TP7 : DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Pour réaliser ce TP, nous aurons besoin du package suivant :

```
install.package("dbscan")
```

La fonction DBSCAN()

L'algorithme DBSCAN est disponible au travers de la fonction *dbscan*.

```
dbscan(x, eps, minPts = 5, weights = NULL,
 borderPoints = TRUE, ...)
```

Pour utiliser cette fonction, il est nécessaire de spécifier au minimum deux paramètres :

— x : un data frame ou une matrice des données

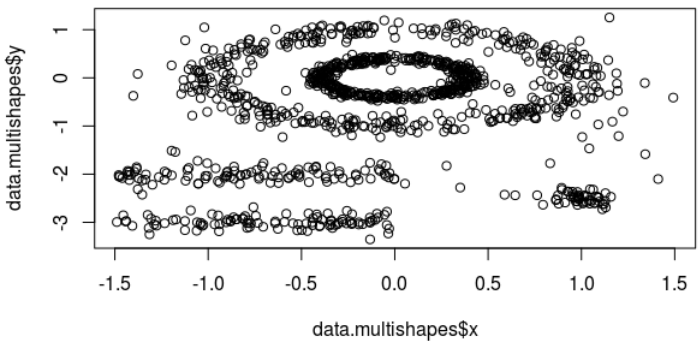
— *eps* : distance maximum pour définir le epsilon voisinage

Utiliser le help pour comprendre les autres paramètres.

Clustering d'un jeu de données

Charger le jeu de données "multishapes.txt" dans un data frame puis visualiser la répartition des points avec la fonction *plot*().

```
library(dbscan)
data.multishapes<- read.table("multishapes.txt", header = TRUE)
plot(data.multishapes$x, data.multishapes$y)
```



Appliquer la fonction *dbscan*() sur le data frame en fixant $eps = 0.15$ et $minPts = 5$

```
db <- dbscan(DF, eps = 0.15, minPts = 5)
db
afficherClusters(DF,length(db$cluster),db$cluster,"x","y")
```

Que contient l'objet *db* ? Combien de clusters ont été identifiés ? Combien de points ont été étiquetés comme bruit ? Faites varier les paramètres *eps* et *minPts*. Que constatez vous ?

Méthode pour déterminer la valeur de epsilon

La méthode que nous allons utiliser correspond au calcul des distances moyennes entre chaque point et ses k plus proches voisins. La valeur de k correspond à la valeur de $minPts$ utilisée dans *dbscan*()

La courbe des distances est tracée selon un ordre croissant. L'objectif est de déterminer le ■coude■ de la courbe, qui correspond au paramètre optimal *eps*.

```
kNNdist(DF, k=5)
kNNdistplot(DF, k =  5)
```