UNIVERSITÉ LUMIÈRE LYON 2
Master 2 Statistique et Informatique pour la Science des Données (SISE)

Rapport de Projet Programmation R

Clustering de Variables

Création d'un package pour R

Algorithmes et Outils pour l'Interprétation des Résultats

Encadré par : Dr. Prof. Ricco Rakotomala

Réalisé par : Maissa Lajimi
Yassine Cheniour
Lamia Hatem

Lyon, France
Novembre 2025

Table de matières

1 Introduction

1.1 Contexte

Lorsqu'on parle de clustering, on pense généralement au regroupement d'observations. Cependant, on peut aussi regrouper les variables elles-mêmes selon leurs corrélations.
Le clustering de variables aide à comprendre la structure des données en révélant quelles variables se comportent de manière similaire. Cette approche complète la classification des individus et permet de mieux comprendre ce qui différencie les groupes. Elle détecte aussi les redondances entre variables, ce qui est utile pour identifier la multicolinéarité. On peut ainsi créer des variables synthétiques qui résument chaque groupe, ce qui réduit le nombre de variables tout en gardant l'information importante. Enfin, cette technique peut être utilisée avant ou après la sélection de variables dans d'autres méthodes d'apprentissage supervisé [1].

1.2 Objectifs du Projet

Ce projet vise à développer un package R complet dédié au clustering de variables, accompagné d'une application Shiny pour la visualisation interactive des résultats. L'utilisateur peut soit installer directement le package depuis GitHub, soit exploiter l'application Shiny afin d'explorer de manière interactive les différentes méthodes proposées.
Le package `ClusteringVariables` implémente trois algorithmes de clustering de variables, chacun accompagné d'outils d'interprétation permettant d'analyser la structure des partitions et le degré d'appartenance aux clusters. L'implémentation s'appuie sur des classes R6 et une documentation complète selon les standards R.

1.3 Méthodes Implémentées

Le package `ClusteringVariables` propose une approche flexible et complète pour le regroupement de variables, en offrant plusieurs méthodes adaptées à différents types de données et objectifs analytiques. Chaque méthode implémente un principe de clustering particulier, permettant de capturer différentes formes de relations entre variables : corrélations linéaires, similarités multi-catégorielles ou structures hiérarchiques complexes.

Méthode	**Type de variables**	**Description**
K-Means	Quantitatives	Méthode de partitionnement par réallocation itérative, adaptant l'algorithme classique au contexte du clustering de variables
VarClus	Quantitatives	Méthode hiérarchique divisive basée sur l'ACP, créant des clusters quasi-unidimensionnels de variables fortement corrélées
CAH	Qualitatives	Classification Ascendante Hiérarchique permettant de construire progressivement une hiérarchie par agrégation successive

2 Méthodes de Clustering des Variables

2.1 K-Means pour Variables

2.1.1 Présentation

La méthode K-Means est une technique de partitionnement qui regroupe des variables quantitatives en K clusters homogènes basés sur leurs corrélations. L'objectif est de créer des groupes où les variables d'un même cluster sont fortement corrélées entre elles, mais faiblement corrélées avec les variables des autres clusters. Chaque cluster est résumé par une composante latente unique, correspondant à la première composante principale (PC1) obtenue par ACP sur les variables du groupe.
Dans ce projet, nous l'avons implémentée dans R via une classe R6 qui gère l'ensemble du processus de clustering.

2.1.2 Algorithme

L'algorithme du *K-Means pour variables* repose sur un principe de réallocation itérative adapté au partitionnement de variables. Le fonctionnement général est décrit ci-dessous.

1. Initialisation

Les données sont d'abord standardisées (centrage–réduction). Les p variables sont ensuite réparties aléatoirement dans K clusters, en veillant à ce qu'aucun cluster ne soit vide. Cette étape est répétée sur plusieurs initialisations indépendantes (par défaut 10) afin de retenir la solution présentant l'inertie maximale.

2. Calcul des centres de clusters

Pour chaque cluster k, on construit une composante latente définie comme la première composante principale (PC1) des variables qui lui appartiennent. Cette PC1 joue le rôle de "centre" du cluster. Si X_k représente la matrice des variables du cluster k, la PC1 est obtenue par ACP et normalisée (moyenne 0, écart-type 1). Lorsque le cluster ne contient qu'une variable, sa version normalisée est utilisée directement comme composante latente.

3. Mesure d'adéquation variable–cluster

Pour chaque variable X_j et chaque cluster k, on calcule le coefficient de détermination :

$$R^2(X_j, PC1_k) = \text{corr}^2(X_j, PC1_k),$$

qui quantifie la proportion de variance de X_j expliquée par la composante latente du cluster. Ces valeurs forment une matrice $p \times K$ utilisée pour la réaffectation.

4. Réallocation des variables

Chaque variable est réaffectée au cluster pour lequel son R^2 est maximal :

$$\text{cluster}(X_j) = \arg \max_{k \in \{1,\dots,K\}} R^2(X_j, PC1_k)$$

Si un cluster devient vide après réaffectation, l'algorithme réaffecte automatiquement la variable la moins corrélée à son cluster d'origine afin de rétablir une partition valide.

5. Calcul de l'inertie

Pour un cluster k, l'inertie interne est définie par :

$$\lambda_k = \sum_{j \in k} R^2(X_j, PC1_k),$$

et l'inertie totale est la somme des λ_k sur tous les clusters. Cette inertie mesure la qualité de la partition : plus elle est élevée, mieux les variables sont représentées par leurs composantes latentes respectives.

