DM2: Analyse Discriminante PLS exploratoire

Kateryna Stetsun

2025-11-21

Introduction

L'Analyse Discriminante PLS exploratoire (ADPLS) est une méthode multivariée qui combine les principes de la régression PLS (Partial Least Squares) et de l'analyse discriminante.
Son objectif est de construire des composantes linéaires des variables explicatives X maximisant la capacité de discrimination des groupes définis par la variable qualitative Y.

Autrement dit, on cherche à extraire des axes factoriels $f = XMu$ expliquant à la fois une grande part de la variance de X (inertie totale) et une forte part de la variance expliquée par les classes de Y.
Les notations utilisées sont celles du cours :

- $X \in \mathbb{R}^{n \times p}$: matrice des variables quantitatives centrées ;
- $Y \in \mathbb{R}^{n \times q}$: matrice d'indicatrices des q classes (non centrée) ;
- $W = \text{diag}(w_i)$: matrice diagonale des poids individuels ;
- M : métrique définie positive sur l'espace des variables ;
- $f = XMu$: composante discriminante associée à la direction u ;
- $\Pi_Y = Y(Y'WY)^{-1}Y'W$: projecteur sur l'espace engendré par Y ;
- $\widehat{X} = \Pi_Y X$: projection de X sur cet espace.

Partie 2 — Programmation

Dans cette seconde partie, nous présentons l'implémentation progressive de l'Analyse Discriminante PLS (ADPLS).

L'objectif principal est de reproduire, étape par étape, les calculs nécessaires à la construction des composantes discriminantes et à l'extraction des informations les plus pertinentes pour la discrimination entre groupes.

Définition des variables d'entrée et fonctions utilitaires

Les données d'entrée nécessaires à l'algorithme sont les suivantes :

- la matrice X de dimension $n \times p$, contenant les variables explicatives numériques (non nécessairement centrées) ;

- la matrice Y de dimension $n \times q$, formée des indicatrices (0/1) correspondant aux q modalités de la variable de groupe ;
- le paramètre H, représentant le nombre de composantes discriminantes à extraire ;
- le vecteur de poids w, de longueur n, qui peut être laissé nul pour signifier des poids égaux ;
- la matrice métrique M de dimension $p \times p$, éventuellement identitaire.

Une option `center_scale` permet en outre de centrer et réduire les variables de X si nécessaire.

```
# Définition des entrées
#X <- as.matrix(X)         # matrice n x p
#Y <- as.matrix(Y)         # matrice n x q
#H <- 3                    # nombre de composantes
#w <- rep(1 / nrow(X), nrow(X))   # poids égaux
#M <- diag(ncol(X))        # matrice métrique identité
#center_scale <- TRUE      # centrage-réduction activée
```

Afin de rendre les calculs plus modulaires et lisibles, plusieurs fonctions utilitaires ont été implémentées.

Elles permettent de définir les opérations pondérées et les transformations associées à la métrique M.

```
# matrice de pondération diagonale
Wmat <- function(w) {
  if (is.null(w)) return(NULL)
  diag(as.numeric(w), nrow = length(w))
}

# norme pondérée avec w vecteur poids ou W matrice diag
w_norm2 <- function(v, w) {
  if (is.null(w)) return(as.numeric(t(v) %*% v))
  as.numeric(t(v) %*% (w * v))
}

# produit scalaire pondéré
w_prod <- function(a, b, w) {
  if (is.null(w)) return(as.numeric(t(a) %*% b))
  as.numeric(t(a) %*% (w * b))
}

# racine symétrique de M (M supposée SPD ou semi-définie)
M_half <- function(M) {
  ev <- eigen(M, symmetric = TRUE)
  vals <- pmax(ev$values, 0)
  vecs <- ev$vectors
  vecs %*% diag(sqrt(vals), length(vals)) %*% t(vecs)
}

# inverse de la racine (pseudo-inverse si valeurs nulles)
M_half_inv <- function(M) {
  ev <- eigen(M, symmetric = TRUE)
  vals <- ev$values
  vecs <- ev$vectors
  invvals <- ifelse(vals > 0, 1 / sqrt(vals), 0)
  vecs %*% diag(invvals, length(invvals)) %*% t(vecs)
}
```

1. Calcul des H premières composantes ADPLS

Par definition nous savons que $f_h = XMu_h$, où u_h est le vecteur propre de $E_h = \hat{X}'W\hat{X}$ associé à la plus grande valeur propre, projeté orthogonalement aux composantes précédentes.

```
adpls_part1 <- function(X, Y, H = 2, w = NULL, M = NULL, center_scale = TRUE) {

#préparation
  X <- as.matrix(X)
  Y <- as.matrix(Y)
  n <- nrow(X)
  p <- ncol(X)

  if (nrow(Y) != n) stop(" X et Y doivent avoir le même nombre de lignes (individus)!")
  if (is.null(w)) w <- rep(1 / n, n)
  w <- as.numeric(w)
  if (length(w) != n) stop("longueur de w pas la meme que de n (nrow(X))!")
  W <- diag(w)
  if (center_scale) X <- scale(X, center = TRUE, scale = TRUE)
  if (is.null(M)) M <- diag(1, p)
  if (!all(dim(M) == c(p, p))) stop("M doit être p x p!")

# racines de M
  sqrtM <- M_half(M)
  inv_sqrtM <- M_half_inv(M)

# projecteur Pi_Y
  YYw <- t(Y) %*% (W %*% Y)
  inv_YYw <- solve(YYw)
  PiY <- Y %*% inv_YYw %*% t(Y) %*% W

# projection de X sur l'espace de Y
  Xhat <- PiY %*% X

# matrice E (pour la diagonalisation)
  E <- t(Xhat) %*% (W %*% Xhat)

# réserves pour résultats
  if (is.null(H) || !is.numeric(H) || H < 1) stop("H doit être un entier positif!")
  H <- as.integer(H)
  U_list <- vector("list", H)
  F <- matrix(0, n, H)
  Ftilde <- matrix(0, n, H)

# boucle sur h = 1..H
  for (h in seq_len(H)) {
    if (h == 1) {

# première composante
      Estar <- sqrtM %*% E %*% sqrtM
      ev <- eigen(Estar, symmetric = TRUE)
# vecteur propre principal en métrique transformée
      ustar <- ev$vectors[, 1]
# retour dans l'espace u
```

```
      u <- inv_sqrtM %*% ustar

# normalisation selon la contrainte u'Mu = 1
      unorm <- as.numeric(t(u) %*% (M %*% u))
      if (unorm <= 0) stop("norme u'Mu non strictement positive!")
      u <- u / sqrt(unorm)

    } else {

# composantes suivantes
      Fprev <- F[, 1:(h - 1), drop = FALSE]
      Dprime <- t(Fprev) %*% (W %*% X)
      D <- t(Dprime)
# Pi_D_perp
      inner <- t(D) %*% (M %*% D)
      inv_inner <- solve(inner)
      PiDperp <- diag(1, p) - D %*% inv_inner %*% t(D) %*% M
# matrice à diagonaliser dans l'espace réduit
      Estar_proj <- sqrtM %*% PiDperp %*% E %*% t(PiDperp) %*% sqrtM
      ev <- eigen(Estar_proj, symmetric = TRUE)
      ustar <- ev$vectors[, 1]
      u <- inv_sqrtM %*% ustar
      unorm <- as.numeric(t(u) %*% (M %*% u))
      if (unorm <= 0) stop(paste("norme u'Mu non positive (h=", h, ")!", sep = ""))
      u <- u / sqrt(unorm)
    }

# score f_h
    f_h <- X %*% (M %*% u)
    fnorm2 <- as.numeric(t(f_h) %*% (W %*% f_h))
    if (fnorm2 <= 0) stop(paste("norme de f_h <= 0 (h=", h, ")!", sep = ""))
# normalisation
    ftilde_h <- f_h / sqrt(fnorm2)

# stockage
    U_list[[h]] <- as.numeric(u)
    F[, h] <- as.numeric(f_h)
    Ftilde[, h] <- as.numeric(ftilde_h)
  }

# liste de vecteurs en matrice p x H
  U_mat <- do.call(cbind, U_list)
  colnames(U_mat) <- paste0("u", seq_len(ncol(U_mat)))
  colnames(F) <- paste0("f", seq_len(ncol(F)))
  colnames(Ftilde) <- paste0("f_tilde", seq_len(ncol(Ftilde)))

  return(list(U = U_mat, F = F, Ftilde = Ftilde, Xhat = Xhat,
              E = E, PiY = PiY, M = M, w = w))
}
```

2. Calcul des indicateurs $S(f_h)$ et $R^2(f_h, Y)$

Les formules associées sont les suivantes :

$S(f_h) = \frac{\|f_h\|_W^2}{\mathrm{tr}(X'WX)}$ et $R^2(f_h, Y) = \frac{\|\hat{f}_h\|_W^2}{\|f_h\|_W^2}$,

où

$\hat{f}_h = \Pi_Y f_h = \Pi_Y X M u_h$ et $\Pi_Y = Y(Y'WY)^{-1}Y'W$.

