Contrôle continu, Master 1 MIASHS, 20 novembre 2015, 1h 25 minutes

Prénom, nom : correction

À faire :

- dans le nom de ce fichier, vous changez Prenom et Nom par votre prénom et votre nom ;
- vous répondez dans ce fichier ; vos réponses s'affichent en noir.
- à l'issue du contrôle, vous envoyez le **pdf** de ce fichier à philippe.preux@univ-lille3.fr

---

Exercice 1

Chargez le fichier se trouvant à l'url
http://www.grappa.univ-lille3.fr/~ppreux/ensg/miashs/m1/tps/cc/jd.csv
Le dernier attribut (la colonne portant le numéro le plus élevé) correspond à la classe de la donnée. Tous les autres attributs sont quantitatifs.

Question 1. comment faites-vous pour charger ce fichier dans R ?

```
jd <- read.csv ("http://www.grappa.univ-lille3.fr/~ppreux/ensg/miashs/m1/tps/cc/jd.csv")
```

Question 2. quelle commande R utilisez-vous pour connaître :
le nombre de lignes ?

```
nrow (jd)
```

le nombre de colonnes (attributs) ?

```
ncol (jd)
```

le nom des attributs ?

```
names (jd)
```

Question 3.
Combien y a-t-il de classes ?
On peut faire :

```
table (jd$cluster)
```

Il y a 16 classes.

Quel est l'effectif de chacune des classes ?
250 données dans chacune des classes.

Question 4. Comment déterminez-vous les données dont l'attribut 1 est supérieur à 1 et le deuxième attribut est positif et inférieur à 0,001 ?

```
which ((jd$a > 1) & (jd$b > 0) & (jd$b < 0.001))
```

Comment faites-vous pour savoir combien il y en ?

```
length (which ((jd$a > 1) & (jd$b > 0) & (jd$b < 0.001)))
```

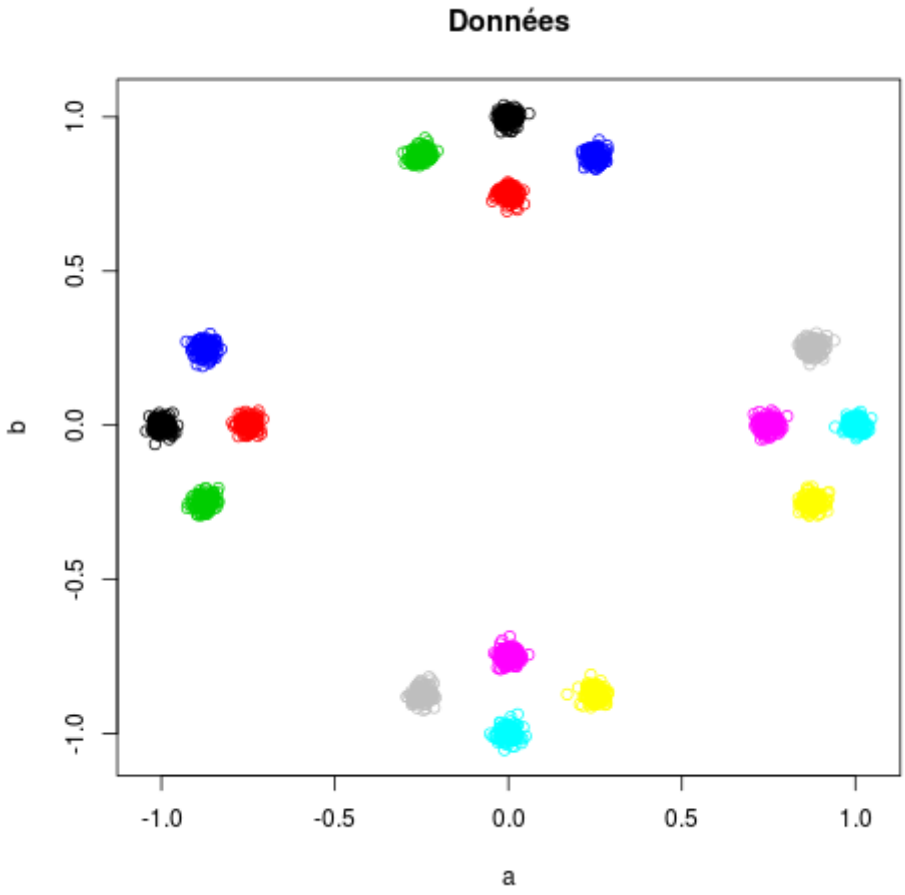
Question 5. On veut réaliser un graphique des données en mettant chaque point en couleur correspondant à la classe de la donnée, sans oublier les légendes.
Quelle(s) commande(s) tapez-vous pour créer ce graphique ?

```
plot (jd$a, jd$b, col = jd$cluster, main = "Données", xlab = "a", ylab = "b")
```

Comment faites-vous pour que le graphique soit mis dans un fichier au format png ?

```
png ("jd.png")
plot (jd$a, jd$b, col = jd$cluster, main = "Données", xlab = "a", ylab = "b")
dev.off ()
```

Insérez-le ci-dessous.



