Université Lumière Lyon 2

Master 2 SISE
Statistique et Informatique pour la Science des données

Projet mmrClustVar

Clustering de variables et implémentation R6

Auteurs :
Marin Nagy
Mazilda Zehraoui
Rina Razafimahefa

Encadrement :
Ricco RAKOTOMALALA

Année universitaire : 2025–2026

Table des matières

Introduction

La plupart des algorithmes de classification non supervisée travaillent sur des *individus* : on cherche à regrouper des lignes d'un tableau de données en classes homogènes. Dans de nombreux contextes appliqués (marketing, sciences sociales, psychologie, musique, etc.), il est tout aussi intéressant de regrouper les *variables* entre elles : identifier des paquets de variables qui mesurent globalement la même chose, ou qui sont fortement redondantes, permet à la fois de simplifier l'interprétation et de réduire la dimension des données.

Ce projet, réalisé dans le cadre du Master 2 SISE de l'Université Lyon 2, s'inscrit dans cette perspective. L'objectif est double :

— du point de vue **méthodologique**, implémenter plusieurs variantes de clustering de variables couvrant différents types de données (numériques, qualitatives, mixtes) ;

— du point de vue **logiciel**, proposer un package R structuré autour de classes R6, avec une interface haut niveau homogène et une application Shiny pour l'exploration interactive.

Nous avons développé le package `mmrClustVar` qui propose quatre algorithmes principaux :

— un k-means de variables pour données entièrement numériques ;

— un k-modes de variables pour données entièrement qualitatives ;

— un k-prototypes de variables pour données mixtes ;

— un k-medoids de variables basé sur une matrice de dissimilarités.

Chaque moteur d'algorithme hérite d'une classe abstraite `mmrClustVarBase` qui gère le flux de travail générique : préparation des données, appel à l'algorithme, stockage des résultats, indicateurs de convergence, visualisation, etc. L'utilisateur final interagit essentiellement avec une classe *façade* `mmrClustVar` et une application Shiny.

Ce document constitue une documentation étendue, non contrainte en longueur, qui sert de référence détaillée à notre travail : il reprend le cahier des charges du projet, le confronte aux choix méthodologiques et à l'architecture logicielle, décrit en détail les quatre algorithmes, les jeux de tests, le protocole d'analyse recommandé, ainsi qu'une étude de cas complète sur un jeu de données fictif `metal_universe`.

Chapitre 1

Contexte du projet et cahier des charges

1.1 Contexte pédagogique et scientifique

Le projet est réalisé dans le cadre du module de Programmation R du Master 2 SISE. Le sujet fournit un canevas général :

— implémenter des algorithmes de **clustering de variables** en s'inspirant notamment du package `ClustOfVar` [1] et des supports de cours de classification et de programmation R [6] ;

— utiliser une **architecture objet** en R, basée sur R6 [8], afin de factoriser les comportements communs et de permettre l'extension future du package ;

— fournir un **front-end utilisateur** convivial via une application Shiny, pour pouvoir :

  — charger des jeux de données (internes ou utilisateurs) ;

  — choisir les variables actives et supplémentaires ;

  — lancer les différents algorithmes de clustering de variables ;

  — visualiser et exporter les résultats.

Sur le plan méthodologique, le projet se situe à l'intersection de plusieurs références classiques : la classification de variables telle qu'implémentée dans `ClustOfVar` [1], les algorithmes de k-means et k-medoids [3, 5], leurs extensions aux données catégorielles [4], et les méthodes d'analyse factorielle de type ACP ou ACM utilisées pour définir des composantes latentes [2].

1.2 Cahier des charges fonctionnel

Le sujet imposait un certain nombre de fonctionnalités, que l'on peut résumer ainsi.

Clustering de variables pour différents types de données

— Permettre le **clustering de variables numériques** : regroupement de colonnes numériques partageant une même structure de corrélation.

— Permettre le **clustering de variables qualitatives** : regroupement de colonnes catégorielles partageant des profils de modalités semblables.

— Permettre le **clustering de variables mixtes** : capacité à traiter simultanément des variables numériques et catégorielles.

Choix d'algorithmes

Le sujet suggère fortement l'utilisation d'un algorithme de type *k-means de variables*, inspiré de [1], où l'on maximise une inertie inter-classe équivalente à une somme de corrélations au carré entre variables et composantes latentes. Il encourage également :

— l'usage d'un **algorithme de réallocation** itératif : affectation des variables aux groupes, recalcul des prototypes, itérations jusqu'à convergence ;

— l'exploration de méthodes alternatives pour les données qualitatives (k-modes) et mixtes (k-prototypes, k-medoids).

Architecture et interface

— Concevoir une **architecture objet** claire avec une classe de base factorisant la logique commune (gestion des données, convergence, inertie, sorties).

