mmrClustVar

Implémentation en R6 du Clustering de Variables

Marin NAGY, Mazilda ZEHRAOUI, Rina RAZAFIMAHEFA

Novembre 2025

Master 2 SISE

(Statistique et Informatique pour la Science des donnéEs)

Supervision : Ricco Rakotomalala

Table des matières

1 Introduction

Ce rapport présente le développement du package **mmrClustVar**, une classe R6 permettant la classification automatique de variables.

1.1 Rappel du cahier des charges

- Création d'un package :
  - proposant plusieurs méthodes de clustering de variables
  - accompagnées d'indicateurs pour l'interprétation des résultats
  - que l'on peut installer directement à partir de GitHub.
  - qui intègre un fichier d'aide en anglais aux normes R, c.-à-d.
    - description des fonctions, de leurs paramètres, des objets fournis en sortie, de la lecture des résultats
    - avec des exemples d'utilisation (voir par ex. `?glm` du package stats)
- Implémentation d'algorithmes de clustering de variables :
  - 3 algorithmes de clustering de variables.
    - au moins un basé sur les techniques réallocation
    - au moins un consacré au traitement des (modalités) des variables qualitatives
  - Mise en place d'une stratégie pour l'identification du nombre de clusters (au moins des approches qui aident à l'identification)
  - Intégrer des outils (tableaux, graphiques) qui permettent d'interpréter les résultats :
    - nature des partitions
    - nature des groupes
    - degré d'appartenance aux clusters
    - implémenter ou utiliser d'autres outils (uniquement packages référencés, ex.`hclust`) → Documenter les choix.
- Classe R6
  - Un constructeur, avec les paramètres éventuels
  - La méthode `$fit(X)` qui lance la modélisation sur les données d'apprentissage. « X » représente un data frame comportant l'ensemble des variables actives.
  - La fonction `$predict(X)` qui prend en entrée un data frame X de variables illustratives compatible avec celui présenté à `fit()`, et qui rattache chaque colonne de X à l'un des clusters (avec éventuellement des indicateurs numériques).
  - La procédure `$print()` qui affiche les informations succinctes sur le modèle
  - La procédure `$summary()` qui affiche les informations détaillées
  - L'objet peut exposer une série de propriétés qui peuvent être exploitées par la suite
  - `initialize`, `fit`, `predict`, `summary`, `plot`
- App Shiny permettant de :
  - Sélectionner un fichier de données (CSV – séparateur tabulation – au moins ; autre éventuellement, ex. « xlsx »...)

    - Choisir les variables explicatives et les variables illustratives
    - Lancer les calculs et présenter les résultats, avec en particulier une mise en valeur des sorties
    - Autres
- Fonctionnalités additionnelles.

1.2 Livrables

- n rapport en français au format PDF de présentation de notre travail :
    - fournir la source .tex
    - doit indiquer les formules, stratégies, algorithmes utilisés pour produire les résultats.
    - douzaine de pages
    - références bibliographiques
- GitHub
    - Le package doit
        - pouvoir être installé directement en ligne à partir de GitHub.
        - comporter les jeux de données exemples utilisés dans le tutoriel.
        - Le code source du package et les documents associés (aide, etc.).
        - Une copie du package au format ZIP (ou tar.gz) directement utilisable sous R.
        - L'application R SHINY avec tous les éléments permettant de le faire fonctionner.
        - Un tutoriel (reproductible) en anglais montrant l'utilisation des fonctionnalités du package.
            - Montrer au moins le fonctionnement de `$fit()` et description des sorties.
            - Montrer le fonctionnement de l'application SHINY.
- Projet complet (rapport, package, source, etc.) sur un drive

2 Structure de la classe R6 `mmrClustVar`

La classe R6 `mmrClustVar` constitue le cœur du package. Elle encapsule :

- le choix de la méthode de clustering de variables (*k-means*, *k-modes*, *k-prototypes*) ;
- les paramètres de modélisation (nombre de groupes, standardisation, pondération des variables qualitatives) ;
- les résultats du modèle (partition, centres/prototypes, inertie) ;
- les fonctionnalités associées ('fit()', 'predict()', 'summary()', graphiques, etc.).

2.0.1 Attributs privés

Les attributs privés stockent l'état interne du modèle et ne sont pas directement modifiables par l'utilisateur. Ils sont néanmoins accessibles de manière contrôlée via les méthodes publiques.

Les principaux attributs sont :

— `FMethod` : chaîne de caractères indiquant la méthode utilisée (*kmeans*, *kmodes*, *kprototypes*, ou *auto* pour la sélection automatique en fonction du type des variables actives).
— `FNbGroupes` : entier strictement positif correspondant au nombre de clusters K.
— `FScale` : booléen indiquant si les variables quantitatives actives doivent être standardisées (centrage-réduction) avant le clustering.
— `FLambda` : paramètre numérique $\lambda > 0$ contrôlant la pondération de la partie catégorielle dans la distance mixte de k-prototypes.
— `FX_active` : data frame des variables actives utilisées pour construire les clusters (colonnes = variables à regrouper).
— `FX_descr` : data frame des variables descriptives éventuellement fournies lors des appels à `predict(X_new)` (colonnes = variables à rattacher aux clusters existants).
— `FClusters` : vecteur de longueur égale au nombre de variables actives, contenant l'étiquette de cluster associée à chaque variable.
— `FCenters` : objet (matrice, liste ou liste de listes) décrivant les centres/modes ou prototypes de chaque cluster, selon la méthode :
  — centres numériques pour k-means ;
  — modes catégoriels pour k-modes ;
  — prototypes mixtes pour k-prototypes.
— `FInertia` : valeur numérique représentant l'inertie intra-cluster.
— `FConvergence` : booléen indiquant si l'algorithme a convergé (aucun changement d'affectation ou nombre maximal d'itérations atteint).
— `FAlgorithme` : chaîne de caractères résumant la variante d'algorithme utilisée.

