

Module 3 - Réduction de dimension (ACP) et clustering

Dans ce module, nous allons aborder les analyses en composantes principales (ACP) ainsi que deux méthodes de clustering non supervisé : la méthode des k-means et la classification ascendante hiérarchique. Avec ces algorithmes, on cherche à regrouper les individus qui se ressemblent avec un maximum d'homogénéité au sein des classes et un maximum d'hétérogénéité entre les classes.

Vous pouvez créer le fichier R Module3.R.

Nous allons supposer que nos données sont sous forme d'une *matrice X* à n lignes et p colonnes.

La ligne X_i de \mathbf{X} représente l'*individu i*, les entrées de cette ligne sont les mesures des p *variables* sur cet individu. La colonne X^j de \mathbf{X} représente la variable j . Chaque entrée est la valeur de cette variable sur un individu. x_i^j est la donnée de la j -ème variable mesurée sur l'individu i .

$$\mathbf{X} = \begin{bmatrix} X^1 & \dots & X^p \end{bmatrix} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} x_1^1 & \dots & x_1^p \\ \vdots & & \vdots \\ x_n^1 & \dots & x_n^p \end{bmatrix} .$$

1 Analyse en composante principale

L'analyse en composante principale, ou ACP, fait partie de la grande famille des méthodes d'analyse factorielle dont l'objectif est de **synthétiser et visualiser** un tableau de données. Le choix de la méthode dépend du type de données (quantitatives, qualitatives, tableau de contingence...)

L'analyse en composante principale est une méthode d'analyse de données qui consiste à **réduire le nombre de dimensions** (2 ou 3) caractérisant nos individus/observations en transformant les variables **quantitatives** (potentiellement liées) à disposition en nouvelles variables décorrélées (les axes). Cela va nous permettre 1/ d'étudier les ressemblances entre individus du point de vue de l'ensemble des variables et de dégager des *profils* d'individus et 2/ d'illustrer les proximités entre les variables.

Les nouvelles variables correspondent à des "axes". Les premiers axes vont expliquer une grande partie de la variabilité/l'hétérogénéité de nos individus.

Dans la suite nous réalisons une ACP sur des données caractérisant 29 variétés de fromage.

```

rm(list=ls())

fromage <- read.table("data/fromage.csv",
header = TRUE, sep =",", row.names = 1)

head(fromage)
summary(fromage)
str(fromage)

```

Il faut centrer (et en général réduire) les données pour que l'ACP prenne tout son sens. La réduction est indispensable quand les variables ont des unités différentes : cela va permettre d'accorder le même poids à chaque variable.

```
fromage.cr <- scale(fromage, center=T, scale=T)
```

On va utiliser les packages factoextra et FactoMineR pour réaliser l'ACP.

```

library(factoextra)
library(FactoMineR)

res.pca <- PCA(fromage.cr)

print(res.pca) # Affichage des résultats stockés

# Valeurs propres
get_eigenvalue(res.pca) # Afficher les valeurs propres
fviz_eig(res.pca, addlabels = TRUE) # Visualisation graphique

```

Le diagramme représente la variance expliquée par chaque dimension. On voit que la première composante (c'est-à-dire la combinaison linéaire des variables qui synthétise le mieux l'ensemble des variables) explique 56% de la variance et la deuxième 20%. Avec les deux premiers axes, on capte ainsi environ 76% de l'information (de l'inertie totale) du tableau de données. On peut à présent représenter le nuage des individus sur le plan Axe1-Axe2.

```

# Nuage des individus
fviz_pca_ind(res.pca,
              repel = TRUE # éviter le chevauchement des "tags"
              )

# On peut colorer les individus selon plusieurs critères
# (variable qualitative, mesure de qualité...)
fviz_pca_ind(res.pca,
              col.ind = "cos2", # Argument qui indique comment
                                # colorer chaque point.
                                # "cos2" : les couleurs vont représenter
                                # la qualité de representation des
                                # individus sur le plan
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
              repel = TRUE
              )

```

Des "individus" éloignés sont des individus différents. On peut même commencer à envisager des clusters d'individus. Ici on peut voir des groupes de fromages se différencier et d'autres se ressembler. Attention à ne pas sur-interpréter la position des individus mal représentés sur le plan.

