Clustering de Variables

Abdourahmane TIMERA, Miléna GORDIEN PIQUET, Marvin CURTY

30 novembre 2025

Table des matières

1 Introduction

Le projet s'inscrit dans le cadre du module Programmation R du Master 2 SISE et consiste à développer un package R complet dédié au clustering de variables, c'est-à-dire des méthodes permettant de regrouper des variables présentant des comportements similaires. Le package a pour vocation d'offrir plusieurs algorithmes complémentaires, adaptés à des variables quantitatives comme qualitatives, et accompagnés d'outils numériques et graphiques facilitant l'interprétation des partitions obtenues. Conformément au cahier des charges du projet, l'objectif est triple :

— Implémenter trois algorithmes distincts, dont au moins un basé sur les techniques de réallocation et un dédié aux variables qualitatives.
— Proposer une architecture logicielle robuste, inspirée des standards modernes (R6, organisation modulaire, cohérence des sorties).
— Fournir un ensemble d'outils d'interprétation permettant d'analyser la nature des clusters, leur cohésion, leur séparation, ainsi que les composantes latentes associées.

Le package est complété par une application Shiny interactive offrant une démonstration des fonctionnalités, l'importation de jeux de données, la sélection des variables et la visualisation des résultats.

2 Programmation orientée objet avec R6

Afin d'assurer une structure claire, réutilisable et conforme aux bonnes pratiques, le développement repose sur le système orienté objet `R6`.

L'usage de `R6` permet :

— de créer des objets persistants contenant données, paramètres et résultats ;
— d'exposer des méthodes standardisées telles que $fit()$,predict(), $summary()$,print() ;
— de se rapprocher des standards scikit-learn, comme demandé dans le cahier des charges.

Chaque algorithme est encapsulé dans une classe dédiée, avec :
— un constructeur définissant les paramètres ;
— une méthode `$fit(X)` effectuant l'apprentissage sur les variables actives ;
— une méthode `$predict(X)` permettant d'affecter de nouvelles variables illustratives ;
— des méthodes `$summary()` et `$print()` pour documenter et inspecter le modèle ;
— des attributs exposant les résultats (clusters, distance, qualité interne, etc.).

3 Architecture interne du package

L'architecture interne du package a été conçue pour être modulaire, lisible et stable. Elle repose sur une organisation en plusieurs dossiers spécialisés, facilitant à la fois l'implémentation, la maintenance et l'extension future du code.

La structure générale du package est la suivante :

```
ClusterVariable/
|-- R/
|   |-- CAH.R                # Classe R6 pour la CAH
|   |-- clusterVariable.R    # Classe R6 pour le K-means variables
|   |-- mon_kmeans.R         # Impl mentation interne du K-means
|   |-- FAMD.R               # Wrapper R6 pour l'AFDM
|   `-- FAMD_finale.R        # Classe pour donn es mixtes
|
|-- man/                     # Documentation (.Rd)
|   |-- CAH.Rd
|   `-- clusterVariable.Rd
|
|-- app/                    # Application Shiny
|   |-- ui.R
|   `-- server.R
|
|-- tests/
|   `-- testthat/
|
|-- DESCRIPTION             # M tadonn es du package
|-- NAMESPACE               # Fonctions export es
`-- README.md             # Documentation principale
```

— **Séparation des algorithmes :** Chaque méthode de clustering est implémentée dans un fichier distinct. Chacun des membre du groupe avait pour but d'implémenter un algorithme.

— **Intégration de l'application Shiny :** L'application interactive est placée dans le répertoire `inst/`, conformément à la structure standard des packages R.

— **Documentation automatique :** Les fichiers d'aide au format `.Rd` sont générés à l'aide de `roxygen2`, garantissant une documentation cohérente, versionnée et intégrée au processus de développement.

