Examen de statistique multidimensionnelle

Groupe 8 :
HUYLENBROECK Florent
BOSSART Laurent

Juin 2020

Contents

1 Introduction

Dans le cadre de notre cours de statistique multidimensionnelle il nous a été demandé de, sur base d'un fichier de donnée nommé *XXData* :

- Effectuer une analyse univariée des données.
- Effectuer une ACP et en discuter les résultats.
- Effectuer une classification des individus et des variables et en discuter les résultats.

Pour cela, nous allons utiliser le langage de programmation *R* via l'outil *RStudio*.
Il nous a aussi été demandé de présenter une technique d'analyse multivariée non vue en cours : *CLARA* (Clustering Large Applications) et d'en décrire un exemple en R.

2 Analyse univariée des données

Pour commencer l'analyse de nos données, commencons par jeter un oeil au fichier de données en utilisant la fonction *head* de R.

```
> head(XXData)
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]       [,7]      [,8]
[1,]  5.132791 25.596215 11.044340 23.532733 20.087409  5.168034  5.9481452 -4.687937
[2,] 18.430779  5.486921 15.935372 16.190221 -1.141461  7.277743  2.8832279  7.525328
[3,] 13.373781 -5.711065  4.606412  1.915319  9.190822 16.116251 -6.1726472 17.822856
[4,] 17.532933 16.311315  1.496686 24.763584  5.019113 12.276417 -3.2111388 12.135562
[5,]  8.329159 15.063473 23.296983 15.518534 -6.445116 -2.824094  0.6239991  2.971019
[6,] -7.302192 -3.019514  6.400583  6.458306 11.306205 11.311792 -1.7364332 -2.587585
```

Figure 1: Appel et résultat de la fonction head sur XXData

On observe qu'il y a 8 variables, lesquelles semblent etre quantitatives continues contenues dans l'intervalle $[-20; 20]$.
La fonction *summary* nous donne plus d'informations :

```
> summary(XXData)
       V1                  V2                  V3                  V4                  V5                  V6                  V7                  V8
 Min.   :-25.598   Min.   :-42.059   Min.   :-17.986   Min.   :-23.1058   Min.   :-22.044   Min.   :-10.915   Min.   :-18.7966   Min.   :-14.163
 1st Qu.: -5.060   1st Qu.:-12.337   1st Qu.:  4.487   1st Qu.: -0.4802   1st Qu.: -4.087   1st Qu.:  3.265   1st Qu.: -5.1445   1st Qu.: -1.972
 Median :  5.059   Median : -2.349   Median :  9.481   Median :  8.1625   Median :  2.401   Median :  7.226   Median :  0.5488   Median :  4.214
 Mean   :  5.394   Mean   : -1.516   Mean   :  9.224   Mean   :  8.0156   Mean   :  2.040   Mean   :  7.314   Mean   :  0.5583   Mean   :  4.310
 3rd Qu.: 15.662   3rd Qu.:  8.897   3rd Qu.: 14.660   3rd Qu.: 16.0057   3rd Qu.:  8.628   3rd Qu.: 12.370   3rd Qu.:  6.7301   3rd Qu.:  9.744
 Max.   : 41.431   Max.   : 40.226   Max.   : 32.375   Max.   : 47.7696   Max.   : 34.189   Max.   : 23.322   Max.   : 26.6583   Max.   : 27.889
```

Figure 2: Appel et résultat de la fonction summary sur XXData

On observe que nos données sont contenues dans l'intervalles $[-50; 50]$, avec des moyennes assez proches l'une de l'autre. Cet appel nous permet aussi de déterminer la médiane, les extremas, le premier et le troisième quartile de chacune de nos variables pour nos données.

Observons ensuite le boxplot de nos données :