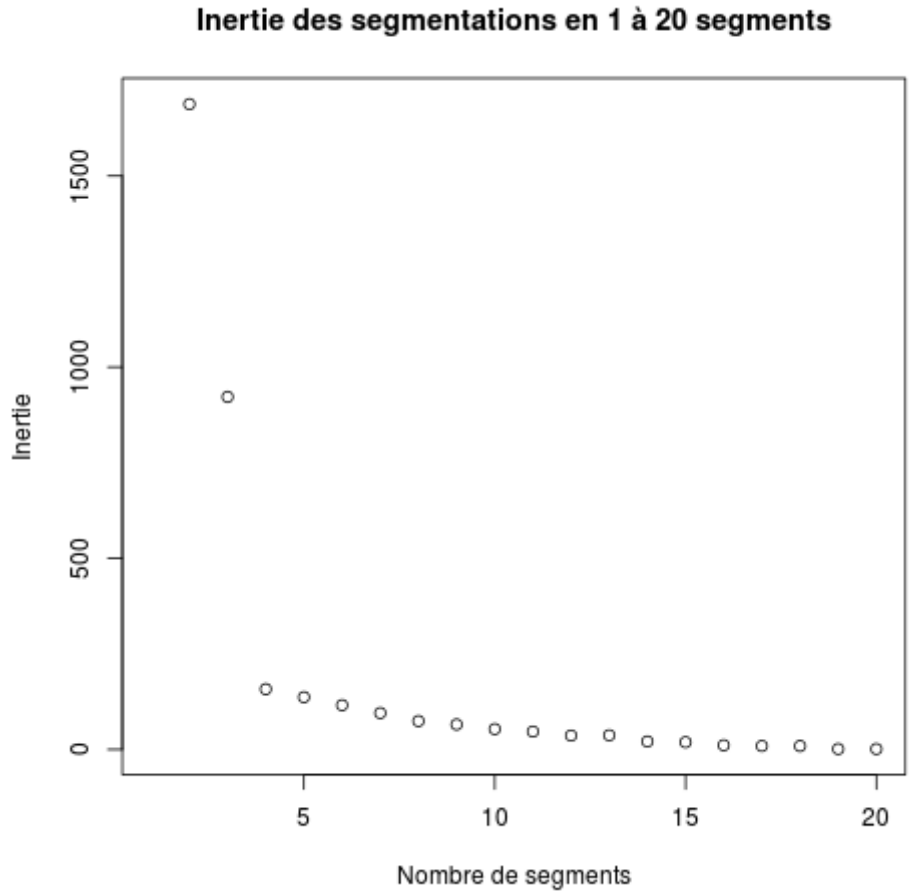
Question 6.

On veut réaliser une segmentation de ce jeu de données. Quand vous effectuez cette segmentation, vous utilisez tous les attributs sauf le dernier. Vous utilisez l'algorithme des k-moyennes. Comment déterminez-vous le nombre de groupes correct ?

On effectue des segmentations ayant de 1 à 20 segments ; pour chacune, on mesure l'inertie intraclasse et on détermine graphiquement le nombre de segments.

```
inertie <- rep (NA, times = 20)
jd.segmentations <- list ()
for (k in 2:20) {
  jd.segmentations [[k]] <- kmeans (jd[,-3], centers = k,
                                    iter.max = 30, nstart = 30)
  inertie [k] <- sum (jd.segmentations[[k]]$withinss)
}
plot (inertie)
```

On obtient ce graphique :



Le coude est à 4.

On veut comparer le résultat de cette segmentation avec celle correspondant à la classe des données (le dernier attribut).
Comment faites-vous cette comparaison graphiquement ?

```
plot (jd$a, jd$b, col = jd.segmentations [[4]]$cluster)
```

On observe 4 groupes, chacun regroupant 4 plus petits groupes.

Sur la graphique réalisé à la question 5, on avait observé les 16 groupes correspondant aux 16 classes, chacun de ces groupes étant compact et bien séparés les uns des autres. Avec la méthode utilisée ici, ce sont les 4 gros groupes qui sont trouvés facilement.

Comment faites-vous une table de confusion ?

```
table (jd$cluster, jd.segmentations [[4]]$cluster)
```

Question 7. Quand on regarde graphiquement le jeu de données, on peut aussi voir 4 ou 16 groupes.
Avec k-moyennes, segmentez le jeu de données en 4 groupes. Le résultat est-il celui que vous attendez ?

Oui, chacun des 4 groupes regroupe 4 petits groupes. C'est bien à cela que l'on s'attend si l'on faut 4 groupes.

Avec k-moyennes, segmentez le jeu de données en 16 groupes. Le résultat est-il celui que vous attendez ?

Non le résultat n'est pas celui que l'on attend. Bien que le résultat de `kmeans()` soit variable, j'obtiens toujours un petit groupe qui mélange deux segments obtenus par `kmeans()`.

La table de contingence confirme ce que l'on voit sur le graphique.

Question 8. Même question en faisant une segmentation hiérarchique.

On calcule la segmentation hiérarchique :

```
jd.hclust <- hclust (dist (jd[,-3]) )
```

que l'on découpe en 4 groupes :

```
jd.hclust.4groupes <- cutree (jd.hclust, k = 4)
```

et en 16 groupes :

```
jd.hclust.16groupes <- cutree (jd.hclust, k = 16)
```

Ensuite, on peut inspecter visuellement les deux segmentations :

```
plot (jd$a, jd$b, col = jd.hclust.4groupes)
plot (jd$a, jd$b, col = jd.hclust.16groupes)
```

Les deux graphiques donnent l'impression que la segmentation est cette fois parfaite. La table de contingence permet de vérifier cela :

```
table (jd$cluster, jd.hclust.4groupes)
table (jd$cluster, jd.hclust.16groupes)
```

L'une ou l'autre des méthodes semble-t-elle mieux retrouver les classes ?

La segmentation hiérarchique fonctionne mieux sur ce jeu de données. Les segmentations attendues sont parfaitement obtenues.