— Fournir une **façade** unique pour l'utilisateur (`mmrClustVar`), avec une méthode `$fit()` et une méthode `$predict()` pour attacher des variables supplémentaires à une partition pré-apprise.

— Développer une **application Shiny** intégrant :

  — chargement de données internes / utilisateur ;

  — paramètres de modèle (méthode, K, mise à l'échelle, paramètre λ pour les mixtes) ;

  — affichage des résumés, graphiques, diagnostics ;

  — export des résultats sous forme de fichiers texte / CSV / ZIP.

1.3 Validation du cahier des charges par mmrClustVar

Le package `mmrClustVar` implémente quatre algorithmes principaux qui couvrent les cas d'usage attendus :

— **k-means de variables** (`mmrClustVarKMeans`) :

  — variables actives strictement numériques ;

  — réallocation itérative des variables pour maximiser la somme des r^2 à des composantes principales locales ;

  — inertie définie comme $\sum_j (1 - r^2(x_j, Z_{c(j)}))$.

— **k-modes de variables** (`mmrClustVarKModes`) :

  — variables actives strictement catégorielles ;

  — distance basée sur la proportion de désaccords (*simple matching*) entre chaque variable et un profil modal de groupe ;

  - algorithme de réallocation itératif analogue à k-means.
- **k-prototypes de variables** (`mmrClustVarKPrototypes`) :
  - variables actives mixtes (numériques + qualitatives) ;
  - prototype mixte composé d'une composante numérique Z_k (ACP locale) et d'un profil catégoriel $m_k(i)$;
  - distance hybride $d_{\text{total}} = d_{\text{num}} + \lambda\, d_{\text{cat}}$;
  - réallocation itérative.
- **k-medoids de variables** (`mmrClustVarKMedoids`) :
  - variables actives potentiellement mixtes ;
  - matrice de dissimilarités entre paires de variables :

$$d(x_j, x_\ell) = \begin{cases} 1 - r^2(x_j, x_\ell) & \text{si les deux sont numériques,} \\ \text{simple matching}(x_j, x_\ell) & \text{si les deux sont qualitatives,} \\ 1 & \text{sinon (types mixtes);} \end{cases}$$

  - médoïde comme variable la plus centrale de chaque cluster.

Les quatre moteurs héritent d'une classe de base `mmrClustVarBase` qui :

- garantit une interface homogène (`$fit()`, `$predict()`, `$summary()`, `$plot()`) ;
- gère les structures communes : données actives, clusters, prototypes, inertie, convergence, types de variables ;
- fournit un mécanisme de **réallocation** pour les méthodes k-means, k-modes et k-prototypes ;
- expose des indicateurs de **membership** (adhésion, distance intra-classe, inertie expliquée).

L'application Shiny `mmrClustVar_app` permet enfin de valider l'aspect interactif demandé : chargement de jeux internes (dont un jeu fictif `metal_universe`), exécution des algorithmes, restitution graphique, export.

Chapitre 2

Rappels théoriques

2.1 Clustering de variables vs clustering d'individus

Dans un contexte classique de classification non supervisée, on dispose d'un tableau de données X de dimension $n \times p$ où n est le nombre d'individus et p le nombre de variables. La plupart des méthodes (k-means, k-medoids, hierarchique, etc.) opèrent en ligne : on clusterise les individus.

Le **clustering de variables** adopte une perspective duale :

— les individus $\{1, \ldots, n\}$ restent des points de référence ;

— les variables $\{x_1, \ldots, x_p\}$ sont vues comme des vecteurs de $\mathbb{R}^n$ (numérique) ou des profils de modalités (qualitatif) ;

— l'objectif est de regrouper les colonnes $\{x_j\}$ en K groupes homogènes, de sorte que les variables d'un même groupe "mesurent la même chose".

Dans le cas numérique, une approche typique est de chercher, pour chaque cluster k, une **composante principale locale** $Z_k \in \mathbb{R}^n$ qui résume les variables du groupe [1]. On maximise alors :

$$\sum_{k=1}^{K} \sum_{x_j \in C_k} r^2(x_j, Z_k),$$

où $r^2(x_j, Z_k)$ est le carré de la corrélation entre la variable x_j et la composante Z_k.

Pour des variables qualitatives, on peut, au contraire, utiliser des profils modaux : pour chaque cluster k, on définit un profil $m_k(i)$ qui, pour chaque individu i, donne la modalité la plus fréquente au sein du groupe. On cherche alors à minimiser une dissimilarité de type simple matching.

Dans le cas mixte, les deux idées sont combinées dans un prototype qui possède une partie numérique et une partie catégorielle, pondérées par un facteur $\lambda > 0$.

