Assignment 1 de Data Mining

VP5002RP

24/04/2021

Description of the project

Le jeu de données que nous utilisons est constitué de caractéristiques de chiffres manuscrits de '0' à '9' extraits d'une collection de cartes utilitaires néerlandaises. 200 motifs par classe (pour un total de 2 000 motifs) ont été numérisés en images binaires. Ces chiffres sont représentés en termes des six ensembles de caractéristiques suivants (fichiers) :

1. mfeat-fou: 76 Fourier coefficients of the character shapes;
2. mfeat-fac: 216 profile correlations;
3. mfeat-kar: 64 Karhunen-Love coefficients;
4. mfeat-pix: 240 pixel averages in 2 x 3 windows;
5. mfeat-zer: 47 Zernike moments;
6. mfeat-mor: 6 morphological features.

Dans chaque fichier, les 2000 motifs sont stockés en ASCI sur 2000 lignes. Les 200 premiers motifs sont de la classe '0', suivis par des ensembles de 200 motifs pour chacune des classes `1' -9'`. Les motifs correspondants dans les différents ensembles de caractéristiques (fichiers) correspondent au même caractère original.

data import

In this part, we import the 06 data files:

```
mfeat.fac <- read.table("C:/Users/STUDENT/Desktop/Cours/Data Mining/mfeat/mfeat-fac", quote="\"", comme
mfeat.mor <- read.table("C:/Users/STUDENT/Desktop/Cours/Data Mining/mfeat/mfeat-mor", quote="\"", comme
mfeat.fou <- read.table("C:/Users/STUDENT/Desktop/Cours/Data Mining/mfeat/mfeat-fou", quote="\"", comme
mfeat.kar <- read.table("C:/Users/STUDENT/Desktop/Cours/Data Mining/mfeat/mfeat-kar", quote="\"", comme
mfeat.pix <- read.table("C:/Users/STUDENT/Desktop/Cours/Data Mining/mfeat/mfeat-pix", quote="\"", comme
mfeat.zer <- read.table("C:/Users/STUDENT/Desktop/Cours/Data Mining/mfeat/mfeat-zer", quote="\"", comme
```

Donnée mfeat.fac

Nous allons commencer avec les donnees mfeat.fac. Nous n'avons pas assez d'information sur les données

```
#head(mfeat.fac, n=5)
```

```
# class(mfeat.fac)        # checking your data class

# dim(mfeat.fac)          # getting the dimension of your data

# names(mfeat.fac)        # getting the column names

# str(mfeat.fac)          # checking data structure

# anyNA(mfeat.fac)        # checking for missing values
```

```
# summary(mfeat.fac)      # summary of the data

# View(mfeat.fac)         # view your data
```

Library for PCA

```
## Warning: package 'FactoMineR' was built under R version 3.6.3
```

```
## Warning in as.POSIXlt.POSIXct(Sys.time()): unable to identify current timezone 'T':
## please set environment variable 'TZ'
```

```
## Warning: package 'factoextra' was built under R version 3.6.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

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

Pour la Description du jeu de données, ou identification de groupes d'individus et liens entre variables nous allons utiliser l'ACP pour décrire ce jeu de données comportant de nombreux individus et variables quantitatives. L'analyse doit permettre d'extraire l'information pertinente et la synthétiser sous forme de composantes principales, nouveaux axes pour décrire le jeu de données.

La fonction PCA()

Nous utilisons la fonction 'PCA()' de 'FactoMineR', elle centre et réduit les variables avant de réaliser l'ACP. Cette étape est importante afin que toutes les variables aient le même poids dans la construction des plans de l'ACP.

```
res.pca <- PCA(mfeat.fac,ncp=10, graph = FALSE)
print(res.pca)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 2000 individuals, described by 216 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Valeurs propres

Pour le choix du nombre de d'axe principale, nous pouvons utiliser la regle de kaiser et la regle de Coude. Nous allons donc afficher le nombre d'axe ou les valeurs propres sont superieur a 1

Comme décrit, les valeurs propres mesurent la quantité de variance expliquée par chaque axe principal. Les valeurs propres sont grandes pour les premiers axes et petits pour les axes suivants. Autrement dit, les premiers axes correspondent aux directions portant la quantité maximale de variation contenue dans le jeu de données.

Nous examinons les valeurs propres pour déterminer le nombre de composantes principales à prendre en considération en fonction de la regle de kaiser et de coude

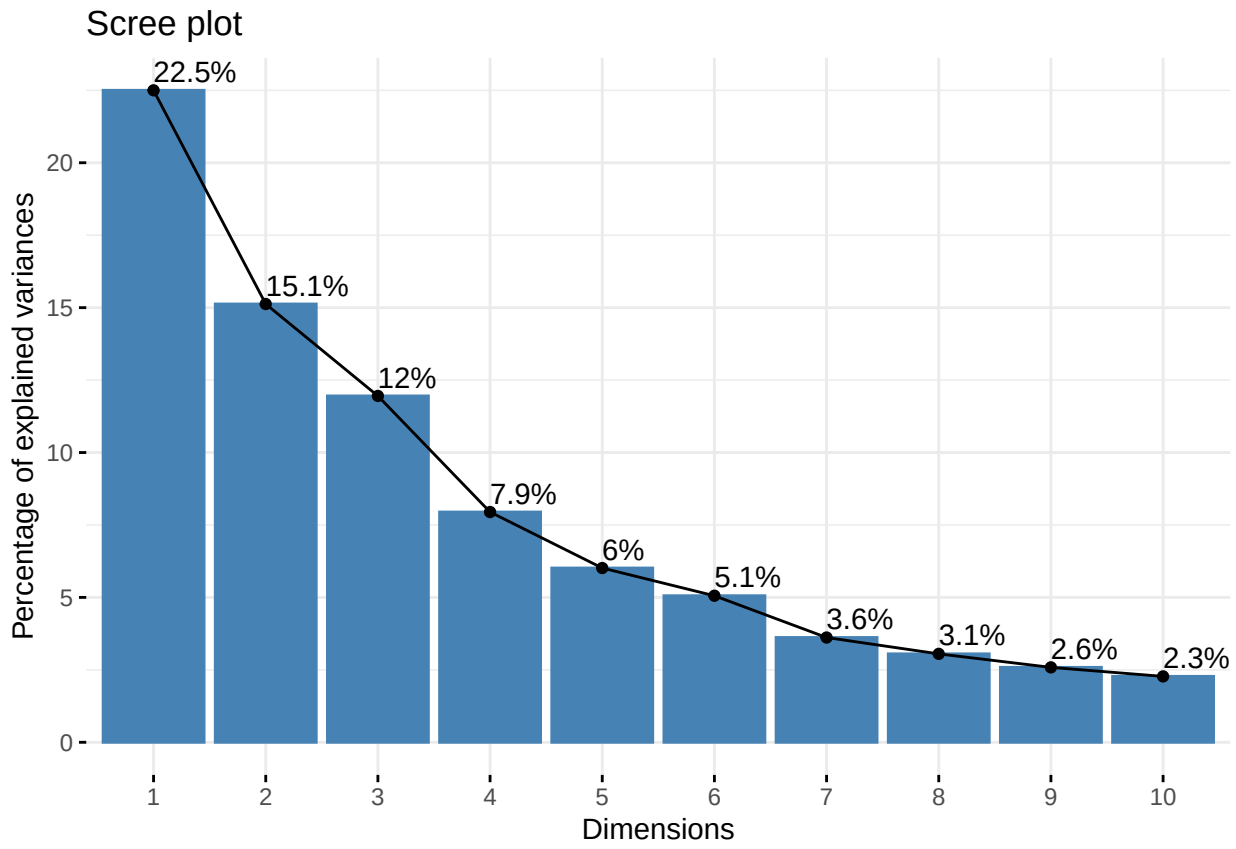
Nous visualisons le tableau seulement pour les lignes dont les valeurs propres sont supperieur a 1

```
dt=as.data.frame(res.pca$eig)
dt[dt$eigenvalue >= 1, ]
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   48.598186             22.4991600                            22.49916
## comp 2   32.670302             15.1251400                            37.62430
## comp 3   25.816359             11.9520178                            49.57632
## comp 4   17.164597              7.9465726                            57.52289
## comp 5   12.990413              6.0140799                            63.53697
## comp 6   10.924094              5.0574507                            68.59442
## comp 7    7.810826              3.6161232                            72.21054
## comp 8    6.590874              3.0513306                            75.26187
## comp 9    5.585551              2.5859034                            77.84778
## comp 10   4.913005              2.2745394                            80.12232
## comp 11   3.590295              1.6621736                            81.78449
## comp 12   3.437068              1.5912351                            83.37573
## comp 13   2.756064              1.2759555                            84.65168
## comp 14   2.648700              1.2262502                            85.87793
## comp 15   2.402019              1.1120457                            86.98998
## comp 16   2.272275              1.0519790                            88.04196
## comp 17   1.905618              0.8822304                            88.92419
## comp 18   1.763603              0.8164827                            89.74067
## comp 19   1.603026              0.7421419                            90.48281
## comp 20   1.332279              0.6167957                            91.09961
## comp 21   1.281669              0.5933655                            91.69297
## comp 22   1.197824              0.5545481                            92.24752
## comp 23   1.036903              0.4800478                            92.72757
```

La règle de Kaiser repose sur une idée simple. Dans une ACP normée, la somme des valeurs propres étant égale au nombre de variables, leur moyenne vaut 1. Nous considérons par conséquent qu'un axe est intéressant si sa valeur propre est supérieure 1 Nous avons ici 23 valeurs propres superieur a 1, ce qui rend l'interpretation beaucoup plus compliquer.

```
fviz_eig(res.pca, addlabels = TRUE)
```



En règle générale, le coude est très marqué lorsque nous traitons des variables fortement corrélées. Lorsqu'elles le sont faiblement ou lorsqu'il y a des blocs de variables corrélées, plutôt qu'une solution unique « évidente », nous devons faire face à plusieurs scénarios

Distribution de l'inertie

L'inertie des axes factoriels indique d'une part si les variables sont structurées et suggère d'autre part le nombre judicieux de composantes principales à étudier.

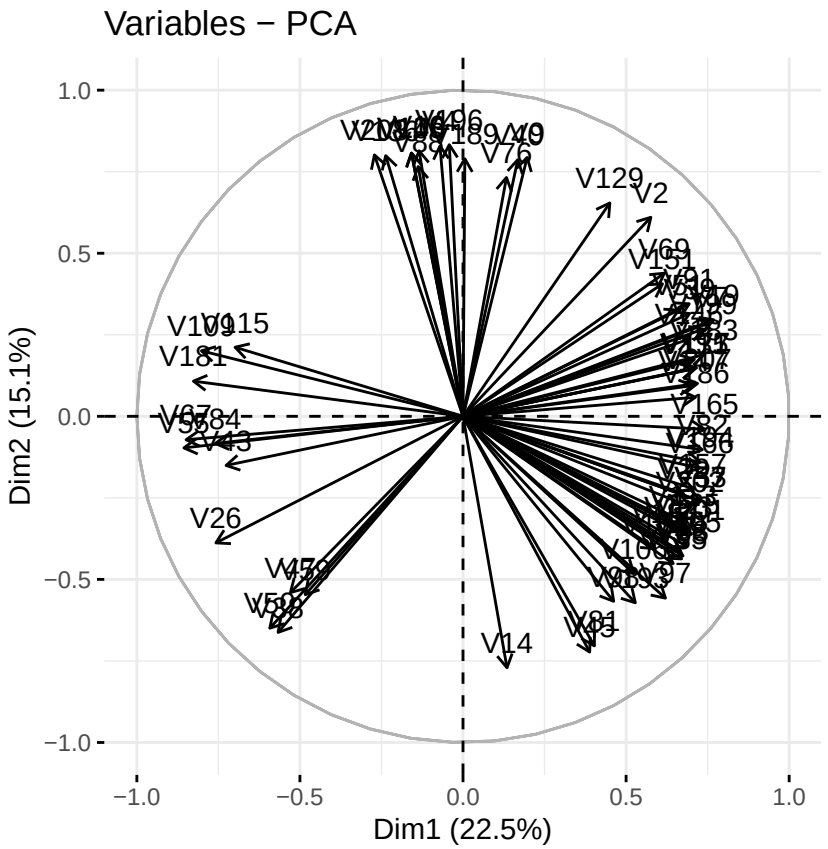
Les 2 premiers axes de l' analyse expriment 37.62% de l'inertie totale du jeu de données ; cela signifie que 37.62% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan. C'est un pourcentage relativement moyen, et le premier plan représente donc seulement une part de la variabilité contenue dans l'ensemble du jeu de données actif.

Du fait de ces observations, il serait alors probablement nécessaire de considérer également les dimensions supérieures ou égales à la troisième dans l'analyse.

Cercle de corrélation

La corrélation entre une variable et une composante principale est utilisée comme coordonnées de la variable sur la composante principale.

```
fviz_pca_var(res.pca, select.var = list(cos2 = 0.5))
```



En prenant les variables dont les contibutions sont superieur a 0.5 sur l'une des deux axes, on obtient le cercle de correlation ci-dessus. Si nous devons retenir toutes les 23 axes, et faire la meme representation, il nous sera tres difficile de l'interpreter.

Pour ce faire nous allons faire une classifiaction avec la ''methode de ward" et ''methode K-Mean" sur les coordonnées des individus pour les rasembler en different classe.

Classification par la methode K-Mean

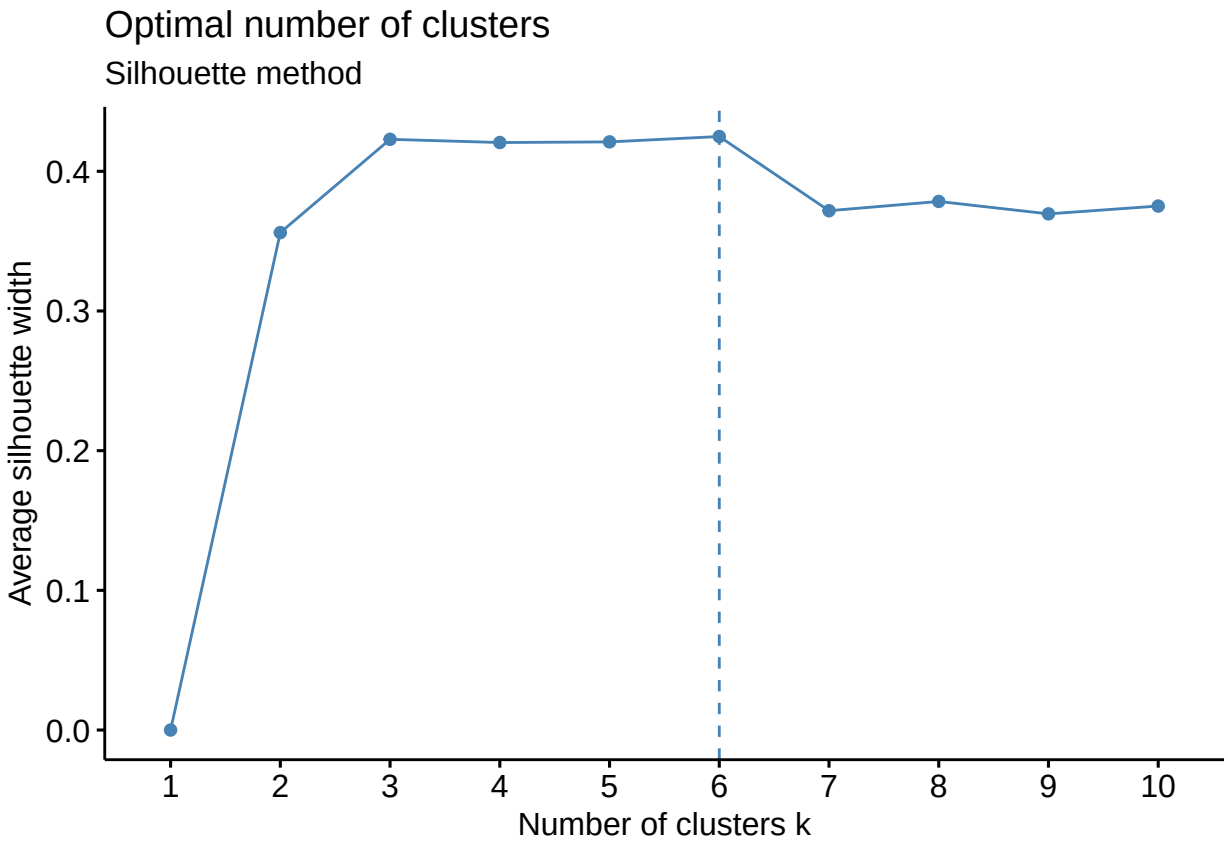
Deux principaux axes

Dans cette partie nous utilisons les deux principaux axe pour la classification des individus, sachant que les 2 premiers axes de l' analyse expriment 37.62% de l'inertie totale du jeu de données ; cela signifie que 37.62% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan

Determimination du nombre de Clusters optimaux

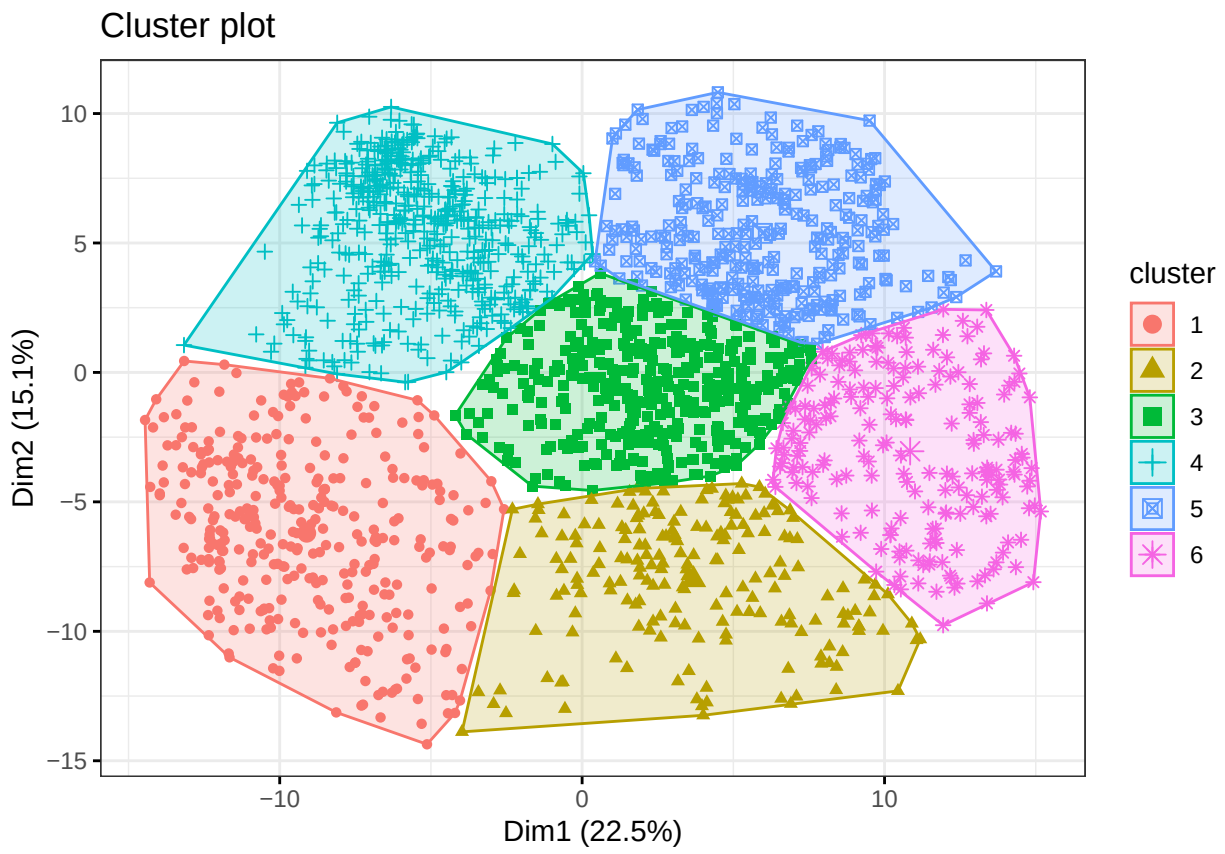
```
## K-Mean

res.pca <- PCA(mfeat.fac,ncp=2, graph = FALSE,scale.unit = TRUE)
ncp=res.pca$ind$coord
df <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



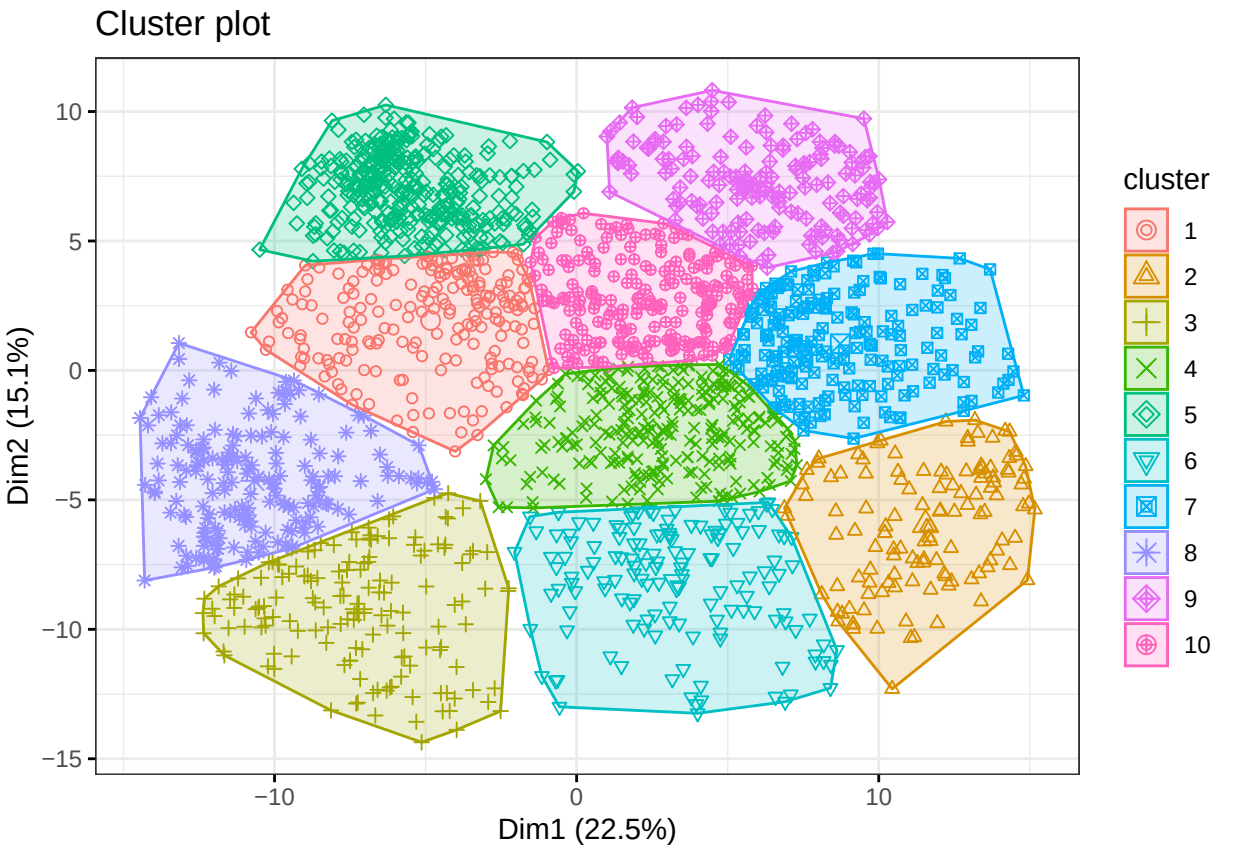
Lorsque nous prenons les deux axes principaux, on se retrouve avec 6 clusters optimaux. Cependand on peut bien voir que en prenant 10 cluster on s'eloigne pas de la performance que apportent les 6 cluster

```
km.res <- kmeans(df, 6,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.fac,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Dans la figure précedente, nous avons representés nos individus en 6 classe, vu le nombre de classe optimal.

```
km.res <- kmeans(df, 10,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.fac,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Nous resemblons nos individus sous 10 cluster, ceci est satisfessant dans car nous avons 10 classe de chiffre dans notre ensemble de données

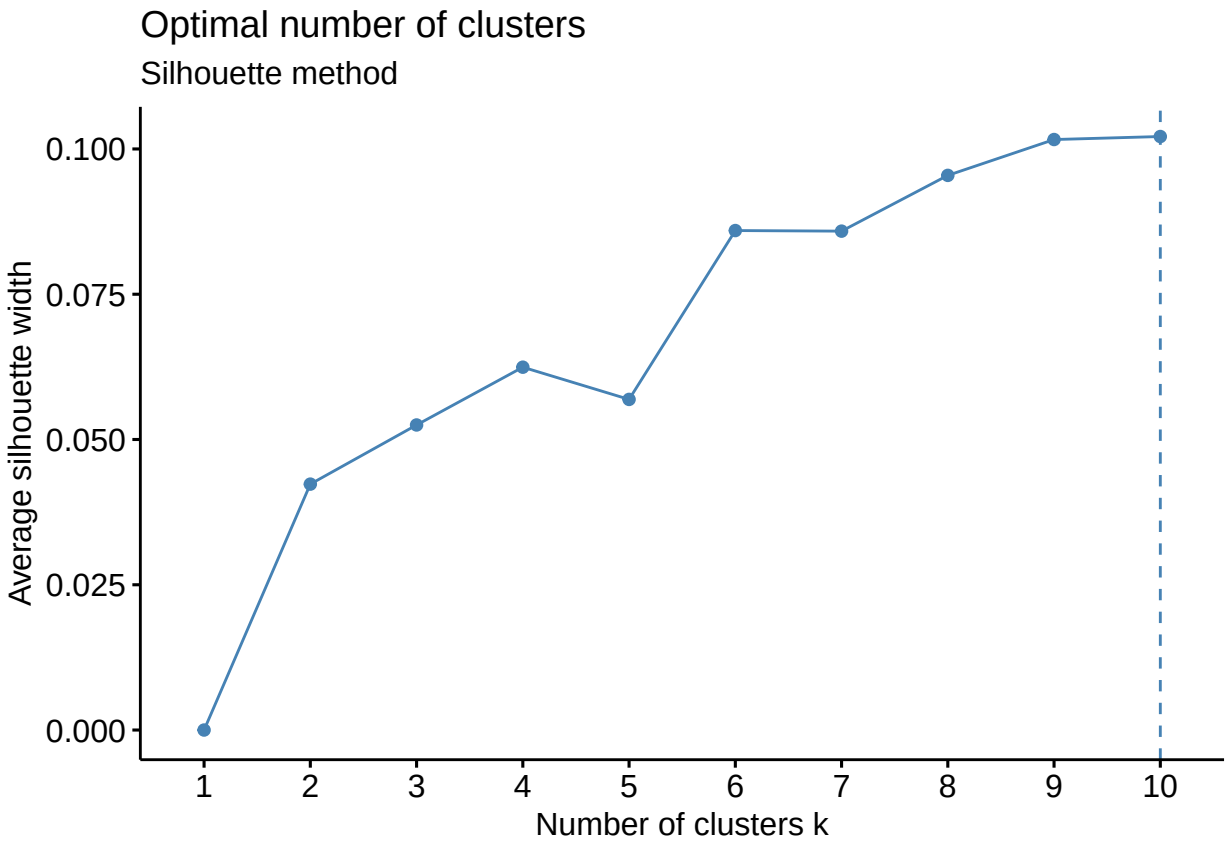
23 principaux axes

Dans cette partie nous utilisons les 23 principaux axe pour la classification des individus, sachant que les 23 premiers axes de l' analyse expriment 92,72% de l'inertie totale du jeu de données ; cela signifie que 92,72% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ces plans

Determimination du nombre

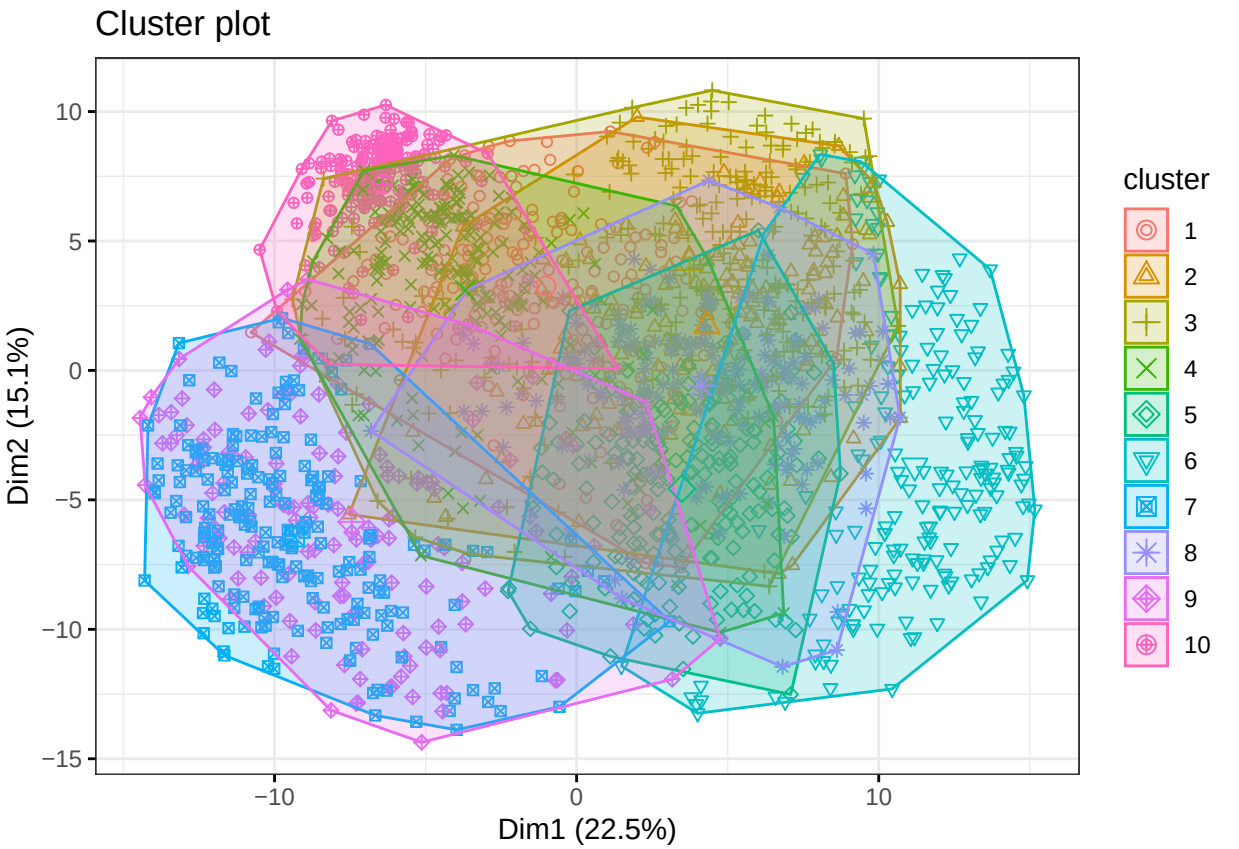
```
## K-Mean

res.pca23 <- PCA(mfeat.fac,ncp=23, graph = FALSE,scale.unit = TRUE)
ncp=res.pca23$ind$coord
df23 <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df23, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



