Université de Technologie de Compiègne
SY09 : Analyse des données et Apprentissage automatique

SY09 : TP 04 : Analyse en compostantes principales

Julien Jerphanion

Printemps 2018

Table des matières

1 Données Notes **2**
1.1 Analyse . . . 2
1.2 Analyse en composantes principales . . . 6

2 ACP avec R **12**

3 Données Crabs **16**
3.1 Analyse exploratoire . . . 16
3.1.1 Analyse multivariée en fonction du sexe . . . 17
3.1.2 Analyse multivariée en fonction de l'espèce . . . 19
3.1.3 Corrélation entre variables . . . 20
3.2 ACP . . . 22

4 Données Pima **28**
4.1 Analyse exploratoire . . . 29
4.2 ACP . . . 32

5 Données Mutations **34**

1 Données Notes

Le jeu de données se trouve dans le fichier `sy02-p2016.csv`. Il contient des informations de résultats des étudiants ayant suivis SY02 au semestre de printemps 2016.

```
levels = c("Cor1","Cor2","Cor3","Cor4","Cor5","Cor6","Cor7","Cor8")

notes = read.csv("./donnees/sy02-p2016.csv", na.strings="", header=T)
notes$nom = factor(notes$nom, levels=notes$nom)
notes$correcteur.median = factor(notes$correcteur.median,
                                 levels = levels)
notes$correcteur.final = factor(notes$correcteur.final,
                                levels = levels)
notes$niveau = factor(notes$niveau, ordered=T)
notes$resultat = factor(notes$resultat,
                        levels=c("F","Fx","E","D","C","B","A"),
                        ordered=T)

head(notes)
```

```
##    nom specialite niveau statut dernier.diplome.obtenu note.median
## 1 Etu1         GI      4    UTC                   BAC         6.5
## 2 Etu2         GB      2    UTC                   DUT         6.5
## 3 Etu3        GSU      4    UTC                   BAC        14.0
## 4 Etu4        GSM      2    UTC                   BAC        13.0
## 5 Etu5         GI      4    UTC                   DUT         7.5
## 6 Etu6         GM      4    UTC                   DUT         8.5
##   correcteur.median note.final correcteur.final note.totale resultat
## 1              Cor6        5.0             Cor4         5.6        F
## 2              Cor6       13.0             Cor4        10.4        E
## 3              Cor2       17.0             Cor3        15.8        B
## 4              Cor2       13.5             Cor3        13.3        C
## 5              Cor2       11.5             Cor3         9.9       Fx
## 6              Cor4       10.5             Cor5         9.7       Fx
```

1.1 Analyse

On peut dresser une analyse descriptive générale du jeu de données.

```
summary(notes)
```

```
##       nom        specialite niveau      statut
##  Etu1   :  1   GB     :65   1: 43   Echange:  6
##  Etu2   :  1   GM     :61   2:160   UTC    :290
##  Etu3   :  1   GSM    :53   3:  5
##  Etu4   :  1   GI     :45   4: 71
##  Etu5   :  1   GSU    :40   5:  7
##  Etu6   :  1   GP     :16   6: 10
##  (Other):290   (Other):16
##         dernier.diplome.obtenu  note.median    correcteur.median
##  BAC                   :109    Min.   : 0.00   Cor4   :49
##  DUT                   : 92    1st Qu.: 8.00   Cor5   :49
##  CPGE                  : 41    Median :11.00   Cor6   :49
##  ETRANGER SUPERIEUR:  11       Mean   :10.92   Cor7   :49
##  LICENCE               :  9    3rd Qu.:14.00   Cor2   :48
```

```
##  (Other)           : 28        Max.   :20.00   (Other):49
##  NA's              :  6        NA's   :3       NA's   : 3
##    note.final    correcteur.final  note.totale        resultat
##  Min.   : 0.00   Cor3   :48       Min.   : 0.900   C      :58
##  1st Qu.: 9.50   Cor5   :47       1st Qu.: 9.775   F      :49
##  Median :13.00   Cor6   :47       Median :12.300   D      :44
##  Mean   :12.38   Cor7   :47       Mean   :11.845   B      :42
##  3rd Qu.:16.00   Cor4   :46       3rd Qu.:14.800   E      :37
##  Max.   :19.50   (Other):49       Max.   :18.400   (Other):54
##  NA's   :12      NA's   :12       NA's   :12       NA's   :12
```

Ici `nom`, `specialite`, `status`, `dernier.diplome.obtenu`, `correcteur.median`, `correcteur.final` sont des variables qualitative nominales ; `niveau` et `resultat` sont des variables qualitative ordinales et les dernières – `note.median`, `note.final` et `note.totale` – sont quantitatives.

`note.median`, `note.final` et `note.totale` varient dans $[0, 20]$.

```
domaineNotes = unique(c(notes$note.median,notes$note.final,notes$note.final))
print(min(domaineNotes, na.rm = TRUE))
```

```
## [1] 0
```

```
print(max(domaineNotes, na.rm = TRUE))
```

```
## [1] 20
```

`specialite` varient dans les branches de l'UTC (`GI`, `GB`, `GSU`, `GSM`, `GM`) avec le `TC` mais aussi `Hutech` et le master `ISS`.

```
domaineSpecialite = unique(notes$specialite)
domaineSpecialite
```

```
## [1] GI     GB     GSU    GSM    GM     TC     GP     ISS    HuTech
## Levels: GB GI GM GP GSM GSU HuTech ISS TC
```

`status` varie dans entre échange et UTC :

```
domaineStatut = unique(notes$statut)
domaineStatut
```

```
## [1] UTC     Echange
## Levels: Echange UTC
```

`niveau`, qui représente le semestre, varie dans $[[1, 6]]$.