6. Convergence

Les étapes 2 à 5 sont répétées jusqu'à ce que la partition ne change plus ou que la variation d'inertie devienne inférieure au seuil de tolérance fixé (par défaut 10^{-4}). Le nombre maximal d'itérations est également limité (par défaut 100).

7. Sélection de la meilleure solution

Parmi les multiples initialisations réalisées, la partition retenue est celle maximisant l'inertie totale, garantissant des clusters de variables les plus homogènes possible.

8. Détermination du nombre optimal de clusters

Contrairement à certaines méthodes hiérarchiques, le K-Means nécessite de spécifier le nombre de clusters K à l'avance. Si ce paramètre n'est pas connu, on propose la méthode *elbow()* qui utilise la méthode du coude pour déterminer automatiquement le nombre optimal de clusters.

Principe de la méthode du coude
La méthode du coude consiste à calculer l'inertie totale pour différentes valeurs de K et à identifier le point où l'amélioration devient marginale.

Algorithme de elbow() :

1. Pour chaque k dans l'intervalle testé (par défaut $k = 2, \ldots, 10$), on ajuste un modèle K-Means complet avec k clusters.
2. Pour chaque valeur de k, on enregistre l'inertie totale :

$$I_k = \sum_{i=1}^{k} \lambda_i = \sum_{i=1}^{k} \sum_{j \in \text{cluster}_i} R^2(X_j, PC1_i)$$

3. On trace I_k en fonction de k. La méthode identifie le k optimal comme le point où l'augmentation de l'inertie devient marginale, c'est-à-dire où le gain marginal $I_k - I_{k-1}$ commence à décroître significativement.
4. Le nombre optimal de clusters est le k pour lequel le gain marginal est maximal, correspondant visuellement au "coude" de la courbe.

Limitation

Si la courbe I_k augmente régulièrement sans cassure nette, la méthode peut ne pas identifier clairement un k optimal. Il est recommandé de vérifier visuellement le graphique avec `elbow()` et d'ajuster manuellement si nécessaire en tenant compte du contexte métier.

2.1.3 Implémentation

Le modèle peut être appliqué ainsi :

```
# Donnees quantitatives
X_num <- crime[, -1]  # exclure une variable pour predire ou
   illustrative

# Initialisation et ajustement
km <- KMeansVariablesQuant$new(k = 3, n_init = 20, seed = 42)
km$fit(X_num)

# Resultats
km$summary()
```

Principaux paramètres :

Paramètre	Description
`data`	Matrice ou dataset numérique
`method`	"kmeans"
`n_clusters`	Nombre de clusters (NULL pour sélection automatique via elbow)
`max_iter`	Nombre maximal d'itérations par initialisation (défaut: 100)
`n_init`	Nombre d'initialisations aléatoires (défaut: 10)
`tol`	Seuil de convergence sur la variation d'inertie (défaut: 1e-4)
`seed`	Graine pour la reproductibilité (optionnel)

Valeurs retournées par la fonction :

Élément	Description
`clusters`	Vecteur d'affectation indiquant le cluster de chaque variable
`centers`	Matrice des composantes latentes (PC1) de chaque cluster
`r2_matrix`	Matrice des R^2 entre chaque variable et chaque cluster
`inertia_total`	Inertie totale de la partition retenue
`inertia_by_cluster`	Vecteur des inerties λ_k de chaque cluster
`n_iter`	Nombre d'itérations du meilleur run
`print()`	Méthode pour afficher un résumé concis du modèle
`summary()`	Méthode pour obtenir un résumé détaillé incluant qualité globale, résumé par cluster, et corrélations entre composantes latentes
`predict()`	Méthode pour prédire le cluster d'une nouvelle variable
`illustrative()`	Méthode pour analyser des variables illustratives
`elbow()`	Méthode du coude pour déterminer le nombre optimal de clusters
`plot_correlation_circle()`	Cercle des corrélations des variables actives
`plot_biplot()`	Biplot des variables dans l'espace factoriel

Note : Interface ClusterEngine. Paramètres avancés de KMeansVariablesQuant : k (clusters), n_init (initialisations, défaut 10), max_iter (100), seed (reproductibilité). Documentation : ?KMeansVariablesQuant

2.1.4 Outils d'Interprétation

La classe K-Means fournit plusieurs outils pour interpréter les résultats du clustering de variables et visualiser la structure des clusters.

Visualisations graphiques

plot_correlation_circle() Génère un cercle des corrélations représentant les variables actives dans l'espace des composantes latentes.

plot_biplot() Produit un biplot des variables dans l'espace factoriel issu d'une ACP. Permet d'observer la disposition spatiale des variables et la séparation entre les clusters dans un espace réduit.

elbow() Trace la courbe d'inertie totale en fonction du nombre de clusters et identifie automatiquement le point de coude optimal.

