Scénario: spectrométrie RMN et métabolome

Résumé

Ce scénario illustre l'utilisation d'une décomposition de spectres RMN sur une base d'ondelettes pour l'études de données métabolomiques. Exploration des données par analyse en composantes principales (ACP) ; recherche du niveau de détails de la décomposition et des zones discriminantes du spectre par sparsePLS DA (Lê Cao et al. 2011 [3]) et par une méthode d'agrégation de modèles ; cette démarche pourrait être utilisée pour la recherche de biomarqueurs.

1 Introduction

1.1 Ondelettes et spectres

Il est devenu courant d'utiliser une décomposition sur une base d'ondelettes Mallat(1999)[5] pour analyser [4] des données de spectrométrie présentant des pics et singularités. En revanche un lissage spline est plus adapté à des données de spectrométrie NIR régulières. Cette décomposition permet de combiner différentes transformations selon le niveau de détails sélectionné pour, dans cet exemple, débruiter le signal et également retirer une *l*igne de base encore artificiellement présente et qui pertube très sensiblement l'analyse. L'avantage de la décomposition en ondelettes, par rapport à celle de Fourier et pour les objectifs poursuivis, est le caractère local des fonctions de base et donc des coefficients. Cette localité permet d'identifier des portions de courbes donc de spectre pertinentes en fonction de l'objectif recherché plutôt que des composantes fréquentielles non localisées en temps.

1.2 Objectifs

A travers cette stratégie, l'objectif est triple :

1. "nettoyer" les données (bruit et ligne de base) en sélectionnant le ou les bons niveaux de détails de la décomposition,
2. sélectionner les portions de spectre les plus discriminantes pour permettre au biologiste d'identifier les métabolites les plus perturbés par l'influence du DEHP,
3. "valider" la sélection obtenue par une approche de type "apprentissage statistique".

En effet, les données sont de "grande dimension" avec $p = 1200$, le nombre de variables largement plus grand que $n = 60$, la taille de l'échantillon. Le risque est évident d'identifier des variables ou coefficients de la décomposition ou portions de spectre certes discriminants mais seulement sur l'échantillon considéré car, compte tenu de p, la discrimination est toujours possible par sur-apprentissage. Le principe de "validation" repose sur le caractère plus ou moins prédictif des coefficients identifiés. Si le modèle sélectionné est "généralisable" : capable de prévoir raisonnablement les facteurs d'un échantillon indépendant, qui n'a donc pas participé à l'estimation des paramètres du modèle, on peut penser que le modèle et donc la sélection obtenus sont suffisamment robustes pour orienter le biologiste vers des métabolites pertinents.

Remarques :

- Cette approche est strictement équivalente à celle qui serait poursuivie lors de la recherche de bio-marqueurs pour, par exemple, identifier des métabolites susceptibles de définir un diagnostic précoce. Dans le cas présent, la bonne prévision des facteurs : génotype et dose, n'a évidemment pas d'intérêt clinique mais constituera un indicateur de confiance.
- L'identification des métabolites est une analyse à part entière qui demande des compétences très spécifiques, elle ne peut être abordée dans ce scénario.

Hormis l'utilisation d'une base d'ondelettes, le déroulement du travail reste classique : après lecture des données et vérification, celles-ci sont décomposées sur une base de Haar. Les spectres bruts sont explorés analyse en composantes principales (ACP) pour détecter les problèmes avant d'identifier les coefficients discriminants par *spase* PLS DA (Lê Cao et al. 2011 [3]) et une méthode d'agrégation de modèles. Enfin, une dernière section estime des erreurs de prévision pour "valider" la sélection obtenue.