Les valeurs manquantes peuvent s'expliquer :

- pour `note.median` et `note.final`, elles correspondent au nombre d'étudiants absent respectivement au médian et au final ;
- pour `resultat`, `note.totale` cela correspond aussi au étudiants non présents ;
- pour `correcteur.median` et `correcteur.final`, cela s'explique très certainement parce que les correcteurs ont dû se répartir les corrections, certains les faisant pour le médian et d'autres pour le final ;

Les liens statistique peut se déterminer :

- via les *coefficients de corrélation de Pearson ou de Spearman* pour une comparaison quantitatif contre quantitatif ;
- via une *ANOVA* (ou encore une *ANCOVA* (Analyse de la Covariance)) dans le cas qualitatif contre quantitatif ;
- via un *test du* χ^2 pour le cas qualitatif contre qualitatif.

Le coefficient de Pearson met en exergue des liens linéaires entre variables, tandis que Spearman peut témoigner d'un lien monotone entre celles-ci.

On peut calculer le coefficient de corrélation de Pearson entre les notes du médian et les notes du final pour voir s'il existe un lien entre la réussite au médian et la réussite au final.

```
cor.test(notes$note.median,notes$note.final, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  notes$note.median and notes$note.final
## t = 7.9848, df = 282, p-value = 3.592e-14
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3294949 0.5198286
## sample estimates:
##       cor
## 0.4294183
```

On ne peut rejetter l'indépendance des notes obtenues au médian des notes obtenues au final.

Essayons maintenant de voir si le dernier diplôme obtenu influe sur la note finale obtenue pour l'UV. Pour se faire, on peut construire un modèle linéaire où l'on cherche à exprimer la note totale en fonction de dernier diplôme et d'un terme de biais :

```
model = lm(notes$note.totale ~ 1 + notes$dernier.diplome.obtenu)
model
```

```
##
## Call:
## lm(formula = notes$note.totale ~ 1 + notes$dernier.diplome.obtenu)
##
## Coefficients:
##                                          (Intercept)
##                                              11.2667
##          notes$dernier.diplome.obtenuAUTRE 2E CYCLE
##                                              -6.7667
## notes$dernier.diplome.obtenuAUTRE DIPLOME SUPERIEUR
##                                               0.6333
##                      notes$dernier.diplome.obtenuBAC
##                                               2.0220
##                      notes$dernier.diplome.obtenuBTS
##                                              -0.1381
##                     notes$dernier.diplome.obtenuCPGE
##                                              -0.4282
##                     notes$dernier.diplome.obtenuDEUG
##                                              -1.1667
##                      notes$dernier.diplome.obtenuDUT
##                                              -0.1723
##     notes$dernier.diplome.obtenuETRANGER SECONDAIRE
##                                               1.3083
##      notes$dernier.diplome.obtenuETRANGER SUPERIEUR
##                                               0.4333
##                notes$dernier.diplome.obtenuINGENIEUR
##                                               1.4333
##                  notes$dernier.diplome.obtenuLICENCE
```

```
##                                                 1.9667
```

On peut ensuite procéder à une ANOVA avec ce modèle :

```
anova(model)
```

```
## Analysis of Variance Table
##
## Response: notes$note.totale
##                               Df  Sum Sq Mean Sq F value    Pr(>F)
## notes$dernier.diplome.obtenu  11  393.77  35.797  3.2983 0.0002901 ***
## Residuals                    267 2897.80  10.853
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La p-value est relativement faible ici ($p = 0.002901$) : il semble donc il y avoir une relation entre le précédent niveau d'étude et le résultat final obtenu en SY02.

De manière plus générale, on peut aussi *réaliser une ANOVA avec plusieurs variables qualitatives* pour préciser l'analyse. Cette fois-ci, on choisit d'introduire la branch des étudiants, `specialite`, dans l'analyse :

```
model = lm(note.totale ~ 1 + dernier.diplome.obtenu + specialite, data = notes)
anova(model)
```

```
## Analysis of Variance Table
##
## Response: note.totale
##                          Df  Sum Sq Mean Sq F value   Pr(>F)
## dernier.diplome.obtenu   11  393.77  35.797  3.4930 0.000143 ***
## specialite                8  243.51  30.439  2.9702 0.003371 **
## Residuals               259 2654.29  10.248
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dans un autre registre, on peut essayer de voir s'il y a des correcteurs plus stricts au final. Cela se fait avec un test du χ^2 :

```
chisq.test(notes$correcteur.final, notes$note.final)
```

```
## Warning in chisq.test(notes$correcteur.final, notes$note.final): Chi-
## squared approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  notes$correcteur.final and notes$note.final
## X-squared = 224.24, df = 210, p-value = 0.2382
```

Ici le test de correlation peut être incorrect car les effectifs minimum nécessaire pour répondre au hypothèse de l'analyse ne sont pas suffisants – nous sommes avertis de cela par le message d'avertissement en orange. On peut utiliser un autre test que le test de base – qui est un test asymptotique – pour résoudre ce léger problème. Cela utilise des simulations à base de méthodes de Monte-Carlo;

```
chisq.test(notes$correcteur.final,
           notes$note.final,
           simulate.p.value = TRUE, # calcul des p-value via MonteCarlo
           B = 1e3) # nombre de simulations
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 1000
##  replicates)
```

```
##
## data:  notes$correcteur.final and notes$note.final
## X-squared = 224.24, df = NA, p-value = 0.2248
```

Il n'y a ici plus de degrés de liberté car l'on utilise une autre loi que celle du χ^2.

La relation entre note au final et correcteur n'est pas tellement évidente ici au regard de la p-value ($p = 0.2408$).

