Travaux pratiques en langage R

Pierre Legendre
Département de sciences biologiques
Université de Montréal

Octobre 2004

1. Calcul des statistiques de base en langage R: exemple Merle

```
# Il faut d'abord indiquer à R quel est votre dossier de travail.
# Menu File => Change dir...

# Importer le fichier 'Merle' dans l'objet de type 'data frame' 'merle':
  merle <- read.table("Merle.txt")
# ou encore
  merle = read.table("Merle.txt")

# Vérification pour savoir si les données ont été lues correctement:
  merle

# Saisie des valeurs de la première colonne: la longueur de l'aile.
# La longueur est recopiée dans l'objet 'aile':
  aile=merle[,1]
# Vérification du contenu de l'objet 'aile':
  aile

# Transformation du vecteur 'aile' en un objet de type matrice 'aile.mat':
  aile.mat=as.matrix(aile)
# Vérification du contenu de la matrice 'aile.mat':
  aile.mat

# Calcul de la moyenne des longueurs d'aile (commande 'mean'):
  moyenne=mean(aile.mat)
# Vérification de la valeur de la moyenne:
  moyenne

# Calcul de la médiane des longueurs d'aile (commande 'median'):
  mediane=median(aile.mat)
# Vérification de la valeur de la médiane:
  mediane

# Calcul de la variance des longueurs d'aile (commande 'var'):
  variance=var(aile.mat)
# Vérification de la variance:
  variance

# Calcul de la taille de l'échantillon 'n':
  n=nrow(aile.mat)
# Vérification de la valeur de 'n':
  n
```

```
# Calcul du coefficient d'asymétrie A3.
#
# Calcul préparatoire: estimation non biaisée du moment d'ordre 3, k3:
  k3=(n*sum((aile.mat-mean(aile.mat))^3))/((n-1)*(n-2))
# Vérification de k3:
  k3
#
# Calcul du coefficient d'asymétrie, noté A3:
  A3=k3/((sqrt(variance))^3)
# Vérification de A3:
  A3

# Calcul du coefficient d'aplatissement A4.
#
#  Calcul préparatoire: estimation non biaisée du moment d'ordre 4, k4:
  k4=(n*(n+1)*(sum((aile.mat-mean(aile.mat))^4))-3*(n-1)*((sum((aile.mat-
mean(aile.mat))^2))^2))/((n-1)*(n-2)*(n-3))
# Vérification de k4:
  k4
#
# Calcul du coefficient d'aplatissement, noté A4:
  A4=k4/((sqrt(variance))^4)
# Vérification de A4:
  A4

# Calcul de l'étendue de la variation (ou plage de variation):
  etendue=max(aile.mat)-min(aile.mat)
# Vérification de l'étendue:
  etendue

# Calcul de l'écart type, noté Sx:
  sx=sd(aile.mat)
# Vérification de l'écart type:
  sx

# ==========

# Préparation en vue de l'histogramme.

# Calcul du nombre de classes selon la règle de Sturge:
  classes=1+(3.3*log10(n))
# Vérification du nombre de classes:
  classes

# Il faut arrondir le nombre de classes donné par la règle de Sturge
# à l'entier le plus proche. Dans le cas présent, la règle de Sturge
# donne 6,2868. J'arrondis donc à 6 classes.
# Les données vont de 72 mm à 81 mm. L'étendue de variation est de 9 mm.
# Les classes peuvent couvrir chacune 2 mm de longueur d'aile.
# Je fixe la première borne à 71 mm.
```

```
# Voici les limites de classes choisies: 71,73,75,77,79,81,83.

# Indiquer à R les 7 bornes de classes choisies pour l'histogramme:
  coupe=c(71,73,75,77,79,81,83)
# La commande 'c' a combiné les 7 valeurs en un vecteur appelé 'coupe'.

# Titre du graphique. Comme il est long, je l'ai coupé en deux parties
# pour qu'il soit affiché sur deux lignes.
  titre=c("Figure 1: Histogramme de fréquences de la longueur d'aile (mm) de
merles","d'Amérique mesurées au Jardin botanique de Montréal en août 2000.")
#
# Je recopie ensuite le titre dans un objet de type matrice:
  titremat=matrix(titre)
# Comment sont placés les deux lignes de texte dans la matrice?
  titremat

# Création de l'histogramme:
  hist(aile.mat,breaks=coupe,freq=T,right=F,xlab=NULL,ylab="Fréquences
absolues",axes=T,main=NULL)

# Positionnement du titre sur l'histogramme:
  mtext(text="Longueur de l'aile (mm)",side=1,line=2,cex=0.9,font=2)
  mtext(text=titremat[1],side=1,line=3,cex=0.7,font=2)
  mtext(text=titremat[2],side=1,line=4,cex=0.7,font=2)

# L'histogramme apparaît dans une fenêtre qu'on peut imprimer ou sauvegarder.
# La classe modale est celle qui a 80 mm comme indice de classe.

# ==========

# Reprendre l'exercice pour la variable Masse(kg), Hauteur(cm) ou Longueur(cm) du fichier
'Ours.txt'. Attention à la lecture de ce fichier de données!

# ==========

# Essayez les commandes suivantes pour les données d'ours:

  summary(nom-du-data-frame)
  plot(nom-du-data-frame)

# ==========
```

