TP Analyse de données - Apprentissage non supervisé

Introduction

Nous allons utiliser le logiciel `R` (documentation ; possibilité d'utiliser Rstudio). Dans un dossier *AnalyseDeDonnees* créer deux sous dossiers :

- Code - dans lequel vous placerez vos fichiers de code
- Data - dans lequel vous placerez les fichiers de données (à télécharger via Moodle)

Les 3 TP *Analyse de données* présentent différentes méthodes d'analyse de données : le but n'est pas de finir les TP le plus vite possible mais d'analyser les résultats ! Un rapport de 1 page **maximum** vous est demandé pour chacun des 3 TP : ne choisir que les résultats les plus intéressants et les **commenter**.

Ne pas hésiter à utiliser l'aide de `R` grâce à la commande :

```
help(...)
```

Analyse en composantes principales

0. Télécharger le cours sur l'Analyse en Composantes Principales.

1. Créer un fichier *ACP.R* dans le dossier *Code*. Charger les packages d'intérêt en ajoutant dans le fichier :

```
# Adresse du dossier où vous travaillez
setwd("/Users/.../AnalyseDonnees/TP/TP/Code")
# Packages utilisés dans la suite
library(GGally)
library("FactoMineR")
```

2. Charger les données et les manipuler :

```
# Chargement des données
load("../Data/eaux.rda")
# Création de trois objets
X <- data
n <- nrow(X) # nombre d'observations
p <- ncol(X) # nombre de variables
# Manipulation des données
X[, 1:3] # les 3 premières colonnes
X[, c(1,5)] # la première et la 5ème colonne
X[, -c(1,5)] # toutes les colonnes sauf la première et la 5ème
X[1:3, ] # les 3 premières lignes
class(X) # Un data.frame est une liste dont les éléments sont les colonnes
X$saveur.amère # On récupère les élements d'une liste avec le $
X[ ,1]
```

3. Calculer la moyenne et l'écart type de chacune des variables :

```
# Moyenne de la variable saveur amère
sum(X$saveur.amère)/n
mean(X$saveur.amère)
# Ecart-type de la variable saveur amère
x <- X$saveur.amère
sqrt(1/(n-1)*sum((x-mean(x))^2))
sd(x)
# Moyenne et écart-type de toutes les variables
moy <- apply(X, 2, mean) # 1=lignes, 2=colonnes
sigma <- apply(X,2, sd) #standard deviation
```

4. Centrer-réduire (standardiser) les variables :

```
# Création des données centrées ...
Y <- sweep(X, 2, moy, "-")
apply(Y, 2, mean) # les colonnes sont bien centrées
# ... et réduites
Z <- sweep(Y, 2, sigma, "/")
apply(Z, 2, sd) # les colonnes sont bien de variance 1
# ou de manière équivalente
Z <- scale(X)
# ou avec l'écart-type non corrigé (comme en ACP)
Z <- scale(X)*sqrt(n/(n-1))
```

5. Description bivariée des 5 premières variables :

```
# Avec la fonction ggpairs du package GGally
ggpairs(X[,1:5])
# Cacluler et visualiser la matrice de corrélation
z1 <- Z[, 1] # variable saveur amère standardisée
z2 <- Z[, 2]# variable saveur sucrée standardisée
sum(z1*z2)/n # corrélation entre les deux variables
cor(X$saveur.amère, X$saveur.sucrée)
cor(X[,1:5]) # matrice des corrélations entre les 5 première variables
ggcorr(X[,1:5])
```

6.Calculer la matrice des distances entre les individus et les corrélations entre les variables :

```
# Matrice des distances entre les individus
dist(X) # données brutes
dist(Y) # données centrées
dist(Z) # données centrées-réduites
# Corrélation entre les variables
cor(X)
# ou encore
t(Z) %*% Z/n # %*% est le produit matriciel
```

7. Faire l'Analyse en Composantes Principales non normalisée (sur matrice de covariances) :

```
# Fonction PCA du package FactoMineR
# (scale.unit=FALSE)
res <- PCA(X, graph = FALSE, scale.unit = FALSE)
# Figure individus
plot(res,choix = "ind", cex = 1.5, title = "")
# Figure variables
plot(res, choix = "var", cex = 1.5, title = "")
# Equivalent à la décomposition en valeurs propres de la matrice des covariances
Y <- as.matrix(Y)
C <- (t(Y) %*% Y)/n # matrice des covariances
eigen(C)$values
res$eig[, 1]
```

