



# R- Package Clustering de Variables

Maissa Lajimi  
Yassine Cheniour  
Lamia Hatem

Lien du dépôt :

[https://github.com/maissaladjimi/SISE\\_Clustering\\_Variables\\_R/](https://github.com/maissaladjimi/SISE_Clustering_Variables_R/)

# SOMMAIRE

- 01 ARCHITECTURE DU PACKAGE
- 02 K-MEANS → VAR QUANTI
- 03 VARCLUS → VAR QUANTI
- 04 CAH (DICE/ACM) → VAR QUALI
- 05 APPLICATION SHINY



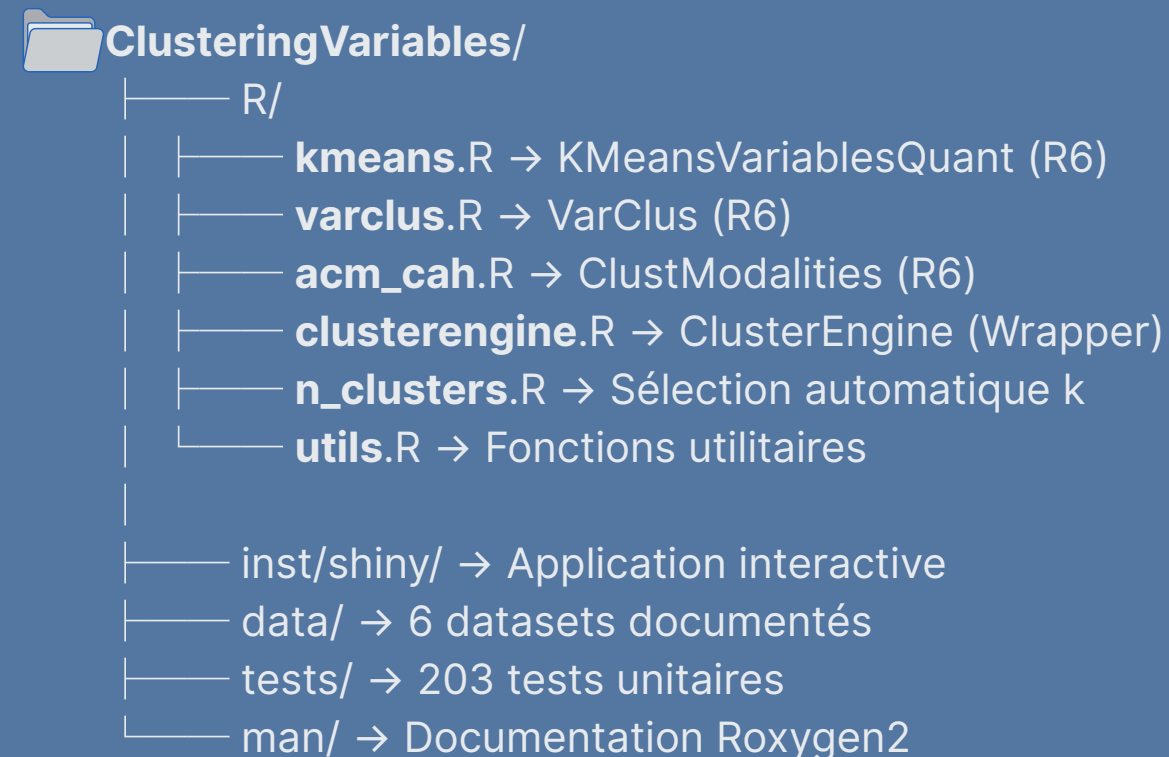
# ARCHITECTURE DU PACKAGE



3 algorithmes = 3 classes indépendantes :

→ Indépendance des développements

→ Faciliter maintenance et correction



## Classes R6 :

- **Constructeur** : `km <- KMeansVariablesQuant$new(k = 3)`
- **Entrainement** : `km$fit(data)`
- **Infos succinctes** : `km$print()`
- **Résultats** : `km$summary()`
- **Visualisation et Interprétation** : `km$plot_correlation_circle()`

# ARCHITECTURE : CLUSTER\_ENGINE

## ClusterEngine

- **data** : Données
- **method** : "kmeans" / "varclus" / ...
- **n\_clusters** : Nombre de clusters
- **model** : Instance de la classe R6

