Clustering K-prototypes

Équipe 19 – ANACLET Kylian, BIN DAIVIN Muhamad Zaiinizee, CHENNOUF Younes, KEBAIRI Ilyes

2025-12-10

Le jeu de données contient à la fois des variables numériques et catégorielles (région, condition d'éclairage, condition atmosphérique, etc.). L'algorithme K-means classique n'est donc pas adapté, car il repose sur la distance euclidienne, qui n'a pas de sens pour les variables catégorielles, même après encodage one-hot.

Il est préférable d'utiliser **K-prototypes**, qui combine la distance euclidienne pour les variables numériques et une mesure de dissimilarité spécifique pour les variables catégorielles. Cet algorithme permet ainsi d'obtenir des clusters plus fiables et cohérents.

K-prototypes

L'algorithme K-prototypes implémente la méthode de partitionnement par minimisation de la distance entre les observations et les centres de clusters. Plus précisément, il cherche à minimiser la fonction suivante :

$$\min_{c_1,\dots,c_k} \sum_{k=1}^{K} \sum_{i,G(i)=k} d(x^{(i)}, c_k)$$

où $d(x^{(i)}, c_k)$ représente la distance entre l'observation $x^{(i)}$ et le centre du cluster c_k. Cette distance combine la distance euclidienne pour les variables numériques et une mesure de dissimilarité pour les variables catégorielles.

La minimisation est réalisée automatiquement par la fonction `kproto()` à travers un processus itératif. L'algorithme affecte chaque observation au cluster dont le centre est le plus proche, puis recalcule les centres, et répète ces étapes jusqu'à convergence. Le résultat de cette minimisation est observable via `tot.withinss`, qui représente la somme totale des distances intra-clusters.

A. Chargement et préparation des données

Le jeu de données final étant très volumineux, un échantillon aléatoire a été extrait pour faciliter le traitement et le clustering.

```
n_sample <- 1000
df <- read.csv('accidents_db_final_clusturing.csv')

# On enlève les variables non utiles pour le clustering
df <- subset(df, select = -c(Region, Moment, nb_tue, nb_hospitalise, nb_blesse, nb_indemne))

# On tire par hasard 1000 accidents
df <- df[sample(nrow(df), n_sample), ]
```

On n'inclut pas ces variables :

- **Region :** Son inclusion forcerait les clusters à se former principalement par région plutôt que par les caractéristiques réelles des accidents.
- **Moment :** Cette variable est redondante avec `lum` (conditions d'éclairage).
- **nb_tue, nb_hospitalise, nb_blesse, nb_indemne :** On les exclut du clustering car ce sont des résultats (gravité de l'accident).

Identifier les variables de type character et les convertir en facteurs.

```
#install.packages("dplyr")
library(dplyr)
```

```
##
## Attachement du package : 'dplyr'

## Les objets suivants sont masqués depuis 'package:stats':
##
##     filter, lag

## Les objets suivants sont masqués depuis 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df <- df %>% mutate(across(where(is.character), as.factor))
```

Avant d'appliquer l'algorithme K-prototypes, il est nécessaire de :