On peut également regarder le nuage des variables, aussi appelé, cercle des corrélations. Il permet de voir quelles variables ont contribué aux axes 1 et 2 ainsi que la corrélation entre certaines variables.

```

# Nuage des variables : le cercle des corrélations
fviz_pca_var(res.pca, col.var = "black")

```

Le cercle des corrélations s'interprète de la manière suivante :

- Plus une variable possède une qualité de représentation élevée dans l'ACP, plus sa flèche est longue ;
- Plus deux variables sont corrélées, plus leurs flèches pointent dans la même direction(dans le cercle de corrélation, le coefficient de corrélation est symbolisé par les angles géométriques entre les flèches) ;
- Plus une variable est proche d'un axe principal de l'ACP, plus elle est liée à lui.

On peut aussi visualiser la fonction `corrplot()` du package du même nom pour visualiser les variables les plus contributives à chaque axe.

```
library(corrplot)
var <- get_pca_var(res.pca) # récupère les résultats sur les variables
corrplot(var$contrib, is.corr=FALSE)

# Avec une coloration selon la contribution de la variable aux axes :
fviz_pca_var(res.pca,
              col.var = "contrib", # Argument qui indique comment
                                   # colorer chaque point
              gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
              repel = TRUE        # placement des "tags"
            )
```

On peut également utiliser la fonction `dimdesc(res.pca)` pour décrire automatiquement les axes de l'ACP (utile si on a beaucoup de variables). Cette fonction trie les variables en fonction de leur corrélation avec la composante de l'ACP et garde uniquement les coefficients de corrélation significatifs.

À partir de ces éléments on peut proposer l'interprétation des axes suivante :

Axe 1

- Les variables les mieux représentées, avec les coordonnées (en valeur absolue) les plus grandes) et ayant la plus forte contribution a l'axe 1 sont les variables : calories, lipides, protéines et cholestérol. Ces variables sont fortement corrélées entre elles
- L'axe 1 peut s'interpréter comme l'apport énergétique, nutritionnel global en quelque sorte du fromage : gras, énergie, etc. On peut voir qu'on n'est pas loin d'un "effet taille" avec 7 variables sur 9 corrélées positivement.

Axe 2

- Les variables qui contribuent le plus a l'axe sont : sodium (sel, chlorure de sodium), retinol (vitamine A), folates (vitamine B9), calcium
- Cet axe peut s'interpréter comme l'opposition est les fromages plutôt "salés, secs" vs. fromages plutôt "laitiers"

On peut représenter à la fois les individus et les variables sur le même graphique :

```

fviz_pca_biplot(res.pca, repel = TRUE,
                 col.var = "#2E9FDF", # Variables color
                 col.ind = "#696969" # Individuals color
)

```

À noter que l'ACP est très intéressante dans le cas où on a beaucoup de variables très corrélées entre elles : elle va ainsi permettre d'extraire des composantes principales orthogonales (décorrélatées entre-elles), ce qui peut permettre de les réutiliser dans d'autres analyses (classification, régression...). Le point faible est que les axes ne sont pas toujours facilement interprétables.

2 La méthode des k-means (k-centroïdes)

La méthode des k-means est une méthode de clustering non supervisé. Cela signifie que nous allons partitionner nos données sans qu'elles soient réellement étiquetées (= sans connaissance a priori). L'objectif est de construire K classes d'individus aussi homogènes que possible, autrement dit de minimiser l'inertie intra-classes. En plus du nombre de clusters K qui est fixé par avance, deux autres paramètres sont à choisir : une mesure de distance entre individus (le plus souvent : distance euclidienne) et un représentant de chaque classe (le plus souvent : la moyenne).