### Méthodes :

- **fit()** → Crée et ajuste le modèle
- **predict()** → Délègue à model\$predict()
- **summary()** → Délègue à model\$summary()
- **elbow()** → Sélection auto k

### Fonctionnement du Wrapper :

- Création de l'instance et délégation selon la méthode souhaitée
- Le constructeur de clusterengine créé en réalité un objet d'une des 3 classes R6 et appelle ses méthodes

### Améliorations :

- L'app Shiny n'utilise pas clusterengine, appelle directement les classes (débuggage)
- elbow ne fonctionne pas très bien pour CAH

Exemples →

```
library(ClusteringVariables)

data(crime)

# quasi-unifiée pour les 3 méthodes
engine <- ClusterEngine$new(crime[,-1], "kmeans", n_clusters=3)

engine$fit() # Délègue vers KMeansVariablesQuant

engine$summary() # Appelle model$summary()

engine$predict(new_vars)
```

# ARCHITECTURE : N\_CLUSTERS

Contient des fonctions de détection du nombre de k optimal

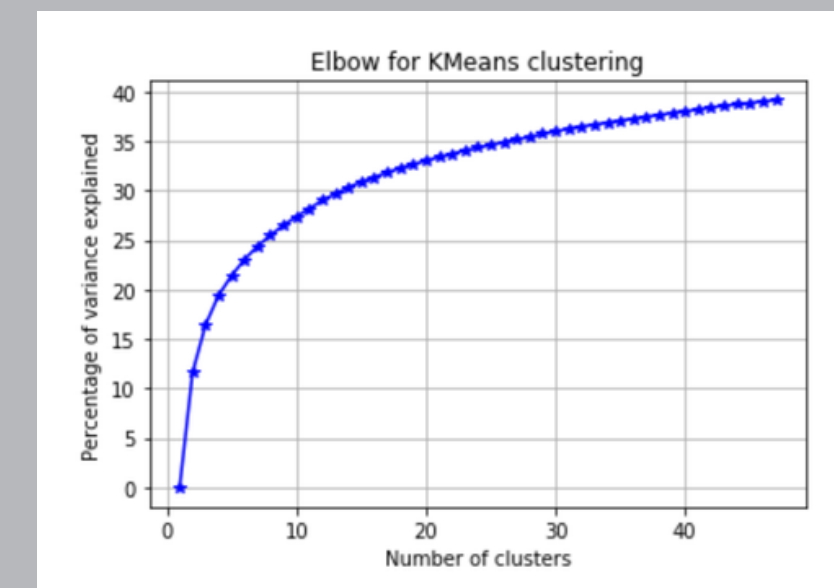
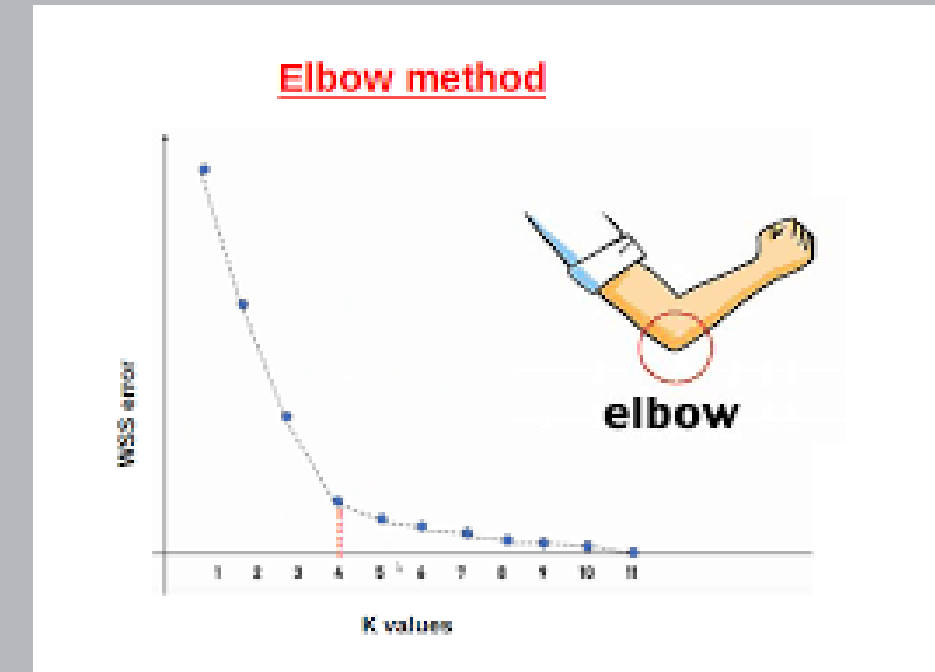
2 approches selon l'algorithme :

→ Kmeans et Varclus : **distance perpendiculaire** (droite de min à max : on cherche le point le plus éloigné de la droite, c'est celui qui indique le meilleur k.)

