TD3 - Implémentation de l'ACP

Courtenay Rebecca & Ducros Chloé & Lasson Marie

Contents

1 Question préliminaires (bonus, 3 points)

a. Montrer que pour toutes variables aléatoires X et Y, on a :

$$\mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])] = \mathrm{E}[XY] - \mathrm{E}[X]\mathrm{E}[Y]$$

On a :
$\mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])]$
$= \mathrm{E}[XY] - \mathrm{E}[X]E[Y] - \mathrm{E}[X]E[Y] + \mathrm{E}[X]E[Y]$
$= \mathrm{E}[XY] - E[X]E[Y]$

b. Soit C la matrice de covariance d'une matrice de données X. Montrer que la matrice C est symétrique.

Les éléments diagonaux de C représentent les variances des données de X. De plus, les autres éléments décrivent la covariance entre X_i et X_j avec i $\neq$ j. Et comme, $cov(X_i, X_j) = cov(X_j, X_i)$ alors, la matrice C est symétrique.

c. Montrer que la matrice C est semi-définie positive. Rappel : la matrice symétrique C est semi-définie positive ssi $\forall x \in \mathbb{R}^p,\ x^T C x \geq 0$

On sait que : $x^T C x = \sum_{i,j} x_i Cov(X_i, X_j) x_j = \sum_{i,j} Cov(x_i X_i, x_j X_j) = Var(x_1 X_1 + ... + x_n X_n) \geq 0$.
Donc la matrice C est semi-définie positive.

d. Montrer que pour toute matrice C, il y a équivalence entre : "C est semi-définie positive" et "les valeurs propres de C sont toutes positives ou nulles".

On suppose que C est une matrice semi-définie positive et λ une de ses valeurs propres.
D'après la question c, $x^T C x \geq 0$, donc $\lambda(X^T X) \geq 0 \Rightarrow \lambda\ |X|^2 \geq 0 \Rightarrow \lambda \geq 0$
On peut donc conclure que : les valeurs propres de C sont toutes positives ou nulles.

2 Implémentation de l'ACP (20 points)

L'implémentation de toutes les fonctions de base demandées sera notée sur 10 points. Tous les apports et améliorations (cf question5) seront notés sur 10 points.

2.1 Première étape : calcul de l'ACP

On utilisera le jeu de données de décathlon :

```
decathlon <- read.table(file = "AnaDo_JeuDonnees_Decathlon.csv",
                        header=TRUE,sep=";",dec=".", row.names=1,
                        check.names=FALSE, fileEncoding="latin1")
```

On pourra donc s'appuyer sur le fichier pdf "Commandes utiles" fourni dans le TD1 pour comparer nos résultats. Vos graphes seront sûrement le symétrique de ceux affichés par le package FactoMineR, vous pouvez y remédier en multipliant votre matrice de données par (-1) mais ce n'est pas obligatoire.

Pour tracer les cercles des corrélations vous aurez sûrement besoin du package `plotrix` qui permet de dessiner des cercles, que vous pouvez installer via la commande `install.packages("plotrix")`.

```
library(plotrix)
```

De la même façon que dans le TD1, on effectuait la commande `res <- PCA(fertilite[,1:6])`, on veut une fonction qui prend en entrée le tableau de données et qui effectue l'ACP.

Dans un premier temps, on ne tiendra pas compte des variables qualitatives supplémentaires, on les traitera plus tard.

En notant X notre matrice de données (sans les variables descriptives supplémentaires), l'ACP consiste à trouver une matrice Y contenant les mêmes données mais dans une base orthogonale différente maximisant l'explicabilité de la variance. Les étapes du calcul de l'ACP sont les suivantes :

- Centrer et réduire X

```
X <- decathlon[,c(1:10)]
ctr <- scale(X)
```

- Calcul de la matrice de covariance C de X

```
C <- cov(ctr)
```

- Calcul des vecteurs propres et valeurs propres de C

```
VAP_VEP <- function(X){
  data <- cov(X)
  vap <- eigen(data)$values
  vep <- eigen(data)$vectors

  return(list(vap, vep))
}
```

- Tri des vecteurs propres et valeurs propres par ordre décroissant de l'importance des valeurs propres

La fonction eigen permet de trier les valeurs. Nous venons de le faire dans la fonction VAP_VEP donc nous n'avons pas besoin d'appliquer la fonction sort/order (qui fait exactement la même chose). Il nous suffit d'appliquer : VAP_VEP(X)[[1]] pour avoir les valeurs propres triées et VAP_VEP(X)[[2]] pour avoir les vecteurs propres triés.

Ecriture des matrices V et L

```
L<-diag(VAP_VEP(X)[[1]]) #matrice diagonale des valeurs propres
V<-VAP_VEP(X)[[2]] #matrice des vecteurs propres
```

- Ecriture des données dans la base orthonormée des vecteurs propres, ce qui correspond à une projection orthogonale de nos données sur la matrice W des vecteurs propres, ce qui se fait avec le produit matriciel $X \cdot W$. Plus d'explications ci-dessous.