3.1 Méthodes standardisées

Toutes les classes suivent une interface commune inspirée des conventions *scikit-learn*. Elles implémentent systématiquement :

— `$new(method, param)` : Initialisation des méthodes et paramètres.
— `$fit(X)` : apprentissage du modèle sur les variables actives ;
— `$predict(X_new)` : attribution des variables illustratives ;
— `$summary()` : résumé détaillé du clustering ;
— `$print()` : aperçu synthétique du modèle ;
— `$plot()` (ou variantes) : visualisations adaptées à l'algorithme.

4 Algorithme K-means pour le clustering de variables

4.1 Principe des Kmeans

Ce K-means est un algorithme d'apprentissage non supervisé utilisé pour le clustering des données. Il regroupe les points de données non étiquetés en groupes ou clusters en fonction de leur similarité. Dans notre contexte, nous appliquons le clustering aux variables plutôt qu'aux individus. Chaque variable devient un point dans un espace à n dimensions (où n est le nombre

d'individus).

On commence par une vérification des données d'entrées.
— Le nombre de clusters souhaités doit être au moins égal à 2.
— Les données doivent être sous forme de dataframe ou de matrice.
Les données sont ensuite transposées de façon à ce que l'algorithme des K-Means s'applique sur les variables et non plus sur les individus.

On s'assure ensuite que le nombre de clusters souhaités est inférieur au nombre de variables du jeu de données avant de sélectionner aléatoirement k variables qui joueront le rôle de centres à l'étape initiale.
Autrement dit, k lignes du jeu de données sont stockées dans k variables distinctes, notées c_i avec i variant de 1 à k.

Commence alors le processus suivant, réitéré jusqu'à convergence.

Calcul des distance de la variable x_j à chaque centre :

$$d^2(\text{variable}_j, \text{centre}_i) = \sum_{l=1}^{n} (x_{j,l} - c_{i,l})^2$$

où :
— $x_{j,obs}$: valeur de la variable j pour l'observation *obs*
— $c_{i,obs}$: valeur du centre i pour l'observation *obs*
— n : nombre total d'observations
Puis, la fonction `which.min` assigne chaque variable au cluster dont le centre est le plus proche.

Mise à jour des centres

$$\mathbf{c}_i(t+1) = \frac{1}{|C_i|} \sum_{j \in C_i} \mathbf{x}_j$$

où :
— C_i : ensemble des variables assignées au cluster i
— $\mathbf{x}_j$: vecteur représentant la variable j
Si un cluster est vide, l'ancien centre est conservé.
L'algorithme s'arrête lorsque la convergence est obtenue, c'est-à-dire lorsque lorsque le nombre de centres reste le même d'une itération à l'autre.

Résultats L'algorithme renvoie les centres des clusters et l'affectation de chaque variable à son cluster.

4.2 Explication de l'algorithme

Algorithme 1 : K-means sur Variables

Entrée : $\mathbf{X}$: matrice $(n \times p)$; k : nombre de clusters ; $T_{\max}$: itérations max
Sortie : Centres $\mathbf{C}$; Assignations $g_1, \ldots, g_p$

1. Validation des entrées
- **si** k *non fourni* **alors**
  - Erreur : "k est obligatoire"
- **fin**
- **si** $\mathbf{X}$ *n'est pas une matrice* **alors**
  - Erreur : "données invalides"
- **fin**
- **si** $k > p$ **alors**
  - Erreur : "k trop grand"
- **fin**

2. Transposition

$\mathbf{X}^T \leftarrow \text{transposer}(\mathbf{X})$; `// Chaque variable devient un point`

3. Initialisation des centres

Sélection aléatoire de k variables : $\mathcal{I} \leftarrow \text{sample}(\{1, \ldots, p\}, k)$