→ CAH : A partir du dendrogramme, on prend le plus grand saut entre deux hauteurs, c'est ce qu'on appelle le **gain marginal** (= hauteur(k-1)- hauteur(k), k optimal est le plus grand saut)

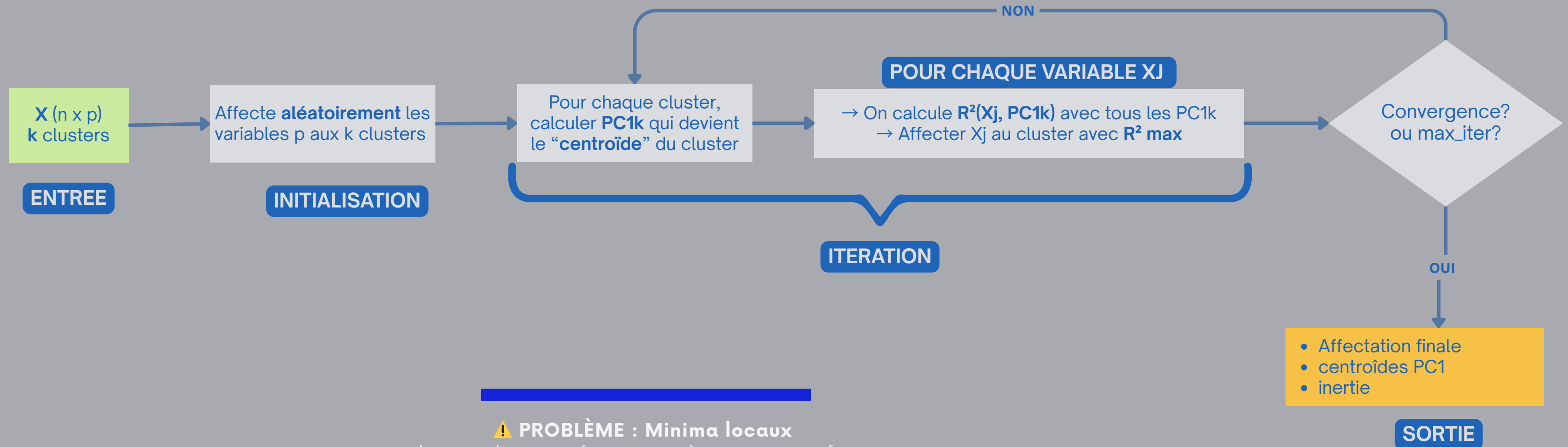
Le k optimal proposé, n'est pas toujours le plus optimal

→ Il est plus fiable de regarder l'elbow plot



# K-MEANS : PRINCIPE

- Regroupe  $p$  variables autour de  $k$  composantes
- Composante = PC1
- Distance =  $1 - R^2$  (corrélation au carré)
- Objectif : maximiser l'inertie totale



## ⚠ PROBLÈME : Minima locaux

SOLUTION : Multi-initialisations ( $n\_init = 10$ ) → Lancer 10 fois avec affectations initiales différentes → Garder la solution avec inertie maximale

# K-MEANS : CLASSE R6

## ATTRIBUTS : 3 TYPES

### PARAMETRES ALGO :

- **k** : Nombre de cluster ( $\geq 2$ )
- **max\_iter** : Max itérations (défaut:100)
- **n\_init** : Nb initialisations (défaut: 10)
- **tol** : Tolérance (défaut:  $1e-4$ )
- **seed** : Graine aléatoire (optionnel)

### DONNÉES :

- **data** : data.frame original
- **X\_scaled** : Matrice centrée-réduite ( $n \times p$ )
- **scale\_center** : Paramètres centrage (predict)
- **scale\_scale** : Paramètres échelle (predict)