Des fonctions internes (non exposées) exploitent ces attributs, par exemple pour vérifier la cohérence des données d'entrée, calculer les distances appropriées ou réaliser une étape de l'algorithme (k-means, k-modes, k-prototypes).

2.0.2 Méthodes publiques

Les méthodes publiques constituent l'interface principale pour l'utilisateur du package et pour l'application Shiny.

— `initialize(method, K, scale = TRUE, lambda = 1, ...)` : **Constructeur de la classe**. Il initialise les paramètres de modélisation : choix de la méthode, nombre de groupes K, option de standardisation des variables quantitatives, valeur de λ pour k-prototypes, ainsi que les attributs internes (`FMethod`, `FNbGroupes`, `FScale`, `FLambda`, etc.).
— `fit(X)` : **Méthode d'apprentissage**. L'argument `X` est un data frame dont les colonnes sont les variables actives à regrouper. La méthode :

1. vérifie la cohérence de `X` (types, valeurs manquantes, etc.) ;
2. applique éventuellement la standardisation des variables quantitatives, colonne par colonne ;
3. choisit automatiquement l'algorithme adapté si `FMethod = "auto"` ;

4. lance l'algorithme de clustering correspondant (`kmeans`, `kmodes` ou `kprototypes`) sur les profils de variables ;
5. et à jour `FClusters`, `FCenters`, `FInertia`, `FConvergence` et conserve `FX_active` pour les traitements ultérieurs.

— `predict(X_new)` : **Méthode de rattachement de variables descriptives**. L'argument `X_new` est un data frame de variables supplémentaires, compatible avec `FX_active` (même nombre d'individus et types adaptés). Pour chaque variable de `X_new`, la méthode :

1. applique les mêmes prétraitements que pour les variables actives (par exemple standardisation individuelle pour les quantitatives) ;
2. calcule sa distance au centre/mode/prototype de chacun des K clusters ;
3. l'affecte au cluster le plus proche, en renvoyant un objet (data frame) qui contient
   — le nom de la variable
   — son cluster
   — un ou plusieurs indicateurs d'adhésion (distance, r^2, etc.).

— `print()` : **Affiche** un résumé succinct du modèle :

1. méthode utilisée ;
2. nombre de groupes ;
3. nombre de variables actives ;
4. état de convergence.

— `summary()` : **Résumé détaillé des résultats**.

1. taille de chaque cluster (nombre de variables) ;
2. inertie intra-cluster ;
3. description synthétique des centres/modes/prototypes ;
4. indicateurs de qualité pour chaque variable (par exemple r^2 avec la composante latente du groupe).

— `plot(type = c("inertia", "clusters", ...))` : **Visualisation**. Selon l'argument `type`, la méthode peut :
   — tracer la courbe de l'inertie en fonction de K (aide au choix du nombre de clusters) ;
   — représenter graphiquement la structure des clusters (par exemple via une carte factorielle ou un diagramme des corrélations) ;
   — mettre en évidence le degré d'appartenance des variables à chaque groupe.

Ces méthodes publiques sont conçues pour être réutilisées directement dans :
— le tutoriel en anglais (exemples reproductibles de `$fit()` et `$predict()`) ;
— l'application Shiny, qui appelle la classe `mmrClustVar` pour effectuer les calculs et afficher les résultats à l'utilisateur final.

2.0.3 Fonctions internes (helpers)

n plus des méthodes publiques, la classe `mmrClustVar` s'appuie sur plusieurs méthodes internes (privées) qui factorisent les opérations récurrentes et séparent clairement :

— la préparation des données,

— les calculs de distance,

— le cœur des algorithmes de clustering.

Les principales méthodes internes sont :

— `check_and_prepare`$_X(X, update\_structure = TRUE)$: $vrifielastructurede$`X`$, identifiecolonnesnu$ $standardiselescolonnesquantitativesactivessi$`FScale = TRUE`.

— `run_kmeans(X)` : implémente le cœur de k-means de variables (construction des composantes latentes `Z_k` via `prcomp`, distances basées sur $1 - r^2$, réallocation, convergence).

— `run_kmodes(X)` : implémente le k-modes de variables (prototypes catégoriels `_k` par *simple matching*, minimisation de la dissimilarité moyenne).

— `run_kprototypes(X‘` : implémente le k-prototypes mixte (prototypes numériques + catégoriels, distance mixte pondérée par `lambda`).

— `predict_one_variable(x_new, var_name)` : rattache une seule variable descriptive au modèle déjà appris, en calculant sa distance à chaque cluster.

Ces *helpers* ne sont pas destinés à être appelés directement par l'utilisateur, mais ils permettent de garder un code plus lisible, mieux testé et plus facile à documenter dans le rapport.

3 Algorithmes de classification de variables

3.1 k-means (variables quantitatives)

3.1.1 Principe

On se place dans le cadre du **clustering de variables** : on dispose d'une matrice de données $X = (x_{ij})_{1 \le i \le n,\, 1 \le j \le p}$ où n est le nombre d'observations et p le nombre de variables **quantitatives actives**.