L'indicateur $S(f_h)$ va mesurer la part d'inertie totale expliquée par la composante f_h. Plus $S(f_h)$ est élevé, plus la composante f_h participe à la structuration globale des données.

L'indicateur $R^2(f_h, Y)$ va mesurer la part de variance de f_h expliquée par les classes Y. Un R^2 proche de 1 indique qu'une grande partie de la variance de f_h est expliquée par la variable de groupe — la composante est donc fortement discriminante.

L'implémentation en R est donnée ci-dessous.

Elle réutilise les résultats obtenus dans la fonction `adpls_part1`, puis effectue, pour chaque composante, les calculs successifs des normes pondérées nécessaires à l'évaluation des deux indicateurs :

```
adpls_part2 <- function(X, Y, H = 2, w = NULL, M = NULL, center_scale = TRUE) {

# calcul de la q1
  part1 <- adpls_part1(X, Y, H, w, M, center_scale)

# préparation
  X <- as.matrix(X)
  W <- diag(part1$w)
  M <- part1$M
  F <- part1$F
  U <- part1$U
  Xhat <- part1$Xhat
  E <- part1$E

  total_inertia <- sum(diag(t(X) %*% (W %*% X)))

#vecteurs résultats
  S_vec <- numeric(H)
  R2_vec <- numeric(H)

# boucle sur h
  for (h in seq_len(H)) {
    f_h <- F[, h, drop = FALSE]
    fnorm2 <- as.numeric(t(f_h) %*% (W %*% f_h))
    S_vec[h] <- fnorm2 / total_inertia
    fhat_h <- part1$PiY %*% f_h
    top_val <- as.numeric(t(fhat_h) %*% (W %*% fhat_h))
    R2_vec[h] <- top_val / fnorm2
  }

# résultats
  part1$S <- S_vec
  part1$R2 <- R2_vec

  return(part1)
}
```

Notre vecteur `S_vec` regroupe les proportions d'inertie expliquées par chacune des composantes discriminantes, tandis que `R2_vec` fournit une mesure synthétique de la corrélation entre les axes extraits et la

variable de classification.

Ces deux indicateurs constituent la base de l'interprétation exploratoire de l'ADPLS : les composantes associées à un $S(f_h)$ élevé et un $R^2(f_h, Y)$ proche de 1 sont celles qui contribuent le plus efficacement à la discrimination entre groupes.

3. Calcul des coordonnées des centres de gravité des q classes sur les H axes discriminants

L'objectif de cette étape est de déterminer, pour chacune des q classes considérées, les coordonnées de son centre de gravité dans l'espace défini par les H premières composantes discriminantes issues de l'ADPLS.

Soit $F = [f_1, f_2, \ldots, f_H]$ la matrice des composantes discriminantes. Chaque individu i a un vecteur de scores $f_i = (f_{i1}, \ldots, f_{iH})$.

Si Y est la matrice d'indicateurs de classes $(n \times q)$: $Y_{ik} = 1$ si l'individu i appartient à la classe k, 0 sinon.

Alors es coordonnées du centre de gravité de la classe k sur l'axe h s'expriment comme une moyenne pondérée des scores individuels :

$$g_{kh} = \frac{\sum_{i=1}^{n} W_i \, Y_{ik} \, f_{ih}}{\sum_{i=1}^{n} W_i \, Y_{ik}}$$

En notation matricielle :

$$G = (Y'WY)^{-1} Y'WF$$

où :

- G est une matrice $q \times H$,
- chaque ligne correspond au barycentre d'une classe dans l'espace factoriel,
- chaque colonne correspond à un axe discriminant.

Ainsi, G fournit la position moyenne de chaque classe dans le sous-espace discriminant, selon la métrique pondérée par W, assurant la cohérence avec la construction précédente des composantes.

L'implémentation en R est donnée ci-dessous :

```r
adpls_part3 <- function(X, Y, H = 2, w = NULL, M = NULL, center_scale = TRUE) {

# calcul de la q2
  part2 <- adpls_part2(X, Y, H, w, M, center_scale)

# préparation
  W <- diag(part2$w)
  F <- part2$F
  Y <- as.matrix(Y)

# centres de gravité
  YtWY <- t(Y) %*% W %*% Y
  YtWF <- t(Y) %*% W %*% F
  G <- solve(YtWY, YtWF)

  rownames(G) <- colnames(Y)
```

```r
  colnames(G) <- paste0("Axe_", seq_len(H))

# résultats
  part2$centers <- G

  return(part2)
}
```

La matrice G obtenue contient donc les coordonnées pondérées des centres de gravité des classes dans le sous-espace discriminant. Chaque ligne de G représente la position barycentrique d'une classe sur les H axes principaux.

Ces coordonnées constituent une étape essentielle pour l'interprétation géométrique : elles permettent de visualiser la disposition relative des groupes dans le plan factoriel (h, m), ce qui sera exploité ultérieurement lors de la représentation graphique (question 5).

4. Calcul des coordonnées des variables de X

Déterminons les coordonnées des variables initiales X_j dans l'espace des H composantes discriminantes issues de l'ADPLS.

Chaque variable X_j peut être corrélée à chacune des composantes discriminantes f_h, ce qui permet d'analyser son rôle dans la discrimination entre les groupes.

La corrélation pondérée entre X_j et f_h est donnée par :

$$r_{jh} = \frac{\langle X_j, f_h \rangle_W}{\|X_j\|_W \, \|f_h\|_W}$$

où :

- X_j : j-ième variable centrée-réduite de X,
- f_h : h-ième composante discriminante,
- $\langle \cdot, \cdot \rangle_W$: le produit scalaire pondéré par la matrice W,
- $\| \cdot \|_W$: la norme associée à ce produit scalaire.

Sous forme matricielle, la matrice des corrélations entre les variables et les axes discriminants s'écrit :

$$C = \text{corr}(X, F) = D_X^{-1} \, (X'WF) \, D_F^{-1}$$

avec :

- D_X la matrice diagonale des normes pondérées des variables de X_j,
- D_F la matrice diagonale des normes pondérées des composantes f_h.

La matrice C (de dimension $p \times H$) contient ainsi les coordonnées factorielles des variables sur les axes discriminants.

Implémentation en R:

```r
adpls_part4 <- function(X, Y, H = 2, w = NULL, M = NULL, center_scale = TRUE) {

# calcul de la q3
  part3 <- adpls_part3(X, Y, H, w, M, center_scale)

# préparation
  W <- diag(part3$w)
  F <- part3$F
#suppose X centré-réduit
  X <- scale(X, center = TRUE, scale = TRUE)
  n <- nrow(X)
  p <- ncol(X)

# init matrice de corrélation
  C <- matrix(0, nrow = p, ncol = H)

# corrélations pondérées entre X_j et f_h
  for (j in 1:p) {
    xj <- X[, j]
    norm_xj <- sqrt(as.numeric(t(xj) %*% (W %*% xj)))

    for (h in 1:H) {
      fh <- F[, h]
      norm_fh <- sqrt(as.numeric(t(fh) %*% (W %*% fh)))
      if (norm_xj == 0 || norm_fh == 0) {
        C[j, h] <- 0
      } else {
        C[j, h] <- as.numeric(t(xj) %*% (W %*% fh)) / (norm_xj * norm_fh)
      }
    }
  }

  rownames(C) <- colnames(X)
  colnames(C) <- paste0("Axe_", seq_len(H))

# résultats
  part3$var_coords <- C

  return(part3)
}
```

Les coefficients de la matrice C traduisent la corrélation entre les variables initiales et les axes discriminants. Une valeur absolue élevée (proche de 1) indique une forte contribution de la variable à la formation de l'axe considéré, tandis qu'une valeur proche de 0 suggère une contribution faible ou nulle.

Ainsi, l'analyse de C constitue une étape essentielle pour relier les composantes discriminantes aux variables d'origine et pour interpréter la signification géométrique des axes de l'ADPLS.

Fonction générale — intégration du processus ADPLS

Afin de regrouper les différentes étapes de calcul développées précédemment (questions 1 à 4), nous avons implémenté une fonction générale `adpls_full()`.