### RÉSULTATS :

- **clusters** : Vecteur affectation (long. p)
- **centers** : Matrice centroïdes PC1 ( $n \times k$ )
- **r2\_matrix** : Matrice  $R^2$  ( $p \times k$ )
- **inertia\_total** : Inertie totale  $\sum \lambda_k$
- **inertia\_by\_cluster** : Inertie par cluster
- **n\_iter** : Nb itérations convergence

## METHODES PRINCIPALES

### OBLIGATOIRES :

- **\$new**(k, max\_iter, n\_init, tol, seed)
  - → Constructeur avec validation paramètres
- **\$fit**(X, k=NULL)
  - → Ajustement : centrage-réduction + clustering
  - → Lance n\_init runs, garde meilleure inertie
- **\$print**()
  - → Affichage succinct (k, n, p, inertie, tailles)
- **\$summary**(print\_output=TRUE)
  - → Statistiques détaillées (qualité,  $R^2$ , corrél.)
- **\$predict**(new\_data)
  - → Affecte nouvelle variable au cluster optimal
  - → Retourne  $R^2$  avec chaque centroïde

### AUTRES METHODES :

- **\$illustrative**(X\_illust, plot=TRUE)
  - → Analyse complète variables illustratives
  - → Régression sur PC1, visualisation ACP
- **\$plot\_elbow**(k\_range=2:10)
  - → Sélection automatique k optimal
- **\$plot\_correlation\_circle**(dims=c(1,2))
  - → Cercle des corrélations (ACP des variables)
- **\$plot\_biplot**(dims=c(1,2))
  - → Biplot variables dans l'espace factoriel
- **\$get\_clusters\_table**() → DataFrame avec affectations triées

## METHODES PRIVEES

### private\$latent\_center(X\_cluster)

- → Calcule PC1 d'un cluster par ACP
- → Gère cas particuliers : cluster vide, 1 variable, variance nulle
- → Retourne centroïde normalisé (mean=0, sd=1)

### private\$single\_run(X, K, max\_iter, tol)

- → Une exécution complète de l'algorithme
- → Init aléatoire → Boucle itérative → Convergence →
- Retourne : clusters, centers, r2\_matrix, inertia

### private\$detect\_elbow(k\_vals, y\_vals)

- → Détection k optimal par distance perpendiculaire
- → Utilisé dans \$plot\_elbow()

# K-MEANS : INTERPRÉTATION

## Métriques

- **Print** (Nb\_clusters (k), max\_iter, Nb\_init, Convergence tolerance, Inertie totale)
- **Summary**
  - **Global quality**
    - Inertia\_total : Variance totale capturée par toutes les PC1
    - avg\_inertia : Inertie moyenne par cluster
    - pct\_explained : % de l'inertie maximale atteinte
  - **Cluster Summary**
    - members : nb de variables
    - Inertia (variation explained) : Valeur propre de la PC1 du cluster
    - proportion explained(  $\lambda_1$ /members) : proportion moyenne de variance expliquée par PC1
  - **Cor\_latent** : correlation entre PC1 de chaque cluster
  - **R<sup>2</sup> matrix** : R<sup>2</sup> entre chaque variable et chaque PC1
- **Cluster table**
  - Table d'affectations