Nous affichons que les 6 premières données.

```
W <- as.matrix(VAP_VEP(X)[[1]],VAP_VEP(X)[[2]])
X2<-as.matrix(X)
head(X2%*%W)
```

```
##               [,1]
## Sebrle    1875.137
## Clay      1810.327
## Karpov    1820.948
## Macey     1865.555
## Warners   1821.186
## Zsivoczky 1856.528
```

1. Ecrire la fonction `ACP` qui prend en entrée un tableau de données et qui retourne l'ensemble des valeurs propres (dans une variable `$vap`) et des vecteurs propres (dans une variable `$vep`) de la matrice de covariance, le tout trié par ordre décroissant. Dans une variable `$data` on stockera également les données centrées réduites. Pour la tester, on pourra faire l'appel suivant : `ACP(decathlon[,1:10])`

```
ACP <- function(X){
  data <- scale(X)
  cov <- cov(data)
  vap <- eigen(cov)$values
  vep <- eigen(cov)$vectors

  return(list("vap"=vap,"vep"=vep,"data"=data,"cov"=cov))
}
```

Valeurs propres :

```
ACP(X)$vap
```

```
##  [1] 3.2719055 1.7371310 1.4049167 1.0568504 0.6847735 0.5992687 0.4512353
##  [8] 0.3968766 0.2148149 0.1822275
```

Vecteurs propres : Nous affichons que les 6 premières données.

```
head(ACP(X)$vep)
```

```
##            [,1]       [,2]        [,3]        [,4]        [,5]        [,6]
## [1,]  0.4282963  0.1419891  0.15557953  0.03678703 -0.36518741  0.29607739
## [2,] -0.4101520 -0.2620794 -0.15372674 -0.09901016 -0.04432336 -0.30612478
## [3,] -0.3441444  0.4539470  0.01972378 -0.18539458 -0.13431954  0.30547299
## [4,] -0.3161944  0.2657761  0.21894349  0.13189684 -0.67121760 -0.46777116
## [5,]  0.3757157  0.4320460 -0.11091758 -0.02850297  0.10597034 -0.33252178
## [6,]  0.4125544  0.1735910  0.07815576 -0.28290068 -0.19857266 -0.09963776
##             [,7]        [,8]       [,9]       [,10]
## [1,] -0.38177608  0.46160211  0.1047577  0.42428269
## [2,] -0.62769317 -0.02101165  0.4826691  0.08104448
## [3,]  0.30972542 -0.31393005  0.4272908  0.39028424
## [4,]  0.09145002  0.12509166 -0.2436605 -0.10642724
## [5,]  0.12442114  0.21339819  0.5521294 -0.41399532
## [6,] -0.35733030 -0.71111429 -0.1501343 -0.09086448
```

Données centrée-réduite :

Nous affichons que les 6 premières données.

```
head(ACP(X)$data)
```

```
##                 100m   Longueur       Poids     Hauteur       400m      110m H
```

```
## Sebrle    -0.5628739  1.8331131 2.283919585  1.60955474 -1.0892025 -1.17818270
## Clay      -2.1216730  2.2123779 0.913271989  0.93502243 -0.3696226 -1.00861538
## Karpov    -1.8935560  1.7382969 1.762345721  1.27228858 -2.4329962 -1.34775002
## Macey     -0.4107959  0.6637134 1.519753226  1.94682089 -0.5603546 -0.09719103
## Warners   -1.4373221  1.5170591 0.003550134 -0.07677604 -1.4273183 -1.26296636
## Zsivoczky -0.3347570 -0.3792648 1.010308987  1.60955474 -0.1875602  0.72944966
##               Disque     Perche     Javelot       1500m
## Sebrle     1.3009450  0.8545364  2.52825135  0.08439142
## Clay       1.7124500  0.4948240  2.36043901  0.25486670
## Karpov     2.1683620 -0.5843134 -0.57524110 -0.07837391
## Macey      1.1884472 -1.3037383  0.02971203 -1.16547502
## Warners   -0.1763283  0.4948240 -0.60631746 -0.08351387
## Zsivoczky  0.3832000 -0.2246009  1.06351893 -0.81253124
```

Matrice de covariance : Nous affichons que les 6 premières données.

```
head(ACP(X)$cov)
```

```
##                100m   Longueur      Poids    Hauteur       400m     110m H
## 100m      1.0000000 -0.5986777 -0.3564823 -0.2462529  0.5202982  0.5798889
## Longueur -0.5986777  1.0000000  0.1833044  0.2946444 -0.6020626 -0.5054101
## Poids    -0.3564823  0.1833044  1.0000000  0.4892115 -0.1384329 -0.2516157
## Hauteur  -0.2462529  0.2946444  0.4892115  1.0000000 -0.1879569 -0.2832891
## 400m      0.5202982 -0.6020626 -0.1384329 -0.1879569  1.0000000  0.5479878
## 110m H    0.5798889 -0.5054101 -0.2516157 -0.2832891  0.5479878  1.0000000
##               Disque       Perche      Javelot       1500m
## 100m     -0.2217076 -0.082536834 -0.157746452 -0.06054645
## Longueur  0.1943101  0.204014112  0.119758933 -0.03368613
## Poids     0.6157681  0.061181853  0.374955509  0.11580306
## Hauteur   0.3692183 -0.156180742  0.171880092 -0.04490252
## 400m     -0.1178794 -0.079292469  0.004232096  0.40810643
## 110m H   -0.3262010 -0.002703885  0.008743251  0.03754024
```

