Fiche TD avec le logiciel : tdr601

Initiation à l'analyse en composantes principales

A.B. Dufour & J.R. Lobry

Une première approche très intuitive et interactive de l'ACP. Centrage et réduction des données.

Table des matières

1 Introduction

Les méthodes d'analyse multivariées sont une branche à part entière des statistiques. Nous vous proposons ici une introduction, très sommaire, à l'une d'entre elles, connue sous le nom d'analyse en composantes principales (ACP). Plus précisément, nous n'utiliserons qu'une des variantes de l'ACP, l'ACP centrée et réduite.

2 Les données

Nous allons utiliser un jeu de données très simple extrait de `survey` du paquetage `MASS` [3].

```
library(MASS)
data(survey)
names(survey)
```

```
 [1] "Sex"    "Wr.Hnd" "NW.Hnd" "W.Hnd"  "Fold"   "Pulse"  "Clap"   "Exer"   "Smoke"
[10] "Height" "M.I"    "Age"
```

Pour ne pas nous embêter par la suite avec les données manquantes, nous ne conservons que les individus entièrement documentés :

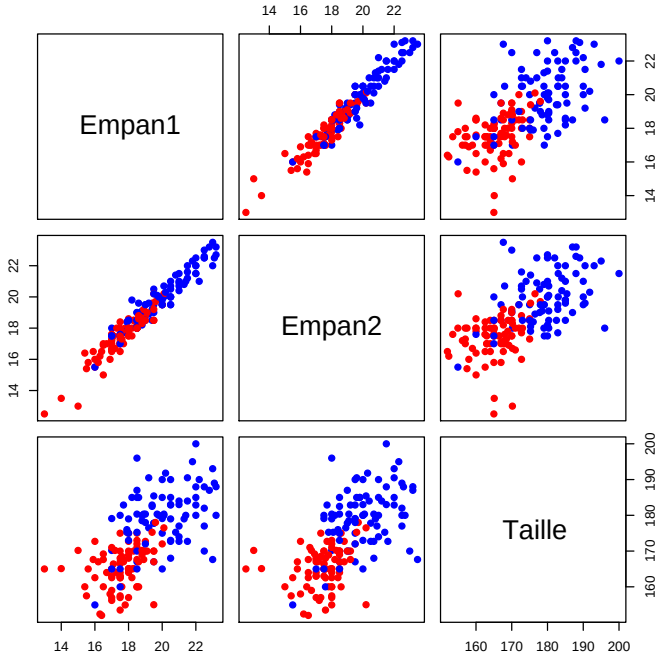
```
survey.cc <- survey[complete.cases(survey), ]
```

Nous récupérons le sexe des individus dans un vecteur, et définissons un vecteur de couleurs pour distinguer facilement les filles de garçons :

```
sexe <- survey.cc$Sex
col <- ifelse(sexe == "Male", "blue", "red")
```

Nous extrayons trois variables quantitatives de ce jeu de données :

```
mesures <- survey.cc[, c("Wr.Hnd", "NW.Hnd", "Height")]
names(mesures) <- c("Empan1", "Empan2", "Taille")
plot(mesures, col = col, pch = 19)
```

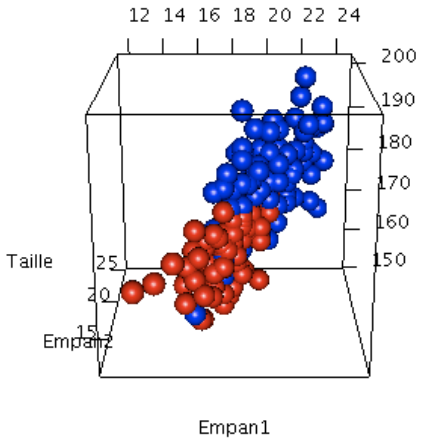


Notez que les variables morphométriques des garçons (en bleu) sont en moyenne plus grandes que celles des filles (en rouge). Ceci nous permettra de nous orienter plus facilement dans les représentations graphiques suivantes.

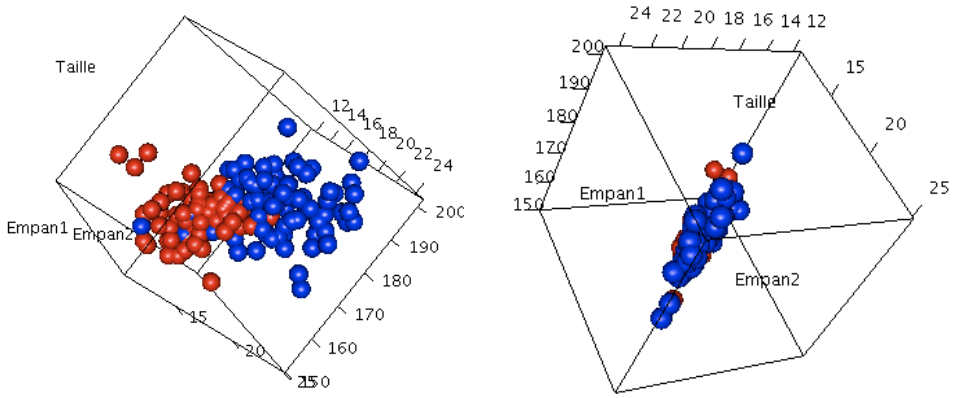
3 Visualisation des données en trois dimensions