## Visualisations

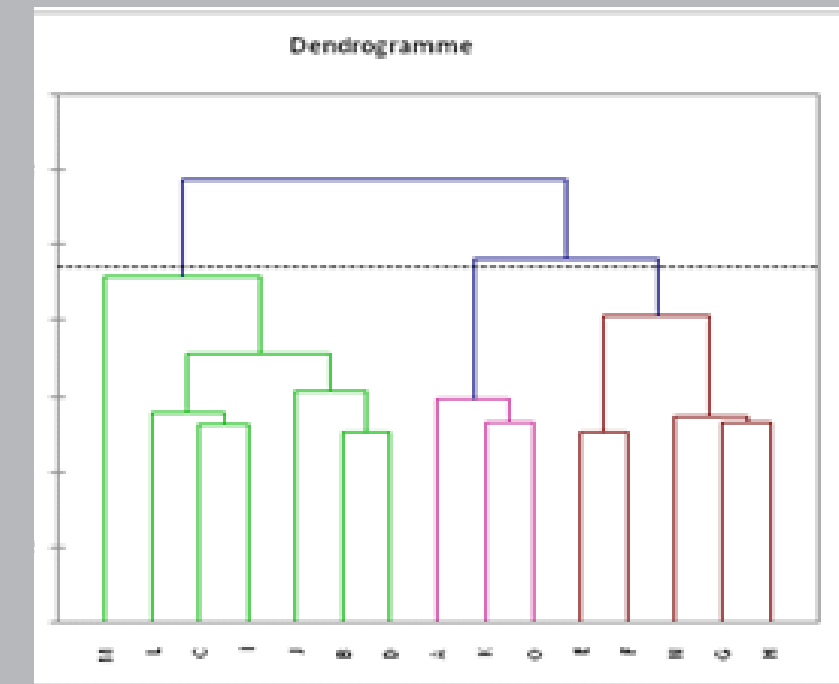
- **Cercle de corrélation**
  - Variables projetées sur les 2 premiers axes factoriels d'une ACP globale
- **Biplot**
  - Projection variables dans espace factoriel
- **Elbow plot**
  - Inertie totale en fonction de k
  - k optimal → calcul avec distance perpendiculaire



# CAH- PRINCIPE

## Contexte :

- > On ne clusterise pas directement les variables mais leurs **modalités**
- > 2 méthodes proposés : **DICE** ou **ACM**
- > Résultat final : **Groupes** de MODALITÉS  
→ Variables regroupées par associations de modalités similaires



## DICE

- > **Table disjonctive** complète (Matrice indicatrice 0/1)
- > Calcule **distance DICE**<sup>2</sup> entre chaque paire de modalités
  - $d^2_{Dice}(mj, mj') = 1/2 \sum_i (x_{ij} - x_{ij'})^2$
- > **CAH** avec critère de ward (ward.D2) pour agréger modalités
  - Agrégation **ascendante** (bottom-up)
  - **Minimise** variance intra-cluster

## ACM

- > **Table disjonctive** complète (Matrice indicatrice 0/1)
- > **Projection** des modalités dans un espace réduit (avec ACM)
- > **Pondération** des coordonnées factorielles par la racine des VP
  - donnent plus de poids aux axes plus informatifs
- > **CAH** sur les coordonnées pondérées avec ward.D
- > **Correction** des VP pour interprétation (Greenacre et Benzécri)
  - Corriger la redondance artificielle

# CAH: CLASSE R6

## ATTRIBUTS : 4 TYPES

### PARAMETRES ALGO :

- method : "dice" ou "acm"
- n\_axes : Nb axes ACM (NULL = auto)

### DONNÉES :

- data : data.frame variables quali
- disj : Table disjonctive complète

### RÉSULTATS ACM :

- acm : Objet ade4::dudi.coa()
- eig\_raw : Valeurs propres brutes
- eig\_benzecri : VP corrigées (Benzecri)
- eig\_greenacre : VP corrigées (Greenacre)
- mod\_coords : Coord. modalités (pondérées)
- ind\_coords : Coord. individus

### RÉSULTATS CLUSTERING :

- dist\_mat : Matrice distances modalités
- hclust : Objet hclust (dendrogramme)
- k : Nombre de clusters
- mod\_clusters : Affectation modalités

## METHODES PRINCIPALES

### OBLIGATOIRES :

- **\$new**(method, n\_axes)
  - → Constructeur avec choix méthode
- **\$fit**(X, k=NULL)
  - → DICE : TDC → distance Dice<sup>2</sup> → CAH (ward.D2)
  - → ACM : TDC → ACM → coord pondérées → CAH (ward.D) + corrections VP
- **\$print**()
  - → Affichage succinct (méthode, nb modalités, k)
- **\$summary**(print\_output=TRUE)
  - → Statistiques détaillées :
    - Basic stats (nb vars, nb modalités, k)
    - ACM stats (VP brutes/corrigées, % variance)
    - Cluster stats (tailles, composition)
- **\$predict**(new\_data)
  - → Affecte nouvelles observations
  - → Retourne cluster + distances