1.2 Analyse en composantes principales

On va s'intéresser aux correcteurs des notes de l'UV ici. Pour cela, on crée un nouveau jeu de données avec les moyennes et écarts-types par correcteur, et ce, pour le médian et le final.

```
# Calcul de moyennes et des écarts-types pour le médian
moy.median = aggregate(note.median ~ correcteur.median,
                       data = notes,
                       FUN = mean)
names(moy.median) = c("correcteur", "moy.median")
std.median = aggregate(note.median ~ correcteur.median,
                       data = notes,
                       FUN = sd)
names(std.median) = c("correcteur", "std.median")
median = merge(moy.median, std.median)

# Calcul de moyennes et des écarts-types pour le final
moy.final = aggregate(note.final ~ correcteur.final,
                      data = notes,
                      FUN = mean)
names(moy.final) = c("correcteur", "moy.final")
std.final = aggregate(note.final ~ correcteur.final,
                      data = notes,
                      FUN = sd)
names(std.final) = c("correcteur", "std.final")
final = merge(moy.final, std.final)

# Construction du jeu de données
correcteurs = merge(median, final, all = T)
correcteurs
```

```
##   correcteur moy.median std.median moy.final std.final
## 1       Cor1   10.70833   3.900715  10.94000  4.583303
## 2       Cor2   12.01042   3.712385        NA        NA
## 3       Cor3         NA         NA  12.57292  3.648068
## 4       Cor4   10.23469   3.043268  13.43478  4.343077
## 5       Cor5   10.97959   4.413473  11.82979  3.971743
## 6       Cor6   11.50000   4.303584  13.41489  4.877097
## 7       Cor7   10.12245   4.030522  11.90426  4.444878
## 8       Cor8   10.74000   4.646056  11.39583  4.872235
```

Les correcteurs `Cor3` et `Cor2` n'ont respectivement pas corrigé le médian et le final. On les écarte donc dans un premier temps du jeu de données pour faire l'ACP et l'on considèrera le jeu de donnée `corr.acp`:

```
corr.acp <- correcteurs[-c(2,3),]
corr.acp
```

```
##   correcteur moy.median std.median moy.final std.final
```

```
## 1       Cor1   10.70833   3.900715  10.94000  4.583303
## 4       Cor4   10.23469   3.043268  13.43478  4.343077
## 5       Cor5   10.97959   4.413473  11.82979  3.971743
## 6       Cor6   11.50000   4.303584  13.41489  4.877097
## 7       Cor7   10.12245   4.030522  11.90426  4.444878
## 8       Cor8   10.74000   4.646056  11.39583  4.872235
```

Dans l'ACP, on va donner la même importance aux individus (ici les correcteurs). Cela se traduit par une pondération uniforme où l'on utilisera – sans les faire techniquement apparaître – les matrices M et D_p comment étant respectivement $M = I_p$ et $D_p = \frac{1}{n} I_n$.

Pour réaliser une ACP, *il faut normaliser les données* pour avoir des distance de normalisation qui ont ensuite du sens ou, du moins, pour que les données soient *commensurables*. Pour cela, on centre d'abord les données et on les réduit par leur écarts types empiriques.

Si cette procédure est généralement à réaliser dans la plupart des cas, on peut ici l'omettre car les données sont déjà commensurables – puisqu'il s'agit de notations. Cette normalisation s'opère en R avec la fonction `scale`. Calculons ainsi la matrice de covariance empirique:

```
corr.acp.quantit = corr.acp[,-c(1)] # retrait des identifiants
X = scale(corr.acp.quantit,
          scale = FALSE) # Pour centrer sans réduire
matriceCov = cov.wt(X,method = "ML")$cov
matriceCov
```

```
##            moy.median  std.median   moy.final  std.final
## moy.median 0.21145165  0.13437397  0.07099528 0.04551820
## std.median 0.13437397  0.26460641 -0.22554848 0.04525501
## moy.final  0.07099528 -0.22554848  0.90772542 0.01270860
## std.final  0.04551820  0.04525501  0.01270860 0.09883305
```

On peut aussi directement calculer celle-ci avec sa définition

$$\Sigma_X = \frac{1}{n} X^T X$$

Cela se fait facilement ainsi :

```
matCov = t(X) %*% X / nrow(X)
matCov
```

```
##            moy.median  std.median   moy.final  std.final
## moy.median 0.21145165  0.13437397  0.07099528 0.04551820
## std.median 0.13437397  0.26460641 -0.22554848 0.04525501
## moy.final  0.07099528 -0.22554848  0.90772542 0.01270860
## std.final  0.04551820  0.04525501  0.01270860 0.09883305
```

```
# Vérification de l'égalité
all(round(matCov,5) == round(matriceCov,5))
```

```
## [1] TRUE
```

De manière générale, la matrice de covariance Σ est *définie*:

$$\forall \alpha \in \mathbb{R}^p,\ \alpha^T \Sigma_X \alpha = 0 \Rightarrow \alpha = 0_{\mathbb{R}^p}$$

Elle est aussi généralement – sans cas dégénérée –*semie-définie positive* :

$$\forall \alpha \in \mathbb{R}^p, \alpha^T \Sigma_X \alpha \geq 0$$

On a la relation :

$$\begin{aligned}\mathbb{V}\mathrm{ar}[\alpha^T X] &= \mathbb{E}[(\alpha^T X - \mathbb{E}[\alpha^T X])(\alpha^T X - \mathbb{E}[\alpha^T X])^T] \\ &= \mathbb{E}[\alpha^T (X - \mathbb{E}[X])(X - \mathbb{E}[X])^T \alpha] \\ &= \alpha^T \mathbb{V}\mathrm{ar}[X]\alpha \\ &= \alpha^T \Sigma \alpha\end{aligned}$$

On peut, à partir de cette matrice de covariance, effectuer l'ACP. Cela se fait relativement bien en calculant les valeurs et vecteurs propres de la matrice de covariance. Cette matrice étant symétrique, on peut indiquer cela pour utiliser une stratégie différente pour le calcul des valeurs et vecteurs propres.

```
eigenCov = eigen(matriceCov,
                 symmetric = TRUE) # pour utiliser une strategie adaptée
valeursPropres = eigenCov$values
totalValeurs = sum(valeursPropres)
# Base de vecteurs propres
U = eigenCov$vectors
```

