Scénario: Classification non supervisée et représentations factorielles

Résumé

Classification non supervisée (CAH, kmeans) et représentation des classes dans un plan factoriel avec des données sous différents formes : tableau de distance et MDS, variables quantitatives et ACP, variables qualitatives et AFCM.

1 Introduction

Le nombre de méthodes de classification non supervisée (clustering) et le nombre d'options dans chacune d'elles conduisent à une combinatoire de possibilités assez lourde. Le but est de mettre en évidence, de façon heuristique et dans des cas simples, quels sont les paramètres qui influencent, de visu, les classes obtenues. Un rôle important est donc attribué à la représentation des classes dans un plan factoriel. Le choix de la méthode : analyse en composantes principales, analyse factorielle des correspondances multiple ou encore positionnement multidimensionnel, dépend du type (quantitatives, qualitatives, distances) des données analysées.

La plupart des analyses ci-dessous pourraient être reproduites avec SAS mais de façon nettement moins souple et pas toutes (PAM n'est pas implémenté). En revanche, SAS l'emporte nettement dès qu'il s'agit d'analyser des très gros ensembles de données.

Le scénario sur l'analyse élémentaire de données transcriptomiques fournit un exemple plus complet, donc plus complexe, d'utilisations conjointes d'algorithmes de classification et de représentations par méthodes factorielles et MDS

2 Positionnement multidimensionnelle ou MDS

le *MultiDimensional Scaling, MDS* ou encore ACP d'un tableau de distances s'applique à des données archivées sous la forme d'une matrice $(n \times n)$ de distances ou dissimilarités. Cet algorithme est appliqué sur un exemple simple comparant des distances entre villes.

Les données se présentent donc sous la forme du triangle inférieur d'une matrice symétrique, par construction, et contenant les distances kilométriques routières, donc non-euclidienne, de 47 villes française ou proches prises 2 à 2 (Source : carte IGN).

2.1 MDS avec SAS

Charger le fichier `mdsville2.dat` puis lire la matrice triangulaire inférieure en précisant qu'il s'agit d'une matrice de distance :

```
data sasuser.mdsville (type=distance) ;
infile "mdsville2.dat" missover lrecl=200;
input ville $ 4. (d1-d47)(4.); run;
```

Calcul de l'ACP du tableau des distances et représentation graphique.

```
proc mds data=sasuser.mdsville dim=2 out=resul
   level=abs;
var d1-d47;
object ville; run;
%gafcx(ident=ville);
```

Commenter la représentation par rapport à la carte "géographique".

2.2 MDS avec R

Lecture des données contenant les distances des villes

La lecture d'une matrice triangulaire inférieure en tant que matrice de distance pose quelques difficultés dans R dans la gestion du type des objets.

Lire le fichier mdsville.dat.

```
mdsville=read.table("mdsville.dat",fill=TRUE)
```

```
# extraction des noms des villes
villes=as.character(mdsville[2:48,1])
# extraction des valeurs des distances
m=mdsville[2:48,2:48]
# transformation en une matrice alphanumérique
m=as.matrix(m)
# retour au numérique avec données manquantes
m=as.numeric(m)
# reformattage en une matrice
m=matrix(m,47,47)
# adjonction des noms des villes en ligne et colonne
dimnames(m)[[1]]=villes
dimnames(m)[[2]]=villes
# transformation en un objet de type distance
d=as.dist(m,diag=TRUE)
d
```

MDS des villes avec R

En revanche, le MDS est immédiat. On remarque que le choix de la dimension, ici $k = 2$ est à traiter comme en ACP. Une visite du manuel permet de voir comment calculer toutes les valeurs singulières afin d'aider ce choix par leur représentation.

```
mds = cmdscale(d, k=2)
plot(mds, type="n", xlab="cp1", ylab="cp2")
text(mds,villes)
```

3 Classification à partir d'un tableau de distances

Les mêmes données sont reprises en vue de regrouper les villes en classes homogènes au regard de leurs distances respectives.