Chaque individu est caractérisé par trois variables, soit par un point dans $\mathbb{R}^3$. La fonction `plot3d()` de la bibliothèque `rgl` [1] vous permet d'explorer facilement un nuage de points en 3 dimensions.

```
library(rgl)
plot3d(mesures, type = "s", col = col)
```



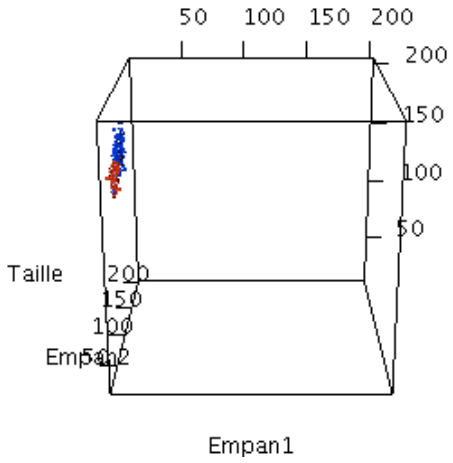
Faites tourner avec le curseur de la souris cette représentation pour en avoir différents points de vue :



4 Centrage et réduction

Les représentation graphiques précédentes sont trompeuses parce que nous n'avons pas utilisé la même échelle en x, y et z. Imposons une échelle commune pour visualiser les données :

```
lims <- c(min(mesures), max(mesures))
plot3d(mesures, type = "s", col = col, xlim = lims, ylim = lims,
    zlim = lims)
```

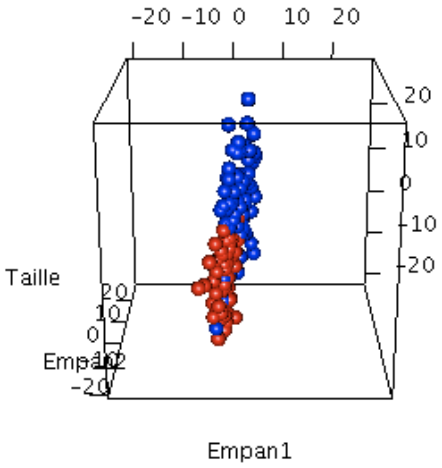


Voici donc à quoi ressemblent réellement les donées. On voit tout de suite le problème, comme les tailles sont en moyenne beaucoup plus grandes que les empans, les points se trouvent complètement collés sur le plan des empans.

4.1 Centrage

L'opération de centrage consiste à enlever la moyenne à chaque variable. La fonction `scale()` permet d'effectuer directement cette opération :

```
mesures.c <- scale(mesures, center = TRUE, scale = FALSE)
lims <- c(min(mesures.c), max(mesures.c))
plot3d(mesures.c, type = "s", col = col, xlim = lims, ylim = lims,
    zlim = lims)
```



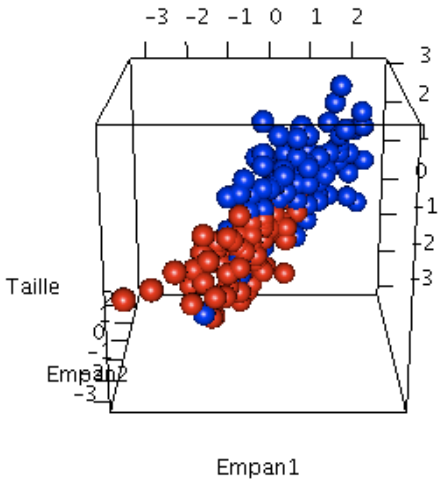
C'est mieux, le nuage est maintenant centré autour de l'origine. Mais comme la variabilité est beaucoup plus forte pour la taille que pour les empans, notre nuage de points a l'aspect d'une galette complètement applatie.

4.2 Centrage et réduction

Cette opération consiste a centrer les données puis diviser les valeurs par l'écart-type. La fonction `scale()` permet d'effectuer directement cette opération :

```
mesures.cr <- scale(mesures)
```

```
lims <- c(min(mesures.cr), max(mesures.cr))
plot3d(mesures.cr, type = "s", col = col, xlim = lims, ylim = lims,
    zlim = lims)
```

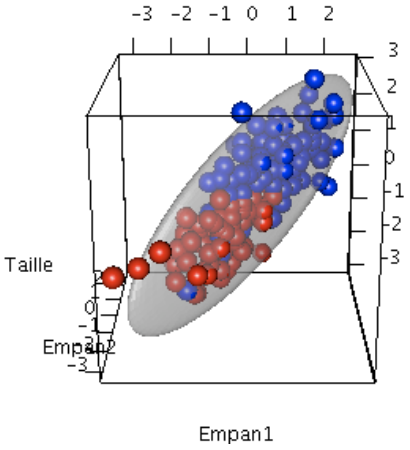


Quand on fait une ACP centrée réduite, on travaille avec les données après centrage et réduction. Il est donc important de bien comprendre à quoi correspondent ces opérations.

5 La forme générale du nuage

Dans l'exemple que nous avons choisi, la forme générale du nuage de points est celui d'une dragée (le terme technique est un ellipsoïde).

