TP 3 – Clustering

AUTEUR·RICE

B. Charlier, C. Chouquet

1 Exercice. Faut-t-il centrer-réduire les données avant de faire la classification ?

On utilise le jeu de données `fromages.txt` » où 29 fromages sont décrits par 9 variables continues.

```
> X <- read.table("https://bcharlier.github.io/KMAK7AC1/tp/3_Clustering/data/fromage.txt",
      sep="    ", header=TRUE, row.names=1)
```

	calories	sodium	calcium	lipides	retinol	folates	proteines	cholesterol	magnesium
CarredelEst	314	353.5	72.6	26.3	51.6	30.3	21.0	70	20
Babybel	314	238.0	209.8	25.1	63.7	6.4	22.6	70	27
Beaufort	401	112.0	259.4	33.3	54.9	1.2	26.6	120	41
Bleu	342	336.0	211.1	28.9	37.1	27.5	20.2	90	27
Camembert	264	314.0	215.9	19.5	103.0	36.4	23.4	60	20
Cantal	367	256.0	264.0	28.8	48.8	5.7	23.0	90	30
Chabichou	344	192.0	87.2	27.9	90.1	36.3	19.5	80	36

1.1 Sans normalisation

Dans cet partie, il s'agit de faire une classification hiérarchique de Ward avec la distance Euclidienne simple (directement sur X) et d'interpréter cette partition à l'aide d'une ACP non normée (analyse du nuage des individus centrés).

1. Avec la fonction `var()` de R, calculer les variances des 9 variables. Que pouvez-vous en conclure ?
2. Faire la classification hiérarchique de Ward avec la distance Euclidienne simple. Regarder le dendrogramme et la partition en 3 classes.
3. Avec la fonction `PCA` du package `FactoMineR`, faire l'ACP non normée (sur matrice des variances-covariances) de X.
4. Représentez le nuage des 29 fromages dans le premier plan factoriel des individus et le nuage des 9 variables dans le premier plan factoriel des variables. Commentez.
5. Interprétez à l'aide de ces graphiques les classes de la partition en trois classes obtenue avec la méthode de Ward. En déduire pourquoi il peut-être utile ici de centrer-réduire les données avant de faire la classification.

1.2 Avec normalisation

Il s'agit de faire une classification hiérarchique de Ward avec la distance Euclidienne normalisée par l'inverse de la variance (i.e distance Euclidienne standard sur la matrice Z centrée-réduite) et d'interpréter les résultats à l'aide d'une ACP normée (analyse du nuage des individus centrés-réduits).

1. Créer la matrice Z des données centrées-réduites avec la fonction `scale()` (en utilisant la variance empirique « population » i.e. non corrigée).
2. Faire la classification hiérarchique de Ward avec la distance Euclidienne normalisée par l'inverse de la variance et visualisez le dendrogramme.
3. Avec la fonction `PCA` du package `FactoMineR`, faire l'ACP normée (sur matrice des corrélations) de X.
4. Interprétez la partition en 4 classes issues de la classification de Ward.

2 Exercice. Clustering de données qualitatives.

2.1 Problématique

On souhaite analyser des données relatives à une enquête réalisée sur 810 clients d'une banque, contenues dans le fichier `banque.csv`. On s'intéresse tout d'abord aux variables suivantes :

- `crcon` : a des crédits à la consommation ? (`0` : non, `1` : < 3000 €, `3` : > 3000 €)
- `crhab` : a des crédits immobiliers ? (`0` : non, `1` : oui)
- `port` : a un portefeuille de titres boursiers ? (`0` : non, `1` : < 3000, `2` : > 3000 et < 15000, `3` : > 15000)
- `epliv` : possède un livret d'épargne ? (`0` : non, `1` : < 3000, `3` : > 3000)
- `eplog` : possède un plan d'épargne logement ? (`0` : non, `1` : < 3000, `3` : > 3000)
- `prfin` : prélèvements par un organisme financier (`0` : non, `1` : < 150, `2` : > 150)
- `prtre` : prélèvements par le Trésor public (`0` : non, `1` : < 150, `2` : > 150)

