TP1 : classification ascendante hiérarchique

Exercice 1

On considère le tableau de données suivant où 4 individus (ici des points) A, B, C et D sont décrit sur deux variables (X1 et X2):

	X1	X2
A	5	4
B	4	5
C	1	-2
D	0	-3

1. Constuire le dendrogramme du **lien maximum** des 4 individus.
2. Constuire le dendrogramme de **Ward** en pondérant les individus par 1.
3. Calculer l'inertie intra-classe de la partition en deux classes issue du dendrogramme de Ward.
4. Calculer l'inertie total et en déduire le pourcentage d'inertie expliquée par cette partition.

Exercice 2

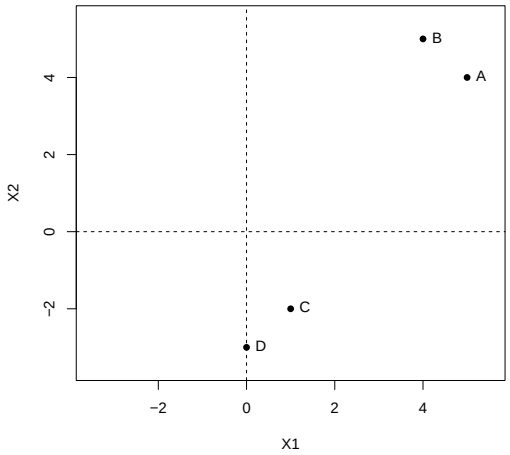
Lobjectif est de retrouver **avec R** les résultats de l'exercice 1.

1. Créer une matrice X contenant les données.

```
X
```

```
##   X1 X2
## A  5  4
## B  4  5
## C  1 -2
## D  0 -3
```

2. Visualiser le jeu de données avec les fonctions `plot`, `text` et `abline`.

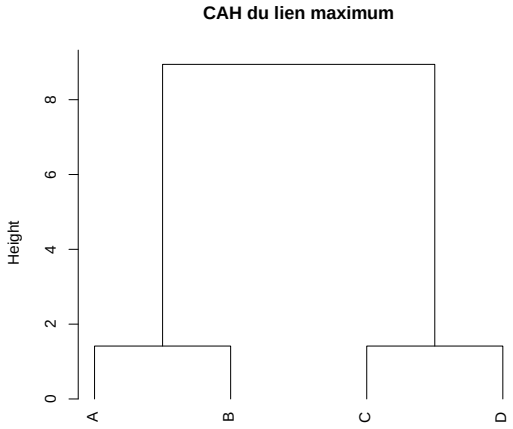


3. Construire la hiérarchie du **lien maximum** avec la fonction `hclust`. Vérifiez que vous retrouver les hauteurs de l'exercice 1.

```
tree$height # donne la hauteur des classes dans le dendrogramme
```

```
## [1] 1.414214 1.414214 8.944272
```

4. Représenter le dendrogramme de cette hiérarchie.



5. Coupez ce dendrogramme pour obtenir la partition en deux classes de la CAH du lien max.

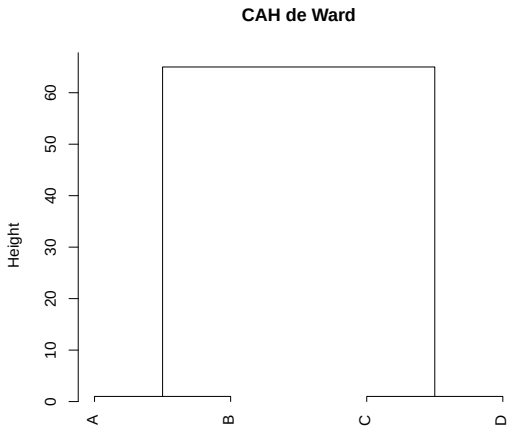
```
cutree(tree,k = 2)
```

```
## A B C D
## 1 1 2 2
```

6. Construire maintenant la hiérarchie de **Ward** en vous aidant de l'annexe en fin du TP pour paramétrer correctement cette fonction et retrouver les hauteurs de l'exercice 1.

```
## [1]  1  1 65
```

7. Représenter le dendrogramme de cette hiérarchie.



8. Retrouver à partir des hauteurs de ce dendrogramme:
   - l'inertie totale,
   - l'inertie inter-classe de la partition en deux classes.