- Vérifier que toutes les variables catégorielles sont bien des facteurs.
- Vérifier que toutes les variable numériques sont bien des numériques.
- Traiter les valeurs manquantes (NA) si nécessaire.

```
str(df)
```

```
## 'data.frame':    1000 obs. of  19 variables:
##  $ lum          : Factor w/ 5 levels "crepuscule ou aube",..: 2 4 4 2 2 2 5 5 2 5 ...
##  $ agg          : Factor w/ 2 levels "en agglo","hors agglo": 1 2 2 2 1 1 1 2 1 2 ...
##  $ int          : Factor w/ 9 levels "4+ branches",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ atm          : Factor w/ 9 levels "autre","brouillard ou fumee",..: 6 6 6 6 6 6 6 8 8 6 ...
##  $ col          : Factor w/ 7 levels "3+ veh - en chaine",..: 4 4 7 7 4 3 3 3 4 6 ...
##  $ catr         : Factor w/ 8 levels "autoroute","autre",..: 3 7 4 4 3 4 3 7 3 4 ...
##  $ circ         : Factor w/ 4 levels "avec voies d'affectation variable",..: 2 3 2 2 2 2 2 2 4 2 ...
##  $ prof         : Factor w/ 4 levels "base de cote",..: 3 3 1 3 3 3 2 2 3 3 ...
##  $ plan         : Factor w/ 4 levels "en courbe a droite",..: 4 4 2 2 4 4 4 4 4 4 ...
##  $ surf         : Factor w/ 7 levels "autre","corps gras",..: 6 6 6 6 6 6 6 5 5 6 ...
##  $ infra        : Factor w/ 10 levels "aucun","autres",..: 1 1 1 1 1 1 1 7 1 1 ...
##  $ situ         : Factor w/ 7 levels "accotement","autres",..: 4 4 2 4 4 6 4 4 4 4 ...
##  $ vma          : int  30 110 80 70 50 50 50 90 50 80 ...
```

```
##  $ nb_velo      : int  0 0 0 0 1 0 0 0 0 0 ...
##  $ nb_moto      : int  2 0 0 0 0 0 1 0 0 0 ...
##  $ nb_voiture   : int  0 2 1 1 0 1 0 1 1 0 ...
##  $ nb_utilitaire: int  0 0 0 0 1 0 0 0 0 1 ...
##  $ nb_camion    : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ nb_transport : int  0 0 0 0 0 0 0 0 0 0 ...
```

```r
if(anyNA(df)) {
  df <- na.omit(df)
}
```

B. Normalisation des variables numériques

Les variables numériques sont standardisées pour garantir que toutes contribuent équitablement au calcul des distances, indépendamment de leur échelle de mesure. Cela évite qu'une variable avec de grandes valeurs domine artificiellement la formation des clusters.

```r
numeric_cols <- sapply(df, is.numeric)
df[numeric_cols] <- scale(df[numeric_cols])
```

Remarque : les variables catégorielles (facteurs) ne sont pas standardisées.

C. Choix du nombre de clusters (k)

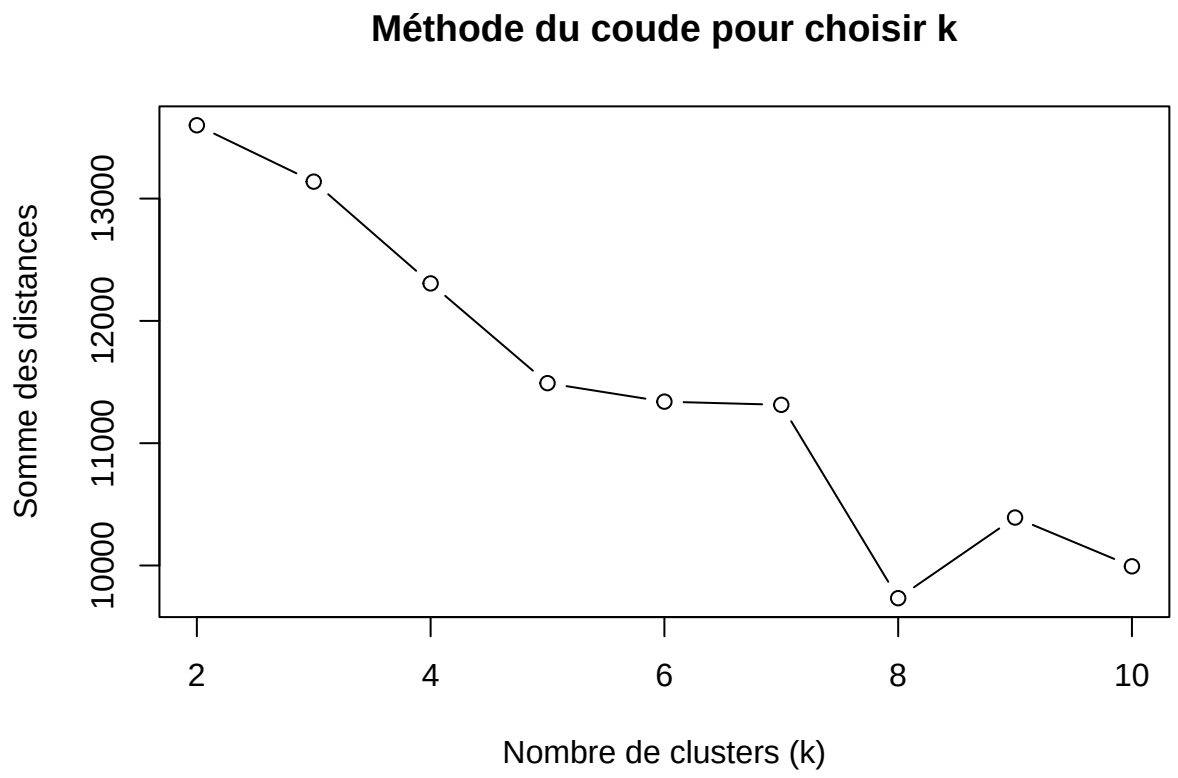
On applique l'algorithme k-prototypes pour différentes valeurs de k.

```r
set.seed(1)
#install.packages("clustMixType")
#install.packages("cluster")
library(clustMixType)
library(cluster)

# Pour la méthode du coude
k_max <- 10
cost <- numeric(k_max)

# Pour la méthode de silhouette
d <- daisy(df, metric = "gower")
```