3.1 CAH et représentation par MDS

Recherche du dendrogramme, du nombre de classes et "coloration" des villes en fonction de l'appartenance à une classe.

```
chv = hclust(d, method="ward")
plot(chv,main=NULL,sub="",xlab="")
# corrélation cophenetic
cor(cophenetic(chv),d)
# choix par décroissance du saut
plot(chv$height[46:30],xlab="nb de classes"
   ,ylab="Hauteur")
color=cutree(chv,k=5)
# silhouettes des classes
library(cluster)
plot(silhouette(color,d))
```

Il est possible de calculer puis représenter la silhouette moyenne pour plusieurs valeurs du nombre de classes et tenter de minimiser cette valeur ; cette stratégie conduisant à un choix "trivial" $(k = 3)$ est laissée de côté.

Les données provenant d'une matrice de distances, le MDS s'impose pour une représentation fatorielle des villes.

```
plot(mds, type="n", xlab="cp1", ylab="cp2")
# représentaiton avec des couleurs
text(mds,villes,col=color)
```

Influence déterminante du choix de la distance entre groupe :

```
chv <- hclust(d, method="single")
cor(cophenetic(chv),d)
plot(chv,main=NULL,sub="",xlab="")
plot(chv$height[46:30],xlab="nb de classes",
    ylab="Hauteur")
color=cutree(chv,k=5)
plot(mds, type="n", xlab="cp1", ylab="cp2")
text(mds,villes,col=color) # représentation
                           #avec des couleurs
```

Tester les autres options de distances entre groupes. Attention, la plus grande corrélation *cohenetic* ne correspond sans doute pas nécessairement au meilleur choix de distance entre groupes.

3.2 PAM et représentation par MDS

L'algorithme de réallocation k-means n'est pas adapté à une matrice de distances ou de dissimilarités ; en revanche, PAM est opérationnel si le nombre d'observations n'est pas trop important ; sinon utiliser l'adaptation `clara`. Ces fonctions sont disponibles dans la librairie `cluster`.

```
library(cluster)
# faire varier le nombre de classes
plot(silhouette(pam(d,6)))
pamv=pam(d,6)
color=pamv$clustering
plot(mds, type="n", xlab="cp1", ylab="cp2")
# représentation avec des couleurs
text(mds,villes,col=color)
# le même avec des ellipses
clusplot(d,pamv$clustering,diss=TRUE,labels=2,
      color=TRUE,col.txt=pamv$clustering,main="")
```

Remarque : le critère de `clusGap` pour la sélection du nombre de classe ne s'applique pas à une matrice de distances mais il pourrait s'appliquer aux coordonnées principales issues du MDS.

4 Classification de données quantitatives

4.1 Données OCDE

Les données (ocdeR.dat) sont celles étudiées comme exemple d'ACP cubique. Comme elles sont connues par une représentation euclidienne, la distance considérée par défaut est euclidienne mais d'autres choix sont possibles (manhattan ou valeurs absolues...).

Télécharger le fichier ocdeR.dat.

4.2 CAH et représentation par ACP

```
#lecture des données avec le nom des variables
#en première ligne
ocde=read.table("ocdeR.dat")
# calcul de la distance une fois avec la
# commande scale, une fois sans
ds=dist(scale(ocde))
# classification hiérarchique
hc.ds <- hclust(ds,method="ward")
plot(hc.ds)  # dendogramme
# choix du nombre de classes
plot(hc.ds$height[67 :58],type="b")
color= cutree(hc.ds,k=4) # couleurs des classes
# silhouette
library(cluster)
plot(silhouette(color,d))
# corrélation cophenetic
cor(cophenetic(hc.ds),ds)
# acp pour représentations
library(FactoMineR)
acp=PCA(ocde,ncp=13,graph=F)
# graphe de l'acp
plot(acp,choix="ind", habillage="ind",
          col.hab=rep(1:17, c(rep(4,17))))
# graphe de l'acp avec les couleurs
#          des classes de la cah
plot(acp,choix="ind", habillage="ind",col.hab=color)
```

Refaire tourner l'algorithme en retirant la réduction des variables (fonction scale), même chose en remplaçant `method="ward"` par `method="single"`.

4.3 Algorithme de ré-allocation et représentation par ACP

Les données étant quantitatives, k-means est opérationnel.

```
# version avec réduction
#  choix de 4 classes suggérées par la cah
```

```
kocde=kmeans(scale(ocde),4)
color=kocde$cluster
plot(acp,choix="ind", habillage="ind",col.hab=color)
# version sans réduction
kocde=kmeans(ocde,4)
color=kocde$cluster
plot(acp,choix="ind", habillage="ind",col.hab=color)
# utilisation de clusGap
library(cluster)
clusGap(ocde, FUN = kmeans, K.max = 8, B = 60)
```

Plusieurs critères sont concurrents pour sélectionner un k optimal une fois les calculs de `clusGap` réalisés par *bootsrap*. le choix reste complexe.