8. Faire l'Analyse en Composantes Principales normalisée (sur matrice des corrélations) :

```
# Analyse en composantes principales normalisée (sur matrice des corrélations)
# (par défaut: scale.unit=TRUE)
res <- PCA(X, graph=FALSE)
# Figure individus
plot(res, choix = "ind", cex = 1.5, title = "") # plan 1-2
plot(res, choix = "ind", axes=c(2,3), cex = 1.5, title = "") # plan 2-3
# Figure variables
plot(res, choix = "var", cex = 1.5, title = "") # plan 1-2
plot(res, choix = "var", axes=c(2,3), cex = 1.5, title = "") # plan 2-3
# Equivalent à la décomposition en valeurs propres de la matrice des corrélations
R <- (t(Z) %*% Z)/n # matrice des corrélations
eigen(R)$values
res$eig[, 1]
# Récuperer les 2 premières compostantes principales
F <- res$ind$coord[, 1:2]
plot(F, pch = 16)
text(F, rownames(X), pos = 3) # on retrouve la figure des individus
# Récuperer les loadings (corrélations aux deux premières CP)
A <- res$var$coord[, 1:2]
plot(A, pch=16)
text(A, colnames(X), pos = 3) # on retrouve la figure des variables
A[1, , drop=FALSE] # corrélations entre saveur amère et les 2 premières CP
cor(F, X$saveur.amère)
# Interprétation du premier plan des individus en fonction des variables ?
```

9. Interprétation des résultats : combien de composantes peut-on retenir ? Utiliser les règles de Kaiser et du coude.

```
# Inertie (variance) des composantes principales
apply(F, 2, var)*(n-1)/n # variances des 2 premières CP
res$eig[, 1]
sum(res$eig[, 1])
res$eig
```

10. Interprétation des résultats : qualité de la projection des individus

```
# Qualité de la projection des individus sur les axes
res$ind$cos2
# Qualité de la projection des individus sur le premier plan
apply(res$ind$cos2, 1, sum)
# Interprétation du premier plan factoriel des individus ?
```

11. Interprétation des résultats : qualité de la projection des variables

```
# Qualité de la projection des variables sur les axes
res$var$cos2
# Qualité de la projection des variables sur le premier plan
apply(res$var$cos2, 1, sum) # ou regarder la longeur des flèches !
# Interprétation du premier plan factoriel des variables ?
```

En plus - Quand vous faites de l'ACP il y a deux erreurs à éviter :

- Attention aux données très asymétriques : par exemple beaucoup de très petites valeurs et quelques très grandes (dans ce cas là une transformation des données peut être utile ...).
- Attention à l'effet *taille* (toutes les variables ont des contributions positives sur un axe) quand les données sont corrélées entre elles.

Clustering

0. Télécharger le cours sur le clustering.

1. Créer un fichier *clustering.R* dans le dossier *Code*. Charger les packages d'intérêt en ajoutant dans le fichier :

```
# Adresse du dossier où vous travaillez
setwd("/Users/.../AnalyseDonnees/TP/TP/Code")
# Packages utilisés dans la suite
library("FactoMineR")
```

Les kmeans

2. Charger les données et les afficher :

```
# Données sur les fromages
X <- read.table("../Data/fromage.txt", sep="", header=TRUE, row.names=1)
n <- nrow(X)
p <- ncol(X)
```

3. Calculer la moyenne et l'écart type de chacune des variables :

```
# Calcul de la moyenne et de l'écart type des variables
moy <- apply(X, 2, mean)
sigma <- apply(X, 2, sd)*sqrt((n-1)/n) #écart-type non corrigé
```

4. Standardiser les données :

```
# Création des données centrées ...
Y <- sweep(X, 2, moy, "-")
# ... et réduites
Z <- sweep(Y, 2, sigma, "/")
```

