

BING-F4002 Acquisition et analyse de données

Fiche TP 2 : Distances et groupements

Dufrêne M. - Gilbert M.

(avec la collaboration initiale de - Barbier N. - Deblauwe V.)

Version d'octobre 2015

Le but du présent TP est de vous permettre de réaliser des analyses multivariées de base pour les distances et les groupements.

Modules utilisés à charger :

- library(vegan)
- library(cluster)

Fichiers de données à charger :

- **carabides_32sta_103esp.txt** = abondance brute de 103 espèces de Carabides de 4 stations dans 8 milieux bien différents (Bas-marais, Landes minérales, Landes sablonneuses, Landes tourbeuses, Pelouses calcaires, Pelouses calaminaires, Prairies alluviales, Tourbières hautes) pendant un cycle annuel (Thèse MD – 1992).
- **carabides_32sta_12eco.txt** = description de 12 variables écologiques de 4 stations dans les 8 milieux bien différents (Habitats, X, Y, Altitude, pHeau, pHKCL, P, K, Ca, Mg, Na, Humidité) - Thèse MD – 1992.
- **Demazy_2013_carabides.txt** = abondance brute de 44 espèces de Carabides récoltées au printemps 2013 dans 18 stations (B = Bio et Non-labour, T = en transition et X = agriculture fonctionnelle) – TFE d'Ana Lerchs) – 2013.
- **Xylobios_ecologie.txt** = données écologiques du projet XYLOBIOS. Pour les codes des variables, voir la table 2 dans le rapport final disponible sur <http://www.belspo.be/belspo/fedra/proj.asp?l=fr&COD=EV/15>.

TP : Q1

Lire les fichiers de données brutes du fichier « carabides_32sta_103esp.txt » et « carabides_32sta_12eco.txt » et identifier une manière simple pour vérifier que les données sont *à priori* correctement lues.

1. Transformation de données

1.1. vegan

Vegan propose la fonction **decostand()** pour assurer une large diversité de transformation des données.

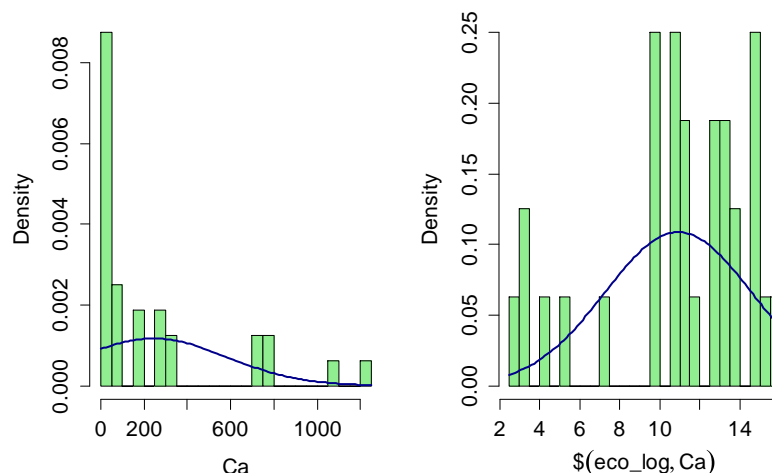
Table_transformee = decostand(table_originale, method="", MARGIN=1 ou 2)

Les différentes **method** sont les suivantes :

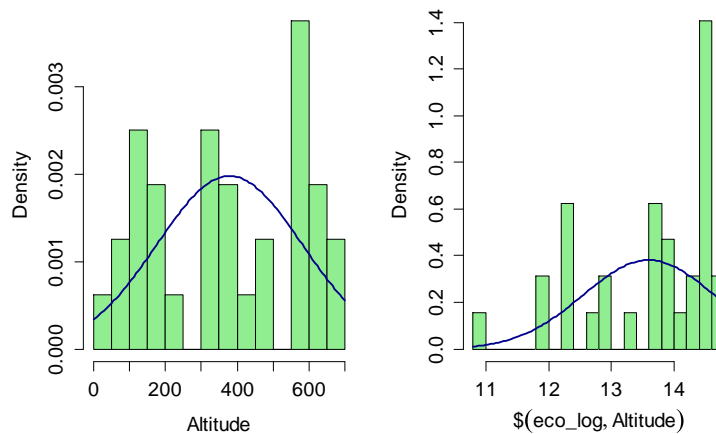
- **total** : divise par le total de la ligne si **margin = 1** ou de la colonne si **margin =2**
- **max** : divise par le maximum (**margin 1 ou 2**)
- **freq** : divise par le total et multiplie par la fréquence (**margin 1 ou 2**)
- **normalize** : somme des carrés des écarts = 1 (**margin 1 ou 2**)
- **range** : ramène les écarts maximum entre 0 et 1 (**margin 1 ou 2**)
- **standardize** : moyenne de 0 et variance unitaire (**margin 1 ou 2**)
- **pa** : en absence/présence
- **chi.square** : cfr AFC (**margin 1 ou 2**)
- **hellinger** : racine carré de **method=total**
- **log** : $\log(x)+1$ la base est définie par l'option **logbase=**

La transformation de Hellinger réduit l'importance des fortes abondances et semble intéressante pour les données d'absence/présence et les abondances d'espèces (Borcard et al, 2011). Toutefois, l'utilisation de pourcentages pour des données d'abondances d'espèces peut provoquer des variations importantes d'abondance d'une espèce alors qu'elle avait au départ les mêmes nombres d'individus. C'est potentiellement problématique lorsque les espèces ont des variations de fréquences fortes ou des détectabilités différentes.

Exemple d'une variable qui a pas besoin d'une transformation log :



Exemple d'une variable qui n'a pas besoin de transformation et où une transformation log provoque un décalage non désiré de la distribution :



TP : Q2

Analyser les 9 variables du fichier « carabides_32sta_12eco.txt » (hors habitats, X et Y) pour visualiser leur distribution et proposer une transformation adhoc, si nécessaire. Après avoir vu la Q3, comment pourriez-vous montrer l'impact des transformations ?