Chaque variable X_j (colonne j) est traitée comme un **objet** décrit par son profil :

$$X_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^\top \in \mathbb{R}^n.$$

L'objectif est de partitionner les variables $\{X_1, \ldots, X_p\}$ en K groupes $G_1, \ldots, G_K$ de manière à regrouper ensemble les variables portant une information redondante. Pour chaque groupe G_k, on définit une **composante latente** Z_k : il s'agit de la première c**omposante principale** (1er axe d'ACP) calculée sur les variables de G_k.

L'algorithme cherche à maximiser

$$W = \sum_{k=1}^{K} \sum_{X_j \in G_k} r^2(X_j, Z_k),$$

où $r(X_j, Z_k)$ est la corrélation entre la variable X_j et la composante latente Z_k.

3.1.2 Distance utilisée

La corrélation entre une variable X_j et la composante latente Z_k mesure leur proximité informationnelle. On utilise le carré de la corrélation $r^2(X_j, Z_k)$ comme mesure de **similarité** : plus r^2 est élevée, plus la variable est représentée par la composante latente.

Pour exprime cette proximité sous forme de **dissimilarité**, on définit la distance :

$$d(X_j, Z_k) = 1 - r^2(X_j, Z_k).$$

Ainsi, affecter X_j au cluster revient à choisir le groupe minimisant cette distance :

$$\text{cluster}(X_j) = \arg \min_{1 \leq k \leq K} d(X_j, Z_k) = \arg \max_{1 \leq k \leq K} r^2(X_j, Z_k).$$

Ainsi, r^2 est la **mesure d'adhésion** d'une variable à un cluster, et $1 - r^2$ est la **distance intern**e utilisée dans l'algorithme.

Avant le calcul des corrélations, les variables quantitatives peuvent être **standardisées** :

$$X_j^{\text{std}} = \frac{X_j - \mu_j}{\rho_j},$$

où μ_j et ρ_j sont la moyenne et l'écart-type de la variable X_j.

3.1.3 Pseudo-code

```
Choisir K
Définir K variables comme noyau des groupes
Calculer les composantes latentes de chaque groupe
TANT QUE non convergence
    POUR toutes les variables
        Affecter la variable à la composante latente pour laquelle r^2 est maximal
    FIN POUR
    Calculer les composantes latentes de chaque groupe
FIN TANT QUE
```

3.2 k-modes (variables qualitatives)

3.3 Principe

On considère une matrice de données $X = (x_{ij})_{1 \leq i \leq n,\, 1 \leq j \leq p}$, où les p variables actives sont **qualitatives**. Chaque variable X_j est vue comme un **objet** décrit par la suite de ses modalités observées sur les n individus :

$$X_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^\top,$$

où chaque x_{ij} appartient à un ensemble fini de modalités.

L'objectif du k-modes de variables est de partitionner l'ensemble $\{X_1, \ldots, X_p\}$ en K groupes $G_1, \ldots, G_K$, de façon à regrouper dans un même cluster des variables qui prennent des modalités similaires sur les mêmes individus (profil catégoriel proche).

Pour chaque groupe G_k, on définit un **mode** (ou prototype) $M_k = (m_k(1), \ldots, m_k(n))^\top$ tel que, pour chaque ligne i, $m_k(i)$ est la modalité la plus fréquente parmi les variables du groupe G_k au niveau de l'individu i.

L'algorithme k-modes cherche une partition et des modes $\{G_k, M_k\}_{k=1}^K$ qui minimisent une **dissimilarité intra-cluster** basée sur la proportion de désaccords entre les profils des variables et le mode du groupe.

3.3.1 Distance utilisée

Pour deux profils catégoriels X_j et M_k, on utilise la **dissimilarité de simple matching** :

$$d(X_j, M_k) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(x_{ij} \neq m_k(i))},$$

c'est-à-dire la proportion de positions où la modalité de la variable X_j diffère de celle du mode M_k.

Plus $d(X_j, M_k)$ est petit, plus la variable X_j est proche du mode du groupe k (profil similaire). L'inertie intra-cluster associée à une partition $(G_1, \ldots, G_K)$ et à des modes $(M_1, \ldots, M_K)$ est alors :

$$\mathcal{I}_{\text{intra}} = \sum_{k=1}^{K} \sum_{X_j \in G_k} d(X_j, M_k) = \sum_{k=1}^{K} \sum_{X_j \in G_k} \left(\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(x_{ij} \neq m_k(i))} \right).$$

Le k-modes de variables cherche à **minimiser** cette inertie.

3.3.2 Pseudo-code

```
Choisir K
Définir K variables comme noyau des groupes
Calculer le mode de chaque groupe
TANT QUE non convergence
    POUR toutes les variables
        Affecter la variable au groupe dont le mode est le plus proche (dissimilarité de
    FIN POUR
    Pour chaque groupe, recalculer le mode à partir des variables qui lui sont affectées
FIN TANT QUE
```

3.4 k-prototypes

3.4.1 Principe

On dispose d'une matrice de données $X = (x_{ij})_{1 \le i \le n,\, 1 \le j \le p}$ contenant des variables **mixtes** :

— un sous-ensemble de variables quantitatives ;

— un sous-ensemble de variables qualitatives.

Chaque variable X_j est vue comme un **objet** à regrouper, décrit par son profil sur les n individus :

$$X_j = (x_{1j}, x_{2j}, \ldots, x_{nj})^\top.$$

L'objectif du k-prototypes de variables est de partitionner l'ensemble des variables $\{X_1, \ldots, X_p\}$ en K groupes $G_1, \ldots, G_K$ pouvant contenir à la fois des variables quantitatives et des variables qualitatives.