$\mathbf{C} \leftarrow \mathbf{X}^T[\mathcal{I}, :]$

4. Itérations du K-means

pour $t = 1$ **à** $T_{\max}$ **faire**

4.1 Calcul des distances

pour $i = 1$ **à** k **faire**

pour $j = 1$ **à** p **faire**

$D_{ij} \leftarrow \sum_{\ell=1}^{n} (X_{j\ell}^T - C_{i\ell})^2$

fin

fin

4.2 Affectation des variables

pour $j = 1$ **à** p **faire**

$g_j \leftarrow \arg\min_i D_{ij}$

fin

4.3 Mise à jour des centres

$\mathbf{C}^{old} \leftarrow \mathbf{C}$

pour $i = 1$ **à** k **faire**

$\mathcal{V}_i \leftarrow \{j : g_j = i\}$

si $|\mathcal{V}_i| > 0$ **alors**

$\mathbf{C}_i \leftarrow \frac{1}{|\mathcal{V}_i|} \sum_{j \in \mathcal{V}_i} X_j^T$

sinon

$\mathbf{C}_i \leftarrow \mathbf{C}_i^{old}$

fin

fin

4.4 Test de convergence

si $\mathbf{C} = \mathbf{C}^{old}$ **alors**

break

fin

fin

5. Résultats

Retourner les centres $\mathbf{C}$ et le vecteur d'assignation g.

4.3 Description des méthodes

Architecture générale

La classe `clusterVariable` encapsule l'algorithme K-means en suivant la logique des classes R6 et la convention des modèles *scikit-learn* (`fit()`, `predict()`, `summary()`, etc.).

Champs publics

— **k** : nombre de clusters ;
— **data** : données d'entraînement après prétraitement ;
— **cluster_result** : centres et affectations.

Methodes principales

`initialize(k, max_iter, auto_clean)` **Constructeur**
— **k** : nombre de clusters ($k \geq 2$) ;
— **max_iter** : nombre maximal d'itérations (défaut : 100) ;
— **auto_clean** : nettoyage automatique (défaut : TRUE).
Fonctionnalités :
— validation de k ;
— initialisation des paramètres ;
— configuration des options de nettoyage.

`fit(X, check_missing)` **Ajustement du modèle**
Paramètres :
— X : data frame ou matrice ($n \times p$) ;
— **check_missing** : vérification des NA.
Processus :

1. validation des données ;
2. prétraitement si `auto_clean = TRUE` :
   — imputation des valeurs manquantes (médiane) ;
   — détection des valeurs aberrantes (IQR) ;
   — normalisation (centrage-réduction) ;
3. vérification que les variables sont numériques ;
4. exécution de `mon_kmeans()` ;
5. calcul des silhouettes.

Sortie : l'objet `self` (permet le chaînage).

`predict(X_new)` **Affectation de nouvelles variables**
Processus :

1. vérification que le modèle a été ajusté ;
2. validation du nombre d'observations ;
3. transposition de X_{new} ;
4. calcul des distances aux centres ;
5. affectation au cluster le plus proche.

Sortie : data frame avec `Variable` et `Cluster`.

`print()` Affiche :
- statut du modèle ;
- silhouette moyenne par cluster.

`plot_clusters()` **Projection ACP** :
- plan factoriel (ACP) avec `ggplot2` ;
- couleurs par cluster ;
- taille proportionnelle aux silhouettes.

`plot_elbow(k_max)` **Méthode du coude**
- exécute K-means pour $k = 1, \ldots, k_{\max}$;
- calcule l'inertie intra-cluster ;
- trace la courbe inertie vs k.

`plot_heatmap()` **Heatmap de corrélation**
- matrice de corrélation ordonnée par cluster ;
- annotation des clusters.

`cluster_quality_report()` **Rapport de qualité**
- silhouette globale + interprétation ;
- répartition des variables par cluster ;
- silhouettes individuelles ;
- cohésion intra-cluster.

5 Algorithme AFDM

5.1 Principes de l'AFDM et du Clustering

L'AFDM, analyse factorielle de données mixtes est une méthode permettant d'analyser des jeux de données contenant des variables qualitatives et quantitatives.
Plusieurs méthodes sont envisageables pour effectuer une AFDM mais celle préservant le mieux l'information et revenant le plus souvent dans les ouvrages consiste à effectuer un codage disjonctif complet sur les variables qualitatives, comme dans une ACM, analyse de composantes multiples. a cette étape, si on effectue une ACP, analyse de composantes principales, le recodage des variables qualitatives en 0/1 est susceptible d'introduire un biais.
Afin d'éviter ceci, les variables sont pondérées de cette façon pour chaque facteur i :

$$\lambda_i = \sum_j r^2(F_i, X_j) + \sum_j \eta^2(F_i, X_j)$$

où r et η correspondent respectivement au coefficient de corrélation et au rapport de corrélation.