2. Calcul de distances

2.1. vegan

Vegan vous propose les distances suivantes avec la fonction **vegdist()** :

Matrice_distance = vegdist(table, method="" [, binary=TRUE])

Avec method =

- "euclidean" = distance euclidienne (D1) =
- "manhattan" = distance de Manhattan (D7) =
- "gower" = distance de Gower (DS15) =
- "bray" = distance de Bray-Curtis ou Steinhaus (D14 ou DS17)
- "jaccard" = distance de Jaccard (DS7)
- "kulczynski" = distance de Kulczynski (DS18)
- ...

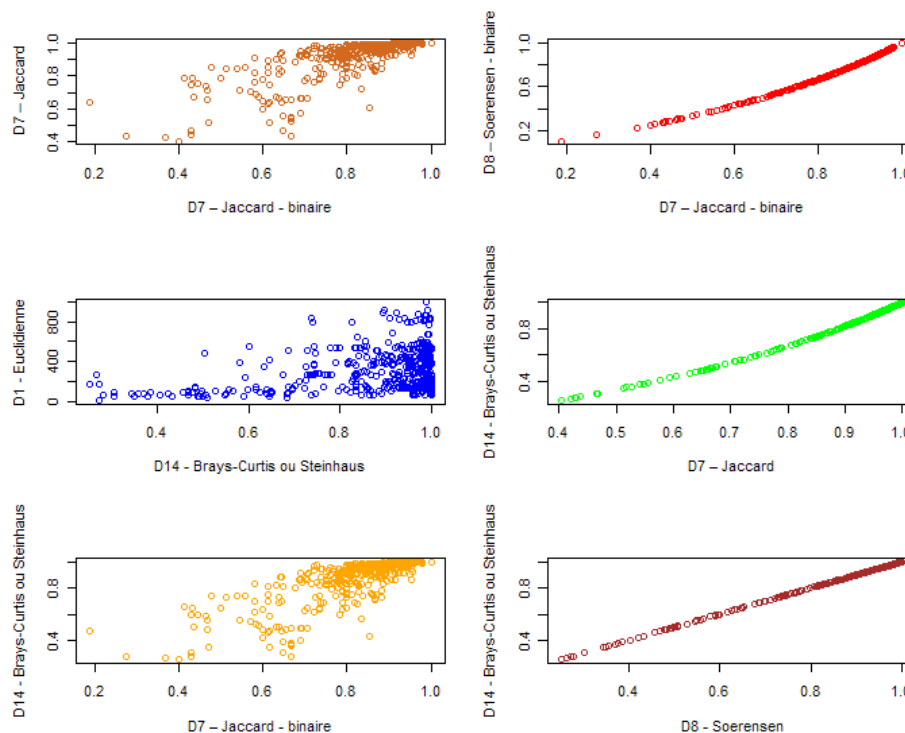
Pour les indices destinés à gérer des **données d'absence-présence** (method= "jaccard" ; method = "bray" pour l'indice de Soerensen; ...), il faut utiliser l'option **binary = TRUE**.

Une autre manière de produire ces indices et d'autres indices basés sur les valeurs a (double présence), b ou c (absence d'une des deux variables) ou d (double absence) est d'utiliser la fonction **designdist()**.

s7 <- designdist(table, method ="(a)/(a+b+c)", abcd= TRUE)

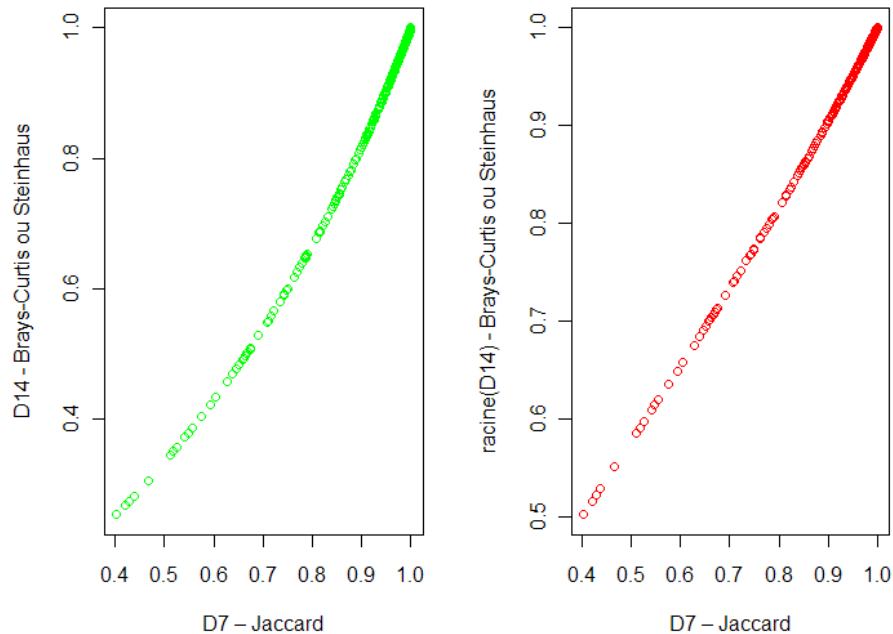
qui permet de construire son propre indice de similarité avec les valeurs a, b, c et d.

- ⚠ Attention, rappel => cette fonction construit un indice de similarité et non de distance. Il faut donc la transformer en distance pour les analyses de groupement et d'ordination avec la transformation $DS7 = 1 - S7$ ou $DS7 = \sqrt{1 - S7}$ pour avoir des distances qui ont plus de chances d'être métrique plutôt que semi-métriques (cfr cours).
- ⚠ Attention, rappel => Si vous utilisez ADE4, les valeurs de distance sont d'office transformées avec une racine carrée.