2.2 Mesures de similarité et de dissimilarité

2.2.1 Corrélation au carré pour les variables numériques

Pour deux variables numériques x et z, le package `mmrClustVar` utilise la corrélation au carré comme mesure de similarité :

$$r^2(x, z) = [\mathrm{cor}(x, z)]^2 ,$$

avec un calcul robuste utilisant uniquement les paires complètes d'observations. La dissimilarité associée est alors :

$$d_{\mathrm{num}}(x, z) = 1 - r^2(x, z),$$

bornée entre 0 et 1 (0 = parfait alignement linéaire, 1 = aucune corrélation).

2.2.2 Simple matching pour les variables qualitatives

Pour deux variables qualitatives x et m observées sur les mêmes individus, on définit une dissimilarité *simple matching* par :

$$d_{\mathrm{cat}}(x, m) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[x_i \neq m_i],$$

avec $\mathbb{I}$ l'indicatrice logique. C'est tout simplement la proportion d'individus pour lesquels les deux variables n'ont pas la même modalité. La similarité associée est :

$$\mathrm{adhésion}(x, m) = 1 - d_{\mathrm{cat}}(x, m),$$

interprétable comme une proportion de "matches".

2.2.3 Dissimilarité mixte

Pour le k-prototypes de variables, on combine les deux mondes via un prototype mixte $\mathcal{C}_k = (Z_k, m_k)$, où :

— Z_k est un profil numérique (composante principale locale) ;

— m_k est un profil catégoriel (modalité dominante par individu).

On définit alors pour une variable x_j :

$$d_{\mathrm{total}}(x_j, \mathcal{C}_k) = \begin{cases} 1 - r^2(x_j, Z_k), & \text{si } x_j \text{ est numérique,} \\ \lambda \cdot d_{\mathrm{cat}}(x_j, m_k), & \text{si } x_j \text{ est catégorielle,} \end{cases}$$

en ignorant le terme non pertinent (une variable numérique ne contribue pas à la partie catégorielle, et inversement).

2.2.4 Dissimilarité entre variables pour k-medoids

Pour k-medoids, on a besoin d'une dissimilarité *symétrique* entre paires de variables (x_j, x_ℓ), utilisable par l'algorithme PAM [5]. Nous utilisons :

$$d(x_j, x_\ell) = \begin{cases} 1 - r^2(x_j, x_\ell), & \text{si les deux sont numériques,} \\ d_{\text{cat}}(x_j, x_\ell), & \text{si les deux sont qualitatives,} \\ 1, & \text{si les types sont mixtes (numérique vs catégoriel).} \end{cases}$$

Chapitre 3

Architecture logicielle du package `mmrClustVar`

3.1 Vue d'ensemble

L'architecture de `mmrClustVar` repose sur trois niveaux :

1. une **classe de base** `mmrClustVarBase` (R6) qui encapsule le comportement générique du clustering de variables ;
2. quatre **classes spécialisées** héritées :
   - `mmrClustVarKMeans` ;
   - `mmrClustVarKModes` ;
   - `mmrClustVarKPrototypes` ;
   - `mmrClustVarKMedoids` ;
3. une **façade** `mmrClustVar` qui :
   - décide de la méthode effective à utiliser (`auto`, `kmeans`, `kmodes`, `kprototypes`, `kmedoids`) ;
   - délègue le travail à l'engine interne ;
   - fournit une interface simple pour l'utilisateur.

Cette structure « base → engines → façade » facilite l'ajout futur de nouvelles méthodes de clustering de variables sans changer l'interface utilisateur.

3.2 Classe de base `mmrClustVarBase`

La classe `mmrClustVarBase` contient :

- les **champs privés** :
  - `FMethod`, `FNbGroupes`, `FScale`, `FLambda` ;
  - `FX_active`, `FX_new` ;
  - `FClusters`, `FCenters`, `FInertia`, `FConvergence` ;
  - `FNumCols`, `FCatCols`.
- les **méthodes publiques**, communes à tous les moteurs :

- `$initialize(K, scale, lambda, method_name)` ;
- `$fit(X)` : vérification des données, filtrage des lignes aberrantes, standardisation optionnelle, appel à `run_clustering()` ;
- `$predict(X_new)` : affectation de variables supplémentaires par appel à `predict_one_variable()` ;
- `$summary()`, `$print()`, `$plot(type)` ;
- getters : `$get_clusters()`, `$get_centers()`, `$get_inertia()`, `$get_convergence()`, `$get_method()`, `$get_K()`, `$get_X_descr()`.

— les **hooks privés abstraits** que les classes enfants doivent implémenter :

- `run_clustering(X)` ;
- `predict_one_variable(x_new, var_name)` ;
- `summary_membership_impl()` ;
- `plot_membership_impl()`, `plot_profiles_impl()`.

L'essentiel de la logique commune (gestion des NA, standardisation, calcul de la corrélation au carré, simple matching) est factorisé dans cette classe.

3.3 Façade `mmrClustVar`

La façade `mmrClustVar` fournit une interface haut niveau :

— choix de la méthode : `method = "auto"` par défaut ;

— choix du nombre de clusters K ;

— option de standardisation des variables numériques : `scale = TRUE` ;

— paramètre de pondération des variables catégorielles : `lambda`.