2.2 Deuxième étape : affichage de l'ACP pour les individus

2. Ecrire la fonction `individusACP` qui trace l'ACP des individus sur les axes sélectionnés. Elle aura donc en entrée un argument "axes" sur lequel tracer les données. On pourra la tester sur les deux premières dimensions avec l'appel : `individusACP(axes=1:2)`

La visualisation de cette fonction se trouve dans la 5ème partie avec une version améliorée.

```
individusACP <- function(axes, PCA) {
  V <- as.matrix(ACP(PCA)$vep)
  D <- as.matrix(ACP(PCA)$data)
  res <- D%*%V

  dim1 <- paste("Dim",as.character(axes[1]))
  dim2 <- paste("Dim",as.character(axes[2]))

  par(mar=c(4,4,4,4),pty = "s")
  plot(res[,axes], xlab = dim1, ylab = dim2, pch = 1, col="red")
  text(res[,axes], labels = row.names(ACP(PCA)$data), cex = 0.6, pos = 1)
  abline(v = 0, lty = 2)
  abline(h = 0, lty = 2)
  title("ACP")
  return(res)
```

```
}
```

2.3 Troisième étape : fonction summary

3. Ecrire la fonction `summary`, permettant de retourner les informations principales de l'ACP. Il est conseillé (mais pas obligatoire) d'écrire plusieurs sous-fonctions pour composer la fonction `summary`.

La visualisation de cette fonction se trouve dans la 5ème partie avec une version améliorée.

```
summary <- function(PCA) {

  #Eigenvalues
  vap <- ACP(PCA)$vap
  vect <- c()
  for (i in seq.int(0,length(vap))) {
    somme <- sum(vap)
    pourcentage <- (vap[i]/somme)*100
    vect <- append(vect, pourcentage)
  }
  eigenvalues <- cbind(vap, vect)
  colnames(eigenvalues) <- c("Variance", "% of var.")
  print("Eigenvalues")
  print(head(eigenvalues))

  #Individuals
  dim <- individusACP(axes=1:2, PCA)
  graphics.off()

    #Dim1
  contr_1 <- c()
  sommecarre_1 <- dim[,1]%*%dim[,1]
  for (i in seq.int(0,length(dim[,1]))) {
    contr_1 <- append(contr_1, ((dim[,1][i])**2/sommecarre_1)*100)
  }
  individuals <- cbind(dim[,1], contr_1)

  cos2_1 <- c()
  for (i in seq.int(0,length(dim[,1]))) {
    cos2_1 <- append(cos2_1, dim[i,1]**2/sqrt(sum(dim[i,]**2))**2)
  }
  individuals <- cbind(individuals, cos2_1)

    #Dim2
  contr_2 <- c()
  sommecarre_2 <- dim[,2]%*%dim[,2]
  individuals <- cbind(individuals, dim[,2])

  for (i in seq.int(0,length(dim[,2]))) {
    contr_2 <- append(contr_2, ((dim[,2][i])**2/sommecarre_2)*100)
  }
  individuals <- cbind(individuals, contr_2)

  cos2_2 <- c()
  for (i in seq.int(0,length(dim[,2]))) {
```

```
  cos2_2 <- append(cos2_2, dim[i,2]**2/sqrt(sum(dim[i,]**2))**2)
}
individuals <- cbind(individuals, cos2_2)

colnames(individuals) <- c("Dim 1","ctr", "cos2", "Dim 2","ctr", "cos2")

print("Individuals")
print(head(individuals))

#Variables
  #Dim1
corr_1 <- c()
for (i in seq.int(0, length(vap))) {
  corr_1 <- append(corr_1, sqrt(vap[1])*ACP(PCA)$vep[i,1])
}
variables <- as.data.frame(corr_1)

cos2_1 <- c()
for (i in seq.int(0, length(corr_1))) {
  cos2_1 <- append(cos2_1, corr_1[i]**2)
}
variables <- cbind(variables, cos2_1)

contr_1 <- c()
sumSquare1 <- ACP(PCA)$vep[,1]%*%ACP(PCA)$vep[,1]
for (i in seq.int(0,length(ACP(PCA)$vep[,1]))) {
  contr_1 <- append(contr_1, (ACP(PCA)$vep[i,1]**2/sumSquare1)*100)
}
variables <- cbind(variables, contr_1)

  #Dim2
corr_2 <- c()
for (i in seq.int(0, length(vap))) {
  corr_2 <- append(corr_2, sqrt(vap[2])*ACP(PCA)$vep[i,2])
}
variables <- cbind(variables, corr_2)

cos2_2 <- c()
for (i in seq.int(0, length(corr_2))) {
  cos2_2 <- append(cos2_2, corr_2[i]**2)
}
variables <- cbind(variables, cos2_2)

contr_2 <- c()
sumSquare2 <- ACP(PCA)$vep[,2]%*%ACP(PCA)$vep[,2]
for (i in seq.int(0,length(ACP(PCA)$vep[,2]))) {
  contr_2 <- append(contr_2, (ACP(PCA)$vep[i,2]**2/sumSquare2)*100)
}
variables <- cbind(variables, contr_2)

nom_Var <- colnames(ACP(PCA)$data)
row.names(variables) <- nom_Var
colnames(variables) <- c("Corr 1", "Cos2 1", "Ctr 1","Corr 2", "Cos2 2", "Ctr 2")
```

```
  print("Variables")
  print(head(variables))
}
```