Ces dessins montrent les relations monotones et les non-relations qui existent entre les différents types de distance. L'indice de Soerensen D8 quantitatif est strictement identique à l'indice de Bray-Curtis ou de Steinhaus D14.

Pour les abondances d'espèces, on utilise souvent l'indice D14 (ou la racine carrée de D14 qui a des propriétés métriques).



En pratique, l'indice de Jaccard quantitatif lui est très fort corrélé (ici, $r = 0.999$) et il a des propriétés métriques. La formule utilisée \Rightarrow Jaccard index is computed as $2B/(1+B)$, avec B = distance de Bray–Curtis.

TP : Q3

Montrer les relations ou les différences entre les indices de distance euclidienne, de Bray-Curtis, de Soerensen (quantitatif et binaire), de Jaccard (quantitatif et binaire) sur des données brutes du fichier « carabides_32sta_103esp.txt ».

3. Méthodes de groupement hiérarchique

3.1. Dendrogramme

R propose la fonction **hclust()** pour réaliser différents groupements hiérarchiques agglomératifs.

cluster <- hclust(matrice_de_distance, method = "")

avec method =

- **"single"** = méthode des liens simples
- **"complete"** = méthode des liens complets
- **"average"** = groupement selon l'association moyenne (= UPGMA)
- **"mcquitty"** = groupement à poids proportionnels (= WPGMA)
- **"median"** = groupement médian (= WPGMC)

- **"centroid"** = groupement centroïde (= UPGMC)
- **"ward"** = deux méthodes de calculer ward.D ou ward.D2 (voir l'aide de hclust()).

Le résultat est un fichier de valeurs qui peut être utilisé pour construire le groupement avec la fonction générique **plot()** :

plot(cluster, cex=0.7, main = "Complete sur D14", xlab = "", hang=-1,)

avec :

- **cex** = taille des labels
- **main** = titre du dessin
- **xlab, ylab** = label des axes x- et y-
- **hang = -1** pour étirer les aligner les branches du cluster jusqu'en bas

Pour visualiser sur le dendrogramme un niveau de groupement particulier, on utilise la fonction **rect.hclust()** :

rect.hclust(cluster, nbclusters) avec nbclusters = 2, 3, ...

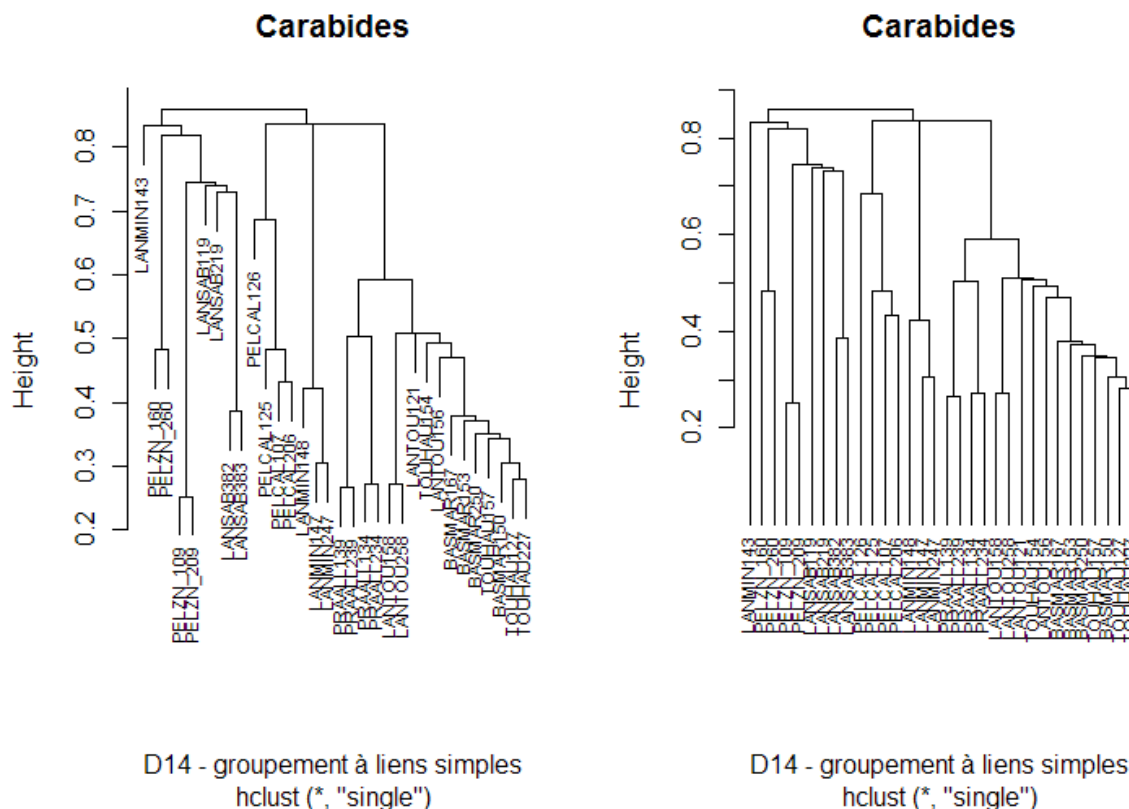
Pour stocker l'appartenance des objets aux groupes d'un niveau de groupement (pour comparer différents groupements ou faire des graphiques d'ordination (voir cours suivant)), on utilise la fonction **cutree()** :

cluster.nbclusterssgr <- cutree(cluster, nbclusters) avec nbclusters = 2, 3, ...

Exemple :

```
# Choix d'une matrice de distance (d14)
d=d14
par(mfrow = c(1, 2))                                # plot sur deux panneaux

# Méthode à liens simples
clust.single <- hclust(d, method = "single")
clust.single                                           # pour voir le contenu du fichier produit
plot(clust.single,
      cex=0.7,                                         # pour modifier la taille des labels
      main = "Carabides",                             # titre principal
      xlab = "D14 - groupement à liens simples",       # label axe des X
)
plot(clust.single,
      cex=0.7,                                         # pour modifier la taille des labels
      main = "Carabides",                             # titre principal
      xlab = "D14 - groupement à liens simples",       # label axe des X
      hang = -1,                                       # étire les branches jusqu'en bas
)
```



Voir http://wiki.math.yorku.ca/index.php/R: Cluster_analysis pour découvrir la grande diversité des modules développés sous R pour faire du clustering.