Résumés statistiques :

print() Affiche un résumé concis du modèle incluant les paramètres utilisés (nombre de clusters, itérations, nombre d'initialisations), les dimensions des données, l'inertie totale, et la taille de chaque cluster.

summary() Retourne un résumé détaillé sous forme de liste contenant :

- **global_quality** : métriques de qualité globale (inertie totale, R^2 moyen, pourcentage de variables bien assignées)
- **cluster_summary** : pour chaque cluster, le nombre de variables, l'inertie λ_k, la variance expliquée, et la proportion d'inertie
- **cluster_members** : pour chaque variable, son R^2 avec son propre cluster (*R2_own*), avec le cluster voisin (*R2_next*), et le ratio $(1 - R^2_{own})/(1 - R^2_{next})$ mesurant la qualité de l'affectation
- **cor_latent** : matrice de corrélations entre les composantes latentes des clusters
- **r2_matrix** : matrice complète des R^2 entre toutes les variables et tous les clusters

Analyse de variables supplémentaires :

illustrative() Analyse en profondeur des variables illustratives en calculant leurs corrélations avec toutes les composantes latentes. Retourne :

- **table** : pour chaque variable illustrative, le cluster assigné, le R^2 avec ce cluster, le R^2 avec le cluster voisin, et le ratio de qualité
- **r2_all_clusters** : matrice complète des R^2 entre chaque variable illustrative et tous les clusters
- **plot** : graphiques en barres montrant la distribution des R^2 pour chaque variable illustrative

2.2 VarClus (Variable Clustering)

2.2.1 Présenattion

La méthode **VarClus** (Variable Clustering) est une méthode de regroupement de variables fondée sur leurs corrélations. Elle cherche à créer des groupes de variables homogènes, où les variables d'un même groupe sont fortement corrélées entre elles, mais les groupes sont faiblement corrélés entre eux. Chaque cluster est quasiment unidimensionnel, c'est-à-dire bien résumé par une seule composante principale.
La méthode a été initialement développée par **SAS/STAT** [2]. Dans ce projet, nous utilisons son implémentation dans R via la fonction `varclus()` du package **Hmisc**, qui reprend les principes de SAS tout en s'adaptant à l'environnement R [3].

2.2.2 Algorithme

L'algorithme de clustering hiérarchique divisive VarClus présenté ci-dessous suit la description fournie dans les cours de [1].

1. Calcul de la matrice de similarité

VarClus commence par mesurer à quel point deux variables sont similaires. Dans Hmisc, on peut choisir la mesure de similarité (par défaut : corrélation de Pearson). La corrélation entre deux variables X et Y est définie par :

$$r_{XY} = \frac{\operatorname{cov}(X, Y)}{\sigma_X \, \sigma_Y}$$

2. Identification du groupe à scinder

VarClus est une méthode de clustering hiérarchique divisive : on commence par un seul cluster contenant toutes les variables, puis on le divise progressivement. Pour décider si un cluster doit être scindé, VarClus effectue une ACP sur les variables du cluster et étudie les deux premières valeurs propres λ_1 et λ_2. Un cluster est divisé si :

$$\lambda_2 > 1$$

Cette condition indique que le cluster n'est pas unidimensionnel : une deuxième dimension porte encore une part importante de variance.

3. Analyse en composantes principales du groupe

Une fois qu'un cluster a été identifié comme multidimensionnel, VarClus utilise l'ACP pour déterminer comment répartir les variables dans les deux sous-groupes issus de la division. L'ACP fournit les deux premières composantes principales : C_1 et C_2.
Chaque variable x_j est ensuite corrélée à chacune de ces composantes. La qualité de l'association entre une variable et une composante est mesurée par le coefficient de détermination :

$$R^2(x_j, C_k) = \left(\operatorname{corr}(x_j, C_k)\right)^2, \qquad k \in \{1, 2\}.$$

Ces valeurs R^2 servent à mesurer la proximité de chaque variable avec les deux nouvelles dimensions issues de l'ACP.

4. Rotation des composantes

Comme les composantes brutes ne séparent pas toujours bien les variables, la méthode Hmisc suit le principe SAS : les deux premières composantes sont pivotées via une rotation *quartimax* afin de mieux aligner les variables sur les axes principaux. [4]
Si X est la matrice des variables et W la matrice de rotation, les composantes pivotées sont :

$$Z = XW$$

Cette opération permet de répartir clairement les variables entre les futures branches du cluster.

5. Attribution des variables aux sous-groupes

Une fois les composantes pivotées obtenues, chaque variable est affectée au cluster dont elle est la plus proche au sens du R^2 :

$$\text{cluster}(X_j) = \begin{cases} G_1 & \text{si } R^2(X_j, C_1) > R^2(X_j, C_2) \\ G_2 & \text{sinon} \end{cases}$$

6. Réaffectation / consolidation

Afin d'améliorer l'unidimensionnalité des clusters, une variable peut être réaffectée au cluster voisin si ce changement augmente la proportion de variance expliquée par la composante principale du cluster.

7. Répétition

Les étapes précédentes (ACP, rotation, attribution et réaffectation) sont répétées jusqu'à ce que tous les clusters soient presque unidimensionnels, c'est-à-dire que la première composante explique la majeure partie de la variance de chaque cluster.

8. Découpage du dendrogramme

Une fois la structure hiérarchique construite, il reste à déterminer combien de clusters finaux conserver. Si le paramètre *n_clusters* est spécifié, le dendrogramme est simplement découpé à la hauteur correspondante. En revanche, si *n_clusters=NULL*, la classe R6 utilise la **méthode du coude** *varclus_elbow()* pour déterminer automatiquement le nombre optimal de clusters.

Principe de la méthode du coude
La méthode du coude consiste à analyser la qualité des clusters pour différents nombres de groupes k et à choisir le point où l'amélioration devient marginale. Ici, la qualité des clusters est mesurée par la *similarité moyenne intra-cluster* (corrélations au carré), et non par une distance comme dans les méthodes classiques. On cherche donc le k qui maximise cette similarité.

