

# BING-F4002 Acquisition et analyse de données

## Fiche TP 3 : Ordinations

---

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>.

## 1. Réaliser une ACP

### 1.1. vegan

Vegan propose la fonction **rda()** pour réaliser une ACP. La même fonction réalise aussi des analyses de redondance (RDA en anglais) quand on veut analyser les relations linéaires entre deux fichiers de données. Ici, si on lui propose un seul fichier, il réalise une ACP.

## ResultatACP = rda(donnees, scale=TRUE ou FALSE)

Seule l'option **scale** peut être utile :

- **FALSE** : réalise une analyse sur la matrice de covariance en conservant les variances des données
- **TRUE** : réalise une analyse sur la matrice de corrélation (par défaut)

Exemple des codes vu au cours :

### # ACP COURS

#### # Lecture de données

```
# ecobrut=read.table('carabides_32sta_12 eco.txt', h=T, sep="\t", row.names="Stations")
attach(ecobrut)
```

#### # Transformation de certaines variables

```
# Humidite est 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 = data.frame(Humidite, sqrt_pHeau, sqrt_pHKCL,
Log_Ca, Log_K, Log_Mg, Log_Na, Log_P, row.names=row.names(ecobrut)) ;
```

#### # Tableau des relations

```
pairs(eco)
```

#### # Matrice de corrélation

```
correlation = cor(eco)
edit(correlation)
```

#### # Appel de l'ACP

```
library(vegan)
eco.pca <- rda(eco, scale=TRUE) # scale=true => matrice de corrélation;
summary(eco.pca)
```

#### # Graphique amélioré

```
plot(eco.pca)
coordo <- scores(eco.pca)
coordovar = data.frame(coordo$species)
arrows(0,0, coordovar$PC1, coordovar$PC2, col='red')
coordoobj = data.frame(coordo$sites)
points(coordoobj$PC1, coordoobj$PC2, col='blue')
```

#### # Cercle des corrélations

```
coordo <- scores(eco.pca)
```

```

correlation = data.frame(cor(eco, coordo$sites))
a <- seq(0,2*pi,length=100)
plot( cos(a), sin(a),
      type = 'l', lty = 3,
      xlab = 'comp 1', ylab = 'comp 2',
      main = "Correlation circle")
arrows(0,0, correlation$PC1, correlation$PC2, col='red')
text(correlation$PC1, correlation$PC2,rownames(correlation), col='blue')

```

### **# Capacité d'explications pour les variables**

```

COS2Var = goodness(eco.pca, display = "species") * 100
COS2Var

```

### **# Contributions relatives des variables aux axes**

```

coordo <- scores(eco.pca)
correlation = data.frame(cor(eco, coordo$sites))
contribut = correlation*correlation
CTRVar1 = contribut$PC1 / sum(contribut$PC1) *100
CTRVar2 = contribut$PC2 / sum(contribut$PC2) *100
CTRVar = data.frame(CTRVar1, CTRVar2, row.names=rownames(contribut))
CTRVar

```

### **# Capacité d'explications pour les objets**

```

COS2Obj = goodness(eco.pca, display = "sites") * 100
COS2Obj

```

### **# Contributions relatives des objets aux axes**

```

Cos2= data.frame(goodness(eco.pca, display = "sites"))
Inertie = data.frame(inertcomp(eco.pca, display = "sites"))
Prd.PC1 = Inertie$CA*Cos2$PC1
CTRObj1 = round(Prd.PC1/sum(Prd.PC1)*100, digits = 2)
Prd.PC2 = Inertie$CA*Cos2$PC2
CTRObj2 = round(Prd.PC2/sum(Prd.PC2)*100, digits = 2)
CTRObj = data.frame(CTRObj1, CTRObj2, row.names=rownames(Cos2))

```

### **# Utilisation d'une routine toute faite**

```

ACP_resultat <-function(pca)
{
  # === EXTRAIT DE LEGENDRE 2008 ===
  # Valeurs propres
  ev = pca$CA$eig
  # Quelles sont les valeurs propres plus grandes que la moyenne?
  ev[ev > mean(ev)]
  # Modele du baton brise (broken stick model)
  n = length(ev)
  bsm = data.frame(j=seq(1:n), p=0)
  bsm$p[1] = 1/n