Voir aussi <http://gastonsanchez.com/blog/how-to/2012/10/03/Dendrograms.html>

D'autres fonctions ont été développées pour augmenter l'intérêt graphique des dendrogrammes mais elles n'ont pas encore été explorées. Voir par exemple les packages ClassDiscovery, mptree, ...

3.2. Comparer des dendrogrammes

On peut comparer les dendrogrammes en les juxtaposant :

```
par(mfrow = c(1, 2))                                # plot sur deux panneaux

clust.average <- hclust(d14, method = "average")
clust.ward <- hclust(d14, method = "ward")

plot(clust.average,
      cex=0.7,                                         # pour modifier la taille des labels
      main = " D14 - groupement UPGMA ", # titre principal
      hang = -1,                                       # étire les branches jusqu'en bas
      )
rect.hclust(clust.average, 6)                         # révéler 6 groupes

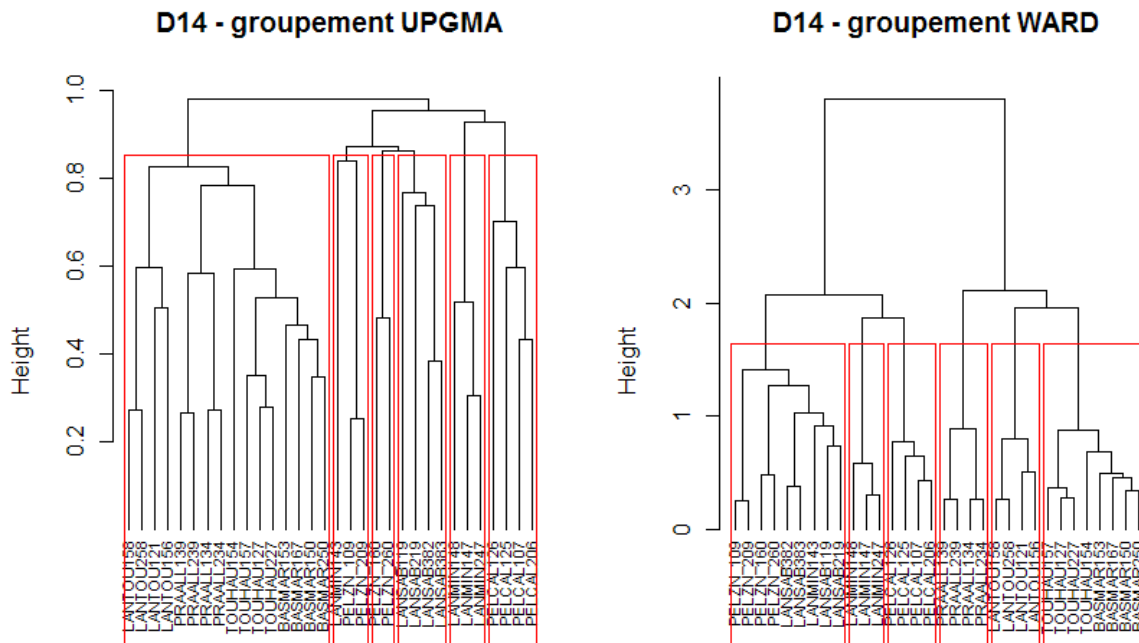
plot(clust.ward,
      cex=0.7,                                         # pour modifier la taille des labels
```

```

main = " D14 - groupement WARD ", # titre principal
hang = -1,                         #étire les branches jusqu'en bas
)
rect.hclust(clust.ward, 8)         # révéler 8 groupes

```

La comparaison des deux dendrogrammes montre que la méthode de Ward produit des groupements bien dissociés. La fonction **rect.hclust()** permet de visualiser un nombre défini de groupes.



Il est toutefois difficile de bien les comparer car des objets qui sont contigus peuvent appartenir à des groupes différents. On voit bien qu'à partir de 8 groupes, on commence à avoir des structures hétérogènes dans certains groupes.

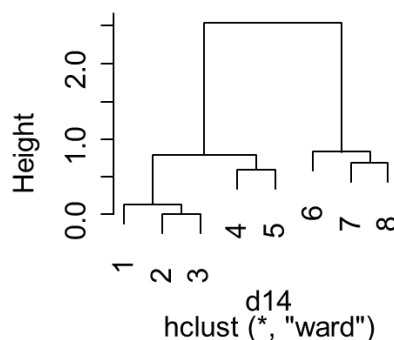
Pour n'avoir qu'une partie du dendrogramme, on peut couper le graphique avec la fonction **clip.clust()** de la library maptree:

```

library(maptree)
clust.ward.dendro8 <- clip.clust(clust.ward, data ,k=8)
plot(clust.ward.dendro8, main=paste("Dendrogramme coupé à k=8"))

```

Dendrogramme coupé à k=8



Attention la numérotation ne correspond pas nécessairement à celle obtenue avec la fonction `cutree()`, voir plus loin ! Elle est simplement ordonnée 1, 2, 3, ... de la gauche à la droite.