L'algorithme est le suivant :

1. *Initialisation* : K individus sont choisis au hasard et considérés comme les centres des K classes (*alternative* : choisir directement des clusters formés aléatoirement plutôt que des centres)
2. Constitution des K classes associées à ces centres : calcul de la distance entre chaque individu et chaque centre de classe → chaque individu est affecté à la classe dont il est le plus proche
3. Le *centroïde* (ou *centre de gravité* ou *moyenne*) de chaque cluster est recalculé sur ces classes ainsi constituées
4. Ce processus est répété jusqu'à convergence :
 - Une fois les classes constituées, on calcule l'inertie (ou variance intra-classe), c'est-à-dire la somme des distances entre chaque individu et son centroïde au carré
 - Si la nouvelle inertie est égale à la précédente (et que ce n'est pas le premier passage dans la boucle), cela signifie qu'on a obtenu la variance intra-classe la plus faible possible → arrêt de l'algorithme

- Sinon on reprend l'algorithme à l'étape 2 : on attribue un centroïde (et donc un nouveau cluster) à chaque individu en calculant le centroïde le plus proche de l'individu en question

Pour comprendre cet algorithme avec des images plutôt qu'avec un texte mathématisé, le lien suivant propose une illustration de ce que réalise l'algorithme des k-means : <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

Dans ce module nous allons simplement appliquer cette méthode sur des données et visualiser le résultat. Les k-means peuvent s'appliquer sur un jeu de données complet ou sur les composantes d'une ACP (intéressant en cas de données volumineuses).

Pour les étudiants les plus avancés, un TP supplémentaire où l'on code l'algorithme des k-means "à la main" est disponible.

La fonction `kmeans()` prend comme argument le jeu de données (en général centré-réduit), le nombre de classes (`centers`) et le nombre d'initialisations aléatoires (`nstart`) : l'algorithme étant sensible aux points initiaux, il est conseillé d'utiliser plusieurs initialisations pour retenir la classification avec la variance intra-classe les plus faible.

```
groupes.kmeans = kmeans(fromage.cr, centers=4, nstart=5)
print(groupes.kmeans)
```

```
# Observation du cluster de chaque fromage
groupes.kmeans$cluster[groupes.kmeans$cluster == 1]
groupes.kmeans$cluster[groupes.kmeans$cluster == 2]
groupes.kmeans$cluster[groupes.kmeans$cluster == 3]
groupes.kmeans$cluster[groupes.kmeans$cluster == 4]
```

On peut essayer de caractériser chaque cluster :

- Cluster 1 : pâtes dures
- Cluster 2 : vaches cru et assimilés
- Cluster 3 : yaourts
- Cluster 4 : plus divers : pâtes molles, chèvre

Nous pouvons visualiser les clusters sur le plan axe principal 1 - axe principal 2 de l'ACP avec la fonction `fviz_cluster` de `factoextra` :

```
fviz_cluster(groupes.kmeans, data = fromage.cr, ellipse.type ="convex",
             xlab = "Axe\u20d71", ylab = "Axe\u20d72")+
theme_minimal()
```

Les commandes ci-dessous permettent de se rendre compte de la décroissance de l'inertie en fonction du nombre de clusters. Cela permet de choisir un nombre de cluster *idéal* en fonction de la cassure observée (critère du coude).

```
inertie_vec<- rep(0,times=10)

for (k in 1:10){
  km <- kmeans(fromage.cr,centers=k,nstart=5)
  inertie_vec[k] <- km$tot.withinss
}

plot(1:10,inertie_vec,xlab="nombre de clusters", type="b")
```

L'algorithme des K-means est plutôt rapide même quand le nombre d'observations est très élevé. En revanche, quand le nombre de variables est élevé, le risque est de subir le *fléau de la dimension*, c'est-à-dire que tous les points soient tellement éloignés les uns des autres qu'on ne peut plus discriminer.