2.4 Quatrième étape : affichage de l'ACP pour les variables

4. Ecrire la fonction `variablesACP` qui trace le cercle de corrélation des variables sur les axes sélectionnés. Elle aura donc en entrée un argument "axes" sur lequel tracer les données. On pourra la tester sur les deux premières dimensions avec l'appel : `variablesACP(axes=1:2)`. Il ne sera pas obligatoire de dessiner des flèches (comme dans le package FactoMineR).

La visualisation de cette fonction se trouve dans la 5ème partie avec une version améliorée.

```
variablesACP <- function(axes, PCA) {

  vep <- ACP(PCA)$vep
  vap <- ACP(PCA)$vap
  cor_v <- c()
  data <- ACP(PCA)$data
  taille_v <- length(vep[,1])

  for (k in seq.int(1, taille_v)) {
    cor_dv <- c()
    for (i in seq.int(0, length(vap))) {
      cor_dv <- append(cor_dv, sqrt(vap[k])*vep[i,k])
    }
    cor_v <- cbind(cor_v, cor_dv)
  }

  dim1 <- paste("Dim",as.character(axes[1]))
  dim2 <- paste("Dim",as.character(axes[2]))

  par(mar=c(4,4,4,4),pty = "s")
  plot(cor_v[,axes],xlim = c(-1,1),ylim = c(-1,1),xlab=dim1,ylab=dim2)
  text(cor_v[,axes], labels = colnames(ACP(PCA)$data), cex = 0.6, pos = 1)
  abline(v = 0, lty = 2)
  abline(h = 0, lty = 2)
  title("Graphique des variables")
  draw.circle(0, 0, 1, nv = 500, border = NULL, col = NA, lty = 1, lwd = 1)
}
```

2.5 Dernière étape : amélioration des fonctions

5. En vous inspirant des fonctions du package FactoMineR utilisées lors du TD1, améliorer toutes les fonctions écrites ci-dessus pour y inclure des options supplémentaires. Par exemple, voici une liste (non exhaustive) des choses qu'il est possible d'améliorer

En annexe se trouvent les fonctions intermédiaires que nous avons créées avant d'avoir ces versions finales.

2.5.1 PCA

- Dans la fonction ACP, ajouter deux paramètres supplémentaires : `quanti.sup` et `quali.sup`, qui indiqueront des colonnes. Si ces paramètres ne sont pas indiqués par l'utilisateur, alors l'ACP se comporte de la même façon que la fonction précédemment écrite. S'ils sont entrés par l'utilisateur, ils indiquent les colonnes des variables quantitatives ou qualitatives supplémentaires. Par exemple, la

fonction répondra à l'appel : `PCA(decathlon, quanti.sup=11:12, quali.sup=13)`. Bien sûr, il est possible de rentrer un seul des deux arguments (comme pour le jeu de donnée de fertilité en Europe)