La méthode `$fit(X)` :

1. vérifie que $p \geq 2$;
2. appelle `decide_effective_method(X)` pour choisir entre k-means, k-modes ou k-prototypes lorsque `method = "auto"` ;
3. construit un engine de type `mmrClustVarKXXX` via `create_engine()` ;
4. appelle `engine$fit(X)` et stocke l'engine dans `private$FEngine`.

La façade propose en outre :

— `$compute_inertia_path(K_seq, X)` : calcul de l'inertie pour plusieurs valeurs de K (méthode du coude) ;

— `$interpret_clusters(style)` : délégation aux engines, pour une interprétation textuelle ;

— `$plot(type)` : accès aux différents graphiques (inertie, tailles de clusters, membership, profils, factor map / dendrogramme).

3.4 Application Shiny

L'application Shiny `mmrClustVar_app` (`inst/shiny/mmrClustVar_app/app.R`) constitue la vitrine interactive du package. Elle offre :

- un volet gauche pour :
  - la sélection de la source de données (jeu interne ou fichier utilisateur CSV/XLSX) ;
  - le choix des variables actives et supplémentaires ;
  - le réglage des paramètres du modèle (méthode, K, standardisation, λ) ;
  - le lancement du clustering et l'attachement des variables supplémentaires.
- un volet principal avec un `tabsetPanel` :
  - onglet `Summary` : affichage de `print()` et `summary()` ;
  - onglet `Interpretation` : interprétation textuelle ;
  - onglet `Clusters` : tableau des affectations et tailles de clusters ;
  - onglet `Plots` : inertie, distribution, membership, profils, factor map / dendrogramme ;
  - onglet `Supplementary variables` : résultats de `$predict()` ;
  - onglet `Diagnostics & export` : indicateurs de convergence et boutons d'export.

Chapitre 4

Description détaillée des quatre algorithmes

Dans ce chapitre, nous reprenons les quatre algorithmes implémentés dans `mmrClustVar`, en donnant à chaque fois :

— une description mathématique ;

— le pseudo-code (tel qu'utilisé dans le rapport court) ;

— les indicateurs de membership ;

— les forces et limitations.

4.1 K-means de variables (numérique)

4.1.1 Principe mathématique

On suppose que les variables actives sont toutes numériques. On cherche à partitionner $\{x_1, \ldots, x_p\}$ en K clusters $\{C_1, \ldots, C_K\}$, et pour chaque cluster C_k à construire une **composante principale locale** Z_k (premier axe d'une ACP locale) telle que l'on maximise :

$$\mathcal{J} = \sum_{k=1}^{K} \sum_{x_j \in C_k} r^2(x_j, Z_k).$$

L'algorithme utilisé est de type *Lloyd* :

— étape de reconstruction des prototypes : pour chaque cluster k, calcul de Z_k ;

— étape de réallocation : pour chaque variable x_j, affectation au cluster $k^\star$ maximisant $r^2(x_j, Z_k)$.

L'inertie intra-classe associée est :

$$W = \sum_{k=1}^{K} \sum_{x_j \in C_k} \left(1 - r^2(x_j, Z_k)\right).$$

4.1.2 Pseudo-code

```
Choisir K
D finir K variables comme noyau initial des groupes
Calculer la composante latente Z_k de chaque groupe (ACP locale)
TANT QUE non convergence
    POUR chaque variable x_j
        Pour chaque groupe k, calculer r^2(x_j, Z_k)
        Affecter x_j au groupe k* pour lequel r^2(x_j, Z_k) est
           maximal
    FIN POUR
    Recalculer chaque composante latente Z_k par ACP locale
FIN TANT QUE
```

4.1.3 Indicateurs de membership

Pour chaque variable x_j :

— **adhésion** : $a_j = r^2(x_j, Z_{c(j)})$;

— **distance intra-classe** : $d_j = 1 - a_j$;

— **inertie expliquée globale** :

$$\mathrm{IE} = 1 - \bar{d} = \bar{a} = \frac{1}{p} \sum_{j=1}^{p} a_j.$$

Ces quantités sont calculées et résumées par `summary_membership_impl()`.