2. Analyse en composantes principales, analyse en coordonnées principales, analyse des correspondances

```
# Il faut d'abord indiquer à R quel est votre dossier de travail.
# Menu File => Change dir...
# ou menu ou Tools => Change Working Directory... (selon la machine).

# Analyse en composantes principales
# Pour cet exemple, les données se trouvent dans le fichier texte "Ex_ACP_p_392.txt".

# Lire les données et les transférer dans une matrice:

  Y = read.table("Ex_ACP_p_392.txt",header=T)
  Y.mat = as.matrix(Y)

# Centrer la matrice Y par colonne:

  Y.cent = apply(Y.mat,2,scale,center=TRUE,scale=FALSE)

# Note: le paramètre 2 indique de calculer le centrage par colonne (et non par ligne)

# Calculer la matrice de covariance:

  Y.cov = cov(Y.mat)

# ou encore (équivalent):

  Y.cov = cov(Y.cent)

# Calculer les valeurs propres et les vecteurs propres:

  Y.eig = eigen(Y.cov)

# Vérifier les valeurs propres et les vecteurs propres:

  Y.eig$values
  Y.eig$vectors

# Transférer les vecteurs propres dans la matrice U:

  U = Y.eig$vectors

# Calculer la matrice F: F = Ycentré * U

  F = Y.cent %*% Y.eig$vectors

# ou encore (équivalent):

  F = Y.cent %*% U
```

```
# Vérifier le contenu de la matrice F:

  F

# Diagramme des 2 premières colonnes de F:

  plot(F[,1],F[,2],xlim=c(-4,4),ylim=c(-3,3),asp=1,xlab="Axe 1",ylab="Axe 2")

# ou encore (équivalent):

  plot(F,xlim=c(-4,4),ylim=c(-3,3),asp=1,xlab="Axe 1",ylab="Axe 2")

# Notes: (-4,4) = bornes de l'abscisse, (-3,3) = bornes de l'ordonnée.
#      asp=1 : le rapport des dimensions abscisse/ordonnée est fixé à 1.

# Ajouter au diagramme les 2 premières colonnes de la matrice U:

  arrows(x0=0,y0=0,U[,1]*3,U[,2]*3)

# Note: pour cet exemple, les coordonnées des vecteurs propres sont multipliées par 3.

# Exercice: faire l'ACP pour le fichier "Spiders_28x12.sp" des araignées des Pays-Bas (van
der Aart & Smeenk-Enserink 1975. Neth. J. Zool. 25: 1-45).

# Pour les fichiers d'abondance d'espèces, on peut faire subir une transformation au fichier
avant de calculer l'ACP qui préserve la distance euclidienne entre les objets. Voir
"Transformations" plus bas.

# ==========

# Analyse en composantes principales à l'aide de la librairie 'vegan'
# de Yari Oksanen <http://cc.oulu.fi/~jarioksa/softhelp/vegan.html>

# Scaling = 1: préserve les distances euclidiennes entre les objets
# Scaling = 2: préserve les corrélations entre les descripteurs

# Appeler d'abord la librairie "vegan":

  library(vegan)

# L'ACP est calculée par la fonction RDA de l'analyse canonique de redondance:

  toto=rda(Y.mat,scale=FALSE)
  summary(toto, scaling=1)
  plot(toto, scaling=1)

# ou encore, pour préserver la corrélation entre les descripteurs:

  summary(toto, scaling=2)
  plot(toto, scaling=2)
```

```
