Familiarisation avec les données

Solution

Exercice 1

Importer les packages: *FactoMineR,Cluster,FSelector,mlbench, factoextra*.

Jeux de données choisi Glass

Soit de Kaggle:

```
> Glass_Dataset =  read.csv("/home/ludmila/Downloads/archive/glass.csv")
> head(Glass_Dataset)
       RI    Na   Mg   Al    Si    K   Ca Ba   Fe Type
1 1.52101 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00    1
2 1.51761 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00    1
3 1.51618 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00    1
4 1.51766 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00    1
5 1.51742 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00    1
6 1.51596 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26    1
>
```

Soit de mlbench:

```
library(mlbench)
data(Glass)
```

<u>Voir les caractéristiques du dataset:</u>

```
> dim(Glass_Dataset)
[1] 214  10
> summary(Glass_Dataset)
       RI              Na              Mg              Al              Si              K               Ca              Ba        
 Min.   :1.511   Min.   :10.73   Min.   :0.000   Min.   :0.290   Min.   :69.81   Min.   :0.0000   Min.   : 5.430   Min.   :0.000  
 1st Qu.:1.517   1st Qu.:12.91   1st Qu.:2.115   1st Qu.:1.190   1st Qu.:72.28   1st Qu.:0.1225   1st Qu.: 8.240   1st Qu.:0.000  
 Median :1.518   Median :13.30   Median :3.480   Median :1.360   Median :72.79   Median :0.5550   Median : 8.600   Median :0.000  
 Mean   :1.518   Mean   :13.41   Mean   :2.685   Mean   :1.445   Mean   :72.65   Mean   :0.4971   Mean   : 8.957   Mean   :0.175  
 3rd Qu.:1.519   3rd Qu.:13.82   3rd Qu.:3.600   3rd Qu.:1.630   3rd Qu.:73.09   3rd Qu.:0.6100   3rd Qu.: 9.172   3rd Qu.:0.000  
 Max.   :1.534   Max.   :17.38   Max.   :4.490   Max.   :3.500   Max.   :75.41   Max.   :6.2100   Max.   :16.190   Max.   :3.150  
       Fe              Type      
 Min.   :0.00000   Min.   :1.00  
 1st Qu.:0.00000   1st Qu.:1.00  
 Median :0.00000   Median :2.00  
 Mean   :0.05701   Mean   :2.78  
 3rd Qu.:0.10000   3rd Qu.:3.00  
 Max.   :0.51000   Max.   :7.00  
>
```

```
> str(Glass_Dataset)
'data.frame':   214 obs. of  10 variables:
 $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
 $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
 $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
 $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
 $ Si  : num  71.8 72.7 73 72.6 73.1 ...
 $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
 $ Ca  : num  8.75 7.83 7.78 8.22 8.07 8.07 8.17 8.24 8.3 8.4 ...
 $ Ba  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
 $ Type: int  1 1 1 1 1 1 1 1 1 1 ...
>
```

Calculer le temps d'exécution avant et après la sélection:

T1<-Sys.time()
 # code R
T2<-Sys.time()
T2-T1

Exemple:

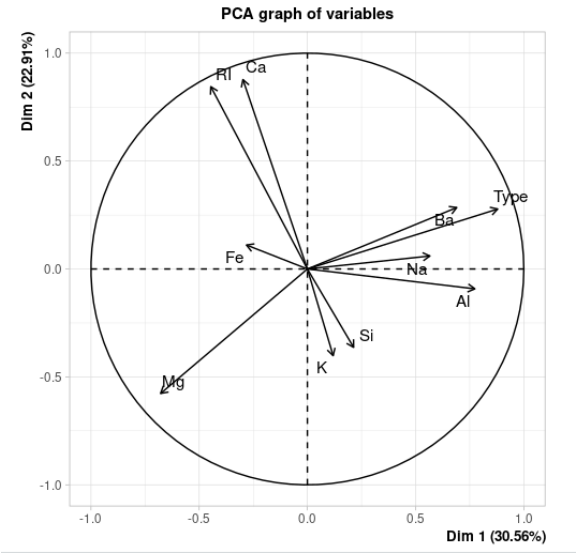
```
T1<-Sys.time()

# CODE R
Glass_Dataset =  read.csv("/home/ludmila/Downloads/archive/glass.csv")
head(Glass_Dataset)
dim(Glass_Dataset)
summary(Glass_Dataset)
# Centrer et réduire les données
donnees <- scale(Glass_Dataset)

# Effectuer l'ACP
resultats_acp <- PCA(donnees)
resultats_acp

T2<-Sys.time()
T2-T1
```