4.2 K-modes de variables (catégoriel)

4.2.1 Principe mathématique

On suppose ici que toutes les variables actives sont qualitatives. Pour chaque cluster C_k, on définit un **profil modal** $m_k(i)$, qui pour chaque individu i est la modalité la plus fréquente parmi les variables du cluster :

$$m_k(i) = \mathrm{mode}\{x_j(i) : x_j \in C_k\}.$$

Pour une variable x_j, la dissimilarité au profil modal est :

$$d_{\mathrm{cat}}(x_j, m_k) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[x_j(i) \neq m_k(i)].$$

On cherche à minimiser l'inertie intra-classe :

$$W = \sum_{k=1}^{K} \sum_{x_j \in C_k} d_{\mathrm{cat}}(x_j, m_k).$$

4.2.2 Pseudo-code

```
Choisir K
D finir K variables comme noyau des groupes
Calculer le profil modal m_k(i) de chaque groupe
    (modalit  la plus fr quente parmi les variables du groupe pour
       chaque individu i)

TANT QUE non convergence
    POUR chaque variable x_j
        Pour chaque groupe k, calculer la dissimilarit  d(x_j, m_k)
            (d = proportion de d saccords : simple matching)
        Affecter x_j au groupe k* pour lequel d(x_j, m_k) est minimale
    FIN POUR

    Recalculer le profil modal m_k(i) de chaque groupe
FIN TANT QUE
```

4.2.3 Indicateurs de membership

Pour chaque variable x_j :

— **distance** $d_j = d_{\text{cat}}(x_j, m_{c(j)})$;

— **adhésion** $a_j = 1 - d_j$;

— inertie expliquée globale $\text{IE} = 1 - \bar{d} = \bar{a}$.

4.3 K-prototypes de variables (mixte)

4.3.1 Principe mathématique

Les variables actives peuvent être numériques, qualitatives, ou un mélange des deux. Pour chaque cluster C_k, on construit un prototype mixte :

$$\mathcal{C}_k = (Z_k, m_k),$$

où :

— Z_k est la composante principale locale calculée sur les variables numériques du cluster (ACP sur les colonnes numériques de C_k) ;

— $m_k(i)$ est le profil modal catégoriel (modalité la plus fréquente par individu parmi les variables qualitatives du cluster).

La distance d'une variable x_j au prototype est :

$$d_{\text{total}}(x_j, \mathcal{C}_k) = \begin{cases} 1 - r^2(x_j, Z_k), & \text{si } x_j \text{ est numérique,} \\ \lambda \cdot d_{\text{cat}}(x_j, m_k), & \text{si } x_j \text{ est catégorielle,} \end{cases}$$

avec $\lambda > 0$ un paramètre de pondération du bloc catégoriel. L'inertie intra-classe devient :

$$W = \sum_{k=1}^{K} \sum_{x_j \in C_k} d_{\text{total}}(x_j, \mathcal{C}_k).$$

4.3.2 Pseudo-code

```
Choisir K et le param tre de pond ration
D finir K noyaux de groupes (variables initiales)
Calculer pour chaque groupe :
    - la composante latente num rique Z_k (ACP locale sur les
       variables num riques)
    - le profil modal cat goriel m_k(i) (modalit  la plus fr quente
        par individu)
TANT QUE non convergence
    POUR chaque variable x_j
        Si x_j est num rique :
            Pour chaque groupe k, calculer d_num(x_j, Z_k) = 1 - r^2(
               x_j, Z_k)
        Si x_j est cat gorielle :
            Pour chaque groupe k, calculer d_cat(x_j, m_k)
                (d_cat = proportion de d saccords : simple matching)
        Pour chaque groupe k :
            Calculer la distance totale
                d_total(x_j, C_k) = d_num(x_j, Z_k) +    * d_cat(x_j,
                   m_k)
                (en ignorant le terme non pertinent si x_j n'est que
                   num rique
                 ou uniquement cat gorielle)
        Affecter x_j au groupe k* pour lequel d_total(x_j, C_k) est
           minimale
    FIN POUR
    Recalculer Z_k (partie num rique) et m_k(i) (partie cat gorielle
       ) pour chaque groupe
FIN TANT QUE
```

4.3.3 Indicateurs de membership

Pour chaque variable x_j :

- type : numérique ou catégoriel ;
- distance d_j (incluant éventuellement le facteur λ) ;
- adhésion a_j :
  - numérique : $a_j = r^2(x_j, Z_{c(j)})$;
  - catégorielle : $a_j = 1 - d_{\text{cat}}(x_j, m_{c(j)})$.

Le package calcule des résumés séparés (adhésion moyenne numérique / catégorielle) pour mieux interpréter la qualité de la partition.

4.4 K-medoids de variables (dissimilarités)

4.4.1 Principe mathématique

Le k-medoids se base sur une matrice de dissimilarités entre variables. On construit d'abord une matrice $D \in \mathbb{R}^{p \times p}$ avec :

$$D_{j\ell} = d(x_j, x_\ell),$$

où d est définie comme dans la section précédente (1 - r^2 pour numériques, simple matching pour qualitatifs, 1 pour mixte).