A ce stade, les variables se trouvent projetées dans un espace euclidien et on établit alors la matrice des distances, construite de cette façon :

$$D_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\|_2 = \sqrt{\sum_{\ell=1}^{d} (c_{i\ell} - c_{j\ell})^2}$$

où $\mathbf{c}_i, \mathbf{c}_j \in \mathbb{R}^d$ sont les coordonnées issues de l'AFDM.

On applique alors la méthode de Ward pour effectuer le clustering de variables.
L'inertie totale peut, d'après le théorème de Huyghens, se décomposer ainsi :

$$I_T = I_W + I_B$$

où :

I_W est l'inertie intra-classe :

$$I_W = \sum_{i=1}^{k} \sum_{j \in C_i} \|\mathbf{c}_j - \boldsymbol{\mu}_i\|^2$$

et

I_B est l'inertie inter-classe :

$$I_B = \sum_{i=1}^{k} |C_i| \cdot \|\boldsymbol{\mu}_i - \mathbf{g}\|^2$$

avec :
— $\{C_1, C_2, \ldots, C_k\}$: partition en k clusters
— $\boldsymbol{\mu}_i$: centre du cluster C_i
— $\mathbf{g}$: centre de gravité global
Une bonne partition consiste à chaque regroupement de variables à minimiser l'augmentation de l'inertie intra-classe et à maximiser l'inertie inter-classe.

La qualité du clustering est ensuite évaluée par le calcul de la silhouette pour chaque variable j :

$$s(j) = \frac{b(j) - a(j)}{\max(a(j), b(j))}$$

où

$a(j)$ est la cohésion intra-cluster :

$$a(j) = \frac{1}{|C_i| - 1} \sum_{\substack{j' \in C_i \\ j' \neq j}} D_{jj'}$$

et

$b(j)$ est la séparation inter-cluster :

$$b(j) = \min_{C_\ell \neq C_i} \left(\frac{1}{|C_\ell|} \sum_{j' \in C_\ell} D_{jj'} \right)$$

Enfin, la projection des variables dans le plan factoriel permet d'identifier les regroupements qui sont ensuite présentés dans le dendrogramme.

5.2 Explication de l'algorithme

Algorithme 2 : AFDM et CAH

Entrée :

- $\mathbf{X}$: Tableau de données ($n \times p$) avec variables quantitatives et/ou qualitatives
- d : Nombre de composantes à retenir
- k : Nombre de clusters souhaités

Sortie :

- $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$: Partition des variables en k clusters
- Silhouette moyenne $\bar{s}$ caractérisant la qualité de la partition
- Visualisations (dendrogramme, projection factorielle)

Étapes :

1. Détecter automatiquement le type de chaque variable j
$\mathcal{Q} \leftarrow \{j : X_j \text{ est quantitative}\}$;
$\mathcal{K} \leftarrow \{j : X_j \text{ est qualitative}\}$;

2. Effectuer une AFDM sur $\mathbf{X}$ pour projeter toutes les variables dans un espace factoriel commun $\mathbb{R}^d$

3. Calculer les coordonnées $\mathbf{c}_j \in \mathbb{R}^d$ de chaque variable j **pour** $j \in \mathcal{Q}$ *(quantitatives)* **faire**
  $\mathbf{c}_j \leftarrow$ coordonnées directes de l'AFDM
fin
pour $j \in \mathcal{K}$ *(qualitatives)* **faire**
  $\mathbf{c}_j \leftarrow \frac{1}{|M_j|} \sum_{m \in M_j} \mathbf{c}_m$
fin

4. Calculer la matrice de distances euclidiennes $\mathbf{D}$
$D_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\|_2 = \sqrt{\sum_{\ell=1}^{d} (c_{i\ell} - c_{j\ell})^2}$;