Algorithme de varclus_elbow() :

1. Pour chaque $k = 2, \ldots, n-1$ (où n est le nombre de variables), on découpe le dendrogramme en k clusters via `cutree()`.
2. Pour chaque cluster i, on calcule la similarité moyenne au sein du cluster :

$$S_i = \frac{2}{|V_i|(|V_i| - 1)} \sum_{p<q \in V_i} r_{pq}^2$$

où V_i est l'ensemble des variables du cluster i et r_{pq} la corrélation de Pearson entre X_p et X_q.

3. On calcule la moyenne de ces similarités sur tous les clusters pour chaque k :

$$\bar{S}_k = \frac{1}{k} \sum_{i=1}^{k} S_i$$

4. On trace $\bar{S}_k$ en fonction de k. La méthode du coude cherche le k correspondant au point de plus grande distance à la droite joignant les extrémités du graphique :

$$\text{distance}(k) = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2 y_1 - y_2 x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}}$$

5. Le nombre optimal de clusters est le k pour lequel cette distance est maximale.

Limitation

Si la courbe $\bar{S}_k$ augmente progressivement sans cassure nette, la méthode peut suggérer un k élevé. Vérifier visuellement avec `plot_elbow` et ajuster manuellement si nécessaire.

2.2.3 Implémentation

Avant de fitter le modèle, et puisque la fonction s'applique uniquement à un ensemble de variables quantitatives, nous avons ajouté une fonction utilitaire `get_numeric_variables()` afin de sélectionner automatiquement les colonnes numériques et d'éliminer les valeurs manquantes (NA). Le modèle peut être appliqué ainsi :

```
# Selection des variables numeriques
X_num <- uscrime[, sapply(uscrime, is.numeric)]

# Initialisation et ajustement
vc <- VarClus$new(similarity = "pearson", n_clusters = NULL)
vc$fit(X_num)
```

Principaux paramètres :

Paramètre	Description
`X_num`	Matrice ou dataset numérique
`method`	"varclus"
`n_clusters`	Nombre de clusters (déterminé automatiquement si NULL)
`similarity`	Mesure de similarité entre variables (pearson, spearman, hoeffding)

Valeurs retournées par la fonction :

Élément	Description
`model`	L'objet complet, contenant toutes les informations du clustering
`clusters`	Dataframe indiquant l'affectation des variables aux clusters
`model$sim`	Matrice de similarité entre les variables

`get_heatmap()`	Graphique de la matrice de corrélation des variables (heatmap)
`model$hclust`	Objet hclust permettant un découpage personnalisé du dendrogramme
`get_dendrogram()`	Dendrogramme représentant la hiérarchie des clusters
`plot_elbow`	Graphique du coude pour la sélection automatique du nombre optimal de clusters (k)
`predict()`	Méthode pour prédire le cluster d'une nouvelle variable
`print()`	Méthode pour afficher un résumé concis du modèle
`summary()`	Méthode pour obtenir un résumé détaillé du modèle et des clusters
`illustrative()`	Méthode pour projeter des variables illustratives sur les clusters existants

Note : Interface ClusterEngine. Paramètres avancés de VarClus : similarity ("pearson", "spearman", "hoeffding"), n_clusters (NULL = auto). Documentation : ?VarClus

2.2.4 Outils d'Interprétation

La classe VarClus fournit plusieurs outils pour interpréter les résultats du clustering de variables et visualiser la structure des clusters.

Visualisations graphiques :

get_heatmap() Génère une heatmap interactive (via Plotly) de la matrice de similarité entre variables. Cette visualisation permet d'identifier rapidement les groupes de variables fortement corrélées et de repérer les structures de corrélation au sein et entre les clusters.

get_dendrogram() Produit un dendrogramme coloré représentant la hiérarchie des clusters. Chaque branche est colorée selon son cluster d'appartenance (via `dendextend`), facilitant l'identification visuelle des groupes de variables. La hauteur des branches reflète la dissimilarité entre les variables ou groupes de variables.

Résumés statistiques :

print() Affiche un résumé concis du modèle incluant la mesure de similarité utilisée, le nombre de variables, le nombre de clusters, et l'état de la modélisation (fitté ou non).

summary() Retourne un résumé détaillé sous forme de liste contenant :

- **text** : résumé textuel du modèle
- **similarity_matrix** : matrice de similarité complète entre toutes les variables
- **cluster_summary** : pour chaque cluster, le nombre de membres, la valeur propre de la première composante principale, et la proportion de variance expliquée
- **R^2_summary** : pour chaque variable, son R^2 avec son propre cluster (*Own_Cluster*), avec le cluster voisin le plus proche (*Next_Cluster*), et le ratio $1 - R^2$ (*1-R2_Ratio*) mesurant la qualité de l'affectation

Analyse de variables supplémentaires :

illustrative() Projette un ou plusieurs ensembles de variables illustratives sur les clusters existants. Pour chaque variable illustrative, la méthode calcule sa corrélation avec les composantes principales de chaque cluster. Elle retourne :

- **table** : tableau des coefficients de corrélation (R), de détermination (R^2), statistiques t et p-valeurs pour chaque variable illustrative vis-à-vis des composantes de chaque cluster
- **plot** : cercle de corrélation PCA montrant la projection simultanée des variables actives (en bleu) et illustratives (en rouge) sur les deux premières composantes principales