Le défaut de l'algorithme des K-means est qu'il est sensible aux points initiaux. Pour éviter l'influence des outliers, l'algorithme K-means++ modifie l'étape d'initialisation en prenant des centres initiaux les plus éloignés possible. Cet algorithme est implémenté dans la fonction `kmeanspp` du package LICORS.

3 Clustering hiérarchique/Classification ascendante hiérarchique

Nous allons tester une autre méthode de clustering, connue sous le nom de clustering hiérarchique ou répartition hiérarchique. Cette technique est également connue sous le nom de classification *ascendante* hiérarchique (CAH).

La classification ascendante hiérarchique est dite ascendante car elle part d'une situation où tous les individus sont seuls dans une classe, puis sont rassemblés en classes de plus en plus grandes. Les partitions ainsi construites étapes par étapes ne sont jamais remises en cause.

Comme vous pourrez le voir sur le plot que vous générerez ci-dessous, le clustering ascendant hiérarchique procède d'une manière très intuitive :

- Initialement, chaque individu forme une classe, soit n classes. On cherche à réduire le nombre de classes, ceci se fait itérativement.
- À chaque étape, on fusionne deux classes, réduisant ainsi le nombre de classes. Les deux classes choisies pour être fusionnées sont celles qui sont les plus *proches*, en d'autres termes celles dont la dissimilarité entre elles est minimale. Cette valeur de dissimilarité est appelée indice d'agrégation. Comme on rassemble d'abord les individus les plus proches, la première itération a un indice d'agrégation faible, mais celui-ci va croître d'itération en itération.
- On aboutit au final au groupe qui contient toutes les observations initiales.

Il nous faut donc définir deux paramètres :

1. Une mesure de la distance entre deux individus (= indice de **dissimilarité**)
2. Une mesure de la distance entre deux classes (= indice d'**agrégation**)
 - La dissimilarité de deux classes $C_1 = \{x\}$, $C_2 = \{y\}$ contenant chacune un individu se définit simplement par la dissimilarité entre ses individus : $dissim(C_1, C_2) = dissim(x, y)$
 - Lorsque les classes ont plusieurs individus, il existe de multiples critères qui permettent de calculer la dissimilarité. Les plus simples sont les suivants :
 - Stratégie du minimum : retient le minimum des distances entre individus de C_1 et C_2 : $dissim(C_1, C_2) = \min_{x \in C_1, y \in C_2} (dissim(x, y))$ → la distance entre deux classes quelconques C_1 et C_2 est donnée par la plus petite dissimilarité parmi les paires d'éléments x appartenant à C_1 , y appartenant à C_2 ;
 - Stratégie du maximum : retient la dissimilarité entre les individus de C_1 et C_2 les plus éloignés : $dissim(C_1, C_2) = \max_{x \in C_1, y \in C_2} (dissim(x, y))$ → la distance entre deux classes quelconques C_1 et C_2 est donnée par la plus grande dissimilarité parmi les paires d'éléments x appartenant à C_1 , y appartenant à C_2 ;
 - Stratégie de la moyenne : consiste à calculer la moyenne des distances entre les individus de C_1 et C_2 : $dissim(C_1, C_2) = moyenne_{x \in C_1, y \in C_2} (dissim(x, y))$;
 - Distance de Ward : minimise la hausse de l'inertie intra-classe à chaque itération (ou maximise l'inertie inter-classe). L'indice de dissimilarité entre deux classes est alors égal à la perte d'inertie inter-classes résultant de leur regroupement (pour plus d'information sur cette distance se reporter aux pages 11 et 12 de <http://www2.agroparistech.fr/IMG/pdf/ClassificationNonSupervisee-AgroParisTech.pdf>). Concrètement, l'indice de Ward a tendance à regrouper les classes de poids faibles et dont les centres de gravité sont proches.