L'algorithme PAM [5] cherche ensuite un ensemble de K **médoïdes** (variables centrales) $\{m_1, \ldots, m_K\}$ qui minimisent :

$$W = \sum_{k=1}^{K} \sum_{x_j \in C_k} d(x_j, m_k).$$

4.4.2 Pseudo-code

```
Choisir K
Construire la matrice de dissimilarit s D entre toutes les p
   variables
    (par exemple : 1 - r^2 pour les num riques , simple matching pour
       les qualitatives)
Choisir K variables comme m do des initiaux
TANT QUE non convergence
    POUR chaque variable x_j
        Pour chaque m do de m_k, calculer d(x_j, m_k)    partir de D
        Affecter x_j au groupe k* du m do de le plus proche
    FIN POUR
    POUR chaque groupe C_k
        Pour chaque variable candidate x_j dans C_k
            Calculer le co t total :
                co t(x_j) = somme des distances d(x_j, x_l) pour
                   toutes les x_l dans C_k
        Choisir comme nouveau m do de la variable x_j avec le co t
           minimal
    FIN POUR
FIN TANT QUE
```

4.4.3 Indicateurs de membership

Pour chaque variable x_j :

— distance $d_j = d(x_j, m_{c(j)})$;

— adhésion $a_j = 1 - d_j$ (bornée par $[0, 1]$ si d est bornée).

Chapitre 5

Jeux de données intégrés et dataset `metal_universe`

5.1 Liste des jeux internes

L'application Shiny propose plusieurs jeux de données internes illustrant différents cas de figure :

- `iris_num` : version numérique d'`iris` (mesures de fleurs) ;
- `iris_mixed` : version mixte avec la variable de classe en qualitatif ;
- `mtcars_num` : données de consommation de voitures, uniquement numériques ;
- `airquality_num` : mesures de qualité de l'air (numérique) ;
- `arthritis_cat`, `titanic_cat`, `housevotes_cat` : jeux essentiellement qualitatifs ;
- `credit_mix`, `adult_small` : jeux mixtes ;
- `metal_universe` : jeu fictif construit autour de groupes de metal.

Ces jeux permettent de tester chacune des méthodes dans des contextes contrôlés (numérique pur, qualitatif pur, mixte).

5.2 Le dataset fictif `metal_universe`

5.2.1 Structure générale

Le jeu `metal_universe` décrit des groupes de metal (ou assimilés), avec les variables suivantes :

- variables **réalistes** :
  - `group_name` : nom du groupe ;
  - `subgenre` : sous-genre (black, death, doom, power, etc.) ;
  - `country` : pays d'origine ;
  - `cat_front_gender` : genre du/de la chanteur · se principal · e.
- variables numérique **fictives mais plausibles** :

    - `num_bpm_pref` : tempo préféré (bpm), tiré au hasard dans des intervalles “réalistes” selon le sous-genre ;
    - `num_blastbeat_tolerance` : tolérance aux blastbeats ;
    - `num_distortion_preference` : préférence pour la distorsion ;
    - `num_aggressivity_score` : score d'agressivité ;
    - `num_chain_listen_hours`, `num_heaviness_score`, `num_speed_score`, etc. ;
- variables de **bruit aléatoire** : colonnes générées purement au hasard, sans structure ;
- variables **dérivées** :
    - une dizaine de dérivées de variables numériques (par exemple, transformations linéaires ou bruitées de variables de base) ;
    - une dizaine de dérivées de variables catégorielles (regroupements, recodages, etc.).

5.2.2 Important : caractère fictif des données

Il est essentiel de souligner que :

- les 4 premières colonnes (`group_name`, `subgenre`, `country`, `cat_front_gender`) sont basées sur des informations réalistes (*mais le jeu n'est pas exhaustif ni vérifié au sens musicologique*) ;
- le reste des colonnes est **fictif** :
    - valeurs générées au hasard dans des plages plausibles (ex. bpm) ;
    - colonnes bruit de type random ;
    - colonnes dérivées construites artificiellement.

Ce dataset est conçu comme un **terrain de jeu pédagogique** permettant d'illustrer :

- la capacité des algorithmes à retrouver des groupes de variables redondantes ;
- la différence de comportement entre les algos (numeric-only vs mixte vs medoids) ;
- la sensibilité aux variables bruitées.

Aucun web scraping réel n'est utilisé dans cette version : toutes les composantes sont soit manuelles soit simulées.

Chapitre 6

Protocole d'analyse recommandé

Dans cette section, nous décrivons un scénario d'utilisation typique de `mmrClustVar`, valable aussi bien via la façade R que via l'application Shiny.

6.1 Étape 1 — Préparation des données

1. Charger le jeu de données depuis R ou via l'interface Shiny (jeu interne ou fichier utilisateur).
2. Vérifier et, si besoin, corriger :
   — les types de variables (numeric vs factor/character) ;
   — la présence de valeurs manquantes extrêmes ;
   — les doublons éventuels de colonnes.
3. Sélectionner :
   — les **variables actives** (celles à clusteriser) ;
   — les **variables supplémentaires** (celles à projeter a posteriori dans les clusters).