5. Fixer le nombre de clusters souhaité :

```
# Nombre de clusters souhaité
K <- 5
```

6. Appliquer l'algorithme des kmeans sur les données standardisées :

```
# kmeans en 5 classes sur données standardisées
km <- kmeans(Z, centers=K, nstart=50)
km$cluster # la partition
table(km$cluster) # effectifs des classes
km$withinss # inertie de chaque classe (within sum of squares)
km$betweenss # inertie inter-classe (between sum of squares)
km$totss # inertie totale (total sum of squares)
sum(km$withinss) + km$betweenss # inertie total= inta + inter
km$betweenss/km$totss # pourcentage d'inertie expliquée par la partition
# Vérifier que ce pourcentage augmente quand K augmente
```

7. Cette partition en 5 classes est une nouvelle variable qualitative :

```
# La partition est une nouvelle variable qualitative
part <- as.factor(km$cluster) # object de classe factor
levels(part) <- paste("cluster", 1:K, sep="") # modalités de la variable
# Un data.frame mélange les colonnes numériques et qualitatives !
Xplus <- data.frame(X, part)
View(Xplus)
```

8. Interpréter cette partition en visualisant les classes sur une ACP normée et avec la fonction `catdes` du package FactoMineR :

```
# ACP normée en mettant la variable "part" en illustrative
res <- PCA(Xplus, quali.sup="part", graph = FALSE)
res$eig
plot(res, choix = "ind", habillage = "part", invisible = "quali")
plot.PCA(res, choix = "var")
# Interprétation des classes avec la fonction catdes
des <- catdes(Xplus, num.var = 10)
des$quanti$cluster1
des$quanti$cluster2
# etc...
```

La CAH de Ward

9. Appliquer la classification ascendante hiérarchique de Ward sur les données standardisées :

```
#Classification hiérarchique de Ward sur données centrées-réduites
D <- dist(Z)
tree <- hclust(D^2/(2*n), method = "ward.D")
plot(tree, hang=-1, main="Dendrogramme de Ward", xlab="", sub = "")
tree$height
sum(tree$height) # Inertie totale
```

10. Partition en 5 classes de Ward et comparaison avec la partition en 5 classes des kmeans :

```
# Partition en 5 classes avec la CAH de Ward
K <- 5
plot(tree, hang=-1, main="", sub="", xlab="")
rect.hclust(tree, k=K)
part_ward <- cutree(tree, k=K) # partition en 5 classes
# Comparaison avec la partition en 5 classes des kmeans
table(part_ward) # effectif des classes de la partition de Ward
table(km$cluster) # effectifs des classes de la partition des kmeans
table(part_ward, km$cluster) # tableau de contingence entre les deux partitions
W <- sum(tree$height[1:(n-K)]) # inertie intra-classe de la partition de Ward
(1-W/9)*100 # Pourcentage d'inertie expliquée à comparer avec les kmeans
```

11. Combiner Ward et kmeans :

```
# 1. Calculer les centres des 5 classes de Ward
sp <- split(Z, part_ward)
centres_ward <- t(sapply(sp, colMeans))
# 2. Appliquer les kmeans en partant des centres des classes de Ward
km <- kmeans(Z, centers = centres_ward) # pas besoin de nstart !
table(part_ward, km$cluster) # deux fromages ont changés de classe
km$betweenss/km$totss # Le pourcentage d'inertie expliquée est un peu meilleur
```

Clustering de données qualitatives

12. Charger les données sur les races de chiens. Les recoder via l'ACM.

```
load("../Data/chiens.rda")
res <- MCA(chiens, graph = FALSE) # Multiple Correspondance Analysis
res$eig # 10 valeurs propres non nulles
res <- MCA(chiens, ncp=10, graph = FALSE) # 10 composantes principales
F <- res$ind$coord # données recodées
```

13. Appliquer la CAH de Ward aux données recodées.

```
tree <- hclust(dist(F)^2/(2*nrow(F)), method="ward.D")
plot(tree, hang=-1)
```

14. Couper le dendrogramme en conséquence et interpréter la partition avec la fonction `catedes`.

```
part <- cutree(tree, k=4) # partition en 4 classes
part <- as.factor(part)
levels(part) <- paste("cluster", 1:4, sep="")
des <- catdes(data.frame(part, chiens), num.var=1)
des$category
```