Examen partiel du 8 mars 2021 - Correction de l'exercice 2

```
install.packages("DiscriMiner")
```

```
##
## The downloaded binary packages are in
##  /var/folders/0x/g2ff954d47g0ldvtz4hvr43r0000gn/T//Rtmpb1n1R5/downloaded_packages
```

```
library(ade4)
library(MASS)
library(DiscriMiner)
```

Question 1

```
data("fgl")
?fgl
head(fgl)
```

```
##      RI    Na   Mg   Al    Si    K   Ca Ba   Fe type
## 1  3.01 13.64 4.49 1.10 71.78 0.06 8.75  0 0.00 WinF
## 2 -0.39 13.89 3.60 1.36 72.73 0.48 7.83  0 0.00 WinF
## 3 -1.82 13.53 3.55 1.54 72.99 0.39 7.78  0 0.00 WinF
## 4 -0.34 13.21 3.69 1.29 72.61 0.57 8.22  0 0.00 WinF
## 5 -0.58 13.27 3.62 1.24 73.08 0.55 8.07  0 0.00 WinF
## 6 -2.04 12.79 3.61 1.62 72.97 0.64 8.07  0 0.26 WinF
```

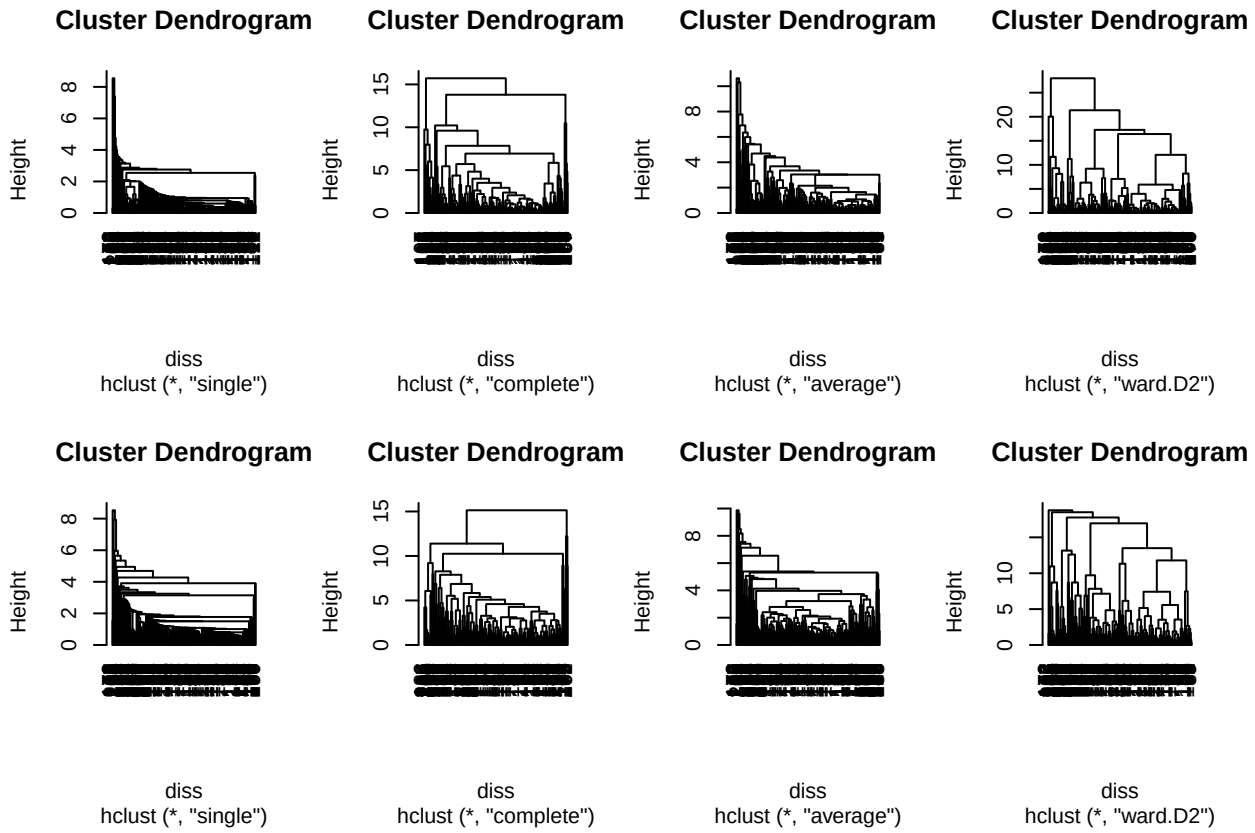
```
donnees = fgl[-10]
```

Question 2

La première variable donne des décimales d'un indice de réfraction ; les suivantes donnent toutes des pourcentages, mais ont des ordres de grandeurs différents. Par conséquent il vaut mieux ne pas utiliser la distance euclidienne simple. On peut utiliser la distance euclidienne normalisée, ou la distance de Mahalanobis.