2 Les données

2.1 Descriptif

Les données (Eveillard et al. 2009 [1]) proviennent d'une étude cherchant à évaluer les mécanismes de toxicité de perturbateurs endocriniens (DHEP) sur le foie de souris. Trente souris génétiquement modifiées (PPARα déficientes) et trente souris sauvages (Wild Type) sont aléatoirement réparties en 6 groupes de même effectif (10). Chaque groupe est soumis pendant 21 jours à un régime contenant différentes doses de DEHP (phtalate). Le plan d'expérience comprend donc 2 facteurs :

- Génotype : souris sauvage (WT) ou mutante (PPAR),
- Traitement avec des doses de DEHP : 0, 20 ou 200 mg/kg/day.

Après euthanasie et extraction des foies, des données de spectrométrie RMN sont acquises. Pour chaque échantillon, les spectres obtenus subissent différentes transformations de référence fournies par le constructeur pour ce type de données (cf. Eveillard et al. (2009) [1] pour les détails).

2.2 Importation sous R

Les données sont disponibles dans le fichier souris-rmn.dat du répertoire d'URL `http://wikistat.fr/data`.

```
# Lire les données
rawData = read.table("souris-rmn.dat",
   header = T, row.names = 1, sep = "", dec = ".")
# Construire les deux facteurs
facteur=expand.grid(
Traitement=factor(c(rep(1,10),rep(2,10),rep(3,10)),
   labels=c("ctrl","D20","D200")),
   Genotype=factor(c(1,2),labels=c("WT","PPAR")))
dim(rawData)
```

Les 1200 variables sont renommées pour correspondre au *déplacement chimique du spectre*. Les lignes sont renommées par un numéro identificateur de souris.

```
variables = round(seq(9.995, -1.995, -0.01), 3)
individus = 10:69
genotype = facteur[, 2]
traitement = facteur[, 1]
# noms explicites des facteurs croisés
facteurs = paste(genotype, traitement, sep = ".")
dimnames(rawData)[[1]] = individus
dimnames(rawData)[[2]] = variables
# Vérification
rawData[1:5,1:10]
# Remise des variables en ordre croissant
rawData = rawData[, dim(rawData)[[2]]:1]
rawData[1:5,1:10]
```

Une première représentation des données signale des variables constantes, elles sont supprimées. D'autre part, seuls sont conservés les déplacements positifs.

```
# boîtes parallèles
boxplot(rawData)
# suppressions des variables de variance nulle
sd2 = apply(rawData, 2, var)
rawData = rawData[,sd2 != 0]
# suppression des déplacements négatifs
MZ = as.numeric(colnames(rawData))
rawData=rawData[,MZ>= 0]
boxplot(rawData)
# Quelques spectres
ts.plot(t(rawData[1:3,]))
ts.plot(t(rawData[10:13,]))
```

3 Décomposition en ondelette

Beaucoup de bases d'ondelettes différentes sont proposées dans la littérature et les librairies. Le choix semble peu influent et est limité ici à la base très classique de Haar. Nous nous intéressons par ailleurs à une classe particulière d'ondelettes, celles UDWT (Undecimated discrete wavelet transform) (Percival et Walden 2000 [6]) qui possède la propriété ici importante d'être invariante par translation d'une valeur, ce qui n'est pas le cas des décompositions en ondelette classiques, et de ne pas nécessité un nombre de temps de mesures puissance de 2.

Les calculs sont réalisés avec la librairie `wmtsa` en demandant 9 niveaux de détails. Les signaux ou spectres sont donc décomposés par filtrages successifs en coefficients d'approximation A_9 et des niveaux de détails D_1 à D_9.