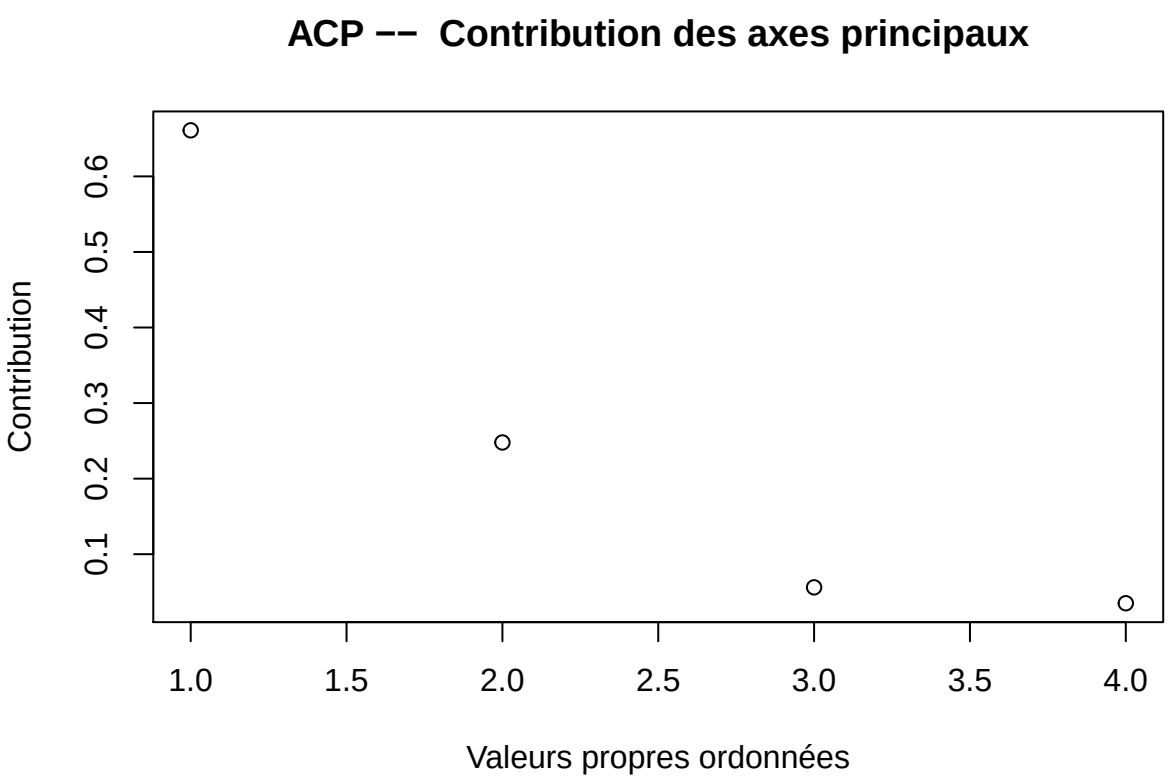
On peut calculer la *contribution de chacun des axes* dans l'explication de l'inertie de $\mathcal{N}_X$:

```
contributionsAxes = valeursPropres / sum(valeursPropres)
contributionsAxes
```

```
## [1] 0.66095613 0.24786460 0.05610544 0.03507382
```

Dressons

```
plot(contributionsAxes,
     xlab = "Valeurs propres ordonnées",
     ylab = "Contribution",
     main = "ACP --  Contribution des axes principaux")
```



Le premier axe explique, à lui seul, plus de la moitié de l'inertie. Les deux premiers axes réunits explique quasiment l'intégralité de l'inertie.

On a aussi obtenu les vecteurs propres de Σ_X et ceux-ci, représentés dans la matrice `U`, *permettent de changer de base pour se placer dans une meilleur représentation.*

```
U
```

```
##               [,1]       [,2]        [,3]        [,4]
## [1,] -0.0368252505 -0.7043052 -0.23322821  0.66947937
## [2,]  0.2941744324 -0.6469013 -0.09425237 -0.69720630
## [3,] -0.9550418742 -0.1719624 -0.02061448 -0.24062211
## [4,] -0.0005681381 -0.2364356  0.96762396  0.08832752
```

On peut se placer dans la base des vecteurs propres comme vu précédement ainsi:

```
XACP = X %*% U
XACP
```

```
##         [,1]       [,2]        [,3]        [,4]
## 1  1.1131294  0.2973223  0.10675046  0.40247613
## 4 -1.5041532  0.8133820  0.01415599  0.06168357
## 5  0.4045437 -0.2338454 -0.61494553 -0.04153965
## 6 -1.1613043 -1.0159209  0.11740385  0.08203415
## 7  0.2520651  0.4929044  0.07733933 -0.32451159
## 8  0.8957193 -0.3538424  0.29929590 -0.18014261
```

Si on calcule de nouveau la matrice de covariance, on peut cette fois-ci voir qu'elle est diagonale, avec comme coefficients diagonaux, la variance expliquée par chaque nouvel axe :

```
round(t(XACP) %*% XACP  / nrow(XACP), 5)
```