4.4 Classification sur composantes principales

La librairie `FactoMineR` propose cette possibilité avec la production de graphiques élaborées mais sans doute pas très utiles. Un autre paramètre est susceptible d'avoir un impact, le nombre `ncp` de composantes retenus.

```
acp=PCA(ocde,ncp=3,graph=F)
res.hcpc=HCPC(acp)
```

5 Classification de données qualitatives

5.1 Données décrivant les races de chien

Plusieurs stratégies sont proposées pour classées des données qualitatives ou mélange de variables quantitatives et qualitatives. La plus simple consiste à recoder les variables ou calculer des "scores" à l'aide des composantes d'une AFCM. Les données (race des chiens) sont celles illustrant l'utilisation de l'AFCM. Encore une fois R est nettement plus souple que SAS pour enchaîner de telles analyses.

5.2 Importation des données

Comme pour les données précédentes, il faudrait adapater le fichier pour une importation facile dans R en ajoutant une première ligne contenant le nom des variables et en re-transformant les modalités. Il est plus simple, et utile à savoir, de transférer directement les données de SAS vers R. Remarque : ces données ont déjà été transférées de SAS à R au cours du TP sur l'AFCM. Il peut être réutilisé en l'état.

```
/* Exportation d'un fichier SAS en format .csv */
proc export data=sasuser.chiens
     outfile= "chiens.csv"
     DBMS=CSV REPLACE;
run;
```

Rechercher où SAS a "rangé" le ficher `chiens.csv`, sans doute dans le répertoire duquel l'exécution a été lancée. Éventuellement le déplacer dans le répertoire courant de R.

```
# retour à R
chiens=read.csv("chiens.csv")
 # Vérifier que tout s'est bien passé
#Attention, la première colonne identifie la race du
#chien et donc chaque ``individu''
dimnames(chiens)[[1]]=as.character(chiens[,1])
chiens=chiens[,-1]
summary(chiens)
```

5.3 CAH sur composantes de l'AFCM

Nous reprenons l'AFCM sur tableau disjonctif complet fournissant des scores ou variables quantitatives sur les individus, ici les races de chiens, il est alors facile de les classer. L'enchaînement par une CAH est la combinaison la plus simple à réaliser à l'aide de la librairie FactoMineR. Attention néanmoins de bien contrôler les options qui toutes sont prises par défaut.

```
library(FactoMineR)
#afcm avec la fonction en supplémentaire
afcm=MCA(chiens,quali.sup=7,graph=F)
plot(afcm, choix="ind",habillage="quali")
#sans les individus
plot(afcm,habillage="quali",invisible="ind")
#cah
```

```
res.hcpc=HCPC(afcm)
# Il est proposé de croiser la variable classe
# avec chacune des variables
res.hcpc$desc.var$test.chi2
#et de chercher les modalités les plus
# "présentes" dans chaque classe
res.hcpc$desc.var$category
```

D'autres outils (arbre de discrimination) s'avéreront plus explicites pour interpréter les classes.

Questions : combien de dimensions sont retenues en AFCM ? Quelle distance est prise en compte entre les individus, les groupes ? Ainsi, une option positionnée (`consol=TRUE`) fait enchaîner automatiquement un algorithme k-means.

```
afcm=MCA(chiens,quali.sup=7, ncp=10, graph=F)
res.hcpc=HCPC(afcm,method="single")
# même chose avec Ward et l'option
# pour conserver l'ordre des individus
res.hcpc=HCPC(afcm,method="ward",order=FALSE)
# le croisement
table(res.hcpc$data.clust$clust,
   res.hcpc$data.clust$fonction)
# permet d'interpréter facilement les classes
color=as.integer(res.hcpc$data.clust$clust)
plot(afcm, choix="ind",habillage="quali",
    col.ind=color)
```

En fait les classes se distinguent assez facilement et restent stables sur ces données sauf en cas d'utilisation du saut minimum.

Bien entendu, la fonction `HCPC` de `FactoMineR` pourrait être remplacée par tout autre algorithme de classification avec ses propres options (`hclust, kmeans, pam...`).