Pour la mesure de distance entre deux individus, on utilise le plus souvent la distance euclidienne; pour la mesure de distance entre 2 classes, on utilise le plus souvent la distance de Ward.

En R, la première étape est de construire la matrice des distances avec la fonction `dist()` :

```
d.fromage = dist(fromage.cr) # matrice de distances, par défaut  
#c'est la distance euclidienne
```

Ensute on peut lancer le clustering hiérarchique avec la commande `hclust()` puis afficher le dendrogramme en utilisant simplement `plot()`. Une alternative est d'utiliser la fonction `agnes()` du package `cluster`.

```
hc.ward = hclust(dist(fromage.cr), method="ward.D2") #notre clustering  
#hiérarchique avec indice d'agrégation de Ward  
plot(hc.ward) # Dendrogramme
```

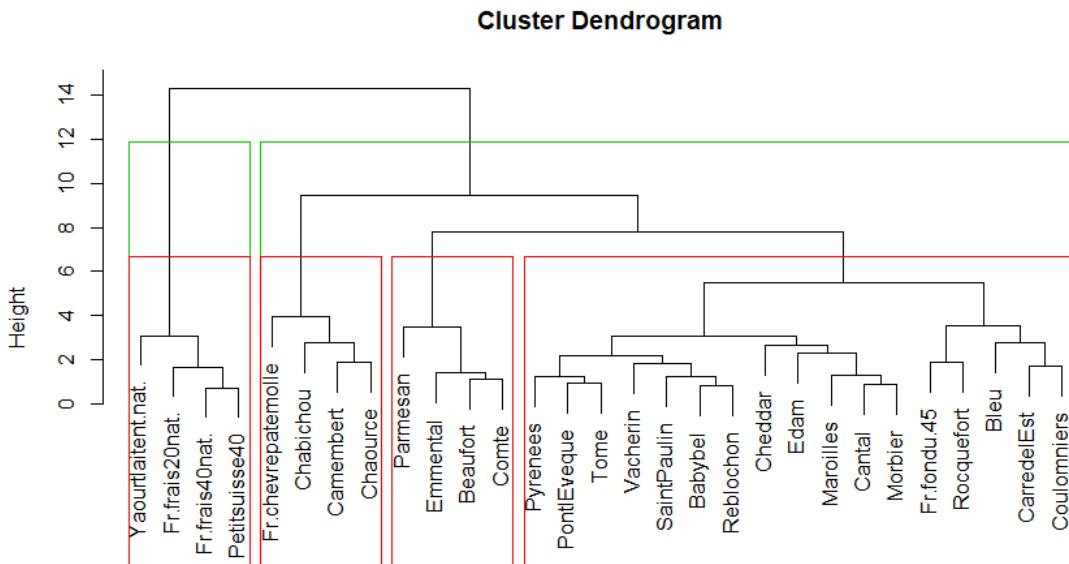
En ordonnée du dendrogramme est affichée la distance correspondant à l'instant où deux groupes (ou observations) fusionnent. Remarquez que, contrairement à k-means, le clustering hiérarchique n'est pas un algorithme d'optimisation, mais plutôt une heuristique ne nécessitant pas d'initialisation. Avec le critère de Ward, l'ordonnée est proportionnelle à l'inertie intra-classe : on agrège deux groupes de manière à provoquer la plus petite augmentation de l'inertie intra-classe. Autrement dit, la hauteur de la coupe donne une idée de la difficulté à agréger deux classes.