En Prenant les 23 axes principaux, on obtient 10 clusters optimaux, ce qui corespond bien au nombre de classe initiale dans le jeux de données

```
km.res <- kmeans(df23, 10,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.fac,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Cette representation est difficile a interpreter, car nous avons 23 axes principal. Meme si elle donne plus d'informations.

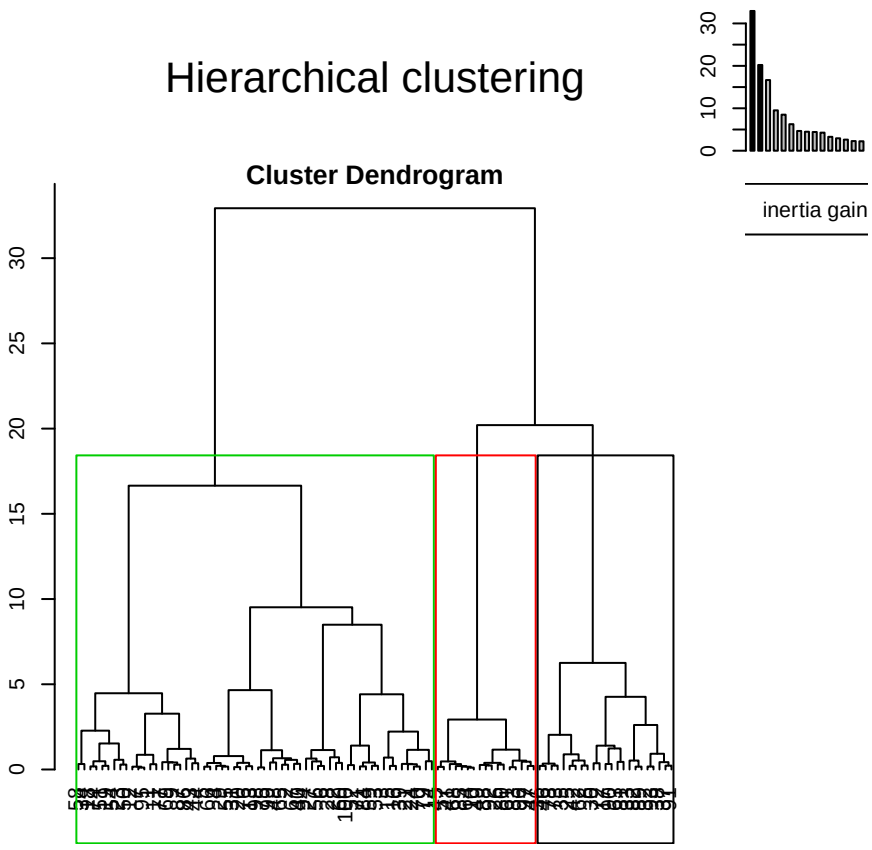
Classiffication avec l'ACP suivi de l'algorithme de Ward.

Nous utilison Dans cette partie Une ACP suivi de la classification hiérachique sur le jeu de donnée res.PCA

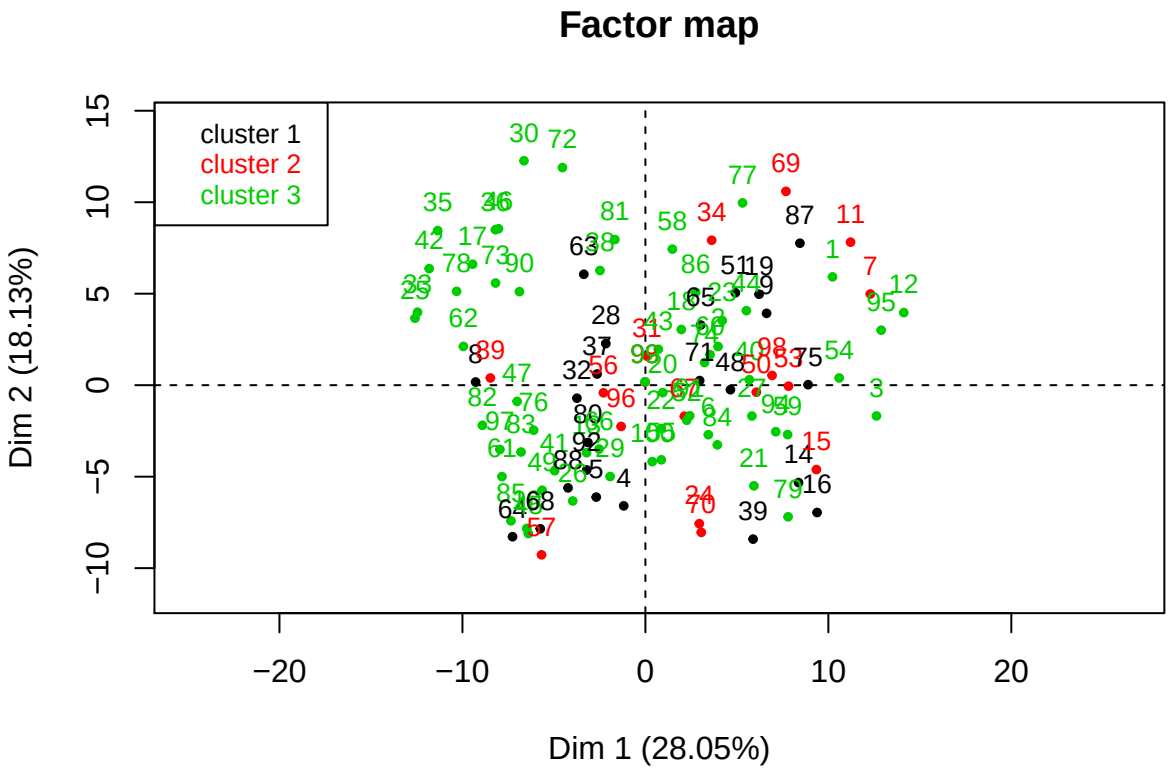
```
res.PCA= PCA(mfeat.fac,ncp=23, graph = FALSE,scale.unit = TRUE) # ACP en conservant 23 dimentions
hc=HCPC(res.PCA,kk=100,description = F,graph = F) # Classification avec prétraitement par Kmean avec kk
```

```
## Warning in HCPC(res.PCA, kk = 100, description = F, graph = F): No consolidation
## has been done after the hierarchical clustering since kk is different from Inf
## (see help for more details)
```

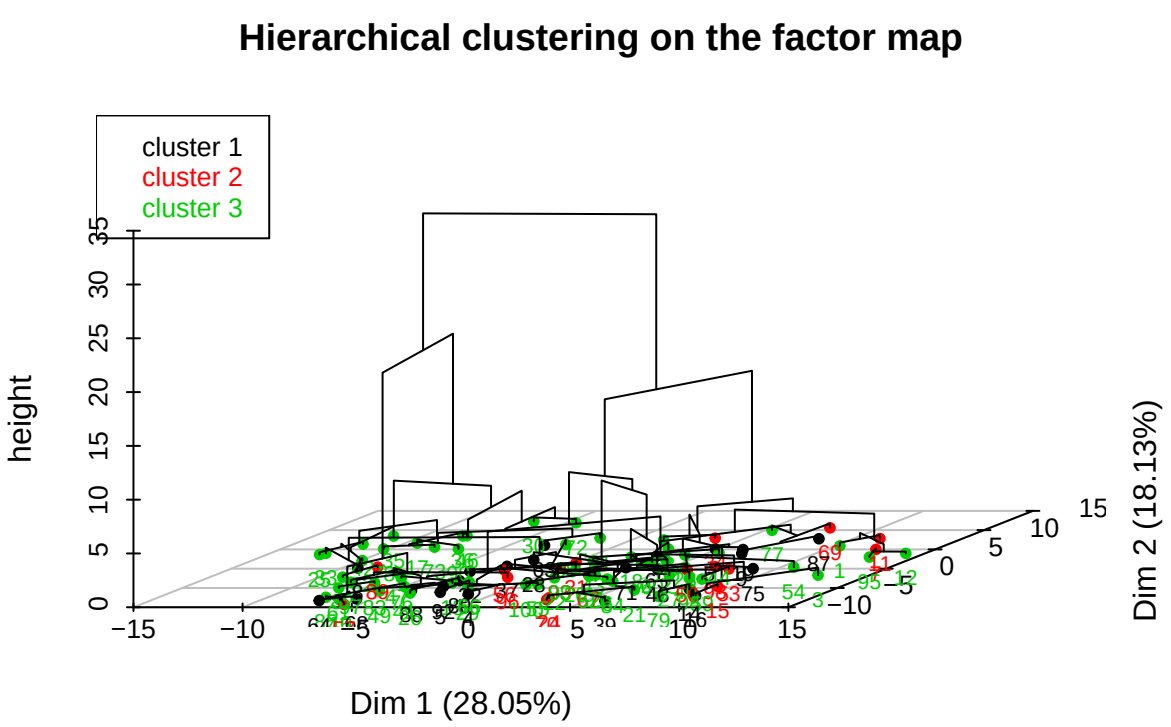
```
plot(hc,choice='tree') # Graphe de l'abre
```




```
plot(hc,choice='map',draw.tree=F) # Plan de l'ACP avec les abres
```




```
plot(hc,choice='3D.map') # Plan de l'ACP avec les abres
```




```
#catdes(hc$data.clust,ncol((hc$data.clust)))  ## caracterisation des classes
```

Classiffication avec l'algorithme de Ward.

Nous utilisons les 23 axes principales, car ayant tous des valeurs propres superieur a 1.

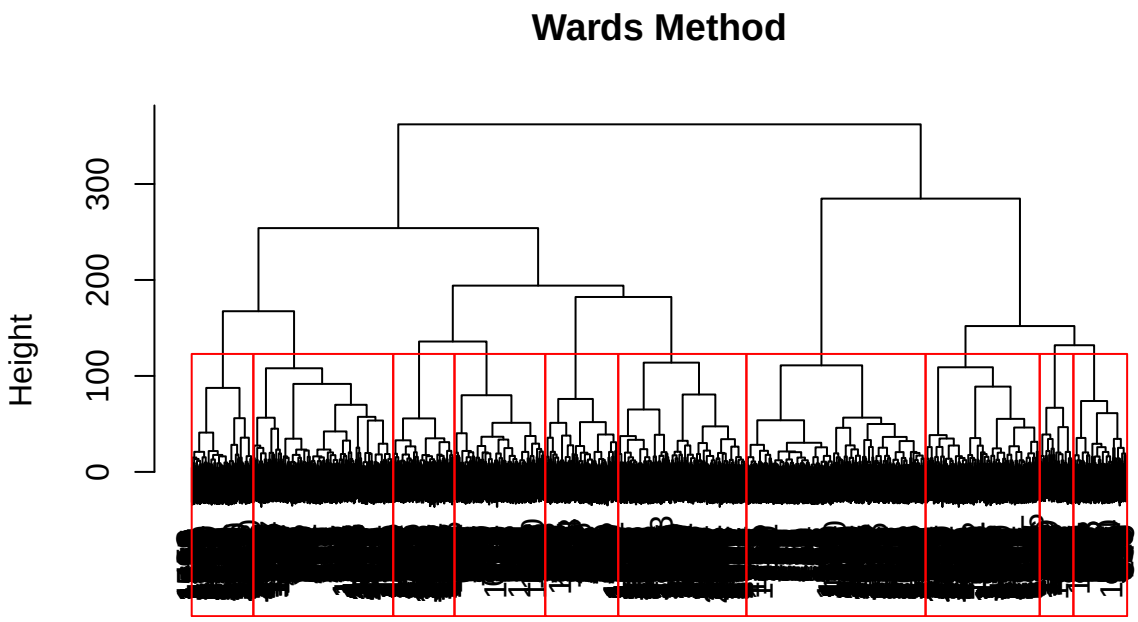
```
res.pca <- PCA(mfeat.fac,ncp=23, graph = FALSE,scale.unit = TRUE)
ncp=res.pca$ind$coord
```

Dans cette partie nous utilisons les cordonnées des individu de l'ACP pour faire la classification avec l'algorithne de ward

```
## standarisation des donnees
new_data_scaled <- ncp
new_data_scaled.dist <- dist(new_data_scaled)      # Calcul de distance
new_data_scaled.distT <- dist(t(new_data_scaled)) # calcul de distance
```

```
# effectuer un regroupement hiérarchique des observations (variables) en utilisant la méthode de Ward
ward = hclust(new_data_scaled.distT,method='ward.D2')
wardT = hclust(new_data_scaled.dist,method='ward.D2')
```

```
plot(wardT, main = 'Wards Method',xlab =" ", sub ="")
rect.hclust(wardT, k =10, border = 'red') ## selecting three clusters
```



Ce cluster nous permet de voir 10 classe.

Conclusion pour la donnée mfeat.fact

Dans cette partie de notre etude nous avons pu momtrer que nos 2000 ligne pouvais etre regrouper en 10 classe a partir des methode de l'ACP en reduissant la dimension a un nombre acceptable pour ensuite faire de la Classification.

Donnée mfeat-fou

Nous allons commencer avec les donnees mfeat.fou. Nous n'avons pas assez d'information sur les données

```
#head(mfeat.fac, n=5)
```

```
# class(mfeat.fac)        # checking your data class

# dim(mfeat.fac)          # getting the dimension of your data

# names(mfeat.fac)        # getting the column names

# str(mfeat.fac)          # checking data structure

# anyNA(mfeat.fac)        # checking for missing values

# summary(mfeat.fac)      # summary of the data

# View(mfeat.fac)         # view your data
```

Pour la Description du jeu de données, ou identification de groupes d'individus et liens entre variables nous allons utiliser l'ACP pour décrire ce jeu de données comportant de nombreux individus et variables quantitatives. L'analyse doit permettre d'extraire l'information pertinente et la synthétiser sous forme de composantes principales, nouveaux axes pour décrire le jeu de données.

La fonction PCA()

Nous utilisons la fonction 'PCA()' de 'FactoMineR', elle centre et réduit les variables avant de réaliser l'ACP. Cette étape est importante afin que toutes les variables aient le même poids dans la construction des plans de l'ACP.

```
res.pca02 <- PCA(mfeat.fou,ncp=10, graph = FALSE)
print(res.pca02)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 2000 individuals, described by 76 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Valeurs propres

Pour le choix du nombre de d'axe principale, nous pouvons utiliser la regle de kaiser et la regle de Coude. Nous allons donc afficher le nombre d'axe ou les valeurs propres son superieur a 1

Comme décrit dans les sections précédentes, les valeurs propres mesurent la quantité de variance expliquée par chaque axe principal. Les valeurs propres sont grandes pour les premiers axes et petits pour les axes suivants. Autrement dit, les premiers axes correspondent aux directions portant la quantité maximale de variation contenue dans le jeu de données.

Nous examinons les valeurs propres pour déterminer le nombre de composantes principales à prendre en considération en fonction de la regle de kaiser et de coude

Nous visualisons le tableau seulement pour les lignes dont les valeurs propres sont supperieur a 1

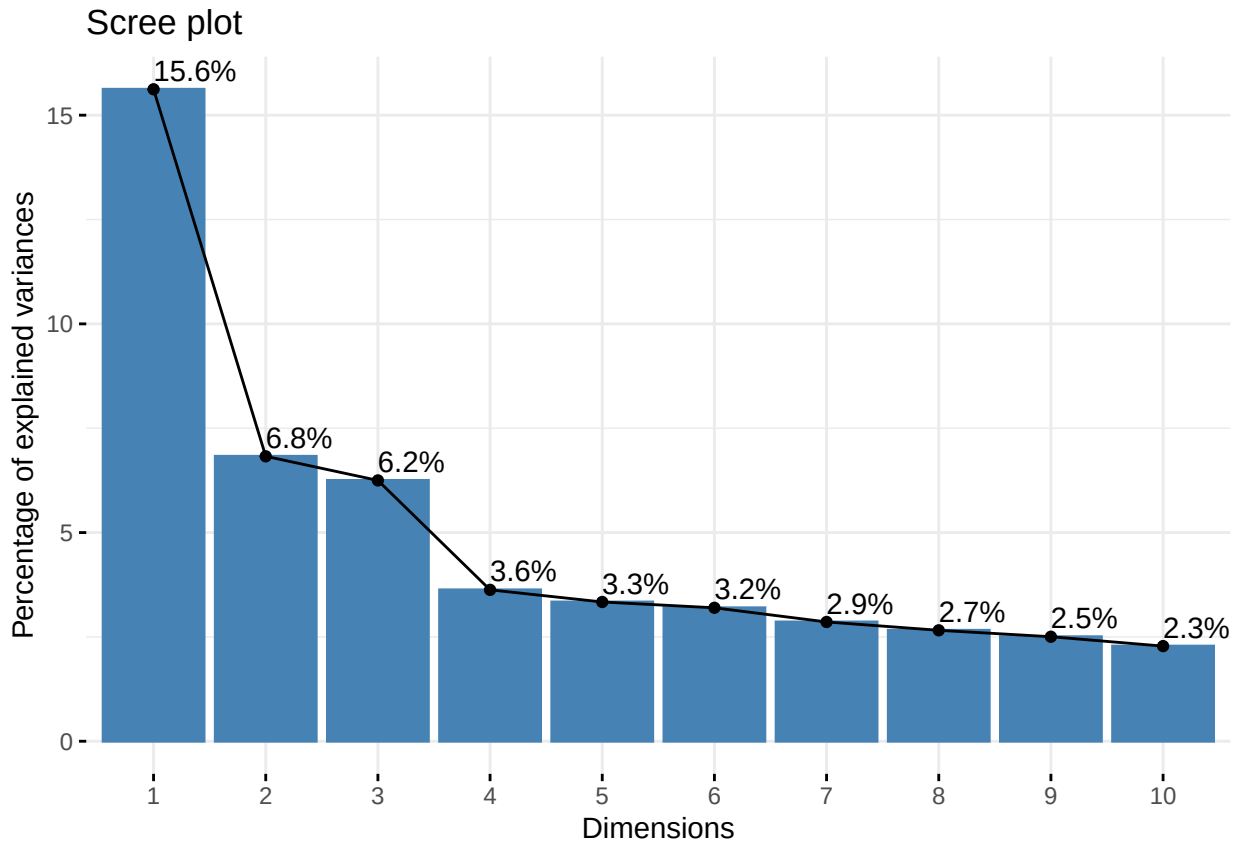
```
dt=as.data.frame(res.pca02$eig)
dt[dt$eigenvalue >= 1, ]
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   11.871456              15.620337                           15.62034
## comp 2    5.188408               6.826853                           22.44719
## comp 3    4.749544               6.249401                           28.69659
```

```
## comp 4    2.756648          3.627168          32.32376
## comp 5    2.535661          3.336397          35.66015
## comp 6    2.430012          3.197384          38.85754
## comp 7    2.171937          2.857812          41.71535
## comp 8    2.020155          2.658099          44.37345
## comp 9    1.903114          2.504097          46.87755
## comp 10   1.732323          2.279372          49.15692
## comp 11   1.629823          2.144504          51.30142
## comp 12   1.589664          2.091663          53.39309
## comp 13   1.496645          1.969269          55.36236
## comp 14   1.444788          1.901036          57.26339
## comp 15   1.359160          1.788369          59.05176
## comp 16   1.333663          1.754820          60.80658
## comp 17   1.203209          1.583170          62.38975
## comp 18   1.119473          1.472991          63.86274
## comp 19   1.068050          1.405329          65.26807
## comp 20   1.050737          1.382549          66.65062
```

La règle de Kaiser repose sur une idée simple. Dans une ACP normée, la somme des valeurs propres étant égale au nombre de variables, leur moyenne vaut 1. Nous considérons par conséquent qu'un axe est intéressant si sa valeur propre est supérieure 1 Nous avons ici 20 valeurs propres superieur a 1, ce qui rend l'interpretation beaucoup plus compliquer.

```
fviz_eig(res.pca02, addlabels = TRUE)
```



En règle générale, le coude est très marqué lorsque nous traitons des variables fortement corrélées. Lorsqu'elles le sont faiblement ou lorsqu'il y a des blocs de variables corrélées, plutôt qu'une solution unique « évidente »,

nous devons faire face à plusieurs scénarios

Ici, on observe une cassure au niveau du deuxieme et du quatrième valeurs propre, mais il preferable de prendre 04 valeurs propres

Distribution de l'inertie

L'inertie des axes factoriels indique d'une part si les variables sont structurées et suggère d'autre part le nombre judicieux de composantes principales à étudier.

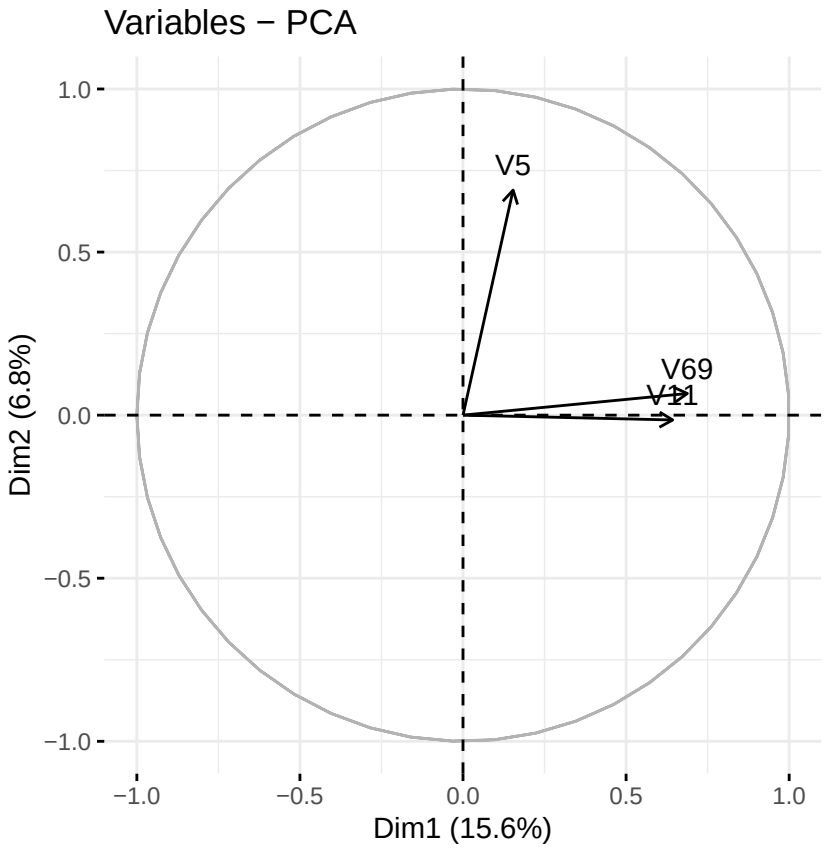
Les 2 premiers axes de l' analyse expriment 22,44% de l'inertie totale du jeu de données ; cela signifie que 22,44% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan. C'est un pourcentage relativement faible, et le premier plan représente donc seulement une part de la variabilité contenue dans l'ensemble du jeu de données actif.

Du fait de ces observations, il serait alors probablement nécessaire de considérer également les dimensions supérieures ou égales à la troisième dans l'analyse et aussi les 20 axes comme le suggère la loi de Kaiser

Cercle de corrélation

La corrélation entre une variable et une composante principale (PC) est utilisée comme coordonnées de la variable sur la composante principale.

```
fviz_pca_var(res.pca02, select.var = list(cos2 = 0.4))
```



En prenant une contribution superieur a 0.5, on obtient aucune variable corrélé avec les axes principaux

En prenant les variables dont les contibutions sont superieur a 0.4 sur l'une des deux axes, on obtient le cercle de correlation ci-dessus. Si nous devons retenir toutes les 23 axes, et faire la meme representation, il nous

sera tres difficile de l'interpreter.

Pour ce faire nous allons faire une classifiaction avec la ''methode de ward" et ''methode K-Mean" sur les coordonnées des individus pour les rasembler en different classe.

Classification par la methode K-Mean

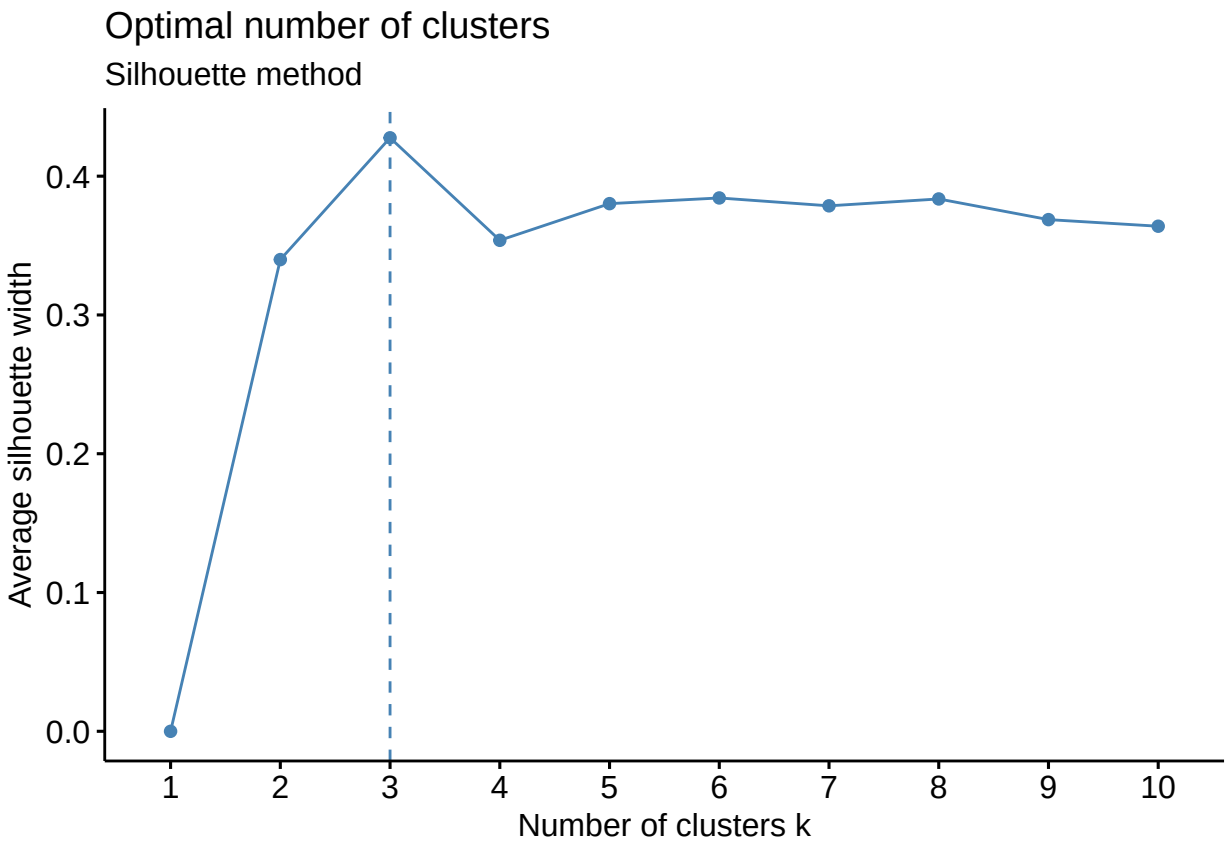
Deux principaux axes

Dans cette partie nous utilisons les deux principaux axe pour la classification des individus, sachant que les 2 premiers axes de l' analyse expriment 22.44% de l'inertie totale du jeu de données ; cela signifie que 22.44% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan

Determimination du nombre

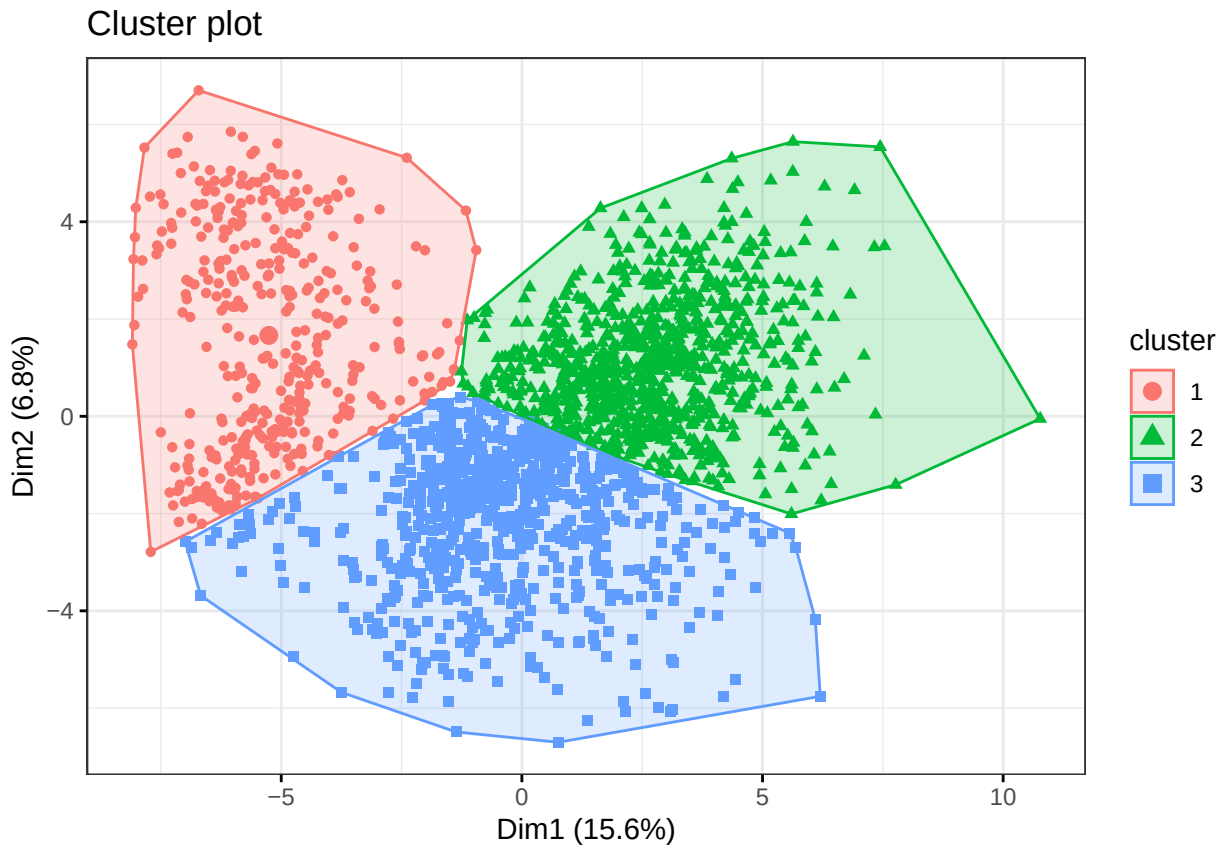
```
## K-Mean

res.pca02 <- PCA(mfeat.fou,ncp=2, graph = FALSE,scale.unit = TRUE)
ncp=res.pca02$ind$coord
df <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



Lorsque nous prenons les deux axes principaux, on se retrouve avec 3 clusters optimal. Cependand nous pouvons envisager de regarder egalement 10 cluster.

```
km.res <- kmeans(df, 3,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.fou,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```




```
km.res <- kmeans(df, 10,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.fou,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```

Nous resemblons nos individus sous 10 cluster, ceci est satisfessant dans car nous avons 10 classe de chiffre dans notre ensemble de données

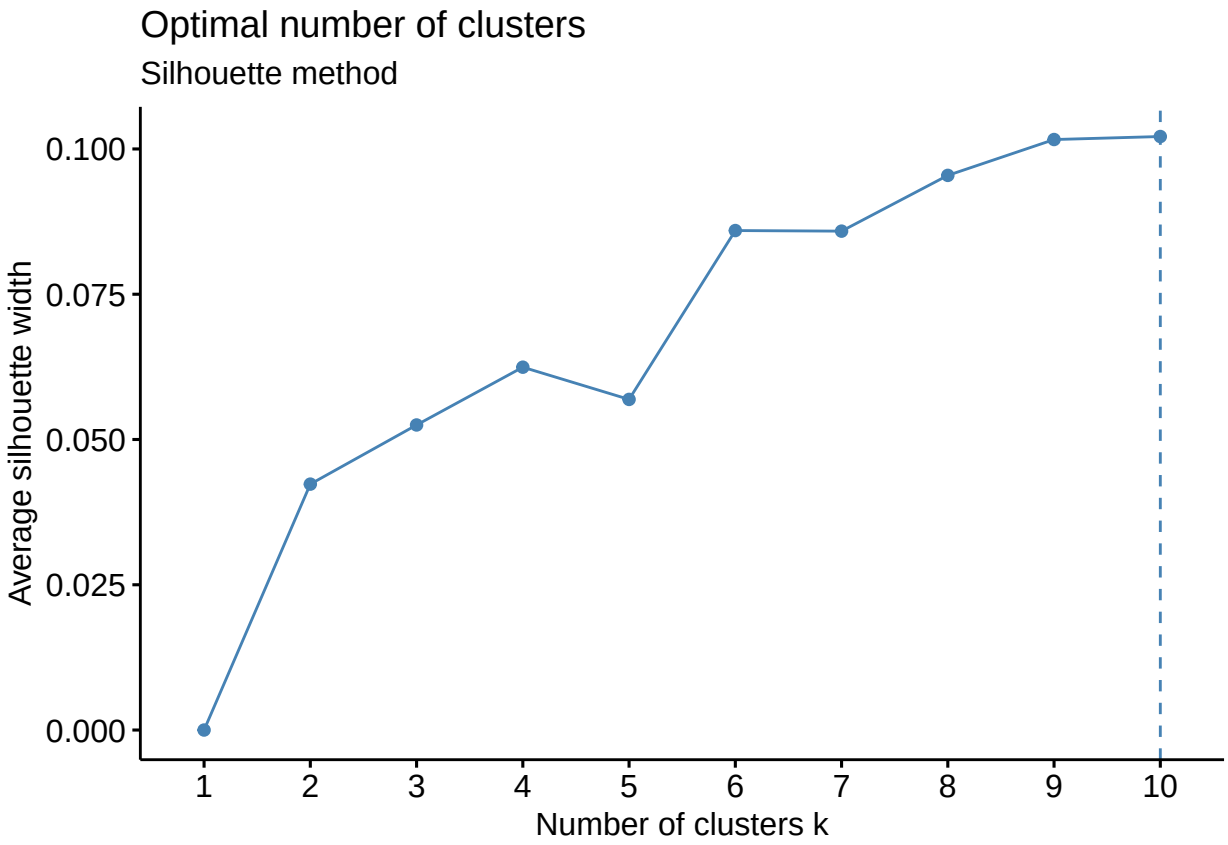
20 principaux axes

Dans cette partie nous utilisons les deux principaux axe pour la classification des individus, sachant que les 23 premiers axes de l' analyse expriment 66.60% de l'inertie totale du jeu de données ; cela signifie que 66,60% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ces plans

Determimination du nombre

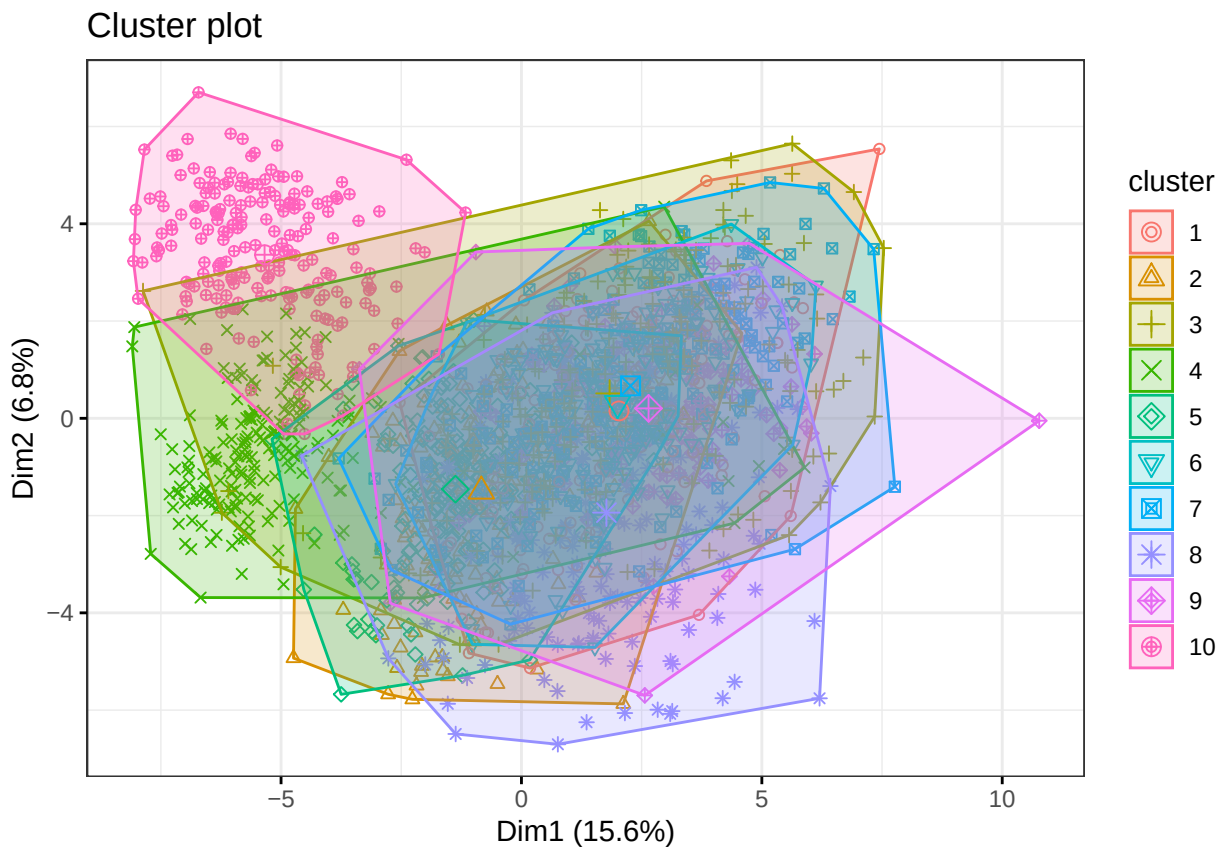
```
## K-Mean

res.pca20 <- PCA(mfeat.fou,ncp=20, graph = FALSE,scale.unit = TRUE)
ncp=res.pca23$ind$coord
df20 <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df20, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