La décomposition sur la base crée un objet de type `wavTransform` avec un affichage spécifique des coefficients :

```
library(wmtsa)
## coefficients pour le spectre i
i=1
plot(wavMODWT(rawData[i, ], wavelet = "haar"))
```

La décomposition est itérée pour tous les spectres et les coefficients stockés dans un objet de type `array` à trois indices (échantillon, temps, niveau), qui perd l'attribut `wavTransform`.

```
longueur = ncol(rawData)
niveaux = 1:9
details = array(dim = c(nrow(rawData), longueur,
   length(niveaux)))
for (k in niveaux) {
for (i in 1:nrow(rawData)) {
        details[i, , k] = wavMODWT(rawData[i, ],
  wavelet = "haar")[[k]]
}
}
```

4 Exploration des spectres bruts

4.1 ACP globale

Une ACP est indispensable à l'étude de la bonne cohérence des données ou plus exactement pour mettre en évidence des comportements atypiques.

```
library(mixOmics)
acp.raw = pca(rawData,center=TRUE, scale.=TRUE)
pch1 = c(rep(15, 30), rep(16, 30))
cex = c(rep(1, 30), rep(1.2, 30))
color=as.numeric(traitement)
plot(acp.raw$x[, 1], acp.raw$x[, 2], cex = cex,
   col = color, pch = pch1,
xlab = "PC 1", ylab = "PC 2")
abline(h = 0, v = 0, lty = 2)
lgd(0.3, 0.61)
```

Ni les génotypes, ni les doses ne se distinguent mais plutôt des échantillons atypiques à identifier. Après identification, ces spectres sont superposés en noir sur les autres en zoomant sur des zones particulières.

```
# Indices des spectres atypiques
atyp=which(acp.raw$x[,1]>18)
zoom = 1:100
color2=rep(2,nrow(rawData))
largeur=rep(1,nrow(rawData))
largeur[atyp]=2
color[atyp]=1
ts.plot(t(rawData[,zoom]),col=color2,lwd=largeur,
   xlab = "chemical shift")
#-- chemical shift : 3.995-5.485 --#
zoom = 400:500
ts.plot(t(rawData[,zoom]),col=color2,lwd=largeur,
   xlab = "chemical shift")
#-- chemical shift : 5.485-6.485 --#
zoom = 500:600
ts.plot(t(rawData[,zoom]),col=color2,lwd=largeur,
   xlab = "chemical shift")
```

Voir en quoi ces spectres se distinguent plus particulièrement par des décalages dans certaines zones. Il s'agit d'un artefact (mauvais recalage de la "ligne de base") insuffisamment pris en compte et source de sérieux problèmes. Ces décalages présentent en effet une variance plus importante que le signal recherché. L'ACP se montre toujours très efficace pour mettre en évidence les artefacts de mesure.

4.2 ACP par niveau de détails

L'ACP est maintenant calculée sur différents niveaux de détails. Le bon choix permet d'éliminer la ligne de base de basse fréquence tout en réduisant le bruit associé à des fréquences plus élevées. Un effort important est consenti pour construire des graphiques facilement lisibles ce qui complique évidemment les commandes R. Penser à exécuter la fonction `lgd()` de l'annexe qui affiche des légendes explicites des graphiques.

```
acp.det_1 = pca(details[, , 1], center = TRUE,
   scale. = TRUE)
acp.det_2 = pca(details[, , 2], center = TRUE,
   scale. = TRUE)
acp.det_3 = pca(details[, , 3], center = TRUE,
   scale. = TRUE)
#-- Visualisation des résultats
# Niveau D1
plot(acp.det_1$x[, 1], acp.det_1$x[, 2], cex =cex,
   col=color,pch=pch1,xlab="PC 1",ylab="PC 2")
abline(h = 0, v = 0, lty = 2)
lgd(0.3, -1.03)
# Niveau D2
plot(acp.det_2$x[, 1], acp.det_2$x[, 2], cex =cex,
   col=color,pch=pch1,xlab="PC 1",ylab="PC 2")
abline(h = 0, v = 0, lty = 2)
lgd(0.3, -1.03)
# Niveau D3
plot(acp.det_3$x[, 1], acp.det_3$x[, 2], cex = cex,
   col = as.numeric(traitement), pch = pch1,
   xlab = "PC 1", ylab = "PC 2")
abline(h = 0, v = 0, lty = 2)
lgd(-1.04, 0.61)
```

Observer l'absence d'individus atypiques comme précédemment ainsi que la bonne séparation des génotypes quelque soit le niveau de détails. Les doses élevées appliquées à des souris sauvages se regroupent plus nettement à certains niveaux.