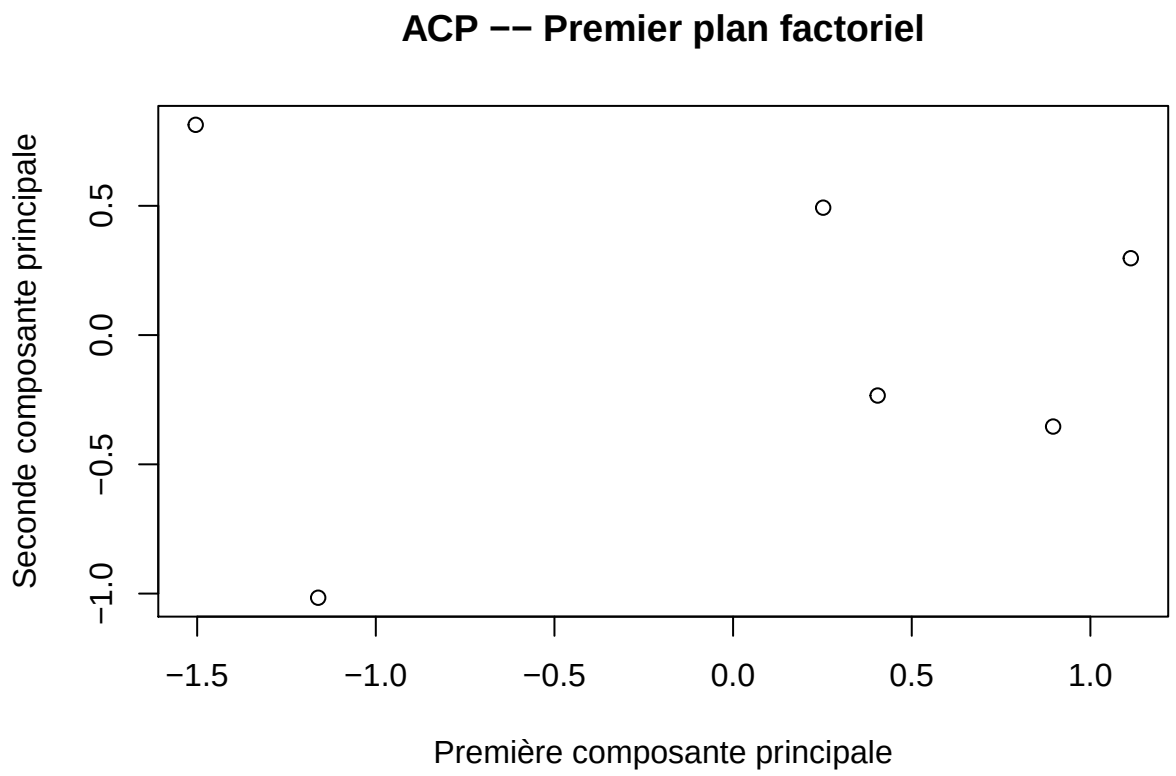
```
##         [,1]    [,2]    [,3]  [,4]
## [1,] 0.97994 0.00000 0.00000 0.000
## [2,] 0.00000 0.36749 0.00000 0.000
## [3,] 0.00000 0.00000 0.08318 0.000
## [4,] 0.00000 0.00000 0.00000 0.052
```

```
round(cov.wt(XACP,method = "ML")$cov, 5)
```

```
##         [,1]    [,2]    [,3]  [,4]
## [1,] 0.97994 0.00000 0.00000 0.000
## [2,] 0.00000 0.36749 0.00000 0.000
## [3,] 0.00000 0.00000 0.08318 0.000
## [4,] 0.00000 0.00000 0.00000 0.052
```

On peut projeter les individus sur le *premier plan factoriel* de cette nouvelle représentations, c'est à dire l'espace vectoriel contenant le deux premières composantes :

```
plot(XACP[,1],
     XACP[,2],
     xlab = "Première composante principale",
     ylab = "Seconde composante principale",
     main = "ACP -- Premier plan factoriel")
```



On peut tracer la représentation des quatre variables dans le premier plan factoriel avec la fonction `biplot` – voir dans la suite.

L'expression $\sum_{\alpha=1}^{k} \mathbf{c}_\alpha \mathbf{u}_\alpha^T$ pour des valeurs de $k \in \{1, 2, 3\}$ donne une recomposition partiel des données.

Pour $k = 4$, cette même expression redonne la matrice X initiale c'est à dire l'entièreté des données.

```
#n = nrow(XACP)
#Dp = diag(n) / n
#Dp
#dim(Dp)
#dim(U)
#dim(XACP)
#numer = t(XACP) %*% Dp %*% U
```

2 ACP avec R

R dispose bien sûr de fonctions qui permettent d'effectuer des ACP comme `princomp`, `loadings`, `plot` et `biplot`. Passons les en revues rapidement:

```
ACP = princomp(X)
ACP
```

```
## Call:
## princomp(x = X)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3    Comp.4
## 0.9899215 0.6062080 0.2884144 0.2280373
##
##  4  variables and  6 observations.
```

Le résumé sur l'ACP donne les informations princiaples dont nous avons besoin, à savoir, pour chaque vecteur propre :