# Exercice: faire l'ACP du fichier "Spiders_28x12.spe" à l'aide de "vegan".

# ==========

# Analyse des correspondances à l'aide de la librairie 'vegan'
# de Yari Oksanen <http://cc.oulu.fi/~jarioksa/softhelp/vegan.html>

# Scaling = 1: matrices F et V, comme dans Canoco
# Scaling = 2: matrices V-chapeau et F-chapeau, comme dans Canoco

# Exemple 1: AFC de Numerical ecology (1998) p. 457, Table 9.11
  library(vegan)
  Table9.11=read.table("Table_9.11.txt")
  Table.afc=cca(Table9.11)
  Table.afc
  summary(Table.afc,scaling=1,axes=2)
  plot(Table.afc,scaling=1)

# Exemple 2: AFC des cafés de Neuchâtel (D. Borcard)
  library(vegan)
  cafes.spe=read.table("Cafes_10x6.spe")
  cafes.afc=cca(cafes.spe)
  cafes.afc
  plot(cafes.afc, scaling=2)
  summary(cafes.afc,scaling=2,axes=5)

# Exemple 3: AFC des poissons de Table 11.3 de Numerical ecology (1998)
  library(vegan)
  Table11.3 = read.table("Table_11-3.txt")
  Table11.3.mat = as.matrix(Table11.3)
  Table11.3.mat
# On ne peut pas calculer l'AFC si le fichier a des lignes ou colonnes qui  somment à 0
# On sélectionnera les lignes qui ont une somme > 0.
# Il s'agit des lignes 1,3,4,5,6,7,8,9,10. On crée un vecteur « choix » :
  choix=c(1,3:10)
  Y.mat=Table11.3.mat[choix,1:6]
  Y.mat
  poissons.afc=cca(Y.mat)
  plot(poissons.afc,scaling=1)
  summary(poissons.afc,scaling=1,axes=2)

# ==========
```

```
# Analyse en coordonnées principales

# On obtient cette analyse à l'aide de la fonction "cmdscale" de la librairie MASS
# Quelques fonctions de distance sont disponibles dans la commande "dist": euclidienne, maximum, Manhattan, Canberra, (D = 1 - Jaccard) (qui porte le nom de "binary"), Minkowski.

# Pour les fichiers d'abondance d'espèces, on obtient d'autres choix en faisant d'abord subir une transformation au fichier (voir ci-dessous) avant de calculer la distance euclidienne.

# Exemple: analyse du fichier "Spiders_28x12.spe"
  spiders=read.table("Spiders_28x12.spe")
  spiders.mat=as.matrix(spiders)

# D'abord: transformation de Hellinger des abondances d'espèces
  spiders.hell=transfodata(spiders.mat,5)
  spiders.D1=dist(spiders.mat,method="eucl")

# Analyse en coordonnées principales
  library(MASS)
  toto4=cmdscale(spiders.D1,5,eig=TRUE)
  x=toto4$points[,1]
  y=toto4$points[,2]
  plot(x,y,xlim=c(-150,50),ylim=c(-100,25),asp=1,xlab="Axe 1",ylab="Axe 2")

# Note: "asp=1" force les deux axes à utiliser la même échelle. De cette façon, les distances entre les objets sont des projections de leurs distances réelles.

# ==========

# Transformations

# Charger d'abord la fonction "transfodata" de Stéphane Dray
# Menu File => Source R code... et sélectionner le fichier-source "Transformations.R"

# Application à un fichier d'abondances d'espèces:
  transfodata(nomfich,1)
# en utilisant le bon numéro de transformation:
#   1 transformation de corde
#   2 métrique du chi2
#   3 distance du chi2
#   4 profiles d'abondances d'espèces
#   5 transformation de Hellinger

# ==========
```