On charge les données

```
> banque <- read.csv("https://bcharlier.github.io/KMAK7AC1/tp/3_Clustering/data/banque.csv",
      header = TRUE, row.names = 1)
>
> donnees.actives=banque[,c(10:13,17:18,21)]
> summary(donnees.actives)
```

2.2 Réduction de dimension

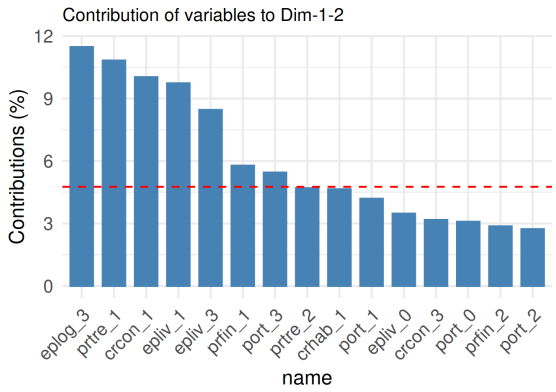
1. Le tableau de Burt $\mathbf{B}$ associé à Q variables qualitatives est une matrice symétrique de taille $\sum_{j=1}^{Q} K_j \times \sum_{j=1}^{Q} K_j$ (où K_j est le nombre de modalités de la variable j).
   - Les **blocs diagonaux** contiennent les tables de contingence de chaque variable avec elle-même (matrices diagonales dont les éléments sont les effectifs marginaux des modalités).
   - Les **blocs extra-diagonaux** contiennent les tables de contingence croisées entre paires de variables qualitatives.

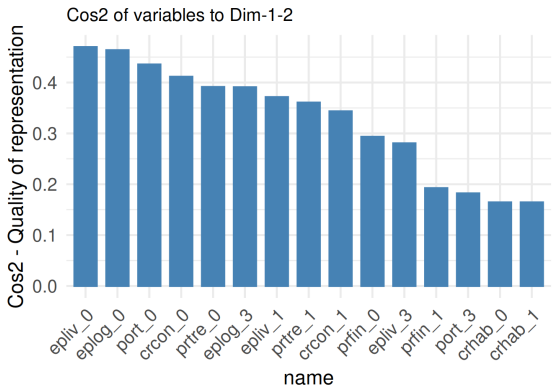
   Calculer le avec la fonction `acm.burt` de la librairie `ade4`, retrouvez sa structure et relevez les 3 catégories à plus fort effectif, et les 3 catégories à plus faible effectif.
2. On procède à une Analyse des Correspondances Multiples (ACM) :
   a. Utilisez la fonction `MCA` de `FactoMineR` sur le tableau `donnees.actives`.

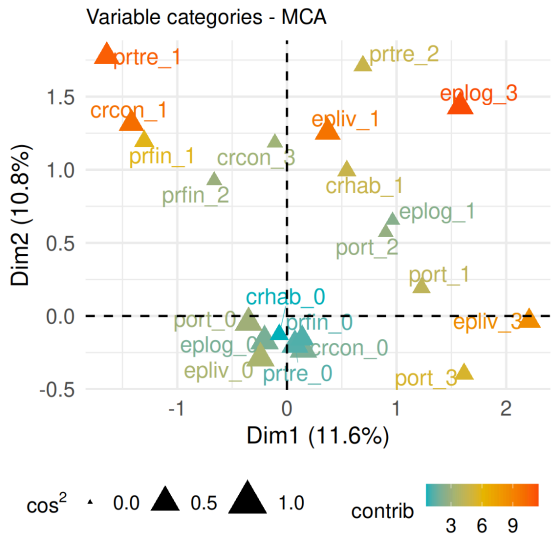
```
> library(FactoMineR)
> acm.b <- MCA(donnees.actives, graph = FALSE, ncp = 10)
```