En déduire la part de la variance totale expliquée par cette partition.

```
B/Tot # part d'inertie expliquée par la partition
```

```
## [1] 0.9701493
```

Exercice 3.

On considère le jeu de données `fromage` où $n = 29$ fromages sont décrit sur $p = 9$ variables quantitatives. On souhaite déterminer automatiquement des groupes de fromages qui se ressemblent et caractériser ces groupes.

Les lignes de commande ci-dessous permettent de réaliser une classification ascendante hiérarchique de Ward (**en pondérant les individus par 1/n**) :

1. Importer les données depuis le fichier `fromage.txt`.

2. Calculer l'écart-type des 9 variables. Que pouvez-vous en conclure ?

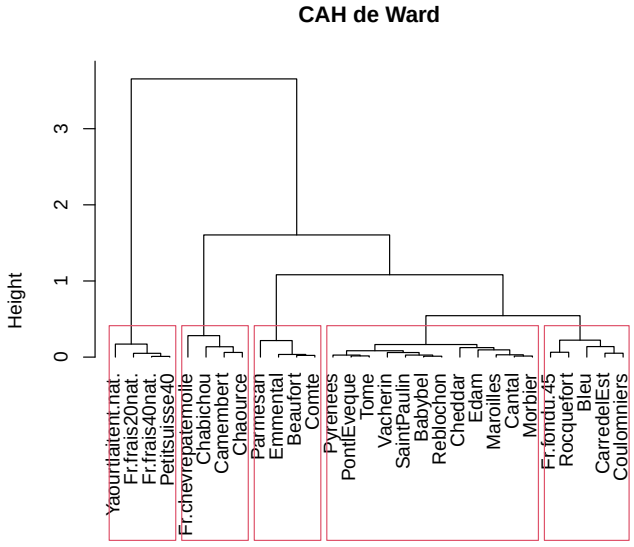
```
round(apply(cheese, 2, sd), digit=2)
```

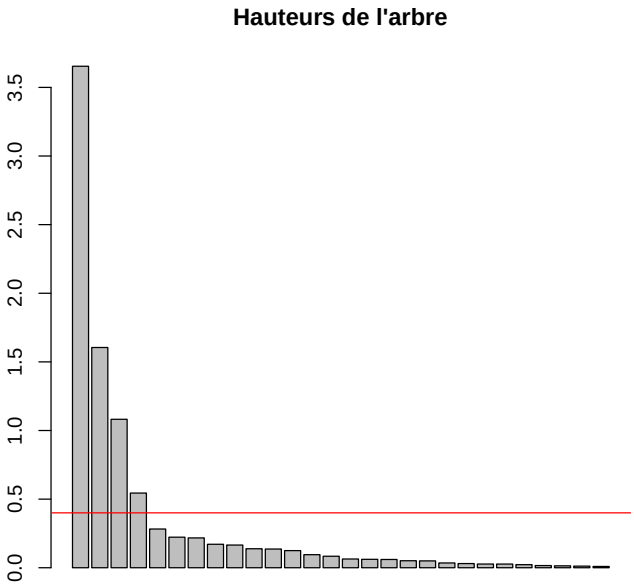
```
##    calories      sodium     calcium     lipides     retinol     folates
##       91.91      108.68       72.53        8.13       24.16       11.72
##   proteines cholesterol   magnesium
##        6.96       28.25       11.32
```

3. Centrer et réduire les données (en utilisant la variance empirique non corrigée). A quoi sont égales les moyennes et les variances des variables ainsi standardisées ?

3. Constuire la CAH de Ward en pondérant tous les individus par $\frac{1}{n}$. Vérifiez que la somme des hauteurs est bien égale à 9. Expliquez pourquoi.

4. Constuire le dendrogramme et l'ébouli des hauteurs de l'arbre. Pourquoi le choix de $K = 5$ classes vous semble pertinent ?