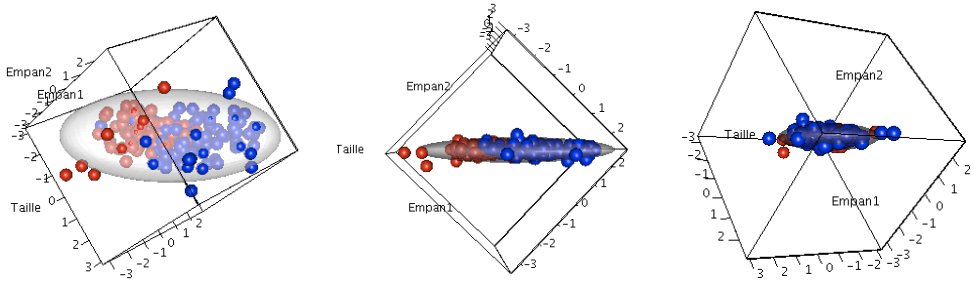
```
plot3d(mesures.cr, type = "s", col = col, xlim = lims, ylim = lims,
    zlim = lims)
plot3d(ellipse3d(cor(mesures.cr)), col = "grey", alpha = 0.5, add = TRUE)
```



Une dragée est définie par ses trois axes :

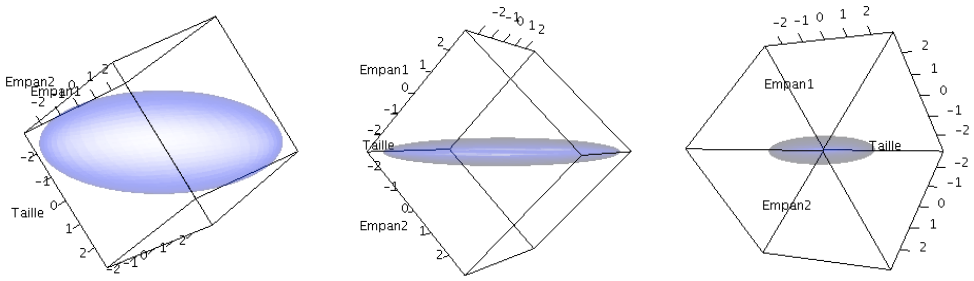
1. Le premier axe correspondant au plus grand diamètre de l'ellipsoïde, la longueur de la dragée.
2. Le deuxième axe correspondant au diamètre moyen de l'ellipsoïde, la largeur de la dragée.
3. Le troisième axe correspondant au plus petit diamètre de l'ellipsoïde, l'épaisseur de la dragée.

Faites tourner le graphique précédent pour représenter le nuage de points dans le plan des axes (1,2), puis (1,3) puis (2,3) :



Faire le même exercice en ne conservant que la dragée :

```
plot3d(ellipse3d(cor(mesures.cr)), col = "blue", alpha = 0.25, xlab = "Empan1",
    ylab = "Empan2", zlab = "Taille")
```



Avez vous noté au passage ? Pour dessiner la dragée, nous n'avons eu besoin que de la matrice de variance co-variance :

```
cor(mesures.cr)
```

```
          Empan1    Empan2    Taille
Empan1 1.0000000 0.9651297 0.6510316
Empan2 0.9651297 1.0000000 0.6296482
Taille 0.6510316 0.6296482 1.0000000
```

Si vous aviez à choisir entre les trois plans que nous avons envisagés ci-dessus, vous n'hésiteriez pas à prendre le plan défini par les deux premiers axes de la dragée parce que c'est dans cette représentation que l'on a le moins de perte d'information par rapport au nuage de points dans $\mathbb{R}^3$: c'est dans cette projection que les points sont les plus étalés dans le plan (on dit aussi que l'on a conservé le maximum possible de l'inertie initiale du nuage de points). Ce faisant, vous avez en fait réalisé une ACP à la main.

6 ACP centrée-réduite dans ade4

6.1 Calculs

Utiliser la fonction `dudi.pca()` du paquetage `ade4` [2] pour exécuter une ACP centrée réduite :

```
library(ade4)
acp <- dudi.pca(mesures, scann = FALSE, n = 3)
names(acp)
```

```
 [1] "tab"  "cw"   "lw"   "eig"  "rank" "nf"   "c1"   "li"   "co"   "l1"   "call"
[12] "cent" "norm"
```

Nous avons utilisé ici les options `scann = FALSE` pour conserver automatiquement `n = 3` facteurs. En général on ne procède pas ainsi : on commence par examiner le graphe des valeurs propres qui exprime quelle fraction de la variance totale est prise en compte par les axes successifs. Essayer avec :

```
acp <- dudi.pca(mesures)
```

Répondre 3 à la question `"Select the number of axes :"`. Dans la pratique, on ne conserve qu'un nombre réduit d'axes, ici nous les avons tous conservés pour des raisons pédagogiques.

L'objet renvoyé par la fonction `dudi.coa()` est très riche. Nous allons examiner tous ses composants un à un.