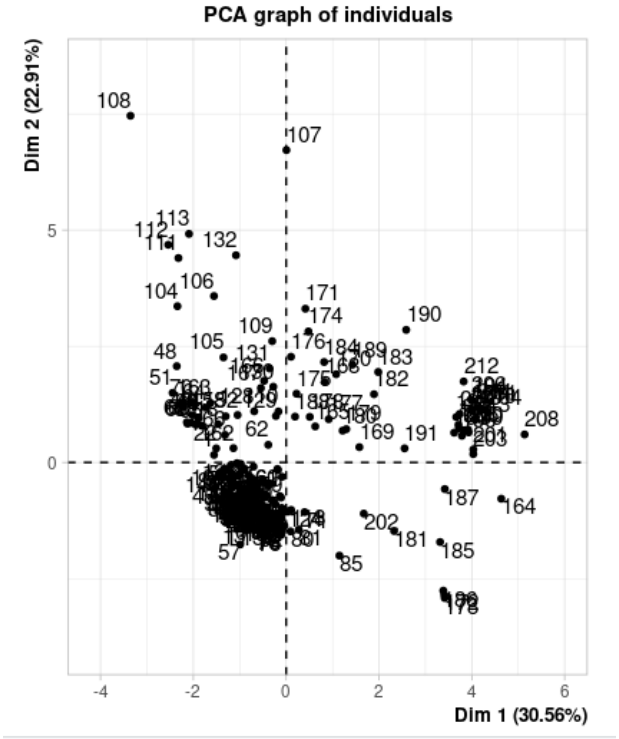
```
> T2-T1
Time difference of 0.7279496 secs
```

Visualiser les données à l'aide d'une ACP:

```
# Separer labels et features
glass_features <- Glass[, 1:9]
glass_labels <- as.factor(Glass[, 10])

# ACP
resultats_acp <- PCA(glass_features, scale =  TRUE)

# Voir resultats
print(resultats_acp)
```

Remarque: Ici on a omis la colonne output “Type” pour pouvoir effectuer une ACP.





Extraire les composantes principales qui correspondent à plus de 80% de qualité:

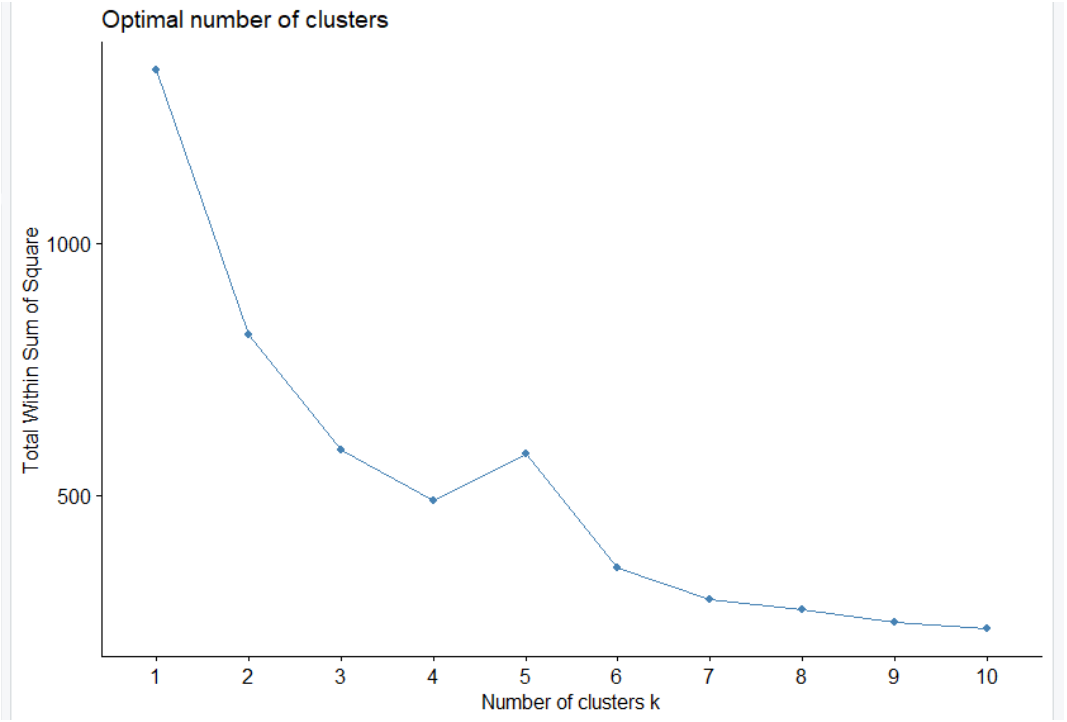
```
> summary(resultats_acp)

Call:
PCA(X = donnees)


Eigenvalues
                       Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7   Dim.8   Dim.9  Dim.10
Variance               3.056   2.291   1.409   1.166   0.914   0.547   0.369   0.183   0.062   0.002
% of var.             30.556  22.913  14.094  11.663   9.141   5.474   3.694   1.827   0.621   0.016
Cumulative % of var.  30.556  53.469  67.564  79.227  88.368  93.842  97.536  99.363  99.984 100.000

Individuals (the 10 first)
```

D'après la prise d'écran on en déduit qu'il faut garder 5 axes pour assurer une inertie cumulée > 80%

Faire du clustering: K-means sur le dataset 'Glass':

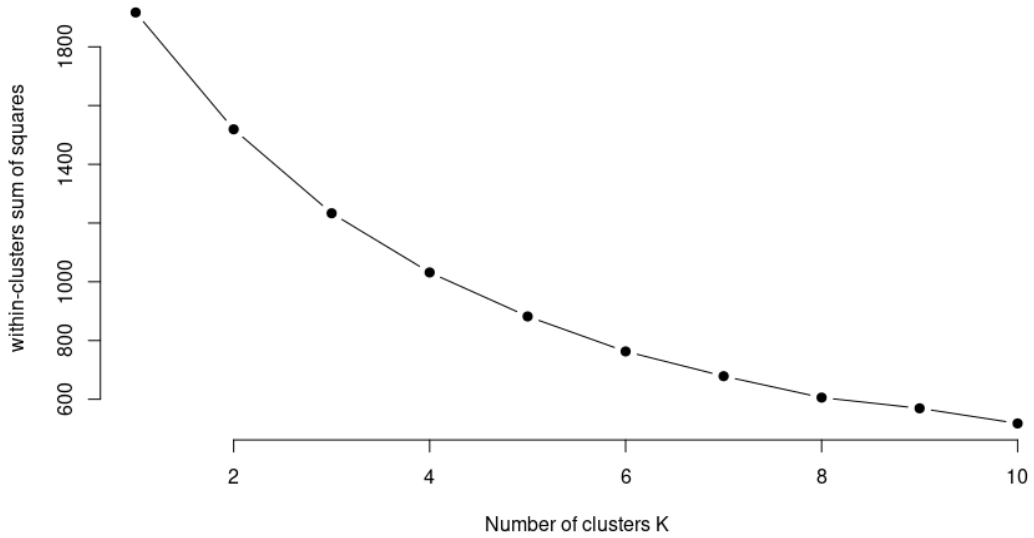
- **Déterminer le nombre optimal de cluster par la méthode Elbow:**

```
> fviz_nbclust(x = Glass_Dataset,FUNcluster = kmeans, method = 'wss' )
```



