TP-Kmeans-DBSCAN

2023/2024

Contents

L'objectif de ce TP est d'illustrer les notions abordées dans le chapitre dédié aux algorithmes de clustering de type Kmeans et DBSCAN. Les librairies R nécessaires pour ce TP :

```
library(mclust)
library(cluster)
library(factoextra)
library(FactoMineR)
library(ppclust)
library(ggplot2)
library(corrplot)
library(gridExtra)
library(circlize)
library(viridis)
library(reshape2)
library(klaR)

library(dbscan)
library(seriation)
```

Clustering des données de vins (quantitatives)

Analyse descriptive des données

Présentation des données de vins

Dans ce TP, on va utiliser le jeu de données `wine` disponible sur la page moodle du cours.

Ce jeu de données comprend des mesures physico-chimiques réalisées sur un échantillon de $n = 600$ vins

(rouges et blancs) du Portugal. Ces mesures sont complétées par une évaluation sensorielle de la qualité par un ensemble d'experts. Chaque vin est décrit par les variables suivantes :

- Qualite : son évaluation sensorielle par les experts ("bad","medium","good"),
- Type : son type (1 pour un vin rouge, 0 pour un vin blanc),
- AcidVol : la teneur en acide volatile (en g/dm3 d'acide acétique),
- AcidCitr : la teneur en acide citrique (en g/dm3),
- SO2lbr : le dosage du dioxyde de soufre libre (en mg/dm3),
- SO2tot : le dosage du dioxyde de soufre total (en mg/dm3),
- Densite : la densité (en g/cm3),
- Alcool : le degré d'alcool (en % Vol.).

Question 1. Récupérez sur moodle le jeu de données `wine.txt` et chargez-le sous R.

```
wine <-read.table("wine.txt",header = T)
```

Vérifiez la nature des variables à l'aide de la fonction `str()`. Modifiez si nécessaire les variables qualitatives (à l'aide de `factor()`) et transformez les modalités "1" et "0" de la variable `Type` en "rouge" et "blanc" respectivement.

```
wine$Qualite <- factor(...,levels =... ,ordered=...)
wine$Type <- factor(...,labels =...)
```

Statistiques descriptives

Question 2. Faites quelques statistiques descriptives pour faire connaissance avec le jeu de données, avec des choix adaptés à la nature des variables (on pourra utiliser `barplot`,`pie`,`boxplot`...). En particulier, étudiez les corrélations entre les variables quantitatives et faites une ACP.

```
# A completer
```

Question 3. Pour la suite, on va utiliser les variables quantitatives pour faire de la classification non supervisée des vins. Les variables *Qualite* et *Type* seront utilisées comme des variables extérieures pour comparer / croiser avec les classifications obtenues pour l'interprétation.

Pensez-vous qu'il est nécessaire de transformer les variables quantitatives dans l'objectif de clustering avec un algorithme des Kmeans ? Si oui, mettez en place cette transformation.

```
# A completer
```

Classification avec l'algorithme des Kmeans

A K=3 fixé

Question 4. A l'aide de la fonction `kmeans()`, faites une classification non supervisée en 3 classes des vins. Regardez les options disponibles dans la fonction `kmeans()`.

```
help(kmeans)
reskmeans<-kmeans(....)
```

Question 5. Combien a-ton de vins par classe ? Visualisez la classification obtenue dans les premiers plans de l'ACP (vous pouvez utiliser la fonction `PCA()` et `fviz_pca_ind` de la librairie `FactoMineR` et la fonction `fviz_cluster` de la librairie `factoextra`).

```
# A COMPLETER

fviz_cluster(...,data=...)
fviz_pca_ind(...,col.ind=as.factor(reskmeans$cluster),geom = ...,axes=...)
...
```

Question 6. La classification obtenue précédemment a-t-elle un lien avec le type de vins ? Avec la qualité du vin ? Vous pouvez vous aider de la fonction `table()`, la fonction `adjustedRandIndex()` de la librairie `mclust`, ...

```
# A COMPLETER
```