2.3 CAH (Classification Ascendante Hiérarchique)

2.3.1 Présentation

La Classification Ascendante Hiérarchique (CAH) est une méthode de clustering hiérarchique agglomératif qui regroupe progressivement les modalités qualitatives les plus similaires. Contrairement aux méthodes divisives comme VarClus, la CAH commence par considérer chaque modalité comme un cluster isolé, puis fusionne itérativement les paires les plus proches jusqu'à obtenir une hiérarchie complète représentée sous forme de dendrogramme [5].
Dans ce projet, la CAH est implémentée via la classe R6 `ClustModalities` et supporte deux approches distinctes :

- **Méthode Dice** : utilise la distance de Dice[2] calculée sur le tableau disjonctif complet, suivie d'une CAH avec liaison de Ward (ward.D2)
- **Méthode ACM** : projette d'abord les modalités dans un espace factoriel réduit via ACM, puis applique une CAH avec liaison de Ward (ward.D) sur les coordonnées factorielles mises à l'échelle [6].

2.3.2 Algorithme

L'algorithme de la CAH se décompose en plusieurs étapes:

1. Préparation des données

Les données qualitatives sont transformées en tableau disjonctif complet (TDC) où chaque modalité devient une variable binaire. Si X contient p variables qualitatives, le TDC résultant contient Q colonnes correspondant au nombre total de modalités.

2. Calcul de la matrice de distances

Selon la méthode choisie, deux approches sont possibles :

Méthode Dice : La distance entre deux modalités j et j' est calculée directement sur leurs vecteurs indicateurs :

$$d_{\text{Dice}}(j, j') = \sqrt{0.5 \sum_{i=1}^{n} (x_{ij} - x_{ij'})^2}$$

où $x_{ij} \in \{0, 1\}$ indique la présence de la modalité j pour l'individu i.

Méthode ACM : On effectue d'abord une Analyse des Correspondances Multiples (ACM) sur le tableau disjonctif :

1. ACM sur le TDC avec extraction des premières composantes (par défaut 2)
2. Récupération des coordonnées factorielles des modalités
3. Mise à l'échelle des coordonnées par $\sqrt{\lambda_l}$ pour chaque axe l :

$$\tilde{F}_{jl} = F_{jl} \times \sqrt{\lambda_l}$$

4. Calcul de la distance euclidienne entre modalités dans cet espace réduit :

$$d_{\text{ACM}}(j, j') = \sqrt{\sum_{l=1}^{L} (\tilde{F}_{jl} - \tilde{F}_{j'l})^2}$$

3. Initialisation

Chaque modalité constitue initialement un cluster isolé. La matrice de distances forme la base pour les fusions successives.

4. Fusions itératives

À chaque itération, l'algorithme :

1. Identifie la paire de clusters (C_i, C_j) présentant la distance minimale selon le critère de Ward
2. Fusionne ces deux clusters en un nouveau cluster $C_{i \cup j}$
3. Met à jour la matrice de distances en calculant la distance entre le nouveau cluster et tous les clusters restants

Le critère de Ward minimise l'augmentation de l'inertie intra-cluster à chaque fusion. Pour deux clusters C_i et C_j, la distance de Ward est définie par :

$$d_{\text{Ward}}(C_i, C_j) = \sqrt{\frac{|C_i| \cdot |C_j|}{|C_i| + |C_j|}} \cdot d(g_i, g_j)$$

où g_i et g_j sont les centres de gravité des clusters et d est la distance euclidienne.

5. Construction du dendrogramme

Le processus se poursuit jusqu'à ce que toutes les modalités appartiennent à un seul cluster. La séquence complète des fusions est enregistrée et représentée sous forme de dendrogramme, où :

- Les feuilles représentent les modalités individuelles
- La hauteur d'une branche indique la distance à laquelle deux clusters ont été fusionnés
- Les branches sont colorées selon l'appartenance aux clusters finaux

6. Découpage du dendrogramme

Pour obtenir une partition en k clusters, le dendrogramme est coupé à une hauteur appropriée. Si le paramètre `k` n'est pas spécifié, la classe utilise la méthode du coude (`elbow()`) pour déterminer automatiquement le nombre optimal de clusters.

Principe de la méthode du coude
La méthode analyse la hauteur d'agglomération (perte d'inertie) à chaque fusion pour identifier le point où l'amélioration devient marginale. Contrairement aux méthodes précédentes qui maximisent une qualité, ici on cherche le point où la hauteur augmente brusquement.

Algorithme de elbow() :

1. Pour chaque $k = 1, \ldots, k_{\max}$ (par défaut $k_{\max} = 10$), on récupère la hauteur d'agglomération correspondante dans le dendrogramme.
2. Les hauteurs sont extraites de la séquence inverse des fusions stockées dans `hclust$height`.
3. On trace la courbe des hauteurs h_k en fonction de k.

4. Le k optimal correspond au point de coude où l'augmentation des hauteurs devient significative, indiquant qu'au-delà de ce point, les fusions regroupent des clusters trop dissimilaires.

Limitation

Si la courbe des hauteurs augmente progressivement sans rupture nette, la méthode peut ne pas identifier clairement un k optimal. Il est recommandé de vérifier visuellement le dendrogramme avec `plot_dendrogram()` et d'ajuster manuellement si nécessaire en tenant compte du contexte métier.