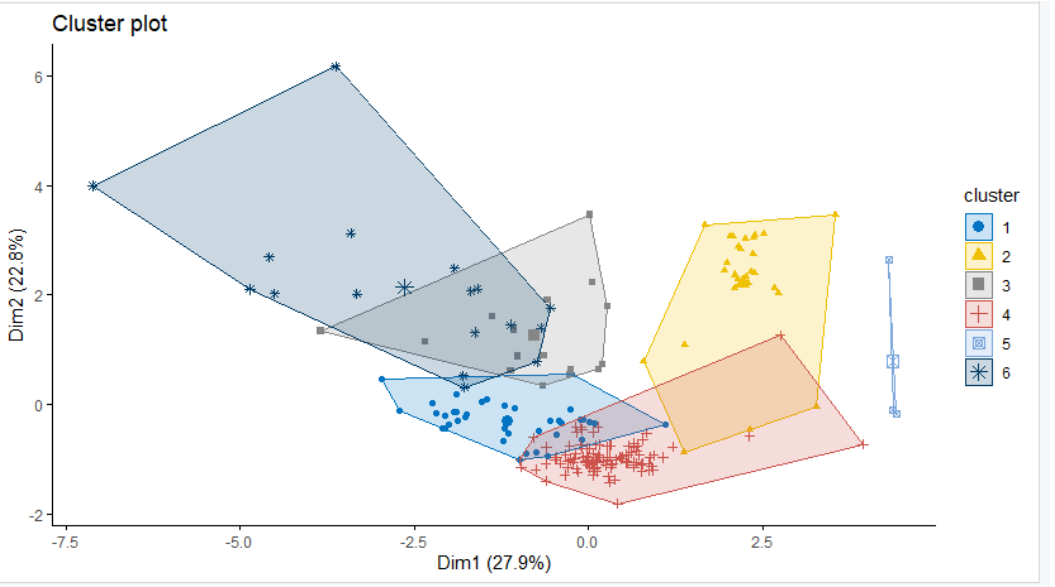
ou à travers ce code:

```
2- # Compute and plot wss for k = 2 to k = 15.
k.max <- 10
data <- scaled_data
wss <- sapply(1:k.max, function(k){kmeans(data, k, nstart=50,iter.max = 15 )$tot.withinss})
wss
plot(1:k.max, wss, type="b", pch = 19, frame = FALSE, xlab="Number of clusters K", ylab="Total within-clusters sum of squares")
```



- **Évaluer ce clustering (avec Elbow, Silhouette):** D'après la méthode du Elbow, on remarque qu'on doit avoir 6 clusters.
- **Résultats du clustering :**
  - **Sans ACP :**

```
> C
K-means clustering with 6 clusters of sizes 38, 27, 17, 16, 111, 5

Cluster means:
        RI       Na        Mg       Al       Si         K        Ca          Ba         Fe
1 1.520482 13.72579 3.6734211 1.013158 71.97763 0.2626316  9.173684 0.025000000 0.06631579
2 1.516230 14.49481 0.1303704 2.097778 73.43407 0.2040741  8.601481 0.950370370 0.01444444
3 1.523889 12.59235 0.2305882 1.274706 72.54412 0.2670588 12.695882 0.185294118 0.07764706
4 1.520871 14.02500 1.8925000 1.573750 71.92625 0.2706250 10.028750 0.136875000 0.06437500
5 1.517139 13.06153 3.4958559 1.384054 72.88495 0.5827928  8.366486 0.004414414 0.06252252
6 1.514240 13.60800 1.8280000 2.718000 71.02600 3.4640000  6.196000 1.004000000 0.00000000
```




```
within cluster sum of squares by cluster:
[1]  34.19231  81.46611 108.67284  44.69015  51.67411  50.57595
 (between_SS / total_SS =  72.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> |
```

```
T1 <- Sys.time()
C = kmeans(data,6)
T2 <- Sys.time()
T2 - T1
```