Pour chaque groupe G_k, on définit un **prototype mixte** P_k constitué de deux parties :

— une partie numérique : pour chaque individu i, une valeur moyenne $p_k^{\text{num}}(i)$ calculée à partir des variables quantitatives du groupe ;

— une partie catégorielle : pour chaque individu i, une modalité $p_k^{\text{cat}}(i)$ correspondant au mode parmi les variables qualitatives du groupe.

Intuitivement :

— les variables quantitatives d'un groupe sont proches de la partie numérique du prototype ;

— les variables qualitatives d'un groupe sont proches de la partie catégorielle du prototype.

L'algorithme cherche une partition et des prototypes $\{G_k, P_k\}_{k=1}^K$ qui minimisent une dissimilarité intra-cluster mélangeant distance numérique et distance catégorielle, pondérées par un paramètre $\lambda > 0$.

3.4.2 Distance utilisée

On distingue deux cas selon la nature de la variable X_j.

Si X_j est quantitative, on compare son profil au prototype numérique du groupe k. La distance utilisée est la même que pour k-means :

$$d_{\text{num}}(X_j, P_k) = 1 - r^2\big(X_j, P_k^{\text{num}}\big),$$

où $r\big(X_j, P_k^{\text{num}}\big)$ est la corrélation entre le profil de la variable et le profil numérique moyen du cluster.

Si X_j est qualitative, on compare son profil au prototype catégoriel du groupe k. On utilise la dissimilarité de simple matching, pondérée par un facteur $\lambda > 0$:

$$d_{\text{cat}}(X_j, P_k) = \lambda \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(x_{ij} \neq p_k^{\text{cat}}(i))}.$$

Le paramètre λ contrôle l'importance relative de la partie qualitative par rapport à la partie quantitative dans la formation des groupes.

Au final, la distance entre une variable X_j et un prototype P_k est définie par :

— $d(X_j, P_k) = d_{\text{num}}(X_j, P_k)$ si X_j est quantitative ;

— $d(X_j, P_k) = d_{\text{cat}}(X_j, P_k)$ si X_j est qualitative.

L'inertie intra-cluster est alors :

$$\begin{aligned}\mathcal{I}_{\text{intra}} &= \sum_{k=1}^{K} \sum_{X_j \in G_k} d(X_j, P_k) \\ &= \sum_{k=1}^{K} \left[\sum_{X_j \in G_k \cap \text{num}} (1 - r^2(X_j, P_k^{\text{num}})) + \sum_{X_j \in G_k \cap \text{cat}} \lambda \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{(x_{ij} \neq p_k^{\text{cat}}(i))} \right].\end{aligned}$$

et l'algorithme k-prototypes cherche à **minimiser** cette quantité.

3.4.3 Pseudo-code

```
Choisir K et le paramètre lambda > 0
Définir K variables comme noyau des groupes

Calculer le prototype mixte de chaque groupe (partie numérique : moyenne, partie catégor
TANT QUE non convergence
    POUR toutes les variables
        SI la variable est quantitative
        Calculer d_num avec chaque prototype
        SINON (variable qualitative)
            Calculer d_cat avec chaque prototype
        FIN SI

        Affecter la variable au groupe de distance minimale
    FIN POUR

    Pour chaque groupe, recalculer le prototype mixte :
        - partie numérique : moyenne des variables quantitatives du groupe
        - partie catégorielle : mode des variables qualitatives du groupe
FIN TANT QUE
```

4 Implémentation

4.1 Organisation du package

```
R
   mmrClustVar.R
   helpers.R
app
```

```
    app_shiny.R
 data
 tests
 DESCRIPTION
 NAMESPACE
 README.md
 rapport
     mmrClustVar.pdf
     mmrClustVar.zip
```

4.2 Choix d'implémentation

4.2.1 Standardisation des variables

Le centrage-réduction des variables quantitatives n'est pas obligatoire pour le clustering de variables, mais nous avons choisi de l'implémenter pour :

1. mettre toutes les variables sur une même échelle ;
2. stabiliser la corrélation utilisée dans k-means et k-prototypes, afin d'éviter les effets numériques liés aux variances proches de zéro.

Dans notre implémentation, la standardisation est appliquée uniquement aux variables **actives**, uniquement si `scale=TRUE`, et est réalisée **individuellement** (colonne par colonne).

Pour une variable X_j, on calcule :

— la moyenne : $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$,

— l'écart-type : $\rho_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\mu_j)^2}$,

— le profil standardisé : $X_j^{\text{std}} = \frac{X_j-\mu_j}{\rho_j}$.

4.2.2 Stratégies d'identification du nombre optimal de clusters K

Le package `mmrClustVar` n'impose pas un choix automatique de K, mais il expose, via `summary()` et `plot()`, les informations nécessaires pour que l'utilisateur puisse appliquer les critères suivants :

Inertie intra-cluster (méthode du coude). La méthode `summary()` affiche, pour un modèle donné, l'inertie intra-cluster totale ainsi qu'une décomposition par cluster. La méthode `plot(type = "inertia")` trace la courbe de l'inertie en fonction de K (pour une grille de valeurs testées), ce qui permet d'identifier visuellement une zone de "coude" où l'ajout de nouveaux clusters n'apporte plus de réduction significative de l'inertie.