Voir aussi la fonction `cluster.stats()` de la library(fpc)

3.3. Récupérer les classifications

On peut récupérer la classification dans un vecteur avec la fonction **`cutree()`** mais **attention**, la numérotation des groupes n'est pas logiquement attendue avec des nombres croissants de droite à gauche.

```
par(mfrow = c(1, 2))           # plot sur deux panneaux

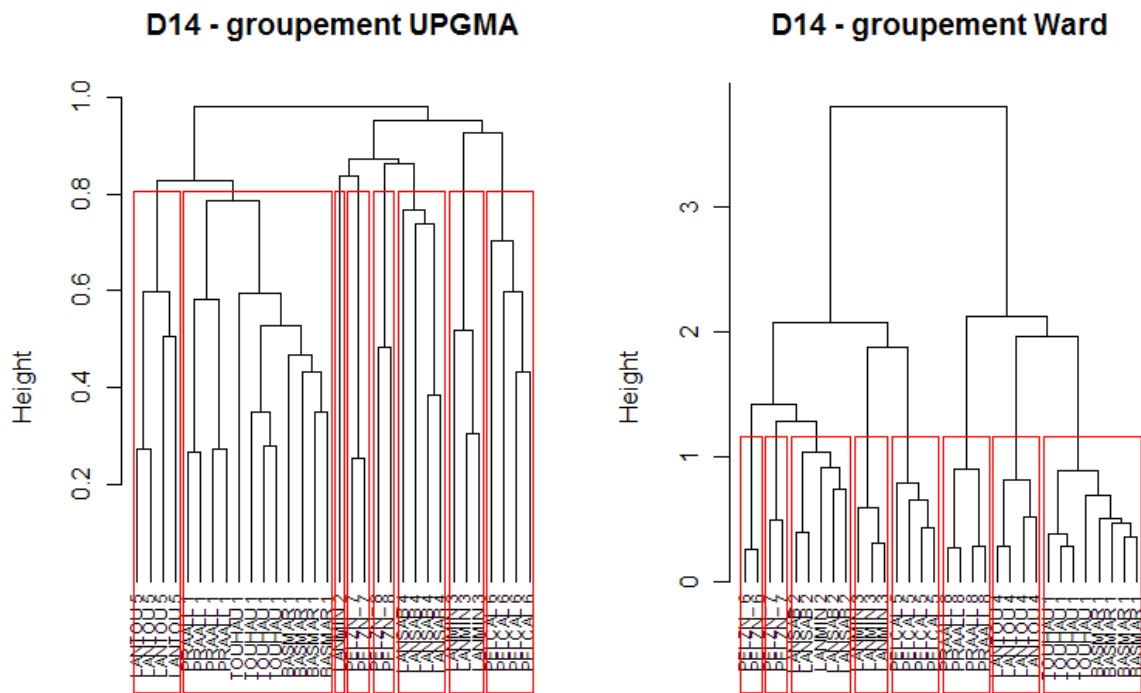
# recuperation des 8 groupes
clust.average.8gr <- cutree(clust.average, 8)
clust.ward.8gr <- cutree(clust.ward, 8)

# preparation d'un label combinant le code du groupe et le biotope
# utilisation de la fonction paste() pour concatener deux chaînes de caractères
clust.average.8gr.label <- paste(substr(rownames(data),1,6), clust.average.8gr);
clust.ward.8gr.label    <- paste(substr(rownames(data),1,6),clust.ward.8gr);

# Dendrogramme average + 8 groupes
plot(clust.average,
     cex=0.7,                # pour modifier la taille des labels
     main = " D14 - groupement UPGMA ",
     hang = -1,              # étire les branches jusqu'en bas
     labels = clust.average.8gr.label,
)
rect.hclust(clust.average, 8)  # révéler 8 groupes

# Dendrogramme ward + 8 groupes
plot(clust.ward,
     cex=0.7,                # pour modifier la taille des labels
     main = " D14 - groupement Ward ",
     hang = -1,              # étire les branches jusqu'en bas
     labels = clust.ward.8gr.label,
)
rect.hclust(clust.ward, 8)    # révéler 8 groupes
```

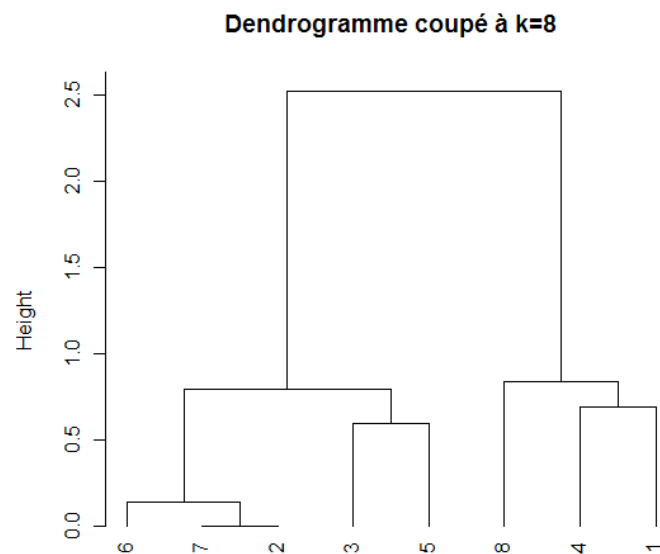
Les labels permettent alors de retrouver sur le dendrogramme à quel groupe au niveau 8 chaque objet appartient. La séquence lisible sur le dendrogramme de Ward est : **6, 7, 2, 3, 5, 8, 4 et 1.**



Ward récupère assez rapidement la structure initiale alors que le groupe 1 de l'UPGMA rassemble encore 3 biotopes.

On peut aussi simplifier la représentation du groupement mais on doit faire attention à la numérotation des groupes en la contrôlant. On peut d'ailleurs y mettre des chaînes de caractères pour résumer la structure du groupement.

```
library(maptree)
clust.ward.dendro8 <- clip.clust(clust.ward, data, k=8);
label8gr = c(6,7,2,3,5,8,4,1) # on introduit ici la sequence lue sur le dendrogramme;
plot(clust.ward.dendro8,
     hang = -1,
     main= paste("Dendrogramme coupé à k=8"),
     labels = label8gr,
)
```



Ce dendrogramme représente correctement la numérotation des groupes récupérées avec la fonction **cutree()**.

