TP4 : Apprentissage non-supervisé

1 Mise en place

Pour illustrer les deux procédures d'apprentissage non-supervisé vues en cours, nous proposons d'étudier des données routières de l'IGN. Ces données contiennent les distances routières entre différentes villes de France et des pays frontaliers (47 lieux).

Afin de faciliter la récupération des données et de disposer de certaines fonctions utiles dans la suite de cette séance, nous commençons par charger le script *tp4.R*,

```
source("http://www.math.univ-toulouse.fr/~xgendre/ens/m2se/tp4.R")
```

Puisque nous manipulerons toujours le même jeu de données dans cette séance, nous le plaçons dès maintenant dans `X`,

```
X <- DataVilles()
```

2 Centres mobiles

La méthode des centres mobiles permet de classifier les individus en k classes avec une valeur de k fixée par le statisticien. Le langage R propose une fonction pour réaliser une classification par la méthode des centres mobiles. Son nom vient du nom anglais de la procédure "k-means" et il s'agit de `kmeans`. L'aide `help(kmeans)` est très sommaire et nous y voyons que pour obtenir la classification avec $k = 5$ classes, il suffit de faire

```
cmob <- kmeans(X, 5)
```

L'objet `cmob` obtenu contient plusieurs informations. Parmi elles, nous avons :

- `cmob$cluster` vecteur d'entiers de 1 à k indiquant le numéro de la classe de chaque individu,
- `cmob$centers` matrice des distances entre les individus et les centres de chaque classe,
- `cmob$size` vecteur des tailles des classes obtenues.

Il est possible d'afficher le résultat sur une carte avec la commande suivante,

```
AfficheVilles(cmob$cluster)
```

Exécuter à nouveau la commande qui vous a permis d'obtenir `cmob` et afficher le résultat sur la carte. Que remarquez-vous ? Expliquez pourquoi le résultat n'est pas le même à chaque fois.

Afin de stabiliser cette variabilité, il est possible d'indiquer à `kmeans` les centres des classes initiales. Nous verrons cela plus en détails à la section 4. Une autre façon consiste à répéter plusieurs fois la procédure et à prendre la classification dont le critère est minimal (voir le cours). Pour étudier cela, nous pouvons utiliser l'information `cmob$tot.withinss`. A quoi correspond cette valeur ? Répéter plusieurs fois le calcul de la classification en affichant `cmob$tot.withinss` à chaque fois. Expliquez avec vos propres mots pourquoi il est utile de répéter ce calcul.

Comme cela a été discuté en cours, il est également possible de travailler avec la variante robuste des centres mobiles, connue sous le nom de PAM (*Partitioning Around Medoids*). Le package R `cluster` met à disposition une fonction `pam` pour utiliser cette procédure,

```
library(cluster)
help(pam)
```

La fonction `pam` est relativement similaire à `kmeans` et nous obtenons la classification en 5 classes de la façon suivante,

```
cpam <- pam(X,5)
AfficheVilles(cpam$clustering)
```

Faites quelques essais et comparez les résultats obtenus avec ceux issus de la commande `kmeans`. Que contient la variable `cpam$id.med` ? Utilisez cette variable et la fonction `VillePos` pour repérer les médoïdes sur la représentation graphique.

3 Classification hiérarchique ascendante

Le principal inconvénient des méthodes de type "centres mobiles" est d'imposer la connaissance a priori du nombre de classes. Pour contourner ce problème, plusieurs procédures ont été proposées et, parmi elles, nous avons vu en cours la classification hiérarchique ascendante. Pour réaliser cette classification, R propose la fonction `hclust`,

```
help(hclust)
```

Nous n'utiliserons ici que les deux premiers arguments de cette fonction :

- `d` les données fournies sous la forme d'une matrice de dissimilarités (nous utiliserons `as.dist` pour convertir notre matrice `X`),
- `method` notre choix de mesure de dissimilarité entre des groupes de données (*single linkage*, *complete linkage*, ...).

Il est possible d'afficher le dendrogramme obtenu avec un simple appel à la fonction `plot` sur l'objet retourné par `hclust`. Ainsi, nous obtenons le résultat suivant pour la classification hiérarchique ascendante réalisée avec la méthode de Ward,

```
cah <- hclust(as.dist(X),method="ward.D2")
plot(cah)
```

L'objet `cah` contient plusieurs informations dont les hauteurs des sauts. Nous pouvons afficher simplement l'éboulis de ces hauteurs,

```
plot(rev(cah$height),type="b")
```

Avec l'aide sur la fonction `hclust`, réalisez la classification hiérarchique ascendante avec différentes méthodes d'agrégation et comparez les dendrogrammes et les éboulis obtenus.

Pour réaliser la classification proprement dite, il nous faut couper l'arbre à une certaine hauteur. La fonction `rect.hclust` peut être utilisée afin de visualiser cette coupe de l'arbre pour un nombre `k` de clusters donné ou à une hauteur `h` fixée.

```
plot(cah)
rect.hclust(cah, k=5, border="red")
rect.hclust(cah, h=1500, border="blue")
```

Pour récupérer les clusters, nous utilisons la fonction `cutree`,

```
help(cutree)
```

Cette fonction retourne la classification obtenue en coupant l'arbre soit de façon à avoir `k` classes, soit à une hauteur `h` donnée.

```
cluster1 <- cutree(cah, k=5) # 5 classes
cluster2 <- cutree(cah, h=1000) # hauteur 1000
```

Affichez les classifications obtenues dans `cluster1` et `cluster2`. Faites varier `k`, `h` ainsi que la méthode d'agrégation utilisée. Commentez les différents résultats obtenus.

4 Stabilisation

Dans le cadre du cours, nous avons vu que l'éboulis des hauteurs donné par la classification hiérarchique ascendante permet de choisir un nombre de classes. Rappelez cette procédure et discutez du nombre de classes choisies pour différentes méthodes d'agrégation.

En particulier, pour obtenir un résultat stable qui tire avantage des deux méthodes présentées dans cette séance, il est possible d'utiliser le nombre de classes proposé par la classification hiérarchique ascendante dans une procédure de type "centres mobiles". Nous initialiserons cette dernière avec la classification issue de la première classification hiérarchique ascendante. Expliquez et exécutez les lignes suivantes,

```
cah <- hclust(as.dist(X),method="ward.D2")
cluster <- cutree(cah, h=2000)
AfficheVilles(cluster)

nc <- max(cluster)
init <- c()
for (i in 1:nc) init <- rbind(init, colMeans(X[cluster==i,]))
cmob <- kmeans(X,init)
AfficheVilles(cmob$cluster)
```

Faites d'autres essais de cette procédure et commentez les résultats obtenus. Pour cela, utilisez la méthode PAM, changez de procédure d'agrégation, faites varier `h`, ...