- Ajouter un paramètre en entrée de l'ACP pour savoir si l'on souhaite centrer et réduire les données. En effet, cela n'est pas toujours souhaitable (notamment quand on a des informations avec beaucoup de bruit, car le fait de normaliser les données va mettre le bruit à la même valeur que les autres données)
- Effectuer les calculs à chaque nouvel appel de `summarize` ou des fonctions graphiques n'est pas optimal (redondant). Le mieux serait, dès l'appel à la fonction `ACP`, de faire tous les calculs nécessaires et de les stocker dans des variables, qui seraient ensuite simplement retournées ou utilisées lors de l'appel des fonctions

```
ACP_FINAL <- function(X, quanti.sup=NULL, quali.sup=NULL, cr = TRUE) {

    #quanti.sub & quali.sub
  supp <- c()
  if (!is.null(quanti.sup)){
    for (i in quanti.sup) {
      supp <- append(supp, i)
    }
  }
  if (!is.null(quali.sup)) {
    for (i in quali.sup) {
      supp <- append(supp, i)
    }
  }
  if (!is.null(supp)) {
    supp <- sort(supp)
    X <- X[,-supp]
  }

    #centree-reduite
  if (cr == FALSE) {
    data <- X
  }
  else {
    data <- scale(X)
  }

    #covariance
  cov <- cov(data)

    #valeurs/vecteurs propres
  vap <- eigen(cov)$values
  vep <- eigen(cov)$vectors

    #Eigenvalues
  vect <- c()
  for (i in seq.int(0,length(vap))) {
    somme <- sum(vap)
    pourcentage <- (vap[i]/somme)*100
    vect <- append(vect, pourcentage)
  }
  cumul_var <- cumsum(vect)
  eigenvalues <- cbind(vap, vect, cumul_var)
  colnames(eigenvalues) <- c("Variance", "% of var.", "Cumulative % of var")
```

```
  #Individus
    #Cor
V <- as.matrix(vep)
D <- as.matrix(data)
cor_i <- D%*%V
    #Contribution
contr_i <- c()
for (k in seq.int(1, length(cor_i[1,]))) {
  sommecarre <- cor_i[,k]%*%cor_i[,k]
  contr_di <- c()
  for (i in seq.int(0,length(cor_i[,k]))) {
    contr_di <- append(contr_di, ((cor_i[,k][i])**2/sommecarre)*100)
  }
  contr_i <- cbind(contr_i, contr_di)
}
    #Cos2
cos2_i <- c()
for (k in seq.int(1, length(cor_i[1,]))) {
  cos2_di <- c()
  for (i in seq.int(0,length(cor_i[,k]))) {
    cos2_di <- append(cos2_di, cor_i[i,k]**2/sqrt(sum(cor_i[i,]**2))**2)
  }
  cos2_i <- cbind(cos2_i, cos2_di)
}

  #Variables
taille_v <- length(vep[,1])
    #Cor
cor_v <- c()
for (k in seq.int(1, taille_v)) {
  cor_dv <- c()
  for (i in seq.int(0, length(vap))) {
    cor_dv <- append(cor_dv, sqrt(vap[k])*vep[i,k])
  }
  cor_v <- cbind(cor_v, cor_dv)
}

  #Cos2
cos2_v <- c()
for (k in seq.int(1, taille_v)) {
  cos2_dv <- c()
  for (i in seq.int(0, length(cor_v[k,]))) {
    cos2_dv <- append(cos2_dv, cor_v[k,i]**2)
  }
  cos2_v <- cbind(cos2_v, cos2_dv)
}
  #Contribution
contr_v <- c()
for (k in seq.int(1, taille_v)) {
  contr_dv <- c()
  sumSquare <- vep[,k]%*%vep[,k]
  for (i in seq.int(0,length(vep[k,]))) {
    contr_dv <- append(contr_dv, (vep[i,k]**2/sumSquare)*100)
```

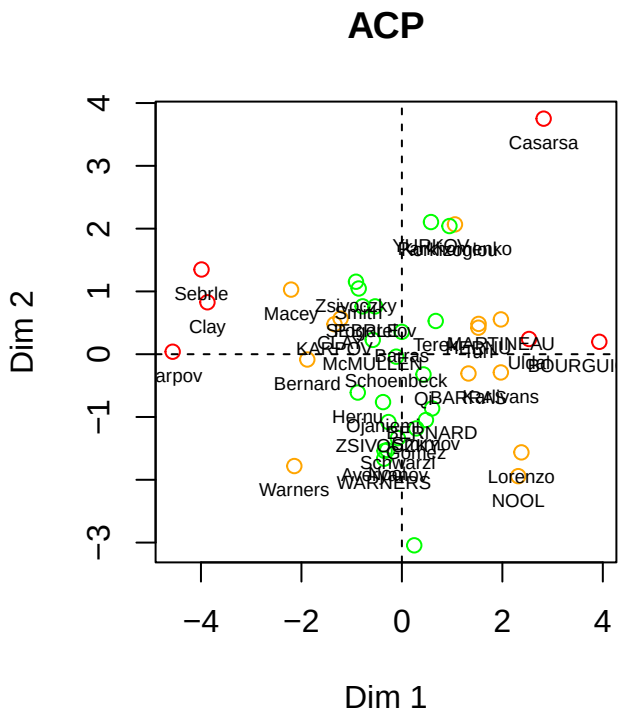
```
    }
    contr_v <- cbind(contr_v, contr_dv)
  }


  return(list("vap"=vap,"vep"=vep,"data"=data,"cov"=cov,"eigenvalues" = eigenvalues,
              "cor_ind" = cor_i, "contr_ind" = contr_i, "cos2_ind" = cos2_i,
              "cor_var" = cor_v, "cos2_var" = cos2_v, "contr_var" = contr_v))
}
```

2.5.2 Individus

- Dans la fonction `individusACP` on pourra alors ajouter un argument "habillage" qui colorie les individus selon leur modalité (ie selon la colonne indiquée par l'argument)