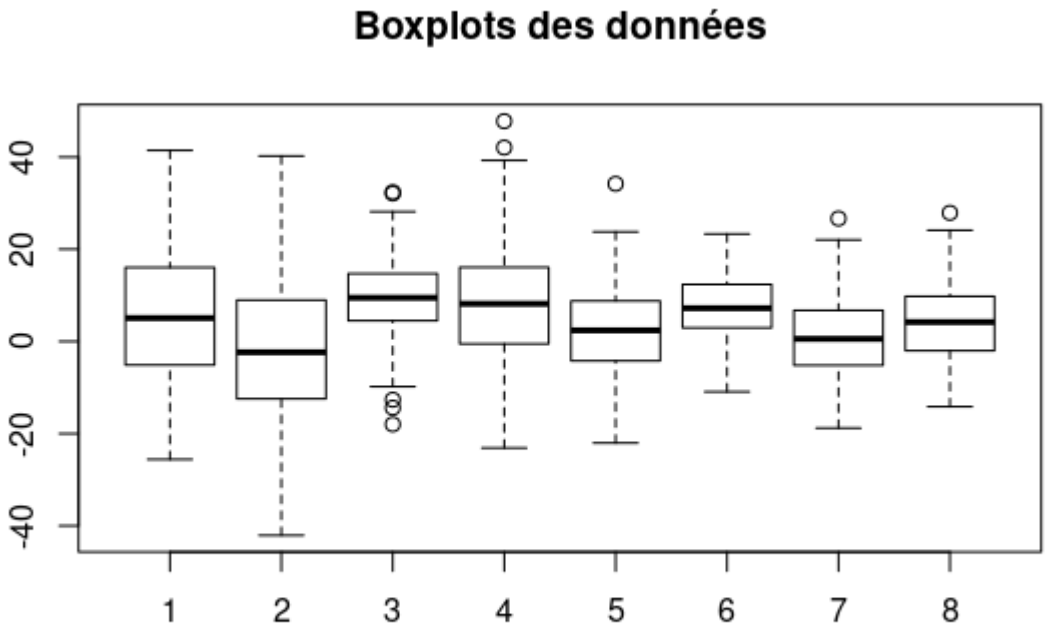

Figure 3: Boxplot des variables de XXData obtenu via la fonction *boxplot*

On observe sur cette figure que, en effet, nos moyennes sont assez proches et que globalement les individus sont proches les uns des autres.
Pour finir, un dernier appel à la fonction *nrow* nous indique qu'il y a 128 individus dans XXData.

3 ACP

Le but de l'ACP (*Analyse en composantes principales*) est de réduire le nombre de variables tout en conservant au maximum l'information.
Pour commencer, malgré que toutes nos données aient la même unité (pas d'unité) et le même intervalle de valeurs, nous allons les centrer et réduire :

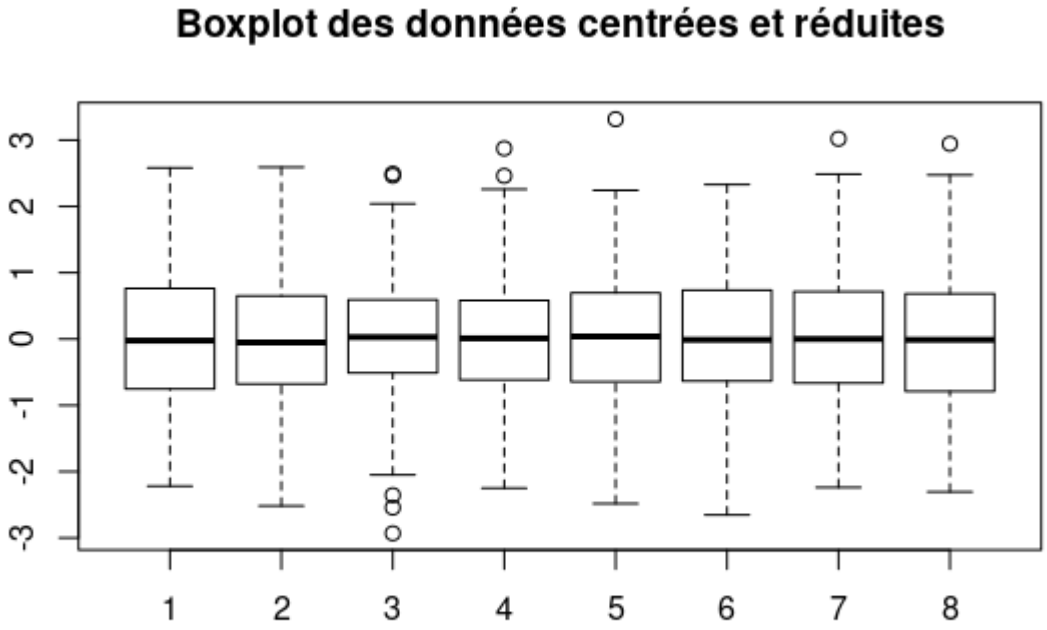

Figure 4: Boxplots des données centrées et réduites.

Nous allons maintenant nous intéresser à la matrice de corrélation de nos variables. La diagonalisation de cette matrice nous permettra de définir le nombre de composantes à conserver.

```
> XXData.cor <- cor(XXData)
> XXData.cor
           [,1]        [,2]        [,3]       [,4]        [,5]        [,6]       [,7]        [,8]
[1,]  1.00000000 -0.21928409 -0.13450546  0.54778649 -0.02474005 -0.40861205  0.29575833  0.63910241
[2,] -0.21928409  1.00000000  0.13534505  0.12750734  0.30524887  0.02109043 -0.28815527 -0.19482880
[3,] -0.13450546  0.13534505  1.00000000 -0.12221569 -0.26936469 -0.50746777  0.07674444 -0.45489298
[4,]  0.54778649  0.12750734 -0.12221569  1.00000000 -0.33504047 -0.41401040  0.71845562  0.03020634
[5,] -0.02474005  0.30524887 -0.26936469 -0.33504047  1.00000000  0.32997839 -0.67257950  0.32083032
[6,] -0.40861205  0.02109043 -0.50746777 -0.41401040  0.32997839  1.00000000 -0.40428573  0.21700070
[7,]  0.29575833 -0.28815527  0.07674444  0.71845562 -0.67257950 -0.40428573  1.00000000 -0.29199724
[8,]  0.63910241 -0.19482880 -0.45489298  0.03020634  0.32083032  0.21700070 -0.29199724  1.00000000
```

