),
                     scale.unit = FALSE,
                     graph = FALSE,
                     quali.sup = 1:2)

PKmeans <- plot.PCA(resPCACluster, choix = "ind", label = "none", axes =
c(1, 2), habillage = 1) +
  ggtitle(paste("Partition des individus si k = 3"))
PKmeans



  # K means PCA si k=4

kmeans.rep4 <- replicate(20, kmeans(dfU, 4))
```

```
kmeans.rep4
best4 <- which.min(unlist(kmeans.rep4['tot.withinss', ]))
best4
resKmeans4 <- kmeans.rep4[, best4]
resKmeans4$cluster

resPCACluster4 <- PCA(cbind.data.frame(ClusterKmeans =
as.factor(resKmeans4$cluster),

                                       dfU),
                      scale.unit = FALSE,
                      graph = FALSE,
                      quali.sup = 1:2)

PKmeans4 <- plot.PCA(resPCACluster4, choix = "ind", label = "none", axes =
c(1, 2), habillage = 1) +
  ggtitle(paste("Partition des individus si k = 4"))
PKmeans4


  # K means PCA si k=5

kmeans.rep <- replicate(20, kmeans(dfU, 5))
kmeans.rep
best <- which.min(unlist(kmeans.rep['tot.withinss', ]))
best
resKmeans <- kmeans.rep[, best]
resKmeans$cluster

resPCACluster <- PCA(cbind.data.frame(ClusterKmeans =
as.factor(resKmeans$cluster),

                                       dfU),
                      scale.unit = FALSE,
                      graph = FALSE,
                      quali.sup = 1:2)

PKmeans <- plot.PCA(resPCACluster, choix = "ind", label = "none", axes =
c(1, 2), habillage = 1) +
  ggtitle(paste("Partition des individus si k = 5"))
PKmeans




  ##### CAH : classification ascendante hierarchique

      # CAH : distance inter groupes complete

XX <- as.matrix(dfU)
?dist
D <- dist(XX) # euclidienne
?hclust
H1 <- hclust(D)
H1
plot(H1, main ="Cluster dendrogram for method complete")
```

```
      # CAH : distance inter groupes ward D2


H2 <- hclust(D, method = "ward.D2")
plot(H2, main = "Cluster dendrogram for method Ward")


    # CAH : distance inter groupes single


H3 <- hclust(D, method = "single")
plot(H3, main = "Cluster dendrogram for method single")


    # CAH distance inter groupes average


H4 <- hclust(D, method = "average")
plot(H4, main = "Cluster dendrogram for method average") ### k = 2 ???
voire 4


    # Comparaison de l'inertie  :

inertieW <- sort(H2$height, decreasing = T)
plot(inertieW[1:20], type = "s", xlab = "Nombre de classes", ylab =
"Inertie",
     main = "decroissance de l'inertie en fonction du nombre de classe  -
     méthode ward")


inertieC <- sort(H1$height, decreasing = T)
plot(inertieC[1:20], type = "s", xlab = "Nombre de classes", ylab =
"Inertie",
     main = "decroissance de l'inertie en fonction du nombre de classe  -
     méthode complete" )


inertieA <- sort(H4$height, decreasing = T)
plot(inertieA[1:20], type = "s", xlab = "Nombre de classes", ylab =
"Inertie",
     main = "decroissance de l'inertie en fonction du nombre de classe  -
     méthode average" )


  ###### Mélanges gaussiens


library(mclust)


BIC <- mclustBIC(dfU, G = 2:20)
plot(BIC, xlab = "nombre de classe", ylab = "BIC",
     main = "évolution du score BIC selon le nombre de classe par structure
de covariance")
```

```
resMclustVEI <- Mclust(data = dfU, modelNames = "VEI", G = 1:20)
summary(resMclustVEI)
resMclustVEI$classification
resMclustVEI$BIC
resMclustVEI$bic
BICVEI <- resMclustVEI$BIC
plot(BICVEI, xlab = "nombre de classe", ylab = "BIC")
title(main = "évolution du BIC  en fonction du nombre de classe - méthode
VEI ")


mod1 <- Mclust(dfU, modelNames = c("EII", "VII", "EEI", "VEI", "EVI", "VVI"
))
mod1$BIC
summary(mod1)
plot (mod1 , what =  "BIC")