6.1.1 tab

Le data.frame `tab` contient les données du tableau initial après centrage et réduction. Par rapport au résultat obtenu avec la fonction `scale()` vous noterez de petites différences :

```
head(acp$tab)
```

```
      Empan1     Empan2      Taille
1 -0.1580263 -0.3711415  0.05273898
2  0.3645805  0.8972021  0.53612965
5  0.6258839  0.6435334 -0.75291214
6 -0.4193297 -0.5233427  0.02454119
7 -0.5761118 -0.5233427  1.04771811
8 -0.9419365 -0.7262777 -1.55856326
```

```
head(mesures.cr)
```

```
      Empan1     Empan2      Taille
1 -0.1575553 -0.3700353  0.05258178
2  0.3634938  0.8945279  0.53453164
5  0.6240183  0.6416152 -0.75066799
6 -0.4180799 -0.5217828  0.02446804
7 -0.5743946 -0.5217828  1.04459524
8 -0.9391290 -0.7241129 -1.55391775
```

Cette petite différence est due à l'utilisation d'une variance en $\frac{1}{n}$ dans `dudi.pca()` contre une variance en $\frac{1}{n-1}$ dans `scale()`. Pour retrouver exactement le tableau utilisé dans `dudi.pca()` faire :

```
var.n <- function(x) sum((x - mean(x))^2)/length(x)
scale.n <- function(x) (x - mean(x))/sqrt(var.n(x))
head(apply(mesures, 2, scale.n))
```

```
      Empan1     Empan2      Taille
1 -0.1580263 -0.3711415  0.05273898
2  0.3645805  0.8972021  0.53612965
5  0.6258839  0.6435334 -0.75291214
6 -0.4193297 -0.5233427  0.02454119
7 -0.5761118 -0.5233427  1.04771811
8 -0.9419365 -0.7262777 -1.55856326
```

6.1.2 cw

Le vecteur `cw` donne le poids des colonnes (*column weight*), c'est à dire le poids des variables. Par défaut, chaque variable a un poids de 1.

```
acp$cw
```

```
[1] 1 1 1
```

6.1.3 lw

Le vecteur `lw` donne le poids des lignes (*line weight*), c'est à dire le poids des individus. Par défaut, chaque individu a un poids de $\frac{1}{n}$.

```
head(acp$lw)
```

```
[1] 0.005952381 0.005952381 0.005952381 0.005952381 0.005952381 0.005952381
```

```
head(acp$lw) * nrow(mesures)
```

```
[1] 1 1 1 1 1 1
```

6.1.4 eig

Le vecteur `eig` donne les valeurs propres (*eigen values*) dans le plus petit des deux espaces diagonalisés.

```
acp$eig
```

```
[1] 2.50871762 0.45683484 0.03444753
```

```
sum(acp$eig)
```

```
[1] 3
```

Les valeurs propres nous renseignent sur la fraction de l'inertie totale prise en compte par chaque axe :

```
(pve <- 100 * acp$eig/sum(acp$eig))
```

```
[1] 83.623921 15.227828  1.148251
```

```
cumsum(pve)
```

```
[1]  83.62392  98.85175 100.00000
```

Dans notre exemple, le premier axe factoriel extrait 83.6 % de l'inertie totale, le deuxième axe factoriel 15.2 % de l'inertie totale. Le premier plan factoriel représente donc 98.9 % de l'inertie initiale. Ceci signifie que lorsque nous projetons le nuage de points intial dans $\mathbb{R}^3$ sur le plan défini par les deux premiers axes factoriels, nous avons perdu peu d'information.

6.1.5 rank

Cet entier donne le rang (*rank*) de la matrice diagonalisée, dans notre cas le nombre de variables indépendantes.

```
acp$rank
```

```
[1] 3
```

```
bismesures <- cbind(mesures, mesures)
head(bismesures)
```

```
  Empan1 Empan2 Taille Empan1.1 Empan2.1 Taille.1
1   18.5   18.0 173.00     18.5     18.0   173.00
2   19.5   20.5 177.80     19.5     20.5   177.80
5   20.0   20.0 165.00     20.0     20.0   165.00
6   18.0   17.7 172.72     18.0     17.7   172.72
7   17.7   17.7 182.88     17.7     17.7   182.88
8   17.0   17.3 157.00     17.0     17.3   157.00
```

```
dudi.pca(bismesures, scann = F, n = 3)$rank
```

```
[1] 3
```

6.1.6 nf

Cet entier donne le nombre de facteurs conservés dans l'analyse :

```
acp$nf
```

```
[1] 3
```

6.1.7 c1

Donne les coordonnées des variables (colonnes). Les vecteurs sont de norme unité :

```
acp$c1
```

```
             CS1        CS2         CS3
Empan1 0.6084890 -0.3420962  0.71603859
Empan2 0.6040404 -0.3855223 -0.69750107
Taille 0.5146613  0.8569380 -0.02794614
```

```
sum(acp$cw * acp$c1$CS1^2)
```

```
[1] 1
```

6.1.8 l1

Donne les coordonnées des individus (lignes). Les vecteurs sont de norme unité :

```
head(acp$l1)
```

```
          RS1         RS2        RS3
1 -0.18511289  0.35854289  0.7771789
2  0.65642982 -0.01654562 -2.0459458
5  0.24122137 -1.63843055  0.1095513
6 -0.35270516  0.54186131  0.3453116
7 -0.08047126  1.91845520 -0.4136080
8 -1.14527378 -1.08502402 -0.6698669
```

```
sum(acp$lw * acp$l1$RS1^2)
```

```
[1] 1
```

6.1.9 co

Donne les coordonnées des variables (colonnes). Les vecteurs sont normés à la racine carré de la valeur propre correspondante :

```
acp$co
```

```
          Comp1      Comp2        Comp3
Empan1 0.9637816 -0.2312213  0.132897100
Empan2 0.9567355 -0.2605728 -0.129456527
Taille 0.8151685  0.5792006 -0.005186817
```

```
sum(acp$cw * acp$co$Comp1^2)
```

```
[1] 2.508718
```