3.4. Comparer des classifications

On peut comparer les typologies pour identifier les groupes similaires et donc les formes fortes ou des structures récurrentes avec la fonction **table()** :

```
# Comparaison average et ward
table(clust.average.8gr, clust.ward.8gr)
```

		clust.ward.8gr							
clust.average.8gr		1	2	3	4	5	6	7	8
1	8	0	0	0	0	0	0	0	4
2	0	1	0	0	0	0	0	0	0
3	0	0	3	0	0	0	0	0	0
4	0	4	0	0	0	0	0	0	0
5	0	0	0	4	0	0	0	0	0
6	0	0	0	0	4	0	0	0	0
7	0	0	0	0	0	2	0	0	0
8	0	0	0	0	0	0	2	0	0

Les groupements sont forts similaires (5 groupes en commun, presque 6), avec un gros groupe de 12 objets pour le cluster.average.8gr.

On peut récupérer l'ensemble de la typologie dans une matrice et les exporter en format text tabulé avec la fonction **write.table()** :

```
# Récupération de 8 groupes dans un vecteur
cl1a8 <- cutree(clust.ward, k = c(1,2,3,4,5,6,7,8))
# Exportation des données du groupement
write.table(cl1a8,'clus_ward_8gr.txt', sep="\t")
```

3.5. Distance et corrélation cophénétique

La fonction **cophenetic()** permet de construire une matrice de distance qui est basée sur le dendrogramme pour la comparer à la distance originale. Pour rappel, les différentes méthodes existent car elles ont des règles différentes d'association d'un objet à un groupe.

On en profite pour récupérer dans le titre des plots la corrélation cophénétique avec la fonction **cor()** qu'on arrondi à 3 décimales avec la fonction **round()**.

```
par(mfrow = c(3, 1)) # plot sur 3 panneaux

# méthode à liens simples
clust.single <- hclust(d14, method = "single")
titre = paste("Liens simples : cor. cophénétique = ", round(cor(d14, cophenetic(clust.single)), digits = 3))
plot(d14, cophenetic(clust.single), col="red", main = titre)
abline(0, 1) # afficher la ligne (1,1)

# méthode UPGMA
```

```

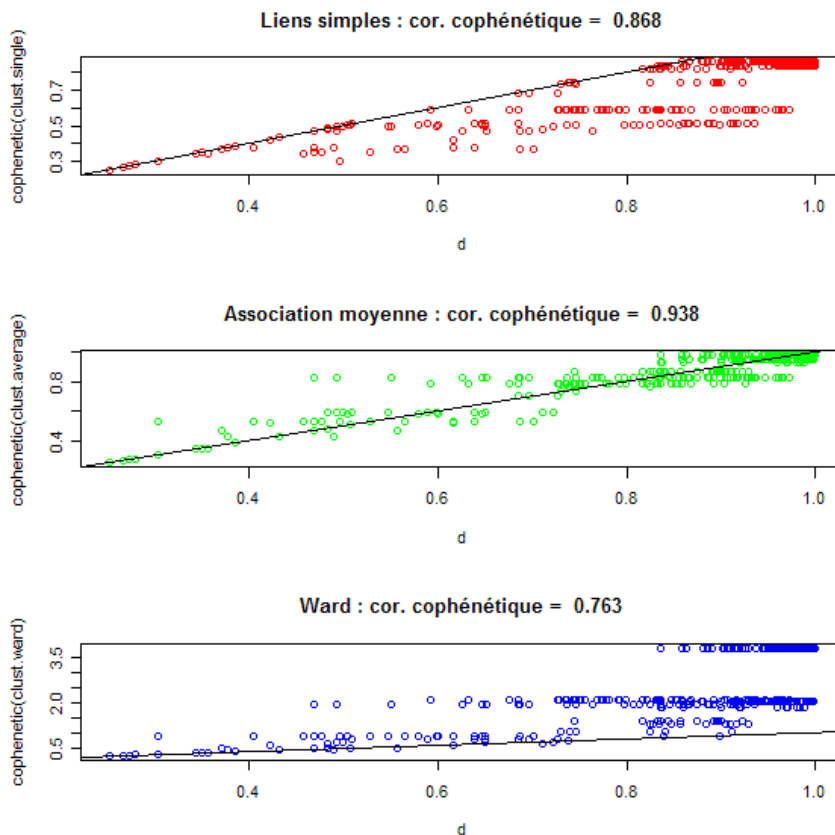
clust.average <- hclust(d14, method = "average")
titre = paste("Association moyenne : cor. cophénétique = ", round(cor(d14,
cophenetic(clust.average)), digits = 3))
plot(d14, cophenetic(clust.average), col = "green", main = titre)
abline(0, 1)

# méthode de ward
clust.ward <- hclust(d14, method = "ward")
titre = paste("Ward : cor. cophénétique = ", round(cor(d14, cophenetic(clust.ward)), digits = 3))
plot(d14, cophenetic(clust.ward), col = "blue", main = titre)
abline(0, 1)

```

Le résultat révèle que la meilleure corrélation est celle par association moyenne, ce qui est logique vu sa manière de travailler. Les liens simples tendent logiquement à sous-estimer les distances réelles alors que Ward tend à les sur-estimer une fois que les groupes les plus évidents sont formés. Cela explique les groupes bien marqués sur le dendrogramme.

Ces graphiques révèlent aussi que plus on avance dans le groupement (on commence par regrouper les objets les plus similaires), plus il impose des différences avec la réalité mesurée dans la matrice de distance. Cela signifie que les derniers niveaux auxquels on s'arrête habituellement, sont en fait les structures les moins fidèles à la matrice de distance.