En Prenant les 23 axes principaux, on obtient 10 clusters optimaux, ce qui corespond bien au nombre de classe initiale dans le jeux de données

```
km.res <- kmeans(df20, 10,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.fou,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Cette representation est difficile a interpreter, car nous avons 20 axes principal . Meme si elle donne plus d'informations.

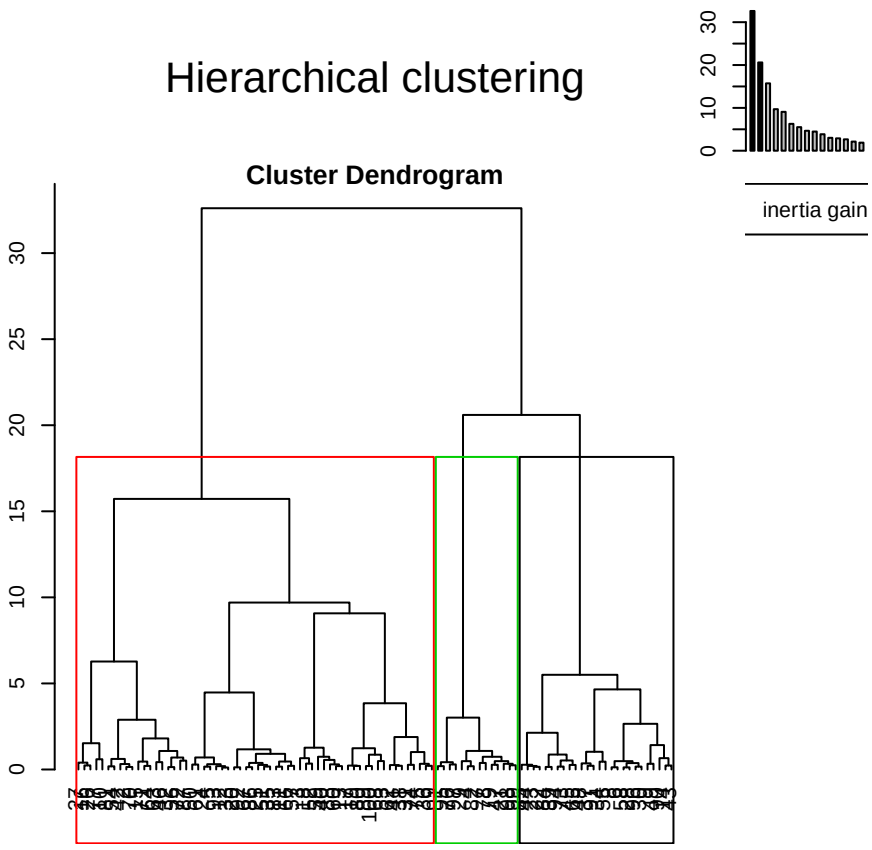
Classiffication avec l'ACP suivi de l'algorithme de Ward.

Nous utilison Dans cette partie Une ACP suivi de la classification hiérachique sur le jeu de donnée res.PCA

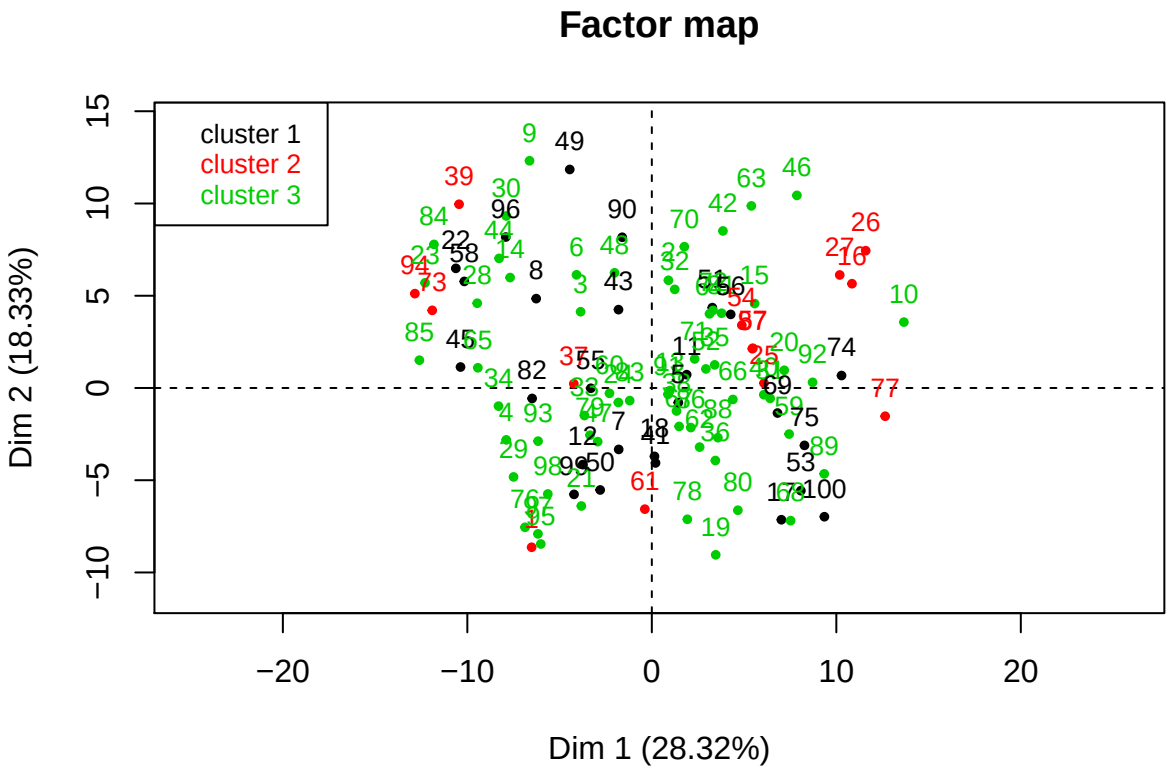
```
res.PCA= PCA(mfeat.fac,ncp=20, graph = FALSE,scale.unit = TRUE) # ACP en conservant 23 dimentions
hc=HCPC(res.PCA,kk=100,description = F,graph = F) # Classification avec prétraitement par Kmean avec kk
```

```
## Warning in HCPC(res.PCA, kk = 100, description = F, graph = F): No consolidation
## has been done after the hierarchical clustering since kk is different from Inf
## (see help for more details)
```

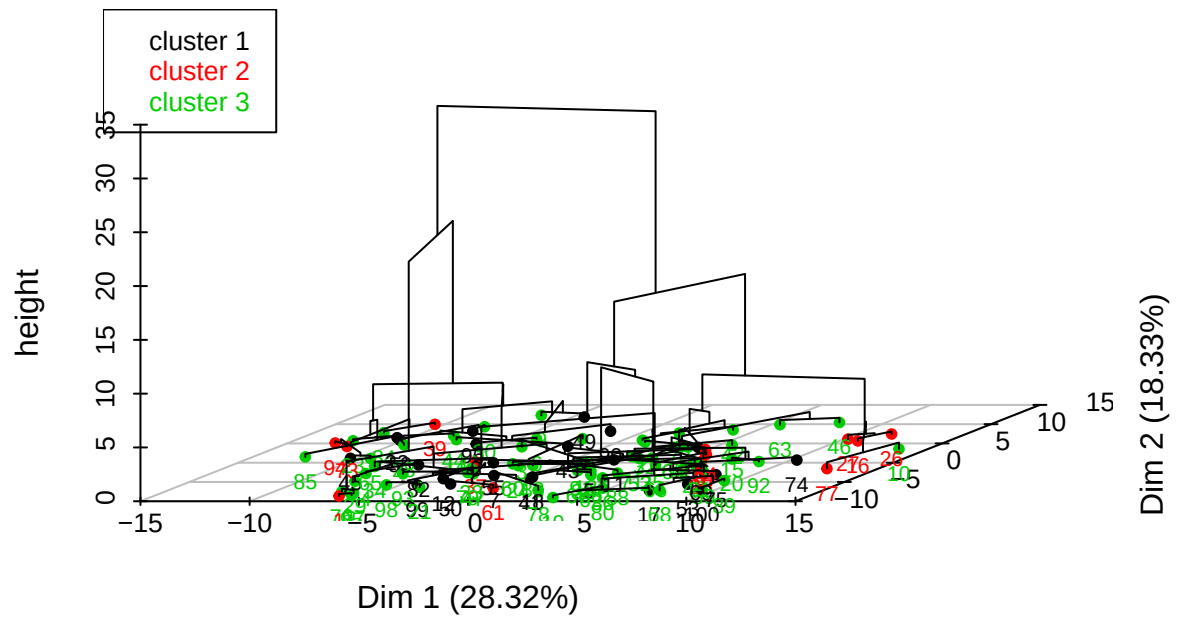
```
plot(hc,choice='tree') # Graphe de l'abre
```




```
plot(hc,choice='map',draw.tree=F) # Plan de l'ACP avec les abres
```




```
plot(hc,choice='3D.map') # Plan de l'ACP avec les abres
```




```
#catdes(hc$data.clust,ncol((hc$data.clust)))  ## caracterisation des classes
```

Nous obtenons bien les trois classe precedentes, avec une bonne probabilité pour chaque variable de se trouver dans chacune des classe.

Classiffication avec l'algorithme de Ward.

Nous utilisons les 20 axes principales, car ayant tous des valeurs propres superieur a 1.

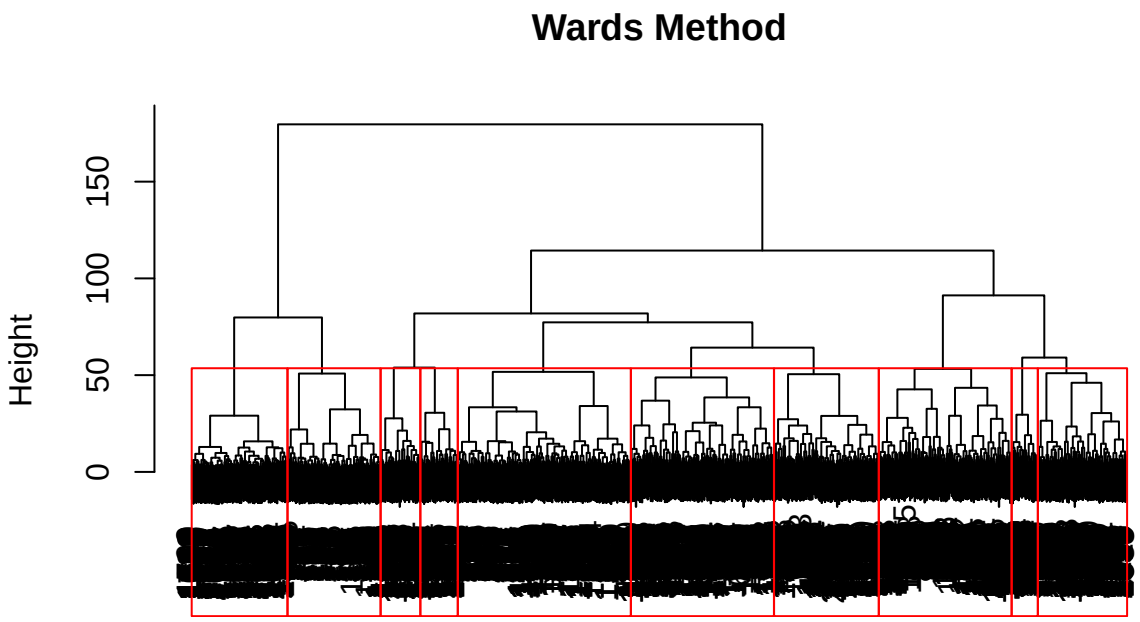
```
res.pca <- PCA(mfeat.fou,ncp=20, graph = FALSE,scale.unit = TRUE)
ncp=res.pca$ind$coord
```

Dans cette partie nous utilisons les cordonnées des individu de l'ACP pour faire la classification avec l'algorithne de ward

```
## standarisation des donnees
new_data_scaled <- ncp
new_data_scaled.dist <- dist(new_data_scaled)     # Calcul de distance
new_data_scaled.distT <- dist(t(new_data_scaled)) # calcul de distance
```

```
# effectuer un regroupement hiérarchique des observations (variables) en utilisant la méthode de Ward
ward = hclust(new_data_scaled.distT,method='ward.D2')
wardT = hclust(new_data_scaled.dist,method='ward.D2')
```

```
plot(wardT, main = 'Wards Method',xlab =" ", sub ="")
rect.hclust(wardT, k =10, border = 'red') ## selecting three clusters
```



Ce cluster nous permet de voir 10 classe.

Conclusion pour la donnée mfeat.fou

Dans cette partie de notre etude nous avons pu momtrer que nos 2000 ligne pouvais etre regrouper en 10 classe a partir des methode de l'ACP en reduissant la dimension a un nombre acceptable pour ensuite faire de la Classification.

Donnée mfeat.Kar

Nous allons commencer avec les donnees mfeat.kar. Nous n'avons pas assez d'information sur les données

```
#head(mfeat.fac, n=5)
```

```
# class(mfeat.kar)        # checking your data class

# dim(mfeat.kar)          # getting the dimension of your data

# names(mfeat.kar)        # getting the column names

# str(mfeat.kar)          # checking data structure

# anyNA(mfeat.kar)        # checking for missing values

# summary(mfeat.kar)      # summary of the data

# View(mfeat.kar)         # view your data
```

La fonction PCA()

Nous utilisons la fonction 'PCA()' de 'FactoMineR', elle centre et réduit les variables avant de réaliser l'ACP. Cette étape est importante afin que toutes les variables aient le même poids dans la construction des plans de l'ACP.

```
res.pca03 <- PCA(mfeat.kar,ncp=10, graph = FALSE)
print(res.pca03)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 2000 individuals, described by 64 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Valeurs propres

Pour le choix du nombre de d'axe principale, nous pouvons utiliser la regle de kaiser et la regle de Coude. Nous allons donc afficher le nombre d'axe ou les valeurs propres sont superieur a 1

Comme décrit, les valeurs propres mesurent la quantité de variance expliquée par chaque axe principal. Les valeurs propres sont grandes pour les premiers axes et petits pour les axes suivants. Autrement dit, les premiers axes correspondent aux directions portant la quantité maximale de variation contenue dans le jeu de données.

Nous examinons les valeurs propres pour déterminer le nombre de composantes principales à prendre en considération en fonction de la regle de kaiser et de coude

Nous visualisons le tableau seulement pour les lignes dont les valeurs propres sont supperieur a 1

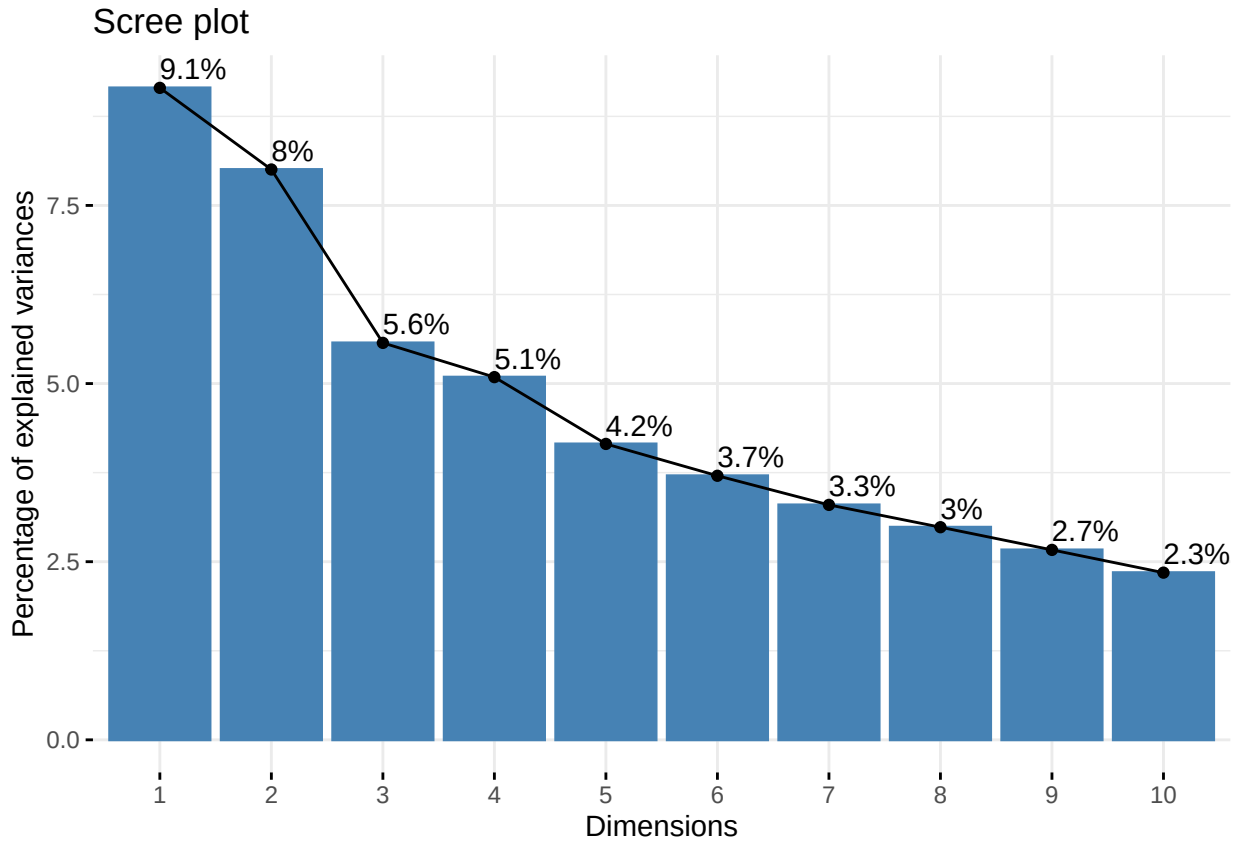
```
dt=as.data.frame(res.pca03$eig)
dt[dt$eigenvalue >= 1, ]
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1    5.855979               9.149967                           9.149967
## comp 2    5.122755               8.004305                          17.154272
## comp 3    3.565545               5.571164                          22.725436
## comp 4    3.257980               5.090594                          27.816031
## comp 5    2.657952               4.153051                          31.969081
## comp 6    2.372063               3.706348                          35.675429
## comp 7    2.110405               3.297507                          38.972936
## comp 8    1.910001               2.984376                          41.957312
```

```
## comp 9    1.706146          2.665853          44.623165
## comp 10   1.501758          2.346496          46.969661
## comp 11   1.482327          2.316136          49.285797
## comp 12   1.437686          2.246384          51.532181
## comp 13   1.378451          2.153830          53.686010
## comp 14   1.317281          2.058252          55.744262
## comp 15   1.255113          1.961114          57.705376
## comp 16   1.164983          1.820287          59.525662
## comp 17   1.096309          1.712983          61.238645
## comp 18   1.068237          1.669121          62.907765
## comp 19   1.028558          1.607122          64.514888
## comp 20   1.003582          1.568097          66.082985
```

La règle de Kaiser repose sur une idée simple. Dans une ACP normée, la somme des valeurs propres étant égale au nombre de variables, leur moyenne vaut 1. Nous considérons par conséquent qu'un axe est intéressant si sa valeur propre est supérieure 1 Nous avons ici 20 valeurs propres superieur a 1, ce qui rend l'interpretation beaucoup plus compliquer.

```
fviz_eig(res.pca03, addlabels = TRUE)
```



En règle générale, le coude est très marqué lorsque nous traitons des variables fortement corrélées. Lorsqu'elles le sont faiblement ou lorsqu'il y a des blocs de variables corrélées, plutôt qu'une solution unique « évidente », nous devons faire face à plusieurs scénarios

Distribution de l'inertie

L'inertie des axes factoriels indique d'une part si les variables sont structurées et suggère d'autre part le nombre judicieux de composantes principales à étudier.

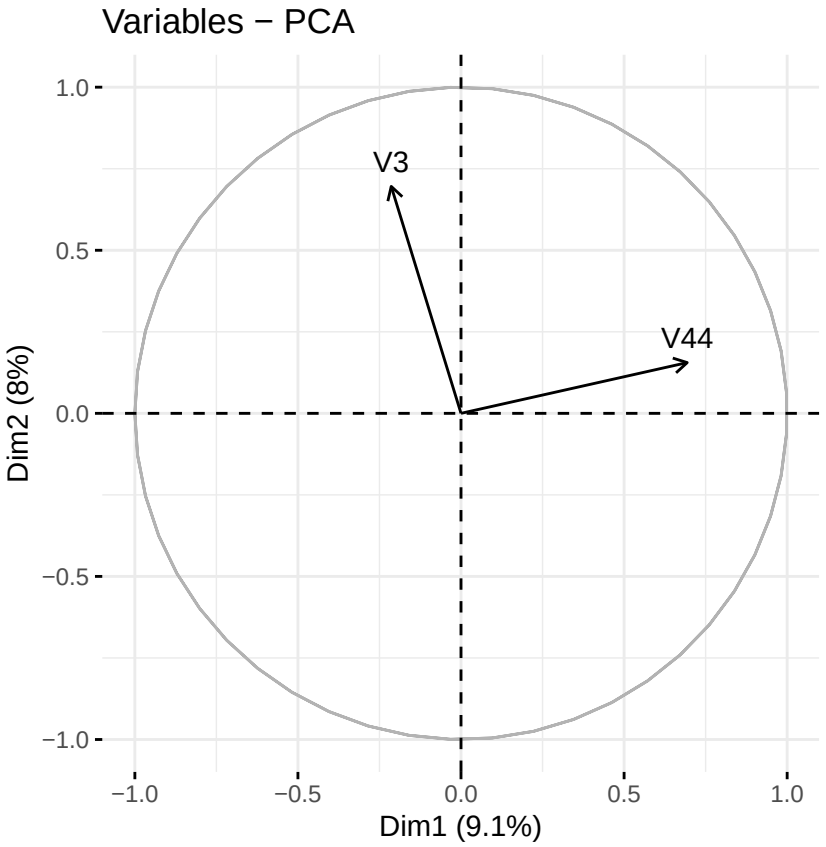
Les 2 premiers axes de l' analyse expriment 17,15% de l'inertie totale du jeu de données ; cela signifie que 17,15% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan. C'est un pourcentage relativement faible, et le premier plan représente donc seulement une part de la variabilité contenue dans l'ensemble du jeu de données actif.

Du fait de ces observations, il serait alors probablement nécessaire de considérer également les dimensions supérieures ou égales à la troisième dans l'analyse.

Cercle de corrélation

La corrélation entre une variable et une composante principale est utilisée comme coordonnées de la variable sur la composante principale.

```
fviz_pca_var(res.pca03, select.var = list(cos2 = 0.5))
```



En prenant les variables dont les contibutions sont superieur a 0.5 sur l'une des deux axes, on obtient le cercle de correlation ci-dessus. Si nous devons retenir toutes les 20 axes, et faire la meme representation, il nous sera tres difficile de l'interpreter.

Pour ce faire nous allons faire une classifiaction avec la ''methode de ward" et ''methode K-Mean" sur les coordonnées des individus pour les rasembler en different classe.

Classification par la methode K-Mean

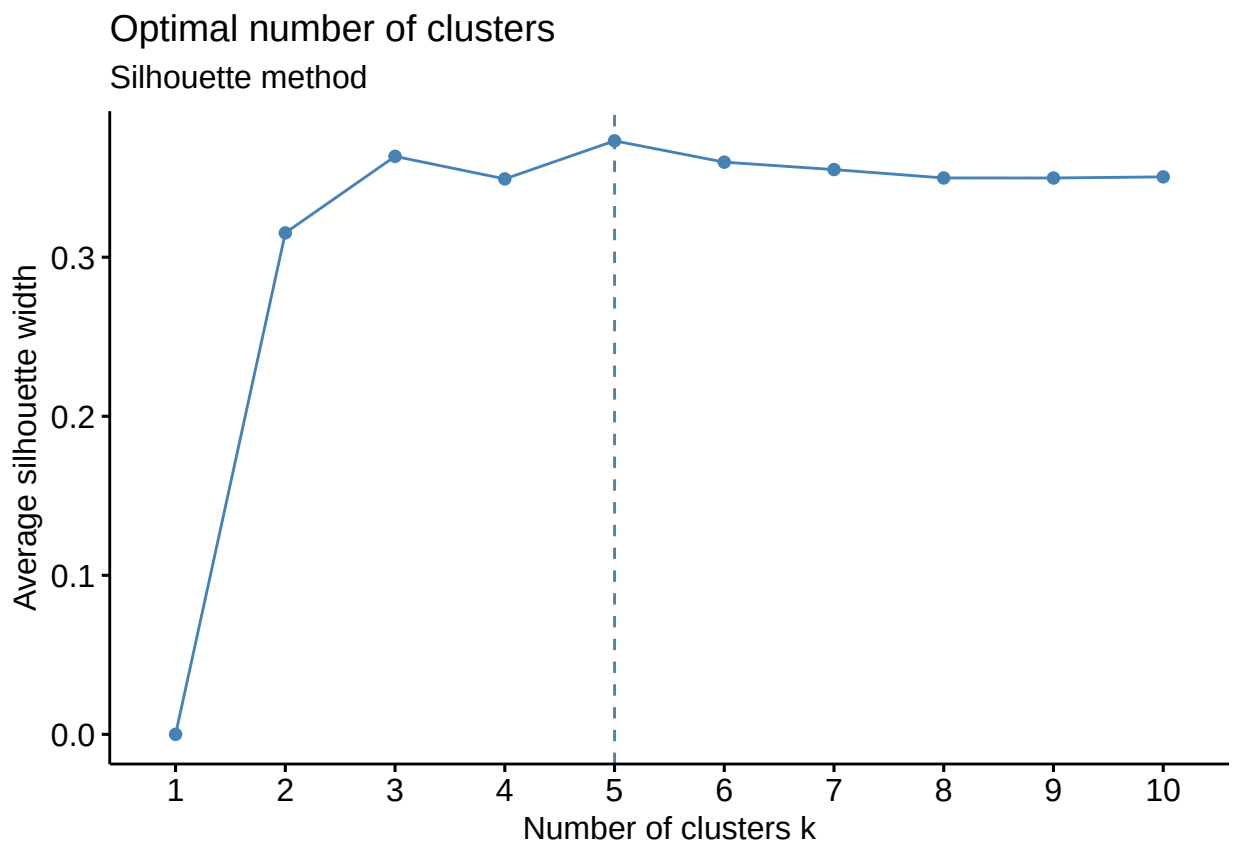
Deux principaux axes

Dans cette partie nous utilisons les deux principaux axe pour la classification des individus, sachant que les 2 premiers axes de l' analyse expriment 17.15% de l'inertie totale du jeu de données ; cela signifie que 17.15% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan

Determimination du nombre de Clusters optimaux

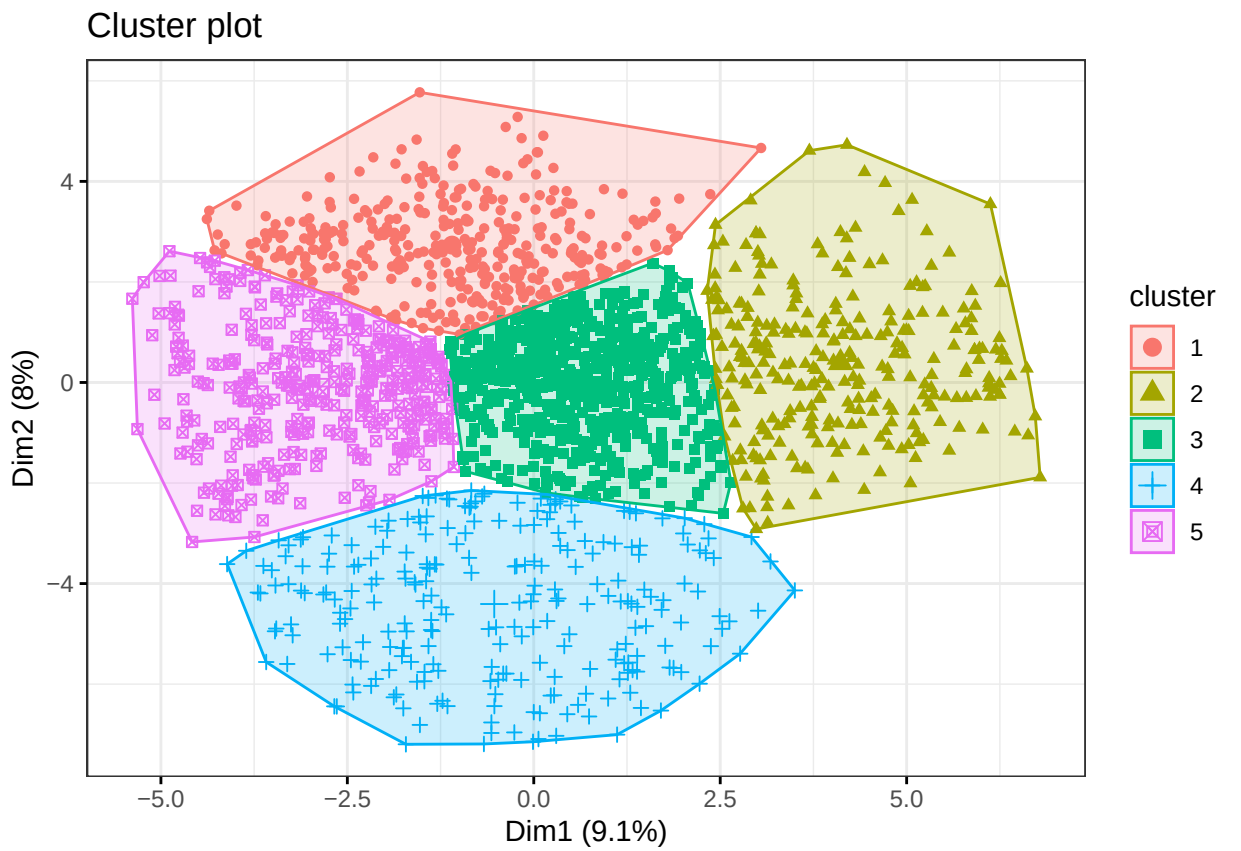
```
## K-Mean

res.pca03 <- PCA(mfeat.kar,ncp=2, graph = FALSE,scale.unit = TRUE)
ncp=res.pca03$ind$coord
df <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df, kmeans,iter.max = 1000,  nstart = 50,method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



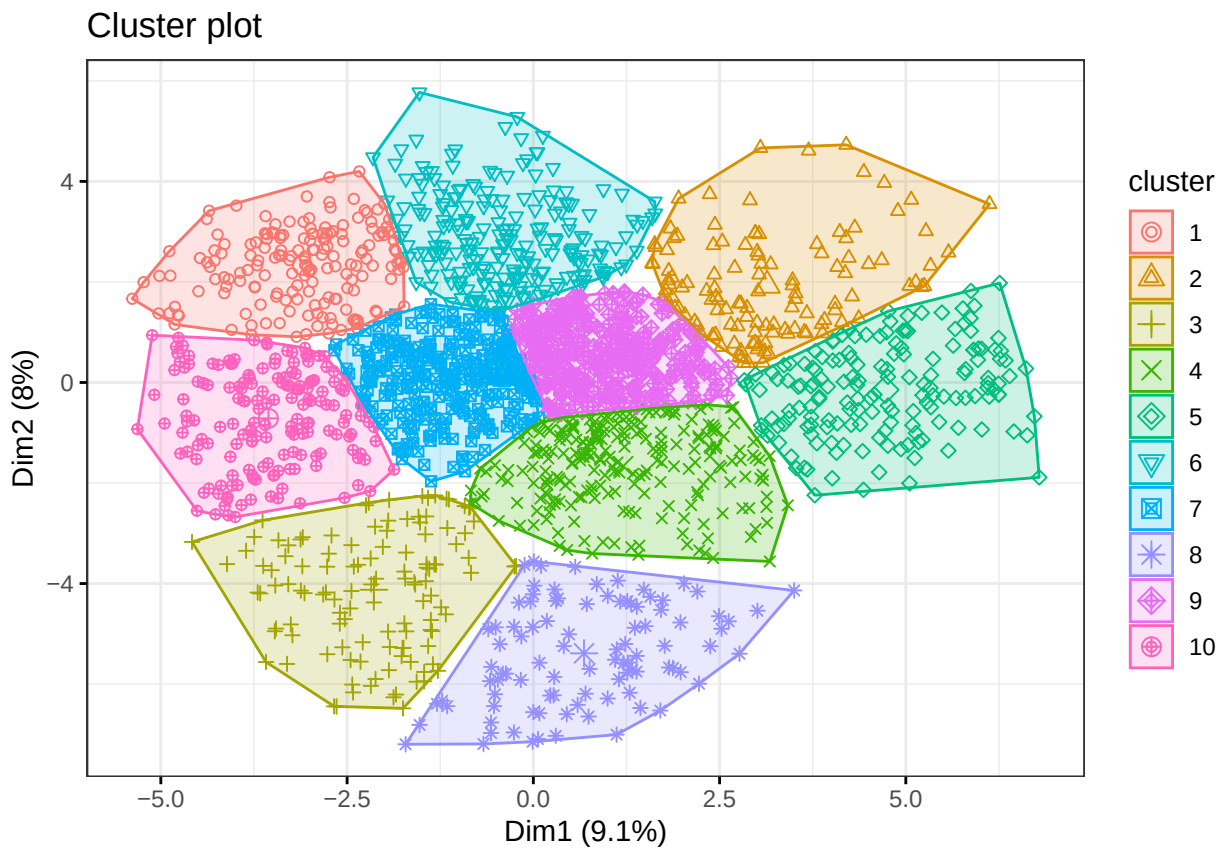
Lorsque nous prenons les deux axes principaux, on se retrouve avec 5 clusters optimaux.