```

```

for (i in 2:n) {
  bsm$p[i] = bsm$p[i-1] + (1/(n + 1 - i))
}
bsm$p = 100*bsm$p/n
bsm
# Dessiner les valeurs propres et le % de variance de chaque axe
par(mfrow=c(2,1))
barplot(ev, main="Valeurs propres", col="bisque", las=2)
abline(h=mean(ev), col="red")      # moyenne des valeurs propres
legend("topright", "Moyenne des valeurs propres", lwd=1, col=2, bty="n")
barplot(t(cbind(100*ev/sum(ev), bsm$p[n:1])), beside=T,
        main="% variance", col=c("bisque",2), las=2)
legend("topright", c("% valeur propre", "Broken stick model"),
      pch=15, col=c("bisque",2), bty="n")

# Dessin des variables environnementales pour les axes 1 et 2
# *****

# Scores des sites et "species" (ici les variables environnementales)
# Defaut: cadrage 2
scores(pca)
scores(pca, choices=1:2)
sit.sc2 = scores(pca, display="wa")
spe.sc2 = scores(pca, display="sp")
# Cadrage 1
sit.sc1 = scores(pca, display="wa", scaling=1)
spe.sc1 = scores(pca, display="sp", scaling=1)

# Dessin des variables seulement (fleches!)
windows()
# Mac: quartz()
plot(pca, display="sp", type="n",
     main="ACP correlation - Variables environnementales ")
text(pca, display="sp", cex=1, pos=4, col="red")
arrows(0, 0, spe.sc2[,1], spe.sc2[,2], length=0.07, angle=20, col="red")

# 1c. Biplots (cadrages 1 et 2)
# *****

windows(12,6)
par(mfrow=c(1,2))
# Cadrage 1: "distance biplot"
plot(pca, scaling=1, main="Biplot PCA distance - cadrage 1 - D1 conservées")
arrows(0, 0, spe.sc1[,1], spe.sc1[,2], length=0.07, angle=20, col="red")
# Cadrage 2 (default): "correlation biplot"
plot(pca, main="Biplot PCA correlation - cadrage 2 - corrélations conservées")
arrows(0, 0, spe.sc2[,1], spe.sc2[,2], length=0.07, angle=20, col="red")

```

```

# On peut utiliser 2 differents cadrages des vecteurs propres a l'aide de
# l'argument scaling :
# scaling = 1 : preserve les distances euclidiennes entre les objets
# scaling = 2 : preserve les correlations entre les descripteurs (ou variables)
# Les cadrages de vegan ne correspondent pas tout a fait a ceux de la theorie.
# Jari Oksanen leur fait subir quelques ajustements un peu mysterieux...
# =====
}

ACP_resultat(eco.pca);
# Une bonne combinaison : cadrage 1 pour les objets + Cercle des corrélations

# Graphique amélioré
plot(eco.pca, type = "n") # plot vide
coordo <- scores(eco.pca, scaling=1)
coordovar = data.frame(coordo$species)
coordobj = data.frame(coordo$sites)

#rangepc1 = (max(coordobj$PC1) - min(coordobj$PC1)) * 0.5
#rangepc2 = (max(coordobj$PC2) - min(coordobj$PC2)) * 0.5

coordobj$pos <- ifelse(coordobj$PC1 < 0 , 2, 4)

points(coordobj$PC1, coordobj$PC2,
       col='blue',
       pch = 16)

text(coordobj$PC1, coordobj$PC2,
     labels = rownames(eco),      # label de stations sur le graphique
     col = 'chartreuse4',         # couleur du label (peut être un vecteur)
     pos = coordobj$pos,          # position du label
     cex = 0.6)                   # taille du label

# Values of 1, 2, 3 and 4, respectively indicate positions below, to the left of, above and to
the right of the specified coordinates.

```

---

#### TP : Q1

Réalisez une ACP sur le fichier **Xylobios\_ecologie.txt** en ne prenant pas en compte les variables X, Y, Sampl\_structure, Alt, Region et Cov\_flr ?

Comment interpréter-vous les résultats et que décidez-vous de faire ?

Si le troisième caractère du nom des stations indique qu'il s'agit d'une station « Riche » ou « Pauvre » en bois mort, comment pouvez-vous les révéler sur le graphique ?

---

## 2. Réaliser une AFC

### 2.1. vegan

Comme pour l'acp, Vegan propose la fonction `cca()` pour réaliser une AFC. La même fonction réalise aussi des analyses canoniques des correspondances (CCA en anglais) quand on veut analyser des relations non-linéaires entre deux fichiers de données. Ici, si on lui propose un seul fichier, il réalise une AFC.