## AUTRES METHODES

- **\$illustrative**(X\_illust\_quali)
  - → Projette modalités illustratives qualitatives
  - → Affichage table croisée
- **\$illustrative\_numeric**(X\_illust\_quant) [ACM only]
  - → Cercle des corrélations variables quanti
  - → Interprétation axes factoriels
- **\$plot\_dendrogram**(k=NULL)
  - → Dendrogramme coloré par cluster
- **\$plot\_factorial\_map**(dims=c(1,2), k=NULL) [ACM only]
  - → Carte factorielle modalités colorées
- **\$plot\_screes**(cumulative=FALSE) [ACM only]
  - → Éboulis valeurs propres (brutes/corrigées)
- **\$plot\_contrib**(dim, top=10) [ACM only]
  - → Contributions modalités aux axes
- **\$elbow**(k\_max=10, plot=TRUE)
  - → Sélection k optimal (gain marginal max)
- **\$cluster\_table**()
  - → Table croisée variables × clusters
- **\$get\_clusters\_table**()
  - → DataFrame (modalité, cluster)

### PRIVEES :

- private**\$compute\_dice\_distances**(disj\_mat)
  - → Calcule matrice Dice<sup>2</sup> :  $d^2(j,j') = \frac{1}{2} \sum_i (x_{ij} - x_{ij'})^2$
  - → Retourne objet dist()
- private**\$check\_data**(X)
  - → Valide et convertit en facteurs
  - → Gère colonnes non-facteurs

# CAH : INTERPRÉTATION

## Métriques

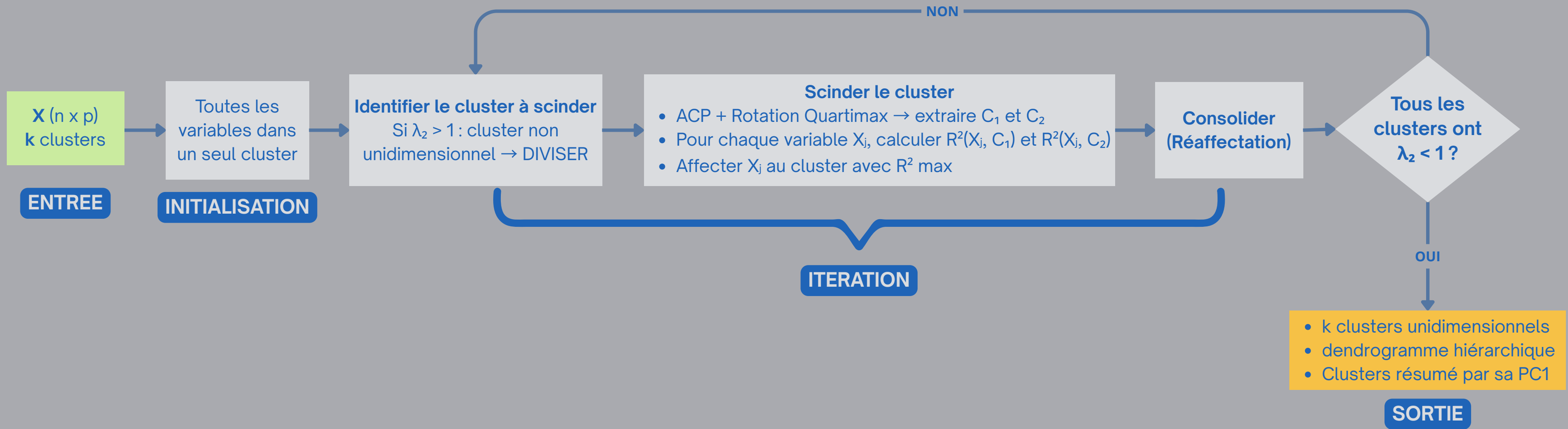
- **Print** (method : Dice/acm, nb var, nb mod, nb clusters(k))
- **Summary**
  - **Basic stats** (nb\_obs, nb\_var, nb\_mod, k)
  - **ACM Stats** :
    - Axe
    - Valeurs Propres brutes
    - Valeurs Propres corrigées Benzécri
    - Valeurs Propres corrigées Greenace
    - % variance expliquée
    - % variance cumul
- **Cluster table**
  - Table d'affectations