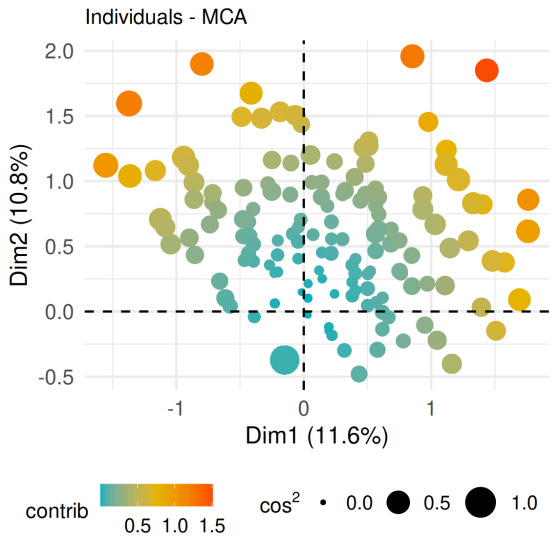
b. Combien d'axes peut-on raisonnablement conserver ?

c. Quelles sont les catégories qui déterminent ces axes factoriels ? Que peut-on dire de la qualité de représentation de ces catégories sur les axes ?







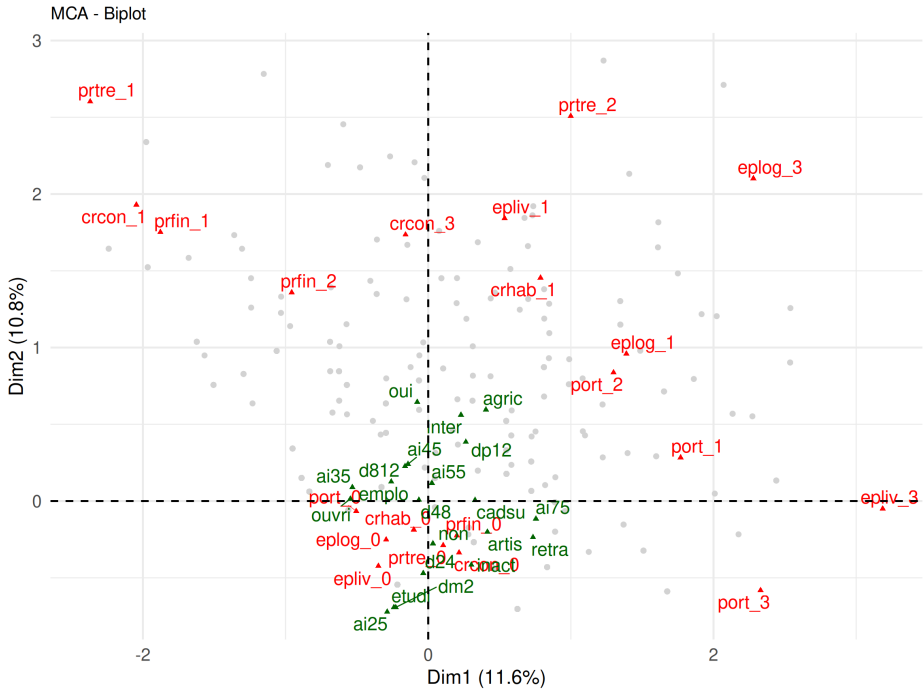


d. Lisez la documentation de `factoextra::fviz_ellipses` et représentez le nuage d'individus en y ajoutant les ellipses `norm` :

3. On cherche à préciser les caractéristiques des axes en termes de type de client. On s'intéresse donc aux variables supplémentaires suivantes :
   - `csp` : `agric` (agriculteur), `artis` (artisan), `cadsu` (cadre supérieur), `emplo` (employé), `etudi` (étudiant), `inact` (inactif), `inter` (profession intermédiaire), `ouvri` (ouvrier), `retra` (retraité) ;
   - `age` (âge du client) : `ai25` ($[18, 25[$), `ai35` ($[25, 35[$), `ai45` ($[35, 45[$), `ai55` ($[45, 55[$), `ai75` ($[55, 75[$) ;
   - `sexe` du client : `hom`, `fem` ;
   - `duree` (durée depuis laquelle la personne est cliente) : `dm2` (moins de 2 ans), `d24` (de 2 à 4 ans), `d48` (de 4 à 8 ans), `d812` (de 8 à 12 ans), `dp12` (plus de 12 ans) ;
   - `oppo` (a déjà fait opposition à un chèque ?) : `oui`, `non` ;
   - `interdit` (a déjà été interdit bancaire ?) : `oui`, `non` ;
   - `cableue` (possède une carte bleue ?) : `oui`, `non`.