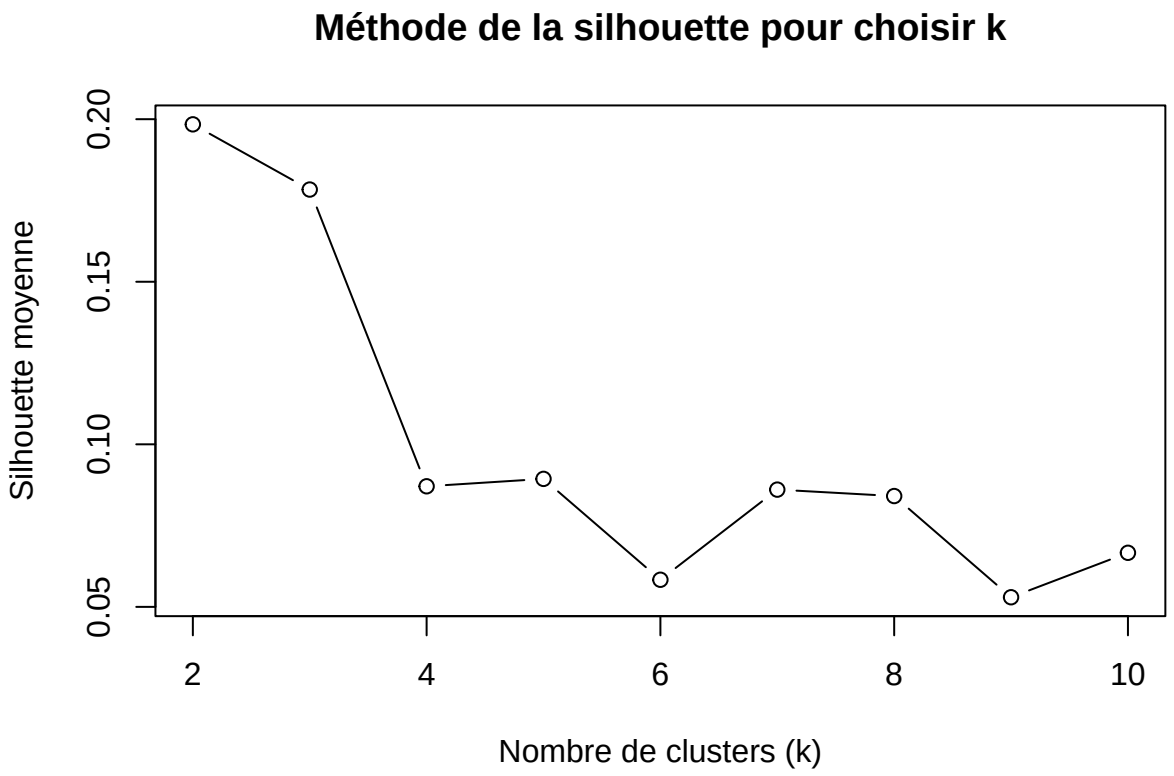
```
## Warning in daisy(df, metric = "gower"): la ou les variables binaires 19 sont
## traitées comme des intervalles standardisés
```

```r
sil_avg <- numeric()


for (k in 2:k_max) {
  kp <- kproto(df, k, verbose = FALSE)
  cost[k] <- kp$tot.withinss

  sil <- silhouette(kp$cluster, d)
```

```r
  sil_avg[k] <- mean(sil[, 3])
}

# Plot méthode du coude
plot(2:10, cost[2:k],
     main = "Méthode du coude pour choisir k",
     type = "b", xlab = "Nombre de clusters (k)",
     ylab = "Somme des distances")
```




```r
# Plot méthode de la silhouette
plot(2:10, sil_avg[2:10],
     main = "Méthode de la silhouette pour choisir k",
     type = "b", xlab = "Nombre de clusters (k)",
     ylab = "Silhouette moyenne")
```



