TP-4 Classification automatique des données

Exercice 1 : Le jeu de données `decathlon` concerne les résultats aux épreuves du décathlon lors de deux compétitions d'adathlétisme. Il s'agit de 41 athlètes décrits par 13 variables. Plus précisément, pour chaque athlète, on a recueilli ses performances à chacune des 10 épreuves, son classement final, son nombre de points final et la compétition à laquelle il a participé : les Jeux Olympiques d'Athènes (août 2004) ou le Décastar (septembre 2004). On a effectué une classification hiérarchique sur les variables relatives aux 10 épreuves de la base `decathlon`.

1. Classification hiérarchique avec les fonctions `agnes` et `hclust`.

2. Détermination du bon nombre de classe à partir du critère du plus haut saut.

3. Description des classes avec la fonction `catdes`

4. Représentation des classes sur un plan factoriel.

5. Représentation des classes sur un plan factoriel avec la fonction `HCPC`.

```
library(cluster)
library(FactoMineR)
data("decathlon")
X=decathlon[,1:10]


#1. Classification hiérarchique des données

# CAH avec agnes
classif<-agnes(X, method="ward")
plot(classif,xlab="individu",main="")
title("Dendrogramme")


# CAH avec hclust
# La fonction hclust de cluster prend en argument aussi bien une matrice
# ind*var qu'une matrice de dissimilarité

# Remarque  :
# Pour une classif sur var. quali, utiliser  dist.binary (1) ou dist.dudi (2) de ade4.
# (1) https://www.rdocumentation.org/packages/ade4/versions/1.7-6/topics/dist.binary
# (2) http://forums.cirad.fr/logiciel-r/viewtopic.php?t=5128

d=dist(scale(X),method = "euclidian")
hc1=hclust(d,method = "ward.D") # or complete or average
plot(hc1)
plot(hc1, hang = -1)
```

```
# Output de hclust
hc1$dist.method
hc1$method
hc1$height
hc1$order## Ordre des indivuds sur le dendogramme
hc1$merge## Déroulement du processus d'aggrégation.


#2. Détermination de  la meilleure partition

classif2<-as.hclust(classif)
plot(rev(classif2$height), type="h", ylab="hauteurs")
rect.hclust(hc1,k=2)
classes<-cutree(classif2,k=2)
classes

#Visualisation de l'effet coude.

plot(1:40,hc1$height[40:1],type="b")
inertie = sort(hc1$height,decreasing=TRUE)
plot(inertie[1:20],type="s",xlab="Nombre de classes", ylab="Inertie")


#3. Description des classes
# Rajout de la classe d'affectation de chaque individu en tant que variable quali.
decathlon.comp<-cbind.data.frame(decathlon, as.factor(classes))
colnames(decathlon.comp)[14]<-"Classe"
head(decathlon.comp)

# Description des classes avec la fonction catdes
res.cat=catdes(decathlon.comp, num.var=14)


#4. Représentation des classes sur un plan factoriel

res.pca<-PCA(X.comp,quali.sup=11)
plot(res.pca,choix="ind",habillage=11)


#5. Représentation des classes sur un plan factoriel : la fonction HCPC

res.pca<-PCA(decathlon,quanti.sup=11:12, ncp=Inf, graph=F, quali.sup=13)
#res.hcpc<-HCPC(res.pca,consol=FALSE)
res.hcpc<-HCPC(res.pca)
```

Exercice 2 : On considère le jeu de données "`Iris de Fisher`" sous R. On voudrait effectuer une classification des 150 iris en utilisant les 4 variables quantitatives les décrivant et en déterminant la meilleure partition possible de ce jeu de données.

1. Visualisation des 3 classes d'iris sur le premier plan factoriel.

Objectif 1 : Déterminer la combinaisin (méthode + input) qui permet d'avoir la partition la plus proche de la partition naturelle (i.e. celle données par la cinquième variable).

2. Classif hiérarchique (CAH) sur variables puis sur facteurs (les 2 premiers) et comparer les résultats.

3. Comparer la partition obtenue (en 3 classes) avec la partition naturelle des Iris : Tableau croisé + Indice de Rand et Rand corrigé (cf. package fossil). On pourra visualiser les classes obtenues.

4. Faire les questions 2 et 3 avec la méthode des k-means.

Objectif 2 : Déterminer la meilleure partition (i.e. le meilleur nombre de classes)

5. Déterminer le meilleur nombre de classes avec NbClust et Clvalid pour CAH et k-means.

6. Détermination de la meilleure partition avec la fonction HCPC (classification mixte avec consolidation)

CAH et k-means sous Python :
http ://eric.univ-lyon2.fr/ ricco/cours/didacticiels/Python/cah_kmeans_avec_python.pdf

Autres méthodes : SOM, Algorithme EM.

7. SOM : classification puis description des 3 classes obtenues avec SOM suivie d'une classification hiérarchique. On commentera les codebook.

SOM sous R :
http ://eric.univ-lyon2.fr/ ricco/tanagra/fichiers/fr_Tanagra_Kohonen_SOM_R.pdf

8. EM : classification et détermination (intégrée avec le critère BIC) du bon nombre de classes

EM sous R :
(1) https ://cran.r-project.org/web/packages/mclust/vignettes/mclust.html
(2) https ://www.ncbi.nlm.nih.gov/pmc/articles/PMC5096736/

EM sous Python :
https ://jakevdp.github.io/PythonDataScienceHandbook/05.12-gaussian-mixtures.html