Cette fonction constitue une synthèse opérationnelle de l'approche ADPLS et permet d'obtenir, à partir des matrices d'entrée X et Y, l'ensemble des résultats nécessaires à l'analyse discriminante partielle par moindres carrés.

```
adpls_full <- function(X, Y, H=2, w=NULL, M=NULL, center_scale=TRUE){

# fonctions utilitaires internes
Wmat <- function(w) {
  if (is.null(w)) return(NULL)
  diag(as.numeric(w), nrow = length(w))
}

w_norm2 <- function(v, w) {
  if (is.null(w)) return(sum(v^2))
  as.numeric(t(v) %*% (w * v))
}

w_prod <- function(a, b, w) {
  if (is.null(w)) return(as.numeric(t(a) %*% b))
  as.numeric(t(a) %*% (w * b))
}

M_half <- function(M) {
  ev <- eigen(M, symmetric = TRUE)
  vals <- pmax(ev$values, 0)
  vecs <- ev$vectors
  vecs %*% diag(sqrt(vals), length(vals)) %*% t(vecs)
}

M_half_inv <- function(M) {
  ev <- eigen(M, symmetric = TRUE)
  vals <- ev$values
  vecs <- ev$vectors
  invvals <- ifelse(vals > 0, 1 / sqrt(vals), 0)
  vecs %*% diag(invvals, length(invvals)) %*% t(vecs)
}


 if (is.null(w)) w <- rep(1 / nrow(X), nrow(X))

# étapes
  part1 <- adpls_part1(X = X, Y = Y, H = H, w = w, M = M, center_scale = center_scale)
  # cat("H =", H, " | length =", length(H), "\n")
  part2 <- adpls_part2(X = X, Y = Y, H = H, w = w, M = M, center_scale = center_scale)
  part3 <- adpls_part3(X = X, Y = Y, H = H, w = w, M = M, center_scale = center_scale)
  part4 <- adpls_part4(X = X, Y = Y, H = H, w = w, M = M, center_scale = center_scale)

 # résultats
  return(list(F=part1$F, Ftilde=part1$Ftilde,
              S=part2$S, R2=part2$R2,
              centers=part3$centers,
              var_coords=part4$var_coords))
}
```

5. Représentation graphique des individus, des centres de gravité et des variables

Dans cette dernière étape, nous devons visualiser les résultats obtenus dans les étapes précédentes.

L'objectif est triple :

1. Afficher les individus dans le plan factoriel (h, m) choisi par l'utilisateur.
2. Afficher les centres de gravité des q classes sur ce même plan.
3. Afficher les variables de X projetées sur le plan choisi et tracer le cercle unité.

La réponse sera la fonction `plot_adpls()` qui va afficher graphique dans le plan (h, m).

Cette fonction trace :

- les individus colorés selon leur classe,
- les centres de gravité des q classes,
- les variables de X projetées sur le plan choisi,
- le cercle unité, pour faciliter l'interprétation des corrélations.

```r
library(RColorBrewer)

plot_adpls <- function(out, Y = NULL, Y_labels = NULL, h = 1, m = 2,
                       show_variables = TRUE, circle = TRUE,
                       main = NULL, cex_ind = 0.7, cex_cent = 1.0,
                       xlim = NULL, ylim = NULL) {

  Ftilde <- out$Ftilde
  centers <- out$centers
  var_coords <- out$var_coords
  n <- nrow(Ftilde)

# identification des étiquettes de classes
  if (is.null(Y_labels) & !is.null(Y)) {
    lab <- apply(Y, 1, function(r) which(r == 1))
    Y_labels <- factor(lab)
  } else if (is.null(Y_labels)) {
    stop("Y_labels ou Y doivent être fournis!")
  }

# individus
  xind <- Ftilde[, h]
  yind <- Ftilde[, m]
  classes <- as.factor(Y_labels)
  cols <- rainbow(length(levels(classes)))

#palette
  n_classes <- length(levels(classes))
  cols <- brewer.pal(min(max(n_classes, 3), 8), "Set2")
  if (n_classes > 8) cols <- rep(cols, length.out = n_classes)

# tracé des individus
  plot(xind, yind,
       col = adjustcolor(cols[as.numeric(classes)], alpha.f = 0.7),
       pch = 19,
```

```
       cex = cex_ind,
       xlab = paste0("Axe ", h),
       ylab = paste0("Axe ", m),
       main = main,
       bg = "white",
       xlim = xlim,
       ylim = ylim)

legend("topright",
       legend = levels(classes),
       col = cols,
       pch = 19,
       cex = 1,
       title = "Classes",
       bg = "white",
       box.lwd = 0.5,
       box.col = "black",
       text.col = "black")

# centres de gravité
  points(centers[, h], centers[, m], col = "black", pch = 8, cex = cex_cent)
  text(centers[, h], centers[, m], labels = rownames(centers), pos = 3)

# variables dans le plan
  if (show_variables) {
    arrows(rep(0, nrow(var_coords)), rep(0, nrow(var_coords)),
           var_coords[, h], var_coords[, m],
           length = 0.07, col = "grey40", lwd = 1)

    text(var_coords[, h], var_coords[, m],
         labels = rownames(var_coords),
         cex = 0.7, col = "grey20")

# cercle unité pour corrélations
    if (circle) {
      angles <- seq(0, 2*pi, length.out = 200)
      lines(cos(angles), sin(angles), lty = 2, col = "black")
    }
  }
}

# opérateur utilitaire : renvoie a si non nul, sinon b
`%||%` <- function(a, b) if (!is.null(a)) a else b
```

Interprétation du graphique:

Le graphique obtenu illustre de manière synthétique la structure discriminante des données :

- Les individus sont représentés sous forme de points colorés selon leur appartenance à une classe donnée. La proximité de deux individus indique une similarité de profil selon les variables discriminantes.
- Les centres de gravité matérialisent la position moyenne de chaque groupe dans l'espace factoriel. Leur distance traduit la séparation effective entre classes.
- Les flèches des variables indiquent les directions les plus contributives à la discrimination. Une variable située proche du cercle unité et bien alignée avec un axe traduit une forte corrélation avec celui-ci.

- Le cercle unité fournit un repère visuel : seules les variables proches de son pourtour exercent une influence significative sur la discrimination.

Exemple d'application avec les données `Datagenus` du premier TP :

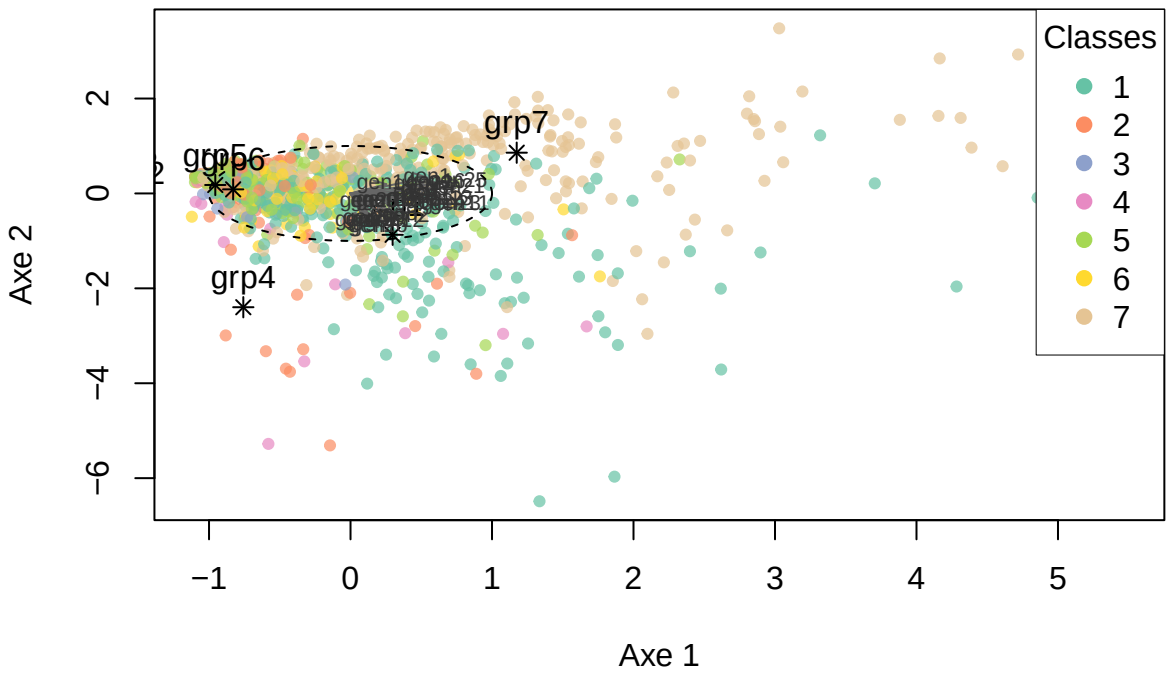
```
data <- read.table("Datagenus.csv", header = TRUE)
X <- scale(as.matrix(data[, paste0("gen", 1:27)]), center = TRUE, scale = TRUE)
grp <- as.factor(data$forest)
Y <- model.matrix(~ grp - 1)
out <- adpls_full(X, Y, H = 3, w = rep(1 / nrow(X), nrow(X)), M = NULL, center_scale = FALSE)

data.frame(
  Axe = 1:3,
  S = round(out$S, 3),
  R2 = round(out$R2, 3),
  Discriminant = round(out$S * out$R2, 3)
)
```

```
##   Axe     S    R2 Discriminant
## 1   1 0.184 0.200        0.037
## 2   2 0.116 0.178        0.021
## 3   3 0.051 0.090        0.005
```

Le tableau des indicateurs montre que la première composante (axe 1) concentre la part la plus importante de l'inertie totale ($S_1 = 18{,}4\%$) et qu'elle est la plus corrélée avec la variable de groupe ($R^2_1 = 0{,}20$). Son pouvoir discriminant (0,037) est donc le plus élevé. Elle représente l'axe principal de séparation entre les types de forêts. La deuxième composante (0,021) contribue de manière secondaire, tandis que la troisième (0,005) joue un rôle marginal.