Observations

Le graphique de la méthode du coude montre que la courbe se stabilise à k = 6, tandis que la silhouette présente deux maxima à k = 2 et k = 4.

Bien que le pic le plus élevé de la silhouette corresponde à k = 2, nous avons choisi k = 4 car il est cohérent avec le coude et permet une segmentation plus détaillée des observations.

Remarque : Ces résultats dépendent fortement de l'échantillon tiré. Avec un autre échantillon aléatoire, les courbes du coude et de la silhouette pourraient suggérer un nombre de clusters différent.

D. Application de k-prototypes et analyse des résultats

```
set.seed(1)
library(clustMixType)
res <- kproto(df, k = 4, diss = "gower", verbose = FALSE)
clusters <- res$cluster
```

```
table(clusters)
```

Distribution des observations par cluster

```
## clusters
##   1   2   3   4
## 288 322 224 166
```

L'analyse des clusters montre que les quatres groupes identifiés présentent des tailles différentes. Le cluster 3 regroupe la majorité des observations, tandis que le cluster 1 correspond a un petit sous-groupe spécifique. Les clusters 2 et 4 occupent des tailles intermédiaires.

Cette répartition révèle l'existence de groupes homogènes mais de taille variées, permettant d'identifier et d'interpéter les profils distincts propres à chaque cluster.

Remarque : Ces effectifs varient selon l'échantillon sélectionné.

Moyennes des variables numériques par cluster

```r
aggregate(. ~ clusters, data = df, FUN = mean)
```

```
##   clusters      lum      agg      int      atm      col     catr     circ
## 1        1 4.361111 1.177083 3.493056 5.972222 5.006944 3.513889 2.458333
## 2        2 3.922360 1.413043 2.903727 6.040373 4.145963 3.956522 2.295031
## 3        3 2.486607 1.156250 3.214286 6.053571 4.111607 3.540179 2.544643
## 4        4 3.993976 1.969880 2.234940 6.048193 3.921687 2.837349 3.042169
##       prof     plan     surf    infra     situ         vma    nb_velo
## 1 2.829861 3.611111 5.802083 1.562500 4.045139 -0.40736285  0.7079423
## 2 2.785714 3.496894 5.816770 1.590062 3.568323 -0.08837229 -0.2743745
## 3 2.852679 3.602679 5.736607 1.566964 3.977679 -0.42989151 -0.3045610
## 4 2.837349 3.469880 5.704819 1.638554 3.662651  1.45826550 -0.2850428
##      nb_moto nb_voiture nb_utilitaire  nb_camion nb_transport
## 1  0.5235263 -0.4733978     0.1786389 -0.1536029    0.2723218
## 2 -0.6343912  0.6825123    -0.2637297 -0.1589521   -0.1101526
## 3  0.5042820 -0.4624343    -0.2290713 -0.1174167   -0.1101526
## 4 -0.3581976  0.1214149     0.5107525  0.7332624   -0.1101526
```