Cohésion/séparation (corrélations intra/inter ou dissimilarités). Dans le cas du k-means de variables, `summary()` fournit, pour chaque cluster, des indicateurs de cohésion basés sur la corrélation moyenne $\overline{r^2(X_j, Z_k)}$ entre les variables et leur composante latente, ainsi que des indicateurs par variable (r^2 avec le cluster d'appartenance).

Pour k-modes et k-prototypes, `summary()` renvoie des dissimilarités moyennes intra-cluster (simple matching ou distance mixte), et un indicateur de dissimilarité pour chaque variable. La méthode `plot(type = "membership")` représente graphiquement ces degrés d'adhésion : variables bien représentées (adhésion forte / dissimilarité faible) vs variables limites.

Ces sorties permettent de juger si, pour un K donné, les clusters sont homogènes intérieurement et bien séparés.

Stabilité empirique des partitions. L'utilisateur peut relancer 'fit()' plusieurs fois (initialisations différentes, éventuellement via un paramètre d'itérations multiples) et comparer les partitions obtenues en s'appuyant sur les mêmes sorties (`summary()`, inertie, tailles de clusters, indicateurs d'adhésion). Des différences fortes d'une exécution à l'autre signalent un K instable, alors qu'une structure de clusters robuste se reconstitue de manière répétée.

Interprétabilité des clusters. `summary()` fournit des tableaux synthétiques par cluster (taille, inertie, indicateurs moyens) et par variable (cluster d'appartenance, adhésion, éventuellement variables les plus représentatives). `plot(type = "clusters")` donne une vue d'ensemble de la structure des groupes.

Ces sorties textuelles et graphiques facilitent l'interprétation qualitative des clusters, par la possibilité de décrire chaque groupe, d'identifier des thématiques communes entre variables, et de repérer les clusters peu lisibles.

Sélection des variables "noyaux" lors de l'initialisation des algorithmes de clustering. Pour chaque méthode (k-means, k-modes, k-prototypes), l'initialisation repose sur la sélection de K variables "noyaux". Plusieurs stratégies ont été considérées :

- initialisation aléatoire simple (pratique, reproductible via `set.seed()` ; adoptée par défaut.
- initialisation par maximum de diversité (sélection de variables maximisant leurs distances mutuelles)
- variantes inspirées de Forgy / McQueen.

4.2.3 Choix des variables descriptives ($\texttt{FX}_d escr$)

Le choix des variables descriptives ne repose pas sur un test statistique, mais sur des considérations méthodologiques analogues à celles de l'ACP et de PCAmix.

Nous considérons comme "descriptives" les variables qui :

1. ne doivent pas intervenir dans la construction des clusters ;
2. sont utilisées uniquement pour caractériser et interpréter les groupes ;
3. ne doivent pas influencer l'élaboration des centres, modes ou prototypes ;
4. sont techniquement compatibles (même nombre d'individus, type quantitatif ou qualitatif, pas de valeurs constantes).

Ainsi, les variables **actives** sont celles sur lesquelles le regroupement de variables est effectué, tandis que les variables **descriptives** sont rattachées a posteriori via la fonction `predict()`, analogie directe aux "variables supplémentaires" en ACP ou PCAmix.

Ainsi, `predict(X`$_new$`)` *traite chaque variable descriptive ind pendamment, et retourne* :

`son cluster prédit,`

`sa distance au centre/mode/prototype,`

`un indicateur d'adhésion (`r^2 `pour k-means, dissimilarité pour k-modes/k-prototypes)`

4.2.4 Fonctionnalités additionnelles

La classe `mmrClustVar` expose plusieurs fonctionnalités additionnelles destinées à faciliter l'utilisation du package et l'interprétation des résultats, notamment dans le contexte de l'application Shiny.

Extraction des distances variable-cluster. ne fonction dédiée permet d'extraire, pour chaque variable et pour chaque cluster, la distance utilisée par la méthode choisie.

Les résultats sont fournis sous forme de tableau long (variables × clusters) ou large (une ligne par variable, une colonne par cluster). Cela permet de visualiser le degré d'adhésion relatif d'une variable à tous les clusters, et d'identifier les variables "limites", proches de plusieurs groupes.

Profil moyen par cluster. Pour chaque cluster, `mmrClustVar` calcule un profil moyen :

- moyenne des variables quantitatives du groupe, par individu ;
- modalité majoritaire pour les variables qualitatives.

Ces profils moyens peuvent être utilisés comme résumé du contenu du cluster, comme base pour des graphiques (profils moyens comparés entre clusters), ou encore pour interpréter la nature de chaque groupe de variables.

Autres fonctionnalités envisagées. D'autres fonctionnalités ont été envisagées ou partiellement intégrées afin d'améliorer encore l'usage :

- **classement des variables au sein de chaque cluster** selon un indicateur d'adhésion (par exemple r^2 décroissant pour k-means), ce qui permet d'identifier les variables "centrales" et les variables "périphériques".
- **diagnostics de stabilité** : possibilité de relancer l'algorithme avec plusieurs initialisations et de calculer un indicateur de stabilité (taux d'accord des clusters variable par variable).
- **aide à la sélection de** K : fonctions utilitaires pour calculer l'inertie pour plusieurs valeurs de K et retourner une table prête à être tracée ou affichée dans Shiny.
- fonctions d'export des résultats : sauvegarde des partitions, des distances et des profils sous forme de fichiers CSV, permettant de documenter une analyse ou d'alimenter un rapport externe.