L'essentiel de l'information discriminante est donc concentré sur les deux premiers axes.

```
plot_adpls(out, Y = Y, Y_labels = grp, h = 1, m = 2, show_variables = TRUE)
```



Le graphique de projection sur le plan formé par les axes 1 et 2 illustre cette structure. Chaque point représente une forêt colorée selon sa classe, et les croix indiquent les centres de gravité des groupes. On observe une séparation nette entre les classes 4 et 7, qui apparaissent bien distinctes dans l'espace factoriel. Cela suggère que ces deux types de forêts possèdent des compositions en genres d'arbres clairement différenciées.

En revanche, les classes 5 et 6 sont proches l'une de l'autre : leurs centres de gravité sont situés dans une même zone du plan, ce qui traduit une similarité partielle dans leur composition floristique ou un recouvrement des caractéristiques discriminantes identifiées par les deux premières composantes. Cette proximité peut indiquer soit une continuité écologique entre ces deux types de forêts, soit une difficulté du modèle à les séparer de manière nette.

Conclusion

Dans cette partie, nous avons programmé pas à pas l'Analyse Discriminante PLS (ADPLS).

Nous avons d'abord construit les composantes discriminantes, puis calculé les indicateurs essentiels, les centres de gravité des classes et les corrélations entre les variables et les axes.

Enfin, nous avons créé une fonction graphique permettant de représenter les individus, les classes et les variables dans le plan factoriel, ce qui nous a permis de mieux comprendre la structure des données et la capacité du modèle à bien séparer les groupes.