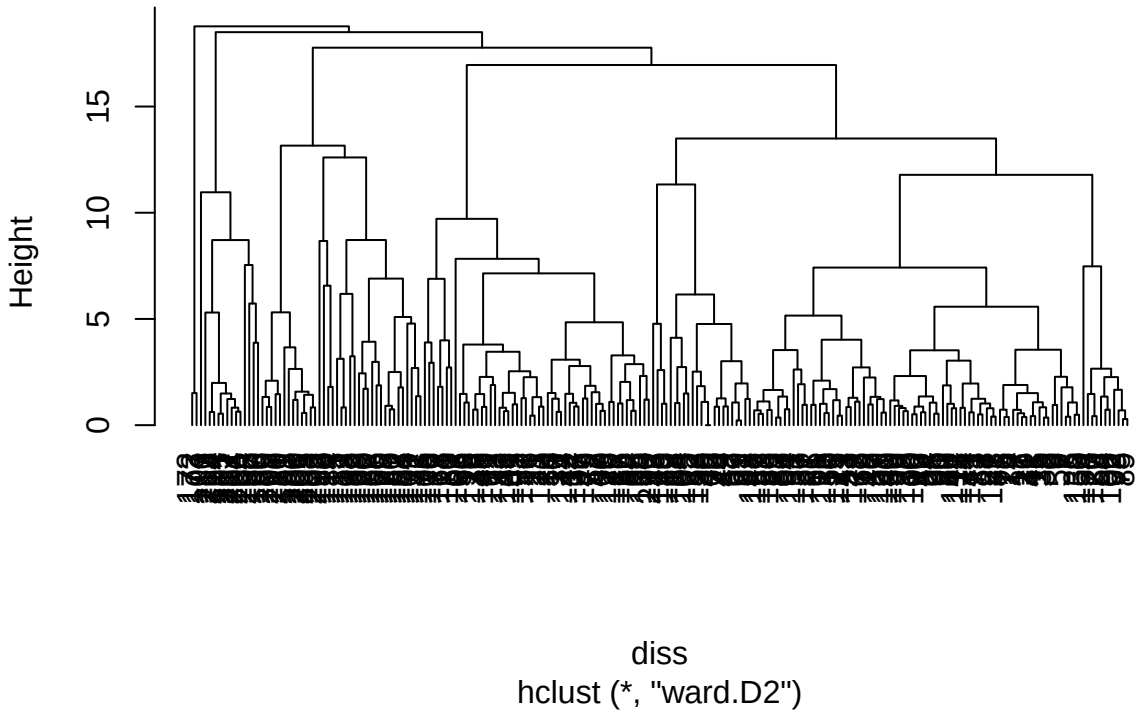
Question 3

```
dissimilarites = c(2,3) # distance euclidienne normalisée ou de Mahalanobis
strategies = c("single", "complete", "average", "ward.D2")
par(mfrow=c(2,4))
for(d in dissimilarites)
{
  diss = dist.quant(donnees, d)
  for(s in strategies)
  {
    cah = hclust(diss, method=s)
    plot(cah, hang=-1)
  }
}
```



Les deux dendrogrammes obtenus avec la stratégie de Ward sont les seuls qui présentent des branches pas trop déséquilibrées. On va choisir la dissimilarité de Mahalanobis avec la stratégie de Ward (dengrogramme en bas à droite), car le dendrogramme montre un saut net du critère pour le passage de 5 à 4 classes, ce qui donnera une partition plus intéressante qu'avec la distance euclidienne normalisée (partition en deux classes assez déséquilibrées)

```
diss = dist.quant(donnees, 3)
cah = hclust(diss, method="ward.D2")
plot(cah,hang=-1)
```



Question 4

Le saut maximal du critère semble clairement correspondre au passage de 5 à 4 classes. Vérifions le par le calcul. On calcule d'abord tous les sauts successifs du critère :

```
sauts = diff(cah$height)
length(sauts)
```

```
## [1] 212
```

Il y a 212 sauts du critère, le dernier correspondant à la dernière étape (passage de 2 à 1 classe). Déterminons à présent la position du saut maximal :

```
which.max(sauts)
```

```
## [1] 209
```