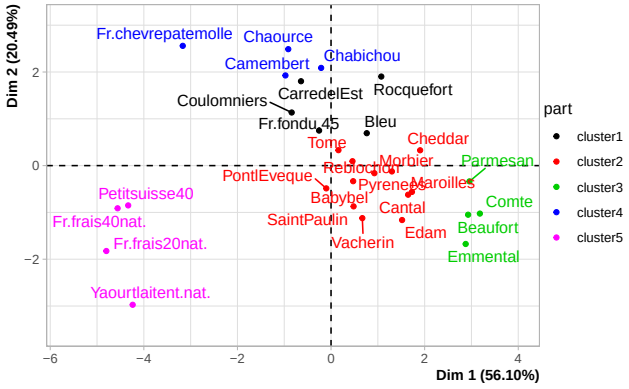


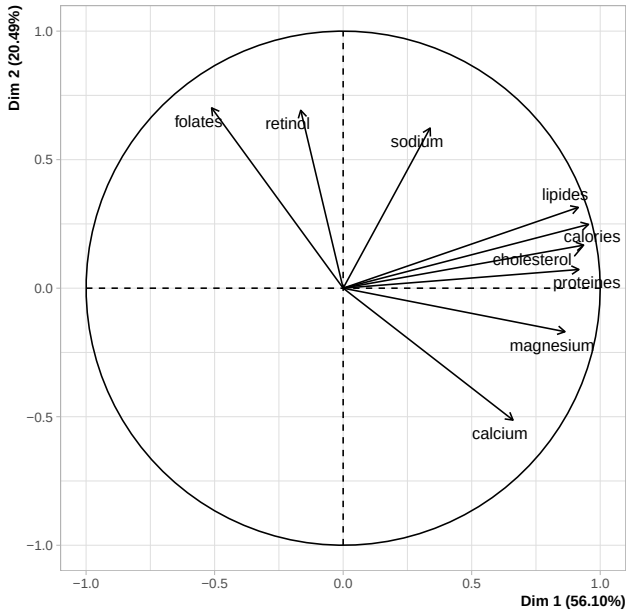
4. Quelle est le pourcentage de l'inertie des données expliquée par cette partition en 5 classes ?
5. On veut maintenant interpreter la partition en 5 classes des fromages à l'aide d'une ACP.

```
#----------------------------------------
# Partition en 5 classes
#----------------------------------------
K <- 5
part <- cutree(tree,k=K)
```

Retrouvez et interprétez les résultats et les graphiques ci-dessous et en déduire une première interprétation des classes.

```
##        eigenvalue percentage of variance cumulative percentage of variance
## comp 1      5.049                 56.101                            56.101
## comp 2      1.844                 20.494                            76.595
## comp 3      0.868                  9.642                            86.237
## comp 4      0.577                  6.415                            92.652
## comp 5      0.355                  3.947                            96.599
## comp 6      0.175                  1.948                            98.547
## comp 7      0.097                  1.083                            99.630
## comp 8      0.028                  0.316                            99.946
## comp 9      0.005                  0.054                           100.000
```





6. On veut maintenant interpréter les classes à l'aide de la fonction `cates` du package FactoMineR. On s'intéresse aux résultats pour la classe 5 obtenus avec le code ci-dessous:

```
##             v.test Mean in category Overall mean sd in category Overall sd
## magnesium     -3.0             11.2           27            1.6       11.1
## sodium        -3.3             44.8          210           27.7      106.8
## proteines     -4.0              7.2           20            2.0        6.8
## cholesterol   -4.3             18.2           75            7.7       27.8
## calories      -4.6            101.8          300           28.6       90.3
## lipides       -4.7              6.3           24            3.0        8.0
##               p.value
## magnesium   2.8e-03
## sodium      1.0e-03
## proteines   6.0e-05
## cholesterol 1.7e-05
## calories    3.4e-06
## lipides     2.2e-06
```

- Retrouver la valeur-test -3 de la modalité `magnesium` et en déduire la p-valeur de 0.0028 (utiliser la fonction `pnorm`).
- Pourquoi toutes les modalités ne sont pas affichées dans ces résultats ?
- Interpréter la classe 5.