4.2.5 Choix des sorties de `print()`

Afin de garder une méthode `print()` succincte et d'éviter la surcharge, celle-ci affiche uniquement :

— la méthode ;
— le nombre de clusters ;
— le nombre de variables actives ;
— l'état de convergence ;
— l'inertie finale.

4.2.6 Choix des sorties de `summary()`

La méthode `summary()` a été conçue comme l'outil principal d'interprétation du modèle. Elle fournit une vision hiérarchisée des résultats :

1. un **résumé global** du modèle
   — la méthode de partitionnement utilisée ;
   — le nombre de clusters K ;
   — le nombre de variables actives ;
   — l'état de convergence (`TRUE` / `FALSE`) ;
   — l'inertie intra-cluster totale.
2. des **indicateurs par cluster** : Pour chaque cluster $k = 1, \ldots, K$, `summary()` affiche une petite table contenant :
   — la **taille du cluster** (nombre de variables dans G_k) ;
   — un indicateur d'**inertie intra** pour le cluster ;
   — un ou plusieurs indicateurs **de cohésion/adhésion**, selon la méthode de partitionnement utilisée.
3. des **indicateurs par variable** :
   — le nom de la variable ;
   — son cluster d'appartenance ;
   — un indicateur de **degré d'adhésion** au cluster :
4. des compléments spécifiques à la méthode de clustering utilisée :
   — - k-means : affichage des corrélations les plus fortes entre chaque variable et les composantes latentes, éventuellement limité aux top variables par cluster.
   — k-modes : résumé des modes par cluster (par exemple, modalités les plus fréquentes pour un sous-ensemble de modalités ou d'individus), afin de caractériser les profils catégoriels des groupes.
   — k-prototypes : décomposition de la distance moyenne en deux composantes contribution numérique / contribution catégorielle (pondérée par λ), pour voir si un cluster est plutôt structuré par les variables quantitatives ou par les variables qualitatives.

En complément de `plot(type = "inertia")`, les sorties de `summary()` aident à :
— juger si les clusters sont bien définis (cohésion, taille) ;

— vérifier que le nombre de clusters n'est ni trop faible (clusters très hétérogènes) ni trop grand (clusters minuscules ou redondants) ;
— préparer le rapport en fournissant directement des tableaux interprétables (taille des groupes, variables caractéristiques, indicateurs d'adhésion).

4.2.7 Choix des sorties de `plot()`

La méthode `plot()` fournit des visualisations pour aider à analyser un clustering de variables. Nous avons retenu les quatre graphiques suivants :

1. Inertie (`plot(type="inertia")`)
   — utilisation : visualiser la qualité globale du modèle.
   — contenu : la valeur de l'inertie intra-cluster, et éventuellement la position du modèle dans un *scree plot* si plusieurs K ont été testés.
2. Répartition des variables (`plot(type="clusters")`)
   — utilisation : comprendre la structure du modèle.
   — contenu : un barplot de la taille des clusters, et éventuellement une table variables-clusters dans une vignette Shiny.
3. Adhésion des variables `‘plot(type="membership")`)
   — utilisation : évaluer la stabilité des rattachements.
   — contenu : barres triées par coefficients d'adhésion, couleur par cluster.
4. Profils moyens (`plot(type="profiles")`)
   — utilisation : interpréter chaque cluster.
   — contenu : profils numériques moyens (k-means), fréquences modales (k-modes), ou combinaison des deux (k-prototypes).

4.2.8 Choix guidés par l'optimisation du code

Plusieurs choix d'implémentation ont été faits pour obtenir un code plus lisible, robuste et performant :

1. **Séparation des variables quantitatives et qualitatives** : dès `check_and_prepare_X()`, les indices numériques et catégoriels sont mémorisés, ce qui permet d'éviter des tests répétés, simplifie `fit()` et accélère les runs succesifs.
2. **Standardisation locale et à la volée** : les variables quantitatives sont centrées-réduites individuellement, sans stocker les paramètres → moins d'objets internes, helpers unifiés, prétraitement symétrique entre variables actives et descriptives.
3. **Prototypes représentés sous forme de listes** : pour chaque cluster, le prototype numérique (resp. catégoriel) sous forme de vecteurs numérique de longueur n (resp. vecteur de modalités dominantes) → structure simple, mise à jour rapide, pas de duplication inutile.

4. **Extraction ciblée des colonnes** : les sous-matrices sont extraites à la demande dans les algorithmes (`X[, cars_k, drop=FALSE]`) → usage mémoire réduit, évite les copies massives.
5. **Calculs vectorisés autant que possible** : distances calculées via `apply`ou opérations matricielles ; prototypage numérique basé sur `rowMeans` ; corrélation calculée en lot pour k-means/k-prototypes → boucles de mise à jour cluster par cluster.
6. **Préparation pour la parallélisation** : la structure choisie permettrait de paralléliser la mise à jour des prototypes cluster par cluster, et les calculs de distance variable par variable.
7. **Compatibilité Shiny** : les objets internes sont stockés dans un format compatible avec une visualisation rapide.

4.3 Implémentation du code R6

L'implémentation de la classe `mmrClustVar` repose sur une architecture modulaire structurée autour de trois méthodes principales : `initialize()`, `fit()` et `predict()`. Chaque méthode s'appuie sur un ensemble de fonctions internes (helpers) dédiées à des tâches élémentaires et testables, telles que la préparation des données, la standardisation, le calcul des distances ou la mise à jour des prototypes.