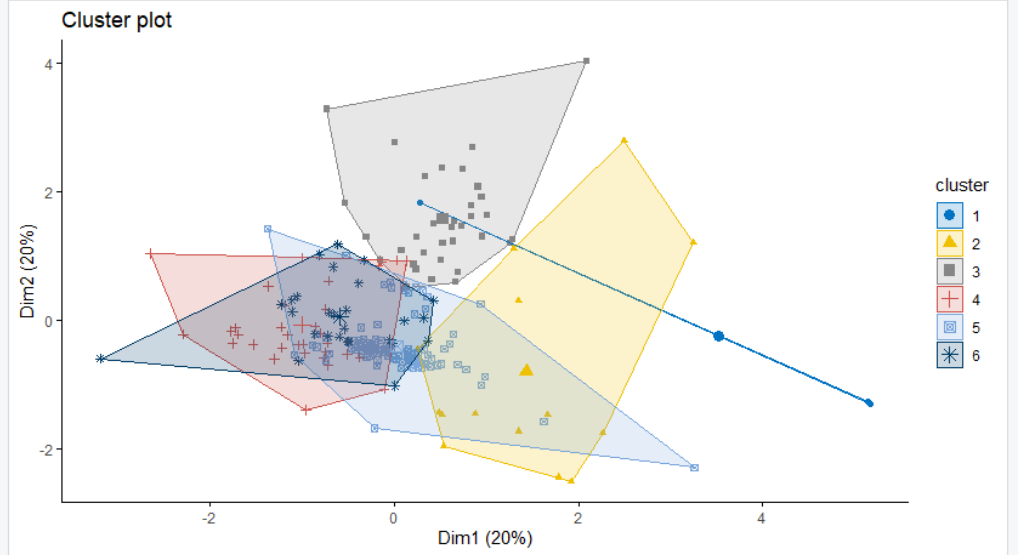
```
> T2 - T1
Time difference of 0.004213095 secs
```

T.exec = 0.0042s
Evaluation = 72.4%

- **Avec ACP :**

```
data1 = resultats_acp$ind$coord[,1:5]
C1 = kmeans(data1,6)
C1
```

```
> C1
K-means clustering with 6 clusters of sizes 35, 27, 34, 101, 14, 3

Cluster means:
       Dim.1      Dim.2      Dim.3       Dim.4      Dim.5
1 -1.4796968  0.3995960 -0.3675877 -1.24948187 -0.1270171
2  2.1647199  2.4264358 -0.8743201  0.20189555  0.3566150
3 -0.3844879 -0.9273896  0.5590263  0.52495424  1.4943442
4  0.3441865 -0.8166111 -0.1327830  0.01894786 -0.3960058
5 -2.9638027  2.2948297  0.7908881  1.58493481 -0.7288055
6  4.3816492  0.7939090  6.6013225 -1.22352643 -1.9302816
```




```
Within cluster sum of squares by cluster:
[1] 111.99580  82.31433  65.94441 153.32202 141.72223  30.88698
 (between_SS / total_SS =  65.9 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

```
# AVEC ACP
T1 <- Sys.time()
C1 = kmeans(data1,6)
T2 <- Sys.time()
T2 - T1
```

```
> T2 - T1
Time difference of 0.001994371 secs
```

T.exec = 0.0019s
Evaluation = 65.9 %

- **Avec FS :**
  Faire une sélection d'attributs:

Exemple:
data = Glass
library(FSelector)
result <- cfs(Type ~ ., data)
result
f <- as.simple.formula(result, "Type")