5. Appliquer la CAH (méthode de Ward) sur $\mathbf{D}$ pour construire l'arbre hiérarchique $\mathcal{T}$

6. Découper l'arbre $\mathcal{T}$ à la hauteur appropriée pour obtenir k clusters
$\{C_1, C_2, \ldots, C_k\} \leftarrow \texttt{couper}(\mathcal{T}, k)$;

7. Évaluer la qualité du clustering avec le coefficient de silhouette s, dépendant de la cohésion interne d'un cluster a et de la distance inter-cluster b
pour *chaque variable* $j \in C_i$ **faire**
  $a(j) \leftarrow \frac{1}{|C_i|-1} \sum_{\substack{j' \in C_i \\ j' \neq j}} D_{jj'}$
  $b(j) \leftarrow \min_{C_\ell \neq C_i} \left(\frac{1}{|C_\ell|} \sum_{j' \in C_\ell} D_{jj'} \right)$
  $s(j) \leftarrow \frac{b(j) - a(j)}{\max(a(j), b(j))}$;
fin
$\bar{s} \leftarrow \frac{1}{p} \sum_{j=1}^{p} s(j)$

8. Visualiser les résultats (dendrogramme, projection dans le plan factoriel)

fin

return $\mathcal{C}, \bar{s}$

5.3 Explications complémentaires

Dans cette partie, les différentes étapes ainsi que leur principe sont expliqués.

1. (a) Dans la première partie, l'algorithme détecte le type de variables présentes dans le tableau d'entrée, qualitatives ou quantitatives, et stocke l'information dans deux booléens distincts.
   (b) Les variables qualitatives, si elles existent, sont typées en facteurs.
2. (a) L'AFDM est effectuée et les variables des deux types sont projetées dans le même espace facotriel : chaque variable a des coordonnées sur les axes principaux.
   (b) Les coordonnées sont extraites puis recombinées.
3. Le clustering hiérarchique se déroule comme suit :
   (a) On effectue le calcul de la distance euclidienne entre les coordonnées factorielles des variables i et j :
   (b) La CAH est effectuée par la méthode de Ward, c'est-à-dire que chaque intégration d'une variable à un cluster se fait en minimisant l'écart intra-cluster généré et en maximisant l'écart inter-cluster.
   (c) L'arbre est coupé à la hauteur d'agrégation correspondant au nombre de clusters souhaités.
   (d) Les centres des clusters sont calculés.
4. Cette partie consiste à prédire le cluster auquel les nouvelles variables d'un ensemble appartiennent comme suit :
   (a) Une nouvelle AFDM est effectuée sur l'ensemble incluant les nouvelles variables.
   (b) Chaque nouvelle variable est intégrée au cluster dont le centre est le plus proche.
5. Evaluations et visualisations
   (a) Le graphique présentant le coude trace l'inertie intra-cluster en fonction du nombre de clusters k.
   (b) Le dendogramme présente une vision du clustering sous forme d'arbre.
   (c) La silhouette permet de caractériser la qualité de la classification effectuée :
      — $s(j) \approx 1$ Variable bien assignée (proche de son cluster, loin des autres)
      — $s(j) \approx 0$ Variable à la frontière entre deux clusters
      — $s(j) < 0$ Variable probablement mal assignée

6 Classification Ascendante Hiérarchique (CAH) appliquée aux variables

6.1 Principes de la CAH

Dans cette partie, nous présentons l'algorithme de Classification Ascendante Hiérarchique (CAH). L'objectif est de regrouper les variables « qui portent les mêmes informations » en clusters. afin :

— de révéler la structure interne du jeu de données ;
— de réduire la redondance informationnelle ;
— de synthétiser chaque groupe par une **composante latente**.

6.2 Explication de l'algorithme

Algorithme 3 : CAH sur Variables (version simplifiée)

Entrée : $\mathbf{X}$: matrice $(n \times p)$;
Méthode d'agrégation : Ward.D2

Sortie : Clusters ; Composantes latentes Z_k ; k_{optimal}

1. Préparation des données

- Retirer les variables non numériques.
- Retirer les variables constantes.
- Suppression des lignes contenant des NA.
- Vérifier $n \geq 3$ et $p \geq 3$.