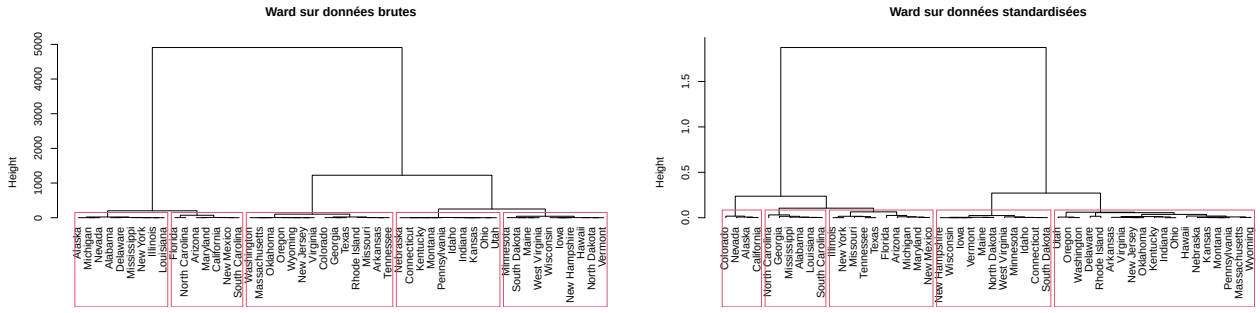
Exercice 4 : pourquoi standardiser les données ?

On utilise les données `USArrests` où 50 états Amériains sont décrit par :

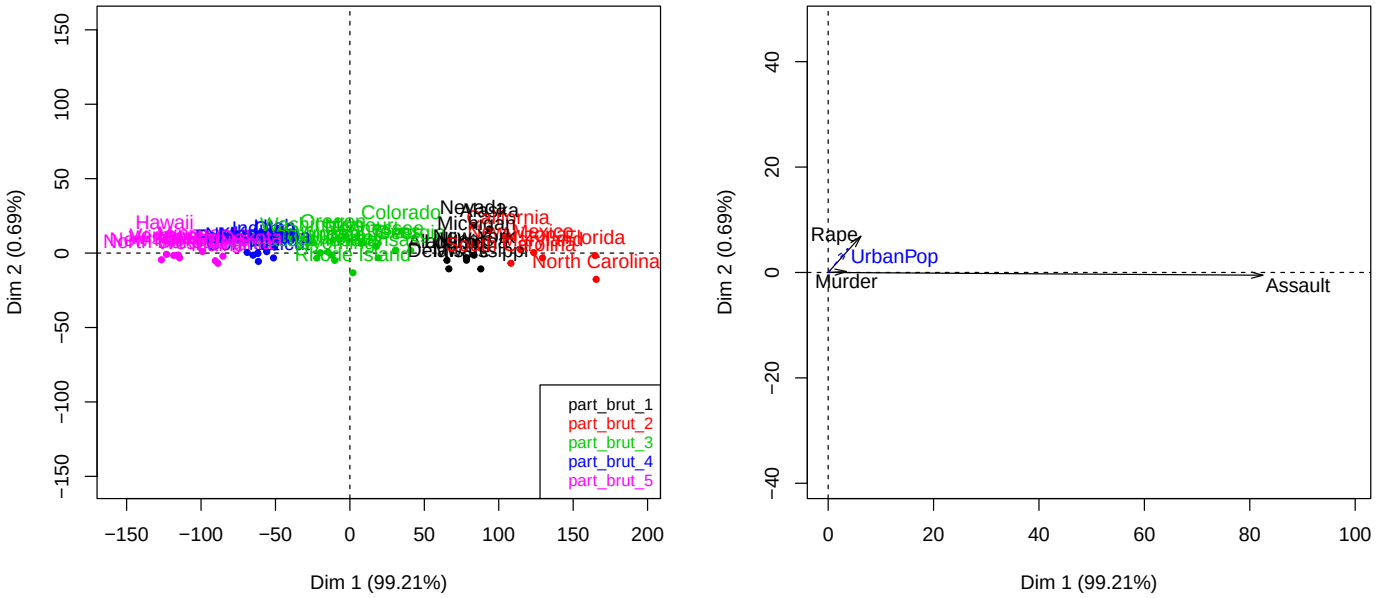
- 3 variables indiquant le nombre (pour 100000 habitants) d'arrestation pour meurtre (Murder), agressions (Assault), viol (Rape),
- une variable indiquant la proportion de population urbaine (UrbanPop).

On veut trouver des classes d'états qui se ressemblent sur les 3 variables de crimes.