Figure 5: Matrice de corrélation obtenue via la fonction *cor*

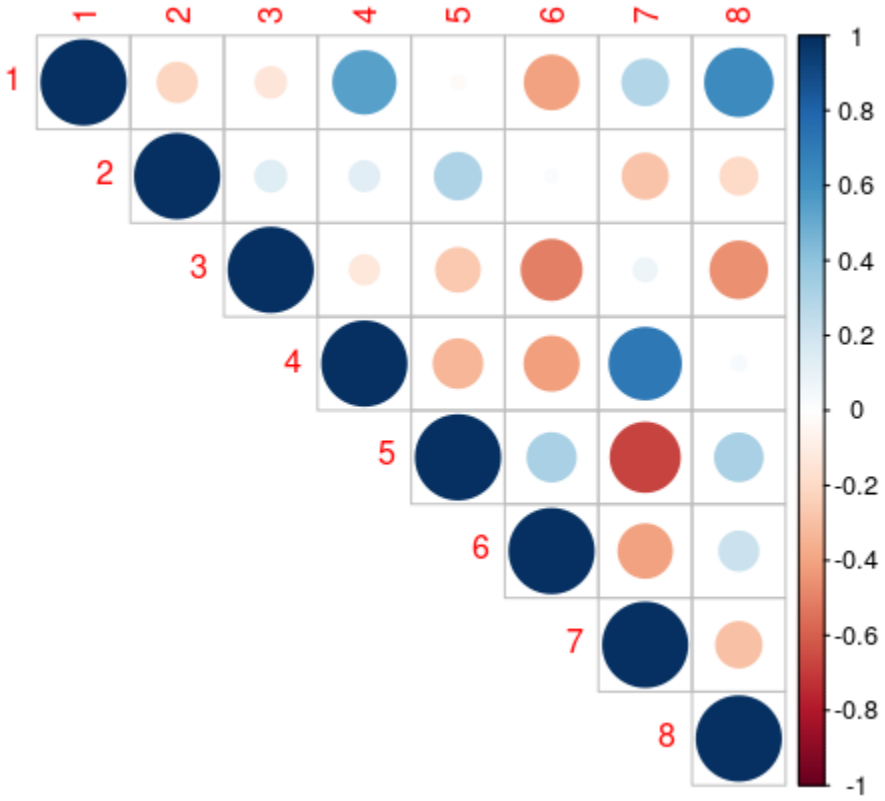

Figure 6: Représentation de la matrice de corrélation générée via la fonction *corrplot*

On remarque une forte corrélation entre les variables $V5$-$V7$, $V4$-$V7$ et $V8$-$V1$, le reste des variables ne semblent pas fortement corrélées.

Calculons maintenant les valeurs propres de cette matrice à l'aide de la fonction *eigen* de R, nous obtenons :

```
> XXData.eigen <- eigen(XXData.cor)
> XXData.eigen$values
[1] 2.81728269 2.09181102 1.25259652 1.03705552 0.40408241 0.25261766 0.08268080 0.06187339
```

Figure 7: Valeurs propres de la matrice de corrélation

Selon le critère de Kaiser, nous devons conserver les 4 premières composantes car leur valeur propres sont > 1.
Regardons le graphe en éboulis du pourcentage de la variance expliqué par composantes pour renforcer notre décision :