2. Construction de la distance

Calculer la matrice de corrélation : $\mathbf{R} = \text{corr}(\mathbf{X})$.
Construire la dissimilarité :

$$d_{ij} = 1 - |\rho_{ij}|.$$

3. CAH avec Ward.D2

Appliquer `hclust` sur la matrice des distances. Obtenir les hauteurs de fusions $h_1, \ldots, h_{p-1}$.

4. Détection de k optimal

Calculer les ruptures :

$$\Delta h_i = h_{i+1} - h_i.$$

Identifier le plus grand saut $\max \Delta h_i$. Déduire $k_{\text{optimal}} = p - i^\star$ avec $k \in [2, p-1]$.

5. Découpage

Découper en k classes :

$$\text{cluster}(j) = \texttt{cutree}(k).$$

6. Construction des composantes latentes

pour *chaque cluster* C_k **faire**
  si $|C_k| > 1$ **alors**
    Centrer-réduire les variables du groupe.
    Calculer une ACP locale.
    Retenir la première composante latente Z_k.
  fin
  sinon
    $Z_k \leftarrow$ variable unique normalisée.
  fin
fin

7. Calcul des métriques de qualité

- η^2 : corrélation au carré avec Z_k
- R^2 global : moyenne des η^2
- Silhouette : cohésion/séparation des clusters

8. Affectation de nouvelles variables

pour *chaque nouvelle variable* x **faire**
  Standardiser x.
  Calculer :

$$c_k = |\text{corr}(x, Z_k)|.$$

  Assigner x au cluster $\arg\max_k c_k$.
fin

Fin : retourner les clusters, Z_k, k_{optimal}.

6.3 Description des méthodes

Méthode `fit()`

1. Préparation des données

Nous commençons par la préparation d'un jeu de données homogène et pertinent et garantir la cohérence de la distance corrélative utilisée par la CAH. Notre implémentation applique

automatiquement :

- la sélection des variables quantitatives ;
- la suppression des variables constantes ; Une variable de variance nulle ne contribue pas à la structure corrélative et est retirée.
- le retrait des lignes contenant des valeurs manquantes ;
- la vérification d' au moins 3 variables et 3 observations car elles sont nécessaires pour une CAH valide.

2. Mesure des proximités : distance corrélative

Nous utilisons une mesure dérivée de la corrélation, car deux variables fortement associées doivent être considérées proches, qu'elles soient corrélées positivement ou négativement. « On se concentre sur l'intensité de la relation ». On peut comparer des variables comme le prix en € et la surface en m^2 car il s'agit d'une mesure invariante.

La distance retenue est :

$$d(X, Y) = \sqrt{1 - |\text{corr}(X, Y)|}.$$

Il s'agit de la corrélation de Pearson absolue.

Cette matrice de distances est ensuite utilisée comme entrée de la CAH. On utilise la fonction `hclust()`.

La CAH est ensuite réalisée avec la méthode **Ward.D2**, qui minimise à chaque fusion la perte d'inertie intra-classe et renforce la logique de classification autour des variables latentes.

3. Détection de k optimal

Nous avons pris le choix dans ce cluster de nous intéresser à la détection du nombre de clusters optimal. Il est possible, pour l'utilisateur d'entrer ou non le nombre de clusters souhaité. Si k reste vide, alors on utilisera le k optimal prédit. Nous utiliserons l'approche qui consiste à analyser les hauteurs du dendrogramme, c'est-à-dire les pertes d'inertie successives lors des fusions.

(Note : Graphique ici — dendrogramme + tableau des fusions)

Dans notre implémentation, si h_i désigne la hauteur de la fusion i, nous calculons les variations successives :

$$\Delta h_i = h_{i+1} - h_i$$

et identifions le plus grand saut.