À partir du dendrogramme, on peut construire les classes en effectuant une coupe horizontale dans l'arbre et récupérer les classes ainsi définies. La commande `cutree` permet de préciser le nombre de clusters que l'on veut et `rect.hclust()` de les représenter sur le dendrogramme. Au préalable, pour nous aider à choisir le nombre de classes, nous pouvons représenter les sauts d'inertie du dendrogramme selon le nombre de classes retenues.

```
inertie <- sort(hc.ward$height, decreasing = TRUE)  
plot(inertie[1:20], type = "s",  
     xlab = "Nombre de classes",  
     ylab = "Inertie")
```

```
# Des sauts relativement nets apparaissent pour k=2 et k=4

clusterCAH = cutree(hc.ward, 4)
plot(hc.ward)
rect.hclust(hc.ward, 2, border = "green3") # test avec 2 classes
rect.hclust(hc.ward, 4, border = "red")
```



dist(fromage.cr)
hclust (*, "ward.D2")

Comme pour les k-means, on peut afficher les clusters sur le plan Axe1-Axe2 de l'ACP :

```
fviz_cluster(list(cluster = clusterCAH, data = fromage.cr),
            ellipse.type = "convex",
            xlab = "Axe 1", ylab = "Axe 2") +
            theme_minimal()
```

Vous pouvez à présent caractériser les groupes formés selon le nombre de clusters, par exemple en ajoutant une variable `cluster` dans votre data frame. Vous pouvez également comparer, selon les méthodes d'agrégation, les différents clusters obtenus.

Par rapport aux k-means, la CAH a l'avantage de ne pas nécessiter de fixer K a priori et de ne pas dépendre de conditions initiales.

En revanche, le désavantage de la CAH est qu'elle est très coûteuse en temps de calcul. Les résultats sont aussi sensibles à la stratégie d'agrégation choisie. En cas de données volumineuses, une option est de faire un premier k-means avec un grand nombre de classes (par exemple K=1000) puis faire une CAH sur les centres des classes obtenues.

À l'inverse, on peut consolider une CAH par les k-means, c'est à dire faire une CAH pour déterminer le nombre de classes K , puis lancer un k-means à K classes.

Par ailleurs, il peut-être intéressant de réaliser une ACP avant la CAH de façon à supprimer l'information contenue dans les derniers axes, souvent vue comme du bruit (par exemple, on garde les 6 premiers axes de l'ACP, qui représentent 92% de l'inertie totale, sur lesquels on réalise la CAH). La fonction HCPC du package FactoMineR implémente cette démarche.

4 Exercice - Module 3

Rappel : les exercices sont à faire en Quarto (ou Rmarkdown) et doivent être rendu dans le même fichier. Vous devez donc compléter votre fichier `Exercice_R_NOM_PRENOM.qmd`.

Dans cet exercice :

1. Importer les données SAHeart et observer la structure des données.
2. Dans un premier temps nous allons conserver seulement les variables quantitatives. Supprimer deux colonnes du jeu de données en conséquence.
3. Transformer les données afin d'effectuer une ACP. Tracer le nuage des individus et le nuage des variables. Donnez une interprétation aux deux premiers axes. Quel pourcentage de la variance des observations est expliquée par les deux premiers axes ? Commenter le nuage des variables.
4. Dans le nuage des individus, colorer les points en fonction des deux variables non quantitatives que vous avez initialement supprimées. Commenter.
5. Réaliser l'algorithme des k-means avec les variables quantitatives pour différents k . Quel est le paramètre k que vous retenez ? Pourquoi ? Représenter les clusters créés sur le plan Axe1-Axe2 de l'ACP. Commenter les valeurs des variables non quantitatives initialement supprimées par rapport aux différents clusters créés par l'algorithme des k-means.
6. Réaliser l'algorithme du clustering hiérarchique (CAH) avec les variables quantitatives. Couper le dendrogramme pour fournir différents clusters (avec différents nombres de clusters). Quel est le nombre de clusters qui vous paraît le plus adapté ? Pourquoi ? Représenter les clusters créés sur le plan Axe1-Axe2 de l'ACP. Commenter les valeurs des

variables non quantitatives initialement supprimées par rapport aux différents clusters définis par l'algorithme de la CAH.