6.2 Étape 2 — Choix du nombre de clusters K

Le choix de K peut être guidé par :

— un raisonnement métier (par exemple, 3 grands blocs de variables attendus) ;

— une exploration systématique de l'inertie intra-classe via `$compute_inertia_path()` :
   — on calcule l'inertie pour $K = 2, \ldots, K_{\max}$ (avec $K_{\max} \leq p$) ;
   — on cherche un "coude" dans la courbe inertie vs K.

6.3 Étape 3 — Choix de la méthode

En fonction des types de variables actives :

— **purement numérique** :
   — méthode `"kmeans"` (ou `"auto"`) ;

  - l'interprétation se fait via les r^2 aux composantes locales.
- **purement catégoriel** :
  - méthode `"kmodes"` (ou `"auto"`) ;
  - interprétation via les profils modaux et les proportions de matches.
- **mixte** :
  - méthodes possibles : `"kprototypes"` ou `"kmedoids"` ;
  - `kprototypes` travaille sur un prototype latent (numérique + catégoriel) ;
  - `kmedoids` travaille sur des variables-médoïdes, plus interprétables mais potentiellement plus coûteux.

6.4 Étape 4 — Ajustement et diagnostics

Une fois le modèle ajusté :

- vérifier la convergence (`$get_convergence()`) ;
- examiner l'inertie totale et, si besoin, comparer plusieurs K ;
- regarder les **tailles de clusters** et surveiller l'apparition de groupes très petits ou trop gros ;
- analyser les **indicateurs de membership** :
  - adhésion moyenne par cluster ;
  - variables avec faible adhésion (potentiellement peu typiques de leur groupe).

6.5 Étape 5 — Interprétation et variables supplémentaires

Enfin :

- utiliser `$interpret_clusters()` pour une première lecture textuelle des clusters ;
- utiliser la méthode `$predict()` pour attacher des variables supplémentaires à la partition existante ;
- visualiser les résultats :
  - barplots de membership ;
  - heatmaps de profils (moyennes par individu ou proportion de matches) ;
  - dendrogrammes sur les prototypes ou médoïdes ;
  - factor map (PCA) pour k-means lorsque le nombre de variables est raisonnable.

Chapitre 7

Étude de cas : `metal_universe`

7.1 Objectif de l'étude

L'étude de cas `metal_universe` a un double objectif :

— illustrer l'usage de `mmrClustVar` sur un jeu de données mixte comprenant à la fois des informations catégorielles (sous-genre, pays, genre du lead singer) et des scores numériques (bpm préférés, agressivité, etc.) ;

— tester la capacité des algorithmes à :

  — regrouper des variables fortement liées par construction (variables dérivées) ;

  — distinguer des blocs thématiques (par exemple, "rythmique", "violence du son", "profil socio-géographique") ;

  — identifier des variables bruitées.

7.2 Sélection des variables actives et supplémentaires

Dans une configuration typique, on peut choisir :

— **variables actives** :

  — variables numériques liées à l'esthétique musicale :

    — `num_bpm_pref`, `num_speed_score`, `num_tempo_variability`,

    — `num_distortion_preference`, `num_heaviness_score`,

    — `num_aggressivity_score`, `num_growl_tolerance`, etc. ;

  — variables catégorielles :

    — `subgenre`, `country`, `cat_front_gender`.

— **variables supplémentaires** :

  — variables de bruit explicite ;

  — dérivées secondaires, si l'on souhaite voir comment elles se positionnent a posteriori.

7.3 Choix de la méthode : k-prototypes

Comme les données sont mixtes, la méthode naturelle est `"kprototypes"` :

— la partie numérique capture les tendances communes (par exemple, groupes très rapides et agressifs vs plus lents et atmosphériques) ;

— la partie catégorielle capture des profils de sous-genres et de pays.

Le paramètre λ permet d'ajuster le poids donné au bloc catégoriel. Dans nos essais, une valeur $\lambda = 1$ donne déjà des partitions interprétables ; des valeurs plus élevées rendent les clusters plus sensibles aux modalités.

7.4 Exemple de résultat (K = 5)

Pour illustrer, voici un exemple de résumé obtenu pour $K = 5$ (configuration typique sur l'une de nos versions de `metal_universe`) :

— inertie intra-classe totale : environ 1.63 ;

— adhésion globale moyenne : 0.875 ;

— adhésion moyenne numérique : ≈ 0.82 ;

— adhésion moyenne catégorielle : 1.0 (variables catégorielles parfaitement alignées sur leur prototype dans cet exemple).

Les clusters récupérés peuvent s'interpréter comme :

— un bloc fortement corrélé autour de la distorsion et d'une certaine agressivité ;

— un bloc plutôt rythmique (bpm, speed score, tempo variability) ;

— un bloc mélangeant certaines caractéristiques géographiques et de style ;

— etc.