Le lien entre les c1 et les co s'obtient par :

```
acp$c1$CS1 * sqrt(acp$eig[1])
```

```
[1] 0.9637816 0.9567355 0.8151685
```

```
t(t(acp$c1) * sqrt(acp$eig))
```

```
             CS1        CS2         CS3
Empan1 0.9637816 -0.2312213  0.132897100
Empan2 0.9567355 -0.2605728 -0.129456527
Taille 0.8151685  0.5792006 -0.005186817
```

6.1.10 li

Donne les coordonnées des individus (lignes). Les vecteurs sont normés à la racine carré de la valeur propre correspondante :

```
head(acp$li)
```

```
       Axis1       Axis2       Axis3
1 -0.2931990  0.24233756  0.14424478
2  1.0397147 -0.01118311 -0.37972849
5  0.3820689 -1.10740798  0.02033277
6 -0.5586473  0.36624167  0.06409000
7 -0.1274579  1.29667540 -0.07676584
8 -1.8139913 -0.73336294 -0.12432761
```

```
sum(acp$lw * acp$li$Axis1^2)
```

```
[1] 2.508718
```

```
head(acp$l1$RS1 * sqrt(acp$eig[1]))
```

```
[1] -0.2931990  1.0397147  0.3820689 -0.5586473 -0.1274579 -1.8139913
```

```
head(t(t(acp$l1) * sqrt(acp$eig)))
```

```
         RS1         RS2         RS3
1 -0.2931990  0.24233756  0.14424478
2  1.0397147 -0.01118311 -0.37972849
5  0.3820689 -1.10740798  0.02033277
6 -0.5586473  0.36624167  0.06409000
7 -0.1274579  1.29667540 -0.07676584
8 -1.8139913 -0.73336294 -0.12432761
```

6.1.11 call

Cet objet garde une trace de la façon dont ont été conduits les calculs lors de l'appel de la fonction `dudi.pca()` :

```
acp$call
```

```
dudi.pca(df = mesures, scannf = FALSE, nf = 3)
```

La fonction `eval()` permet de refaire les mêmes calculs :

```
eval(acp$call)
```

```
Duality diagramm
class: pca dudi
$call: dudi.pca(df = mesures, scannf = FALSE, nf = 3)
$nf: 3 axis-components saved
$rank: 3
eigen values: 2.509 0.4568 0.03445
  vector length mode    content
1 $cw    3      numeric column weights
2 $lw    168    numeric row weights
3 $eig   3      numeric eigen values

  data.frame nrow ncol content
1 $tab       168  3    modified array
2 $li        168  3    row coordinates
3 $l1        168  3    row normed scores
4 $co        3    3    column coordinates
5 $c1        3    3    column normed scores
other elements: cent norm
```

```
identical(eval(acp$call), acp)
```

```
[1] TRUE
```

6.1.12 cent

Ce vecteur donne les moyennes (cent pour centrage) des variables analysées :

```
acp$cent
```

```
  Empan1   Empan2   Taille
18.80238 18.73155 172.47631
```

```
colMeans(mesures)
```

```
  Empan1   Empan2   Taille
18.80238 18.73155 172.47631
```

6.1.13 norm

Ce vecteur donne les écarts-types (sur $\sqrt{n}$) des variables analysées :

```
acp$norm
```

```
  Empan1   Empan2   Taille
1.913484 1.971075 9.929857
```

```
sd.n <- function(x) sqrt(var.n(x))
apply(mesures, 2, sd.n)
```

```
  Empan1   Empan2   Taille
1.913484 1.971075 9.929857
```

6.2 Dé-réduction et dé-centrage

Si vous avez bien compris en quoi consiste l'opération de centrage et réduction, vous devez pouvoir être capables de faire l'opération inverse. À partir des objets `acp$tab`, `acp$cent` et `acp$norm` reconstituez les données de départ, placez le résultat dans l'objet `recon` :

```
head(recon)
```

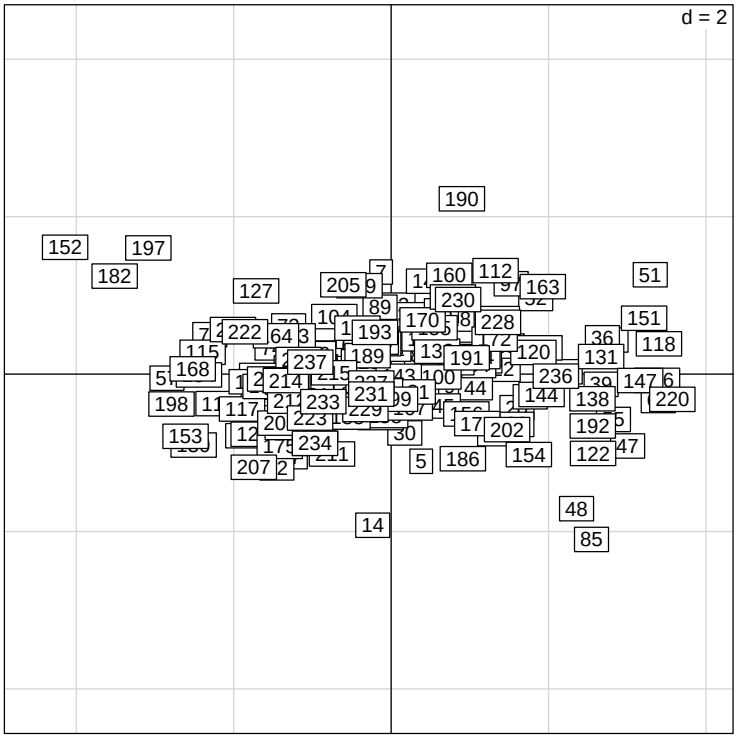
```
  Empan1 Empan2 Taille
1   18.5   18.0 173.00
2   19.5   20.5 177.80
5   20.0   20.0 165.00
6   18.0   17.7 172.72
7   17.7   17.7 182.88
8   17.0   17.3 157.00
```