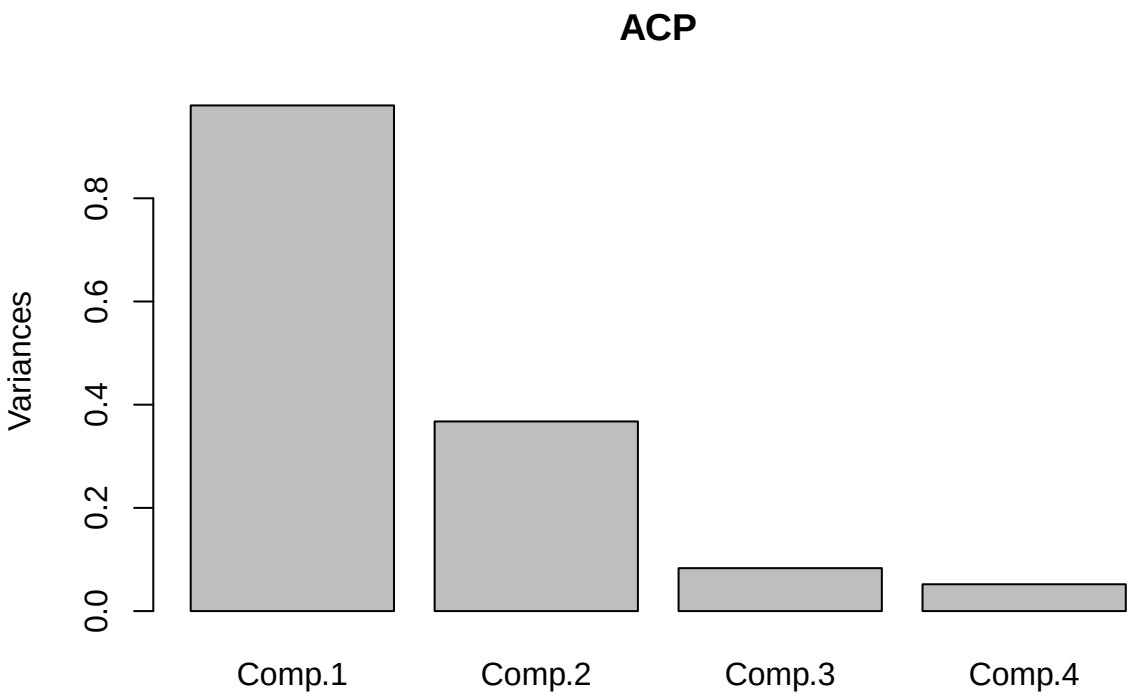
— sa valeur propre;
— sa proportion de contribution à la l'explication de la variance;
— la proportion cumulée à cette même contribution.

```
summary(ACP)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4
## Standard deviation     0.9899215 0.6062080 0.28841438 0.22803734
## Proportion of Variance 0.6609561 0.2478646 0.05610544 0.03507382
## Cumulative Proportion  0.6609561 0.9088207 0.96492618 1.00000000
```

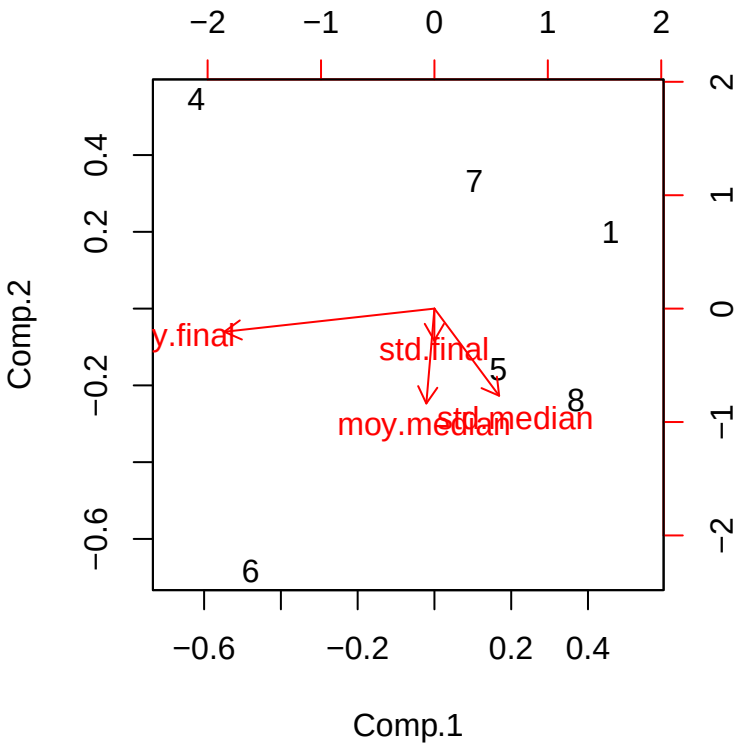
Ploter l'ACP donne un diagramme en boîte pas très intéressant des valeurs propres :

```
plot(ACP)
```



On préfèrera utiliser `biplot` pour afficher les vecteurs de la base initial dans la base de l'ACP:

```
biplot(ACP)
```



On peut voir que l'on peut facilement retrouver ces résultats graphiques en regardant les deux premiers vecteurs propres : chaque ligne représente les composantes de la base initiale dans le graphique plus haut:

```
U
```

```
##               [,1]       [,2]        [,3]        [,4]
## [1,] -0.0368252505 -0.7043052 -0.23322821  0.66947937
## [2,]  0.2941744324 -0.6469013 -0.09425237 -0.69720630
## [3,] -0.9550418742 -0.1719624 -0.02061448 -0.24062211
## [4,] -0.0005681381 -0.2364356  0.96762396  0.08832752
```

`loadings` donne des informations supplémentaires sur l'ACP:

```
loadings(ACP)
```

```
##
## Loadings:
##             Comp.1 Comp.2 Comp.3 Comp.4
## moy.median          -0.704 -0.233  0.669
## std.median   0.294 -0.647        -0.697
## moy.final   -0.955 -0.172        -0.241
## std.final           -0.236  0.968
##
##                Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
```

Il existe aussi un autre moyen de réaliser une ACP en utilisant `prcomp` :

```r
prcomp(X)
```

```
## Standard deviations (1, .., p=4):
## [1] 1.0844046 0.6640676 0.3159421 0.2498024
##
## Rotation (n x k) = (4 x 4):
##                       PC1        PC2         PC3         PC4
## moy.median  0.0368252505 -0.7043052  0.23322821 -0.66947937
## std.median -0.2941744324 -0.6469013  0.09425237  0.69720630
## moy.final   0.9550418742 -0.1719624  0.02061448  0.24062211
## std.final   0.0005681381 -0.2364356 -0.96762396 -0.08832752
```