```
km.res <- kmeans(df, 5,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.kar,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Dans la figure précedente, nous avons representés nos individus en 6 classe, vu le nombre de classe optimal.

```
km.res <- kmeans(df, 10,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.kar,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Nous resemblons nos individus sous 10 cluster, ceci est satisfessant dans car nous avons 10 classe de chiffre dans notre ensemble de données

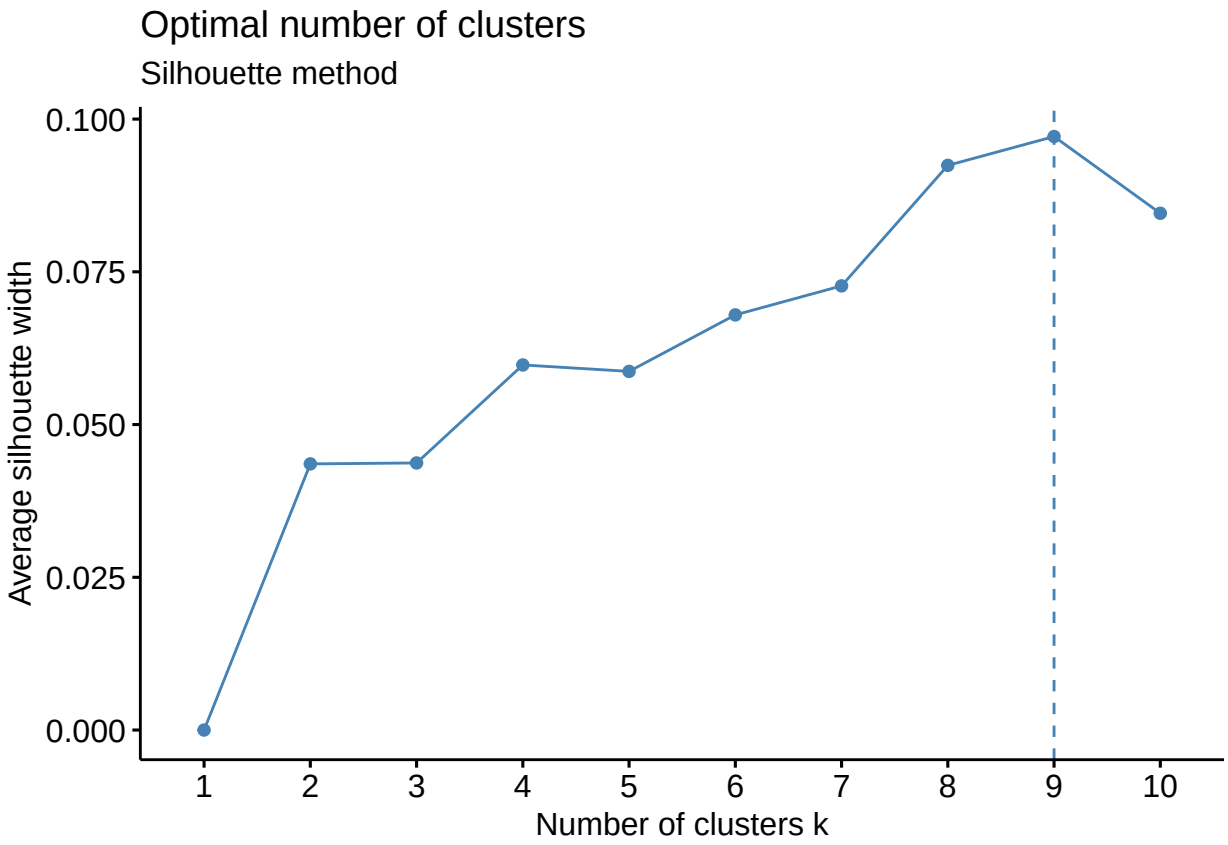
23 principaux axes

Dans cette partie nous utilisons les 23 principaux axe pour la classification des individus, sachant que les 20 premiers axes de l' analyse expriment 66.08% de l'inertie totale du jeu de données ; cela signifie que 66.08% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ces plans

Determimination du nombre

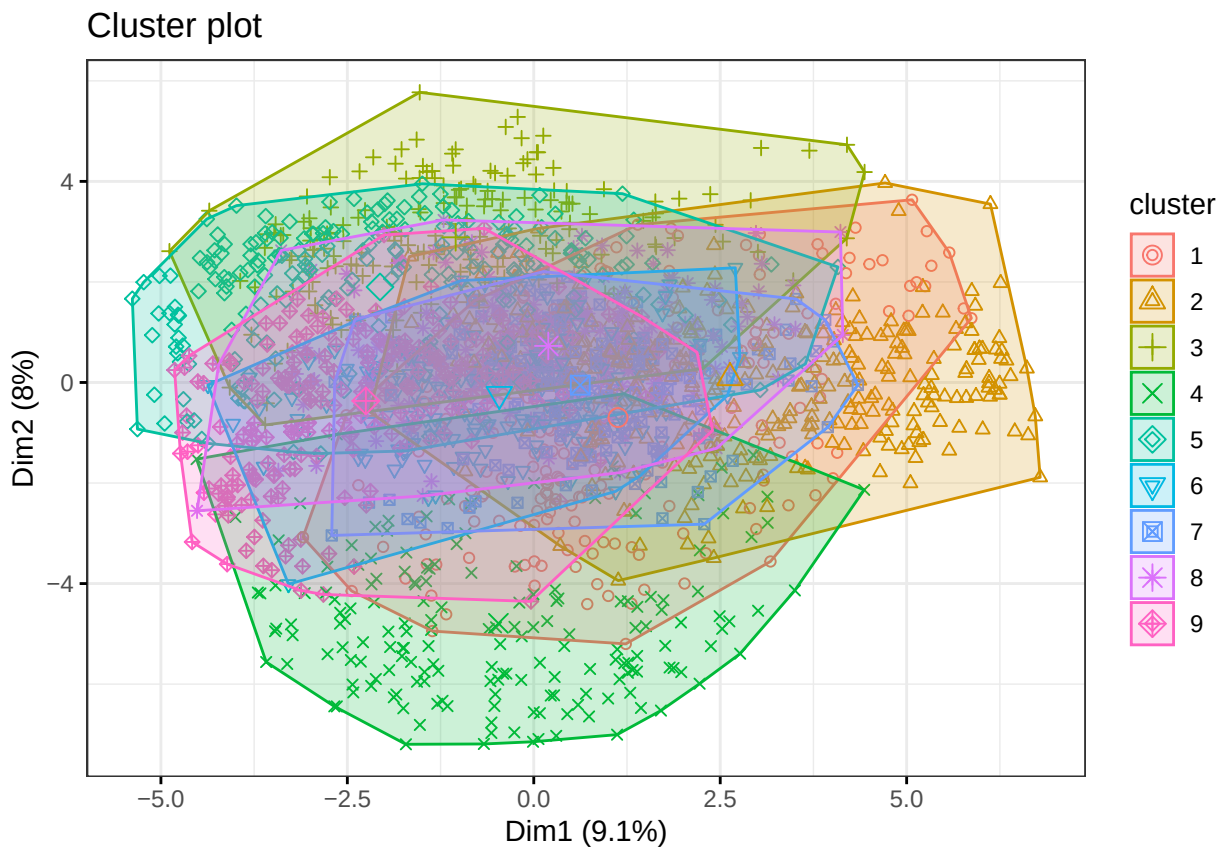
```
## K-Mean

res.pc <- PCA(mfeat.kar,ncp=20, graph = FALSE,scale.unit = TRUE)
ncp=res.pc$ind$coord
d <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(d, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



En Prenant les 20 axes principaux, on obtient 9 clusters optimaux, ce qui corespond bien au nombre de classe initiale dans le jeux de données

```
km.res <- kmeans(d, 9,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.kar,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Cette representation est difficile a interpreter, car nous avons 20 axes principal. Meme si elle donne plus d'informations.

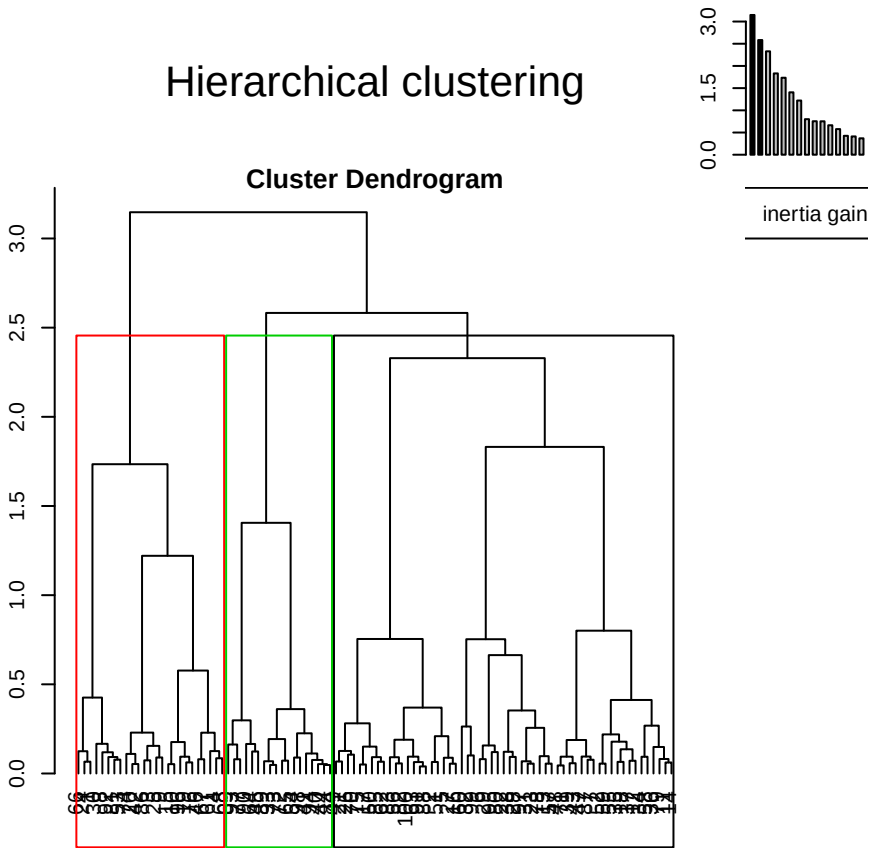
ACP suivit de la Classiffication avec l'algorithme de Ward.

Nous utilison Dans cette partie Une ACP suivi de la classification hiérachique sur le jeu de donnée res.PCA

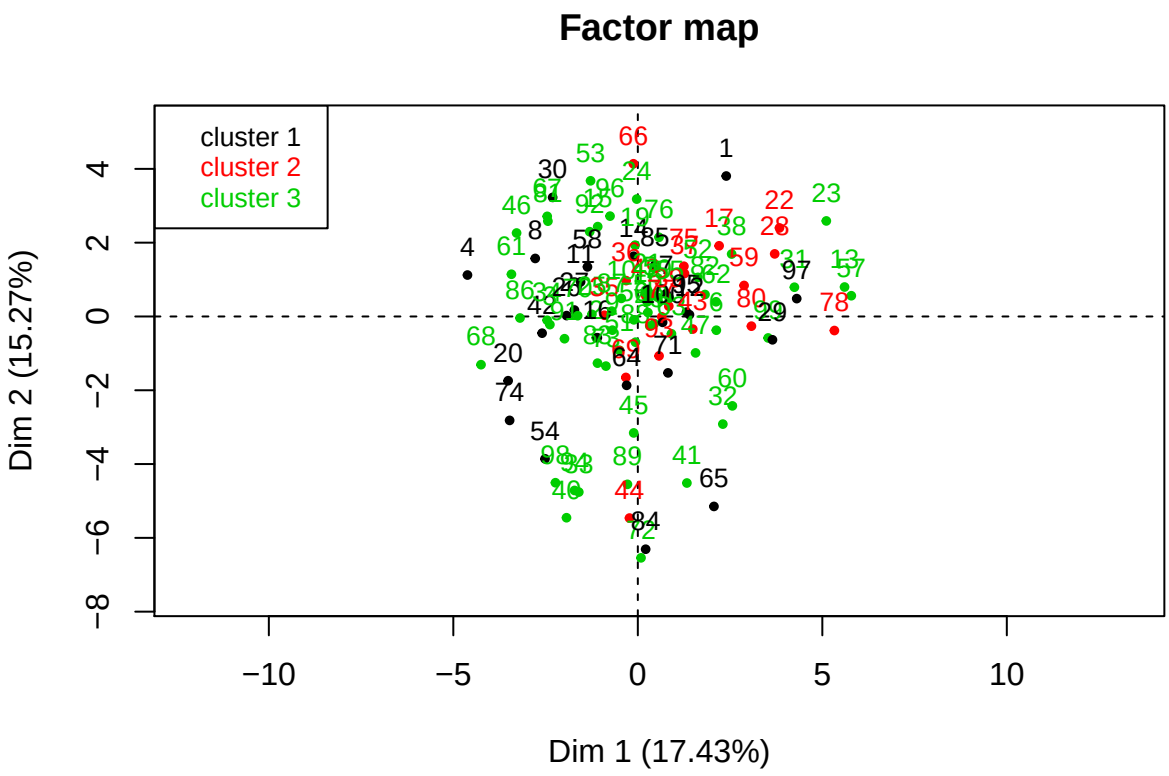
```
res.PCA= PCA(mfeat.kar,ncp=20, graph = FALSE,scale.unit = TRUE) # ACP en conservant 23 dimentions
hc=HCPC(res.PCA,kk=100,description = F,graph = F) # Classification avec prétraitement par Kmean avec kk
```

```
## Warning in HCPC(res.PCA, kk = 100, description = F, graph = F): No consolidation
## has been done after the hierarchical clustering since kk is different from Inf
## (see help for more details)
```

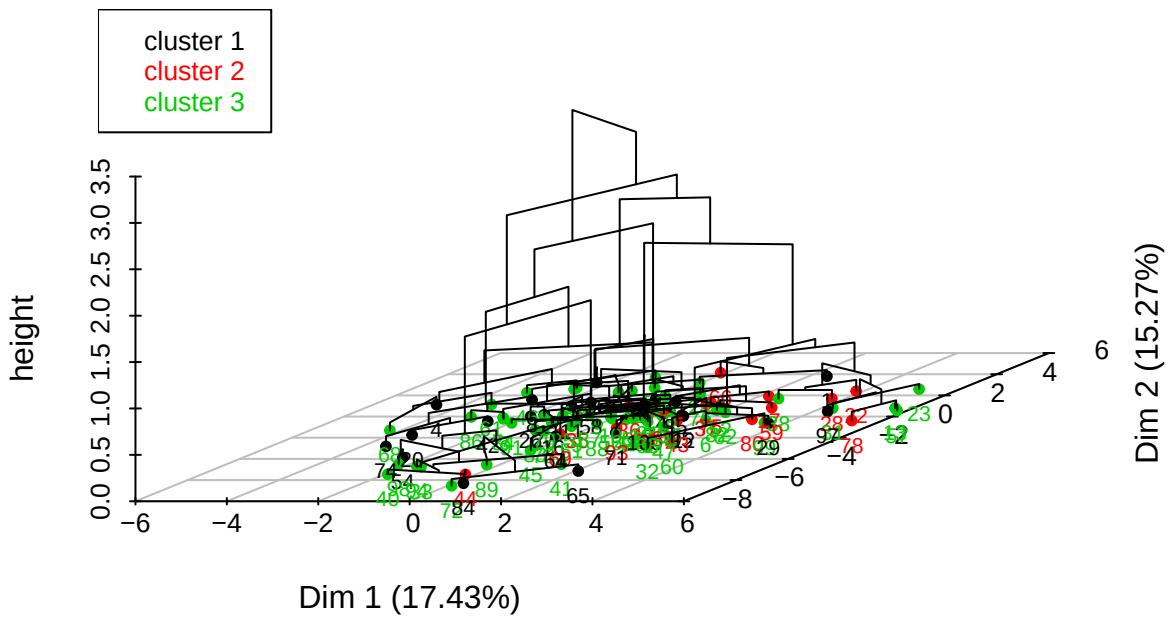
```
plot(hc,choice='tree') # Graphe de l'abre
```




```
plot(hc,choice='map',draw.tree=F) # Plan de l'ACP avec les abres
```




```
plot(hc,choice='3D.map') # Plan de l'ACP avec les abres
```




```
#catdes(hc$data.clust,ncol((hc$data.clust)))  ## caracterisation des classes
```

Classiffication avec l'algorithme de Ward.

Nous utilisons les 20 axes principales, car ayant tous des valeurs propres superieur a 1.

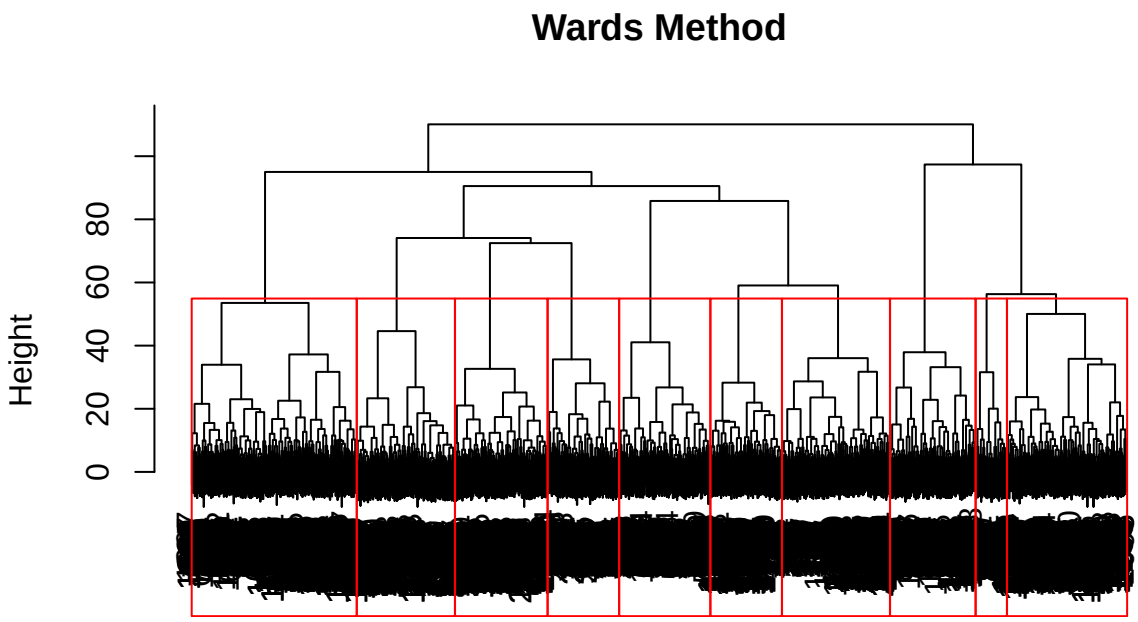
```
res.pca03 <- PCA(mfeat.kar,ncp=20, graph = FALSE,scale.unit = TRUE)
ncp=res.pca03$ind$coord
```

Dans cette partie nous utilisons les cordonnées des individu de l'ACP pour faire la classification avec l'algorithne de ward

```
## standarisation des donnees
new_data_scaled <- ncp
new_data_scaled.dist <- dist(new_data_scaled)      # Calcul de distance
new_data_scaled.distT <- dist(t(new_data_scaled)) # calcul de distance
```

```
# effectuer un regroupement hiérarchique des observations (variables) en utilisant la méthode de Ward
ward = hclust(new_data_scaled.distT,method='ward.D2')
wardT = hclust(new_data_scaled.dist,method='ward.D2')
```

```
plot(wardT, main = 'Wards Method',xlab =" ", sub ="")
rect.hclust(wardT, k =10, border = 'red') ## selecting three clusters
```



Ce cluster nous permet de voir 10 classe.

Conclusion pour la donnée mfeat.kar

Dans cette partie de notre étude nous avons pu momtrer que nos 2000 lignes pouvais être regrouper en 10 classe a partir des methode de l'ACP en reduissant la dimension a un nombre acceptable pour ensuite faire de la Classification. La methode de ACP suivit de Kmean donne donne 5 clusters pour deux axes principaux et 9 cluster pour 20 axe principaux

Donnée mfeat.fac

Nous allons commencer avec les donnees mfeat.mor. Nous n'avons pas assez d'information sur les données

Pour la Description du jeu de données, ou identification de groupes d'individus et liens entre variables nous allons utiliser l'ACP pour décrire ce jeu de données comportant de nombreux individus et variables quantitatives. L'analyse doit permettre d'extraire l'information pertinente et la synthétiser sous forme de composantes principales, nouveaux axes pour décrire le jeu de données.

La fonction PCA()

Nous utilisons la fonction 'PCA()' de 'FactoMineR', elle centre et réduit les variables avant de réaliser l'ACP. Cette étape est importante afin que toutes les variables aient le même poids dans la construction des plans de l'ACP.

```
res.pca04 <- PCA(mfeat.mor,ncp=10, graph = FALSE)
print(res.pca04)
```

```
## **Results for the Principal Component Analysis (PCA)**
```

```
## The analysis was performed on 2000 individuals, described by 6 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Valeurs propres

Pour le choix du nombre de d'axe principale, nous pouvons utiliser la regle de kaiser et la regle de Coude. Nous allons donc afficher le nombre d'axe ou les valeurs propres sont superieur a 1

Comme décrit, les valeurs propres mesurent la quantité de variance expliquée par chaque axe principal. Les valeurs propres sont grandes pour les premiers axes et petits pour les axes suivants. Autrement dit, les premiers axes correspondent aux directions portant la quantité maximale de variation contenue dans le jeu de données.

Nous examinons les valeurs propres pour déterminer le nombre de composantes principales à prendre en considération en fonction de la regle de kaiser et de coude

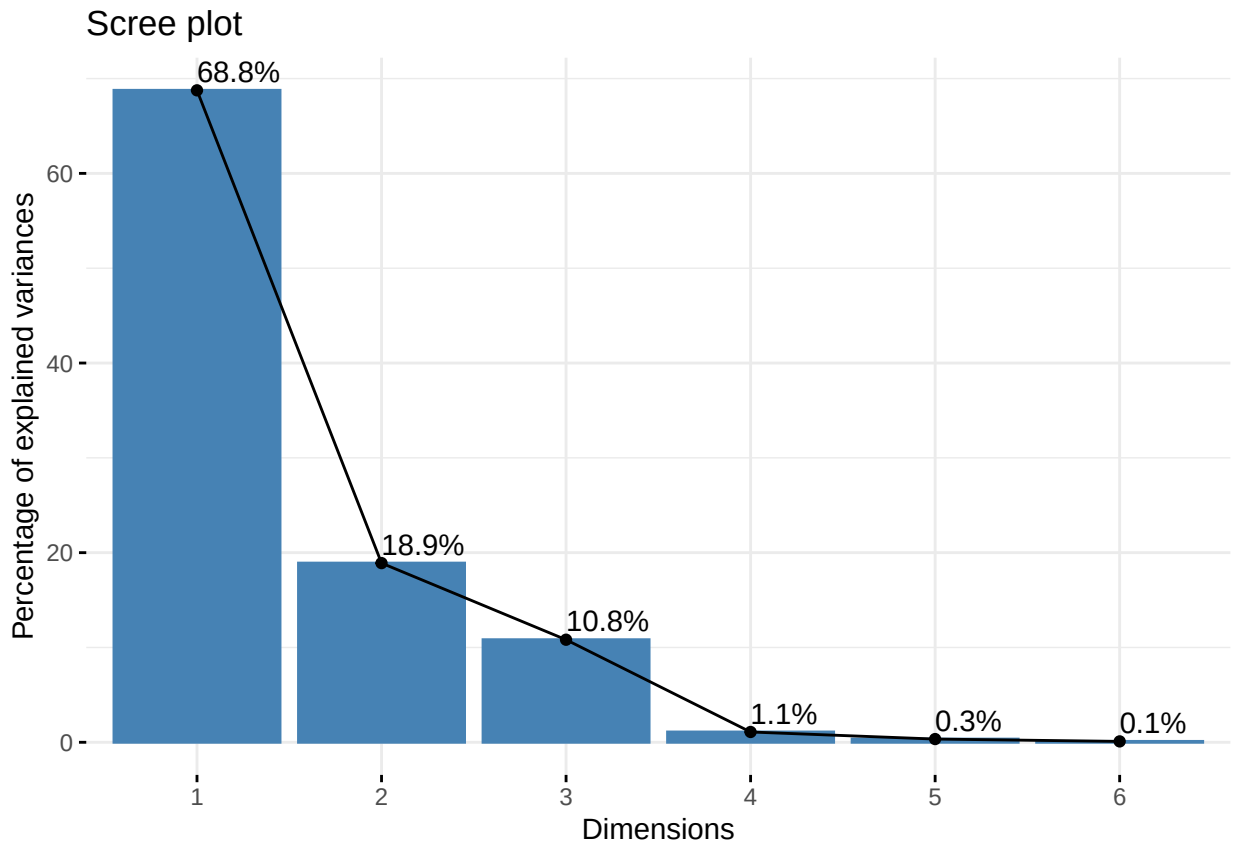
Nous visualisons le tableau seulement pour les lignes dont les valeurs propres sont supperieur a 1

```
dt=as.data.frame(res.pca04$eig)
dt[dt$eigenvalue >= 1, ]
```

```
##        eigenvalue percentage of variance cumulative percentage of variance
## comp 1   4.125586              68.75977                          68.75977
## comp 2   1.133905              18.89842                          87.65819
```

La règle de Kaiser repose sur une idée simple. Dans une ACP normée, la somme des valeurs propres étant égale au nombre de variables, leur moyenne vaut 1. Nous considérons par conséquent qu'un axe est intéressant si sa valeur propre est supérieure 1 Nous avons ici 2 valeurs propres superieur a 1, ce qui rend l'interpretation beaucoup plus simple.

```
fviz_eig(res.pca04, addlabels = TRUE)
```



La regle de coude nous permet de choisir deux axes principaux

Distribution de l'inertie

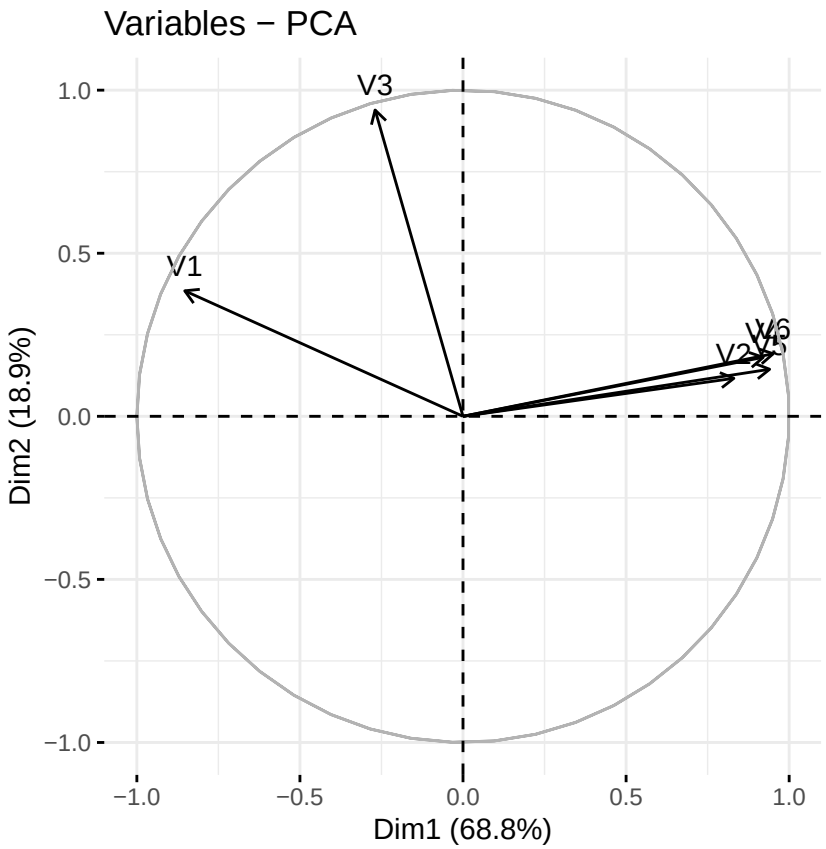
L'inertie des axes factoriels indique d'une part si les variables sont structurées et suggère d'autre part le nombre judicieux de composantes principales à étudier.

Les 2 premiers axes de l' analyse expriment 87.65% de l'inertie totale du jeu de données ; cela signifie que 87.65% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan. C'est un pourcentage relativement bon, et le premier plan représente donc seulement une part important de la variabilité contenue dans l'ensemble du jeu de données actif.

Cercle de corrélation

La corrélation entre une variable et une composante principale est utilisée comme coordonnées de la variable sur la composante principale.

```
fviz_pca_var(res.pca04, select.var = list(cos2 = 0.5))
```



En prenant les variables dont les contibutions sont superieur a 0.5 sur l'une des deux axes, on obtient le cercle de correlation ci-dessus.

Pour ce faire nous allons faire une classifiaction avec la ''methode de ward" et ''methode K-Mean" sur les coordonnées des individus pour les rasembler en different classe.

Classification par la methode K-Mean

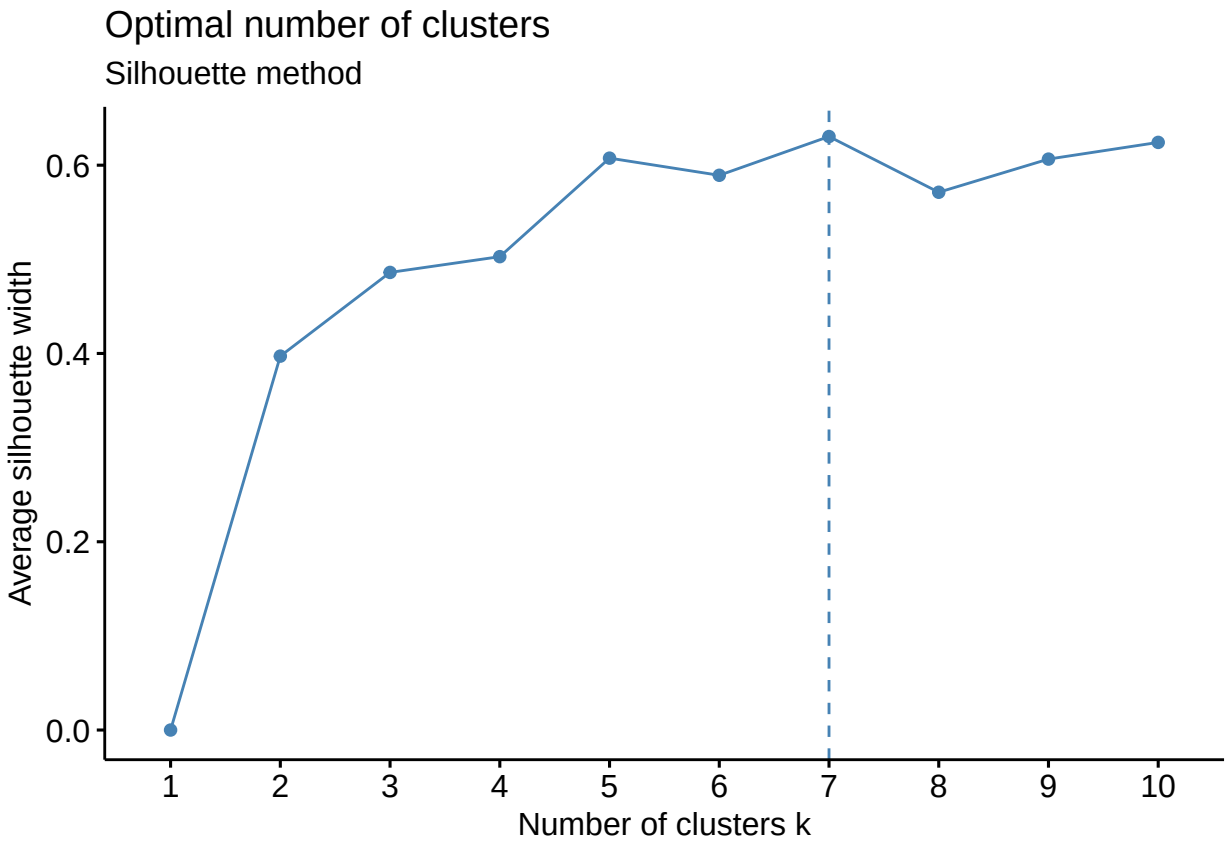
Deux principaux axes

Dans cette partie nous utilisons les deux principaux axe pour la classification des individus, sachant que les 2 premiers axes de l' analyse expriment 87.65% de l'inertie totale du jeu de données ; cela signifie que 87.65% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan

Determimination du nombre de Clusters optimaux

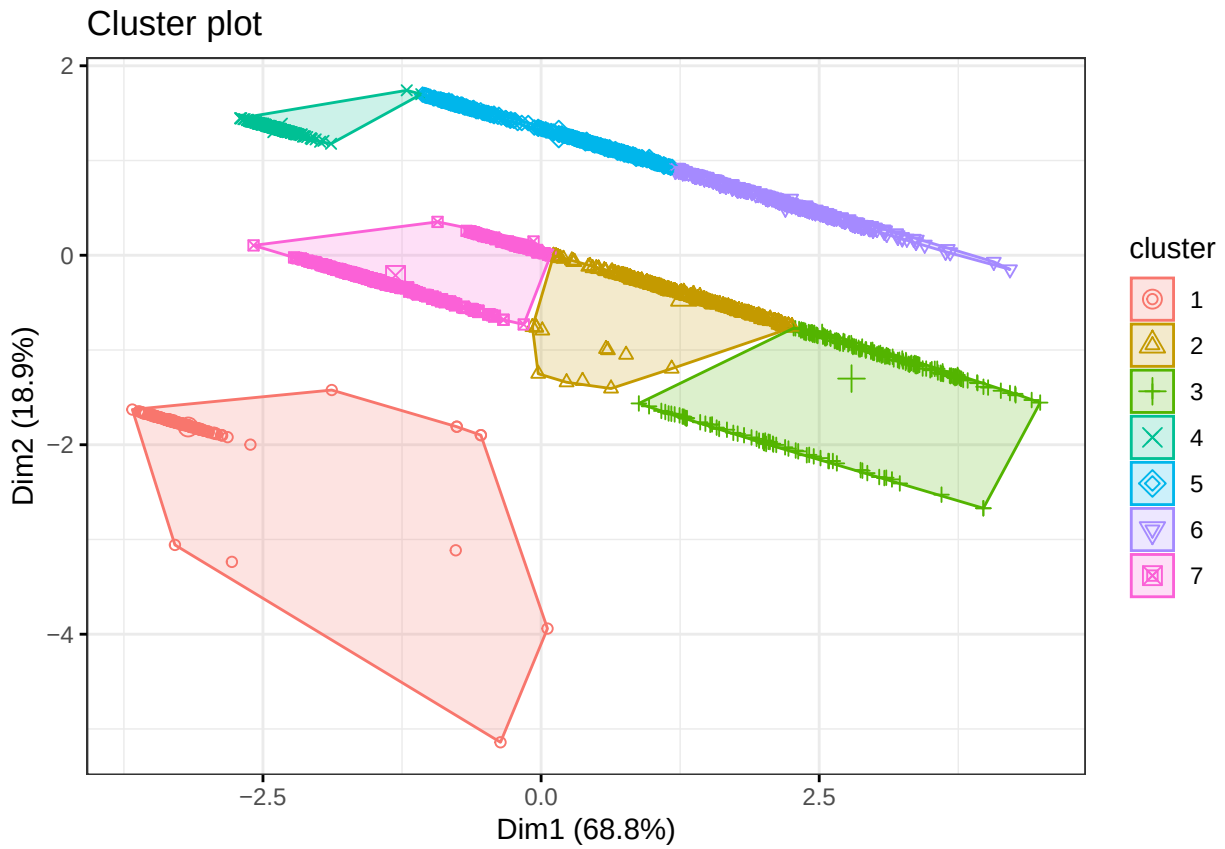
```
## K-Mean

res.pca04 <- PCA(mfeat.mor,ncp=2, graph = FALSE,scale.unit = TRUE)
ncp=res.pca04$ind$coord
df04 <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df04, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



Lorsque nous prenons les deux axes principaux, on se retrouve avec 7 clusters optimaux.