3. Régression multiple en langage R

```
# Lire les données du fichier 'Ours.txt' pour produire le 'data frame' Ours:
  Ours=read.table("Ours.txt",header=T,row.names=1)
  Ours

# Ce tableau contient des caractères (lettres) en dernière colonne.
# Créer une matrice ne contenant que la portion numérique du 'data frame':
  Ours.mat=as.matrix(Ours[,1:4])

# Pour décrire le modèle de régression, on peut utiliser le 'data frame' Ours
# dans une commande 'lm' ('linear model'):
  toto=lm(Ours[,2] ~ Ours[,1] + Ours[,3] + Ours[,4])

# On peut aussi utiliser la matrice dans la commande 'lm':
  toto=lm(Ours.mat[,2] ~ Ours.mat[,1] + Ours.mat[,3] + Ours.mat[,4])

# La commande peut également s'écrire:
  toto=lm(Ours.mat[,2] ~ Ours.mat[,c(1,3,4)])

# On peut utiliser directement les noms des variables du 'data frame'
# dans la description du modèle (il est utile de prévoir des noms courts):
  toto=lm(Masse.kg. ~ Age.estim. + Hauteur.cm. + Longueur.cm., data=Ours)

# Les coefficients de régression se trouvent dans l'objet 'toto'.
  toto

Call:
lm(formula = Ours[, 2] ~ Ours[, 1] + Ours[, 3] + Ours[, 4])

Coefficients:
(Intercept)    Ours[, 1]    Ours[, 3]    Ours[, 4]
   12.4854      10.4577      -0.6903       1.0860
```

```
# On obtient les coefficients de régression et les tests par 'summary':
  summary(toto)

Call:
lm(formula = Ours[, 2] ~ Ours[, 1] + Ours[, 3] + Ours[, 4])

Residuals:
    Min      1Q  Median      3Q     Max
-48.803  -7.842  -1.358   9.980  32.583

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  12.4854    41.8943   0.298 0.768150
Ours[, 1]    10.4577     2.6408   3.960 0.000549 ***
Ours[, 3]    -0.6903     0.5561  -1.241 0.225990
Ours[, 4]     1.0860     0.1022  10.630 9.21e-11 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 17.18 on 25 degrees of freedom
Multiple R-Squared: 0.9224, Adjusted R-squared: 0.9131
F-statistic: 99.06 on 3 and 25 DF,  p-value: 5.253e-14


# On peut obtenir un tableau de tests additionnels par la commande:
  anova(toto)
# Attention: ce tableau est d'emploi rare et difficile à interpréter.
# On y trouve le test de l'effet additionnel de chaque variable
# dans l'ordre de leur inclusion dans le modèle.

# On peut obtenir les valeurs ajustées au modèle de régression:
  Ajuste=predict(toto)          ou encore       Fitted=fitted.values(toto)
# On vérifie les valeurs obtenues :
  Ajuste                        ou encore       Fitted

# On peut également obtenir les résidus du modèle :
  Resid=residuals(toto)
# On vérifie les valeurs obtenues :
  Resid

# Quatre graphiques sont produits en séquence par la commande:
  plot(toto)

# On peut comparer deux modèles emboîtés:
  modele1=lm(Ours.mat[,2] ~ Ours.mat[,c(1,3,4)])
  modele2=lm(Ours.mat[,2] ~ Ours.mat[,c(1,3)])
  anova(modele1,modele2)

# ==========
```