3 Données Crabs

On va s'intéresser à un jeu de données assez connu est disponible dans la bibliothèque `MASS`. Le jeu de données `crabs` représentant 200 crabes décrit par huit variables, trois étant qualitatives et 5 étant quantitatives. Dans l'ACP, on travaillera avec une sous partie – nommées `crabsquant` – du jeu de données `crabs` où l'on se restreindra aux variables quantitatives.

```
library(MASS)
data(crabs)
crabsquant <- crabs[,4:8]
head(crabsquant)
```

```
##     FL  RW   CL   CW  BD
## 1  8.1 6.7 16.1 19.0 7.0
## 2  8.8 7.7 18.1 20.8 7.4
## 3  9.2 7.8 19.0 22.4 7.7
## 4  9.6 7.9 20.1 23.1 8.2
## 5  9.8 8.0 20.3 23.0 8.2
## 6 10.8 9.0 23.0 26.5 9.8
```

```
head(crabs)
```

```
##   sp sex index   FL  RW   CL   CW  BD
## 1  B   M     1  8.1 6.7 16.1 19.0 7.0
## 2  B   M     2  8.8 7.7 18.1 20.8 7.4
## 3  B   M     3  9.2 7.8 19.0 22.4 7.7
## 4  B   M     4  9.6 7.9 20.1 23.1 8.2
## 5  B   M     5  9.8 8.0 20.3 23.0 8.2
## 6  B   M     6 10.8 9.0 23.0 26.5 9.8
```

3.1 Analyse exploratoire

On peut faire une analyse multivariée comme on a l'habitude.

```
summary(crabs)
```

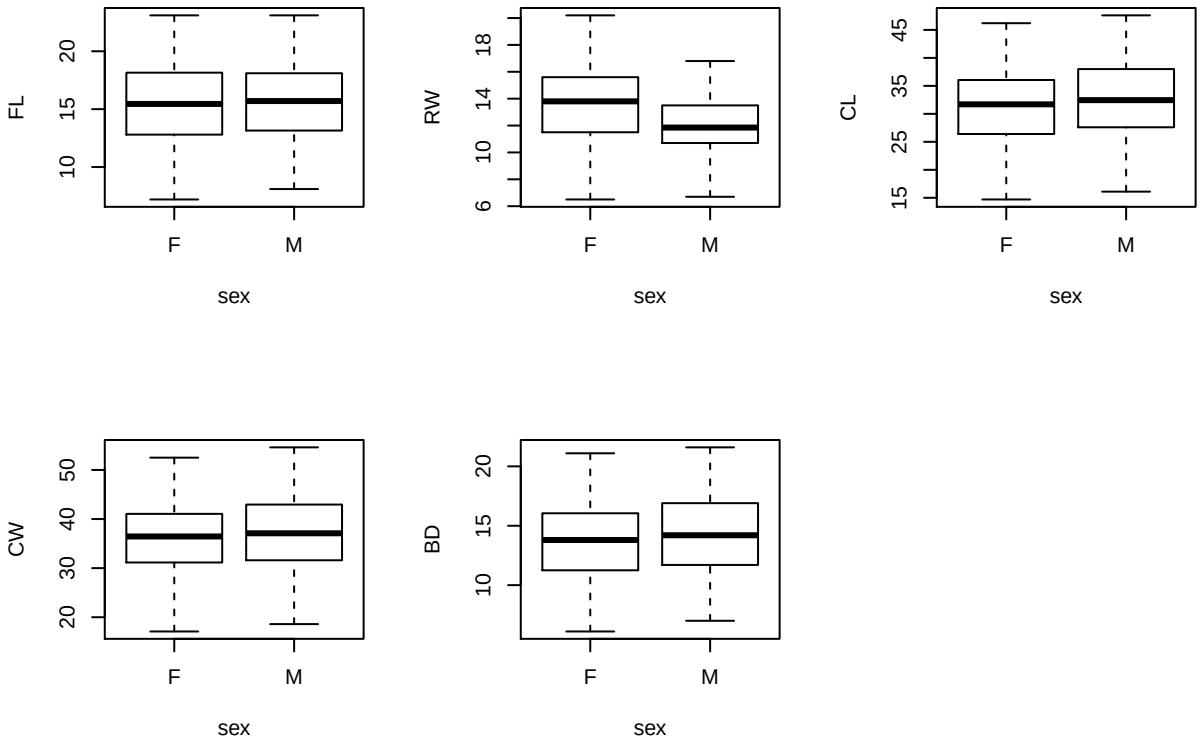
```
##  sp      sex         index            FL              RW
##  B:100   F:100   Min.   : 1.0   Min.   : 7.20   Min.   : 6.50
##  O:100   M:100   1st Qu.:13.0   1st Qu.:12.90   1st Qu.:11.00
##                  Median :25.5   Median :15.55   Median :12.80
##                  Mean   :25.5   Mean   :15.58   Mean   :12.74
##                  3rd Qu.:38.0   3rd Qu.:18.05   3rd Qu.:14.30
##                  Max.   :50.0   Max.   :23.10   Max.   :20.20
##        CL              CW              BD
##  Min.   :14.70   Min.   :17.10   Min.   : 6.10
##  1st Qu.:27.27   1st Qu.:31.50   1st Qu.:11.40
##  Median :32.10   Median :36.80   Median :13.90
##  Mean   :32.11   Mean   :36.41   Mean   :14.03
##  3rd Qu.:37.23   3rd Qu.:42.00   3rd Qu.:16.60
##  Max.   :47.60   Max.   :54.60   Max.   :21.60
```

On peut s'interroger notamment sur *les différences de caractéristiques morphologiques, en particulier selon l'espèce ou le sexe.*

3.1.1 Analyse multivariée en fonction du sexe

Traçons des diagrammes à moustache de quelques variables en fonction du sexe des crabes:

```
cols = names(crabs)
def.par <- par(no.readonly=T)
par(mfrow=c(2,3))
for(i in 4:8) {
  plot(crabs[,i]~crabs$sex, xlab = "sex", ylab = cols[i])
}
```



Il semble y avoir une différence notable entre les femelles et les mâles sur le paramètre `RW`. Aussi, il semble qu'il y a une légère différence sur les facteurs `CW`, `CL` et `BD` en fonction du sex.