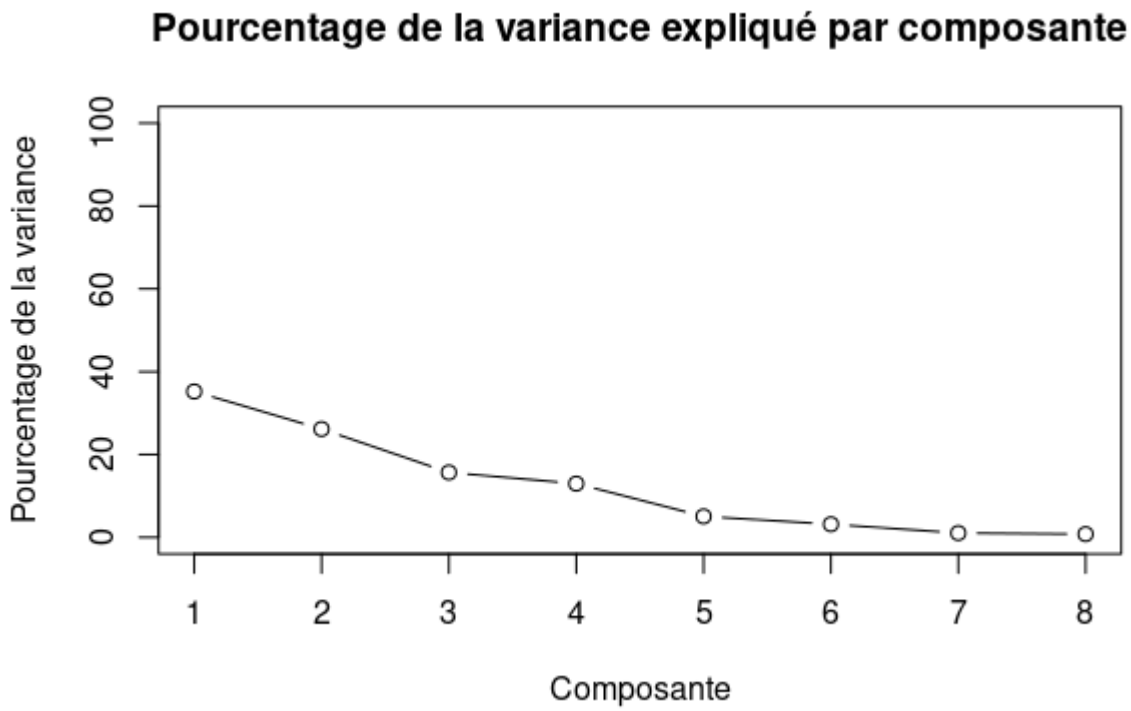

Figure 8: Graphe en éboulis du pourcentage de variance expliqué par composante

Nous espérions observer un coude dans ce graphe, mais il n'y en a qu'un très léger après la 3e composante. Ce coude n'est pas assez satisfaisant que pour en déduire un nombre de composante à conserver. Nous décidons donc de conserver 4 composantes selon le critère de Kaiser, lesquelles nous permettrons d'expliquer $\approx 90\%$ de la variance totale comme nous l'indique la figure 9.

Nombre de composantes	Variance expliquée cumulée (%)
1	35.21
2	61.36
3	77.02
4	89.98
5	95.03
6	98.19
7	99.22
8	100

Figure 9: Pourcentages de la variance expliquée cumulés

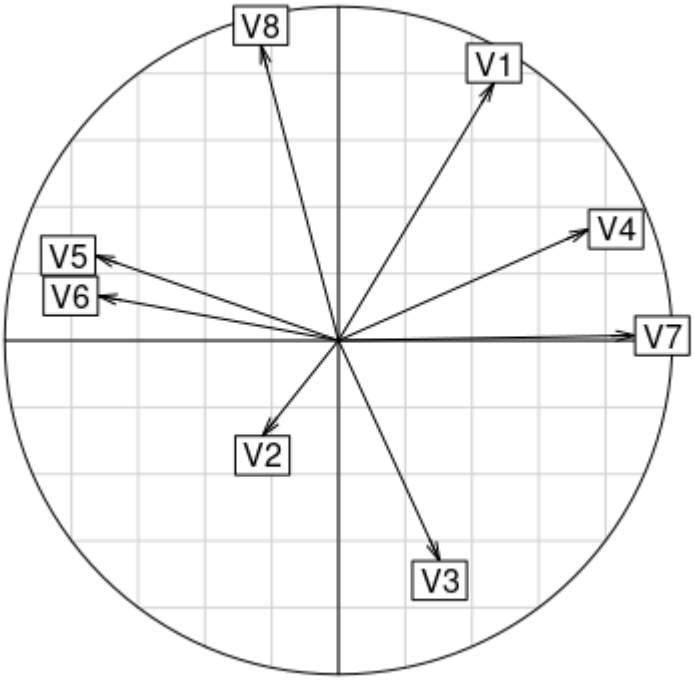

Figure 10: Cercle de corrélation généré via *corcirle*

Le cercle des corrélations corresponds à ce que nous avions observé dans la matrice de corrélations, $V5$ est opposé à $V7$ ce qui suggère une forte corrélation négative, $V1$ est proche de $V8$ et $V4$ est proche de $V7$ ce qui suggère une corrélation forte.