Une borne inférieure de 2 et une borne supérieure de $p - 1$ sont appliquées afin d'éviter des partitions triviales. (Une partition triviale à un seul groupe n'apporte aucune information)

Méthode `cutree()`

1. Découpage du dendrogramme

Le découpage du dendrogramme permet de transformer la hiérarchie continue en un ensemble de k groupes disjoints. Le nombre de classes doit se situer entre 2 et $p - 1$.

2. Composante latente

L'étape suivante consiste à caractériser chaque cluster de variables et repose sur l'approche « classification autour de composantes latentes ».

L'idée centrale est :

> Chaque groupe de variables est résumé par une dimension synthétique notée Z_k qui représente au mieux l'information commune aux variables du cluster.

La composante latente est définie comme la variable Z_k qui maximise la somme des corrélations au carré entre Z_k et les variables du groupe. Il est montré que « La variable moyenne est définie comme le premier axe factoriel de l'ACP du groupe. » Cette construction permet :

— de résumer un cluster par un score latent unique ;
— d'évaluer la cohésion interne ;
— d'attribuer ensuite de nouvelles variables (predict).

3. Indicateurs de qualité

Nous calculons ensuite plusieurs indicateurs classiques :

— R^2 global (principe de Huygens) ;
— silhouette (qualité de séparation) ;
— η^2 (variance expliquée variable par variable).

Méthode `predict()`

L'objectif de `predict()` est d'attribuer une ou plusieurs nouvelles variables à un cluster existant. La variable nouvelle est standardisée puis comparée aux composantes latentes.

Elle est affectée au groupe pour lequel :

$$|\text{corr}(X_{\text{new}}, Z_k)| \text{ est maximal.}$$

Cette règle garantit :

— la cohérence avec la distance corrélative ;
— la reproductibilité du modèle ;
— une interprétation directe : « la variable partage le plus d'information avec la structure du groupe k ».

Interprétation

Pour interpréter les résultats de la CAH sur les variables, nous avons conçu trois outils complémentaires : `plot()`, `print()` et `summary()`.

1. plot()

La fonction `plot()` fournit plusieurs représentations graphiques (dendrogramme, ACP, MDS, silhouette, elbow). Ces visualisations sont essentielles car :

— Dendrogramme : « Le dendrogramme et les méthodes factorielles permettent de comprendre la structure induite par la classification. » Ricco Rakotomalala. Elles permettent d'observer les fusions de la CAH.
— MDS/ACP : vérifier visuellement la cohérence des groupes.
— Silhouette : évaluer la qualité du partitionnement.
— Elbow : conforter le choix du nombre optimal de clusters.

2. print()

La fonction `print()` fournit une vue d'ensemble : dimensions, méthode, distance, corrélation moyenne, best k, tailles de groupes. « L'interprétation commence toujours par un résumé des groupes et des effectifs. »

Notre `print()` affiche trois éléments clés :

1. **Taille des clusters** : le tableau des effectifs indique combien de variables composent chaque groupe.
2. **Tableau BSS Ratio / GAP** : pour chaque k, le `print()` calcule l'inertie expliquée (BSS ratio) et son évolution (GAP).
3. **Barycentres et affectation des variables** : le `print()` affiche les barycentres des groupes puis la liste des variables affectées à chaque cluster.

3. summary()

Enfin, la fonction `summary()` fournit une analyse détaillée de la CAH, en suivant les recommandations méthodologiques utilisées dans les supports de cours.

Elle présente quatre éléments majeurs :

1. **Qualité globale de la partition** : le `summary()` calcule et affiche :
   — R^2 global,
   — silhouette moyenne,
   — η^2 (variance expliquée variable par variable).
   → Ces indicateurs permettent d'évaluer la stabilité, la séparation, et la pertinence des clusters.
2. **Structure des clusters** : le `summary()` affiche :
   — le nombre de groupes,
   — leur taille,
   — la liste des variables par cluster.
   Ces informations décrivent la structure interne de la partition et facilitent l'interprétation thématique des groupes.
3. **Composantes latentes et parangons** : pour chaque cluster, le `summary()` affiche :
   — les corrélations^2 des variables avec la composante latente Z_k,
   — la variable la plus représentative du groupe (parangon).
   Cela permet d'identifier chaque cluster et d'expliquer pourquoi certaines variables sont regroupées.
4. **Variables supplémentaires (predict)** : Si le modèle a reçu de nouvelles variables, le `summary()` affiche :
   — leur cluster d'affectation,
   — leurs corrélations aux composantes latentes.
   → Cela garantit une interprétation cohérente entre les données actives et les données ajoutées.