Partie 3 — Application: types forestiers du bassin du Congo

q1

```r
data <- read.table("Datagenus.csv", header = TRUE)
str(data)
```

```
## 'data.frame':    1000 obs. of  70 variables:
##  $ code     : int  1299 2644 1838 534 3213 1861 3445 865 3644 561 ...
##  $ gen1     : int  0 9 9 0 1 19 1 4 3 0 ...
##  $ gen2     : int  0 0 0 4 1 3 4 0 0 0 ...
##  $ gen3     : int  0 3 0 0 0 1 0 0 2 0 ...
##  $ gen4     : int  0 0 0 0 0 11 0 0 0 0 ...
##  $ gen5     : int  0 2 1 9 0 0 2 0 0 0 ...
##  $ gen6     : int  2 2 0 1 0 1 0 1 1 1 ...
##  $ gen7     : int  0 1 3 11 4 0 0 0 1 0 ...
##  $ gen8     : int  55 14 0 19 5 33 12 1 2 0 ...
##  $ gen9     : int  2 1 4 0 1 4 0 0 1 0 ...
##  $ gen10    : int  11 20 63 5 5 13 11 38 17 9 ...
##  $ gen11    : int  16 29 30 24 15 63 10 9 17 0 ...
##  $ gen12    : int  5 5 20 38 23 2 3 0 11 0 ...
##  $ gen13    : int  9 10 13 17 4 76 2 6 12 0 ...
##  $ gen14    : int  0 3 3 0 6 14 0 9 1 0 ...
##  $ gen15    : int  0 0 7 0 0 3 0 0 1 0 ...
##  $ gen16    : int  1 7 0 0 5 36 2 0 1 0 ...
##  $ gen17    : int  0 0 0 14 4 0 2 1 4 0 ...
##  $ gen18    : int  0 0 0 1 0 0 0 0 0 0 ...
##  $ gen19    : int  21 0 0 108 7 3 37 0 8 0 ...
##  $ gen20    : int  2 0 4 3 2 0 0 2 2 0 ...
##  $ gen21    : int  5 5 5 6 0 11 1 3 3 3 ...
##  $ gen22    : int  0 0 0 0 0 2 1 0 0 0 ...
##  $ gen23    : int  0 7 0 6 3 18 3 0 2 0 ...
##  $ gen24    : int  13 5 1 14 7 48 16 0 5 0 ...
##  $ gen25    : int  32 19 3 34 5 130 8 0 6 0 ...
##  $ gen26    : int  1 0 21 87 18 1 9 0 13 0 ...
##  $ gen27    : int  0 2 7 30 8 0 20 0 1 0 ...
##  $ altitude : chr  "398,6220397949" "400,4576416016" "362,2293701172" "600,8876342773" ...
##  $ pluvio_yr: chr  "1455" "1461" "1447" "1401,2221679688" ...
##  $ forest   : int  2 7 5 1 1 7 1 2 1 2 ...
##  $ pluvio_1 : chr  "73,6666641235" "50,6666679382" "72,007019043" "22,5343647003" ...
##  $ pluvio_2 : chr  "74,5" "72,8333358765" "77,605255127" "61,7834854126" ...
##  $ pluvio_3 : chr  "133,8333282471" "118,5" "147,5456085205" "118,2560348511" ...
##  $ pluvio_4 : chr  "139,3333282471" "125" "144,0175476074" "138,0808105469" ...
##  $ pluvio_5 : chr  "120,5" "132,5" "133,798248291" "154,5223999023" ...
##  $ pluvio_6 : chr  "110,3333358765" "112,8333358765" "104,957901001" "149,6374664307" ...
##  $ pluvio_7 : chr  "105,6666641235" "134,1666717529" "110,329826355" "154,465637207" ...
##  $ pluvio_8 : chr  "105,8333358765" "136,6666717529" "104,8368453979" "151,3883514404" ...
##  $ pluvio_9 : chr  "155,1666717529" "157" "151,8578796387" "169,9227142334" ...
##  $ pluvio_10: chr  "169,8333282471" "175" "158,3087768555" "171,9880371094" ...
##  $ pluvio_11: chr  "150" "99,8333358765" "144,1842193604" "83,4226989746" ...
##  $ pluvio_12: chr  "92,1666641235" "58,3333320618" "86,1842041016" "24,5842113495" ...
##  $ geology  : int  3 6 6 5 6 3 3 3 2 3 ...
```

```
##  $ evi_1    : chr  "0,3910871148" "0,3378965855" "0,3610805273" "0,3889489174" ...
##  $ evi_2    : chr  "0,4112656713" "0,381662339" "0,375120908" "0,4088948369" ...
##  $ evi_3    : chr  "0,4365606308" "0,3899267614" "0,4241586626" "0,4338231981" ...
##  $ evi_4    : chr  "0,4635847807" "0,4462485611" "0,4610377252" "0,4899954498" ...
##  $ evi_5    : chr  "0,4911327064" "0,493188709" "0,4872943163" "0,4880128205" ...
##  $ evi_6    : chr  "0,5538878441" "0,5008920431" "0,5046898723" "0,5283816457" ...
##  $ evi_7    : chr  "0,5438943505" "0,5073742867" "0,507065475" "0,5550786257" ...
##  $ evi_8    : chr  "0,5508648753" "0,5237532258" "0,5101108551" "0,5455245972" ...
##  $ evi_9    : chr  "0,5431039333" "0,5151152611" "0,5113150477" "0,5536313057" ...
##  $ evi_10   : chr  "0,5202928782" "0,4978065789" "0,5001526475" "0,5269128084" ...
##  $ evi_11   : chr  "0,5086370707" "0,4807990491" "0,4540195763" "0,5338335633" ...
##  $ evi_12   : chr  "0,4672316313" "0,4335366488" "0,436607182" "0,5167191625" ...
##  $ evi_13   : chr  "0,4851673841" "0,461047858" "0,440245837" "0,5197646022" ...
##  $ evi_14   : chr  "0,4788505137" "0,4410296082" "0,4437948167" "0,5073350668" ...
##  $ evi_15   : chr  "0,5039389729" "0,4584658146" "0,4673160017" "0,4811374545" ...
##  $ evi_16   : chr  "0,5205193162" "0,4493008852" "0,4729451239" "0,5135068297" ...
##  $ evi_17   : chr  "0,5576840639" "0,4633820653" "0,5137551427" "0,4921848178" ...
##  $ evi_18   : chr  "0,4989081919" "0,455308795" "0,4976354837" "0,5154051185" ...
##  $ evi_19   : chr  "0,506745398" "0,4603865445" "0,4795286059" "0,4748531878" ...
##  $ evi_20   : chr  "0,489985764" "0,4533531964" "0,4982682168" "0,5097850561" ...
##  $ evi_21   : chr  "0,4897886813" "0,4377965927" "0,4773576856" "0,4463077486" ...
##  $ evi_22   : chr  "0,489938885" "0,4394916892" "0,4482271075" "0,4376213849" ...
##  $ evi_23   : chr  "0,3840098083" "0,3435108066" "0,398150295" "0,4012963176" ...
##  $ lon      : chr  "16,0056502102" "17,0176638107" "16,4554340326" "15,2934924913" ...
##  $ lat      : chr  "1,5014159492" "2,6258755053" "1,4264519788" "3,862781017" ...
##  $ surface  : chr  "5" "15" "17,5" "20,5" ...
```

```
head(data)
```

```
##   code gen1 gen2 gen3 gen4 gen5 gen6 gen7 gen8 gen9 gen10 gen11 gen12 gen13
## 1 1299    0    0    0    0    0    2    0   55    2    11    16     5     9
## 2 2644    9    0    3    0    2    2    1   14    1    20    29     5    10
## 3 1838    9    0    0    0    1    0    3    0    4    63    30    20    13
## 4  534    0    4    0    0    9    1   11   19    0     5    24    38    17
## 5 3213    1    1    0    0    0    0    4    5    1     5    15    23     4
## 6 1861   19    3    1   11    0    1    0   33    4    13    63     2    76
##   gen14 gen15 gen16 gen17 gen18 gen19 gen20 gen21 gen22 gen23 gen24 gen25 gen26
## 1     0     0     1     0     0    21     2     5     0     0    13    32     1
## 2     3     0     7     0     0     0     0     5     0     7     5    19     0
## 3     3     7     0     0     0     0     4     5     0     0     1     3    21
## 4     0     0     0    14     1   108     3     6     0     6    14    34    87
## 5     6     0     5     4     0     7     2     0     0     3     7     5    18
## 6    14     3    36     0     0     3     0    11     2    18    48   130     1
##   gen27       altitude       pluvio_yr forest      pluvio_1      pluvio_2
## 1     0 398,6220397949            1455      2 73,6666641235          74,5
## 2     2 400,4576416016            1461      7 50,6666679382 72,8333358765
## 3     7 362,2293701172            1447      5  72,007019043  77,605255127
## 4    30 600,8876342773 1401,2221679688      1 22,5343647003 61,7834854126
## 5     8  375,827911377 1403,3333740234      1 50,8944587708 77,6038284302
## 6     0 611,9286499023 1540,3333740234      7 27,2052745819 47,3074798584
##         pluvio_3       pluvio_4       pluvio_5       pluvio_6       pluvio_7
## 1 133,8333282471 139,3333282471          120,5 110,3333358765 105,6666641235
## 2          118,5            125          132,5 112,8333358765 134,1666717529
## 3 147,5456085205 144,0175476074  133,798248291  104,957901001  110,329826355
```

```
## 4 118,2560348511 138,0808105469 154,5223999023 149,6374664307  154,465637207
## 5 106,0520172119  123,153755188 113,6365127563 127,2843933105 133,5442962646
## 6 138,4563598633 136,5421600342 146,0544128418 155,2628936768 154,4674987793
##         pluvio_8       pluvio_9      pluvio_10      pluvio_11     pluvio_12
## 1 105,8333358765 155,1666717529 169,8333282471            150 92,1666641235
## 2 136,6666717529            157            175  99,8333358765 58,3333320618
## 3 104,8368453979 151,8578796387 158,3087768555 144,1842193604 86,1842041016
## 4 151,3883514404 169,9227142334 171,9880371094  83,4226989746 24,5842113495
## 5 147,4994354248  150,261138916 164,7403869629 102,7822799683 43,3062477112
## 6 183,2274017334 181,4293365479 223,4661560059   91,842086792 28,2040882111
##   geology        evi_1        evi_2        evi_3        evi_4        evi_5
## 1       3 0,3910871148 0,4112656713 0,4365606308 0,4635847807 0,4911327064
## 2       6 0,3378965855  0,381662339 0,3899267614 0,4462485611  0,493188709
## 3       6 0,3610805273  0,375120908 0,4241586626 0,4610377252 0,4872943163
## 4       5 0,3889489174 0,4088948369 0,4338231981 0,4899954498 0,4880128205
## 5       6 0,3528017104 0,3939922452 0,3909000456 0,4738618433 0,4881713688
## 6       3 0,3509942591 0,3660480976 0,4051557779 0,4467102587 0,4803439975
##          evi_6        evi_7        evi_8        evi_9       evi_10       evi_11
## 1 0,5538878441 0,5438943505 0,5508648753 0,5431039333 0,5202928782 0,5086370707
## 2 0,5008920431 0,5073742867 0,5237532258 0,5151152611 0,4978065789 0,4807990491
## 3 0,5046898723  0,507065475 0,5101108551 0,5113150477 0,5001526475 0,4540195763
## 4 0,5283816457 0,5550786257 0,5455245972 0,5536313057 0,5269128084 0,5338335633
## 5 0,4783726931 0,5415314436 0,5563444495 0,5323090553 0,5286363959 0,4923691452
## 6 0,5062088966 0,4861875474 0,5268591642 0,4800225496 0,4694581926 0,4842900038
##         evi_12       evi_13       evi_14       evi_15       evi_16       evi_17
## 1 0,4672316313 0,4851673841 0,4788505137 0,5039389729 0,5205193162 0,5576840639
## 2 0,4335366488  0,461047858 0,4410296082 0,4584658146 0,4493008852 0,4633820653
## 3  0,436607182  0,440245837 0,4437948167 0,4673160017 0,4729451239 0,5137551427
## 4 0,5167191625 0,5197646022 0,5073350668 0,4811374545 0,5135068297 0,4921848178
## 5 0,4729484022 0,4772336781  0,441865176 0,5021463037 0,5134362578  0,513906002
## 6 0,4731833935 0,4818029106 0,4438877702 0,4317378998 0,4680766761 0,5362793207
##         evi_18       evi_19       evi_20       evi_21       evi_22       evi_23
## 1 0,4989081919  0,506745398  0,489985764 0,4897886813  0,489938885 0,3840098083
## 2  0,455308795 0,4603865445 0,4533531964 0,4377965927 0,4394916892 0,3435108066
## 3 0,4976354837 0,4795286059 0,4982682168 0,4773576856 0,4482271075  0,398150295
## 4 0,5154051185 0,4748531878 0,5097850561 0,4463077486 0,4376213849 0,4012963176
## 5  0,501724422 0,4987980723 0,4892227352 0,4510650635 0,4174048901 0,3798615038
## 6  0,494964689 0,4127091169 0,4341135621 0,4097899199 0,3784380257 0,3613479733
##             lon          lat surface
## 1 16,0056502102 1,5014159492       5
## 2 17,0176638107 2,6258755053      15
## 3 16,4554340326 1,4264519788    17,5
## 4 15,2934924913  3,862781017    20,5
## 5 17,5424116035 2,7758034461    10,5
## 6 16,4554340326 3,6378891058      20
```

$M = I$

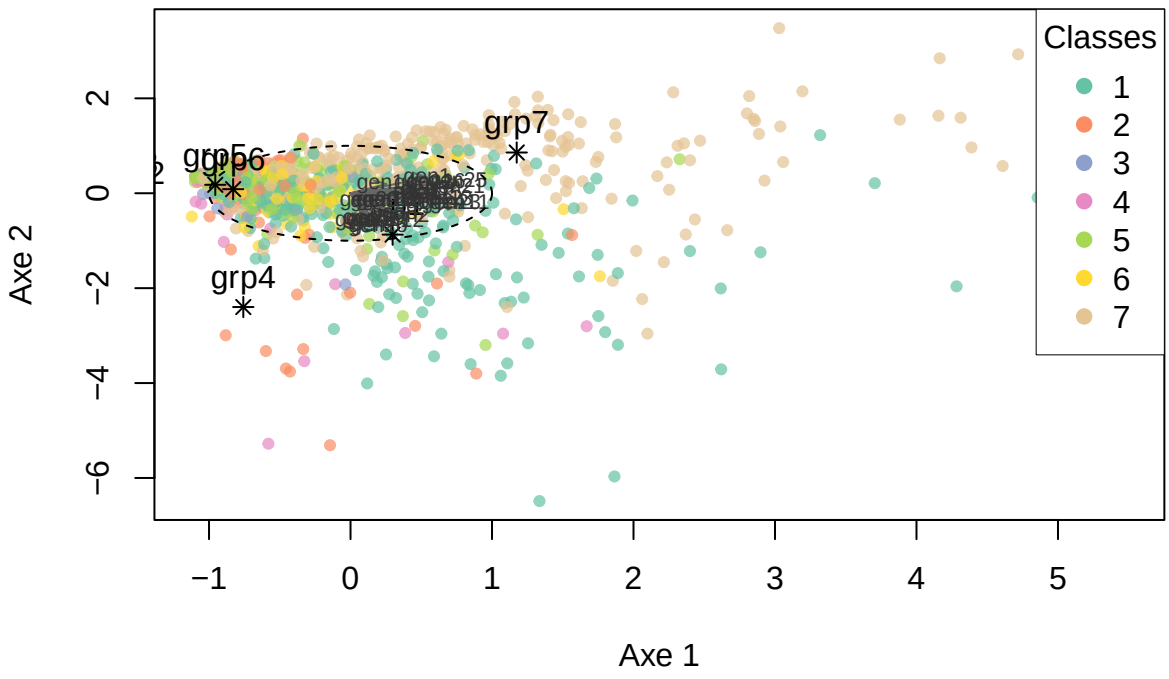
```
X <- scale(as.matrix(data[, paste0("gen", 1:27)]), center = TRUE, scale = TRUE)
grp <- as.factor(data$forest)
Y <- model.matrix(~ grp - 1)

#choosing M
M <- diag(1, ncol(X))
```

```r
#adls
out <- adpls_full(X, Y, H = 3, w = rep(1 / nrow(X), nrow(X)), M = M, center_scale = FALSE)
```

```r
data.frame(
  Axe = 1:3,
  S = round(out$S, 3),
  R2 = round(out$R2, 3),
  Discriminant = round(out$S * out$R2, 3)
)
```

```
##   Axe     S    R2 Discriminant
## 1   1 0.184 0.200         0.037
## 2   2 0.116 0.178         0.021
## 3   3 0.051 0.090         0.005
```

```r
plot_adpls(out, Y = Y, Y_labels = grp, h = 1, m = 2, show_variables = TRUE)
```