Choix du nombre de classes

Question 7. On s'intéresse dans cette section au choix du nombre de classes K en étudiant l'évolution de l'inertie intraclasse. En faisant varier K entre 2 et 15, calculez l'inertie intraclasse associée à chaque classification obtenue. Tracez l'évolution de l'inertie intraclasse en fonction du nombre de classes. Qu'en concluez-vous ?

```
# A completer
Kmax<-15
reskmeanscl<-matrix(0,nrow=nrow(wine),ncol=Kmax-1)
Iintra<-NULL
for (k in 2:Kmax){
  resaux<-kmeans(...)
  reskmeanscl[,k-1]<-resaux$...
  Iintra<-c(Iintra,resaux$...)
}

df<-data.frame(K=2:15,Iintra=Iintra)
ggplot(df,aes(x=K,y=Iintra))+geom_line()+geom_point()+xlab("Nombre de classes")+ylab("Inertie intraclass
```

Question 8. Reprendre la question du choix du nombre de classes en utilisant le critère silhouette (vous pouvez vous aider de la fonction `silhouette()`et la fonction `daisy` pour calculer les distances entre individus). Pour la classification sélectionnée, représentez les poids $s(i)$ de chaque individu à l'aide de la fonction `fviz_silhouette()`.

```
# A COMPLETER
Silhou<-NULL
for (k in 2:Kmax){
   aux<-silhouette(...)
   Silhou<-c(Silhou,mean(aux[,3]))
}

df<-data.frame(K=2:Kmax,Silhouette=Silhou)
ggplot(df,aes(x=K,y=Silhouette))+
  geom_point()+
  geom_line()+theme(legend.position = "bottom")

aux<-silhouette(reskmeanscl[,3], daisy(wine[,-c(1:2)]))
fviz_silhouette(aux)+theme(plot.title = element_text(size =9))
```

Classification avec l'algorithme DBSCAN

DBSCAN à paramètres fixés

Question 9. Dans un premier temps, utilisez l'algorithme DBSCAN avec les paramètres `minPts=` 7 et `eps=` 1 à l'aide de la fonction `dbscan()` de la librairie `dbscan`. Quels sont les effectifs par classe ? Combien d'individus ne sont pas classés ?

```
# A COMPLETER
minPts<-7
```

```
eps<-1
res.db <- dbscan(...)
table(...)
```

```
fviz_cluster(res.db, wine[,-c(1:2)], geom="point",ellipse="FALSE")+
  theme(legend.position="none")+
  xlab("")+ylab("")+ggtitle("Avec DBSCAN")
```

Influence des paramètres de DBSCAN

Question 10. Pour étudier l'influence des paramètres `minPts` et `eps`, évaluez le nombre de classes obtenues et le nombre d'individus non classés pour différentes valeurs de ces paramètres.

```
minPts <- ...
eps <- ...
NBCluster <- matrix(0,nrow=length(minPts),ncol=length(eps))
NBNonCl <-matrix(0,nrow=length(minPts),ncol=length(eps))
for (i in 1:length(minPts)){
  for (j in 1:length(eps)){
    res<-dbscan(wine[,-c(1,2)], eps=eps[j], minPts=minPts[i])
    NBCluster[i,j] <- ...
    NBNonCl[i,j] <- ...
  }
}

df<-data.frame(eps=rep(eps,each=length(minPts)),
               minPts=as.factor(rep(minPts,length(eps))),
               NBCluster=c(NBCluster),
               NBNonCl=c(NBNonCl)*100/nrow(wine))

ggplot(df,aes(x=eps,y=NBCluster,col=minPts))+geom_point()+geom_line()
ggplot(df,aes(x=eps,y=NBNonCl,col=minPts))+geom_point()+geom_line()
```

Question 11. Pour une valeur de `minPts=7`, tracez le graphe de distance kNN afin de choisir le paramètre `eps`. Vous pouvez utiliser la fonction `kNNdistplot()`. Qu'en pensez-vous ?

```
# A COMPLETER
```