```
km.res <- kmeans(df04, 7,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.mor,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Dans la figure précedente, nous avons representés nos individus en 7 classe, vu le nombre de classe optimal. Nous avons de bonne presentation pour certaine classe, une bonne separation

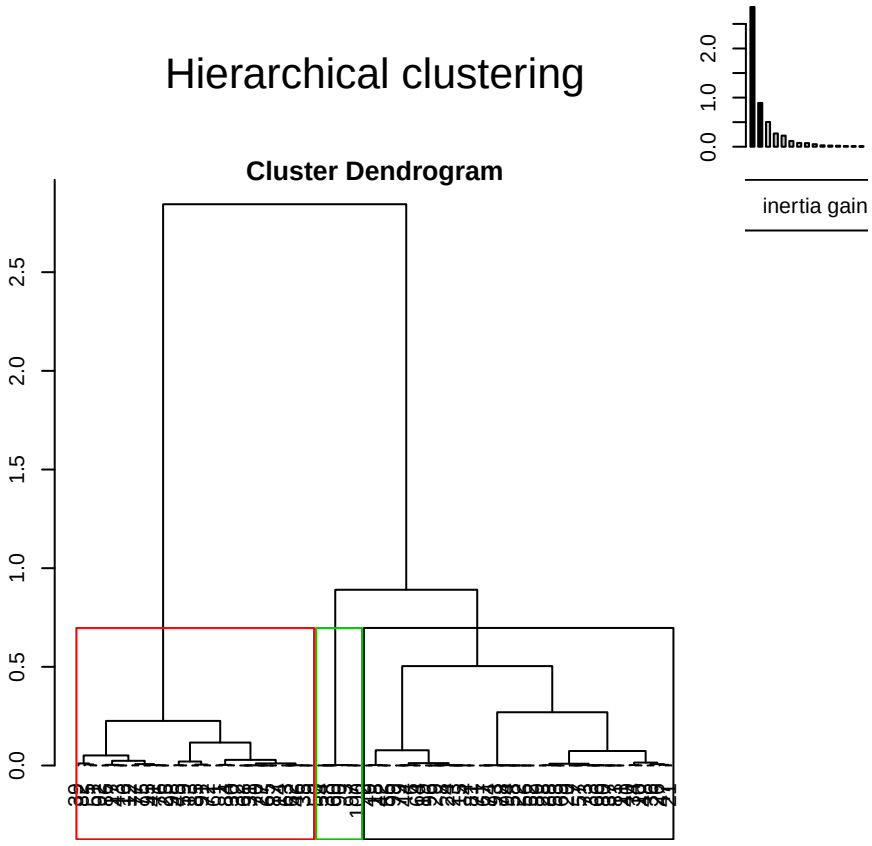
Classiffication avec l'ACP suivi de l'algorithme de Ward.

Nous utilison Dans cette partie Une ACP suivi de la classification hiérachique sur le jeu de donnée res.PCA

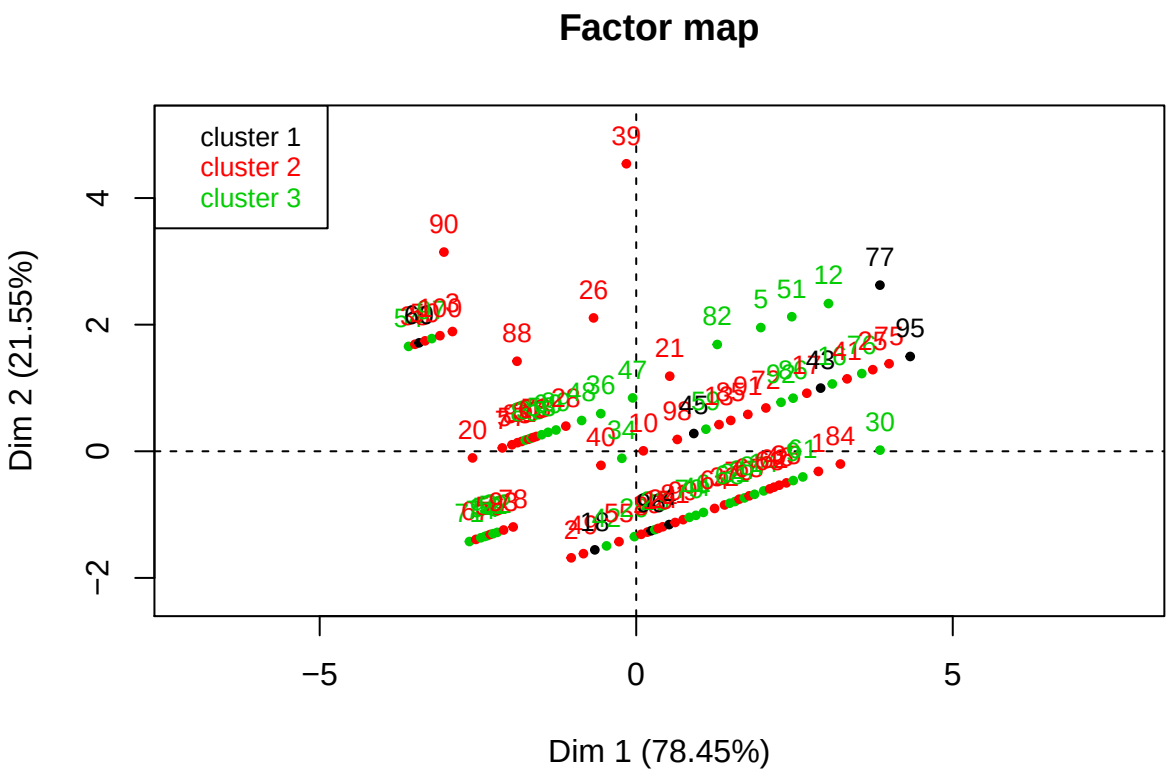
```
res.PCA= PCA(mfeat.mor,ncp=2, graph = FALSE,scale.unit = TRUE) # ACP en conservant 2 dimentions
hc=HCPC(res.PCA,kk=100,description = F,graph = F) # Classification avec prétraitement par Kmean avec kk
```

```
## Warning in HCPC(res.PCA, kk = 100, description = F, graph = F): No consolidation
## has been done after the hierarchical clustering since kk is different from Inf
## (see help for more details)
```

```
plot(hc,choice='tree') # Graphe de l'abre
```




```
plot(hc,choice='map',draw.tree=F) # Plan de l'ACP avec les abres
```




```
#catdes(hc$data.clust,ncol((hc$data.clust)))  ## caracterisation des classes
```

Classiffication avec l'algorithme de Ward.

Nous utilisons les 2 axes principales, car ayant tous des valeurs propres superieur a 1.

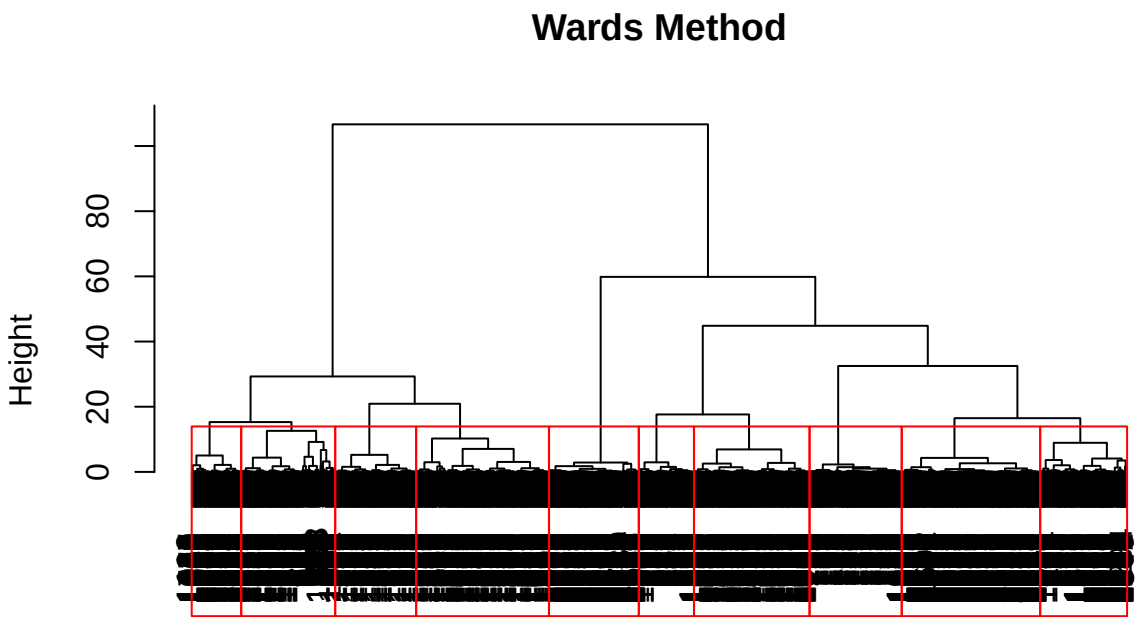
```
res.pca04 <- PCA(mfeat.mor,ncp=2, graph = FALSE,scale.unit = TRUE)
ncp=res.pca04$ind$coord
```

Dans cette partie nous utilisons les cordonnées des individu de l'ACP pour faire la classification avec l'algorithne de ward

```
## standarisation des donnees
new_data_scaled <- ncp
new_data_scaled.dist <- dist(new_data_scaled)      # Calcul de distance
new_data_scaled.distT <- dist(t(new_data_scaled)) # calcul de distance
```

```
# effectuer un regroupement hiérarchique des observations (variables) en utilisant la méthode de Ward
ward = hclust(new_data_scaled.distT,method='ward.D2')
wardT = hclust(new_data_scaled.dist,method='ward.D2')
```

```
plot(wardT, main = 'Wards Method',xlab =" ", sub ="")
rect.hclust(wardT, k =10, border = 'red') ## selecting three clusters
```



Ce cluster nous permet de voir 10 classe.

Conclusion pour la donnée mfeat.mor

Dans cette partie de notre etude nous avons pu momtrer que nos 2000 ligne pouvais etre regrouper en 10 classe a partir des methode de l'ACP en reduissant la dimension a un nombre acceptable pour ensuite faire de la Classification.

Le KMean a montré 7 cluster

Donnée mfeat.fac

Nous allons commencer avec les donnees mfeat.fac. Nous n'avons pas assez d'information sur les données

Pour la Description du jeu de données, ou identification de groupes d'individus et liens entre variables nous allons utiliser l'ACP pour décrire ce jeu de données comportant de nombreux individus et variables quantitatives. L'analyse doit permettre d'extraire l'information pertinente et la synthétiser sous forme de composantes principales, nouveaux axes pour décrire le jeu de données.

La fonction PCA()

Nous utilisons la fonction 'PCA()' de 'FactoMineR', elle centre et réduit les variables avant de réaliser l'ACP. Cette étape est importante afin que toutes les variables aient le même poids dans la construction des plans de l'ACP.

```
res.pca05 <- PCA(mfeat.zer,ncp=10, graph = FALSE)
print(res.pca05)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 2000 individuals, described by 47 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Valeurs propres

Pour le choix du nombre de d'axe principale, nous pouvons utiliser la regle de kaiser et la regle de Coude. Nous allons donc afficher le nombre d'axe ou les valeurs propres sont superieur a 1

Comme décrit, les valeurs propres mesurent la quantité de variance expliquée par chaque axe principal. Les valeurs propres sont grandes pour les premiers axes et petits pour les axes suivants. Autrement dit, les premiers axes correspondent aux directions portant la quantité maximale de variation contenue dans le jeu de données.

Nous examinons les valeurs propres pour déterminer le nombre de composantes principales à prendre en considération en fonction de la regle de kaiser et de coude

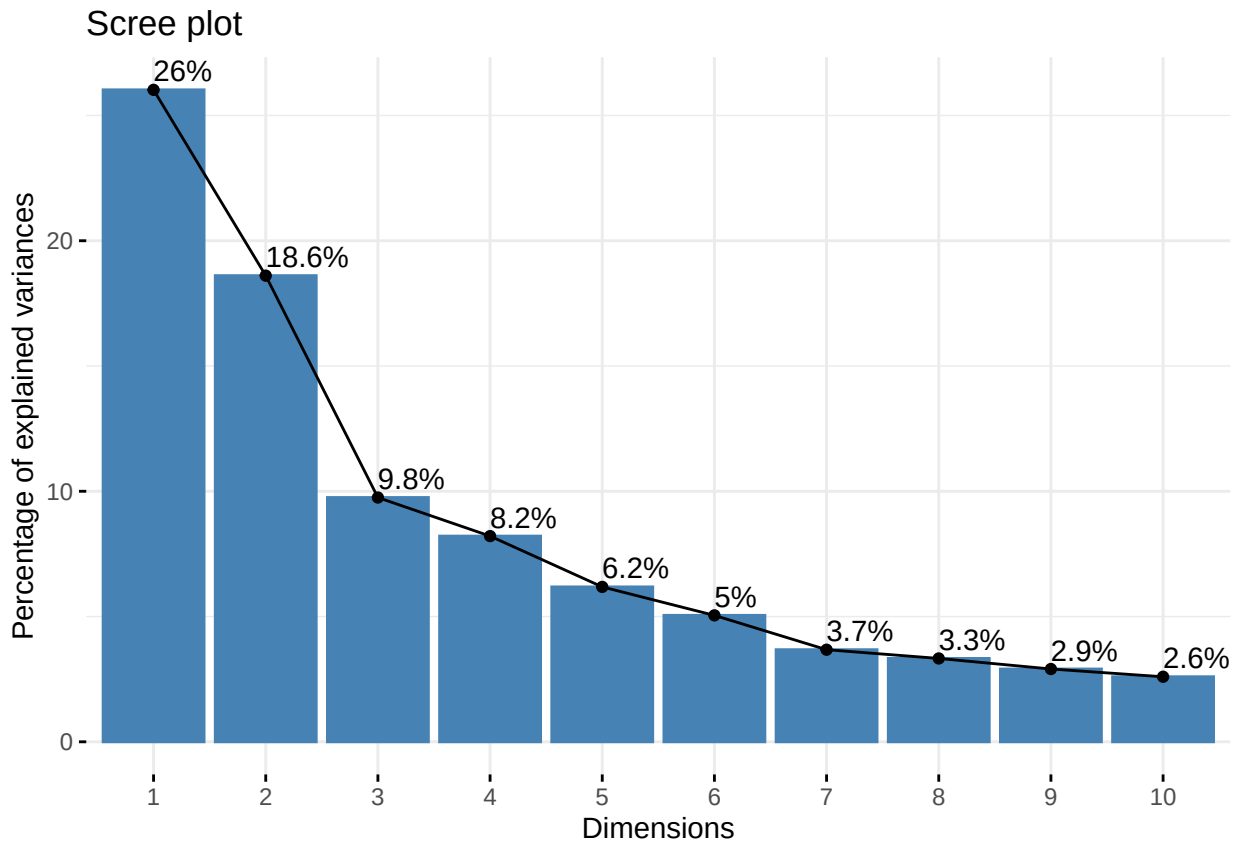
Nous visualisons le tableau seulement pour les lignes dont les valeurs propres sont supperieur a 1

```
dt=as.data.frame(res.pca05$eig)
dt[dt$eigenvalue >= 1, ]
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   12.230354              26.022030                           26.02203
## comp 2    8.744983              18.606348                           44.62838
## comp 3    4.582542               9.750089                           54.37847
## comp 4    3.859125               8.210903                           62.58937
## comp 5    2.906020               6.183021                           68.77239
## comp 6    2.372048               5.046911                           73.81930
## comp 7    1.728062               3.676727                           77.49603
## comp 8    1.563967               3.327588                           80.82362
## comp 9    1.365773               2.905900                           83.72952
## comp 10   1.220392               2.596580                           86.32610
## comp 11   1.080892               2.299769                           88.62587
```

La règle de Kaiser repose sur une idée simple. Dans une ACP normée, la somme des valeurs propres étant égale au nombre de variables, leur moyenne vaut 1. Nous considérons par conséquent qu'un axe est intéressant si sa valeur propre est supérieure 1 Nous avons ici 11 valeurs propres superieur a 1, ce qui rend l'interpretation beaucoup plus compliquer.

```
fviz_eig(res.pca05, addlabels = TRUE)
```



La regele de coude devais nous permettre de choisir 03 axes principaux

Distribution de l'inertie

L'inertie des axes factoriels indique d'une part si les variables sont structurées et suggère d'autre part le nombre judicieux de composantes principales à étudier.

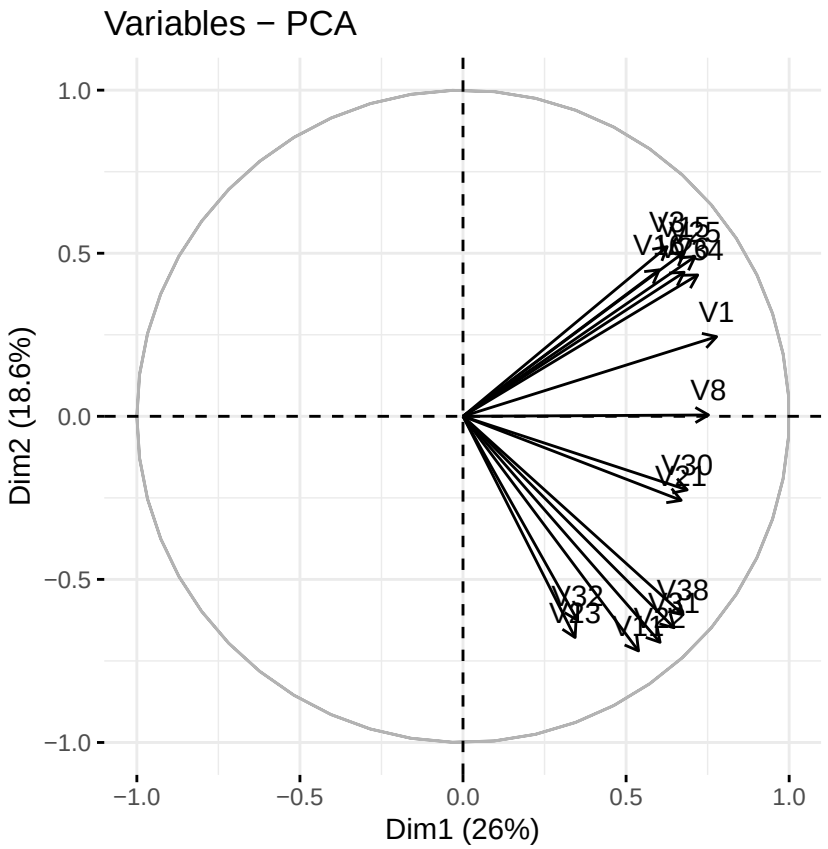
Les 2 premiers axes de l' analyse expriment 44.62% de l'inertie totale du jeu de données ; cela signifie que 44.62% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan. C'est un pourcentage relativement moyen, et le premier plan représente donc seulement une part de la variabilité contenue dans l'ensemble du jeu de données actif.

Du fait de ces observations, il serait alors probablement nécessaire de considérer également les dimensions supérieures ou égales à la troisième dans l'analyse.

Cercle de corrélation

La corrélation entre une variable et une composante principale est utilisée comme coordonnées de la variable sur la composante principale.

```
fviz_pca_var(res.pca05, select.var = list(cos2 = 0.5))
```



En prenant les variables dont les contibutions sont superieur a 0.5 sur l'une des deux axes, on obtient le cercle de correlation ci-dessus. Tous les variables dont la contribution est superieur a 0.5 est corrélé au premier axe.

Nous allons faire une classifiaction avec la ''methode de ward" et ''methode K-Mean" sur les coordonnées des individus pour les rasembler en different classe.

Classification par la methode K-Mean

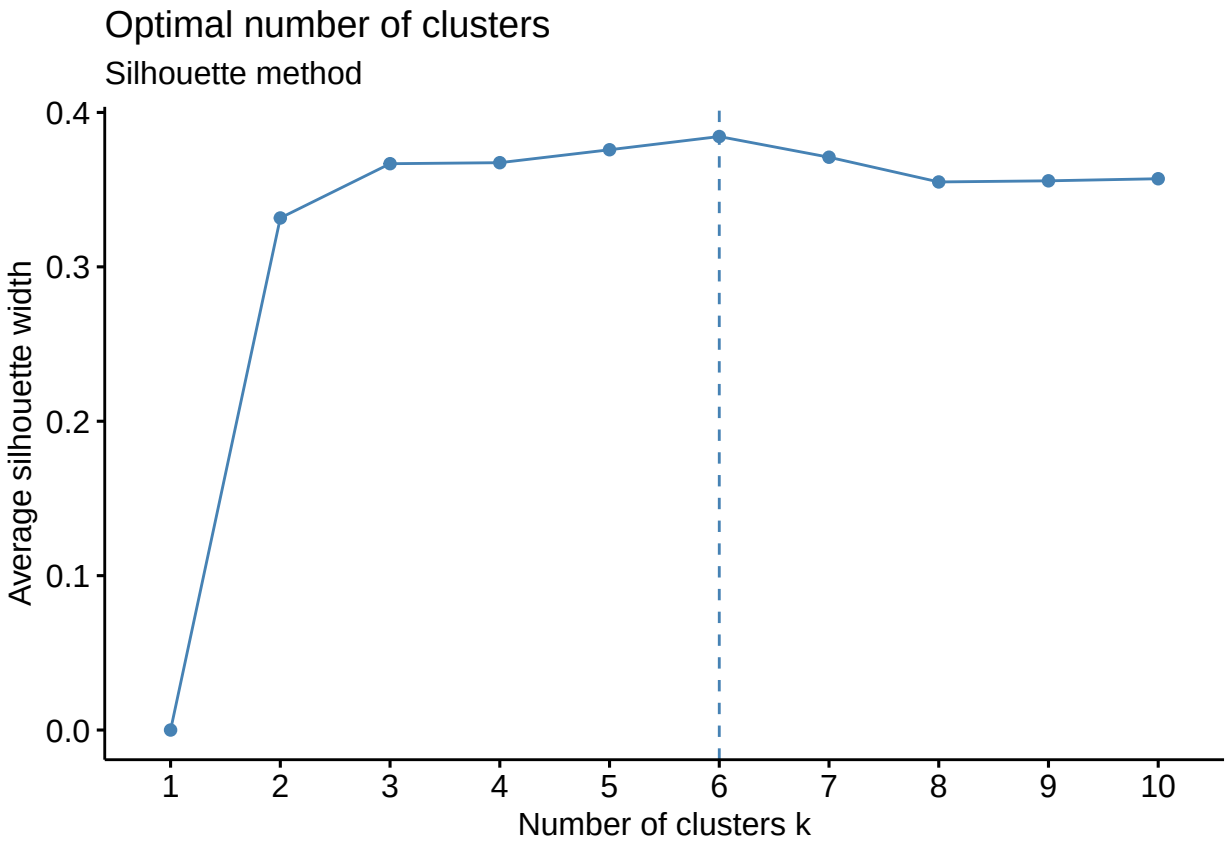
Deux principaux axes

Dans cette partie nous utilisons les deux principaux axe pour la classification des individus, sachant que les 2 premiers axes de l' analyse expriment 44.62% de l'inertie totale du jeu de données ; cela signifie que 44.62% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan

Determimination du nombre de Clusters optimaux

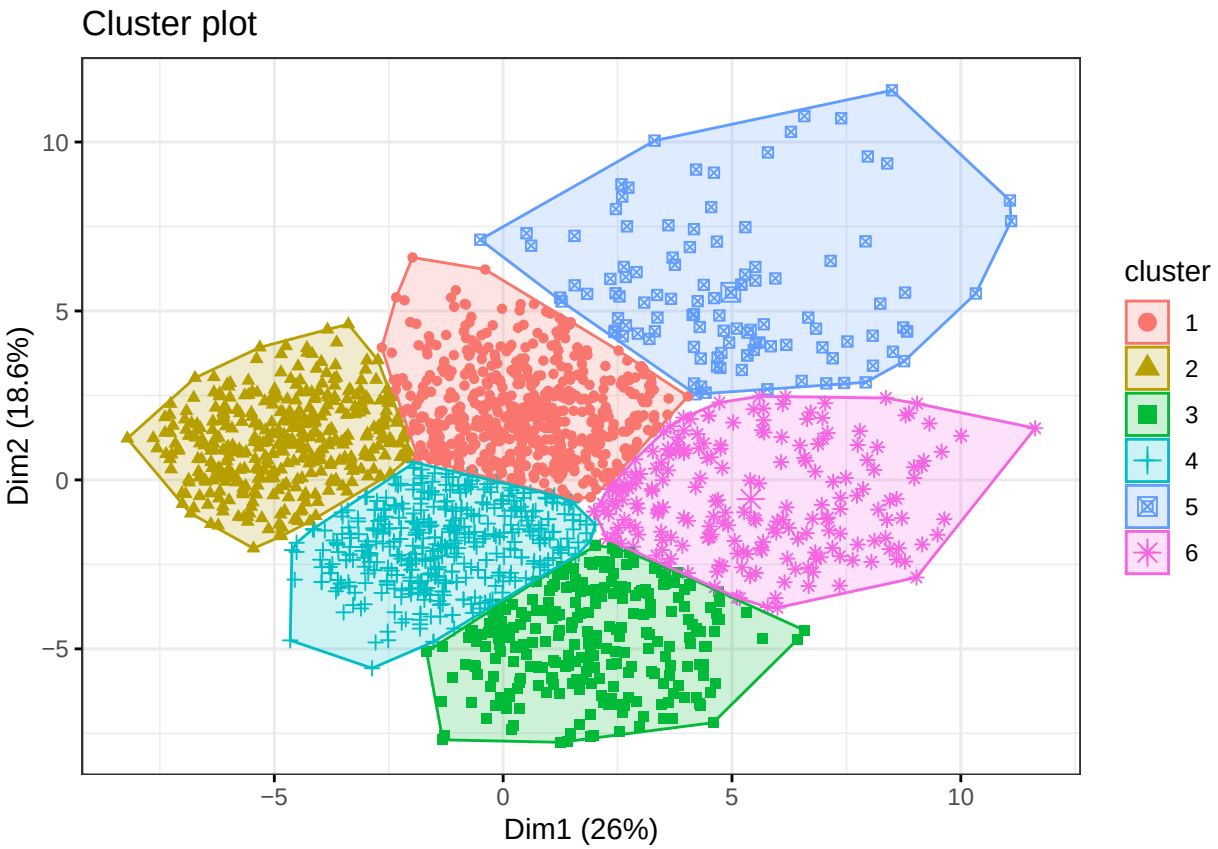
```
## K-Mean

res.pca05 <- PCA(mfeat.zer,ncp=2, graph = FALSE,scale.unit = TRUE)
ncp=res.pca05$ind$coord
df05 <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df05, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



Lorsque nous prenons les deux axes principaux, on se retrouve avec 6 clusters optimaux. Cependand on peut bien voir que en prenant 10 cluster on s'eloigne pas de la performance que apportent les 6 cluster

```
km.res <- kmeans(df05, 6,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.zer,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Dans la figure précedente, nous avons representés nos individus en 6 classe, vu le nombre de classe optimal.

23 principaux axes

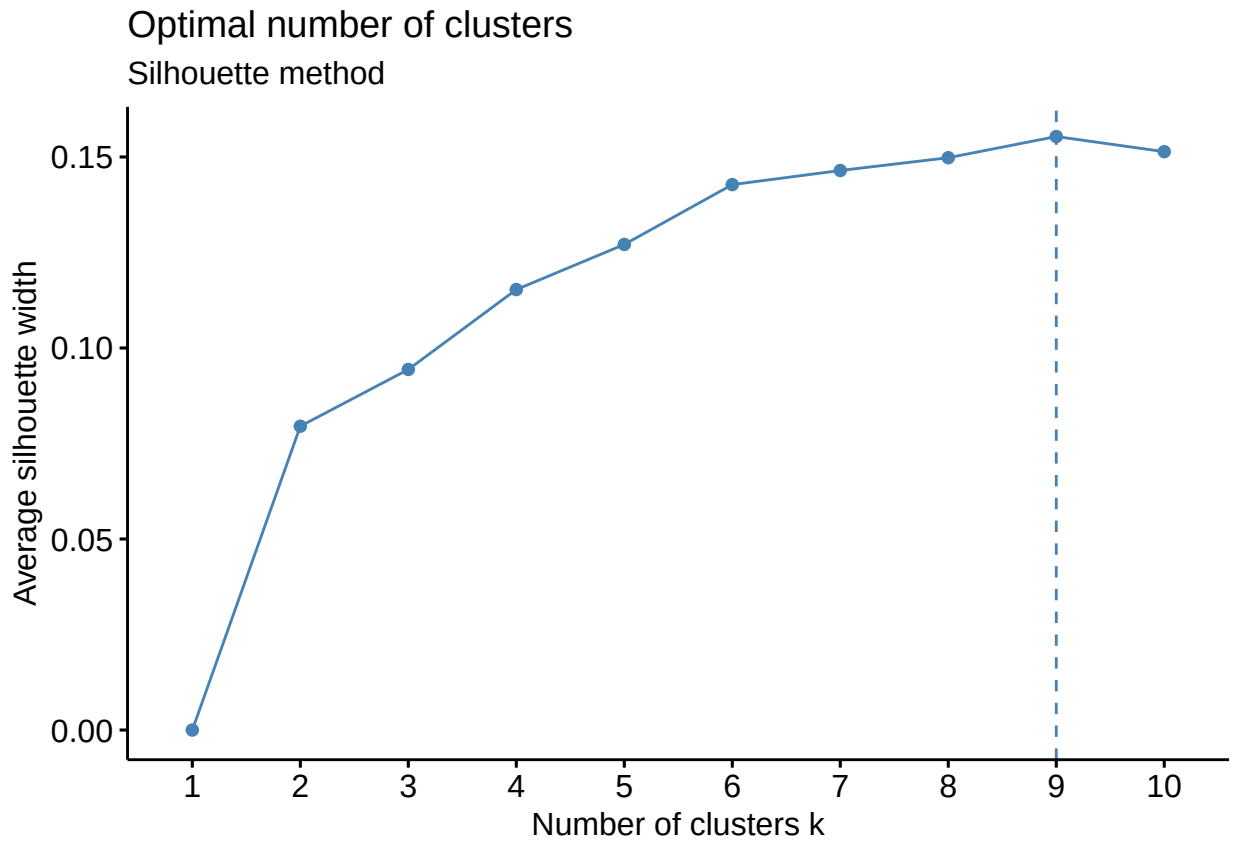
Dans cette partie nous utilisons les 11 principaux axe pour la classification des individus, sachant que les 11 premiers axes de l' analyse expriment 88.62% de l'inertie totale du jeu de données ; cela signifie que 88.62% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ces plans

Determimination du nombre

```
## K-Mean

res.pca11 <- PCA(mfeat.zer,ncp=11, graph = FALSE,scale.unit = TRUE)
ncp=res.pca11$ind$coord
df11 <- scale(ncp) # Scaling the data
```

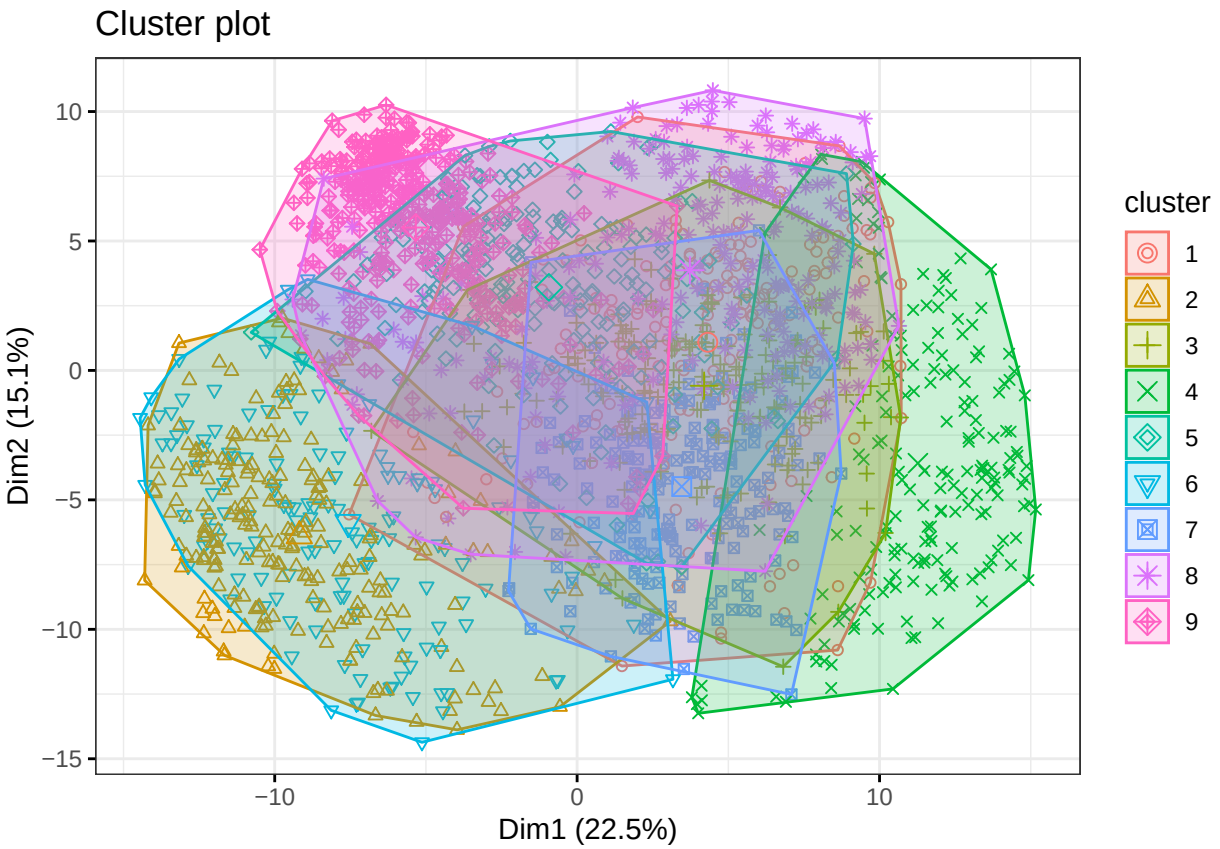
```
# Silhouette method
fviz_nbclust(df11, kmeans,nstart = 500,iter.max = 500, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)
```



En Prenant les 11 axes principaux, on obtient 9 clusters optimaux, ce qui corespond bien au nombre de classe initiale dans le jeux de données

```
km.res <- kmeans(df23, 9,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.fac,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Cette representation est difficile a interpreter, car nous avons 11 axes principal. Meme si elle donne plus d'informations.