7 Tests de performance

Afin d'évaluer la robustesse et la scalabilité de notre implémentation, nous avons mesuré les temps d'exécution de la CAH et du K-means pour des jeux de données contenant respectivement 100, 900, 2000 et 5000 variables (toujours avec 300 individus).

Comme attendu, la CAH appliquée aux variables présente une complexité quadratique $O(p^2)$, principalement liée :

— au calcul de la matrice de corrélation en $p \times p$;
— à l'algorithme hiérarchique, dont les fusions successives dépendent directement du nombre de variables.

Le K-means appliqué aux variables présente une complexité quasi-linéaire en p, soit environ $O(p \times k \times it)$, principalement liée :
— à la répétition des étapes d'assignation-centroides sur l'ensemble des variables ;
— au nombre d'itérations nécessaires pour stabiliser les centres des clusters.

La méthode AFDM appliquée aux données mixtes présente une complexité plus élevée, de l'ordre de $O(p^2)$, en raison :
— du calcul des matrices de dissimilarité spécifiques à chaque type de variable ;
— de la combinaison des dimensions latentes issues de l'optimisation en espace mixte.

Méthode	**100 var**	**900 var**	**2000 var**	**5000 var**
Quanti – CAH	0.87	2.11	3.53	25.50
Quanti – K-means	0.22	4.91	8.57	42.55
Mixtes – AFDM	0.45	3.21	12.87	58.34
Quali – AFDM	0.52	4.15	15.23	72.41

TABLE 1 – Comparaison des temps de calcul (en secondes) selon la méthode et la dimension (avec n variables et 300 individues)

8 Bibliographie

Références

[1] Rakotomalala, R. (2020).*Classification ascendante hiérarchique* Université Lumière Lyon 2.

[2] Rakotomalala, R. (2020). *Analyse de données multidimensionnelles.* Cours en ligne, Université Lumière Lyon 2.

[3] Rakotomalala, R. (2020). *Les classes R6 sous R (Programmation orientée objet sous R).* Cours en ligne, Université Lumière Lyon 2.

[4] Rakotomalala, R. (2019). *Tanagra - Hierarchical Agglomerative Clustering with PCA.* Cours en ligne, Université Lumière Lyon 2.

[5] Rakotomalala, R. (2020). *CAH et K-Means sous Python.* Cours en ligne, Université Lumière Lyon 2.

[6] Rakotomalala, R. (2025). *Orange Data Mining – Clustering – CAH des variables.* Vidéo sur Youtube , Université Lumière Lyon 2.

[7] Rakotomalala, R. (2025). *Orange Data Mining – Analyse en composantes principales (ACP).* Vidéo sur Youtube , Université Lumière Lyon 2.

[8] Rakotomalala, R. (2022). *Tanagra - ACP 6 - Tandem clustering : ACP + CAH* . Vidéo sur Youtube , Université Lumière Lyon 2.

[9] Rakotomalala, R. *Pratique Methodes Factorielles v1.0 - Chaps 5 et 6* . Cours en ligne, Université Lumière Lyon 2.

[10] Rakotomalala, R. *Classification Variables qualitatives* . Cours en ligne, Université Lumière Lyon 2.

[11] Rakotomalala, R. *Analyse factorielle des données mixtes.* Cours en ligne, Université Lumière Lyon 2.

[12] Lebart, L., Morineau, A., Piron, M. *Statistique exploratoire multidimensionnelle : Visualisation et inférences en fouille de données (4e éd.). Dunod*