Dans ce cas, nous avons prédéfini les intervalles de couleurs nous même : il y a trois intervalles de couleur allant du vert au rouge en passant par l'orange. Plus nous nous approchons du vert plus l'individu, avec la dimension indiqué dans l'argument habillage, est significatif. A contrario de la couleur rouge.

```
individusACP_FINAL <- function(axes, PCA, habillage) {
  res <- ACP_FINAL(PCA)$cor_ind
  colour <- c()

  for (i in seq.int(1:nrow(res))) {
    if (abs(res[i,habillage]) < 1) {
      clr <- "green"
    }
    else if (abs(res[i,habillage]) < 2.5) {
      clr <- "orange"
    }
    else {
      clr <- "red"
    }
    colour <- append(colour, clr)
  }

  dim1 <- paste("Dim",as.character(axes[1]))
  dim2 <- paste("Dim",as.character(axes[2]))
  par(mar=c(4,4,4,4),pty = "s")
  plot(res[,axes], xlab = dim1, ylab = dim2, pch = 1, col = colour )
  text(res[,axes], labels = row.names(ACP_FINAL(X)$data), cex = 0.6, pos = 1)
  abline(v = 0, lty = 2)
  abline(h = 0, lty = 2)
  title("ACP")
}
individusACP_FINAL(axes=1:2, PCA=X, habillage=1)
```



2.5.3 Summary

- Dans la fonction **summary**, ajouter un paramètre "nbelements" (par défaut égal à 10) qui détermine le nombre d'individus qui doivent être affichés
- Dans la fonction **summary**, pour les eigenvalues, afficher en plus leur pourcentage cumulatif de variance

```
summary_FINAL <- function(PCA, nbelements=10) {

  #Eigenvalues
  print("Eigenvalues")
  print(head(ACP_FINAL(PCA)$eigenvalues))

  #Individuals

  individuals <- cbind(
    ACP_FINAL(PCA)$cor_ind[,1], ACP_FINAL(PCA)$contr_ind[,1],
    ACP_FINAL(PCA)$cos2_ind[,1],ACP_FINAL(PCA)$cor_ind[,2],
    ACP_FINAL(PCA)$contr_ind[,2], ACP_FINAL(PCA)$cos2_ind[,2])


  colnames(individuals) <- c("Dim 1","ctr", "cos2", "Dim 2","ctr", "cos2")

  print("Individuals")
  print(individuals[1:nbelements,])

  #Variables
  variables <- cbind(ACP_FINAL(PCA)$cor_var[,1], ACP_FINAL(PCA)$contr_var[,1],
                     ACP_FINAL(PCA)$cos2_var[,1],ACP_FINAL(PCA)$cor_var[,2],
                     ACP_FINAL(PCA)$contr_var[,2], ACP_FINAL(PCA)$cos2_var[,2])
```

```
  nom_Var <- colnames(ACP_FINAL(PCA)$data)
  row.names(variables) <- nom_Var
  colnames(variables) <- c("Corr 1", "Cos2 1", "Ctr 1","Corr 2", "Cos2 2", "Ctr 2")

  print("Variables")
  print(head(variables))
}
summary_FINAL(X, nbelements = 4)
```

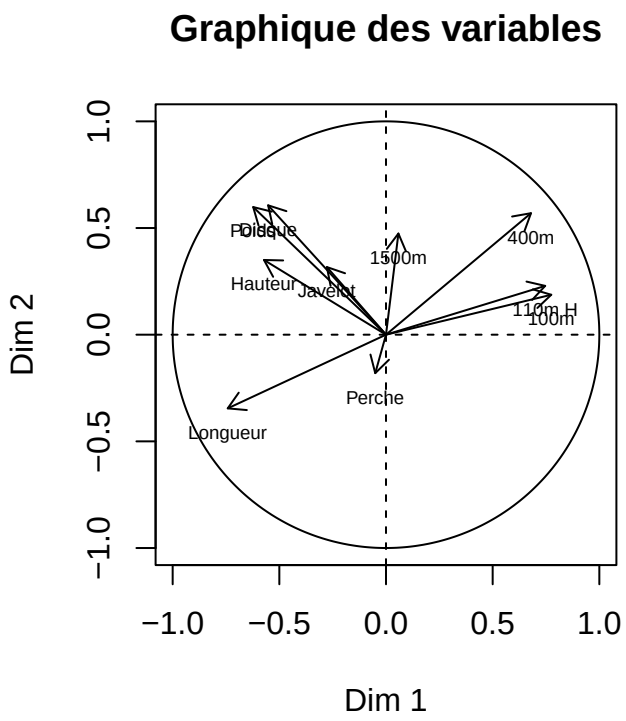
```
## [1] "Eigenvalues"
##       Variance % of var. Cumulative % of var
## [1,] 3.2719055 32.719055            32.71906
## [2,] 1.7371310 17.371310            50.09037
## [3,] 1.4049167 14.049167            64.13953
## [4,] 1.0568504 10.568504            74.70804
## [5,] 0.6847735  6.847735            81.55577
## [6,] 0.5992687  5.992687            87.54846
## [1] "Individuals"
##            Dim 1       ctr      cos2      Dim 2         ctr         cos2
## Sebrle -3.988895 12.157506 0.6954102 1.34906683 2.619234357 7.954314e-02
## Clay   -3.871273 11.451090 0.7112052 0.82669151 0.983545343 3.243204e-02
## Karpov -4.563298 15.910981 0.8517553 0.03950447 0.002245949 6.383365e-05
## Macey  -2.206055  3.718536 0.4230486 1.02898331 1.523786399 9.203950e-02
## [1] "Variables"
##              Corr 1    Cos2 1       Ctr 1     Corr 2    Cos2 2       Ctr 2
## 100m       0.7747198 18.343770 0.600190812  0.1871420  2.016090 0.550415232
## Longueur -0.7418997 16.822467 0.035022125 -0.3454213  6.868559 0.119315870
## Poids    -0.6225026 11.843540 0.034005993  0.5983033 20.606785 0.033200868
## Hauteur  -0.5719453  9.997887 0.001430221  0.3502936  7.063694 0.010360317
## 400m       0.6796099 14.116229 0.091322660  0.5694378 18.666374 0.001345279
## 110m H     0.7462453 17.020115 0.052532985  0.2287933  3.013382 0.056158895
```