5 Sélection des détails

5.1 Contexte

La version parcimonieuse ou *sparse* de la régression PLS en version "analyse discriminante" (Lê Cao et al. 2011 [3]) est utilisée pour sélectionner les 25 détails de niveau 2 les plus discriminants. Cette version de PLS est obtenu en considérant un groupe de variables Y construites à partir des fonctions indicatrices des deux facteurs.

Bien sûr, beaucoup d'autres méthodes d'apprentissage peuvent être utilisées pour sélectionner les "meilleurs" coefficients d'ondelettes susceptibles de bien discriminer les traitements dans ce problème de classification supervisée. La plupart des autres scénarios de Wikistat sur l'apprentissage statistique mettent systématiquement en place une comparaison des performances de prévision de différentes méthodes. La mise en œuvre d'une technique comme *random forest*, souvent très performantes dans beaucoup d'exemples, est testée mais n'a pas apporté d'amélioration très significative. L'accent est donc mis sur la sparse PLS-DA pour les capacités de cette approche à produire des graphiques facilement interprétables et montrant une bonne discrimination des traitements et génotypes ; il n'est sans doute pas nécessaire de mettre en place une stratégie trop lourde. Attention à ne pas généraliser, chaque jeu de données et problème possède ses propres caractéristiques.

5.2 Représentation et sélection par sparse PLS-DA

La régression PLS2, version "canonique" plutôt que "régression", traite deux paquets de variables quantitatives. En considérant celui X des coefficients de la décomposition en ondelettes à un niveau donnée et celui Y constitué des indicatrices des facteurs permet de construire une version "analyse discriminante" de la régression PLS.

```
library(mixOmics)
# Sélection du niveau de détails
niv = 2
# Variables X
X = details[, , niv]
# Variables Y
Y = cbind(as.numeric(genotype) - 2,
```

```
   as.numeric(traitement))
colnames(Y) = c("genotype", "traitement")
# sparse PLS DA
det.res = spls(X, Y, ncomp = 2, keepX = c(25, 25),
   mode = "regression")
# Représentation des individus
plotIndiv(det.res, comp = c(1, 2), ind.names=F,
   col = color, pch = pch1, cex = cex)
lgd(-0.3, -1)
```

Vérifier la toujours bonne séparation des génotypes mais aussi celle des doses de DEHP, toujours plus nette pour les souris sauvages.

Il s'agit ensuite d'identifier et sélectionner les détails ou portions de spectre associées à ces discriminations.

```
# Sélection des plus forts ''loadings''
varsel = apply(abs(det.res$loadings$X), 1, sum)>0
# Identificateur en fonction du
# rôle sur l'axe 1 ou l'axe 2
X.label = rep("", ncol(X))
X.label[det.res$loadings$X[, 1] > 0] = 1
X.label[det.res$loadings$X[, 1] < 0] = 2
X.label[det.res$loadings$X[, 2] > 0] = 3
X.label[det.res$loadings$X[, 2] < 0] = 4
# Coloration des identificateurs
# (blanc si pas sélectionné)
xcol = rep("white", ncol(X))
x.col = c("orange", "brown", "darkmagenta",
   "darkcyan", "white")
xcol[!varsel] = 5
xcol = x.col[as.numeric(X.label)]
ycol = rep("black", 6)
cols = list(xcol, ycol)
# Construction des indicatrices des facteurs
fact = as.factor(facteurs)
ind.mat = unmap(as.numeric(fact))
det.res$Y = ind.mat
# Graphique des variables
plotVar(det.res, X.label = X.label,
   Y.label = levels(fact), col = cols)
```

Sur la première composante, les détails marqués 1 et 2 participent clairement à la bonne séparation des génotypes. Ceci sous-entend que les portions de spectres considérés et donc des quantités de métabolites associés diffèrent selon le génotype. En revanche, c'est sur la deuxième composante que se distinguent les détails, marqués 3 et 4, participant le plus à la bonne séparation des doses de traitement.