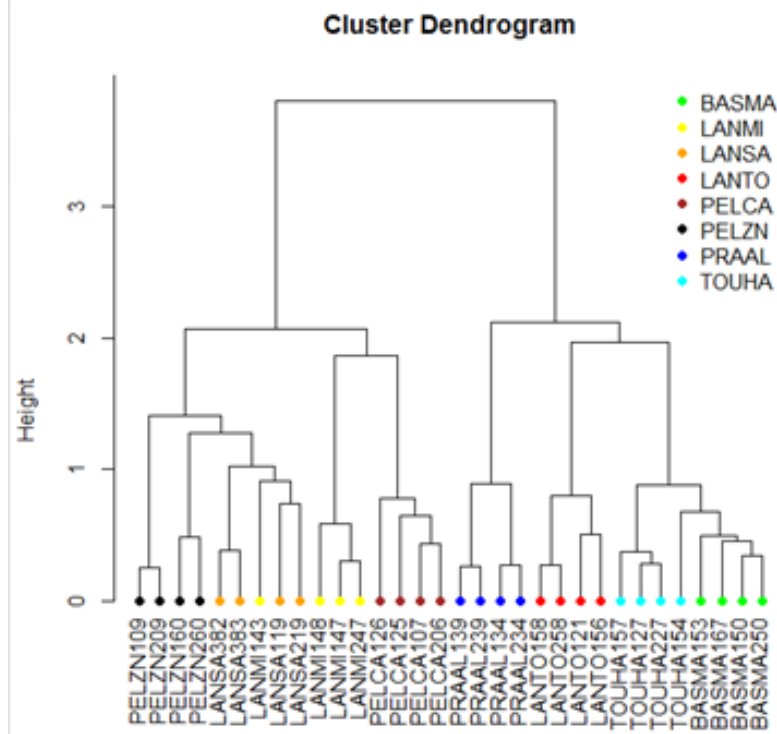
Comparer différentes approches de combinaison de calcul de distances (minimum 2 distances) et de méthodes de groupement (minimum deux distances) sur la base du fichier de données brutes du fichier « Demazy_2013_carabides.txt » et montrer les différences.

Analyser les résultats et proposer les arguments pour défendre le résultat qui vous semble le plus convaincant.

3.6. Ajouter des symboles sur un graphique

Il est possible d'ajouter des symboles de forme et de taille différente pour représenter une classification a priori sur le résultat d'un cluster.

```
par(mfrow = c(1, 1))
biotope <- paste(substr(rownames(data),1,5))
colvec <- c("green", "yellow", "orange", "red", "brown", "black", "blue", "cyan")
names(colvec) = c("BASMA", "LANMI", "LANSA", "LANTO", "PELCA", "PELZN", "PRAAL", "TOUHA")
plot(clust.ward, hang = -1)
col = colvec[biotope[clust.ward$order]]
symbols(1:nrow(data), rep(0, nrow(data)), circles=rep(1, nrow(data)), add=TRUE, inches=.05,bg=col,
fg=col, xpd=TRUE)
legend("topright", legend = levels(factor(biotope)), bty = "n", col = colvec, pch = 21, pt.bg = colvec)
```



4. Groupement non-hiérarchique *k*-means

R propose la fonction **kmeans()** pour réaliser une partition non-hiérarchique qui maximise la variance inter-groupe. Elle fonctionne de manière simple puisqu'il faut lui donner un dataframe et spécifier le nombre *k* de groupes. Comme elle se base sur la variance, les données doivent respecter certaines règles pour que la typologie ne dépende pas des unités des variables ou de trop fortes différences de variables. Vu le nombre de zéros, les données d'abondance d'espèces ne sont donc pas adaptées à cette procédure et des données quantitatives écologiques doivent en principe être standardisées pour donner la même chance à toutes les variables de participer au groupement. Mais cela dépend des situations.

4.1. Classification directe sur une matrice brute

On commence par standardiser le fichier de données puis on applique un groupement hiérarchique de Ward pour le comparer au résultat de *k*-means.

```
# Initiation et remise à zéro des données précédentes
rm(list=ls())                                     # on élimine les données stockées

# Lecture des données
eco= read.table("carabides_32sta_12eco.txt", h=T, sep="\t", row.names="Stations")
attach(eco)

# Transformation de certaines variables (cfr Q2)
# Humidité et Altitude sont OK ;
# pHeau et pHKCL => racine()
sqrt_pHeau=sqrt(pHeau) ;
sqrt_pHKCL=sqrt(pHKCL) ;
# Ca, K, Mg, Na, P => log base 2 ou népérien
Log_Ca =log(Ca+1, base=2);
Log_K =log(K+1, base=2);
Log_Mg =log(Mg+1, base=2);
Log_Na =log(Na +1, base=2);
Log_P =log(P+1, base=2);

# Reconstruction d'un dataframe
eco_transforme = data.frame(Humidite, Altitude, sqrt_pHeau, sqrt_pHKCL,
  Log_Ca, Log_K, Log_Mg, Log_Na, Log_P, row.names=row.names(eco)) ;

# Standardisation de la matrice transformée (moyenne = 0 et std = 1)
library(vegan)
eco_std = decostand(data.frame(Altitude, Humidite, Log_Ca, Log_K, Log_Mg, Log_Na, Log_P,
  sqrt_pHeau, sqrt_pHKCL, row.names = row.names(eco)), "standardize")

# Calcul de distance euclidienne sur une partie des données
```

```
eco.d1 <- vegdist(eco_std, method="euclidean")

# Calcul d'un groupement de ward
eco.clust.ward <- hclust(eco.d1, method = "ward")
eco.clust.ward.8gr <- cutree(eco.clust.ward, 8)

# S k-means
eco.kmeans <- kmeans(eco_std,
  centers=8,           # nombre de groupes k
  iter.max=5000,       # nombre d'iterations maximum possible pour tester une
  configuration de départ
  nstart=1000,         # nombre d'essais avec des situations de départ différentes
)
eco.kmeans
```