Experiment 1

$M = \text{diag}\left(\frac{1}{\text{var}(X_j)}\right)$

```r
# data
data <- read.table("Datagenus.csv", header = TRUE)
X <- as.matrix(data[, paste0("gen", 1:27)])
```

```r
grp <- as.factor(data$forest)
Y <- model.matrix(~ grp - 1)

# metrique M
variances <- apply(X, 2, var)
M <- diag(1 / variances)

# standartisation
X_scaled <- scale(X, center = TRUE, scale = TRUE)

# adpls
out_M <- adpls_full(X_scaled, Y, H = 3, M = M)

# S, R2, Discriminant
data.frame(
  Axe = 1:3,
  S = round(out_M$S, 3),
  R2 = round(out_M$R2, 3),
  Discriminant = round(out_M$S * out_M$R2, 3)
)
```

```
##   Axe     S    R2 Discriminant
## 1   1 0.017 0.181         0.003
## 2   2 0.009 0.130         0.001
## 3   3 0.008 0.051         0.000
```

Experiment 2

$M = D_X^{-1}$ ou $D_X = \text{diag}(\text{sd}(X_j)^2)$

```r
# data
data <- read.table("Datagenus.csv", header = TRUE)
X_raw <- as.matrix(data[, paste0("gen", 1:27)])  # raw data, not centered
grp <- as.factor(data$forest)
Y <- model.matrix(~ grp - 1)

# metrique
vars <- apply(X_raw, 2, var)
M <- diag(1 / vars)

out <- adpls_full(X_raw, Y, H = 3, M = M, center_scale = FALSE)

data.frame(
  Axe = 1:3,
  S = round(out$S, 3),
  R2 = round(out$R2, 3),
  Discriminant = round(out$S * out$R2, 3)
)
```

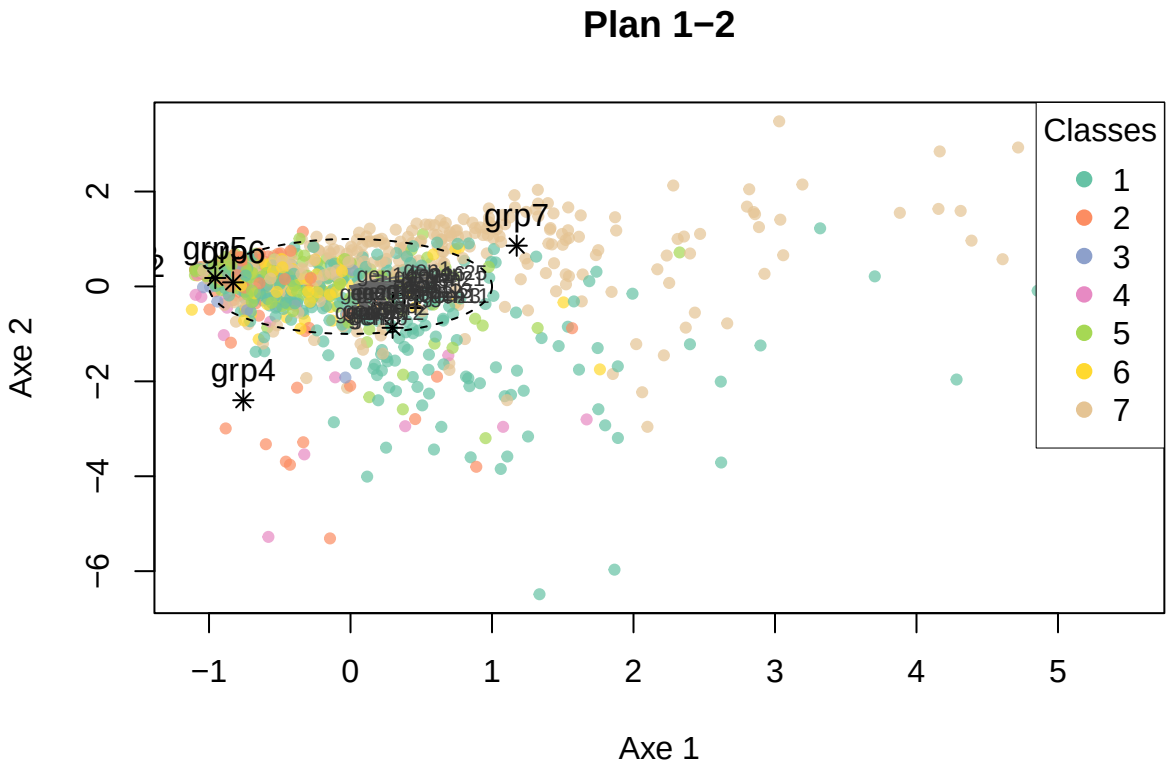
```
##   Axe     S    R2 Discriminant
## 1   1 0.002 0.668         0.001
## 2   2 0.000 0.224         0.000
## 3   3 0.000 0.134         0.000
```