5.3 Sélection par forêt aléatoire

Les méthodes d'agrégation de modèles se montrent souvent très performantes pour chercher les variables les plus discriminantes même en très grande dimension avec $p >> n$; *random forest* en est un représentant. Il s'agit d'un problème de classification supervisée dans lequel la variable à prévoir est le facteur `fact` à 2×3 niveaux et croisant dose et génotype.

```
# vérification de la variable Y
Y=fact
Y
# Sélection du niveau de détails
niv = 2
# Variables X
X = details[, , niv]
# Librairie
library(randomForest)
result.rf=randomForest(X,Y,importance=T,mtry=800)
# Détails jugés globalement importants
imp.detail=sort(result.rf$importance[,7],
   decreasing=T)[1:20]
par(xaxt="n")
plot(imp.detail,type="h",ylab="Importance",
   xlab="Variables")
points(imp.detail,pch=20)
par(xaxt="s")
axis(1,at=1:20,labels=names(imp.detail),
```

```
   cex.axis=0.8,las=3)
```

Remarques :

- Le taux d'erreur estimé *out of bag* ne semble pas significativement meilleur que ceux calculés ci-dessous pour la sPLS-DA ; nous nous limitons aux résultats de la sPLS-DA.
- L'intersection des ensembles de détails révélés par PLS-DA et par *random forest* est vide. Il faudrait une évaluation "biologique" de la pertinence des métabolites concernés. Celle-ci n'a été faite que pour la liste issue de la PLS-DA (Gonzales et al. 2013)[2].

5.4 Identification des détails

Le dernier graphique va permettre de repérer sur le spectre le positionnement des détails les plus discriminants. C'est l'analyse de ces décalages qui permettra au spécialiste de RMN d'identifier des métabolites associées et donc des fonctions biologiques concernées par d'une part l'effet génotype et d'autre part l'effet dose.

```
# Indices des détails sélectionnés
bio.d1 = det.res$loadings$X[, 1] > 0
bio.d2 = det.res$loadings$X[, 1] < 0
bio.d3 = det.res$loadings$X[, 2] > 0
bio.d4 = det.res$loadings$X[, 2] < 0
# Calcul du spectre median  pour le graphique
spect.med = apply(rawData, 2, median)
# Abscisses
xTick = c(1, 105, 2*105, 3*105, 4*105, 5*105,
   6*105, 7*105, 8*105, ncol(rawData))
raw.x = ncol(rawData):1
bio.d1 = raw.x[bio.d1]
bio.d2 = raw.x[bio.d2]
bio.d3 = raw.x[bio.d3]
bio.d4 = raw.x[bio.d4]
# Tracé du spectre médian
plot(1:ncol(rawData), spect.med, type = "l",
   lwd = 1.7, ylim = c(-1.8, max(spect.med)),
   xlab = "chemical shift", ylab = "",axes=F)
axis(1, xTick, colnames(rawData)[xTick])
# Positionnmeent des légendes des détails
text(970,-0.5,"1",cex=0.7,col="orange",font=2)
text(970,-1.1,"2",cex=0.7,col="brown",font= 2)
text(970,-1.7,"3",cex=0.7,col="darkmagenta",font=2)
text(970,-2.3,"4",cex=0.7,col="darkcyan",font= 2)
# Positionnement des détails sur le spectre
y = -0.0275
rug(bio.d1, ticksize = 0.128 + y, col = "orange")
rug(bio.d1, ticksize = 0.106 + y, col = "white")
rug(bio.d2, ticksize = 0.104 + y, col = "brown")
rug(bio.d2, ticksize = 0.080 + y, col = "white")
rug(bio.d3, ticksize = 0.078 +y,col="darkmagenta")
rug(bio.d3, ticksize = 0.054 + y, col = "white")
rug(bio.d4, ticksize = 0.052 + y, col = "darkcyan")
rug(bio.d4, ticksize = 0.028 + y, col = "white")
# Quelques lignes en plus
box()
yy = 10
lines(c(1, ncol(rawData) + yy), c(-0.85, -0.85),
   col = 2, lty = 1)
lines(c(1, ncol(rawData) + yy), c(-1.45, -1.45),
   col = 2, lty = 1)
lines(c(1, ncol(rawData) + yy), c(-2.05, -2.05),
   col = 2, lty = 1)
```

