TP 6 : Classification

L'objectif est de définir une segmentation des individus en classes homogènes au regard des variables d'intérêt considérées. Les algorithmes de classification étudiés sont adaptés à des variables *quantitatives* ou des matrices de distances. Ce TP propose d'étudier deux approches sur des jeux de données différents : classification à partir des seules variables quantitatives ou classification à partir des scores issues d'une AFCM. D'autres approches sont envisageables sur des variables qualitatives qui nécessitent la définition d'une distance ou dissimilarité entre individus adaptée aux variables qualitatives. Mais, nécessitant la construction de la matrice $n \times n$ des distances des individus deux à deux, elles ne sont pas adaptées aux très grands tableaux.

1 Données `Températures`

Le jeu de données correspond à des relevés de températures moyens par mois dans des villes européennes. On cherche à regrouper ces villes en classes homogènes de sorte que les villes d'une même classe présentent des températures semblables tous les mois de l'année. *Une fois ces classes construites, il est important de caractériser les classes à partir des variables ou à partir d'individus particuliers.*

1.1 Classification sur variables quantitatives

1. Charger le package `FactoMineR`, les données `temperat.csv` et réaliser une première analyse statistique uni-dimensionnelle.

```
temperature0 <- read.table(file.choose(),header=TRUE, sep=";", dec=".",
                row.names=1)
```

2. Ne conserver que les capitales européennes et les températures par mois.

```
temperature <- temperature0[-(24:35),1:12]
# Si les variables considérées ne sont pas homogènes en variance, il est utile
# de réduire (standardiser) les variables avant une classification.
#temperature <- scale(temperature)
```

3. Réaliser la Classification Ascendante Hiérarchique et choisir le nombre de classes.

```
res.hclust <- hclust(dist(temperature),method="ward")
plot(res.hclust,xlab="nb de classes",ylab="Hauteur")
cl.hclust <- identify(res.hclust)
nb <- length(cl.hclust)
class.hclust <- rep(0,dim(temperature)[1])
for (i in 1:nb) {class.hclust[cl.hclust[[i]]] <- i}
```

4. Si le nombre d'individus n avait été trop grand, le calcul de la matrice de distance $n \times n$ peut s'avérer impossible à stocker. Dans ce cas là, nous réalisons la méthode des `kmeans` en choisissant un nombre de classes nettement plus petit que le nombre d'individus. Une règle peut être de prendre 10% de la valeur nombre d'individus. Ensuite, on réalise une Classification Ascendante Hiérarchique sur les barycentres des classes et on choisit le nombre de classes à retenir.

```
resAUX.class <- kmeans(temperature,centers=10)
resAUX.hclust <- hclust(dist(resAUX.class$centers),method="ward")
plot(resAUX.hclust,xlab="nb de classes",ylab="Hauteur")
```

5. Relancer le programme de la question précédente plusieurs fois. Que remarquez-vous ?

6. La CAH permet de faire de la classification sans connaissance a priori sur le nombre de classes. Une fois le nombre de classes choisi, on peut utiliser cette valeur pour lancer la méthode des `kmeans`. A faire.

```
# Choisir le nombre de classes : 3,4,5 ?
res.class <- kmeans(temperature,centers=3)
class.kmeans <- res.class$cluster
```

7. Pour interpréter les classes obtenues sur les données "brutes", il est intéressant de représenter ces groupes sur le(s) plan(s) factoriel(s) d'une ACP. Réaliser l'ACP et analyser les résultats.

```
res.pca <- PCA(temperature0, scale.unit=TRUE, ind.sup=24:35,quanti.sup=13:16,
           quali.sup=17)
plot.PCA(res.pca, choix="ind", habillage=17)
res.pca$eig
barplot(res.pca$eig[,1])
boxplot(res.pca$ind$coord)
```

8. Représenter les groupes obtenus par chacune des méthodes sur le premier plan factoriel de l'ACP.

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

col.hclust <- as.character(factor(class.hclust,levels=1:5,
              labels=c("black","red","green","blue","purple")))
plot.PCA(res.pca,col.ind=col.hclust)
# A comparer avec :
#res.hcpc <- HCPC(temperature)

col.kmeans <- as.character(factor(class.kmeans,levels=1:5,
              labels=c("black","red","green","blue","purple")))
plot.PCA(res.pca,col.ind=col.kmeans)
```

1.2 Classification sur composantes principales

Une deuxième approche, peut-être pas nécessaire ici mais utile lorsque le nombre de variables est très important, consiste à enchaîner une technique de classification sur les variables principales d'une ACP.

9. Réaliser l'ACP et la Classification Ascendante Hiérarchique sur les composantes.

```
# Sans perte d'information #
res.pca <- PCA(temperature0, scale.unit=TRUE, ncp=Inf, ind.sup=24:35,
           quanti.sup=13:16, quali.sup=17)
# En éliminant le bruit #
# res.pca <- PCA(temperature0, scale.unit=TRUE, ncp=12, ind.sup=24:35,
             quanti.sup=13:16, quali.sup=17)

res.hcpc <- HCPC(res.pca)
```

10. Afficher et commenter : le nombre de classes optimal, l'inertie intra, le gain d'inertie inter et le rapport de deux inerties intra successives.

```
res.hcpc$call$t$nb.clust
res.hcpc$call$t$within
res.hcpc$call$t$inert.gain
res.hcpc$call$t$quot
```

11. Dessiner l'arbre complet en 3 dimensions.

```
res.hcpc <- HCPC(res.pca,t.levels="all")
```

12. Décrire les classes.

```
res.hcpc$desc.var
```

13. Définir les individus les plus proches du centre de leur classe et les individus les plus éloignés du centre de la classe la plus proche, hormis la sienne.

```
res.hcpc$desc.axe
res.hcpc$spec
```

2 Données Bancaires

2.1 Classification sur scores d'une AFCM

On réalise les procédures de classification sur les valeurs des composantes principales de l'AFCM.

14. Chargement et pré-traitement des données.

```
credit0 <- read.csv(file.choose(),header=TRUE,sep=";",row.names=1)
credit=credit0[-68,c(1,2,3,10,11,4,5,6,7,8,9)]
for (i in 1:ncol(credit)) credit[,i] <- factor(as.character(credit[,i]))
```

15. Réaliser l'AFCM sur ce jeu de données et la Classification Ascendante Hiérarchique.

```
res.MCA=MCA(credit[-67,],quali.sup=6:11,graph=FALSE)
HCPC(res.MCA)
```

16. Analyser les classes obtenues.