4. Analyse canonique de redondance (RDA) en langage R

```
# Lire les données du fichier "Table_11-3.txt" et les transférer dans la matrice Table11.3.mat,
# puis dans les matrices Y.mat et X.mat:

  Table11.3 = read.table("Table_11-3.txt")
  Table11.3.mat = as.matrix(Table11.3)
  Y.mat=Table11.3.mat[,1:6]
  X.mat=Table11.3.mat[,7:10]
  Y.mat
  X.mat

# Centrer la matrice Y, centrer et réduire la matrice X:
# Note: le paramètre 2 indique de calculer le centrage par colonne

  Y = apply(Y.mat,2,scale,center=TRUE,scale=FALSE)
  X = apply(X.mat,2,scale,center=TRUE,scale=TRUE)
  Y
  X

# La 4ème colonne de X est colinéaire aux colonnes 2 et 3.
# Créer la matrice X3 contenant les 3 premières colonnes de X:

  X3=X[,1:3]

# Créer X'X et calculer inv[X'X]:

  library(MASS)
  invX3 = ginv(t(X3) %*% X3)

# Calculer la matrice de projection, projection(nxn) = X inv[X'X] X':

  projX3 = X3 %*% invX3 %*% t(X3)

# Calculer les valeurs ajustées aux régressions, Yhat = projX * Y:

  Yhat3 = projX3 %*% Y
  Yhat3

#
# Peut-on calculer Yhat à partir de X (4 colonnes) plutôt que X3?
  invX = ginv(t(X) %*% X)
  projX = X %*% invX %*% t(X)
  Yhat = projX %*% Y
  Yhat
# Pourquoi cela est-il possible malgré la colinéarité de X?
# Consultez la documentation de la fonction 'ginv' !
```

```
# Calculer la matrice de covariance des valeurs ajustées avant l'ACP:

   Yhat3.cov = cov(Yhat3)

# Calculer les valeurs propres et les vecteurs propres (ACP):

   Yhat3.eig = eigen(Yhat3.cov)

# Vérifier les valeurs propres et les vecteurs propres:

   Yhat3.eig$values
   Yhat3.eig$vectors

# Trouver les valeurs propres de la RDA telles que calculées
# par Canoco. Elles sont exprimées comme des fractions de la
# variance totale de Y.
# La somme des variances de Y est d'abord obtenue en sommant
# les termes diagonaux de la matrice de covariance de Y:

   trace = sum(diag(cov(Y)))
   trace

   EigvalCanoco = Yhat3.eig$values[1:3] / trace
   EigvalCanoco

# Transférer les 3 premiers vecteurs propres dans la matrice U:
# F = Yhat * U (cadrage de type 1 en ACP)

   U = Yhat3.eig$vectors[,1:3]

# Calculer la matrice F ('Sample scores' de Canoco):

   F = Y %*% U
   F

# Calculer la matrice Z: Z = Yhat * U (cadrage de type 1 en ACP)
# (ce sont les 'Sample scores which are linear combinations of
#  environmental variables' de Canoco)

   Z = Yhat3 %*% U
   Z

# Diagramme des 2 premières colonnes de F, avec l'axe 2 inversé;
# ajouter au diagramme les 2 premières colonnes de la matrice U*10:

   plot(F[,1],-F[,2],xlim=c(-10,16),ylim=c(-10,10),asp=1,xlab="Axe 1",ylab="Axe 2")
   arrows(x0=0,y0=0,U[,1]*10,-U[,2]*10,code=0)
```

```
# Diagramme des 2 premières colonnes de Z, avec l'axe 2 inversé:

  plot(Z[,1],-Z[,2],xlim=c(-10,16),ylim=c(-10,10),asp=1,xlab="Axe 1",ylab="Axe 2")
  arrows(x0=0,y0=0,U[,1]*10,-U[,2]*10,code=0)

# Notes: bornes de l'abscisse = (-10,16), bornes de l'ordonnée = (-10,10).
# asp=1: le rapport des dimensions abscisse/ordonnée est fixé à 1.
# Arrows, code=0: flèches sans pointes; code=2: pointe à l'extrémité.

# Représentation des variables explicatives dans le diagramme.
# Calculer d'abord la matrice des corrélations entre X et Z:

  corXZ=cor(X,Z)
  corXZ

# Former la matrice diagonale 'D' des poids, sqrt(lambda(k)/trace):

  D = diag(sqrt(Yhat3.eig$values[1:3]/trace))

# Calculer les 'Biplot scores of environmental variables' de Canoco;
# tracer les variables environnementales dans le diagramme:

  posX = corXZ %*% D
  arrows(x0=0,y0=0,posX[,1]*10,-posX[,2]*10,code=2)

# ==========

# Exemple d'analyse canonique partielle
#
# Calculer la contribution partielle (unique) de la variable
# 'Profondeur' à Y en présence des autres variables de la matrice X.

# Lire les données, etc.

  Table11.3 = read.table("Table_11-3.txt")
  Table11.3.mat = as.matrix(Table11.3)
  Y.mat=Table11.3.mat[,1:6]
  X.mat=Table11.3.mat[,7:10]
  Y = apply(Y.mat,2,scale,center=TRUE,scale=FALSE)
  X = apply(X.mat,2,scale,center=TRUE,scale=TRUE)
  Y
  X

# Former la matrice XX contenant la variable 'Profondeur' et la
# matrice de covariables W contenant les 3 autres variables de X:

  XX = X[,1]
  W = X[,2:4]
```

```
# Régresser la profondeur (XX) sur les covariables (W);
# calculer les résidus de cette régression:

  library(MASS)
  invW = ginv(t(W) %*% W)
  projW = W %*% invW %*% t(W)
  XXres = XX - (projW %*% XX)

# Calculer l'analyse canonique (RDA) de Y par XXres:

  invXXres = ginv(t(XXres) %*% XXres)
  projXXres = XXres %*% invXXres %*% t(XXres)
  YhatXX = projXXres %*% Y
  YhatXX.cov = cov(YhatXX)
  YhatXX.eig = eigen(YhatXX.cov)
  Z = YhatXX %*% YhatXX.eig$vectors[,1]

# Écrire la valeur propre et la position des objets sur l'axe canonique:

  YhatXX.eig$values[1]
  Z

# Trouver la valeur propre de la RDA partielle telle que calculée
# par Canoco pour 'Profondeur' (= 0.083, P = 0.001). Elle est
# exprimée comme une fraction de la variance totale de Y:

  EigvalCanoco = YhatXX.eig$values[1] / sum(diag(cov(Y)))
  EigvalCanoco

# Cette valeur propre indique la contribution partielle de la variable
# 'Profondeur' à Y en présence des autres variables de la matrice X.

# ==========
```