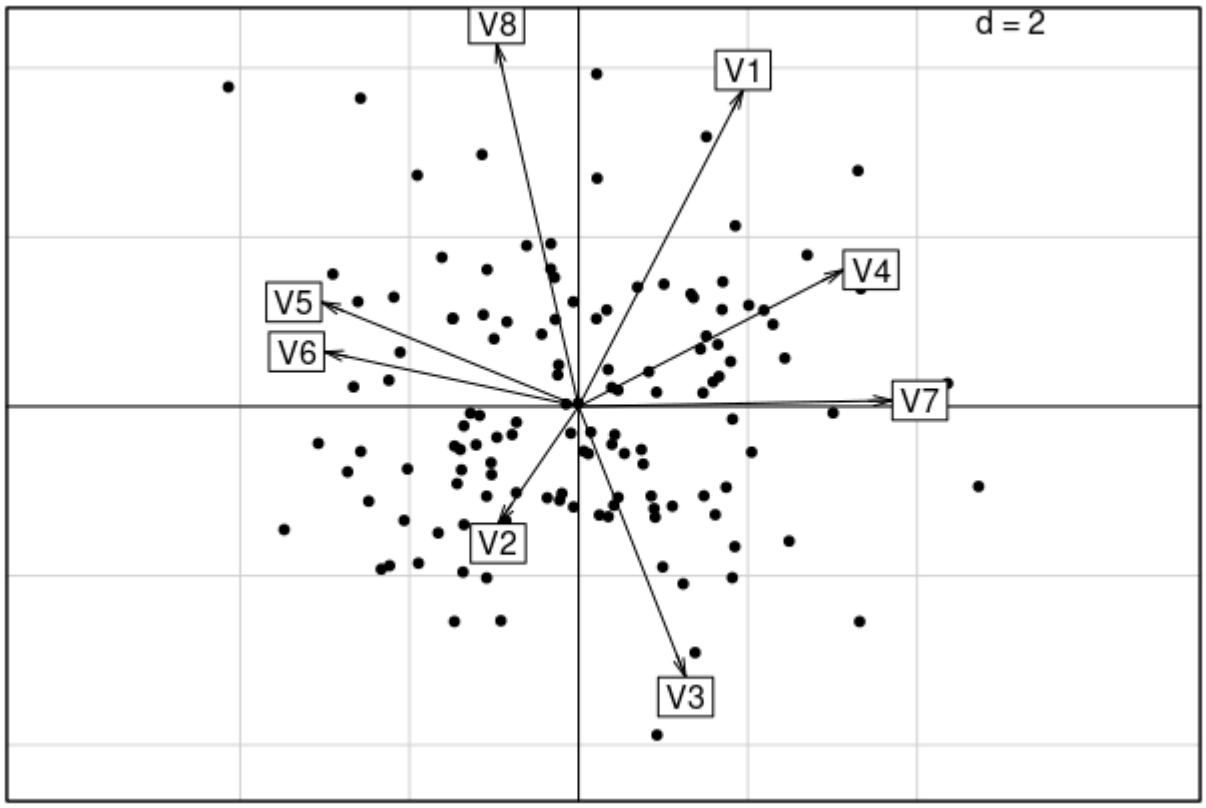

Figure 11: Scatterplot généré via la fonction *scatter*

Etant donné la dispersion des données sur le scatterplot, il est compliqué de tirer des conclusions.

4 Classification

Pour tenter de classifier nos données, nous allons utiliser la méthode du *k-means clustering*. La première étape est de déterminer un nombre de cluster correct à utiliser. Pour ce faire nous allons étudier la variance intra groupe en fonction du nombre de cluster considérés, afin d'y localiser un coude ou une très nette diminution de la décroissance de la variance intra groupe.