q2

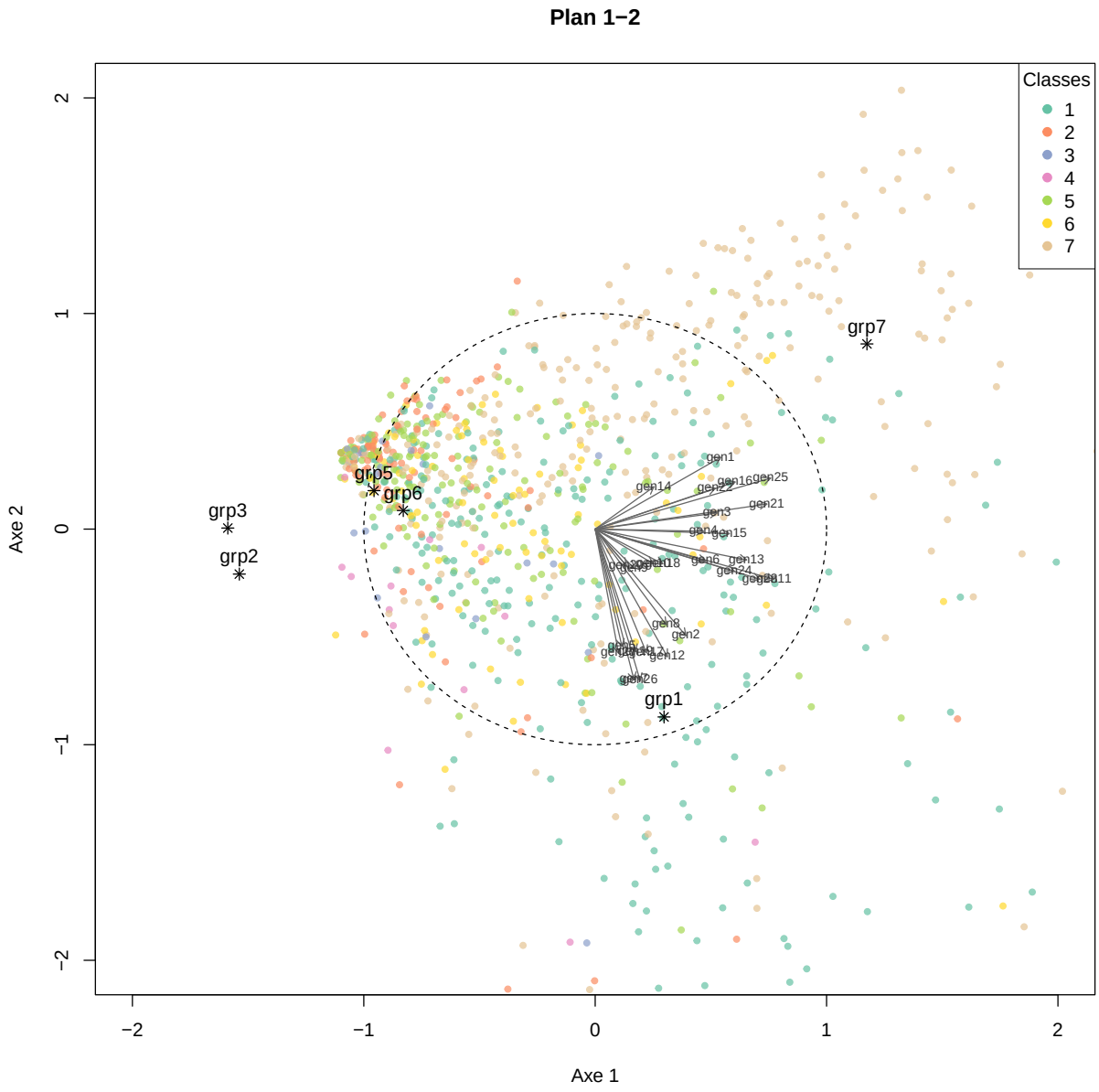
```r
data <- read.table("Datagenus.csv", header = TRUE)

X <- scale(as.matrix(data[, paste0("gen", 1:27)]), center = TRUE, scale = TRUE)
grp <- as.factor(data$forest)
Y <- model.matrix(~ grp - 1)

#choosing M
M <- diag(1, ncol(X))

#adls
out <- adpls_full(X, Y, H = 3, w = rep(1 / nrow(X), nrow(X)), M = M, center_scale = FALSE)

data.frame(
  Axe = 1:3,
  S = round(out$S, 3),
  R2 = round(out$R2, 3),
  Discriminant = round(out$S * out$R2, 3)
)
```

```
##   Axe     S    R2 Discriminant
## 1   1 0.184 0.200        0.037
## 2   2 0.116 0.178        0.021
## 3   3 0.051 0.090        0.005
```

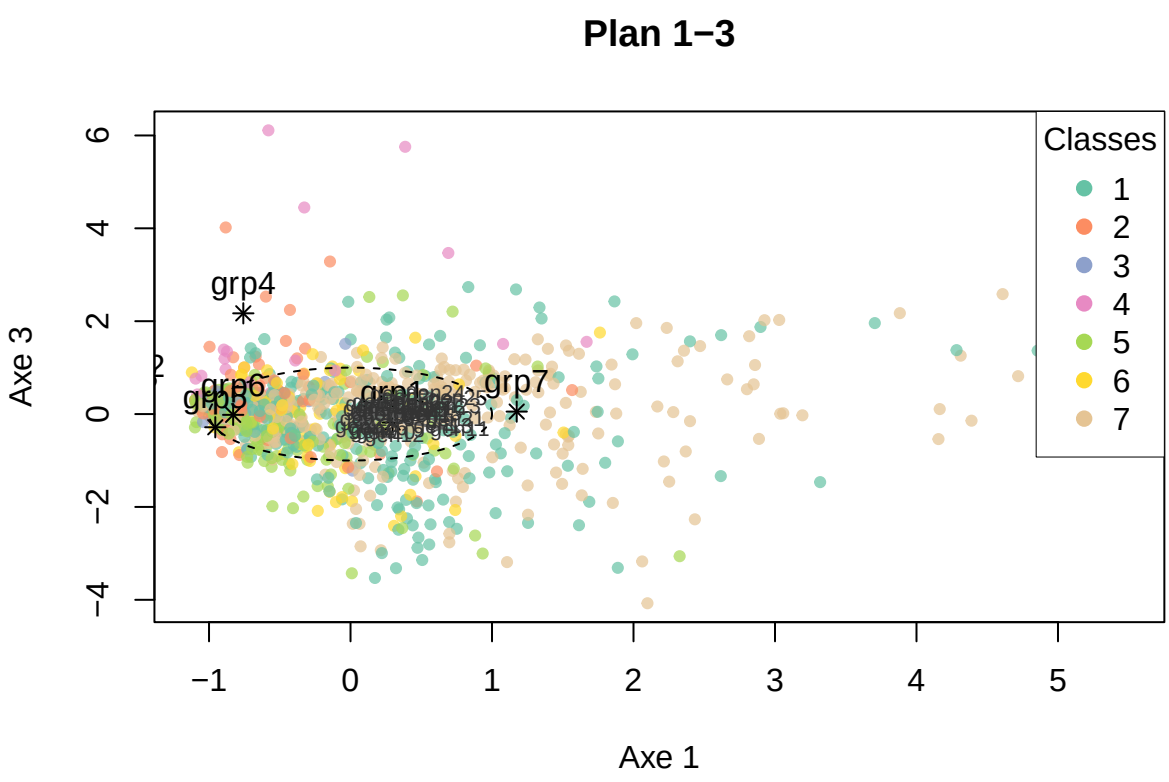
```r
plot12 <- plot_adpls(out, Y = Y, Y_labels = grp, h = 1, m = 2, show_variables = TRUE, main = "Plan 1-2"
```



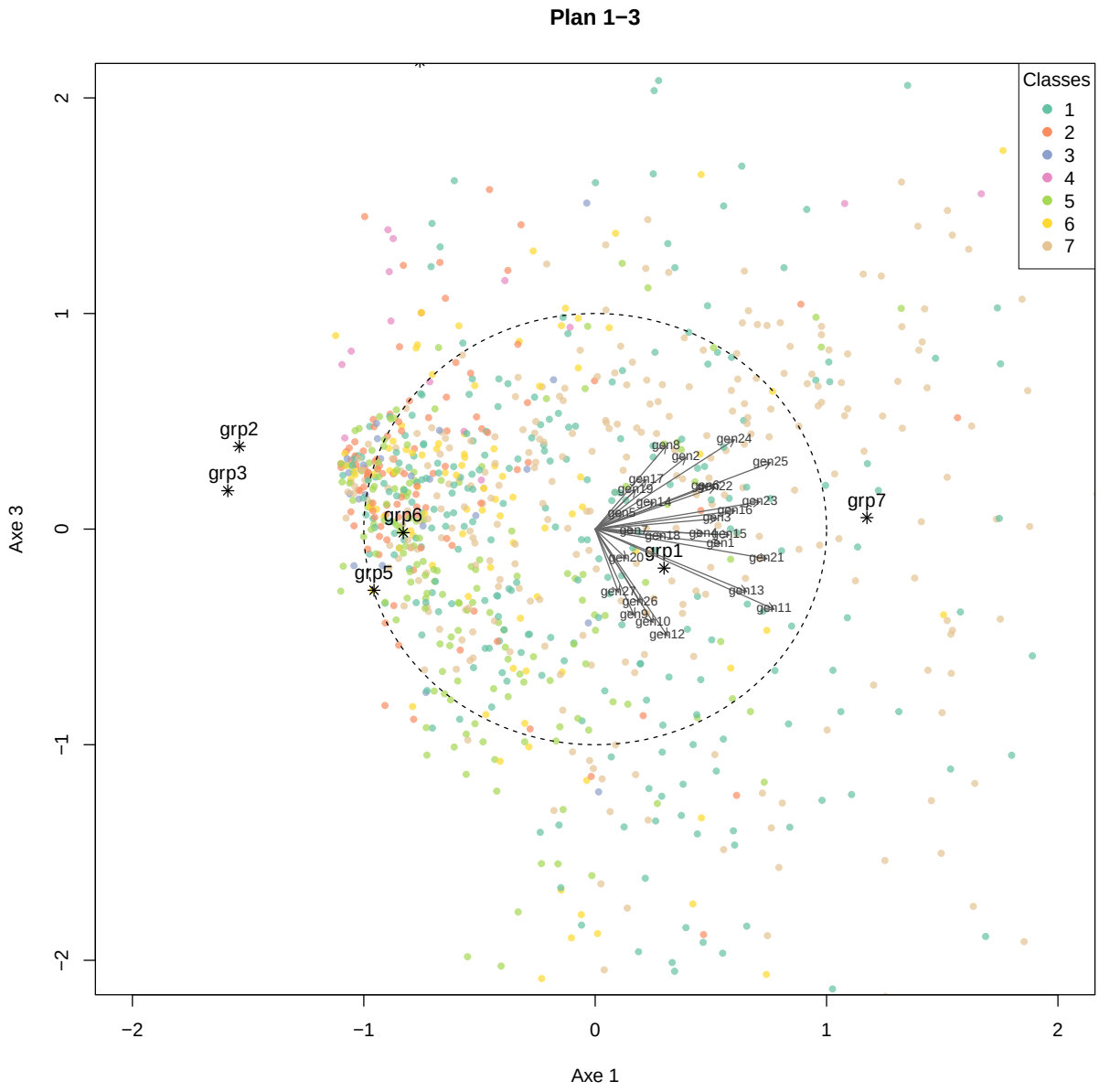

```
plot_adpls(
  out,
  Y = Y,
  Y_labels = grp,
  h = 1,
  m = 2,
  show_variables = TRUE,
  main = "Plan 1-2",
  xlim = c(-2, 2),
  ylim = c(-2, 2)
)
```