4.3.1 Constructeur `initialise()`

Le constructeur `initialize(method, K, scale = TRUE, lambda = 1, ...)` :
— enregistre les paramètres fournis par l'utilisateur (`method`, `K`, scale, `lambda`, etc.) ;
— initialise les attributs internes (`FMethod`, `FNbGroupes`, `FScale`, `FLambda`, `FConvergence`, `FX_active`, etc.) ;
— vérifie la cohérence des arguments (méthode valide, $K2$, valeur positive de `lambda`, etc.).

4.3.2 Méthode d'apprentissage : `fit(X)`

La méthode `fit()` effectue successivement :

1. **Validation et préparation des données** : Appel à `check_and_prepare_X()` qui
   — vérifie la présence de valeurs manquantes ;
   — identifie les colonnes numériques et catégorielles ;
   — retire ou gère les variables constantes ou incompatibles ;
   — sépare proprement les deux types (num/cat) dans des structures adaptées.
2. **Prétraitements**
   — standardisation des variables quantitatives via `scale_active_variables()` ;
   — aucune transformation sur les variables catégorielles (respect du principe de simple matching pour k-modes et k-prototypes).
3. **Sélection de l'algorithme** : appel à `run_kmeans()`, `run_kmodes` ou `run_kprototypes()`.

4. **Exécution de l'algorithme** : Chaque fonction interne
   — initialise ses prototypes ;
   — itère jusqu'à convergence ou nombre maximal d'itérations ;
   — utilise des fonctions vectorisées pour le calcul des distances ;
   — met à jour centres, modes ou prototypes mixtes ;
   — renvoie un objet contenant :
      — les clusters,
      — les prototypes,
      — l'inertie intra-cluster,
      — un indicateur de convergence.
5. **Stockage des résultats dans l'objet** : L'objet met à jour
   — `FClusters` : vecteur des affectations,
   — `FCenters` : centres / modes / prototypes,
   — `FInertia` : inertie totale,
   — **FConvergence** : booléen,
   — `FX_active` : variables actives utilisées.

La méthode `fit()` ne retourne rien, mais modifie l'état interne de l'objet

4.3.3 Méthode de rattachement : `predict(X_new)`

La méthode `predict()` rattache des variables descriptives (numériques ou catégorielles) aux clusters déjà construits par `fit()` en trois étapes :

1. **Validation des données nouvelles** : Vérification du nombre d'individus, de la présence éventuelle de valeurs manquantes, et de la compatibilité des types (quantitatif / facteur).
2. **Prétraitement**
   — Les variables descriptives ne sont pas standardisées explicitement car la distance utilisée repose sur la corrélation de Pearson, déjà invariante aux changements d'échelle.
   — Les variables catégorielles sont converties en facteur/caractère si nécessaire.
3. **Calcul des distances** r^2 : Pour chaque variable descriptive,
   — calcul de la distance à chaque centre / mode / prototype selon la méthode ;
   — sélection du cluster le plus proche ;
   — extraction d'indicateurs d'adhésion (corrélation, dissimilarité, distance mixte).

La sortie est un data frame contenant le nom de la variable, l'indice du cluster retenu et un indicateur d'adhésion.

4.3.4 Rôle des helpers internes

Chaque tâche élémentaire de `fit()` et `predict()` est isolée dans une fonction interne (helper).

5 Tests et validation

5.1 Tests unitaires réalisés pour les algorithmes de clustering

- Test minimal de l'algorithme : création de l'objet, apprentissage et affichage succinct
- Sorties principales
- Test de sensibilité à K
- Test des fonctions internes
- Robustesse de la convergence
- Cohérence de la prédiction

5.2 Test de l'interface Shiny avec différents réglages

- Appel du constructeur
- Appel de `fit(X)` sur un jeu de données adapté
- Appel à `summary()` pour vérifier
  - la taille des clusters,
  - l'inertie,
  - les indicateurs d'adhésion,
  - la lisibilité en console
- Appel de `plot(type = "inertia")`, `plot(type = "clusters")` et `plot(type = "membership")` pour valider la génération sans erreur des graphiques de base
- Appel de `predict(X_new)` sur des variables descriptives pour vérifier le bon rattachement aux clusters et la cohérence des indicateurs retournés.

6 Application Shiny

En complément de la classe R6 `mmrClustVar`, une application Shiny a été développée afin de fournir une interface interactive pour le clustering de variables. Cette application permet à un utilisateur non spécialiste de :

- charger ses propres données,
- choisir les variables actives et descriptives,
- paramétrer la méthode de clustering,
- lancer les calculs,
- explorer les résultats via des tableaux et des graphiques.

L'objectif est double : illustrer l'utilisation du package dans un contexte concret et proposer un outil pédagogique pour la compréhension du clustering de variables.

6.0.1 Fonctionnalités

L'application Shiny permet notamment de :

- importer un fichier de données (CSV, séparateur configurable) ;

— sélectionner séparément les variables actives/descriptives :
— choisir la méthode de clustering, la standardisation, K, λ.
— lancer l'apprentissage et afficher les principaux résultats ;
— lancer le rattachement des variables descriptives (prédiction) ;
— visualiser les indicateurs et les graphiques fournis.

6.0.2 Structure de l'interface

7 Pistes d'amélioration

Inventaire des pistes d'amélioration envisagées.

7.1 Refactorisation des calculs distance / adhésion

Actuellement, les calculs de similarité sont implémentés de manière indépendante dans les trois méthodes de clustering.