6.3 Représentations graphiques dans ade4

6.3.1 Représentation des individus

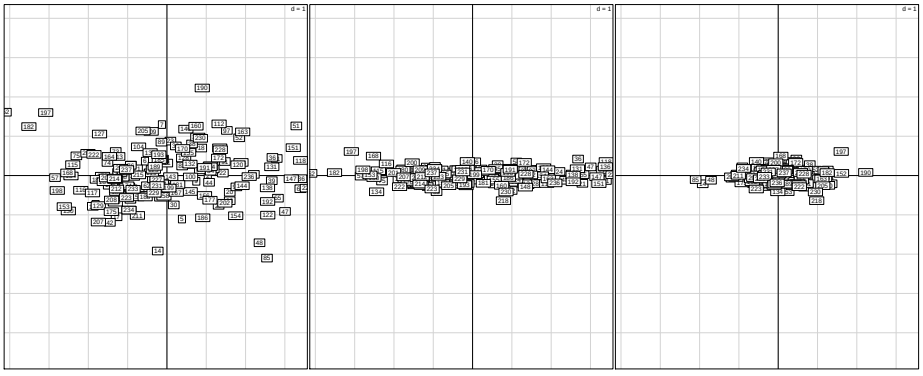
La fonction `s.label()` permet de représenter les individus sur les différents plans factoriels, par exemple sur le premier plan factoriel :

```
s.label(acp$li, xax = 1, yax = 2)
```



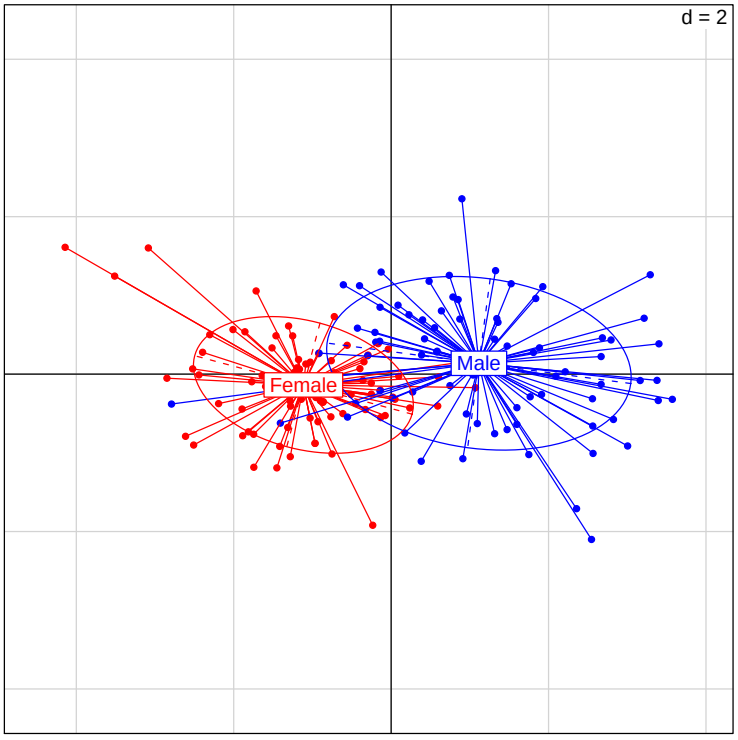
Faire les représentations dans les plans (1,2), (1,3) et (2,3) avec une échelle commune pour tous les graphiques :

```
par(mfrow = c(1, 3))
lims <- c(min(acp$li), max(acp$li))
s.label(acp$li, xax = 1, yax = 2, xlim = lims, ylim = lims)
s.label(acp$li, xax = 1, yax = 3, xlim = lims, ylim = lims)
s.label(acp$li, xax = 2, yax = 3, xlim = lims, ylim = lims)
```