L'intérêt principal est de constater que les variables artificiellement corrélées (dérivées) sont généralement regroupées dans le même cluster, tandis que les variables de bruit sont soit mal attachées (faible adhésion) soit diffusées.

7.5 Comparaison avec k-medoids

Sur le même jeu `metal_universe`, l'utilisation de `"kmedoids"` permet de travailler directement sur une matrice de dissimilarités entre variables. On observe généralement :

— des médoïdes qui sont des variables très représentatives de leur groupe (par exemple, `num_bpm_pref` comme médoïde du cluster rythmique) ;

— une partition globalement cohérente avec celle de k-prototypes, mais parfois plus robuste aux variables aberrantes.

Ce type de comparaison permet de mieux comprendre les forces et faiblesses de chaque algorithme sur un même terrain de jeu.

Chapitre 8

Limites actuelles et perspectives d'évolution

8.1 Limites identifiées

Parmi les limites actuelles du package `mmrClustVar`, on peut citer :

- la dépendance à des composantes principales locales (ACP) pour la partie numérique :
  - ce choix est cohérent avec [1] mais suppose une certaine linéarité des structures ;
- la pondération fixe λ pour le bloc catégoriel :
  - elle est à choisir par l'utilisateur, sans mécanisme automatique ;
- la complexité de k-medoids pour des tableaux très larges en nombre de variables :
  - la construction de la matrice de dissimilarités est en $O(p^2)$, ce qui peut devenir coûteux.

8.2 Perspectives

Plusieurs pistes d'amélioration sont envisageables :

- sélection automatique ou guidée de λ (par exemple, via une égalisation des variances contribution numérique / catégorielle) ;
- stratégies d'initialisation plus sophistiquées pour k-means / k-modes / k-prototypes (inspiration k-means++ [1]) ;
- intégration d'autres mesures de similarité pour les blocs numériques (corrélations non linéaires, par exemple) ;
- extension à des méthodes hiérarchiques de clustering de variables ;
- amélioration de l'interface Shiny (sauvegarde de sessions, thèmes, etc.).

1. Non implémenté ici, mais potentiellement intéressant.

Conclusion

Le projet `mmrClustVar` nous a permis de mettre en pratique, de manière cohérente, plusieurs éléments centraux du Master 2 SISE : modélisation statistique, programmation R avancée, architecture logicielle et visualisation interactive.

Sur le plan méthodologique, nous avons implémenté et unifié quatre algorithmes de clustering de variables (k-means, k-modes, k-prototypes, k-medoids), couvrant ainsi les cas de données numériques, qualitatives et mixtes. Les mesures de similarité (corrélation au carré, simple matching) et d'inertie expliquée fournissent des indicateurs interprétables pour l'utilisateur.

Sur le plan logiciel, la structure en classes R6 (base + engines + façade) offre un socle robuste pour l'évolution future du package. L'application Shiny vient compléter l'ensemble en proposant un accès interactif aux fonctionnalités de `mmrClustVar` : chargement de données, paramétrage, visualisation, export.

Enfin, le jeu de données fictif `metal_universe`, spécialement construit pour ce projet, joue le rôle de laboratoire contrôlé pour tester et illustrer les comportements des différents algorithmes sur des structures de corrélation et de redondance connues.

Ce rapport long constitue notre documentation personnelle exhaustive. Il pourra servir de référence pour de futures évolutions du package ou comme base pour d'autres projets de classification de variables.

Bibliographie

[1] Chavent, M., Kuentz-Simonet, V., Labenne, A., & Saracco, J. (2012). ClustOfVar : An R Package for the Clustering of Variables. *Journal of Statistical Software*, 50(13), 1–16.
https://arxiv.org/pdf/1112.0295

[2] Husson, F., Josse, J., & Pagès, J. (2017). *Exploratory Multivariate Analysis by Example using R* (2e éd.).
Chapman and Hall/CRC.
http://staff.ustc.edu.cn/~ynyang/vector/books/Husson-Le-Pages.pdf

[3] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297).
https://www.cs.cmu.edu/~bhiksha/courses/mlsp.fall2010/class14/macqueen.pdf

[4] Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2, 283–304.
https://cse.hkust.edu.hk/~qyang/Teaching/537/Papers/huang98extensions.pdf

[5] Kaufman, L., & Rousseeuw, P. J. (2005). *Finding Groups in Data : An Introduction to Cluster Analysis*.
Wiley.
https://www.researchgate.net/publication/220695963_Finding_Groups_in_Data

[6] Rakotomalala, R. (2025). R programming & classification lectures. Université Lyon 2.
https://tutoriels-data-science.blogspot.com/

[7] R Core Team. (2025). *R : A Language and Environment for Statistical Computing*.
R Foundation for Statistical Computing.
https://www.r-project.org/

[8] Chang, W. (2025). *R6 : Encapsulated Object-Oriented Programming for R*.
R package documentation.
https://r6.r-lib.org