212-209=3, donc le saut maximal correspond au passage de 5 à 4 classes. On sélectionne donc la partition en 5 classes :

```
cl = cutree(cah, k=5)
cl
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   2   1   1   1   2   2   1   2   2   1   1   1   1   1   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   1   1   1   1   1   1   1   1   1   2   1   2   2   1   1   1   1   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   1   1   1   2   1   2   1   1   1   1   2   1   1   2   2   2   1   1   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   1   1   1   1   1   1   1   1   1   1   2   2   1   1   1   1   1   1   2   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
```

```
##   1   1   1   2   2   1   1   2   1   1   2   1   2   1   1   1   2   2   1   1
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   1   2   1   1   2   3   4   4   4   4   4   4   2   1   1   2   1   2   1
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   1   2   1   1   1   2   1   2   2   2   4   4   1   2   1   2   2   1   1   1
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   1   2   2   1   2   2   1   1   2   1   2   1   1   1   1   1   1   1   1   2
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   2   2   3   4   4   4   4   4   4   4   5   5   4   2   2   4   1   4   4
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   4   4   4   4   4   3   3   1   1   3   1   4   4   3   3   4   4   4   4   4
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214
##   4   4   4   3   4   3   3   3   4   4   3   1   3   3
```

Question 5

```
table(cl,fgl[,"type"])
```

```
##
## cl  WinF WinNF Veh Con Tabl Head
##   1   53    39  12   0    1    4
##   2   17    28   5   2    0    0
##   3    0     1   0   1    0   12
##   4    0     8   0   8    8   13
##   5    0     0   0   2    0    0
```

La partition ne correpond que partiellement à la répartition en tyep de verre : les classes 1 et 2 obtenues correspondent à peu près aux trois premiers types de verre WinF, WinNF, Veh, tandis que le type Head se retrouve principalement dans les classes 3 et 4. A part cela, les partitions sont assez différentes.

Question 6

Calculer directement l'inertie intra classe sur les données supposerait que l'on travaille avec la distance euclidienne simple ; or on a dit que ce n'était pas un bon choix. Par conséquent on va normaliser les données auparavant pour faire en sorte de travailler avec les distances euclidiennes normalisées (car les distances euclidienne normalisée correspondent aux distances euclidiennes simples sur les données normalisées). N.B. On pourrait également transformer les données pour que le calcul corresponde aux distances de Mahalanobis.

Pour calculer les inerties intra classes on utilise la fonction withinSS, qui renvoie une matrice ; dont il faut en fait prendre la somme des coefficients diagonaux.

```
donnees_normalisees = scale(donnees)
# inertie intra pour la partition obtenue par la CAH :
sum(diag(withinSS(donnees_normalisees,cl)))
```

```
## [1] 1100.128
```

```
# inertie intra pour la partition induite par le type de verre :
sum(diag(withinSS(donnees_normalisees,fgl[,"type"])))
```

```
## [1] 1413.027
```

On ne peut pas a priori comparer des inerties intra classe pour des partitions n'ayant pas le même nombre de classes. En effet, plus le nombre de classes augmente, et plus l'inertie intra classe a tendance à diminuer (jusqu'à une inertie intra classe égale à 0 pour la partition finale triviale en n classes). Cependant ici on voit que l'inertie intra classe pour la partition en 4 classes induite par la CAH (=1100) est plus petite que l'inertie intra classe pour la partition en 6 classes par type de verre (=1413). Ceci signifie que la partition par type de verre n'est clairement pas optimale vis à vis du critère de l'inertie intra classe. En choisissant une partition

en 6 classes (question suivante) pour la CAH, on s'attend à avoir une inertie intra classe inférieure à 1100, et donc bien plus petite que celle pour la répartition par type de verre.

Question 7

```
# partition en 6 classes induite par la CAH :
cl = cutree(cah, k=6)
# inertie intra pour cette partition :
sum(diag(withinSS(donnees_normalisees,cl)))
```

```
## [1] 972.4876
```

```
# inertie intra pour la partition induite par le type de verre :
sum(diag(withinSS(donnees_normalisees,fgl[,"type"])))
```

```
## [1] 1413.027
```

Cette fois les valeurs des inerties intra classe sont directement comparables, et comme attendu, la valeur pour la partition induite par la CAH (=972) est bien inférieure à celle induite par le type de verre (=1413). On en conclut que la partition par type de verre n'est pas du tout optimale vis à vis du critère de l'inertie intra classe. Autrement dit, les classes correspondant aux types de verre sont assez mélangées ; on arrive à mieux partitionner les observations en groupes séparés par des méthodes de classification non supervisée.