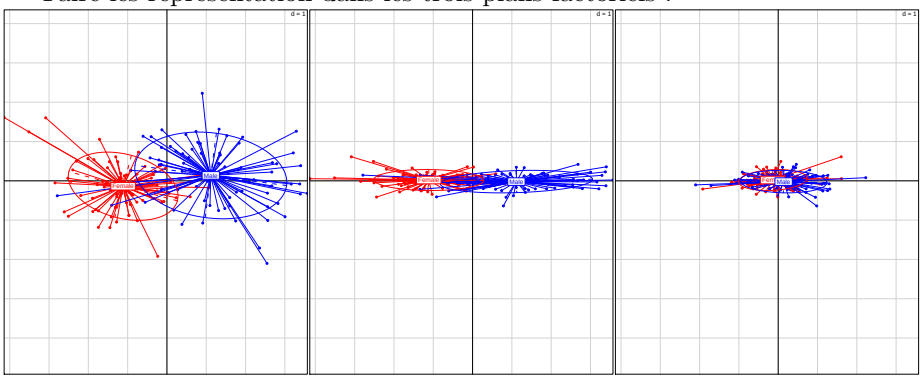


La fonction `s.class()` permet de porter en information supplémentaire des groupes d'individus, par exemple :

```
gcol <- c("red", "blue")
s.class(dfxy = acp$li, fac = sexe, col = gcol, xax = 1, yax = 2)
```



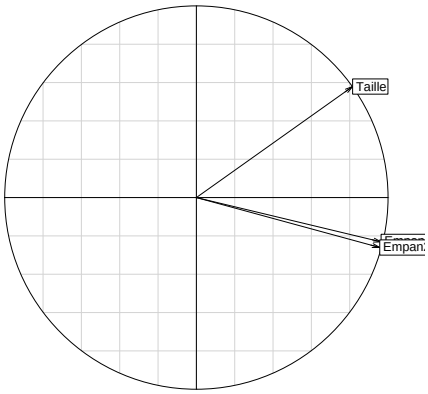
Faire les représentation dans les trois plans factoriels :



6.3.2 Représentation des variables

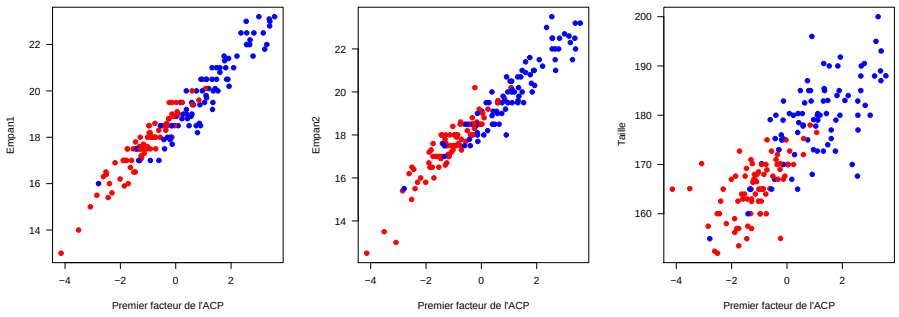
La fonction `s.corcircle()` représente les variables par le cercle des corrélations :

```
s.corcircle(acp$co, xax = 1, yax = 2)
```



Le premier facteur de l'ACP est corrélé positivement aux trois variables de départ, on dit que c'est un effet "taille". L'ACP joue ici son rôle de recherche de *variable latente*. La première composante principale prédit les trois variables. C'est une *explicative cachée*. C'est la variable cachée qui prédit au mieux les autres, c'est aussi la variable cachée qui est le mieux prédite par toutes les autres.

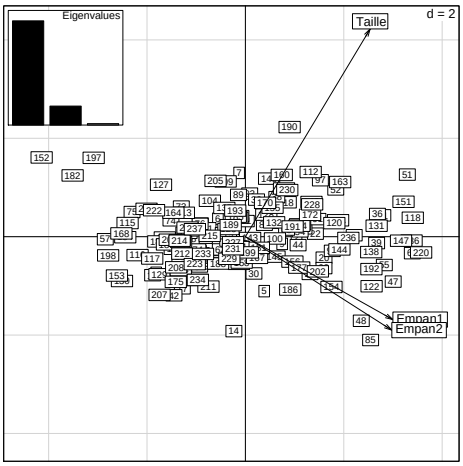
```
par(mfrow = c(1, 3))
for (i in 1:3) {
    plot(x = acp$li[, 1], y = mesures[, i], pch = 19, col = col,
        xlab = "Premier facteur de l'ACP", las = 1, ylab = colnames(mesures)[i])
}
```



6.3.3 Représentation simultanée des individus et des variables

La fonction `scatter()` permet de représenter simultanément les individus et les variables :

```
scatter(acp)
```

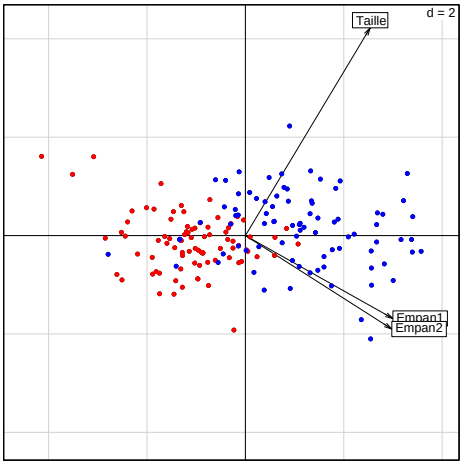


Enrichir le graphique en portant l'information sur les groupes :

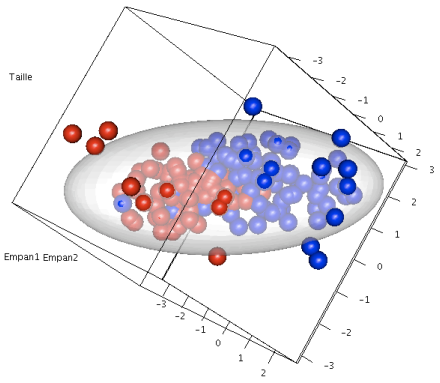
```
scatter(acp, clab.row = 0, posieig = "none")
```

```
NULL
```

```
s.class(acp$li, sexe, col = gcol, add.plot = TRUE, cstar = 0, clabel = 0,
    cellipse = 0)
```