```
> result <- cfs(Type~ ., Glass)
> result
[1] "Mg" "Al" "K"  "Ca" "Ba"
> f <- as.simple.formula(result, "Type")
```

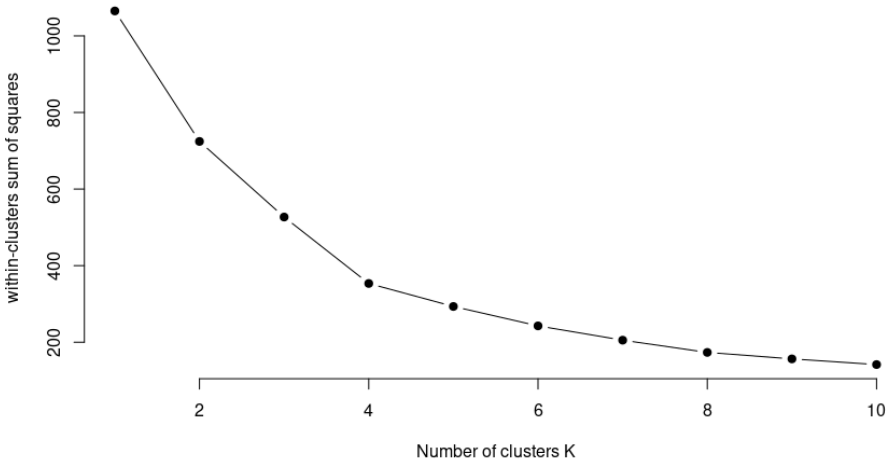
Les variable les plus importantes sont : Mg , Al , K , Ca , Ba donc on laisse seulement ces derniers dans
notre dataset

Refaire le clustering sur les données sélectionnées (en considérant juste les attributs sélectionnés):

```
f <- as.simple.formula(result, "Type")
data$RI = NULL
data$Na = NULL
data$Si = NULL
data$Fe = NULL
data$Type = NULL
head(data)
```

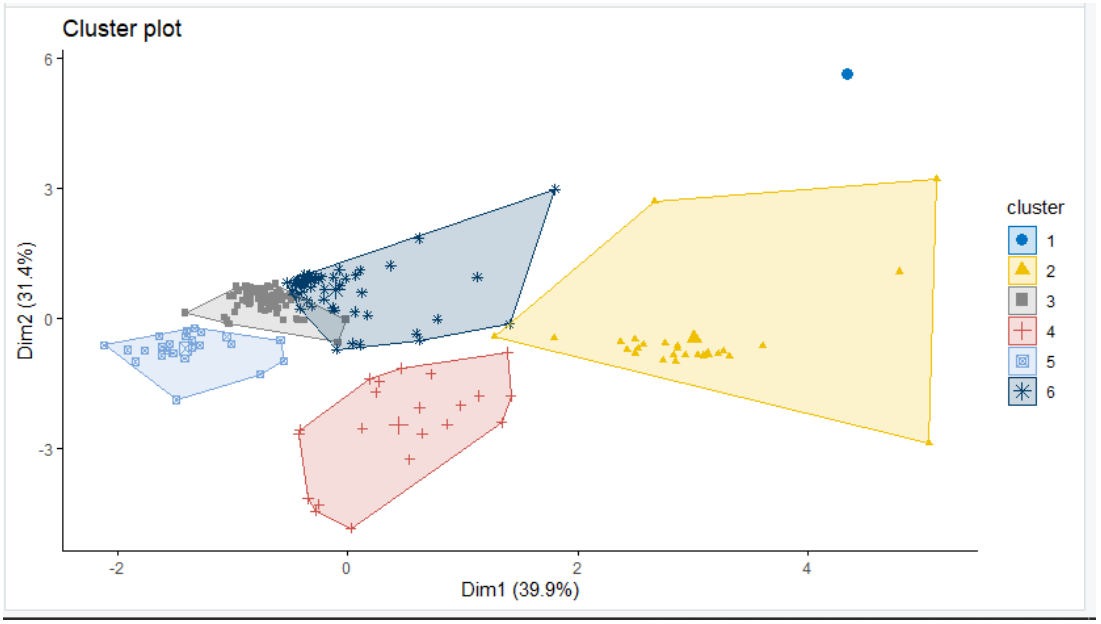
```
> head(data)
    Mg   Al    K   Ca Ba
1 4.49 1.10 0.06 8.75  0
2 3.60 1.36 0.48 7.83  0
3 3.55 1.54 0.39 7.78  0
4 3.69 1.29 0.57 8.22  0
5 3.62 1.24 0.55 8.07  0
6 3.61 1.62 0.64 8.07  0
>
```

Avec la méthode du Elbow:



On garde 6 Clusters.