2.5.4 Variables

- Lors de l'affichage de l'ACP pour les individus et les variables, afficher en plus les axes des abscisses et des ordonnées. Afficher les variables sous forme de vecteurs

```
variablesACP_FINAL <- function(axes, PCA) {

  V <- ACP_FINAL(PCA)$cor_var

  dim1 <- paste("Dim",as.character(axes[1]))
  dim2 <- paste("Dim",as.character(axes[2]))

  par(mar=c(4,4,4,4),pty = "s")
  plot(V[,axes],xlim = c(-1,1),ylim = c(-1,1),xlab=dim1,ylab=dim2, cex=0)
  text(V[,axes], labels = colnames(ACP_FINAL(PCA)$data), cex = 0.6, pos = 1)
  abline(v = 0, lty = 2)
  abline(h = 0, lty = 2)
  title("Graphique des variables")
  draw.circle(0, 0, 1, nv = 500, border = NULL, col = NA, lty = 1, lwd = 1)
  arrows(x0 = rep(0, times = length(V[,1])),
         y0 = rep(0, times = length(V[,1])),
         x1 = V[,1], y1 = V[,2],
```

```
         code = 2, length = 0.1)

}
variablesACP_FINAL(axes=1:2, X)
```



3 Annexes

Voici les codes intermédiaires entre les premières et les dernières fonctions que nous avons créées tout au long de ce projet. Par exemple, pour l'ACP, entre ACP et ACP_FINAL nous avons créé ACP_N.

```
ACP_N <- function(X, quanti.sup=NULL, quali.sup=NULL, cr = TRUE) {

  supp <- c()

  if (!is.null(quanti.sup)){
    for (i in quanti.sup) {
      supp <- append(supp, i)
    }
  }

  if (!is.null(quali.sup)) {
    for (i in quali.sup) {
      supp <- append(supp, i)
    }
  }

  if (!is.null(supp)) {
    supp <- sort(supp)
```

```
    X <- X[,-supp]
  }

  if (cr == FALSE) {
    data <- X
  }
  else {
    data <- scale(X)
  }

  cov <- cov(data) #ne peut pas faire le scale d'une variable qualitative
  vap <- eigen(cov)$values
  vep <- eigen(cov)$vectors

  return(list("vap"=vap,"vep"=vep,"data"=data,"cov"=cov))
}
```

Individus

```
individusACP_N <- function(axes, PCA, habillage) {
  V <- as.matrix(ACP_N(X)$vep)
  D <- as.matrix(ACP_N(X)$data)
  res <- D%*%V
  colour <- c()

  for (i in seq.int(1:nrow(res))) {
    if (abs(res[i,habillage]) < 1) {
      clr <- "green"
    }
    else if (abs(res[i,habillage]) < 2.5) {
      clr <- "orange"
    }
    else {
      clr <- "red"
    }
    colour <- append(colour, clr)
  }


  dim1 <- paste("Dim",as.character(axes[1]))
  dim2 <- paste("Dim",as.character(axes[2]))

  par(mar=c(4,4,4,4),pty = "s")
  plot(res[,axes], xlab = dim1, ylab = dim2, pch = 1, col = colour )
  text(res[,axes], labels = row.names(ACP_N(PCA)$data), cex = 0.6, pos = 1)
  abline(v = 0, lty = 2)
  abline(h = 0, lty = 2)
  title("ACP")
  return(res)
}
```