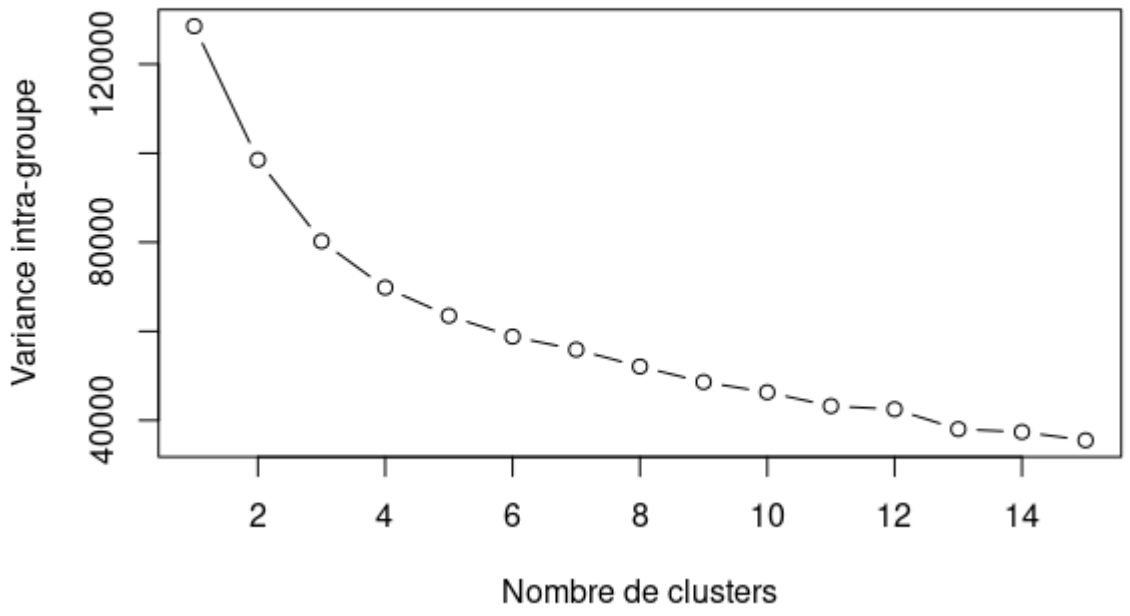

Figure 12: Variance intra groupe en fonction du nombre de cluster considéré

Il n'y a pas de net coude dans ce graphe, nous allons donc considérer 3 clusters car cela semble être le moment ou la diminution de la variance intra groupe devient plus faible.
La fonction *kmean* appliquée sur nos composantes nous permet d'obtenir une classification.

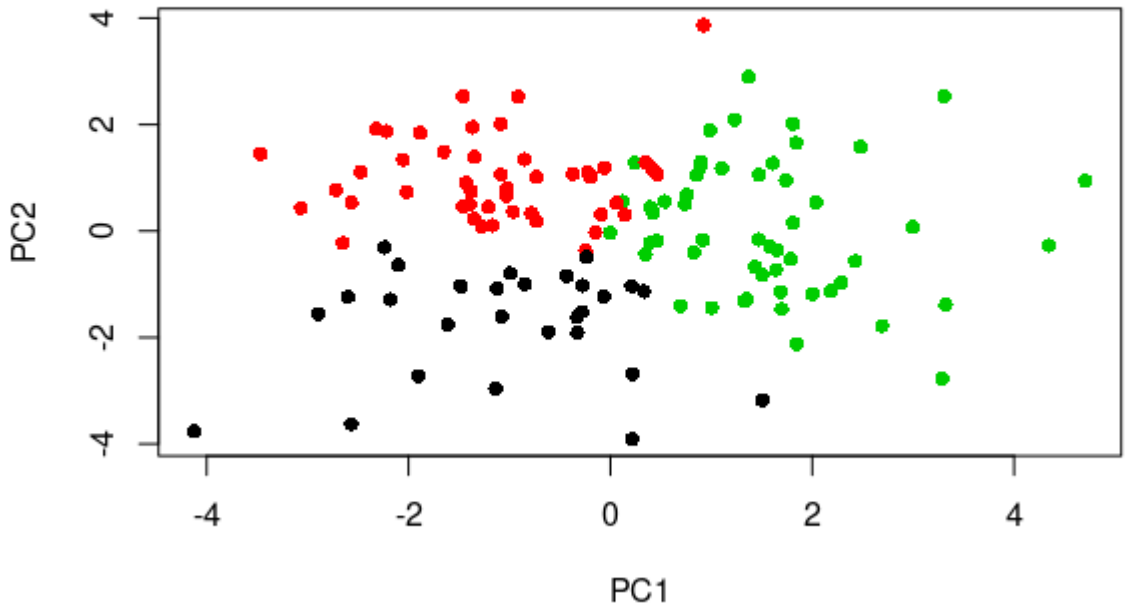

Figure 13: Classification obtenue avec la méthode *kmeans* pour $k = 3$

4.1 Conclusion

L'analyse en compsante principales nous a permis de réduire la dimension de nos données de 8 à 4 en conservant $\approx 90\%$ de l'information. Nous avons ensuite appliqué un algorithme de *k-means clustering* afin de classifier nos données. Il est compliqué de discuter de la cohérence de cette classification étant donné l'aspect aléatoire et la similarité de nos variables dans le jeu de données originel.

5 CLARA (Clustering LARge Applications

Le *clustering* (regroupement) d'un ensemble d'objets avec CLARA se fait en deux étapes. Premièrement, un échantillon est pioché aléatoirement dans l'ensemble des objets et regroupé en k sous-ensembles en utilisant la méthode k-*medoid*, qui donne aussi k objets représentatifs.
Ensuite, chaque objet n'appartenant pas a l'échantillon est affecté au plus proche des k objets représentatifs. De cette façon, on obtient un clustering de l'ensemble des données. Une mesure de la qualité de ce clustering est obtenue en calculant la distance moyenne entre chaque objet de l'ensemble des données et de son objet représentatif. Apres avoir pioché aléatoirementf et regroupé 5 échantillons, celui qui a la plus petite distance moyenne est séléctionné.