Répartition des variables catégorielles par cluster

```r
cat_cols <- sapply(df, is.factor)

for (col in names(df)[cat_cols]) {
  cat("\nVariable :", col, "\n")
  print(table(df[[col]], clusters))
}
```

```
##
## Variable : lum
##                                clusters
##                                   1   2   3   4
##   crepuscule ou aube              27  34  32  23
##   nuit avec eclairage allume      19  50 134   6
##   nuit avec eclairage non allume   5   4   3   4
##   nuit sans eclairage              9  53  27  49
```

```
##   plein jour                    228 181  28  84
##
## Variable : agg
##             clusters
##                1   2   3   4
##   en agglo   237 189 189   5
##   hors agglo  51 133  35 161
##
## Variable : int
##                    clusters
##                       1   2   3   4
##   4+ branches         4   3   1   0
##   aucune            142 219 127 149
##   autre               8   8  16   7
##   en T               50  31  25   2
##   en X               52  39  35   6
##   en Y                7   5   6   0
##   giratoire          15  16  10   2
##   passage a niveau    3   0   0   0
##   place               7   1   4   0
##
## Variable : atm
##                         clusters
##                            1   2   3   4
##   autre                    2   1   3   3
##   brouillard ou fumee      0   3   4   6
##   couvert                  9  11   6   4
##   eblouissant              9   9   1   0
##   neige ou grele           0   1   0   0
##   normale                241 253 174 122
##   pluie forte              7   8   9   3
##   pluie legere            20  34  27  28
##   vent fort ou tempete     0   2   0   0
##
## Variable : col
##                       clusters
##                          1   2   3   4
##   3+ veh - en chaine     0  13   0  12
##   3+ veh - multiple      6  16   0   8
##   autre                 20 123 122  56
##   deux veh - arriere    35  37  21  40
##   deux veh - cote      165  52  41  23
##   deux veh - frontal    29  54  14   9
##   sans                  33  27  26  18
##
## Variable : catr
##                   clusters
##                      1   2   3   4
##   autoroute          5   6   4  98
##   autre              1   0   1   0
##   communale        174  79 151   6
##   departementale    87 203  42  25
##   hors reseau        0   1   0   0
##   metropole         12  18  16   4
```

```
##   nationale        9  14  10  33
##   parc             0   1   0   0
##
## Variable : circ
##                                      clusters
##                                         1   2   3   4
##   avec voies d'affectation variable     0   1   1   2
##   bidirectionnel                      210 260 148  26
##   chaussee separees                    24  26  27 101
##   sens unique                          54  35  48  37
##
## Variable : prof
##                  clusters
##                     1   2   3   4
##   base de cote      3   8   5   2
##   pente            47  59  25  26
##   plat            234 249 192 135
##   sommet de cote    4   6   2   3
##
## Variable : plan
##                       clusters
##                          1   2   3   4
##   en courbe a droite    20  30  14  22
##   en courbe a gauche    24  29  23  10
##   en S                   4  14   1   2
##   rectiligne           240 249 186 132
##
## Variable : surf
##               clusters
##                  1   2   3   4
##   autre          4   1   2   1
##   corps gras     1   0   0   0
##   flaques        1   0   0   0
##   inondee        0   0   0   1
##   mouillee      30  59  51  43
##   normale      252 257 169 120
##   verglacee      0   5   2   1
##
## Variable : infra
##                           clusters
##                              1   2   3   4
##   aucun                    248 273 189 140
##   autres                     7  11  10   0
##   bretelle ou échangeur      3   2   0  13
##   carrefour                 15  17  14   1
##   chantier                   3   4   1   0
##   peage                      0   0   1   0
##   pont ou autopont           2   6   3   7
##   souterrain ou tunnel       3   4   3   5
##   voie ferree                4   1   0   0
##   zone pietonne              3   4   3   0
##
## Variable : situ
##                         clusters
```

```
##                          1   2   3   4
##   accotement             6  51  14  13
##   autres                 1  12   5   5
##   bande arret urgence    0   0   0   7
##   chaussee             262 240 181 141
##   piste cyclable         9   2   4   0
##   trottoir               6  15  17   0
##   voie speciale          4   2   3   0
```