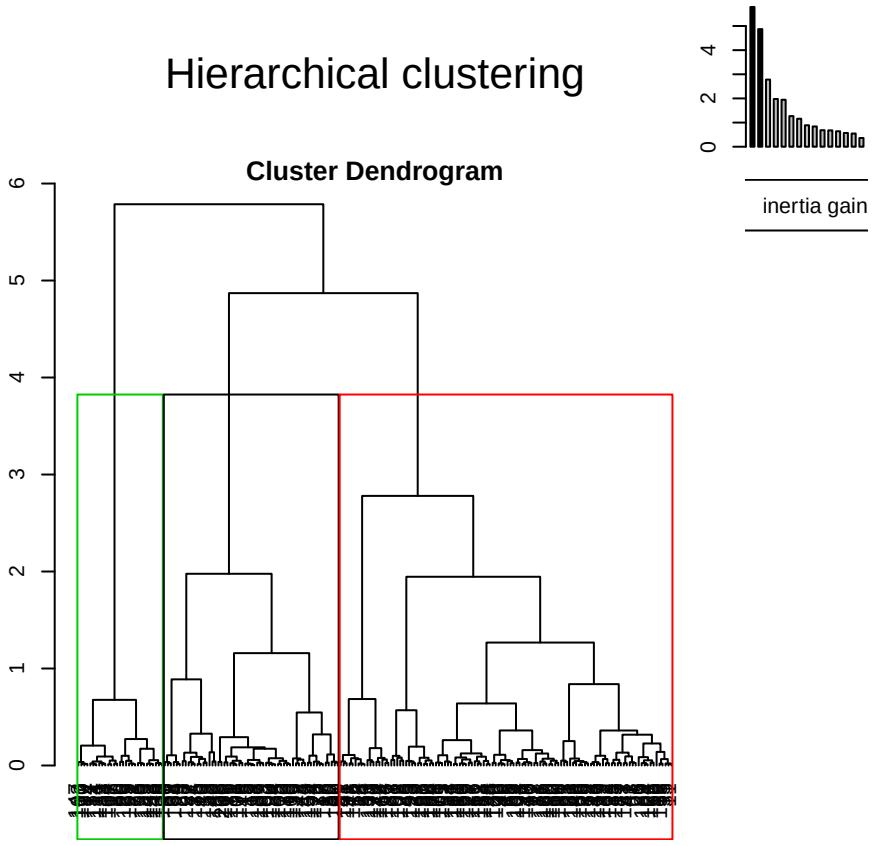
Classiffication avec l'ACP suivi de l'algorithme de Ward.

Nous utilison Dans cette partie Une ACP suivi de la classification hiérachique sur le jeu de donnée res.PCA

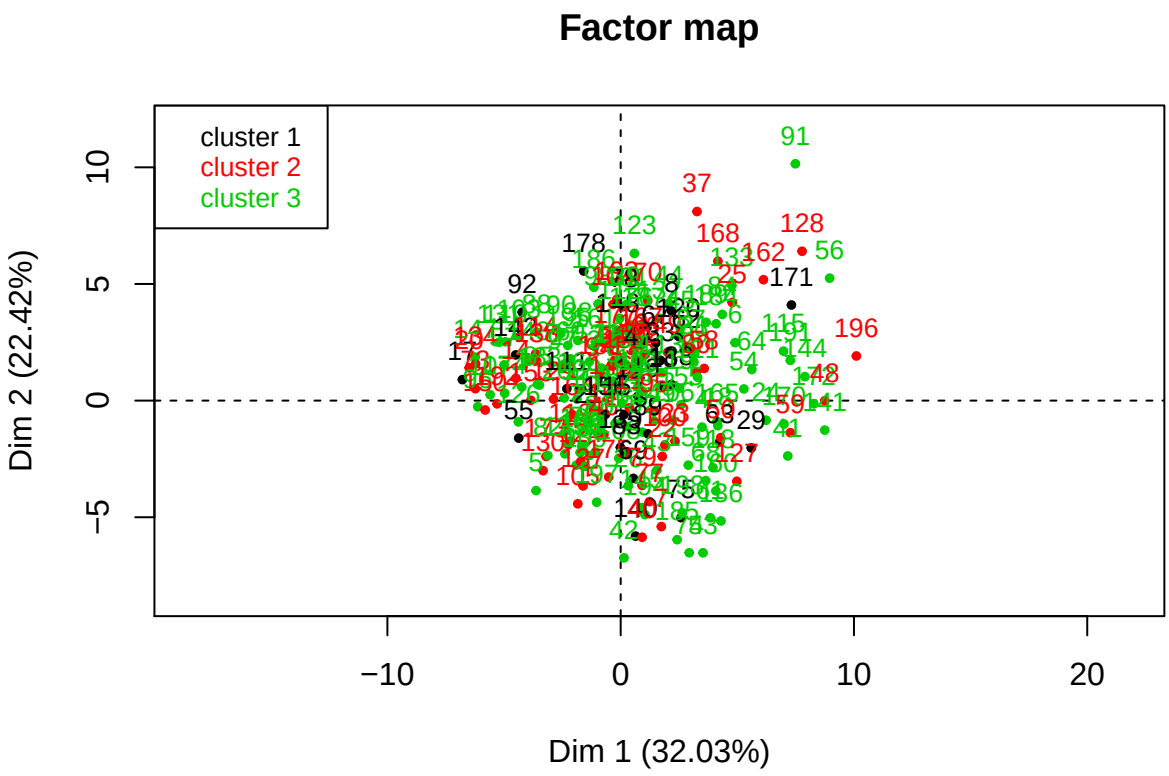
```
res.PCA= PCA(mfeat.zer ,ncp=11, graph = FALSE,scale.unit = TRUE) # ACP en conservant 21 dimentions
hc=HCPC(res.PCA,kk=200,description = F,graph = F) # Classification avec prétraitement par Kmean avec kk
```

```
## Warning in HCPC(res.PCA, kk = 200, description = F, graph = F): No consolidation
## has been done after the hierarchical clustering since kk is different from Inf
## (see help for more details)
```

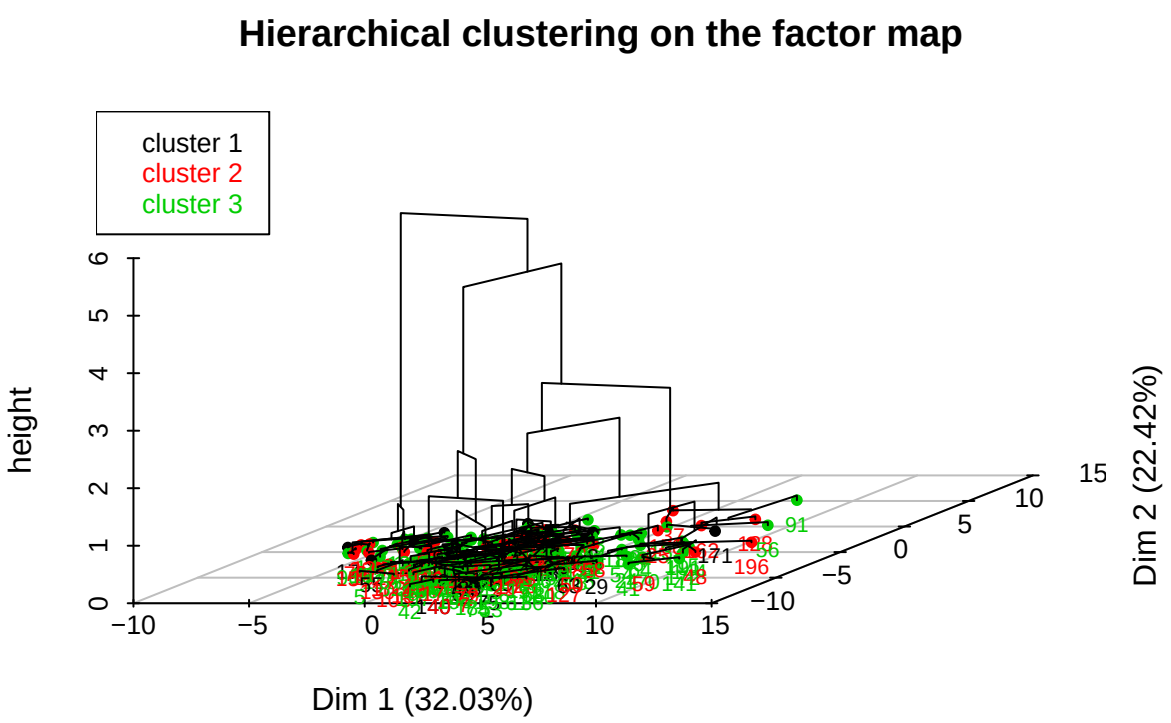
```
plot(hc,choice='tree') # Graphe de l'abre
```




```
plot(hc,choice='map',draw.tree=F) # Plan de l'ACP avec les abres
```




```
plot(hc,choice='3D.map') # Plan de l'ACP avec les abres
```




```
#catdes(hc$data.clust,ncol((hc$data.clust)))  ## caracterisation des classes
```

Classiffication avec l'algorithme de Ward.

Nous utilisons les 23 axes principales, car ayant tous des valeurs propres superieur a 1.

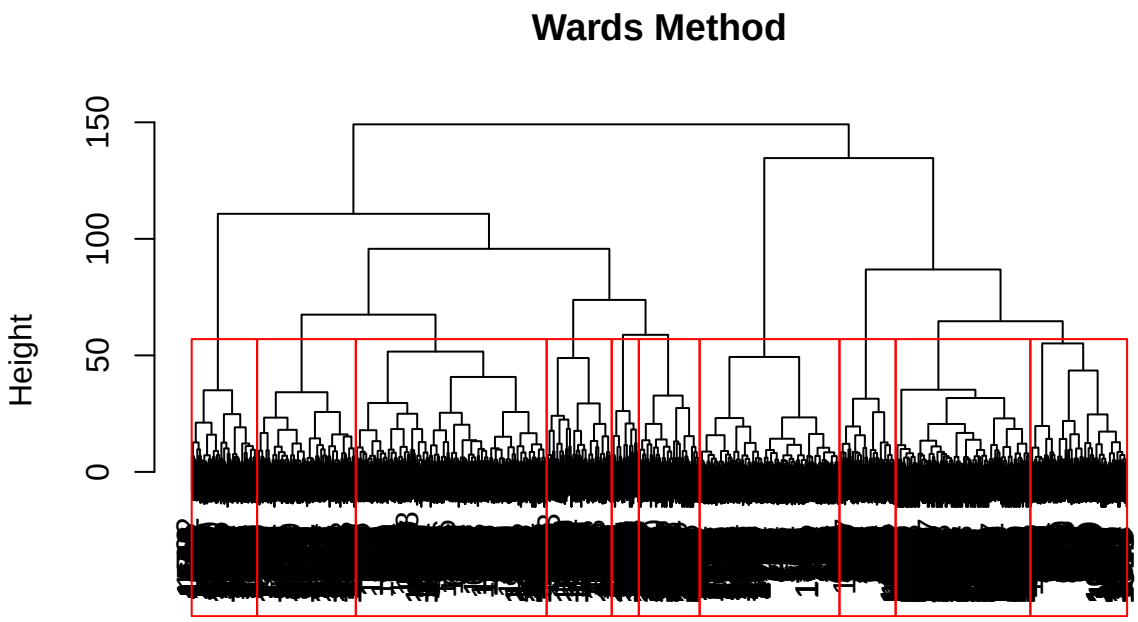
```
res.pca05 <- PCA(mfeat.zer,ncp=11, graph = FALSE,scale.unit = TRUE)
ncp=res.pca05$ind$coord
```

Dans cette partie nous utilisons les cordonnées des individu de l'ACP pour faire la classification avec l'algorithne de ward

```
## standarisation des donnees
new_data_scaled <- ncp
new_data_scaled.dist <- dist(new_data_scaled)      # Calcul de distance
new_data_scaled.distT <- dist(t(new_data_scaled)) # calcul de distance
```

```
# effectuer un regroupement hiérarchique des observations (variables) en utilisant la méthode de Ward
ward = hclust(new_data_scaled.distT,method='ward.D2')
wardT = hclust(new_data_scaled.dist,method='ward.D2')
```

```
plot(wardT, main = 'Wards Method',xlab =" ", sub ="")
rect.hclust(wardT, k =10, border = 'red') ## selecting three clusters
```



Ce cluster nous permet de voir 10 classe.

Conclusion pour la donnée mfeat.zer

Dans cette partie de notre etude nous avons pu momtrer que nos 2000 ligne pouvais etre regrouper en 10 classe a partir des methode de l'ACP en reduissant la dimension a un nombre acceptable pour ensuite faire de la Classification. La methode de KMean donne 6 clusteurs avec deux axes principaux et 9 classes avec les 11 axes principaux

Donnée mfeat.pix

Nous allons commencer avec les donnees mfeat.pix Nous n'avons pas assez d'information sur les données

```
#head(mfeat.fac, n=5)
```

```
# class(mfeat.pix)        # checking your data class

# dim(mfeat.pix)          # getting the dimension of your data

# names(mfeat.pix)        # getting the column names

# str(mfeat.pix)          # checking data structure

# anyNA(mfeat.pix)        # checking for missing values

# summary(mfeat.pix)      # summary of the data
```

```
# View(mfeat.pix)          # view your data
```

Pour la Description du jeu de données, ou identification de groupes d'individus et liens entre variables nous allons utiliser l'ACP pour décrire ce jeu de données comportant de nombreux individus et variables quantitatives. L'analyse doit permettre d'extraire l'information pertinente et la synthétiser sous forme de composantes principales, nouveaux axes pour décrire le jeu de données.

La fonction PCA()

Nous utilisons la fonction 'PCA()' de 'FactoMineR', elle centre et réduit les variables avant de réaliser l'ACP. Cette étape est importante afin que toutes les variables aient le même poids dans la construction des plans de l'ACP.

```
res.pca <- PCA(mfeat.pix,ncp=10, graph = FALSE)
print(res.pca)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 2000 individuals, described by 240 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Valeurs propres

Pour le choix du nombre de d'axe principale, nous pouvons utiliser la regle de kaiser et la regle de Coude. Nous allons donc afficher le nombre d'axe ou les valeurs propres sont superieur a 1

Comme décrit, les valeurs propres mesurent la quantité de variance expliquée par chaque axe principal. Les valeurs propres sont grandes pour les premiers axes et petits pour les axes suivants. Autrement dit, les premiers axes correspondent aux directions portant la quantité maximale de variation contenue dans le jeu de données.

Nous examinons les valeurs propres pour déterminer le nombre de composantes principales à prendre en considération en fonction de la regle de kaiser et de coude

Nous visualisons le tableau seulement pour les lignes dont les valeurs propres sont supperieur a 1

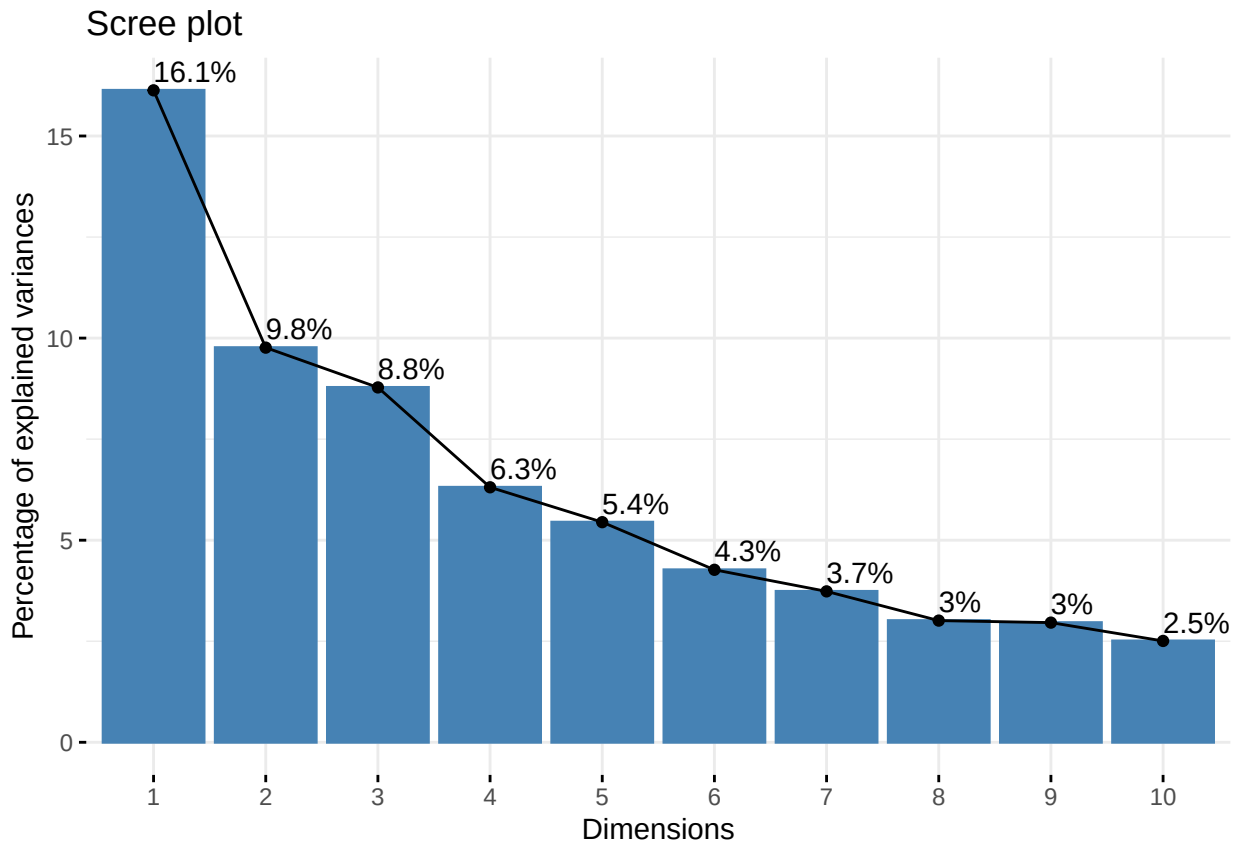
```
dt=as.data.frame(res.pca$eig)
dt[dt$eigenvalue >= 1, ]
```

```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   38.711392             16.1297469                           16.12975
```

```
## comp 2   23.434178            9.7642407                   25.89399
## comp 3   21.070930            8.7795542                   34.67354
## comp 4   15.141867            6.3091113                   40.98265
## comp 5   13.071404            5.4464185                   46.42907
## comp 6   10.242836            4.2678482                   50.69692
## comp 7    8.961783            3.7340763                   54.43100
## comp 8    7.225731            3.0107211                   57.44172
## comp 9    7.106756            2.9611483                   60.40287
## comp 10   6.017829            2.5074289                   62.91029
## comp 11   4.717092            1.9654551                   64.87575
## comp 12   4.424831            1.8436797                   66.71943
## comp 13   4.089342            1.7038925                   68.42332
## comp 14   3.732242            1.5551010                   69.97842
## comp 15   3.459768            1.4415698                   71.41999
## comp 16   3.146119            1.3108830                   72.73088
## comp 17   2.940066            1.2250276                   73.95590
## comp 18   2.721420            1.1339250                   75.08983
## comp 19   2.481963            1.0341513                   76.12398
## comp 20   2.400511            1.0002129                   77.12419
## comp 21   2.135865            0.8899439                   78.01414
## comp 22   1.946739            0.8111411                   78.82528
## comp 23   1.897935            0.7908061                   79.61608
## comp 24   1.814519            0.7560496                   80.37213
## comp 25   1.635836            0.6815984                   81.05373
## comp 26   1.467939            0.6116412                   81.66537
## comp 27   1.377241            0.5738506                   82.23922
## comp 28   1.231813            0.5132555                   82.75248
## comp 29   1.217846            0.5074358                   83.25991
## comp 30   1.144235            0.4767645                   83.73668
## comp 31   1.120121            0.4667173                   84.20340
## comp 32   1.112510            0.4635460                   84.66694
## comp 33   1.013212            0.4221715                   85.08911
```

La règle de Kaiser repose sur une idée simple. Dans une ACP normée, la somme des valeurs propres étant égale au nombre de variables, leur moyenne vaut 1. Nous considérons par conséquent qu'un axe est intéressant si sa valeur propre est supérieure 1 Nous avons ici 33 valeurs propres superieur a 1, ce qui rend l'interpretation beaucoup plus compliquer.

```
fviz_eig(res.pca, addlabels = TRUE)
```



En règle générale, le coude est très marqué lorsque nous traitons des variables fortement corrélées. Lorsqu'elles le sont faiblement ou lorsqu'il y a des blocs de variables corrélées, plutôt qu'une solution unique « évidente », nous devons faire face à plusieurs scénarios

Distribution de l'inertie

L'inertie des axes factoriels indique d'une part si les variables sont structurées et suggère d'autre part le nombre judicieux de composantes principales à étudier.

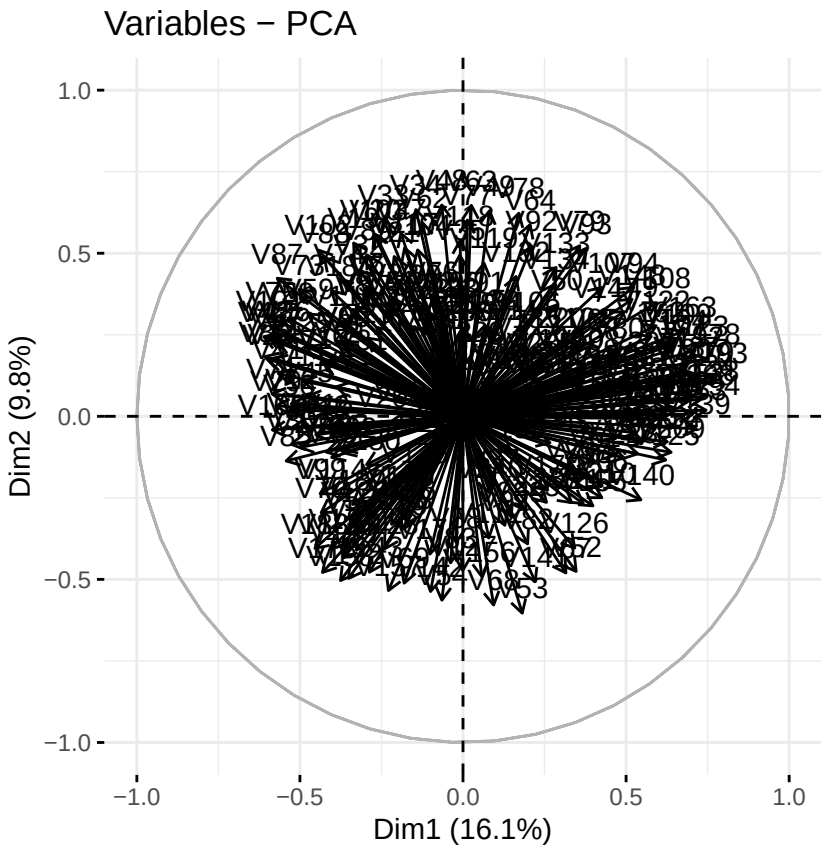
Les 2 premiers axes de l' analyse expriment 25.89% de l'inertie totale du jeu de données ; cela signifie que 25.89% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan. C'est un pourcentage relativement moyen, et le premier plan représente donc seulement une part de la variabilité contenue dans l'ensemble du jeu de données actif.

Du fait de ces observations, il serait alors probablement nécessaire de considérer également les dimensions supérieures ou égales à la troisième dans l'analyse.

Cercle de corrélation

La corrélation entre une variable et une composante principale est utilisée comme coordonnées de la variable sur la composante principale.

```
fviz_pca_var(res.pca)
```



En prenant les variables dont les contibutions sont superieur a 0.5 sur l'une des deux axes, on obtient le cercle de correlation ci-dessus. Si nous devons retenir toutes les 33 axes, et faire la meme representation, il nous sera tres difficile de l'interpreter.

Pour ce faire nous allons faire une classifiaction avec la ''methode de ward" et ''methode K-Mean" sur les coordonnées des individus pour les rasembler en different classe.

Classification par la methode K-Mean

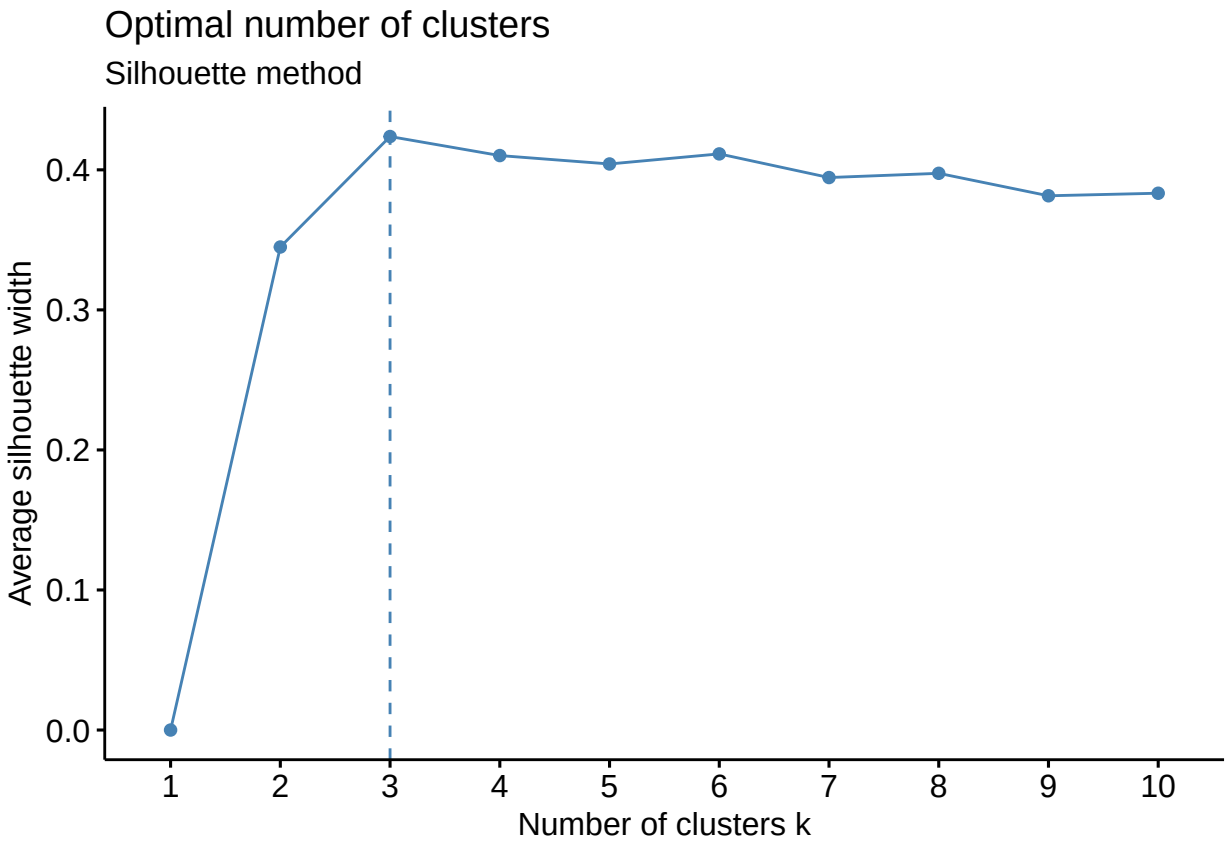
Deux principaux axes

Dans cette partie nous utilisons les deux principaux axe pour la classification des individus, sachant que les 2 premiers axes de l' analyse expriment 25.89% de l'inertie totale du jeu de données ; cela signifie que 25.89% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan

Determimination du nombre de Clusters optimaux

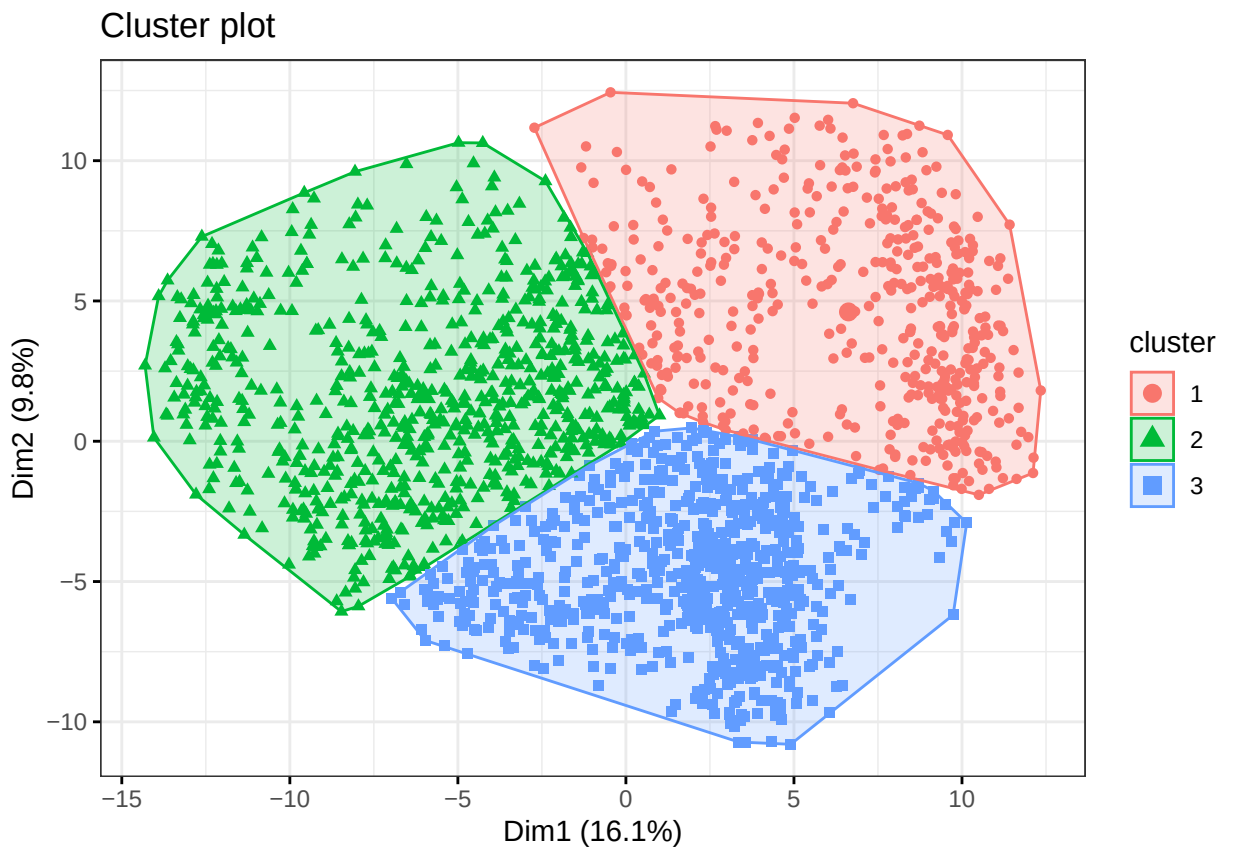
```
## K-Mean

res.pca <- PCA(mfeat.pix,ncp=2, graph = FALSE,scale.unit = TRUE)
ncp=res.pca$ind$coord
df <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df, kmeans,iter.max = 50,nstart = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



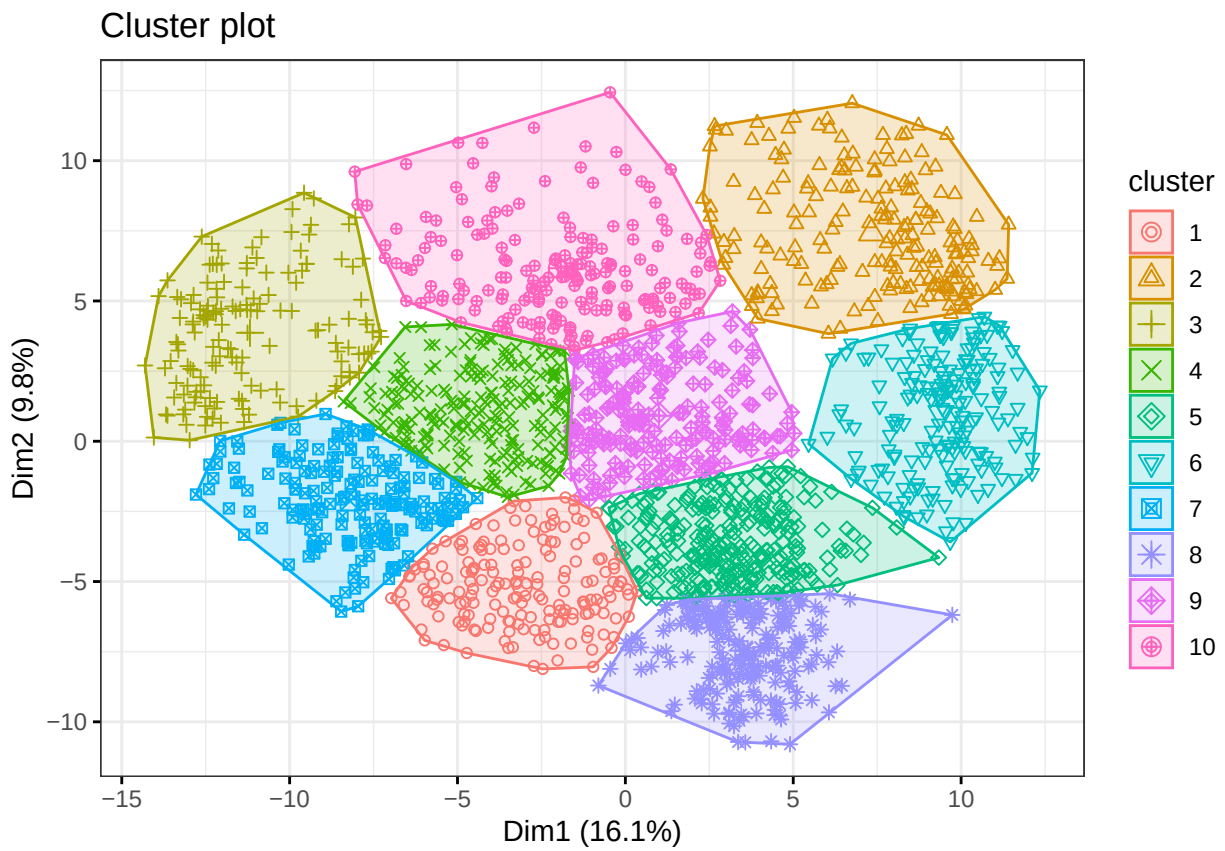
Lorsque nous prenons les deux axes principaux, on se retrouve avec 3 clusters optimaux.

```
km.res <- kmeans(df, 3,iter.max = 1000, nstart = 50)

fviz_cluster(km.res, data = mfeat.pix,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Dans la figure précedente, nous avons representés nos individus en 3 classe, vu le nombre de classe optimal.