6 Caractère prédictif des "bio-marqueurs"

6.1 Principe

La question qui se pose est la suivante : les détails sélectionnés sont-ils bien représentatifs des effets doses. deux approches sont possibles à ce niveau : une batterie de tests (ANOVA) appliqués à l'ensemble des coefficients assortie d'un contrôle de type "Benjamini Hochberg" du taux de fausses découvertes (FDR) pour chercher ceux significativement différents entre les conditions. Voir à ce propos le scénario de mise en œuvre des tests multiples et de calcul du FDR.

L'autre approche dite global ou *wrapper* consiste à sélectionner les variables ou coefficients qui présentent de bonnes propriétés prédictives ou de "généralisation" pour éviter de se focaliser, à cause de la très grande dimension, sur des artefacts spécifiques de l'échantillon tiré.

6.2 Calculs

Il est nécessaire d'exécuter la fonction `classificateur` de l'annexe. Il s'agit donc d'évaluer la performance de la sélection des coefficients pour chaque niveau de détails et de comparer avec celle sur les données brutes. L'échantillon est aléatoirement séparé entre un sous-ensemble d'apprentissage et un sous ensemble de test. Les facteurs ou classes sont équilibrées entre les deux sous-ensembles. Les résultats de cette procédure de validation sont calculés pour chacun des détails D1 à D5 et pour les données brutes. Remarque : le choix du nombre de variables ou de coefficients (`nvar=25`) par niveau de détails n'a pas été optimisé par validation croisée comme c'est classique en apprentissage statistique mais déterminé *a priori* par le biologiste ; la raison en est la faible taille de l'échantillon.

```
library(mixOmics)
library(grid)
# Initialisation des paramètres
niveaux = 1:5  # Niveaux de details de D1 à D5
      N = 50   # Nombre de validations
  ncomp = 6    # Nombre de composantes
   nvar = 25   # Nombre de variables à retenir
# Initialisation des matrices
X=rawData
Y=cbind(as.numeric(genotype)-2,
   as.numeric(traitement))
colnames(Y) = c("genotype", "traitement")
geno = trea = both = matrix(NA,
   nrow = length(niveaux) + 1, ncol = ncomp)
colnames(geno) = colnames(trea) =
   colnames(both) = paste("Comp", 1:ncomp)
rownames(geno) = rownames(trea) = rownames(both) =
   c("raw", paste("det.", niveaux, sep = ""))
# Modèle à tester pour les rawData
raw.mod = spls(X, Y, ncomp = ncomp,
   keepX = rep(nvar, ncomp))
# Variables sélectionnées dans le modèle
raw.keepX = (raw.mod$loadings$X != 0)
raw.keepY = (raw.mod$loadings$Y != 0)
# Procédurer de validation itérée N fois
for (j in 1:N) {
#-- training set --#
tr.1 = sample( 1:10, 6)
tr.2 = sample(11:20, 6)
tr.3 = sample(21:30, 6)
tr.4 = sample(31:40, 6)
tr.5 = sample(41:50, 6)
tr.6 = sample(51:60, 6)
train = c(tr.1, tr.2, tr.3, tr.4, tr.5, tr.6)
raw.train.mod = spls.model(X[train, ], Y[train, ],
   ncomp = ncomp, mode = "regression",
   keepX = raw.keepX, keepY = raw.keepY)
raw.test.pred=
   predict(raw.train.mod,X[-train,])$predict
n = nrow(X[-train, ])
for (comp in 1:ncomp) {
  pred=t(apply(raw.test.pred[,,comp],1,
           classificateur,Y = Y))
        geno[1,comp]=sum(Y[-train,1]!=pred[,1])/n
        trea[1,comp]=sum(Y[-train,2]!=pred[,2])/n
        both[1,comp]=sum(!((Y[-train,1]==pred[,1])&
           (Y[-train, 2] == pred[, 2]))) / n
}
for (niv in 1:length(niveaux)) {
# Modèle à tester pour les détails
det.mod = spls(details[ , , niv], Y, ncomp =
         ncomp, keepX = rep(nvar, ncomp))
# Variables sélectionnées dans le modèle
det.keepX = (det.mod$loadings$X != 0)
det.keepY = (det.mod$loadings$Y != 0)
```

```
det.train.mod=spls.model(details[train,,niv],
   Y[train,],ncomp=ncomp,mode="regression",
   keepX=det.keepX,keepY=det.keepY)
det.test.pred=predict(det.train.mod,
   details[-train,,niv])$predict
for (comp in 1:ncomp) {
pred = t(apply(det.test.pred[ , , comp], 1,
   classificateur, Y = Y))
geno[1+niv,comp]=sum(Y[-train,1]!=pred[,1])/n
trea[1+niv,comp]=sum(Y[-train,2]!=pred[,2])/n
both[1+niv,comp]=sum(!((Y[-train,1]==pred[,1])&
   (Y[-train,2]==pred[,2])))/n
        }}}
```