E. Visualisation des clusters k-prototypes

```
library(factoextra)
```

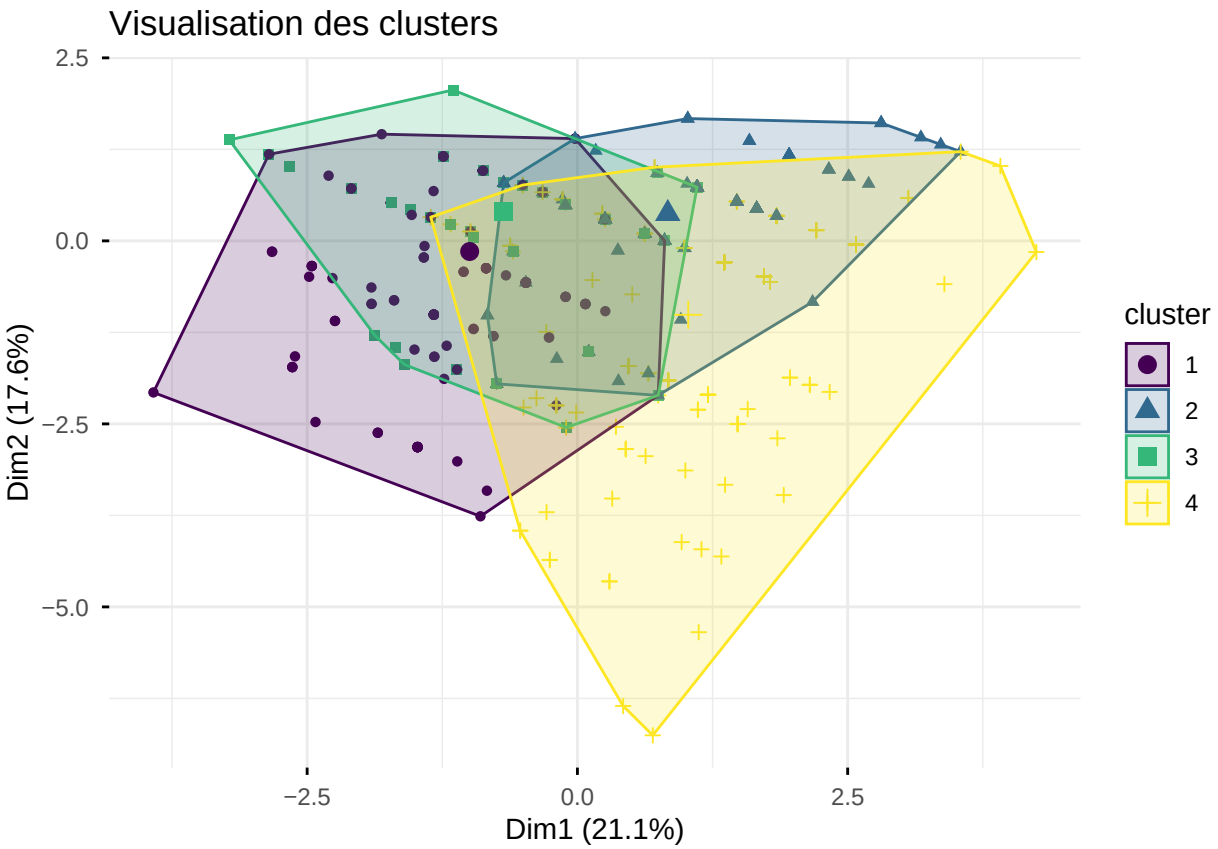
```
## Le chargement a nécessité le package : ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(viridis)
```

```
## Le chargement a nécessité le package : viridisLite
```

```
# Extraire les clusters
clusters <- res$cluster

# Ne garder que les colonnes numériques
num_cols <- sapply(df, is.numeric)
df_num <- df[, num_cols]

fviz_cluster(list(data = df_num, cluster = clusters),
             geom = "point",
             palette = viridis(4),
             ellipse.type = "convex",
             repel = TRUE,
             show.clust.cent = TRUE,
             ggtheme = theme_minimal(),
             main = "Visualisation des clusters")
```



Observations

Les groupes peuvent se chevaucher car la visualisation 2D (via PCA) projette un espace multidimensionnel sur un simple plan. Cette réduction dimensionnelle entraîne une perte d'information, et des clusters bien séparés dans l'espace original peuvent alors apparaître mélangés sur le graphique.

F. Conclusion

L'algorithme K-prototypes a identifié quatre groupes d'accidents distincts en combinant variables numériques et catégorielles. Cette segmentation permet d'analyser les différents profils d'accidents et d'identifier les facteurs de risque spécifiques à chaque groupe.

Ces résultats dépendent de l'échantillon sélectionné et pourraient varier avec un échantillonnage différent.