Ce code génère un rapport de kmeans qui donne notamment le résultat du groupement qui est stocké dans la variable **eco.kmeans\$cluster** :

```
Clustering vector:
BASMA150 BASMA153 BASMA167 BASMA250 LANMI143 LANMI147 LANMI148 LANMI247
      5      2      5      5      3      3      3      3
LANSA119 LANSA219 LANSA382 LANSA383 LANTO121 LANTO156 LANTO158 LANTO258
      6      6      3      3      6      1      1      1
PELCA107 PELCA125 PELCA126 PELCA206 PELZN109 PELZN160 PELZN209 PELZN260
      8      8      8      8      4      7      4      7

PRAAL134 PRAAL139 PRAAL234 PRAAL239 TOUHA127 TOUHA154 TOUHA157 TOUHA227
      2      2      2      2      1      1      1      1
```

Comparaison des résultats des deux groupements

```
table(eco.kmeans$cluster, eco.clust.ward.8gr)
```

```
eco.clust.ward.8gr
  1 2 3 4 5 6 7 8
1 0 0 0 7 0 0 0 0
2 1 0 0 0 0 0 0 4
3 0 6 0 0 0 0 0 0
4 0 0 0 0 0 2 0 0
5 3 0 0 0 0 0 0 0
6 0 0 3 0 0 0 0 0
7 0 0 0 0 0 0 2 0
8 0 0 0 0 4 0 0 0
```

Les deux solutions proposées sont très similaires et elles ne se différencient que par un objet. Vous pouvez vérifier la sensibilité de k-means au nombre d'essais de configuration de départ. La valeur défaut de 10 produit ici des résultats bien différents.

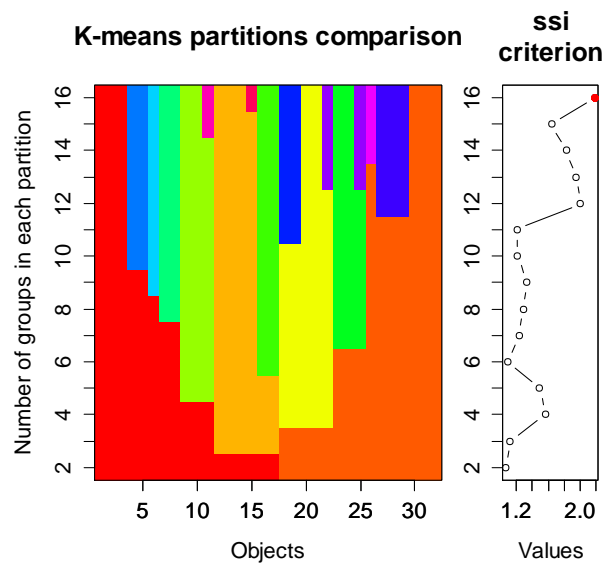
4.2. Variation sur kmeans

Afin d'éviter de faire des k-means en cascade, on peut utiliser la fonction `cascadeKM()` de `vegan` qui permet de créer différentes partitions allant de $k=2$ à $k=x$. Cela permet de

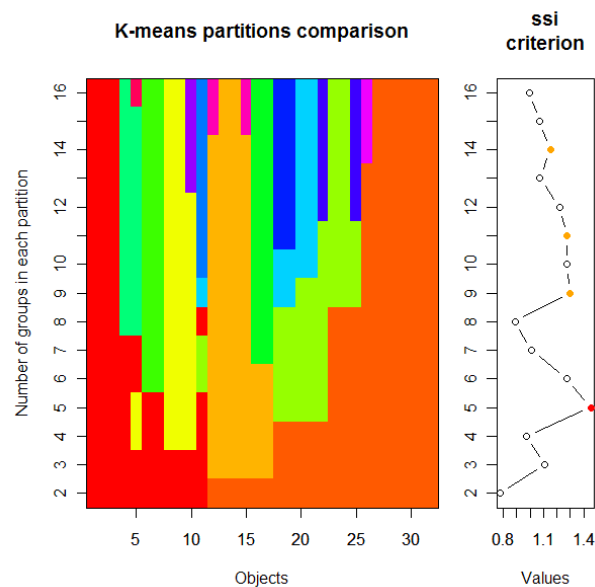
visualiser notamment l'évolution de différents indicateurs qui nous permettraient d'identifier le nombre optimal de groupes.

```
eco.kmeans.2to16 <- cascadeKM(eco_std,
  inf.gr=2,      # nombre de groupes de départ
  sup.gr=16,     # nombre de groupes final
  iter =200,     # nombre d'essais avec des situations de départ différentes
  criterion = "ssi" # critère d'évaluation
)
plot(eco.kmeans.2to16, sortg=TRUE)
```

Ce graphique permet de suivre l'affectation des stations (abscisse) dans les différents niveaux de groupement.



Un autre exemple permet de comprendre la signification du critère ssi.





























Le graphique à droite (ssi) est censé aider au choix d'un niveau de partition. Le niveau maximal = point rouge et d'autres niveaux intéressants = ceux qui révèlent une remontée après une diminution (point orange). Les niveaux 5 et 9 sont à explorer ...

Par ailleurs, on peut aussi utiliser k-means pour reproduire une logique de groupement hiérarchique divisif en commençant par deux groupes, puis en continuant ainsi sur chacun des deux lots séparés. Cela n'a toutefois de sens que lorsque le nombre de réallocations semble limité.

* * *

4.2.1. R Plot PCH Symbols Chart

Following is a chart of PCH symbols used in R plot. When the PCH is 21-25, the parameter "col=" and "bg=" should be specified. PCH can also be in characters, such as "#", "%", "A", "a", and the character will be plotted.

0: 	10: 	20: 	A: A
1: 	11: 	21: 	a: a
2: 	12: 	22: 	B: B
3: 	13: 	23: 	b: b
4: 	14: 	24: 	S: S
5: 	15: 	25: 	`: `
6: 	16: 	@: @	.: .
7: 	17: 	+: +	,: ,
8: 	18: 	?: %	?: ?
9: 	19: 	#: #	*: *

From <http://www.endmemo.com/program/R/pchsymbols.php>