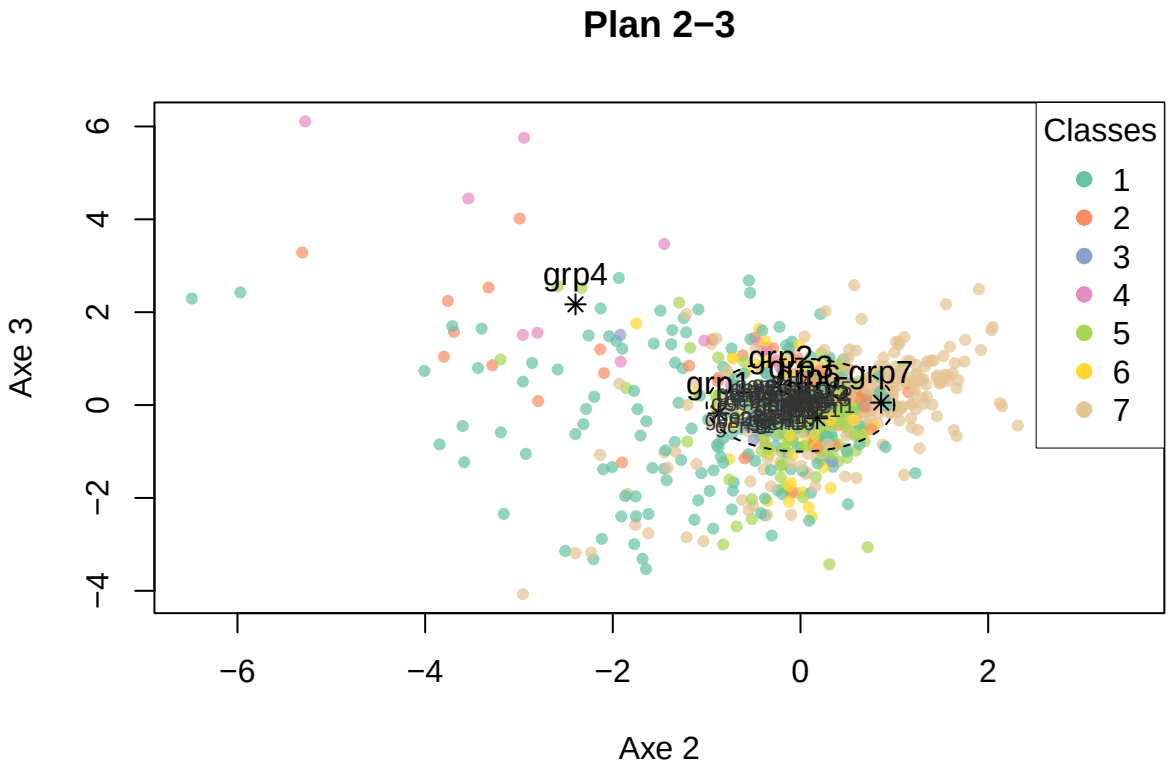

```
plot13 <- plot_adpls(out, Y = Y, Y_labels = grp, h = 1, m = 3, show_variables = TRUE, main = "Plan 1-3"
```



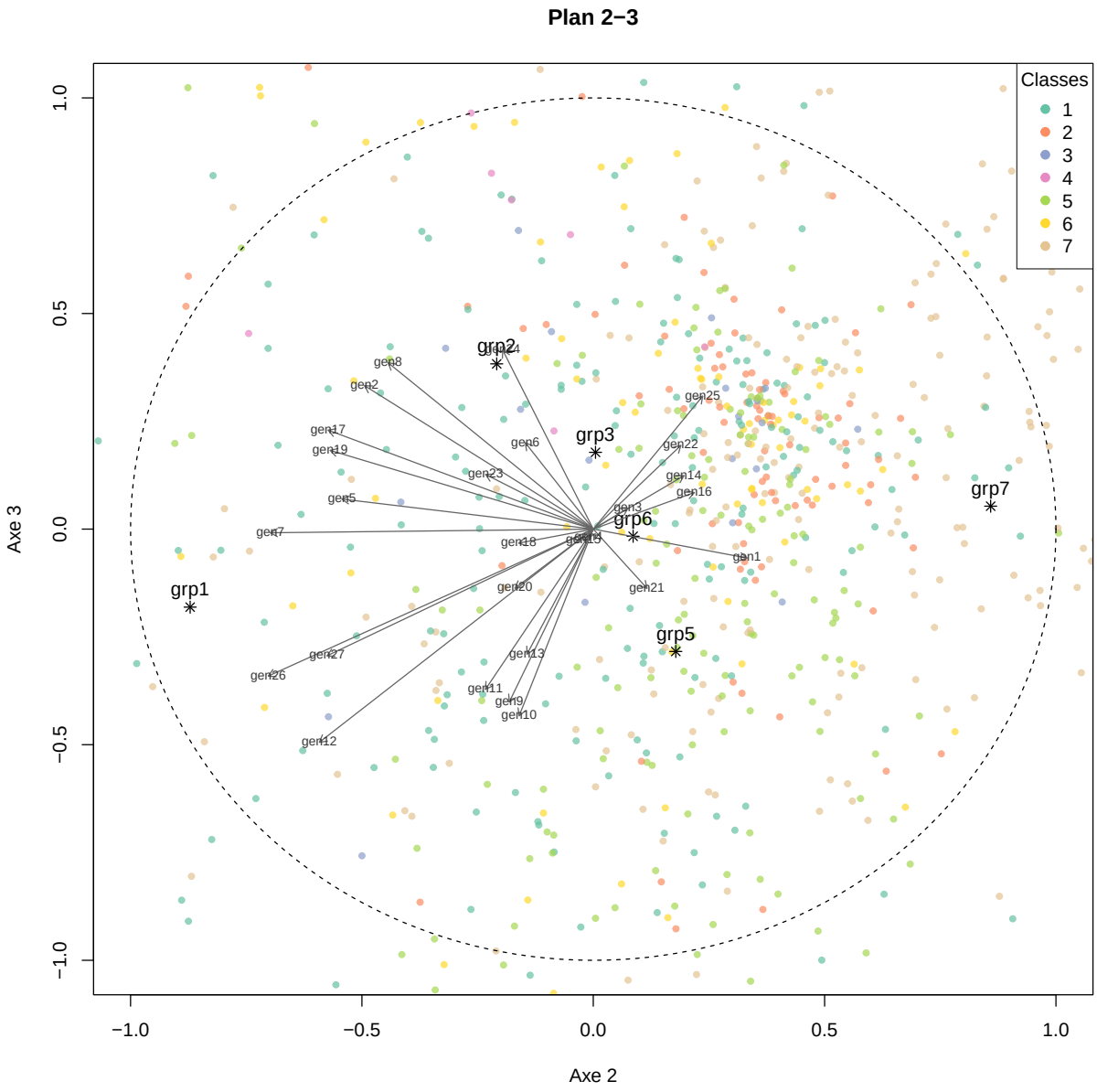

```
plot_adpls(
  out,
  Y = Y,
  Y_labels = grp,
  h = 1,
  m = 3,
  show_variables = TRUE,
  main = "Plan 1-3",
  xlim = c(-2, 2),
  ylim = c(-2, 2)
)
```




```
plot23 <- plot_adpls(out, Y = Y, Y_labels = grp, h = 2, m = 3, show_variables = TRUE, main = "Plan 2-3"
```




```
plot_adpls(
  out,
  Y = Y,
  Y_labels = grp,
  h = 2,
  m = 3,
  show_variables = TRUE,
  main = "Plan 2-3",
  xlim = c(-1, 1),
  ylim = c(-1, 1)
)
```


Plan 2−3
Classes
1
2
3
4
5
6
7
grp1
grp2
grp3
grp4
grp5
grp6
grp7
gen1
gen2
gen3
gen4
gen5
gen6
gen7
gen8
gen9
gen10
gen11
gen12
gen13
gen14
gen16
gen17
gen18
gen19
gen20
gen21
gen22
gen23
gen25
gen26
gen27
Axe 2
Axe 3
−1.0
−0.5
0.0
0.5
1.0