```
km.res <- kmeans(df, 10,iter.max = 1000, nstart = 50)

fviz_cluster(km.res, data = mfeat.pix,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Nous resemblons nos individus sous 10 cluster, ceci est satisfessant dans car nous avons 10 classe de chiffre dans notre ensemble de données

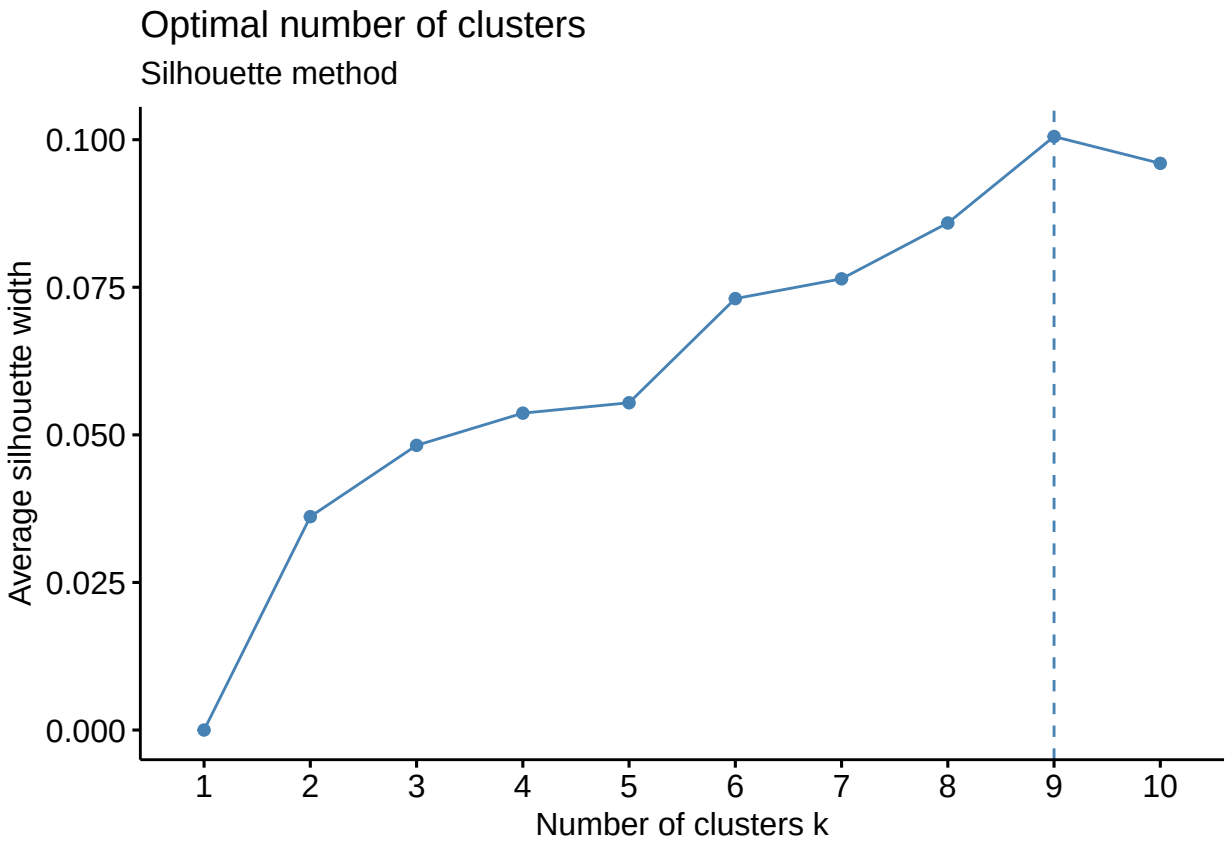
23 principaux axes

Dans cette partie nous utilisons les 33 principaux axe pour la classification des individus, sachant que les 33 premiers axes de l' analyse expriment 85.08 de l'inertie totale du jeu de données ; cela signifie que 85.08% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ces plans

Determimination du nombre

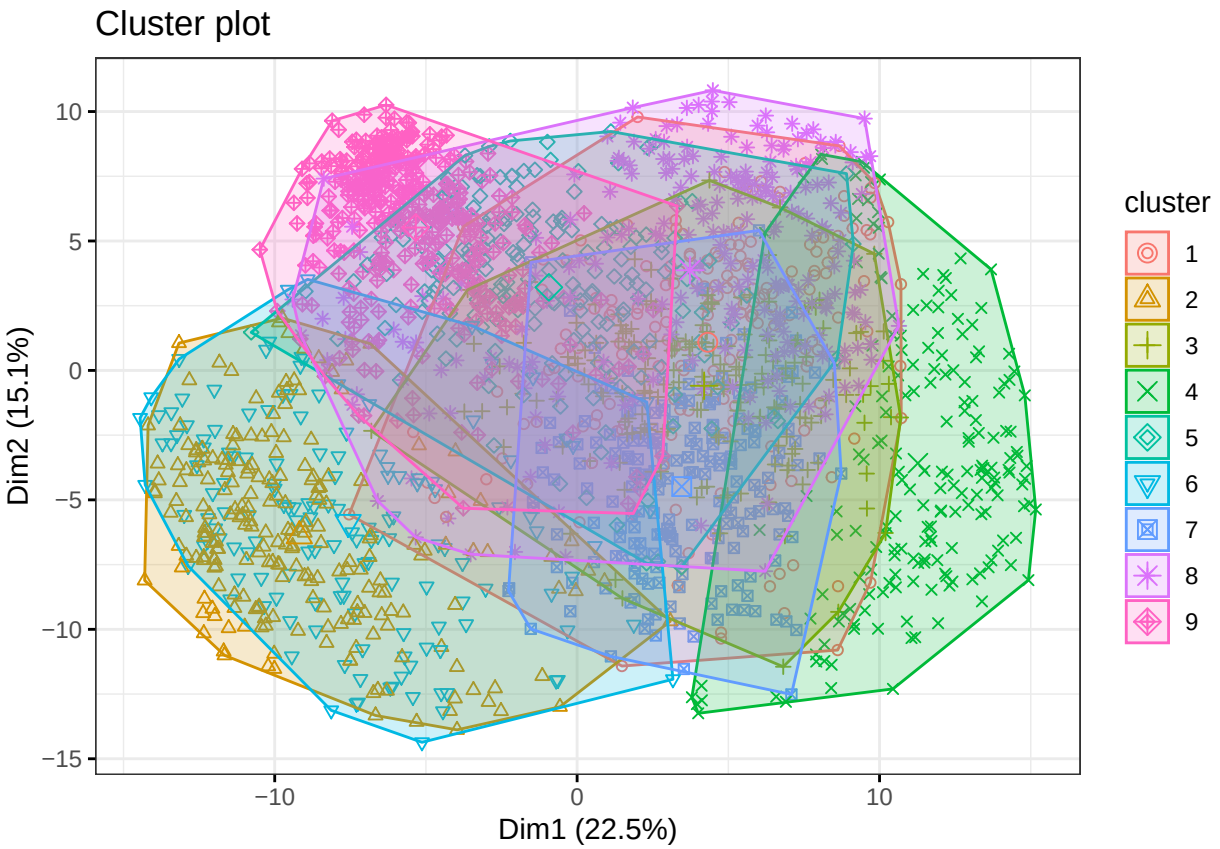
```
## K-Mean

res.pca <- PCA(mfeat.pix,ncp=23, graph = FALSE,scale.unit = TRUE)
ncp=res.pca$ind$coord
df <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



En Prenant les 23 axes principaux, on obtient 9 clusters optimaux, ce qui corespond bien au nombre de classe initiale dans le jeux de données

```
km.res <- kmeans(df23, 9,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = mfeat.fac,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Cette representation est difficile a interpreter, car nous avons 33 axes principal. Meme si elle donne plus d'informations.

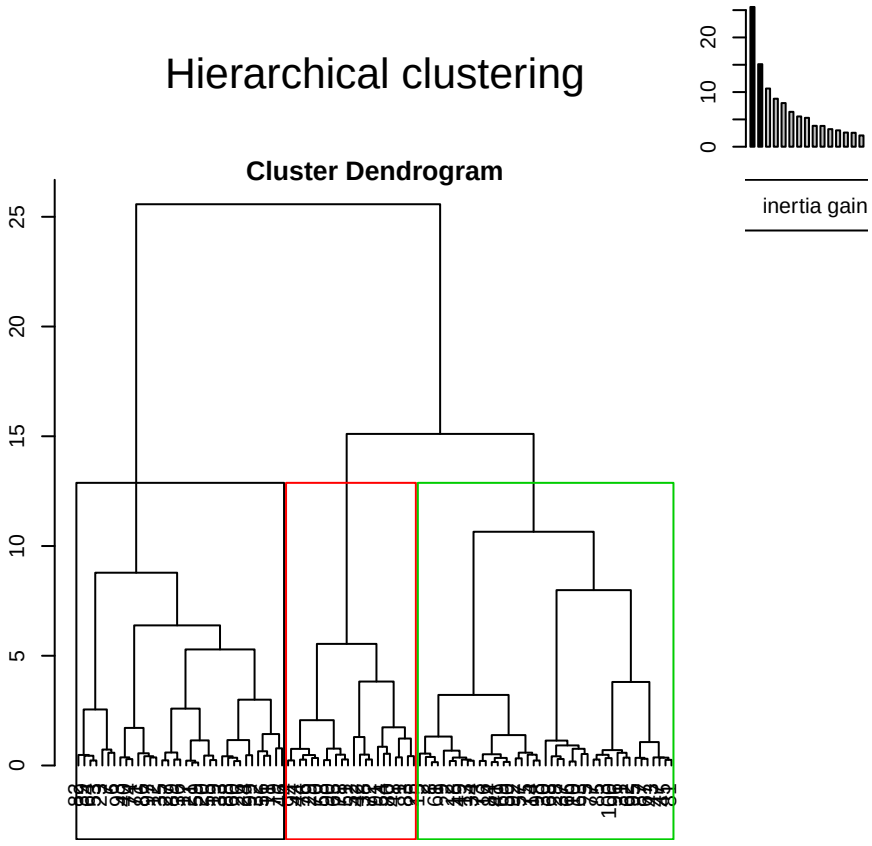
Classiffication avec l'ACP suivi de l'algorithme de Ward.

Nous utilison Dans cette partie Une ACP suivi de la classification hiérachique sur le jeu de donnée res.PCA

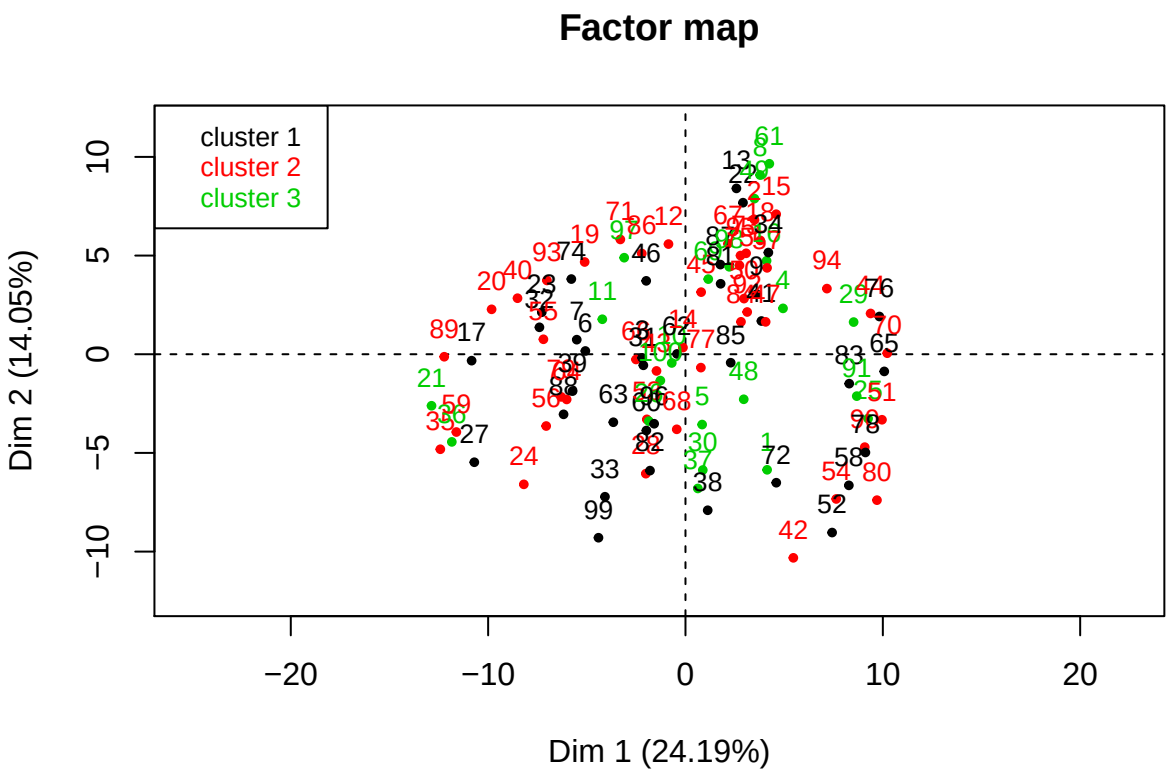
```
res.PCA= PCA(mfeat.pix,ncp=33, graph = FALSE,scale.unit = TRUE) # ACP en conservant 33 dimentions
hc=HCPC(res.PCA,kk=100,description = F,graph = F) # Classification avec prétraitement par Kmean avec kk
```

```
## Warning in HCPC(res.PCA, kk = 100, description = F, graph = F): No consolidation
## has been done after the hierarchical clustering since kk is different from Inf
## (see help for more details)
```

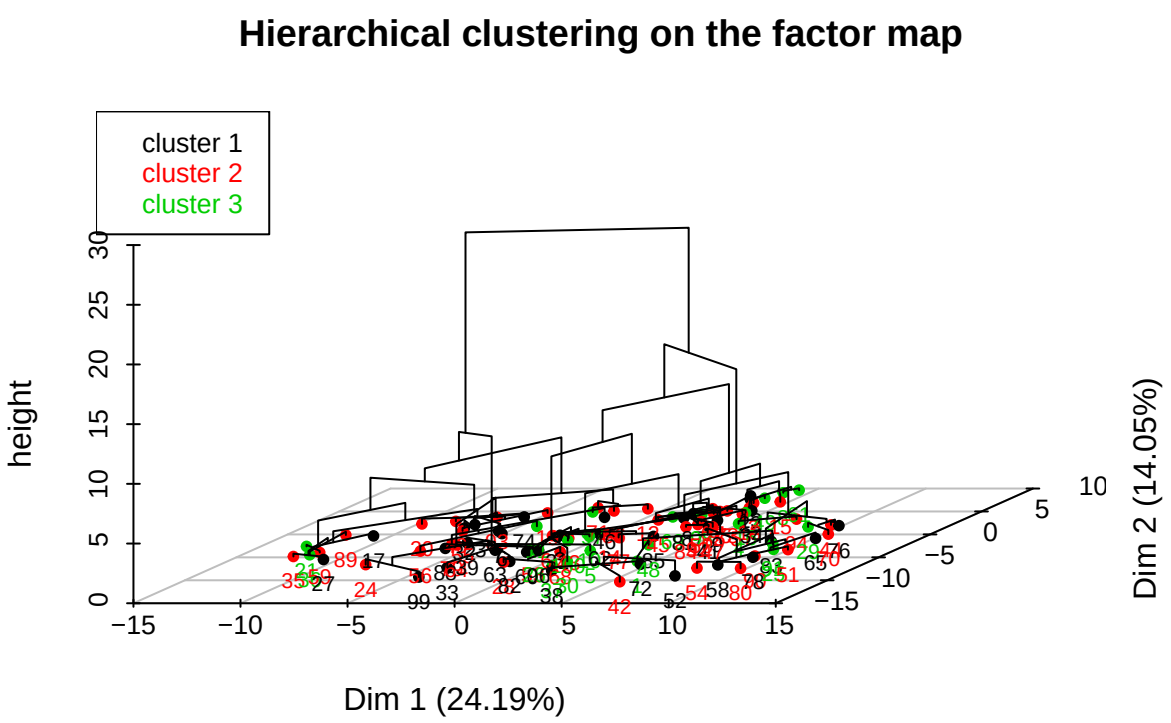
```
plot(hc,choice='tree') # Graphe de l'abre
```




```
plot(hc,choice='map',draw.tree=F) # Plan de l'ACP avec les abres
```




```
plot(hc,choice='3D.map') # Plan de l'ACP avec les abres
```




```
#catdes(hc$data.clust,ncol((hc$data.clust)))  ## caracterisation des classes
```

Classiffication avec l'algorithme de Ward.

Nous utilisons les 33 axes principales, car ayant tous des valeurs propres superieur a 1.

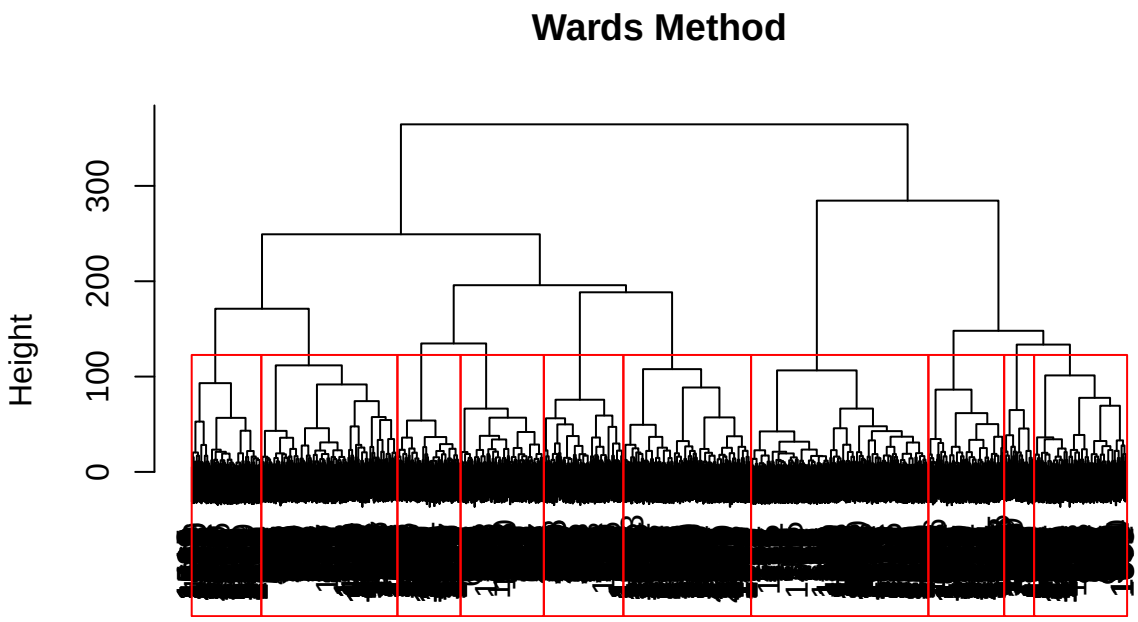
```
res.pca <- PCA(mfeat.fac,ncp=33, graph = FALSE,scale.unit = TRUE)
ncp=res.pca$ind$coord
```

Dans cette partie nous utilisons les cordonnées des individu de l'ACP pour faire la classification avec l'algorithne de ward

```
## standarisation des donnees
new_data_scaled <- ncp
new_data_scaled.dist <- dist(new_data_scaled)      # Calcul de distance
new_data_scaled.distT <- dist(t(new_data_scaled)) # calcul de distance
```

```
# effectuer un regroupement hiérarchique des observations (variables) en utilisant la méthode de Ward
ward = hclust(new_data_scaled.distT,method='ward.D2')
wardT = hclust(new_data_scaled.dist,method='ward.D2')
```

```
plot(wardT, main = 'Wards Method',xlab =" ", sub ="")
rect.hclust(wardT, k =10, border = 'red') ## selecting three clusters
```



Ce cluster nous permet de voir 10 classe.

Conclusion pour la donnée mfeat.fact

Dans cette partie de notre etude nous avons pu momtrer que nos 2000 ligne pouvais etre regrouper en 10 classe a partir des methode de l'ACP en reduissant la dimension a un nombre acceptable pour ensuite faire de la Classification.

Les KMean permet de d'avoir 3 clusters avec deux axes principaux et 9 clusters avec 33

DOnnee Entière (Donnee Concatené)

concatenir l'ensembles de donnée pour en faire un data Set

```
data= cbind(mfeat.fou,mfeat.kar)
data= cbind(data,mfeat.pix)
data= cbind(data,mfeat.zer)
data= cbind(data,mfeat.fac )
data= cbind(data,mfeat.zer)

data=data[,c(1:649)]
```

```
res.pca <- PCA(data,ncp=79, graph = FALSE)
print(res.pca)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 2000 individuals, described by 649 variables
```

```
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Valeurs propres

Pour le choix du nombre de d'axe principale, nous pouvons utiliser la regle de kaiser et la regle de Coude. Nous allons donc afficher le nombre d'axe ou les valeurs propres sont superieur a 1

Comme décrit, les valeurs propres mesurent la quantité de variance expliquée par chaque axe principal. Les valeurs propres sont grandes pour les premiers axes et petits pour les axes suivants. Autrement dit, les premiers axes correspondent aux directions portant la quantité maximale de variation contenue dans le jeu de données.

Nous examinons les valeurs propres pour déterminer le nombre de composantes principales à prendre en considération en fonction de la regle de kaiser et de coude

Nous visualisons le tableau seulement pour les lignes dont les valeurs propres sont supperieur a 1

```
dt=as.data.frame(res.pca$eig)
dt[dt$eigenvalue >= 1, ]
```

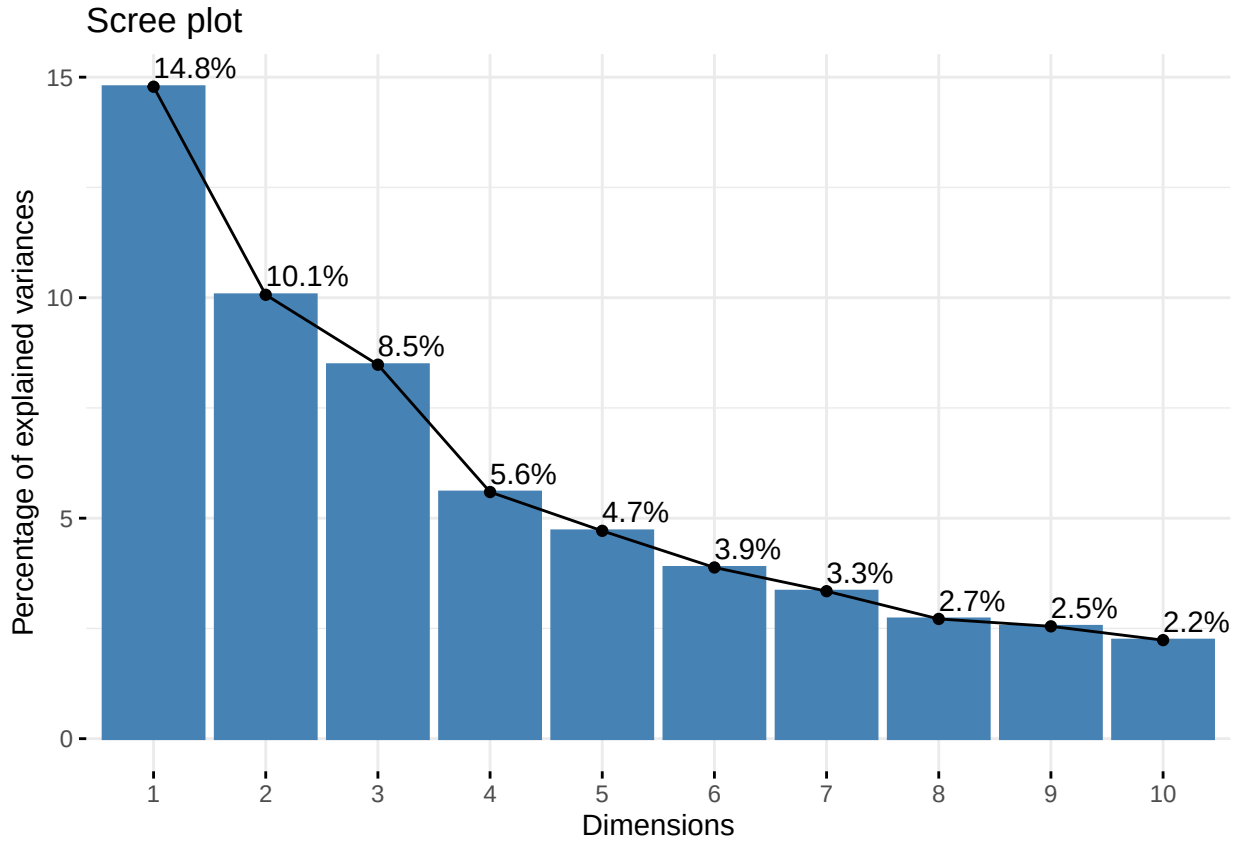
```
##         eigenvalue percentage of variance cumulative percentage of variance
## comp 1   95.947666             14.7839240                           14.78392
## comp 2   65.326162             10.0656644                           24.84959
## comp 3   55.035814              8.4800946                           33.32968
## comp 4   36.297778              5.5928779                           38.92256
## comp 5   30.586075              4.7128005                           43.63536
## comp 6   25.198743              3.8827030                           47.51806
## comp 7   21.707088              3.3446977                           50.86276
## comp 8   17.624372              2.7156198                           53.57838
## comp 9   16.535347              2.5478193                           56.12620
## comp 10  14.503817              2.2347947                           58.36100
## comp 11  11.230990              1.7305069                           60.09150
## comp 12  10.439437              1.6085419                           61.70004
## comp 13   9.012207              1.3886298                           63.08867
## comp 14   7.947132              1.2245196                           64.31319
## comp 15   7.757980              1.1953744                           65.50857
## comp 16   7.356436              1.1335032                           66.64207
## comp 17   6.581667              1.0141244                           67.65620
## comp 18   6.201224              0.9555045                           68.61170
```

```
## comp 19   5.477648            0.8440135                 69.45571
## comp 20   4.951262            0.7629063                 70.21862
## comp 21   4.711509            0.7259644                 70.94458
## comp 22   4.419383            0.6809527                 71.62554
## comp 23   4.314188            0.6647440                 72.29028
## comp 24   4.038741            0.6223021                 72.91258
## comp 25   3.740183            0.5762994                 73.48888
## comp 26   3.504305            0.5399545                 74.02884
## comp 27   3.491302            0.5379510                 74.56679
## comp 28   3.299214            0.5083535                 75.07514
## comp 29   3.158103            0.4866107                 75.56175
## comp 30   3.084679            0.4752973                 76.03705
## comp 31   2.992861            0.4611496                 76.49820
## comp 32   2.882252            0.4441067                 76.94231
## comp 33   2.807183            0.4325398                 77.37485
## comp 34   2.562138            0.3947824                 77.76963
## comp 35   2.495870            0.3845717                 78.15420
## comp 36   2.441678            0.3762216                 78.53042
## comp 37   2.328351            0.3587597                 78.88918
## comp 38   2.205901            0.3398924                 79.22907
## comp 39   2.198830            0.3388027                 79.56788
## comp 40   2.176923            0.3354273                 79.90330
## comp 41   2.096097            0.3229733                 80.22628
## comp 42   2.047085            0.3154214                 80.54170
## comp 43   1.980798            0.3052077                 80.84691
## comp 44   1.896723            0.2922532                 81.13916
## comp 45   1.854597            0.2857622                 81.42492
## comp 46   1.824058            0.2810567                 81.70598
## comp 47   1.774627            0.2734402                 81.97942
## comp 48   1.721088            0.2651907                 82.24461
## comp 49   1.674620            0.2580308                 82.50264
## comp 50   1.665408            0.2566114                 82.75925
## comp 51   1.616708            0.2491076                 83.00836
## comp 52   1.600298            0.2465790                 83.25494
## comp 53   1.578366            0.2431997                 83.49814
## comp 54   1.544047            0.2379117                 83.73605
## comp 55   1.520811            0.2343314                 83.97038
## comp 56   1.446780            0.2229245                 84.19331
## comp 57   1.444048            0.2225036                 84.41581
## comp 58   1.425090            0.2195825                 84.63539
## comp 59   1.395022            0.2149495                 84.85034
## comp 60   1.359614            0.2094936                 85.05983
## comp 61   1.322969            0.2038472                 85.26368
## comp 62   1.303501            0.2008475                 85.46453
## comp 63   1.276884            0.1967463                 85.66128
## comp 64   1.267759            0.1953404                 85.85662
## comp 65   1.229814            0.1894937                 86.04611
## comp 66   1.206279            0.1858674                 86.23198
## comp 67   1.184377            0.1824926                 86.41447
## comp 68   1.168269            0.1800107                 86.59448
## comp 69   1.139060            0.1755100                 86.76999
## comp 70   1.130936            0.1742582                 86.94425
## comp 71   1.105303            0.1703087                 87.11456
## comp 72   1.065168            0.1641244                 87.27868
```

```
## comp 73   1.060338              0.1633802                87.44206
## comp 74   1.041065              0.1604106                87.60247
## comp 75   1.036251              0.1596689                87.76214
## comp 76   1.015406              0.1564570                87.91860
## comp 77   1.001281              0.1542806                88.07288
```

La règle de Kaiser repose sur une idée simple. Dans une ACP normée, la somme des valeurs propres étant égale au nombre de variables, leur moyenne vaut 1. Nous considérons par conséquent qu'un axe est intéressant si sa valeur propre est supérieure 1 Nous avons ici 79 valeurs propres superieur a 1, ce qui rend l'interpretation beaucoup plus compliquer.