Comparaison avec les Kmeans

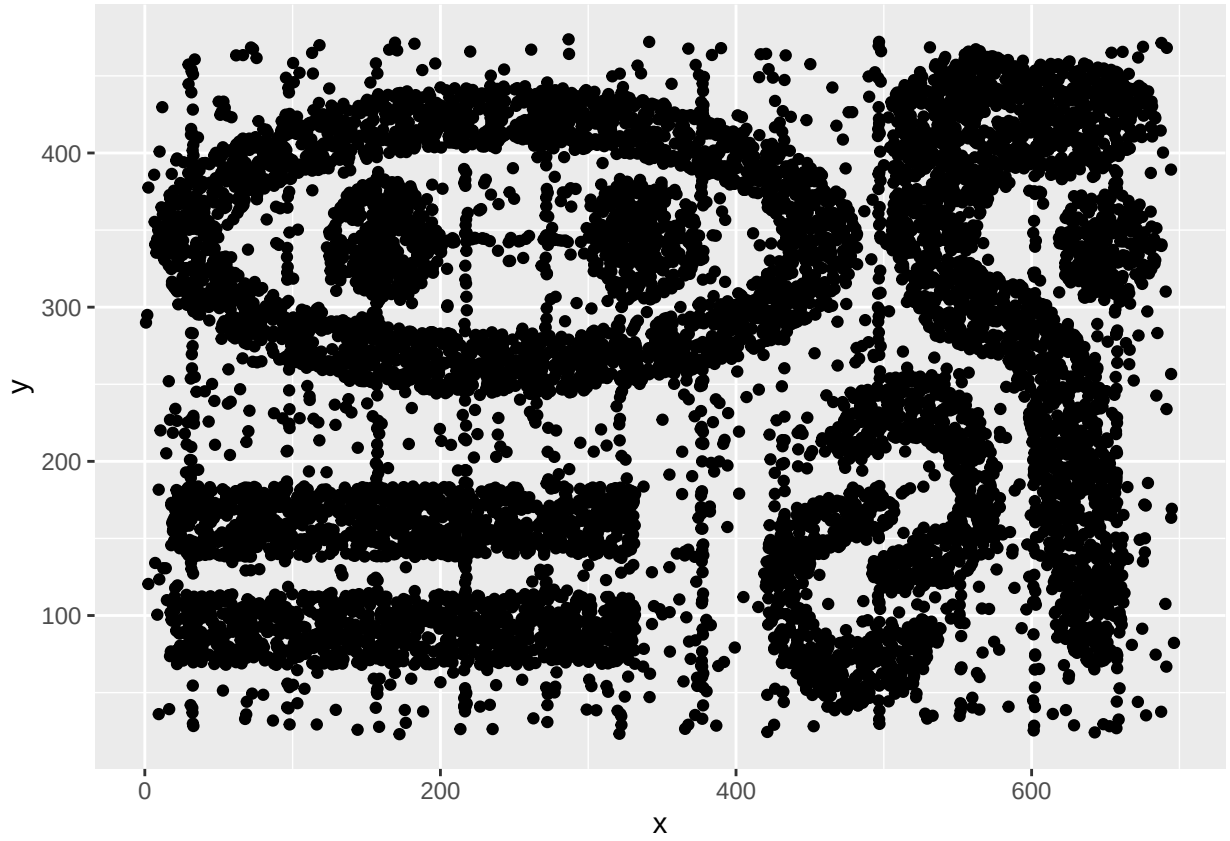
Question 12. A l'aide des questions précédentes, choisissez des paramètres pour obtenir un clustering à 4 classes. Comparez cette classification avec celle obtenue par les Kmeans pour le même nombre de classes.

```
# A COMPLETER
```

Clustering sur données simulées

Dans cette partie, on considère les données simulées "chameleon_ds7" disponibles dans la librairie `seriation`.

```
library(seriation)
data(Chameleon)
ggplot(chameleon_ds7,aes(x=x,y=y))+geom_point()
```



Question 13. Mettez en place une stratégie de classification de ces données par DBSCAN et par Kmeans. Comparez les résultats.

```
# A COMPLETER
```