2.3.3 Implémentation

Le modèle peut être appliqué ainsi :

```
# Selection des variables qualitatives
vote_vars <- vote[, -1]

# Methode DICE
cm_dice <- ClustModalities$new(method = "dice")
cm_dice$fit(vote_vars, k = 3)

# Methode ACM (alternative)
cm_acm <- ClustModalities$new(method = "acm", n_axes = 5)
cm_acm$fit(vote_vars, k = 3)
```

Principaux paramètres :

Paramètre	Description
`method`	"dice" (distance Dice + Ward.D2) ou "acm" (ACM + Ward.D)
`n_axes`	Nombre d'axes ACM à utiliser (uniquement pour method="acm", défaut: 2)
`X_quali`	Data.frame contenant uniquement des variables qualitatives
`k`	Nombre de clusters souhaités

Valeurs retournées par la fonction :

Élément	Description
`hclust`	Objet hclust contenant la hiérarchie complète des fusions
`dist_mat`	Matrice de distances utilisée pour la CAH
`mod_clusters`	Vecteur d'affectation des modalités aux clusters
`disj`	Tableau disjonctif complet des modalités
`acm`	Objet ACM complet (uniquement si method="acm")
`mod_coords`	Coordonnées factorielles des modalités (uniquement si method="acm")
`ind_coords`	Coordonnées factorielles des individus (uniquement si method="acm")

`eig_raw`	Valeurs propres brutes (uniquement si method="acm")
`print()`	Méthode pour afficher un résumé concis du modèle
`summary()`	Méthode pour obtenir un résumé détaillé incluant statistiques globales et composition des clusters
`predict()`	Méthode pour prédire le cluster de nouvelles modalités
`illustrative()`	Méthode pour analyser des variables qualitatives illustratives
`illustrative_numeric()`	Méthode pour projeter des variables quantitatives (cercle des corrélations, uniquement ACM)
`elbow()`	Méthode du coude pour déterminer le nombre optimal de clusters
`plot_elbow()`	Visualisation de la courbe des hauteurs d'agglomération
`plot_dendrogram()`	Dendrogramme coloré par cluster
`plot_factorial_map()`	Carte factorielle des modalités (uniquement si method="acm")
`plot_scree()`	Scree plot de l'inertie expliquée (uniquement si method="acm")
`plot_contrib()`	Contributions des modalités aux axes (uniquement si method="acm")
`get_heatmap()`	Heatmap de la matrice de similarité (non implémenté actuellement)
`cluster_table()`	Table d'affectation modalité-cluster
`get_clusters_table()`	Table ordonnée des clusters avec leurs modalités

Note : Interface ClusterEngine. Paramètres avancés de ClustModalities : method ("dice"/"acm"), n_axes (axes ACM, défaut 2), k (clusters). Documentation : ?ClustModalities

2.3.4 Jeux de Données Fournis

Le package intègre six jeux de données documentés permettant d'illustrer les différentes méthodes de clustering :

Dataset	Dimensions	Type	Description
`crime`	47 × 14	Quantitatif	Statistiques de criminalité par État US
`uscrime`	47 × 16	Quantitatif	Variables socio-économiques et criminalité
`autos`	18 × 9	Mixte	Caractéristiques véhicules (7 num, 2 cat)
`autos2005`	38 × 13	Mixte	Véhicules 2005 (9 num, 4 cat)
`loisirs`	8403 × 23	Qualitatif	Enquête pratiques de loisirs
`vote`	435 × 7	Qualitatif	Votes Congrès US 1984

Table 6: Jeux de données inclus dans le package

Ces datasets sont documentés selon les standards R et accessibles via les commandes :

```
data(crime)       # Charger un dataset
?crime            # Afficher la documentation
data(package = "ClusteringVariables")  # Lister tous les datasets
```

Chaque dataset est accompagné d'une page d'aide détaillée décrivant les variables, les sources et les cas d'usage recommandés.

2.3.5 Outils d'Interprétation

La classe ClustModalities fournit plusieurs outils pour interpréter les résultats du clustering hiérarchique de modalités et visualiser la structure des clusters.

Visualisations graphiques :

plot_dendrogram() Génère un dendrogramme représentant la hiérarchie des fusions successives. Les branches sont colorées selon le cluster d'appartenance des modalités.

plot_elbow() Trace la courbe des hauteurs d'agglomération en fonction du nombre de clusters. Cette visualisation aide à identifier le nombre optimal de clusters en repérant le point où les hauteurs augmentent brusquement.

plot_factorial_map() *(Uniquement pour method="acm")* Produit une carte factorielle des modalités dans l'espace des deux premières composantes principales.

plot_scree() *(Uniquement pour method="acm")* Affiche le scree plot montrant le pourcentage d'inertie expliquée par chaque dimension de l'ACM, permettant d'évaluer la qualité de la réduction dimensionnelle.

plot_contrib() *(Uniquement pour method="acm")* Génère un graphique en barres des contributions (en %) des modalités à une dimension factorielle donnée.

Résumés statistiques :

print() Affiche un résumé concis du modèle incluant la méthode utilisée, le nombre d'individus, de variables qualitatives et de modalités totales, ainsi que la taille de chaque cluster.

summary() Retourne un résumé détaillé sous forme de liste contenant :

- **basic_stats** : métriques de base (nombre d'individus, variables, modalités)
- **acm_stats** : valeurs propres, pourcentage d'inertie expliquée et cumulative (si method="acm")
- **cluster_stats** : pour chaque cluster, le nombre de modalités et le pourcentage du total
- **cluster_composition** : liste détaillée de toutes les modalités par cluster
- **method** : méthode de clustering utilisée

cluster_table() Retourne un data.frame avec l'affectation de chaque modalité à son cluster, trié par numéro de cluster.

get_clusters_table() Équivalent à cluster_table(), retourne la table d'affectation modalité-cluster ordonnée.