```
fviz_eig(res.pca, addlabels = TRUE)
```



En règle générale, le coude est très marqué lorsque nous traitons des variables fortement corrélées. Lorsqu'elles le sont faiblement ou lorsqu'il y a des blocs de variables corrélées, plutôt qu'une solution unique « évidente », nous devons faire face à plusieurs scénarios

Distribution de l'inertie

L'inertie des axes factoriels indique d'une part si les variables sont structurées et suggère d'autre part le nombre judicieux de composantes principales à étudier.

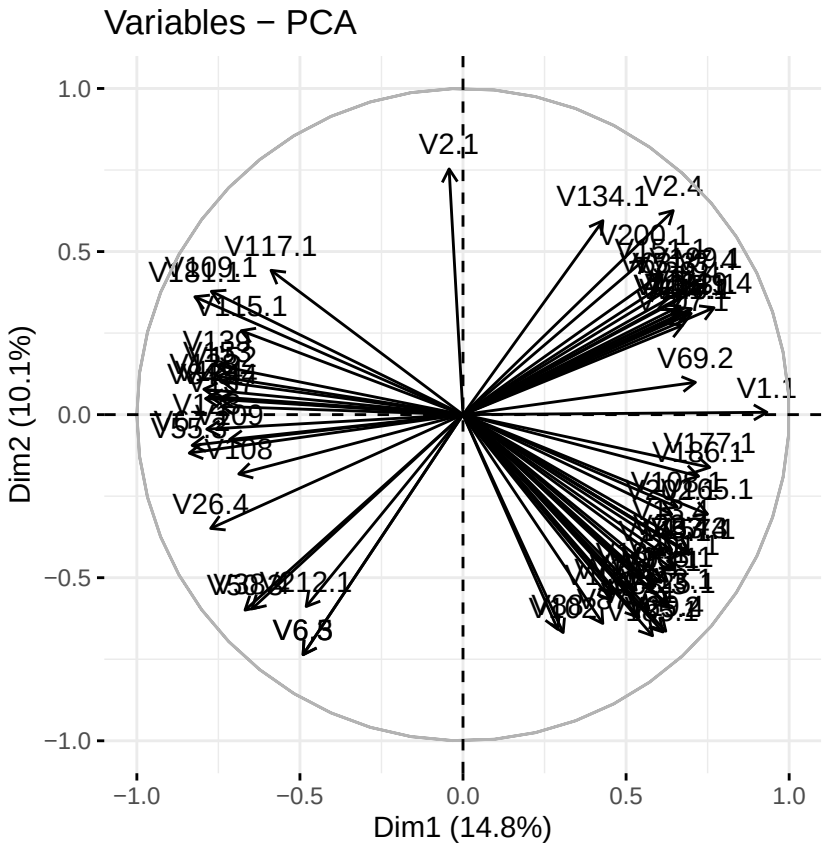
Les 2 premiers axes de l' analyse expriment 24.72% de l'inertie totale du jeu de données ; cela signifie que 24.72% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan. C'est un pourcentage relativement faible, et le premier plan représente donc seulement une part de la variabilité contenue dans l'ensemble du jeu de données actif.

Du fait de ces observations, il serait alors probablement nécessaire de considérer également les dimensions supérieures ou égales à la troisième dans l'analyse.

Cercle de corrélation

La corrélation entre une variable et une composante principale est utilisée comme coordonnées de la variable sur la composante principale.

```
fviz_pca_var(res.pca, select.var = list(cos2 = 0.5))
```



En prenant les variables dont les contibutions sont superieur a 0.5 sur l'une des deux axes, on obtient le cercle de correlation ci-dessus. Si nous devons retenir toutes les 23 axes, et faire la meme representation, il nous sera tres difficile de l'interpreter.

Cependant on peut voir que les variables sont tres bien correle a l'axe 1

Pour ce faire nous allons faire une classifiaction avec la ''methode de ward" et ''methode K-Mean" sur les coordonnées des individus pour les rasembler en different classe.

Classification par la methode K-Mean

Deux principaux axes

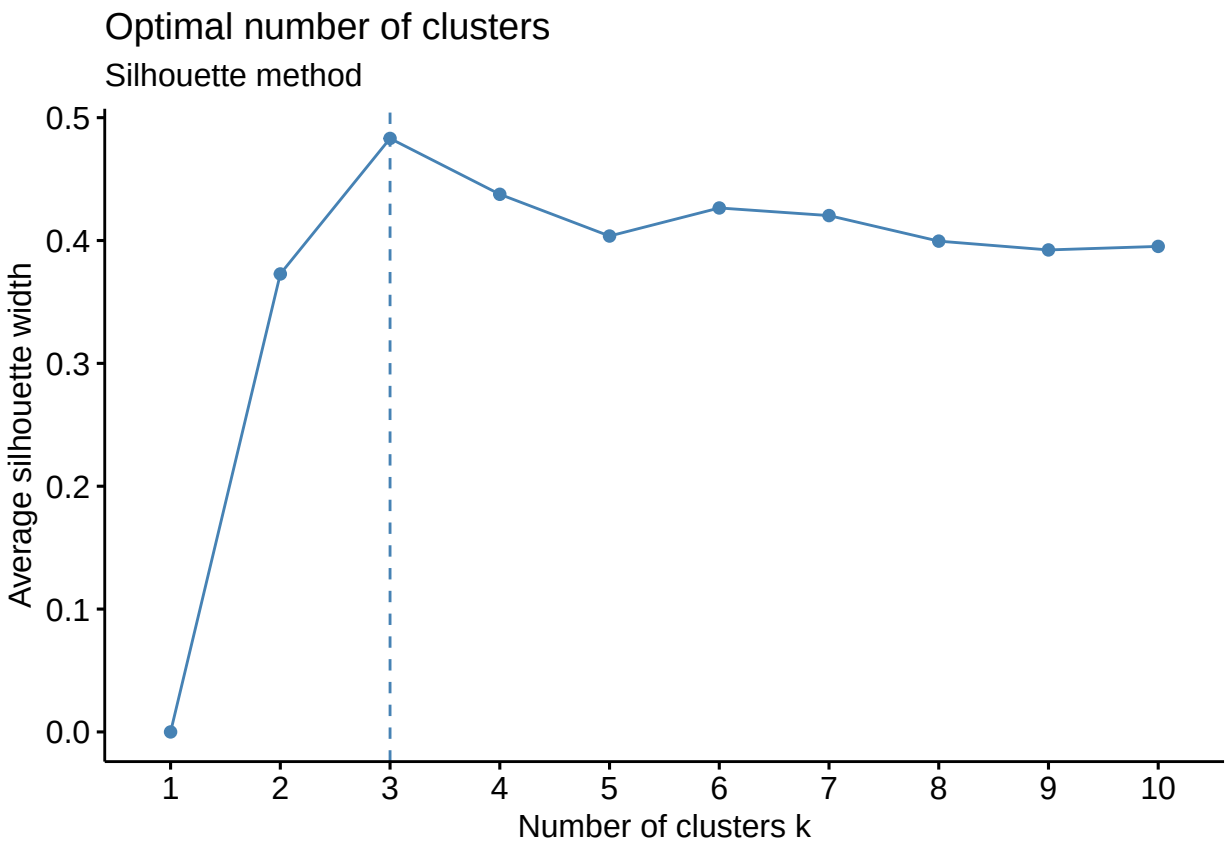
Dans cette partie nous utilisons les deux principaux axe pour la classification des individus, sachant que les 2 premiers axes de l' analyse expriment 24.72% de l'inertie totale du jeu de données ; cela signifie que 24.72% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ce plan

Determimination du nombre de Clusters optimaux

```
## K-Mean

res.pca <- PCA(data,ncp=2, graph = FALSE,scale.unit = TRUE)
```

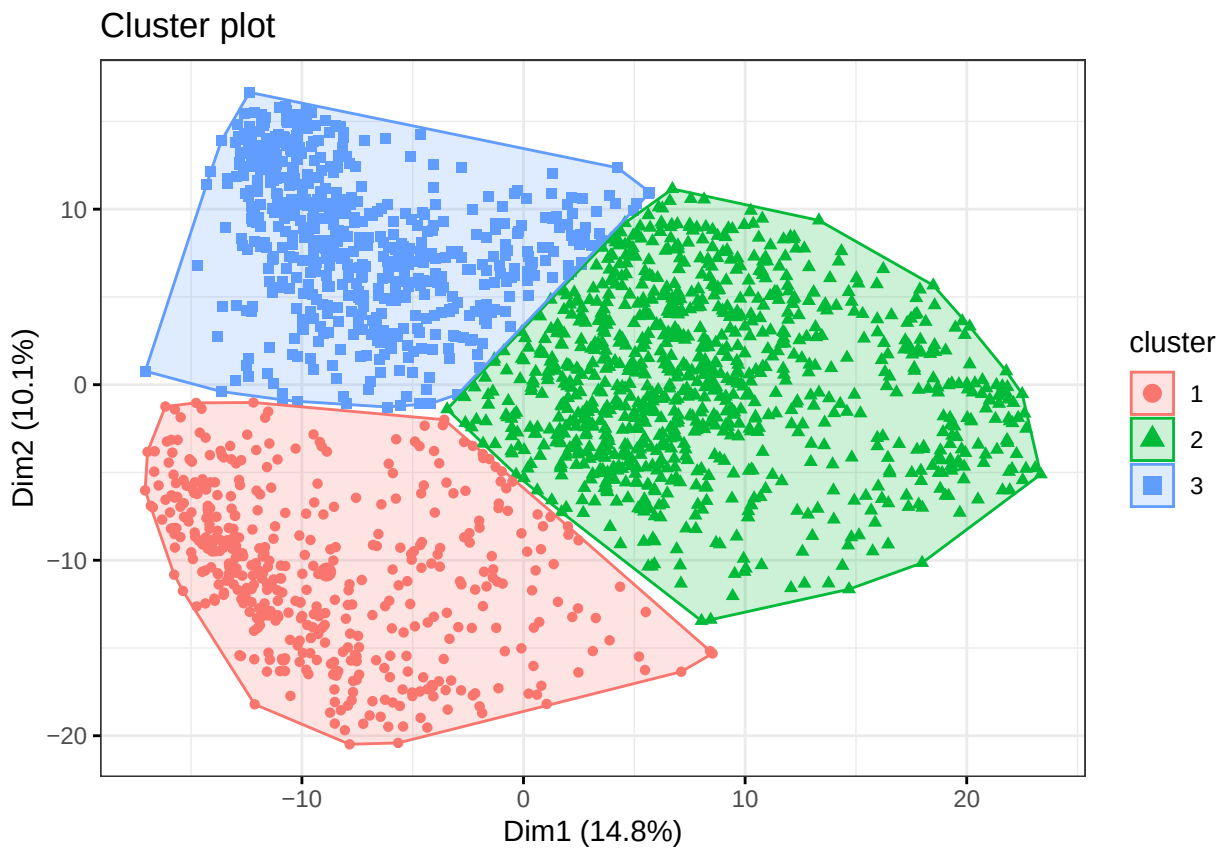
```r
ncp=res.pca$ind$coord
df <- scale(ncp) # Scaling the data
```

```r
# Silhouette method
fviz_nbclust(df, kmeans,iter.max = 50, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```



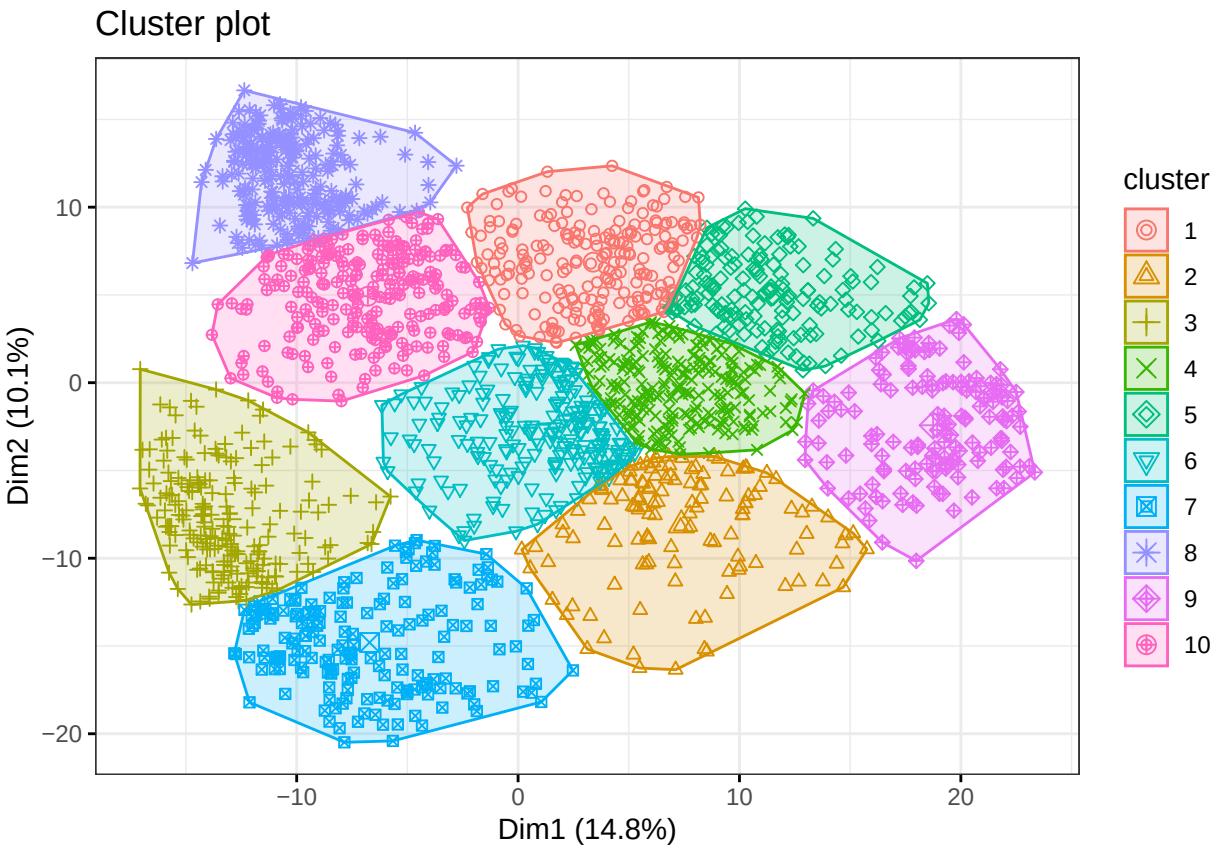
Lorsque nous prenons les deux axes principaux, on se retrouve avec 3 clusters optimaux. Cependand on peut bien voir que en prenant 10 cluster on s'eloigne pas de la performance que apportent les 6 cluster

```r
km.res <- kmeans(df, 3,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = data,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Dans la figure précedente, nous avons representés nos individus en 3 classe, vu le nombre de classe optimal.

```
km.res <- kmeans(df, 10,iter.max = 10000, nstart = 500)

fviz_cluster(km.res, data = data,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Nous resemblons nos individus sous 10 cluster, ceci est satisfessant dans car nous avons 10 classe de chiffre dans notre ensemble de données

79 principaux axes

Dans cette partie nous utilisons les 79 principaux axe pour la classification des individus, sachant que les 79 premiers axes de l' analyse expriment 88.67% de l'inertie totale du jeu de données ; cela signifie que 88.67% de la variabilité totale du nuage des individus (ou des variables) est représentée dans ces plans

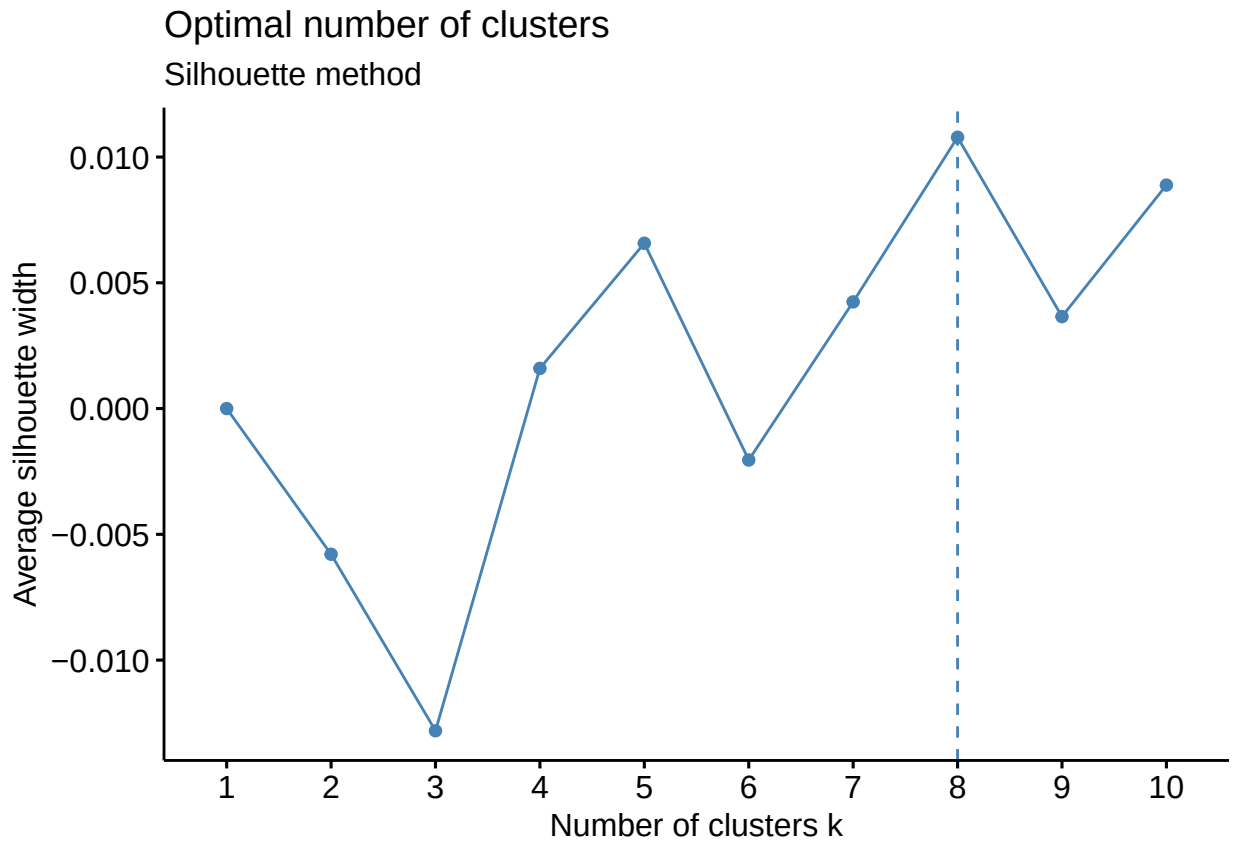
Determimination du nombre optimal de cluster

```
## K-Mean

res.pca <- PCA(data,ncp=79, graph = FALSE,scale.unit = TRUE)
ncp=res.pca$ind$coord
df23 <- scale(ncp) # Scaling the data
```

```
# Silhouette method
fviz_nbclust(df23, kmeans,iter.max = 50,nstart = 100, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```

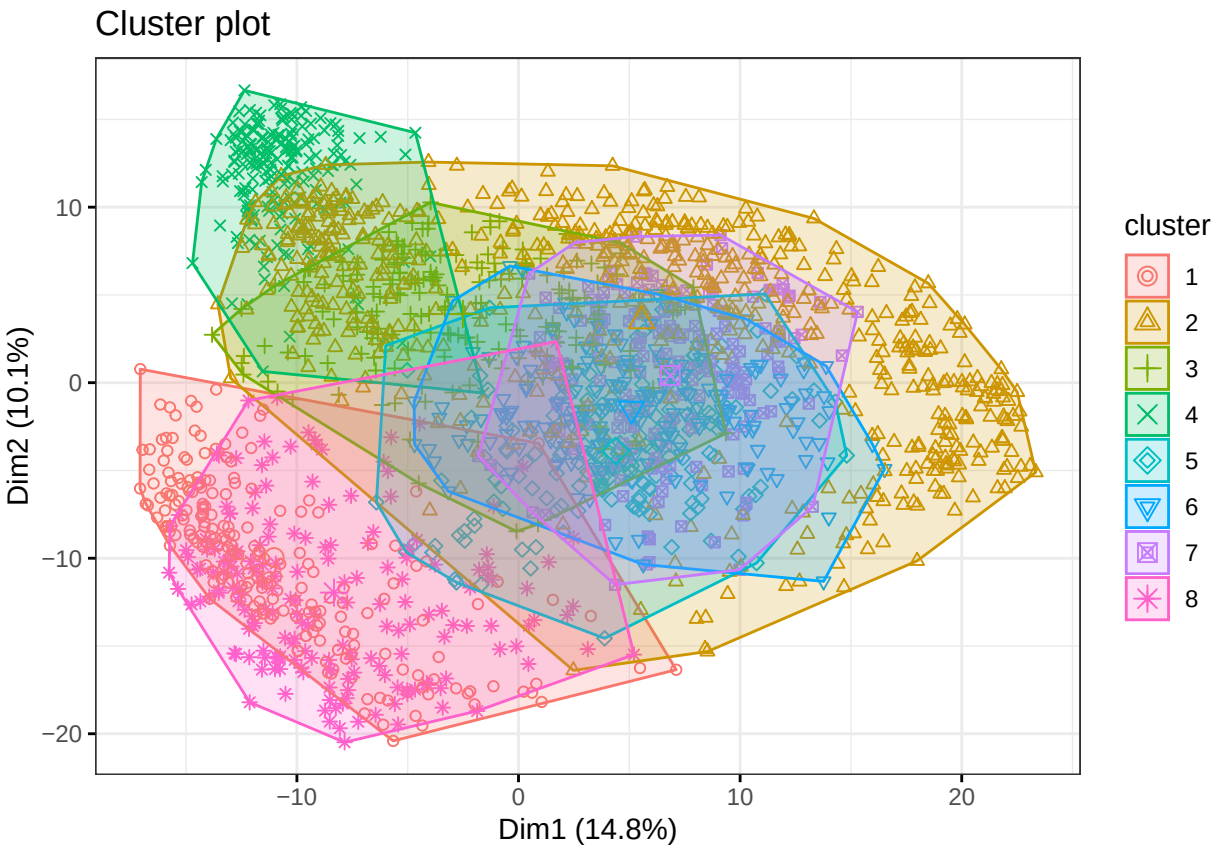
```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 100000)
```



En Prenant les 79 axes principaux, on obtient 8 clusters optimaux

```r
km.res <- kmeans(df23, 8,iter.max = 10000, nstart = 50)

fviz_cluster(km.res, data = data,
             #palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)
```



Cette representation est difficile a interpreter, car nous avons 23 axes principal. Meme si elle donne plus d'informations.

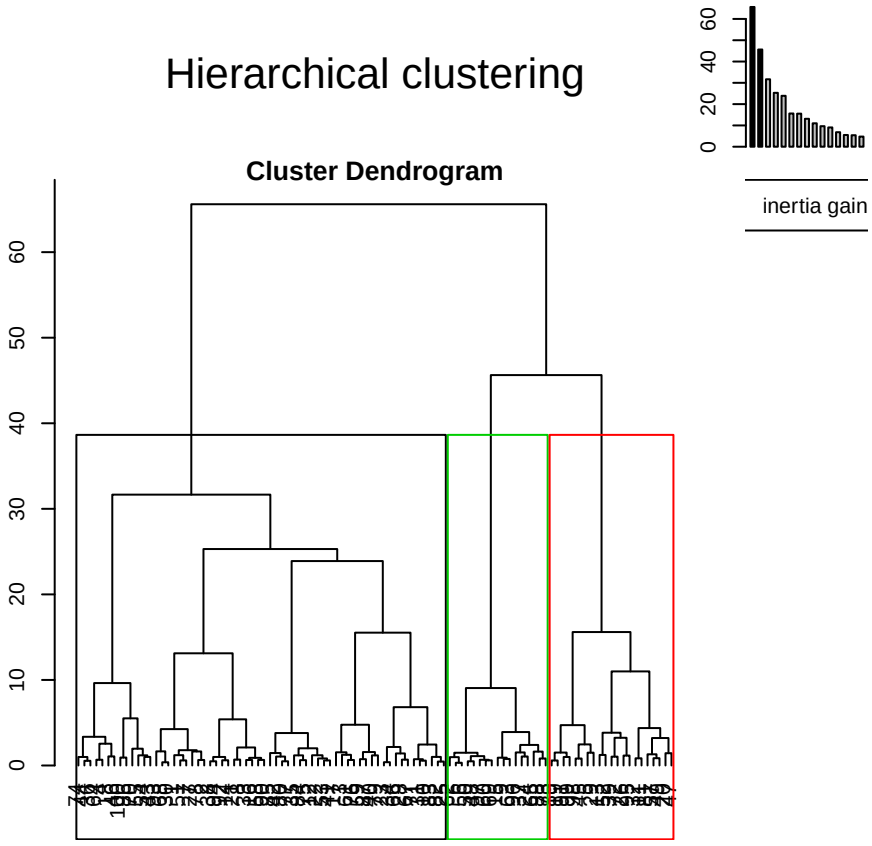
Classiffication avec l'algorithme de Ward.

Nous utilison Dans cette partie Une ACP suivi de la classification hiérachique sur le jeu de donnée res.PCA

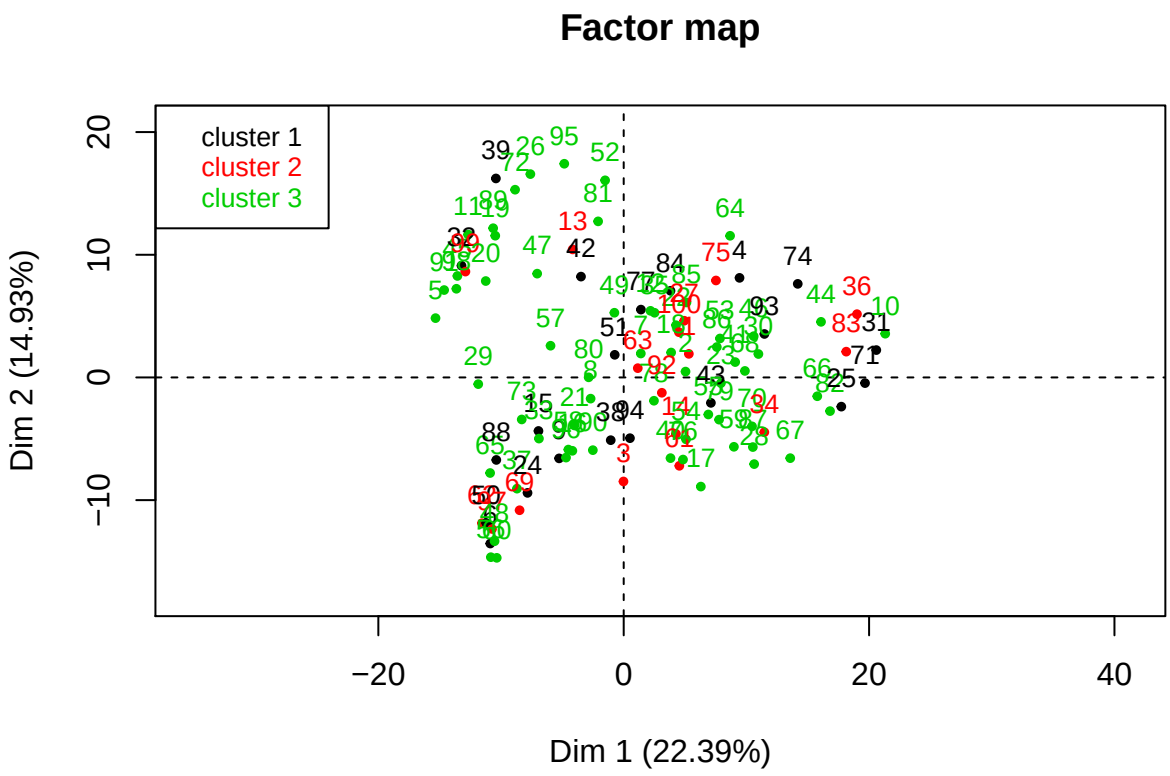
```
res.PCA= PCA(data,ncp=79, graph = FALSE,scale.unit = TRUE) # ACP en conservant 23 dimentions
hc=HCPC(res.PCA,kk=100,description = F,graph = F) # Classification avec prétraitement par Kmean avec kk
```

```
## Warning in HCPC(res.PCA, kk = 100, description = F, graph = F): No consolidation
## has been done after the hierarchical clustering since kk is different from Inf
## (see help for more details)
```

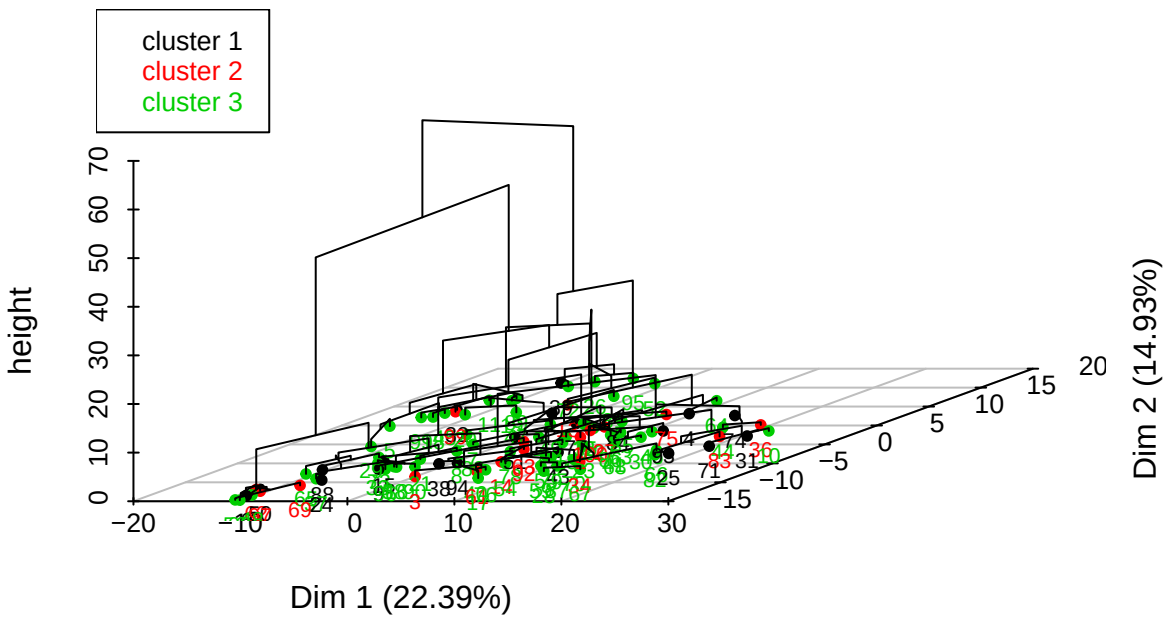
```
plot(hc,choice='tree') # Graphe de l'abre
```




```
plot(hc,choice='map',draw.tree=F) # Plan de l'ACP avec les abres
```




```
plot(hc,choice='3D.map') # Plan de l'ACP avec les abres
```




```
#catdes(hc$data.clust,ncol((hc$data.clust)))  ## caracterisation des classes
```

Classiffication avec l'algorithme de Ward.

Nous utilisons les 79 axes principales, car ayant tous des valeurs propres superieur a 1.

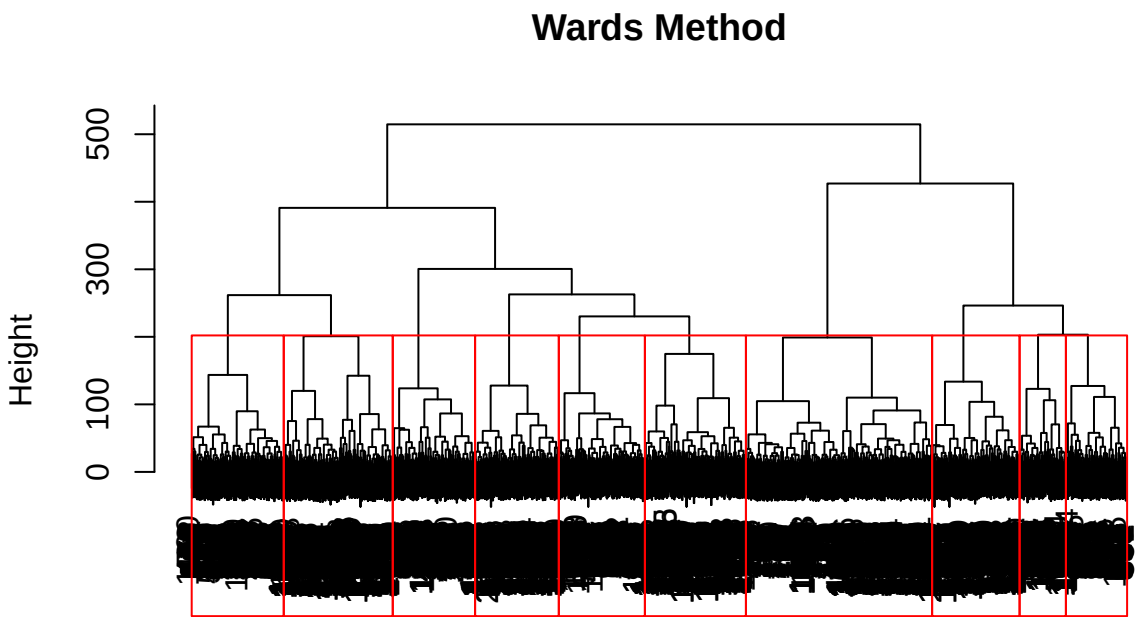
```
res.pca <- PCA(data,ncp=79, graph = FALSE,scale.unit = TRUE)
ncp=res.pca$ind$coord
```

Dans cette partie nous utilisons les cordonnées des individu de l'ACP pour faire la classification avec l'algorithne de ward

```
## standarisation des donnees
new_data_scaled <- ncp
new_data_scaled.dist <- dist(new_data_scaled)      # Calcul de distance
new_data_scaled.distT <- dist(t(new_data_scaled)) # calcul de distance
```

```
# effectuer un regroupement hiérarchique des observations (variables) en utilisant la méthode de Ward
ward = hclust(new_data_scaled.distT,method='ward.D2')
wardT = hclust(new_data_scaled.dist,method='ward.D2')
```

```
plot(wardT, main = 'Wards Method',xlab =" ", sub ="")
rect.hclust(wardT, k =10, border = 'red') ## selecting three clusters
```



Ce cluster nous permet de voir 10 classe.

Conclusion pour la donnée entiere

Dans cette partie de notre étude nous avons pu momtrer que nos 2000 ligne pouvais etre regrouper en 10 classe a partir des methode de l'ACP en reduissant la dimension a un nombre acceptable pour ensuite faire de la Classification. Le KMean donne 3 cluster optimal pour les deux priemiers axe principaux et 8 cluster pour les 79 axes principaux. Donc plus on augmente le nombre de composantes principal, plus on obtient un resultat plus precis.

Conclusion

Pour les données total le nombre de cluster est de 8 avec la methode de KMean. 10 classes ont été egalement obtenue par la méthode de De graphe hiéarichique. De facon génerale, le nombre le nomnbre de cluster n'est pas si differents pour les 6 dataset. Cependant la reduction des dimentions par l'ACP a permis de classifier en groupe d'individu et de trouver des resultats pertinentes.