**ResultatAFC = cca(donnees)**

Pas d'option particulièrement utile. Si besoin chercher R Vegan CCA sur le net.

Exemple des codes vu au cours :

**# AFC COURS**

**# Lecture de données**

```
# esp=read.table('carabides_32sta_103esp.txt', h=T, sep="\t", row.names="Stations")
```

```
esp.ca <- cca(esp)
```

```
summary(esp.ca)
```

**# Graphique amélioré**

```
plot(esp.ca, type = "n")
```

*# plot vide*

```
# plot(esp.ca, type = "n", xlim=c(-2, 2), ylim=c(-2, 2)) # plot vide avec des axes limités
```

```
coordolig <- as.data.frame(scores(esp.ca, scaling=1, display="wa"))
```

```
coordocol <- as.data.frame(scores(esp.ca, scaling=1, display="spe"))
```

**# coordonnées des colonnes**

```
points(coordocol$CA1, coordocol$CA2,  
       col='darkred',  
       pch = 16)
```

```
coordocol$pos <- ifelse(coordocol$CA1 < 0, 2, 4)
```

```
text(coordocol$CA1, coordocol$CA2,
```

```
      labels = rownames(coordocol), # label de stations sur le graphique
```

```
      col = 'darkred', # couleur du label (peut être un vecteur)
```

```
      pos = coordocol$pos, # position du label
```

```
      cex = 0.6) # taille du label
```

**# coordonnées des lignes**

```
points(coordolig$CA1, coordolig$CA2,
```

```
       col='darkcyan',
```

```
       pch = 1)
```

```
text(coordolig$CA1, coordolig$CA2,
```

```
      labels = rownames(coordolig), # label de stations sur le graphique
```

```

col = 'darkcyan',          # couleur du label (peut être un vecteur)
pos = 3,                  # position du label
cex = 0.6)                # taille du label

# Qualité des représentations des espèces
Cos2= data.frame(goodness(esp.ca, display = "species"))
Inertie = data.frame(inertcomp(esp.ca, display = "species"))
#sum(Inertie.spe) = variance du jeu de données
species = rownames(Cos2)
Prd.CA1 = Inertie$CA*Cos2$CA1
CTR.CA1 = round(Prd.CA1/sum(Prd.CA1)*100, digits = 2)
Prd.CA2 = Inertie$CA*Cos2$CA2
CTR.CA2 = round(Prd.CA2/sum(Prd.CA2)*100, digits = 2)
#Prd.CA3 = Inertie$CA*Cos2$CA3          # si on veut un 3eme axe
#CTR.CA3 = round(Prd.CA3/sum(Prd.CA3)*100, digits = 2)
Cos2$CA1 = round(Cos2$CA1*100, digits = 2)
Cos2$CA2 = round(Cos2$CA2*100, digits = 2)
#Cos2$CA3 = round(Cos2$CA3*100, digits = 2)
#Qual = data.frame(species, CTR.CA1, CTR.CA2, CTR.CA3, Cos2$CA1, Cos2$CA2,
Cos2$CA3 )
Qual = data.frame(species, CTR.CA1, CTR.CA2, Cos2$CA1, Cos2$CA2)
Qual

```

---

TP : Q2

Réalisez une AFC sur le fichier **Demazy\_2013\_carabides.txt**.

Comment interpréter-vous les résultats et que décidez-vous de faire ?

Pouvez-vous ajouter le résultat d'un groupement sur ce graphique ?

---

### 3. Réaliser une PCoA ou MDS

#### 3.1. vegan

On utilise la fonction **cmdscale()** pour réaliser une PCoA = MDS. Attention, cette analyse se réalise sur une matrice de distance.