BIC <- mclustBIC(dfU, G = 1:20, modelNames = c("EII", "VII", "EEI", "VEI",
"EVI", "VVI"))
plot(BIC)
print(BIC)

BIC_10 <- mclustBIC(dfU, G = 1:10, modelNames = c("EII", "VII", "EEI",
"VEI", "EVI", "VVI"))
plot(BIC_10)
print(BIC_10)

resMclustVEI <- Mclust(data = dfU, modelNames = "VEI", G = 1:20)
summary(resMclustVEI)
resMclustVEI$classification
resMclustVEI$BIC
resMclustVEI$bic
BICVEI <- resMclustVEI$BIC
plot(BICVEI)



  # Comparaison partitions k = 4 mclust/kmeans

resMclust4 <- Mclust(dfU, modelNames = "VEI", G = 1:10)
resMclust4$BIC
resMclust4$bic
plot(resMclust4$BIC)
resMclust4$classification[1:10]
resKmeans4$cluster[1:10]


resPCAClusterKMEANS <- PCA(cbind.data.frame(ClusterKmeans =
as.factor(resKmeans4$cluster),

                                    dfU),
                     scale.unit = FALSE,
                     graph = FALSE,
                     quali.sup = 1:2)

PKmeans <- plot.PCA(resPCAClusterKMEANS, choix = "ind", label = "none",
axes = c(1, 2), habillage = 1) +
```

```
  ggtitle(paste("Partition des individus si k = 4 (K-means)"))
PKmeans

resPCACluster <- PCA(cbind.data.frame(
                                      ClusterMclust =
as.factor(resMclust4$classification),
                                      dfU),
                     scale.unit = FALSE,
                     graph = FALSE,
                     quali.sup = 1:2)

PMclustMIXT <- plot.PCA(resPCACluster, choix = "ind", label = "none", axes
= c(1, 2), habillage = 1) +
  ggtitle(paste("Partition des individus si k = 4 (Mélange gaussien)"))
PMclustMIXT

PKmeans



  # Comparaison partitions k = 2 mclust/kmeans

resMclust2 <- Mclust(dfU, modelNames = "VEI", G = 2)
resMclust2$BIC
resMclust2$bic

kmeans.rep2 <- replicate(20, kmeans(dfU, 2))
kmeans.rep2
best2 <- which.min(unlist(kmeans.rep2['tot.withinss', ]))
best2
resKmeans2 <- kmeans.rep2[, best2]
resKmeans2
resKmeans2$cluster


resPCAClusterKMEANS2 <- PCA(cbind.data.frame(ClusterKmeans =
as.factor(resKmeans2$cluster),

                                             dfU),
                            scale.unit = FALSE,
                            graph = FALSE,
                            quali.sup = 1:2)

PKmeans2 <- plot.PCA(resPCAClusterKMEANS2, choix = "ind", label = "none",
axes = c(1, 2), habillage = 1) +
  ggtitle(paste("Partition des individus si k = 2 (K-means)"))
PKmeans2



resPCACluster2 <- PCA(cbind.data.frame(
  ClusterMclust = as.factor(resMclust2$classification),
  dfU),
  scale.unit = FALSE,
  graph = FALSE,
  quali.sup = 1:2)
```

```
PMclustMIXT2 <- plot.PCA(resPCACluster2, choix = "ind", label = "none",
axes = c(1, 2), habillage = 1) +
  ggtitle(paste("Partition des individus si k = 2 (Mélange gaussien)"))
PMclustMIXT2

PKmeans2
```

```
### CLASSIFICATION ET REGRESSION SUPERVISEES


set.seed(8)
library(tidyverse)
library(data.table)
ElecGrid <- fread("Data_for_UCI_named.csv", data.table=F)
ElecGrid <- ElecGrid[, -grep("^p1$|^stab$", colnames(ElecGrid))]
any(is.na(ElecGrid))
dim(ElecGrid)
head(ElecGrid, n = 4)
table(ElecGrid$stabf)
seed <- 8
set.seed(seed)
myElecGrid <- ElecGrid[sample(1:nrow(ElecGrid), size = 1000), ]
dim(myElecGrid)
dfS <- myElecGrid

library(FactoMineR)
resPCA <- PCA(myElecGrid, scale.unit = TRUE, graph = FALSE, quali.sup =
ncol(myElecGrid))
plot.PCA(resPCA, choix = "ind", label = "none", axes = c(1, 2), habillage =
ncol(myElecGrid))