5.0.1 Méthode k-medoid en quelques mots

L'algorithme $k - medoid$ est une approche de clustering utilisée pour partitionner un ensemble de données en k groupes ou clusters. Dans cette méthode, chaque cluster est représenté par un seul point de donnée du cluster. Ce point est ce que l'on appelle le *medoid*.
Le terme medoid fait référence à un objet à l'intérieur du cluster pour lequel la dissimilitude moyenne entre ce point et tous les autres points du cluster est minimale. Il s'agit du point le plus au centre du cluster. Les medoids peuvent être considérés comme un exemple représentatif des membres de ce cluster. Cette méthode est aussi appelée PAM (Partioning Around Medoids).

5.1 Description de l'algorithme

Avec l'algorithme utilisé dans PAM, on séléctionne k objets (les medoids) qui sont représentatifs ou localisés centralement, et les k clusters sont construits autour de ces objets. L'effort principal de calcul qui sont fais dans l'algorithme PAM est une recherche parmis un grand nombre de sous-ensembles de k objets, pour un sous-ensemble produisant un regroupement satisfaisant et localement optimal. Si on augmente le nombre de données, la méthode k-medoid exacte est seulement faisable pour un nombre d'objets relativement petit car, dans le cas contraire, le temps de calcul devient énormement grand. L'allocation de la mémoire utilisée par PAM dépend principalement du nombre d'objets, qui est une fonction quadratique.

CLARA éffectue le clustering en conjonction avec la recherche par un ensemble d'objets représentatifs, qui devrait réprésenter un aspect différent de la structure de l'ensemble des données.
La méthode utilisée par CLARA est la sélection aléatoire de 5 (ou plus) échantillons d'objets. La taille des échantillons dépend du nombre de clusters. Pour un clustering en k clusters, la taille de l'échantillon est donnée par $40 + 2k$. Le nombre de clusters doit varier entre 1 et 30, donc les échantillons contiennent entre 42 et 100 objets. Le choix d'utiliser une fonction du nombre de clusters pour la taille des échantillons est motivé par l'objectif d'avoir une probabilité raisonnable de trouver des objets de tous les échantillons "existants" dans au moins un des échantillons généré.
Pour la construction du premier échantillon, les objets sont séléctionnés par un nombre généré aléatoirement et ordonnés par un indice croissant. Chaque fois qu'un objet est pioché aléatoirement, on vérifie qu'il ne fait pas partie des objets déjà piochés. S'il n'avait pas encore été séléctionné, il est inséré à la bonne position dans le tableau.
Si la taille de l'échantillon est juste un petit peu plus petite que le nombre d'objets, il arrivera parfois que le même objet sera pioché plusieurs fois de manière aléatoire. C'est pour cette raison qu'à chaque fois que le nombre d'objets est inférieur au double de la taille de l'échantillon, le générateur

de nombres aléatoires est utilisé pour séléctionner les objets n'appartenants pas à l'échantillon. La construction d'autres échantillons est lancée en tenant compte des medoids qui ont été trouvés dans les échantillons précédents. A chaque étape de l'algorithme, le meilleur ensemble de medoids actuel est stocké dans un tableau. (Ce meilleur ensemble est celui pour lequel la distance moyenne pour l'ensemble des données est la plus petite trouvée jusqu'à présent). Un nouvel échantillon est construit en ajoutant des objets à ce meilleur ensemble, de la même manière que les objets ont été accumulés dans le premier échantillon.
Après le prélèvement d'un échantillon d'objets, celui-ci est divisé en k groupes en utilisant le même algorithme que dans le programme PAM. Cet algorithme consiste en deux parties, appelées BUILD et SWAP. Dans BUILD, les objets représentatifs successifs sont séléctionnés dans le but d'obtenir la plus petite distance moyenne possible entre les objets (de l'échantillon) et leur object représentatif le plus similaire. Dans SWAP, on tente de diminuer la distance moyenne en remplacent les objets représentatifs.
Une fois que k objets représentatifs ont été séléctionnés, chaque objets de l'ensemble de données (et pas seulement de l'échantillon) est attribué à l'objet représentatif le plus proche. La distance moyenne obtenue pour l'affectation est utilisée comme une mesure de la qualité du clustering. Une fois ce calcul effectué pour chacun des 5 échantillons, l'échantillon retenu est celui avec la plus petite distance moyenne possible.
Une analyse plus poussée est alors effectuée sur la dernière partition. La liste d'objets de chaque cluster est donnée, ainsi que le medoid et la taille du cluster (dans l'ensemble des données). Le programme va alors lister, pour chaque cluster, la distance moyenne et maximale pour son medoid. Aussi, la distance maximale est divisée par la distance minimal du medoid par rapport a un autre medoid. Cette valeur donne des informations sur l'étroitesse du cluster. Une petite valeur indique un cluster très serré, alors qu'une valeur qui dépasse 1 suggère un cluster faible.