6.3 Mise en forme des résultats

Représentaiton des zones de spectre qui discriminent les niveaux des différents facteurs.

```
geno=data.frame(t(geno)/N,factor=rep("genotype",
   ncomp), ncomp = 1:ncomp)
trea=data.frame(t(trea)/N,factor=rep("treatment",
   ncomp), ncomp = 1:ncomp)
both = data.frame(t(both) / N, factor =
   rep("genotype x treatment",ncomp),ncomp=1:ncomp)
# Répresentation des résultats
nd = length(niveaux) + 1
pchs = c(16, 2, 0, 3, 5, 6)
cols = c("black", rep(gray(0.75), nd - 1))
labs = c(1, paste(1, 2:ncomp, sep = ":"))
# genotype
xyplot(raw+det.1+det.2+det.3+det.4+det.5 ~
   ncomp | factor,data = geno,type="o",col=cols,
         xlab = "Components", ylab = "Error rate",
   scales=list(x=list(relation="free",at=1:ncomp,
         labels=labs)),main="",lty=1:nd,pch=pchs,
   strip=strip.custom(par.strip.text=list(cex=1),
         bg = gray(0.9)))
# traitement
xyplot(raw+det.1+det.2+det.3+det.4+det.5 ~
   ncomp|factor,data=trea,type="o",col=cols,
   xlab = "Components", ylab = "Error rate",
   scales = list(x = list(relation = "free",
         at = 1:ncomp, labels = labs)),
   main = "", lty = 1:nd, pch = pchs,
   strip=strip.custom(par.strip.text=list(cex=1),
         bg = gray(0.9)))
# genotype x treatment
xyplot(raw+det.1+det.2+det.3+det.4+det.5 ~
   ncomp|factor,data=both,type="o",col=cols,
   xlab = "Components", ylab = "Error rate",
   scales = list(x = list(relation = "free",
         at = 1:ncomp, labels = labs)),
   main = "", lty = 1:nd, pch = pchs,
   strip=strip.custom(par.strip.text=list(cex=1),
         bg = gray(0.9)))
# Legende
lbls = c(expression(raw), expression(italic(D)[1]),
  expression(italic(D)[2]),expression(italic(D)[3]),
  expression(italic(D)[4]),expression(italic(D)[5]),
  expression(italic(D)[6]),expression(italic(D)[7]),
  expression(italic(D)[8]),expression(italic(D)[9]))
titre.leg = expression(scriptscriptstyle(Signal))
xyplot(0 ~ 0,type="n",scales=list(draw=FALSE),
   xlab = "",ylab="",par.settings=list(axis.line =
   list(col = "transparent")),key = list(lines =
   list(lty = 1:nd, type = "o", pch = pchs,cex=0.6,
   col = cols),space = "bottom",
   text = list(lab = lbls[1:6], cex = 0.9),
   columns = 3, title = titre.leg))
```