  # data management


train <- sample(1:1000, size = 700, replace = F)

X <- dfS[, 1:11]
Y <- dfS[, 12]

X_train <- dfS[train, 1:11]
Y_train <- dfS[train, 12]

X_test <- dfS[-train, 1:11]
Y_test <- dfS[-train, 12]


    # standardisation

dfU_scaled <- dfU
for (i in 1:10){
  dfU_scaled[,i] <- (dfU_scaled[,i] - mean(dfU_scaled[,i])) /
sd(dfU_scaled[,i])
}
```

```
(dfU[,1] - mean(dfU[,1])) / sd(dfU[,1])



  ##### AFD : analyse factorielle discriminante

library(mclust)

fit_DA_EII <- MclustDA(data = X_train,
                 class = Y_train,
                 modelType = "EDDA",
                 modelNames = "EII")
summary(fit_DA_EII)
predEII <- predict(fit_DA_EII, X_test)
predEII
table(predEII$classification, Y_test)
1-mean(predEII$classification == Y_test)



fit_DA_EEE <- MclustDA(data = X_train,
                   class = Y_train,
                   modelType = "EDDA",
                   modelNames = "EEE")
summary(fit_DA_EEE)
predEEE <- predict(fit_DA_EEE, X_test)
predEEE
table(predEEE$classification, Y_test)
1-mean(predEEE$classification == Y_test)


cvMclustDA()
fit_tot <- MclustDA(X, Y, modelType = "EDDA", modelNames = "EEE")
cv <- cvMclustDA(fit_tot, nfold = 5)
cv[c("error", "se")]
cv["error"]


mod <- c("EII", "EEI", "EEE")
v <- rep(NA, 3)
for (i in mod) {
  fit_tot <- MclustDA(X, Y, modelType = "EDDA", modelNames = i)
  cv <- cvMclustDA(fit_tot, nfold = 5)
  v[i] <- (cv["error"])
}
err <- as.matrix(v[4:6])
as.vector(err)

plot(1:3, err, type = "l", pch = 3, col = 1, xlab = "Model type (1 : EII /
2 :  EEI / 3 : EEE)", ylab = "Cross validation error", main = "Erreurs pour
modèles d'analyse factorielle discriminante") # modele EEE ou EEI
sélectionné



mod2 <- c("EII", "VII", "EEI", "VEI", "EVI", "VVI", "EEE", "EVE", "VEE",
"EEV", "VEV", "EVV", "VVE", "VVV")
```

```
v <- rep(NA, 14)
for (i in mod2) {
  fit_tot <- MclustDA(X, Y, modelNames = i)
  cv <- cvMclustDA(fit_tot, nfold = 5)
  v[i] <- (cv["error"])
}
v <- v[!is.na(v)]
v
err <- as.matrix(v)
err

plot(err)
```

```
  ##### KNN : k nearest neighbors

library(FNN)

fit_knn <- knn(train = X_train,
               test = X_test,
               cl = Y_train,
               k = 5)
fit_knn
table(fit_knn, Y_test)
1- mean(fit_knn == Y_test)
```

```
W <- rep (0,40)
for (i in 1:40){
  fit <- knn(train = X_train,
             test = X_test,
             cl = Y_train,
             k = i)
  W[i] <- 1-mean(fit == Y_test)
}

w
plot(1:40, W, pch = 3, xlab = "k", ylab = "Validation set error", main =
"Erreurs pour modèles de k nearest neighbors")
min(W)
which.min(W)
```

```
library(caret)
trControl <- trainControl(method  = "cv",
                          number  = 5)

fit <- train(x = X,
             y = Y,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:40),
             trControl  = trControl,
             metric     = "Accuracy")
```

```
fit
1-fit$results[,2]
plot(1:40, 1-fit$results[,2], xlab = "k", ylab = "Cross validation error",
main = "Erreurs pour modèles de k nearest neighbors", pch =3)



  ##### Arbres (C5.0)

library(C50)

tree <- C5.0(train.X, train.Y)
summary(tree)
predtree <- predict(tree, newdata = test.X)
mean(predtree != test.Y)
classError(predtree, test.Y)

#avec boosting ; trials = 5
treeB5 <- C5.0(train.X, train.Y, trials = 5)
predtreeB5 <- predict(treeB5, newdata = test.X)
CEB5 <- classError(predtreeB5, test.Y)