Une amélioration possible serait d'introduire de petites fonctions internes dédiées (helpers privés) pour encapsuler les formules de distance et d'adhésion :

```
# ---- amlioration : ajout d'un helper distance / adh sion ----
mmrClustVar <- R6::R6Class(
    "mmrClustVar",

    private = list(

        # ---- Helpers internes ----

        distance_adhesion_kmeans = function(x, z) {
            # x, z : vecteurs num riques de longueur n
            r <- suppressWarnings(
                stats::cor(x, z, use = "pairwise.complete.obs")
            )
            if (is.na(r)) r <- 0
            d2 <- 1 - r^2
            list(distance = d2, adhesion = r^2)
        },

        distance_adhesion_kmodes = function(x_char, z_cat) {
            # x_char, z_cat : vecteurs de caract res de longueur n
            mismatch <- x_char != z_cat
            d <- mean(mismatch, na.rm = TRUE)
            if (is.na(d)) d <- 1
```

```
            list(distance = d, adhesion = 1 - d)  # proportion de matches
        },

        distance_adhesion_kproto_num = function(x, z_num) {
            # m me logique que k-means
            private$distance_adhesion_kmeans(x, z_num)
        },

        distance_adhesion_kproto_cat = function(x_char, z_cat, lambda) {
            # dissimilarit  simple matching pond r e par lambda
            mismatch <- x_char != z_cat
            d_raw <- mean(mismatch, na.rm = TRUE)
            if (is.na(d_raw)) d_raw <- 1
            list(
                distance = lambda * d_raw,  # utilis  dans l'algo & predict
                adhesion = 1 - d_raw        # score d'adh sion dans [0,1]
            )
        },

    )
)

# --- Cas k-means ---

# refactoriser predict_one_variable()
        predict_one_variable = function(x_new, var_name) {
            if (method == "kmeans") {
            ...
                for (k in seq_len(K)) {
                    zk <- centers[[k]]
                    if (is.null(zk)) {
                        distances[k] <- NA_real_
                        adhesions[k] <- NA_real_
                    } else {
                        da <- private&distance_adhesion_kmeans(x_new, zk)
                        distances[k] <- d2
                        adhesions[k] <- r^2
                    }
                }
            }
```

```
        }

# refactoriser summary()
        summary = function(...) {
                        if (method == "kmeans") {
                # r^2(X_j, Z_k(j))
                X_mat <- as.matrix(X)
                for (j in seq_len(p)) {
                    kj <- clusters[j]
                    zk <- centers[[kj]]
                    xj <- X_mat[, j]
                    da <- private$distance_adhesion_kmeans(xj, zk)
                    membership[j] <- da$adhesion
                }
                membership_label <- "r^2 (correlation avec la composante late
            }
        }

# refactoriser plot(type='membership')
        plot = function(type = c("inertia", "clusters", "membership"), Ks = N

             if (method == "kmeans") {
                X_mat <- as.matrix(X)
                for (j in seq_len(p)) {
                    kj <- clusters[j]
                    zk <- centers[[kj]]
                    xj <- X_mat[, j]
                    da <- private$distance_adhesion_kmeans(xj, zk)
                    membership[j] <- da$adhesion
                }
            }
        }
```

7.2 Extensions

Plusieurs extensions ont également été considérées :

- ajout d'un onglet dédié au choix de K (en testant automatiquement plusieurs valeurs) ;
- intégration d'options de parallélisation pour les jeux de données volumineux ;
- ajout de contrôles plus fins sur la qualité des données (valeurs manquantes, variables constantes, etc.) ;
- templates d'exports automatiques (rapport HTML ou PDF généré à partir d'une analyse

exécutée dans Shiny)

8 Conclusion

Références

[1] Marie CHAVENT et al. « ClustOfVar : An R Package for the Clustering of Variables ». In : *Journal of Statistical Software* 50.13 (2012), p. 1-16. DOI : 10.18637/jss.v050.i13.

[2] Gilbert DE SOETE. « A least squares algorithm for fitting additive trees to proximity data ». In : *Psychometrika* 48.4 (1983), p. 621-626. URL : https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2784845.

[3] Zhexue HUANG. « Extensions to the k-means algorithm for clustering large data sets with categorical values ». In : *Data Mining and Knowledge Discovery* 2 (1998), p. 283-304. URL : https://cse.hkust.edu.hk/~qyang/Teaching/537/Papers/huang98extensions.pdf.

[4] François HUSSON, Julie JOSSE et Jérôme PAGÈS. *Exploratory Multivariate Analysis by Example Using R*. 2e éd. Chapman et Hall/CRC, 2017. URL : http://staff.ustc.edu.cn/~ynyang/vector/books/Husson-Le-Pages.pdf.

[5] Leonard KAUFMAN et Peter J. ROUSSEEUW. *Finding Groups in Data : An Introduction to Cluster Analysis*. Wiley, 2005. URL : https://www.researchgate.net/profile/Peter-Rousseeuw/publication/220695963_Finding_Groups_in_Data_An_Introduction_To_Cluster_Analysis/links/60fbbe85169a1a0103b20e91/Finding-Groups-in-Data-An-Introduction-To-Cluster-Analysis.pdf.

[6] J. MACQUEEN. « Some methods for classification and analysis of multivariate observations ». In : *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. 1967, p. 281-297. URL : https://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class14/macqueen.pdf.

[7] R CORE TEAM. *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. 2024. URL : https://www.r-project.org/.

[8] Ricco RAKOTOMALALA. *Supports de cours : Classification, Programmation R, Classification de variables*. 2024. URL : https://eric.univ-lyon2.fr.