Annexe : Fonctions

Fonction d'affichage des légendes

```
lgd = function(x, y) {
  par(new = "TRUE")
  plot(0, 0, type = "n", axes = FALSE, xlab = "",
            ylab = "")
  text(x + 0.16, y + 0.16, "Wild-type", cex = 0.8)
  text(x + 0.13, y + 0.06, expression(paste("PPAR",
            alpha)), cex = 0.8)
  rect(x, y, x + 0.31, y + 0.23)
  text(x + 0.53,y+0.37,"mg DEHP/kg/day",cex=0.7)
  xpch1 = x + c(0.37, 0.51, 0.65)
  ypch1 = y + rep(0.29, 3)
  text(xpch1, ypch1,c("0","20","200"),cex=0.8)
  rect(x + 0.31, y + 0.23, x + 0.74, y + 0.43)
  ypch1 = y + rep(0.16, 3)
  points(xpch1, ypch1, cex = 1,pch=15,col=1:3)
  points(xpch1, ypch1, cex = 1, pch = 0)
  ypch1 = y + rep(0.06, 3)
  points(xpch1, ypch1, cex=1.2,pch=16,col=1:3)
  points(xpch1, ypch1, cex = 1.2, pch = 1)
  rect(x + 0.31, y, x + 0.74, y + 0.23)
}
```

Fonction de calcul du "classifieur" de la sparse PLS DA

```
classificateur = function(x, Y) {
  Yunique = list()
  Y = as.matrix(Y)
  q = ncol(Y)
  for (i in 1:q) Yunique[[i]] = unique(Y[, i])
  Yclass = expand.grid(Yunique)
  x = matrix(x, nrow = nrow(Yclass), ncol =
           ncol(Yclass), byrow = TRUE)
  if (q > 1) {
    d = apply((x - Yclass)^2, 1, sum)
  }
  else {
    d = (x - Yclass[, 1])^2
  }
  cl.id = which.min(d)
  as.matrix(Yclass[cl.id, ])}
```

Références

[1] A. Eveillard, F. Lasserre, M de. Tayrac, A. Polizzi, S. Claus, C. Canlet, L. Mselli-Lakhal, G. Gotardi, A. Paris, H. Guillou, P.G. Martin et T. Pineau, *Identification of potential mechanisms of toxicity after di-(2-ethylhexyl)-phthalate (DEHP) adult exposure in the liver using a systems biology approach*, Toxicol Appl. Pharmacol **3** (2009), n° 236.

[2] I. González, A. Eveillard, C. Canlet, A. Paris, T. Pineau, P. Besse, Martin P. et S. Déjean, *Selecting the good level of details in undecimated wavelet transform improve the classification of samples from metabolomic data*, JP Journal of Biostatistics **xx** (2013), n° x, xx–xx.

[3] K. A. Lê Cao, S. Boistard et P. Besse, *Sparse PLS Discriminant Analysis : biologically relevant feature selection and graphical displays for multiclass problems*, BMC Bioinformatics **12** (2011), n° 253.

[4] P Lió, *Wavelets in bioinformatics and computational biology : State of art and perspectives*, Bioinformatics (2003).

[5] Stéphane Mallat, *A wavelet tour of signal processing*, Academic Press, 1999.

[6] D.B. Percival et A.T. Walden, *Wavelet methods for time series analysis*, Cambridge University Press, 2000.