#avec boosting ; trials : 10
treeB10<- C5.0(train.X, train.Y, trials = 10)
predtreeB10 <- predict(treeB10, newdata = test.X)
CEB10 <- classError(predtreeB10, test.Y)

#avec boosting ; trials : 20
treeB20<- C5.0(train.X, train.Y, trials = 20)
predtreeB20 <- predict(treeB20, newdata = test.X)
CEB20 <- classError(predtreeB20, test.Y)


#avec boosting ; trials : 30
treeB30<- C5.0(train.X, train.Y, trials = 30)
predtreeB30 <- predict(treeB30, newdata = test.X)
CEB30 <- classError(predtreeB30, test.Y)

#représentation graphique erreur classif en fonction boosting
CEB <- c(CEB5$errorRate, CEB10$errorRate, CEB20$errorRate, CEB30$errorRate)
Boost <- c(5, 10, 20,30)
plot(Boost, CEB, xlab = "nombre de boost", ylab = "% erreur de
classification", type = "l",
     main = "erreur de classfication - méthode : arbres avec boosting")



  ##### Arbres avec library(tree)

library(tree)

tree1 <- tree(factor(stabf) ~ ., data = dfS[train,])
summary(tree1)
pred_tree <- predict(tree1, X_test, type = "class")
mean(Y_test != pred_tree)

tree2 <- tree(factor(stabf) ~ ., data = dfS)
summary(tree2)
```

```
treeCV <- cv.tree(tree2,
                  FUN = prune.misclass
                  )

treeCV
plot(treeCV)
elag <- prune.misclass(tree2, best = 18)
summary(elag)
pred_tree_best <- predict(elag, X_test, type = "class")
mean(Y_test != pred_tree_best)




  ##### Random Forests : forêts aléatoires

library(randomForest)

rf1 <- randomForest(X,
                    factor(Y),
                    ntree = 1000)
rf1
importance(rf1)
rf1$err.rate[,1]

(rf1$err.rate[, 1])[1000]


length(seq(100,3000, by=200))
w <- rep(NA,15)
w

for (i in seq(100,3000, by=200)) {
  rf <- randomForest(X, factor(Y), ntree = i)
  w[i] <- rf$err.rate[, 1][i]
}

w <- w[!is.na(w)]
w
plot(1:15, w, xlab = "Numero du modèle", ylab = "Erreur Out Of Bag", main =
"Erreurs des modèles de forêts aléatoires", pch =3)
seq(100,3000, by=200)



  ###### Regression : random forest / regression lineaire


ElecGrid <- fread("Data_for_UCI_named.csv", data.table=F)
df_stab <- ElecGrid %>%
  select(1:4 | 6:13)
any(is.na(df_stab))

hist(df_stab$stab, main = "Histogramme de la variable STAB", xlab = NA)


mydf_stab <- df_stab[sample(1:nrow(df_stab), size = 1000), ]
dim(mydf_stab)
```

```
df <- mydf_stab


train <- sample(1:1000, size = 700, replace = F)

X_stab <- df[, 1:11]
Y_stab <- df[, 12]

X_train_stab <- df[train, 1:11]
Y_train_stab <- df[train, 12]

X_test_stab <- df[-train, 1:11]
Y_test_stab <- df[-train, 12]



length(seq(100,2000, by = 200))
u <- rep(NA, 10)
u
for (i in seq(100, 2000, by = 200)){
  rf_stab <- randomForest(X_stab, Y_stab, ntree = i)
  u[i] <- rf_stab$mse[i]
  }

u <- u[!is.na(u)]
u
plot(1:10, u, xlab = "numero de modèle", pch = 3, ylab = "MSE", main =
"Erreurs estimées des modèles de forêts aléatoires")
which.min(u)
seq(100,2000, by = 200)
u


library(leaps)

fit1 <- regsubsets(stab~., data =df, nvmax = 11)
s <- summary(fit1)
plot(s$cp, xlab = "nombre de prédicteurs", ylab = "Cp de Mallows", main =
"Cp de Mallows en fonction du nombre de variables", pch = 3)

fit <- lm(stab ~. , data = df[train,])
summary(fit)
pred_lm <- predict(fit, X_test_stab)
pred_lm
mean((pred_lm - Y_test_stab)^2)
```