## Visualisations

- **Dendrogramme**
  - Variables projetées sur les 2 premiers axes factoriels d'une ACP globale
  - Distance : 1 -Dice<sup>2</sup> ou euclidienne (ACM)
  - linkage : ward.D2
- **Carte factorielle**
  - Modalités projetées sur axes 1-2 de l'ACM
- **Elbow**
  - Inertie totale en fonction de k
  - k optimal → Gain marginal
- **Contributions (ACM)** :
  - Barplot op N modalités contribuant le plus à un axe

# VARCLUS: PRINCIPE

- Clustering hiérarchique divisif de variables
- Mesure = Corrélation
- Objectif : clusters unidimensionnels ( $\lambda_2 < 1$ )
- Utilise `Hmisc::varclus()`



# VARCLUS : CLASSE R6

## ATTRIBUTS : 2 TYPES

### PARAMETRES ALGO :

- similarity
- n\_clusters

: "pearson" ou "spearman"

: Nb clusters (null= auto)

### DONNÉES :

- data
- model
- clusters
- dendo

: Matrice numérique (n x p)

: Objet Hmisc::varclus()

: data.frame (variable, cluster)

: Objet dendrogramme

Résultats (calculés à la volée) :

- PC1 par cluster
- Valeurs propres  $\lambda_1$  par cluster
- Proportion variance expliquée
- $R^2$  own/next par variable

## METHODES PRINCIPALES

### OBLIGATOIRES :

- **\$new**(similarity, n\_clusters)
  - → Constructeur avec mesure similarité
- **\$fit**(X)
  - → Appelle Hmisc::varclus()
  - → Coupe dendrogramme si n\_clusters spécifié
  - → Sinon, utilise critère  $\lambda_2 \geq 1$  automatique
- **\$print**()
  - → Affichage succinct (nb vars, clusters,  $\lambda$ )
- **\$summary**(print\_output=TRUE)
  - → Statistiques détaillées par cluster
  - →  $R^2$  own/next/ratio pour chaque variable
- **\$predict**(new\_var)
  - → Affecte variable au cluster avec  $R^2$  max

### AUTRES METHODES :

- **\$illustrative**(X\_illust, plot=TRUE)
  - → Régression sur PC1 des clusters
  - → Visualisation ACP
- **\$get\_dendrogram**
  - → Retourne fonction de plot du dendrogramme
- **\$get\_heatmap**()
  - → Heatmap matrice similarité (corrélations)
- **\$plot\_elbow**(k\_range)
  - → Aide choix k optimal (coude similarité)
- **\$get\_clusters\_table**()
  - → DataFrame avec affectations triées

## METHODES PRIVEES

### private\$compute\_cluster\_pcs\_list()

- → Calcule PC1 de chaque cluster par ACP
- → Gère cluster à 1 variable (PC1 = variable normalisée)
- → Retourne liste de vecteurs PC1

### private\$compute\_cluster\_pcs()

- → Calcule statistiques résumées par cluster :
- Nb variables
- $\lambda_1$  (eigenvalue)
- Proportion variance expliquée par PC1

### private\$compute\_cluster\_R2()

- → Pour chaque variable :
  - $R^2_{\text{own}}$  :  $R^2$  avec PC1 de son cluster
  - $R^2_{\text{next}}$  :  $R^2$  max avec PC1 autres clusters
  - Ratio :  $(1 - R^2_{\text{own}}) / (1 - R^2_{\text{next}})$
- → Ratio < 1 = bonne affectation

# VARCLUS : INTÉRPRÉTATION

## Métriques

- **Cluster Summary**

- Nombre de variables dans chaque cluster
- $\lambda_1$  (variance de PC1)
- % de variance expliquée par PC1

- **R<sup>2</sup> Details**

- Own\_Cluster: Corrélation<sup>2</sup> avec PC1 de son propre cluster
- Next\_Cluster: R<sup>2</sup> maximum avec la PC1 du cluster voisin le plus proche
- 1-R<sup>2</sup>\_Ratio: Ratio de qualité d'affectation