**ResultatPCOA = cmdscale(matricededistance, add= TRUE, eig= TRUE)**

Avec :

- **matricededistance** = matrice de distance calculée par exemple avec `vegdist()`
- **add = TRUE** pour éviter des axes avec des variances négatives et
- **eig = TRUE** pour que les valeurs propres soient disponibles

Exemple des codes vu au cours :

### # PCOA COURS

#### # Lecture de données

```
# esp=read.table('carabides_32sta_103esp.txt', h=T, sep="\t", row.names="Stations")
```

#### # Matrice de distance

```
d14=vegdist(esp, method="bray")
```

#### # PCoA

```
pcoa <- cmdscale(d14, add= TRUE, eig= TRUE)
```

#### # Calcul des valeurs propres

```
valeurs_propres = round(pcoa$eig/sum(pcoa$eig)*100, digits = 2)
```

```
barplot(valeurs_propres, col = "chocolate")
```

#### # graphique de base

```
ordiplot(pcoa)
```

#### # graphique amélioré

```
figure <- ordiplot(pcoa, type="points")
```

```
abline(h=0, lty=3) # axe horizontal (lty = type de la ligne)
```

```
abline(v=0, lty=3) # axe vertical (lty = type de la ligne)
```

```
text(figure, "sites", labels= rownames(esp),col="red", cex=0.9, pos=3)
```

#### # recuperation des coordonnées dans un dataframe

```
pcoa1 = pcoa$points[,1]
```

```
pcoa2 = pcoa$points[,2]
```

```
coordo = data.frame(pcoa1, pcoa2, row.names=rownames(esp))
```

#### # graphique avec les coordonnées pondérées des espèces

```
ordiplot(pcoa, type="points")
```

```
abline(h=0, lty=3); abline(v=0, lty=3)
```

```
text(figure, "sites", labels= rownames(esp),col="red", cex=0.9, pos=3)
```

```
pcoa.wa <- wascores(coordo, esp)
```

```
points(pcoa.wa, col='darkcyan',pch = 1)
```

```
text(pcoa.wa,rownames(pcoa.wa),cex=0.7, col="blue")
```

#### # symboles pour des groupes de stations

```
cluster <- hclust(d14, method = "ward")
```

```
plot(cluster)
```

```
cluster.8gr <- cutree(cluster, 8)
```

#### # ellipse de dispersion à 80%

```
ordiplot(pcoa)
```

```
abline(h=0, lty=3); abline(v=0, lty=3)
```

```
ordiellipse(pcoa,cluster.8gr, conf = 0.8)
```



```
# pour voir le chemin du groupement
# ordiplot(pcoa)
# ordicluster(pcoa,cluster)
```

### # polygones convexes

```
ordiplot(pcoa)
abline(h=0, lty=3); abline(v=0, lty=3)
ordihull(pcoa,cluster.8gr)
```

### # toile d'araignées

```
ordiplot(pcoa)
abline(h=0, lty=3); abline(v=0, lty=3)
ordispider(pcoa,cluster.8gr)
```

### # Ordiplot plus raffiné

```
pcoa1 = pcoa$points[,1]
pcoa2 = pcoa$points[,2]
coordo = data.frame(pcoa1, pcoa2, row.names=row.names(esp))

ordiplot(pcoa, type = "n");
abline(h=0, lty=3); abline(v=0, lty=3)
couleurs <- c("steelblue", "darkred", "darkgreen", "pink", "orange", "blue", "green", "red");
groups = levels(factor(cluster.8gr));
for(i in seq_along(cluster.8gr)){
  points(coordo[factor(cluster.8gr) == groups[i],1], coordo[factor(cluster.8gr) == groups[i],2],
  col = couleurs[i], pch = 16)
}
ordispider(pcoa, factor(cluster.8gr), label = TRUE)
ordihull(pcoa, factor(cluster.8gr), lty = "dotted")
```

### # Utilisation de clusplot()

```
library(cluster);
couleur=rainbow(8) ;
clusplot(d14, # matrice de distance
  diss=TRUE, # pour indiquer que c'est une matrice de distance
  cluster.8gr, # variable de classification
  color=TRUE, # TRUE pour avoir des ellipses colorées
  shade=FALSE, # TRUE pour visualiser la densité des ellipses
  labels=4, # 4 = label des ellipses
  lines=0, # pour visualiser les distances entre les ellipses
  col.clus = couleur, # couleur des ellipses
)
```

---

TP : Q3

Réalisez une PCOA sur le fichier **Demazy\_2013\_carabides.txt**.

Comment interpréter-vous les résultats et que décidez-vous de faire ?

Quel comparaison pouvez-vous faire par rapport à celui de l'AFC ?

Pouvez-vous ajouter le résultat d'un groupement sur ce graphique ?

---

\* \* \*

## Annexes

### R Plot Color Chart






























Voir <http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.pdf>

Ou <http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf>

Ou <http://www.stats4stem.org/r-colors.html>

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

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