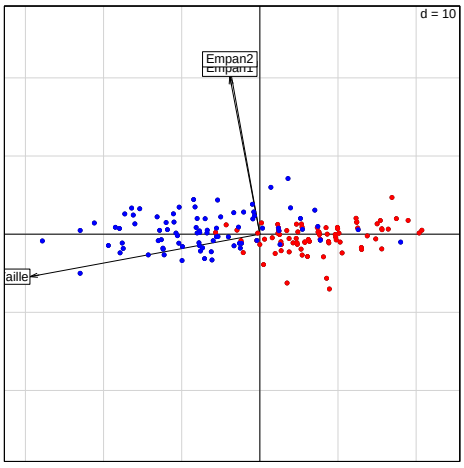
Faire le lien avec ce qui avait été obtenu à la main avec la fonction `plot3d()`, notez en particulier comment les axes de la base initiale se projettent sur le premier plan factoriel :

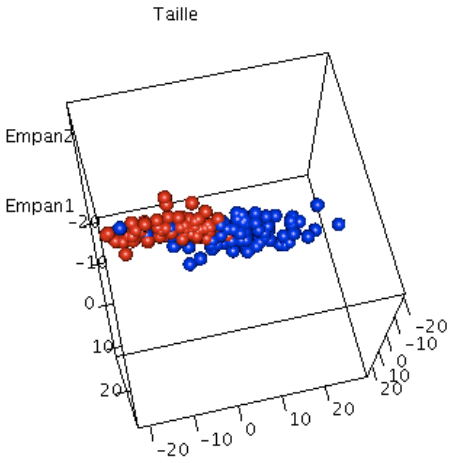


7 Pour aller plus loin

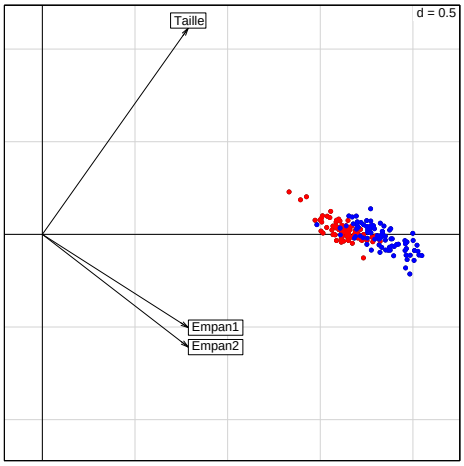
Nous n'avons utilisé jusqu'à présent que l'ACP centrée réduite. Ce n'est pas forcément la meilleure option en fonction de vos objectifs. Réaliser les autres combinaisons possibles.

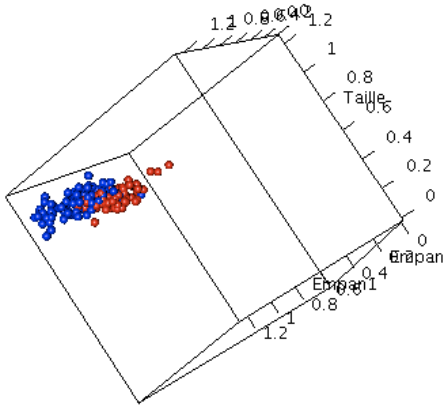
7.1 ACP centrée non réduite



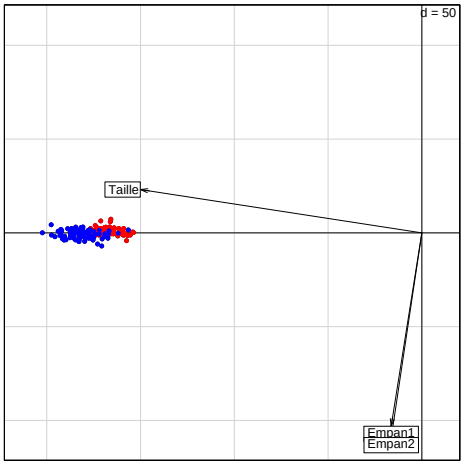


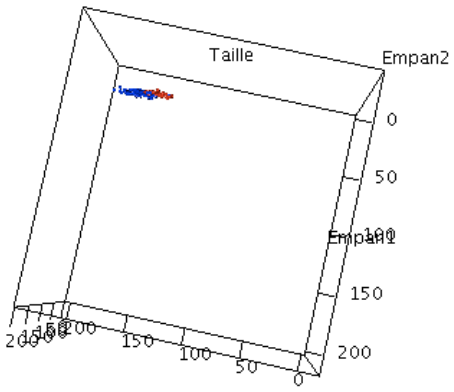
7.2 ACP non centrée réduite





7.3 ACP non centrée non réduite





Références

[1] Daniel Adler and Duncan Murdoch. *rgl : 3D visualization device system (OpenGL)*, 2006. R package version 0.68.

[2] D. Chessel, A.-.B. Dufour, and J. Thioulouse. The ade4 package-I- One-table methods. *R News*, 4 :5–10, 2004.

[3] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.