Summary

```
summary_N <- function(PCA,nbelements=10) {
```

```r
#Eigenvalues
vap <- ACP_N(PCA)$vap
vect <- c()
for (i in seq.int(0,length(vap))) {
  somme <- sum(vap)
  pourcentage <- (vap[i]/somme)*100
  vect <- append(vect, pourcentage)
}
eigenvalues <- cbind(vap, vect,  cumsum(vect))
colnames(eigenvalues) <- c("Variance", "% of var.", "Cumulative % of var")
print("Eigenvalues")
print(eigenvalues)

#Individuals
dim <- individusACP_N(axes=1:2, PCA)

  #Dim1
contr_1 <- c()
sommecarre_1 <- dim[,1]%*%dim[,1]
for (i in seq.int(0,length(dim[,1]))) {
  contr_1 <- append(contr_1, ((dim[,1][i])**2/sommecarre_1)*100)
}
individuals <- cbind(dim[,1], contr_1)

cos2_1 <- c()
for (i in seq.int(0,length(dim[,1]))) {
  cos2_1 <- append(cos2_1, dim[i,1]**2/sqrt(sum(dim[i,]**2))**2)
}
individuals <- cbind(individuals, cos2_1)

  #Dim2
contr_2 <- c()
sommecarre_2 <- dim[,2]%*%dim[,2]
individuals <- cbind(individuals, dim[,2])

for (i in seq.int(0,length(dim[,2]))) {
  contr_2 <- append(contr_2, ((dim[,2][i])**2/sommecarre_2)*100)
}
individuals <- cbind(individuals, contr_2)

cos2_2 <- c()
for (i in seq.int(0,length(dim[,2]))) {
  cos2_2 <- append(cos2_2, dim[i,2]**2/sqrt(sum(dim[i,]**2))**2)
}
individuals <- cbind(individuals, cos2_2)

colnames(individuals) <- c("Dim 1","ctr", "cos2", "Dim 2","ctr", "cos2")

print("Individuals")
print(individuals[1:nbelements,])

#Variables
  #Dim1
```

```
  corr_1 <- c()
  for (i in seq.int(0, length(vap))) {
    corr_1 <- append(corr_1, sqrt(vap[1])*ACP_N(PCA)$vep[i,1])
  }
  variables <- as.data.frame(corr_1)

  cos2_1 <- c()
  for (i in seq.int(0, length(corr_1))) {
    cos2_1 <- append(cos2_1, corr_1[i]**2)
  }
  variables <- cbind(variables, cos2_1)

  contr_1 <- c()
  sumSquare1 <- ACP_N(PCA)$vep[,1]%*%ACP_N(PCA)$vep[,1]
  for (i in seq.int(0,length(ACP_N(PCA)$vep[,1]))) {
    contr_1 <- append(contr_1, (ACP_N(PCA)$vep[i,1]**2/sumSquare1)*100)
  }
  variables <- cbind(variables, contr_1)

    #Dim2
  corr_2 <- c()
  for (i in seq.int(0, length(vap))) {
    corr_2 <- append(corr_2, sqrt(vap[2])*ACP_N(PCA)$vep[i,2])
  }
  variables <- cbind(variables, corr_2)

  cos2_2 <- c()
  for (i in seq.int(0, length(corr_2))) {
    cos2_2 <- append(cos2_2, corr_2[i]**2)
  }
  variables <- cbind(variables, cos2_2)

  contr_2 <- c()
  sumSquare2 <- ACP_N(PCA)$vep[,2]%*%ACP_N(PCA)$vep[,2]
  for (i in seq.int(0,length(ACP_N(PCA)$vep[,2]))) {
    contr_2 <- append(contr_2, (ACP_N(PCA)$vep[i,2]**2/sumSquare2)*100)
  }
  variables <- cbind(variables, contr_2)

  nom_Var <- colnames(ACP_N(PCA)$data)
  row.names(variables) <- nom_Var
  colnames(variables) <- c("Corr 1", "Cos2 1", "Ctr 1","Corr 2", "Cos2 2", "Ctr 2")

  print("Variables")
  print(variables)

}
```

Variables

```
variablesACP_N <- function(axes, PCA) {
  vep <- ACP_N(PCA)$vep
  vap <- ACP_N(PCA)$vap
  cor_v <- c()
```

```
  data <- ACP_N(PCA)$data
  taille_v <- length(vep[,1])

  for (k in seq.int(1, taille_v)) {
    cor_dv <- c()
    for (i in seq.int(0, length(vap))) {
      cor_dv <- append(cor_dv, sqrt(vap[k])*vep[i,k])
    }
    cor_v <- cbind(cor_v, cor_dv)
  }

  dim1 <- paste("Dim",as.character(axes[1]))
  dim2 <- paste("Dim",as.character(axes[2]))

  par(mar=c(4,4,4,4),pty = "s")
  plot(cor_v[,axes],xlim = c(-1,1),ylim = c(-1,1),xlab=dim1,ylab=dim2, cex = 0)
  text(cor_v[,axes], labels = colnames(data), cex = 0.6, pos = 1)
  abline(v = 0, lty = 2)
  abline(h = 0, lty = 2)
  title("Graphique des variables")
  draw.circle(0, 0, 1, nv = 500, border = NULL, col = NA, lty = 1, lwd = 1)
  arrows(x0 = rep(0, times = length(cor_v[,1])),
         y0 = rep(0, times = length(cor_v[,1])),
         x1 = cor_v[,1], y1 = cor_v[,2],
         code = 2, length = 0.1)

}
```