```
K-means clustering with 6 clusters of sizes 2, 27, 79, 21, 26, 59

Cluster means:
          Mg         Al          K         Ca         Ba
1 -1.8611468  3.1748244  8.75960649 -1.4137355 -0.3520514
2 -1.6159292  1.6214443 -0.42878283 -0.2437985  2.2043884
3  0.5881684 -0.3868338  0.06914503 -0.2754628 -0.3271025
4 -1.6280709 -0.2024898 -0.34960335  2.2463121 -0.3520514
5  0.5274874 -1.3610319 -0.57872648  0.4418403 -0.3404484
6  0.3620661  0.3401753  0.18616992 -0.4659120 -0.2835347

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31
  3   3   6   3   3   6   3   3   3   3   6   3   6   3   3   3   3   5   3   6   6   5   3   3   3   3   3   3   3   3   3
 32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62
  3   3   3   3   3   3   3   5   5   3   3   3   5   3   3   3   5   5   3   5   3   3   3   3   3   3   3   3   3   3   3
 63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93
  5   5   5   5   5   5   5   5   6   3   6   6   6   6   6   6   3   6   6   6   3   6   6   6   6   6   6   6   3   6   6
```




```
Within cluster sum of squares by cluster:
[1] 1.024524e-03 1.218383e+02 1.470940e+01 5.342054e+01 2.175626e+01 6.123694e+01
 (between_SS / total_SS =  74.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
> |
```

```
Time difference of 0.002292156 secs
> T1 <- Sys.time()
> C2 = kmeans(data,6)
> T2 <- Sys.time()
> T2 - T1
Time difference of 0.002085924 secs
> C2
```

T.exec = 0.0020s
Evaluation = 74.4%

Conclusion:

Méthode du Clustering	Résultats
Kmeans sans ACP	**Temps d'exec = 0.0042s** **Evaluation ($\frac{between}{total}$) = 72.4%**
Kmeans avec ACP	**Temps d'exec = 0.0019s** **Evaluation ($\frac{between}{total}$) = 65.9 %**
Kmeans avec Feature Selection	**Temps d'exec = 0.0020s**

	Evaluation ($\frac{between}{total}$) = 74.4%

D'après le tableau, on peut voir qu'en effectuant une ACP avant de procéder au Clustering nous a fait gagner en termes de temps d'exécution (). En revanche, on a perdu un taux d'information avec un pourcentage de *6.5%* par rapport à Kmeans sans ACP.
La Feature Selection quant à elle, a non seulement diminué le temps d'exécution mais a aussi obtenu une meilleure évaluation par rapport aux deux autres méthodes (Sans / Avec ACP).

On en déduit qu'effectivement la feature selection peut avoir un rôle important dans l'amélioration du temps d'exécution et l'évaluation du modèle de clustering en réduisant la dimensionnalité des données (réduire le nombre de variables d'entrée à considérer) , en améliorant la qualité des clusters (en éliminant les variables d'entrée qui sont redondantes ou sans importance pour le modèle), en améliorant l'interprétabilité des résultats et en réduisant le risque d'overfitting