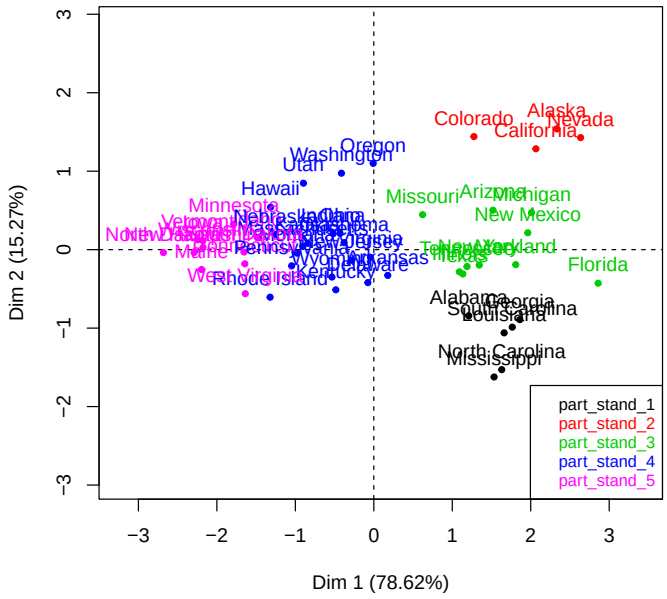
1. A votre avis, faut-il standardiser les données avant de faire un clustering ?
2. Faire un clustering par CAH de Ward sur les données brutes, puis sur les données standardisées.

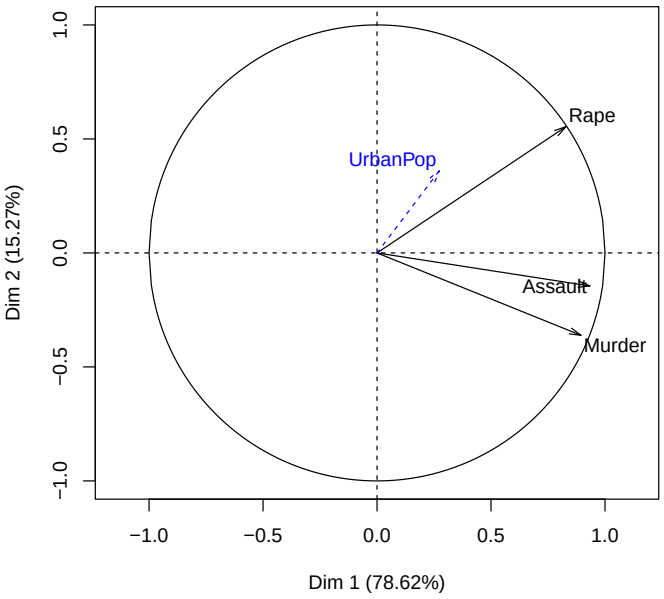


3. Visualiser le clustering des données brutes sur le premier plan factoriel d'une ACP **non normée**. Interpréter la partition en 5 classes obtenue sur les données brutes. Qu'en pensez-vous ?

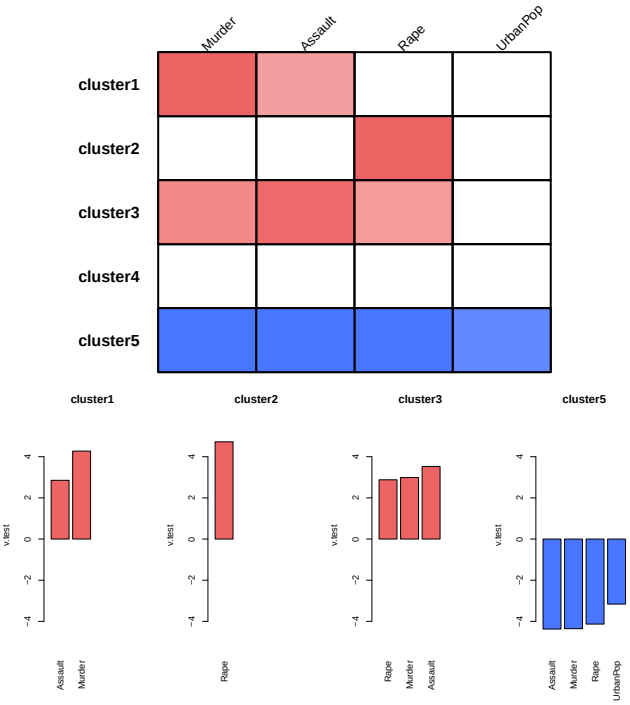


3. Visualiser le clustering des données standardisées sur le premier plan factoriel d'une ACP non normée. Interpréter la partition en 5 classes obtenue sur les données standardisées. Qu'en pensez-vous ?