Analyse de variables supplémentaires :

illustrative() Analyse des variables qualitatives illustratives en les projetant sur les clusters existants. Pour chaque modalité illustrative :

- **Méthode Dice** : calcule la distance moyenne de Dice avec les membres de chaque cluster
- **Méthode ACM** : projette la modalité dans l'espace factoriel et calcule sa distance aux barycentres des clusters

Retourne :

- **table** : pour chaque modalité illustrative, le cluster assigné et les distances à tous les clusters
- **plot** : visualisation des modalités illustratives sur la carte factorielle (ACM) ou graphiques en barres des distances (Dice)

3 Architecture du Package R

3.1 Structure Générale

```
ClusteringVariables/
|-- R/
|   |-- clusterengine.R     # Classe unifiee
|   |-- kmeans.R            # Implementation du K-Means
|   |-- varclus.R           # Implementation de VarClus
|   |-- acm_cah.R           # Implementation de la CAH
|   |-- n_clusters.R        # Methodes de selection du k
|   |-- utils.R             # Fonctions utilitaires
|-- man/                    # Documentation R (fichiers .Rd)
|-- data/                   # Jeux de donnees d'exemple
|-- tests/
|   |-- testthat/           # Tests unitaires
|-- DESCRIPTION             # Metadonnees du package
|-- NAMESPACE               # Exports et imports du package
|-- README.md
```

3.2 Architecture R6

Le package utilise des classes R6 pour garantir une implémentation orientée objet robuste et maintenable [7]. Trois classes spécialisées implémentent les algorithmes de clustering :

- **KMeansVariablesQuant** : K-Means pour variables quantitatives
- **VarClus** : Clustering hiérarchique de variables quantitatives
- **ClustModalities** : CAH pour variables qualitatives (méthodes ACM et DICE)

Une classe wrapper `ClusterEngine` facilite l'utilisation des trois méthodes via une interface commune. Elle stocke les données, la méthode choisie, et le modèle ajusté, puis délègue les appels (`fit()`, `predict()`, `summary()`) à la classe spécialisée appropriée. Cette architecture simplifie l'intégration dans l'application Shiny où l'utilisateur sélectionne la méthode via l'interface graphique.

Utilisation directe des classes spécialisées Les classes peuvent également être utilisées directement pour un contrôle plus fin des paramètres :

```
# K-Means avec parametres avances
km <- KMeansVariablesQuant$new(
    k = 3,
```

```
    n_init = 20,      # 20 initialisations
    max_iter = 100,   # 100 iterations max
    seed = 42         # reproductibilite
)
km$fit(data_quantitatives)

# VarClus avec similarite Spearman
vc <- VarClus$new(
    similarity = "spearman",
    n_clusters = NULL  # selection automatique
)
vc$fit(data_quantitatives)

# CAH avec methode ACM
cm <- ClustModalities$new(
    method = "acm",
    n_axes = 5
)
cm$fit(data_qualitatives, k = 3)
```

Cette double interface offre à la fois simplicité d'utilisation (via `ClusterEngine`) et flexibilite (via les classes spécialisees).

3.3 Dépendances

Le package s'appuie sur plusieurs packages R existants :[6]

- **R6** : implémentation des classes orientées objet
- **Hmisc** : fonction `varclus()` pour le clustering hiérarchique divisif
- **Plotly** : génération de visualisations interactives
- **dendextend** : visualisation colorée des dendrogrammes pour faciliter l'interprétation des structures hiérarchiques
- **FactoMineR** : Analyse en Composantes Multiples (ACM) pour le traitement des variables qualitatives

3.4 Installation et Utilisation

Le package peut être installé directement depuis GitHub à l'aide du package `remotes` :

```
# Installation de remotes si necessaire
if (!requireNamespace("remotes", quietly = TRUE))
  install.packages("remotes")

# Installation du package depuis GitHub
remotes::install_github("maissaladjimi/SISE_Clustering_Variables_R")
```

Une fois installé, le package peut être chargé :

```
library(ClusteringVariables)
```

4 Application Shiny

4.1 Présentation

L'application Shiny accompagne le package `ClusteringVariables` et permet d'explorer de manière interactive les différentes méthodes de clustering de variables sans nécessiter de programmation. Elle offre une interface graphique intuitive pour charger des données, configurer les analyses et visualiser les résultats.

4.2 Lancement de l'Application

L'application nécessite d'être lancée depuis le répertoire `inst/shiny/` du dépôt cloné. Les packages `bslib`, `shiny` et `shinyjs` seront installés automatiquement si absents.

```
# Se placer dans le repertoire shiny
setwd("chemin/vers/SISE_Clustering_Variables_R/inst/shiny")

# Lancer l'application
shiny::runApp()
```

L'application sera alors accessible dans le navigateur à l'adresse indiquée dans la console (typiquement `http://127.0.0.1:XXXX`).