5. Analyses canoniques (RDA et CCA) à l'aide de la librairie « vegan »

```
# Les données se trouvent dans un fichier ASCII "Table_11-3.txt"
#
# Lire les données et les transférer dans la matrice Table11.3.mat,
# puis dans les matrices Y.mat (variables réponse) et X.mat (explicatives):

  Table11.3 = read.table("Table_11-3.txt")
  Table11.3.mat = as.matrix(Table11.3)
  Y.mat=Table11.3.mat[,1:6]
  X.mat=Table11.3.mat[,7:10]
  Y.mat
      Spec1 Spec2 Spec3 Spec4 Spec5 Spec6
Site1     1     0     0     0     0     0
Site2     0     0     0     0     0     0
Site3     0     1     0     0     0     0
Site4    11     4     0     0     8     1
Site5    11     5    17     7     0     0
Site6     9     6     0     0     6     2
Site7     9     7    13    10     0     0
Site8     7     8     0     0     4     3
Site9     7     9    10    13     0     0
Site10    5    10     0     0     2     4

  X.mat
      Depth Coral Sand Other
Site1     1     0    1     0
Site2     2     0    1     0
Site3     3     0    1     0
Site4     4     0    0     1
Site5     5     1    0     0
Site6     6     0    0     1
Site7     7     1    0     0
Site8     8     0    0     1
Site9     9     1    0     0
Site10   10     0    0     1

# Activer la librairie 'vegan' de Jari Oksanen
  library(vegan)
# Analyse canonique de redondance (RDA)
  Table.rda = rda(Y.mat,X.mat)
  Table.rda

Call:
rda(X = Y.mat, Y = X.mat)

              Inertia Rank
Total         112.889
Constrained   108.341    3
```

```
Unconstrained   4.548    4
Inertia is variance

Eigenvalues for constrained axes:
  RDA1   RDA2   RDA3
74.523 24.942  8.876

Eigenvalues for unconstrained axes:
    PC1      PC2      PC3      PC4
4.188785 0.313863 0.037037 0.008463

  plot(Table.rda)


  anova(Table.rda)
Permutation test for rda under reduced model

Model: rda(X = Y.mat, Y = X.mat)
        Df     Var      F  N.Perm Pr(>F)
Model     3 108.341 31.761 100.000 < 0.01 ***
Residual  4   4.548
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

# ==========

# Exemple d'analyse canonique partielle
#
# Calculer la contribution partielle (unique) de la variable
# 'Profondeur' à Y en présence des autres variables de la matrice X.

  Table.rda = rda(Y.mat,X.mat[,1],X.mat[,2:4])
  anova(Table.rda)

Permutation test for rda under reduced model

Model: rda(X = Y.mat, Y = X.mat[, 1], Z = X.mat[, 2:4])
        Df     Var      F  N.Perm Pr(>F)
Model     1   9.341 8.215 100.000 < 0.01 ***
Residual  4   4.548
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

# Autre façon de faire une RDA à l'aide de 'vegan':
# on peut spécifier le modèle, comme en régression multiple

  attach(Table11.3)
  Table.rda = rda(Table11.3[,1:6] ~ Depth + Coral + Sand + Other)

# ==========
```