5.2 Exemple

Tentons d'appliquer CLARA à notre jeu de données $XXData$. Pour ce faire nous allons utiliser la fonction *clara* du package *cluster* de R.

```
> XXData.clara <- clara(XXData, 3, metric = "euclidean", stand = FALSE, samples = 5, pamLike = FALSE)
> print(XXData.clara)
Call:    clara(x = XXData, k = 3, metric = "euclidean", stand = FALSE,      samples = 5, pamLike = FALSE)
Medoids:
          [,1]       [,2]       [,3]      [,4]      [,5]      [,6]      [,7]     [,8]
[1,] -0.8632284  10.975156 14.6004323  5.724836  2.411247 11.520076 -3.279214 8.314470
[2,] 18.2224897 -12.702528  0.7448688 16.931890  3.928550  5.097058  4.030079 6.960663
[3,] -9.9824517  -8.290902 10.1844218 -8.432368 -3.960087 13.695078 -1.435293 4.960927
Objective function:      25.34709
Clustering vector:       int [1:128] 1 1 2 1 1 1 2 1 2 3 1 1 1 2 1 1 2 1 ...
Cluster sizes:           50 43 35
Best sample:
 [1]   2   5   8   9  12  15  17  20  21  26  27  28  34  35  36  37  40  45  51  56  61  69  74  77  78  83
```

Figure 14: Résultat de l'application de la fonction *clara* à notre jeu de données $XXData$

Cette fonction retourne les informations suivantes :

- les **medoids** : les médoids tels que définis plus haut.
- **Clustering** : un vecteur indiquant à quel cluster apaprtient l'objet d'indice correspondant.

- **sample** : Les individus contenus dans le meilleur échantillon, lequel sera utilisé par clara pour le partitionnement.

Et pour finir, le résultat de clara porté sur un graphique :

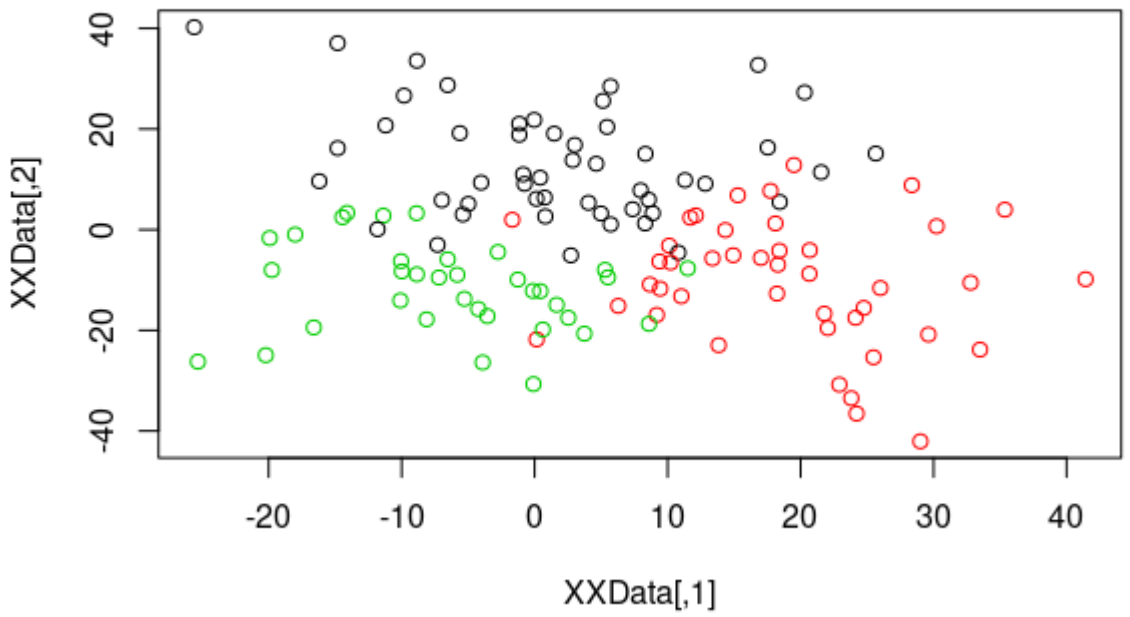

Figure 15: Résultat graphique de Clara sur notre jeu de données XXData

A noter que pour cet exemple nous avons arbitrairement pris $k = 3$ clusters, pour obtenir des résultats comparables à notre classification de la question 1.