4.3 Fonctionnalités

L'application Shiny propose les fonctionnalités suivantes :

- **Chargement des données** : import de fichiers aux formats CSV et Excel. L'interface permet de prévisualiser les données importées et de vérifier leur structure avant l'analyse.
- **Sélection des variables** : choix des variables actives qui seront utilisées pour le clustering, ainsi que des variables illustratives permettant d'enrichir l'interprétation des résultats sans influencer la formation des clusters.
- **Choix de la méthode** : sélection parmi les trois méthodes implémentées (K-Means et VarClus pour variables quantitatives, CAH pour variables qualitatives).
- **Configuration des paramètres** : ajustement du nombre de clusters (avec possibilité de sélection automatique) et configuration des options spécifiques à chaque algorithme.
- **Visualisation des résultats** : génération automatique de graphiques interactifs incluant les dendrogrammes, les heatmaps de corrélation, les graphiques du coude pour la sélection du nombre optimal de clusters, et les cercles de corrélation pour l'interprétation des axes principaux.
- **Exploration interactive** : accès aux résumés statistiques détaillés, tableaux d'appartenance des variables aux clusters avec indicateurs de qualité (R^2), et outils d'interprétation facilitant la compréhension de la structure des données et des relations entre variables.

5 Conclusion

Ce projet a permis de concevoir un package R complet dédié au clustering de variables, intégrant trois méthodes complémentaires : *K-Means pour variables*, *VarClus* et *CAH*. Chaque algorithme

dispose d'une implémentation structurée en classes R6, assurant une interface cohérente ainsi qu'une gestion centralisée des données, des paramètres et des sorties. Le package fournit également un ensemble d'outils d'interprétation tels que matrices de similarité, dendrogrammes, cercles de corrélation, biplots, ou encore ratios d'affectation, facilitant l'analyse de la structure des clusters et la qualité des partitions obtenues. Le package est accompagné de six jeux de données documentés offrant des exemples concrets d'utilisation pour chaque méthode, ainsi qu'une documentation complète en anglais suivant les standards R.
L'application Shiny développée en parallèle offre une dimension interactive au projet : elle permet de charger des données, de sélectionner une méthode de clustering, d'exécuter le modèle et d'examiner immédiatement les résultats sous forme graphique ou tabulaire. Cette plateforme rend le package accessible à un public plus large, y compris non spécialiste de R, et constitue un outil pédagogique pour explorer les méthodes de regroupement de variables.

6 Perspectives

Plusieurs pistes d'amélioration peuvent être envisagées pour enrichir le package :

- **Optimisation des performances** : parallélisation de certaines étapes (par exemple le calcul répété des corrélations ou la gestion de multiples initialisations pour le K-means) afin de réduire le temps de traitement sur des jeux de données volumineux.
- **Uniformisation et factorisation du code** : création de sous-classes dédiées aux visualisations, à la gestion des matrices de similarité ou aux calculs statistiques, permettant d'alléger les classes principales et d'améliorer la maintenabilité du package.
- **Extension à d'autres types de variables** : intégration future de méthodes adaptées aux variables mixtes (quantitatives et qualitatives), par exemple via l'ACM ou des distances spécifiques, afin d'élargir les cas d'usage.
- **Enrichissement de l'application Shiny** : ajout d'options de comparaison directe entre méthodes, d'exports automatisés des résultats ou encore d'une aide interactive guidant l'utilisateur dans l'interprétation des clusters.

En somme, ce projet a posé les bases d'un outil robuste et flexible pour l'analyse des variables, offrant à la fois une approche algorithmique rigoureuse et une interface conviviale pour l'exploration des résultats.

References

[1] Ricco Rakotomalala. Classification de variables autour de composantes latentes. *Tutoriel, Université Lyon 2*.

[2] SAS Institute Inc. *The VARCLUS Procedure*. SAS Institute, 2023. Accessed: 1 February 2025.

[3] Frank Harrell. varclus function documentation. https://search.r-project.org/CRAN/refmans/Hmisc/html/varclus.html, 2024. Accessed: 2025-02-01.

[4] Ricco Rakotomalala. Acp + rotation quartimax et clustering de variables (python). *YouTube*, 2023.

[5] Ricco Rakotomalala. Clustering des modalités de variables catégorielles. *Tutoriel, Université Lyon 2*, 2014.

[6] Ricco Rakotomalala. Analyse des correspondances multiples (acm). *Tutoriel, Université Lyon 2*.

[7] Ricco Rakotomalala. Méthode des centres mobiles: Classification par partition. *Tutoriel, Université Lyon 2*.

[8] Ricco Rakotomalala. Classification de variables qualitatives. *Tutoriel, Université Lyon 2*.

[9] Ricco Rakotomalala. Clustering de variables quantitatives (python). *YouTube*, 2023.

[10] Ricco Rakotomalala. Clustering variables qualitatives – classification des modalités (python). *YouTube*, 2023.

[11] Ricco Rakotomalala. Clustering des modalités de variables qualitatives via leur représentation factorielle (python). *YouTube*, 2023.

[12] Ricco Rakotomalala. Tanagra – acp #5 – varclus, clustering de variables. *YouTube*, 2022.

[13] Ricco Rakotomalala. Orange data mining – clustering – cah des variables. *YouTube*, 2025.

[14] Winston Chang. *R6: Encapsulated Classes with Reference Semantics*. CRAN, 2021. R package version 2.5.1.

[15] Ludovic Lebart, Alain Morineau, and Marie Piron. *Statistique exploratoire multidimensionnelle*. Dunod, Paris, 1995.

[16] H. Abdallah and Gilbert Saporta. Classification d'un ensemble de variables qualitatives. *Revue de Statistique Appliquée*, 46(4):5–26, 1998.

[17] M. Qannari, E. Vigneau, and P. Courcoux. Une nouvelle distance entre variables. application en classification. *Revue de Statistique Appliquée*, 46(2):21–32, 1998.