4. Interprétez maintenant la partition en 5 classes des données standardisées avec la fonction `catdes`. Retrouvez les graphiques ci-dessous avec la méthode `plot` de la classe `catdes`. Pensez-vous que le choix de 5 classes était judicieux ?



Exercice 5 : clustering après réduction de la dimension

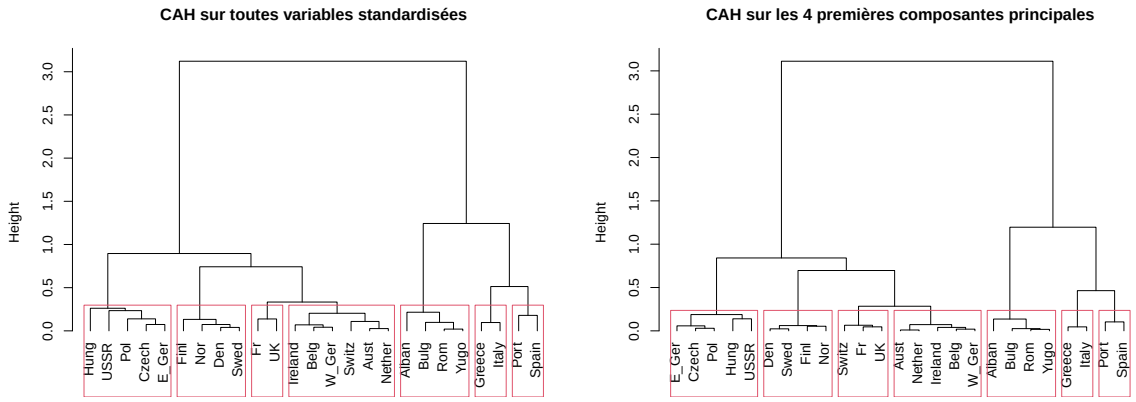
On utilise le jeu de donnéees `protein`.

```
load("../data/protein.rda")
```

L'idée est de réduire la dimension avant d'appliquer le clustering pour trouver automatiquement des groupes de pays ayant les mêmes profils de consommation en protéines. Pour cela, on peut réaliser une ACP et

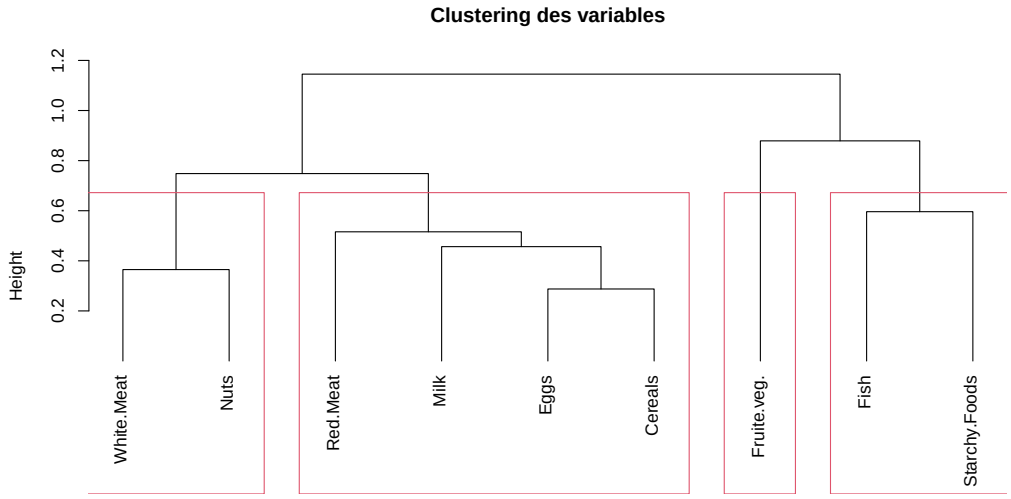
conserver q composantes principales (q à choisir) ou encore réaliser un clustering de variables en J classes et conserver les J variables synthétiques des classes (première composante principale de l'ACP des variables de la classe).

1. On réduit d'abord la dimension par ACP normée avant de faire le clustering. Il faut donc choisir le nombre q de composantes principales à retenir.
   - Quel est le nombre maximal r de composantes principales pour ces données ? Vérifier que le dendrogramme de Ward obtenu avec les données standardisées est identique à celui obtenu avec les r composantes principales (toutes).
   - On décide de conserver $q = 4$ composantes principales. Quel est le pourcentage d'information conservé par ces nouvelles variables ?
   - Comparer alors le dendrogramme de Ward obtenu avec ces 4 premières composantes principales à celui obtenu avec les données standardisées.