Sélection progressive (pas à pas) des variables explicatives en RDA
à l'aide de la fonction "forward.sel" de la librairie "packfor" écrite par Stéphane Dray.
Cette fonction utilise deux fichiers: les variables réponse et les variables explicatives.

Voici quelques lignes tirées de la fenêtre d'information de la fonction "forward.sel":

Forward selection with multivariate Y by permutation under reducel model

Description

Performs a forward selection by permutation of residuals under reduced model. Y can be multivariate.

Usage
function (Y, X, K = nrow(X) - 1, R2tresh = 0.99, nperm = 999,
R2more = 0.001, Xscale = TRUE, Ycenter = TRUE, Yscale = FALSE)

```
# Charger la librairie "packfor"
  library(packfor)

# Charger les deux tableaux de données (variables réponse et explicatives)
  spiders.spe=read.table("Spiders_28x12.spe")
  spiders.env=read.table("Spiders_28x4.env")

# La fonction "forward.sel" produit un tableau montrant l'inclusion pas à pas des variables explicatives. L'utilisateur décidera à partir de ce tableau quelles variables il conservera pour la RDA par "vegan".
  toto=forward.sel(spiders.spe,spiders.env)

# Méthode rapide pour fournir à la fonction RDA de "vegan" les colonnes (variables) de "spiders.env" à conserver:
  rda(spiders.spe,spiders.env[,toto$order[1:3]])

# ==========
```

Analyse canonique des correspondances

```
# Exemple 1 de CCA: récif corallien
  library(vegan)
  Table11.3 = read.table("Table_11-3.txt")
  Table11.3.mat = as.matrix(Table11.3)
  Y.mat=Table11.3.mat[,1:6]
  X.mat=Table11.3.mat[,7:10]
  recif.cca=cca(Y.mat,X.mat)
  recif.cca
  plot(recif.cca, scaling=1)
  summary(recif.cca,scaling=1,axes=3)

# Exemple 2 de CCA: les cafés de Neuchâtel (D. Borcard)
  library(vegan)
  cafes.spe=read.table("Cafes_10x6.spe")
  cafes.env=read.table("Cafes_10x3.env")
  cafes.cca=cca(cafes.spe,cafes.env)
  cafes.cca
  plot(cafes.cca, scaling=1)
  summary(cafes.cca,scaling=1,axes=3)

# ==========
```