a. Quelles sont les catégories supplémentaires les plus pertinentes ? ***Indication: utiliser le coefficient η^2.***

b. À l'aide des graphiques suivants, identifiez les tendances des profils en termes de variables supplémentaires :



4. Synthétisez vos observations et décrivez les groupes possibles de consommateurs de produits bancaires.

2.3 Classification non supervisée sur données qualitatives

Les méthodes de classification non supervisée classiques, telles que la Classification Ascendante Hiérarchique (CAH) utilisant la distance euclidienne ou l'algorithme des k-means, sont spécifiquement conçues pour traiter des variables quantitatives.

Dans le cas présent, le jeu de données étant exclusivement composé de variables qualitatives, il n'est pas possible d'appliquer directement ces algorithmes sur le tableau brut initial.

Une solution éprouvée consiste à réaliser une **Classification sur Composantes Principales (HCPC - *Hierarchical Clustering on Principal Components*)** : on effectue d'abord une Analyse des Correspondances Multiples (ACM), puis on applique la classification non supervisée sur les coordonnées des individus obtenues sur les premiers axes factoriels retenus.

1. Réaliser une CAH sur les 2 premières coordonnées factorielles de l'ACM contenues dans `...$ind$coord`.

   a. Calculer la distance euclidienne sur les coordonnées des axes retenus :

   ```
   > acm.b.clust <- MCA(banque[,c(10:13,17:18,21)], ncp = 2, graph = FALSE)
   ```

   b. Réaliser la classification avec la méthode `ward.D2` et afficher l'arbre (dendrogramme) :

   c. Tracer les sauts d'inertie.

   d. Combien de clusters peut-on choisir ? Representer les groupes sur le premier plan factoriel.

2. Une fois le nombre de clusters choisi, mettre en œuvre la méthode des k-means (toujours à partir des coordonnées factorielles de l'ACM). ***Indication : définir une graine pour le générateur de nombres aléatoires. Utiliser l'option*** `nstart` ***de*** `kmeans`**.**
3. Comparer les deux methodes.
4. Vous pouvez tester la fonction `HCPC` sur les résultats de l'AFCM. Elle a pour avantage de mettre en œuvre une CAH puis des k-means de manière intégrée.

```
> res.hcpc <- HCPC(acm.b.clust, graph = FALSE, nb.clust = 4)
```

La fonction `nb.clust` peut prendre les valeurs :

- `-1` pour que R choisisse automatiquement le nombre de clusters ;
- `0` pour que vous choisissiez le nombre de clusters (en cliquant sur le dendrogramme), en vous basant sur le graphique de perte d'inertie inter-classe ;
- `3` par exemple, pour fixer à 3 le nombre de clusters.

3 Exercice. K-means et quantification des couleurs.

3.1 Introduction et Objectifs

La quantification des couleurs est une technique fondamentale de traitement d'image et de compression. Elle consiste à remplacer un espace colorimétrique étendu (par exemple, $2^{24} \approx 16,7$ millions de couleurs pour une image RGB standard à 8 bits par canal) par un nombre restreint k de couleurs représentatives.

Dans cet exercice, nous allons : 1. Charger une image RGB et la structurer sous forme de nuage de points tridimensionnel (R, G, B). 2. Appliquer l'algorithme des k-means pour réduire la palette à $k = 8$, $k = 16$ et $k = 32$ couleurs. 3. Comparer cette approche non supervisée avec une quantification uniforme par réduction de profondeur binaire. 4. Analyser l'impact visuel et la distorsion colorimétrique.

3.2 Chargement et préparation de l'image

Nous utilisons une image de test pour manipuler les matrices de pixels.