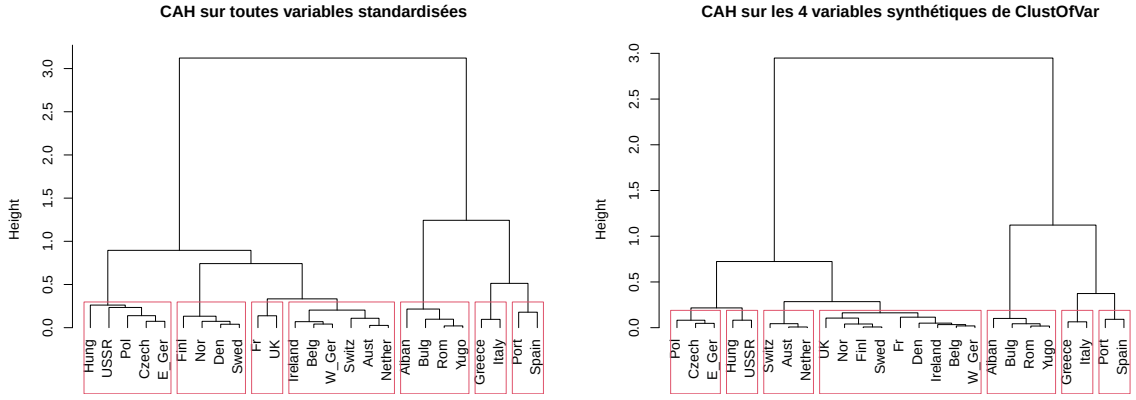
2. On réduit maintenant la dimension par clustering de variables. Il faut donc choisir le nombre J de clusters à retenir.
   - Utiliser le code ci-dessous pour obtenir un arbre hiérarchique des neufs variables. Le clustering de variable a pour objectif de trouver des clusters de variables corrélées entre elles (positivement ou négativement). On choisit en regardant cet arbre de retenir $J = 4$ cluster de variables.

```
library(ClustOfVar)
treevar <- hclustvar(protein)
plot(treevar, main="Clustering des variables")
rect.hclust(treevar, k=4)
```



- Comparer alors le dendrogramme obtenu avec les 4 variables synthétiques à celui obtenu avec les données standardisées. Quel peut-être l'avantage de cette méthode de réduction de dimension ?

```
cov <- cutreevar(treevar, k = 4)
# Nouvelles données avec 4 variables synthétiques
newdat3 <- cov$scores
```



Annexe : faire la CAH de Ward avec hclust

On note $\Delta = [\delta_{ij}]$ la matrice des mesures d'aggrégation de Ward entre les singletons :

$$\delta_{ij} := \frac{w_i w_j}{w_i + w_j} d_{ij}^2,$$

où d_{ij} est la distance Euclidienne entre les individus i et j et w_i et w_j leurs poids.

Le paramétrage de la fonction **hclust** qui permet de faire du "vrai" Ward est obtenu avec :

- `method = "ward.D"`,
- `d` = Δ,
- `members = NULL` pour des poids uniformes,
  `members = w` sinon (ou w est le vecteur des poids des individus).

Lorsque les poids w_i sont uniformes c'est à dire tous égaux à 1 ou à $\frac{1}{n}$ on peut utiliser le code suivant :

```
> d <- dist(X)
> tree <- hclust(d^2/(2*n), method="ward.D") # poids égaux à 1/n
> tree <- hclust(d^2/2, method="ward.D") # poids égaux à 1
```

Sinon on peut utiliser le code ci-dessous

```
> delta <- agr_singleton(d, w)
> tree <- hclust(delta, method="ward.D", members=w)
```

où `agr_singleton` est la fonction :

```
agr_singleton <- function(d, w)
{
  Delta <- as.matrix(d)
  n <- nrow(Delta)
  for (i in 1:(n-1)) {
    for (j in (i+1):n) {
      Delta[i,j] <- Delta[i,j]^2*w[i]*w[j]/(w[i]+w[j])
      Delta[j,i] <- Delta[i,j]
      }
  }
  return(as.dist(Delta))
}
```