- **Cluster Quality**

- mean\_R2\_own: Moyenne des R<sup>2</sup> dans le cluster

## Visualisations

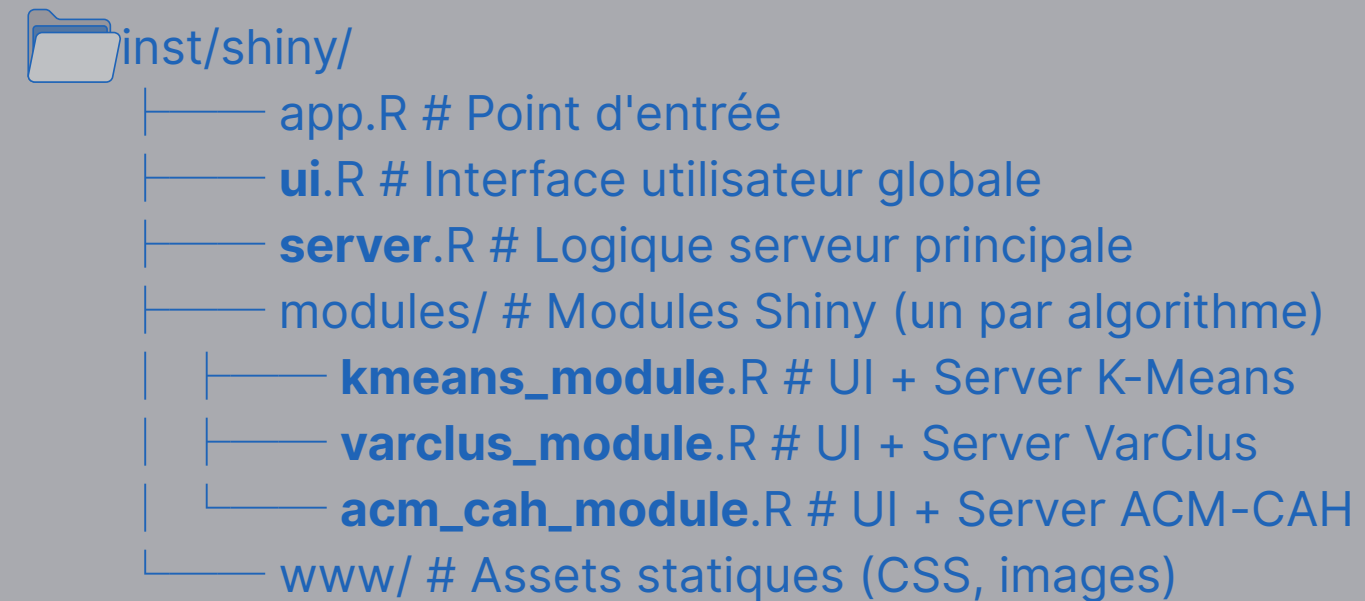
- **Dendogramme**

- **Heatmap de corrélation**

- **Elbow plot**

- Sélection automatique du nombre k optimal
- $S_k$  = similarité intra-cluster moyenne
- k optimal = max distance à la droite

# APPLICATION SHINY



## ARCHITECTURE MODULAIRE

- **Indépendances** entre les algorithmes
- Facilite la **maintenance** et le débogage
- **Scalabilité** (ajouter un algorithme)
- N'utilise pas clusterEngine
  - **Instancie** directement les classes

## FONCTIONNEMENT

- Choix algo puis "**Run Clustering**"
  - **eventReactive** instancie la classe choisie
  - Affichage conditionnel
- Un **algorithme** sélectionné=une **instance** R6 créé= un **module** actif= une **UI** affichée

## STRUCTURE D'UN MODULE

- Chaque module a une **ui** et un **server**
- **UI** → 4 tabs:
    - Summary
    - Visualizations
    - Illustrative Variables
    - Detailed Stats
  - **Server** : Logique server de la classe concerné



# BILAN ET CONCLUSION



## RÉALISÉ

- 3 **algorithmes** complets :
  - K-means (Var. Quantitatives)
  - Varclus Hiérarchique (Var. Quantitatives)
  - DICE/ACM + CAH (Var Qualitatives)
- **Package** R :
  - Architecture R6
  - Documentation
  - Tests et datasets inclus
- Application **Shiny** interactive :
  - Architecture modulaire
  - Sélection k optimal



## AMÉLIORATIONS

- **Consolidation** et **fiabilisation**
  - Améliorer la robustesse
  - Améliorer la flexibilité
  - Confirmer la justesse des algos
  - Corriger si besoin
- **Enrichissements** :
  - Ajout d'autres algorithmes
  - Gestion des variables mixtes
  - Ajout de visualisations
  - Ajout d'outils d'interprétation



# PASSONS À LA SUITE!