Arachnothera Magna

1. Analyser le code suivant. À quoi sert la dernière ligne ?

```
> # Chargement d'une image exemple
> fname <- file.path(get_script_dir(), "data", "Arachnothera_magna_-_Kaeng_Krachan.jpg")
> img <- readJPEG(fname)
> img <- img[, , 1:3]
```

On pourra afficher des images avec ce code :

```
> plot(c(1, 2), c(1, 2), type = 'n', axes = FALSE, xlab = "", ylab = "")
> rasterImage(img, 1, 1, 2, 2)
```

2. Afficher les niveaux RGB du pixel situé à la ligne 5 et colonne 67.
3. On enregistre maintenant les dimensions de l'image originale :

```
> dims <- dim(img)  # H <- dims[1]; W <- dims[2]; C <- dim[3]
```

Créez ensuite la matrice `pixels` de taille $(H \times W) \times 3$, dont les lignes contiennent les niveaux RGB de tous les pixels.

Vérifier alors que

```
> i <- 5
> j <- 67
> H <- dims[1]
>
> stopifnot(all(img[i, j, 1:3] == pixels[(j - 1) * H + i, 1:3]))
```

3.3 Application de l'algorithme des k-means pour $k \in \{8, 16, 32\}$

L'algorithme des k-means partitionne l'espace colorimétrique RGB en k groupes. Chaque pixel est ensuite remplacé par la couleur du centroïde de son cluster d'appartenance. *Indication: L'utilisation de `nstart = 5` permet de stabiliser l'initialisation des centroïdes et d'éviter les minima locaux de l'inertie intra-classe.*

```
> set.seed(42) # Reproductibilité
>
> # Fonction de quantification par k-means
> quantifier_kmeans <- function(pixels_mat, k, dims_orig) {
+   # Appliquer le kmeans à pixels_mat
+   km <- ...
+   # Remplacement des pixels par leurs centroïdes respectifs
+   pixels_quantifies <- ...
+   # Reconstruction du format image 3D
+   img_quant <- array(pixels_quantifies, dim = dims_orig)
+   return(img_quant)
+ }
>
> # Génération des images quantifiées
> img_k8  <- quantifier_kmeans(pixels, k = 8, dims)
> img_k16 <- quantifier_kmeans(pixels, k = 16, dims)
> img_k32 <- quantifier_kmeans(pixels, k = 32, dims)
```

3.4 Comparaison avec la quantification uniforme par réduction de bits

La quantification uniforme consiste à tronquer uniformément les bits de poids faible de chaque canal indépendamment de la distribution réelle des couleurs.

1. À partir de l'image originale, créer des images avec $k = 2, 3$ bit par canal

```
> # Fonction de quantification uniforme sur b bits par canal
> quantifier_bits <- function(img_mat, bits) {
+   niveaux <- 2^bits
+   img_bits <- round(img_mat * (niveaux - 1)) / (niveaux - 1)
+   return(img_bits)
+ }
>
> # Simulation de réductions de profondeur de bits
> img_bit2 <- quantifier_bits(img, bits = 2) # 2 bit/canal (64 couleurs max au total)
> img_bit3 <- quantifier_bits(img, bits = 3) # 3 bits/canal (512 couleurs max au total)
```

2. Comparer les résultats des deux méthodes.

```
> # 4. Visualisation comparative
> par(mfrow = c(2, 3), mar = c(1, 1, 2, 1))
> plot(c(1, 2), c(1, 2), type = 'n', axes = FALSE, xlab = "", ylab = ""); rasterImage(img, 1, 1, 2, 2); title("1. Originale")
> plot(c(1, 2), c(1, 2), type = 'n', axes = FALSE, xlab = "", ylab = ""); rasterImage(img_k8, 1, 1, 2, 2); title("2. K-means (k = 8)")
> plot(c(1, 2), c(1, 2), type = 'n', axes = FALSE, xlab = "", ylab = ""); rasterImage(img_k16, 1, 1, 2, 2); title("3. K-means (k = 16)")
> plot(c(1, 2), c(1, 2), type = 'n', axes = FALSE, xlab = "", ylab = ""); rasterImage(img_k32, 1, 1, 2, 2); title("4. K-means (k = 32)")
> plot(c(1, 2), c(1, 2), type = 'n', axes = FALSE, xlab = "", ylab = ""); rasterImage(img_bit2, 1, 1, 2, 2); title("5. 64 couleurs")
> plot(c(1, 2), c(1, 2), type = 'n', axes = FALSE, xlab = "", ylab = ""); rasterImage(img_bit3, 1, 1, 2, 2); title("6. 512 couleurs")
```