Pour confirmer ou infirmer ces observations, on peut procéder à des ANOVA:

```
modelRW = lm(RW~1+sex, data = crabs)
modelRW
```

```
##
## Call:
## lm(formula = RW ~ 1 + sex, data = crabs)
##
## Coefficients:
## (Intercept)         sexM
##      13.487       -1.497
```

```
anova(modelRW)
```

```
## Analysis of Variance Table
```

```
##
## Response: RW
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## sex         1  112.05  112.05    18.4 2.797e-05 ***
## Residuals 198 1205.74    6.09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pour le paramètre `RW`, on se rend effectivement qu'il est dépendant du sexe.

Faisons de même pour les autres paramètres

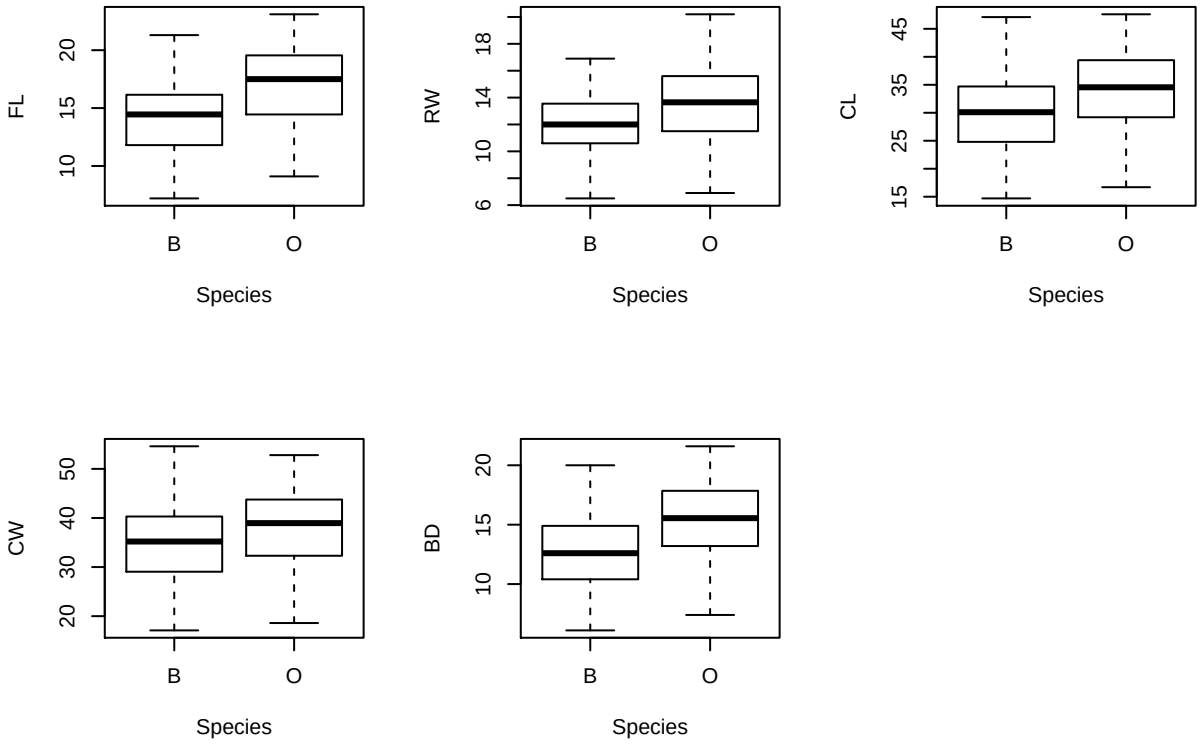
```
cols = names(crabs)
for(i in 4:8) {
  model = lm(crabs[,i] ~ 1 + sex, data = crabs)
  print(anova(model))
}
```

```
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df  Sum Sq Mean Sq F value Pr(>F)
## sex         1    4.56  4.5602  0.3721 0.5426
## Residuals 198 2426.68 12.2560
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## sex         1  112.05  112.05    18.4 2.797e-05 ***
## Residuals 198 1205.74    6.09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df Sum Sq Mean Sq F value Pr(>F)
## sex         1  111.2 111.154  2.2066  0.139
## Residuals 198 9974.1  50.374
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df  Sum Sq Mean Sq F value Pr(>F)
## sex         1    68.3  68.328  1.1032 0.2948
## Residuals 198 12263.2  61.936
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df  Sum Sq Mean Sq F value Pr(>F)
## sex         1   18.79  18.788  1.6068 0.2064
## Residuals 198 2315.30  11.693
```

On voit que la différence n'est significative que pour le paramètre `RW`.

3.1.2 Analyse multivariée en fonction de l'espèce

```
cols = names(crabs)
def.par <- par(no.readonly=T)
par(mfrow=c(2,3))
for(i in 4:8) {
  plot(crabs[,i]~crabs$sp, xlab = "Species", ylab = cols[i])
}
```



Visuellement, on voit que les différentes sont flagrantes entre espèces pour les variables quantitatives, les attributs des individus de l'espèce `O` possédant des valeurs globalement plus élevées.

Cela se confirme par les ANOVAS:

```
cols = names(crabs)
for(i in 4:8) {
  model = lm(crabs[,i] ~ 1 + sp, data = crabs)
  print(anova(model))
}
```

```
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## sp          1  466.35  466.35  46.993 8.842e-11 ***
## Residuals 198 1964.90    9.92
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## sp          1  131.38 131.382  21.926 5.252e-06 ***
## Residuals 198 1186.41   5.992
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df Sum Sq Mean Sq F value    Pr(>F)
## sp          1  838.5  838.45  17.953 3.468e-05 ***
## Residuals 198 9246.9   46.70
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df  Sum Sq Mean Sq F value   Pr(>F)
## sp          1   576.3  576.30  9.7069 0.002108 **
## Residuals 198 11755.3   59.37
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: crabs[, i]
##            Df  Sum Sq Mean Sq F value   Pr(>F)
## sp          1  419.05  419.05  43.327 4.06e-10 ***
## Residuals 198 1915.03    9.67
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.1.3 Corrélation entre variables

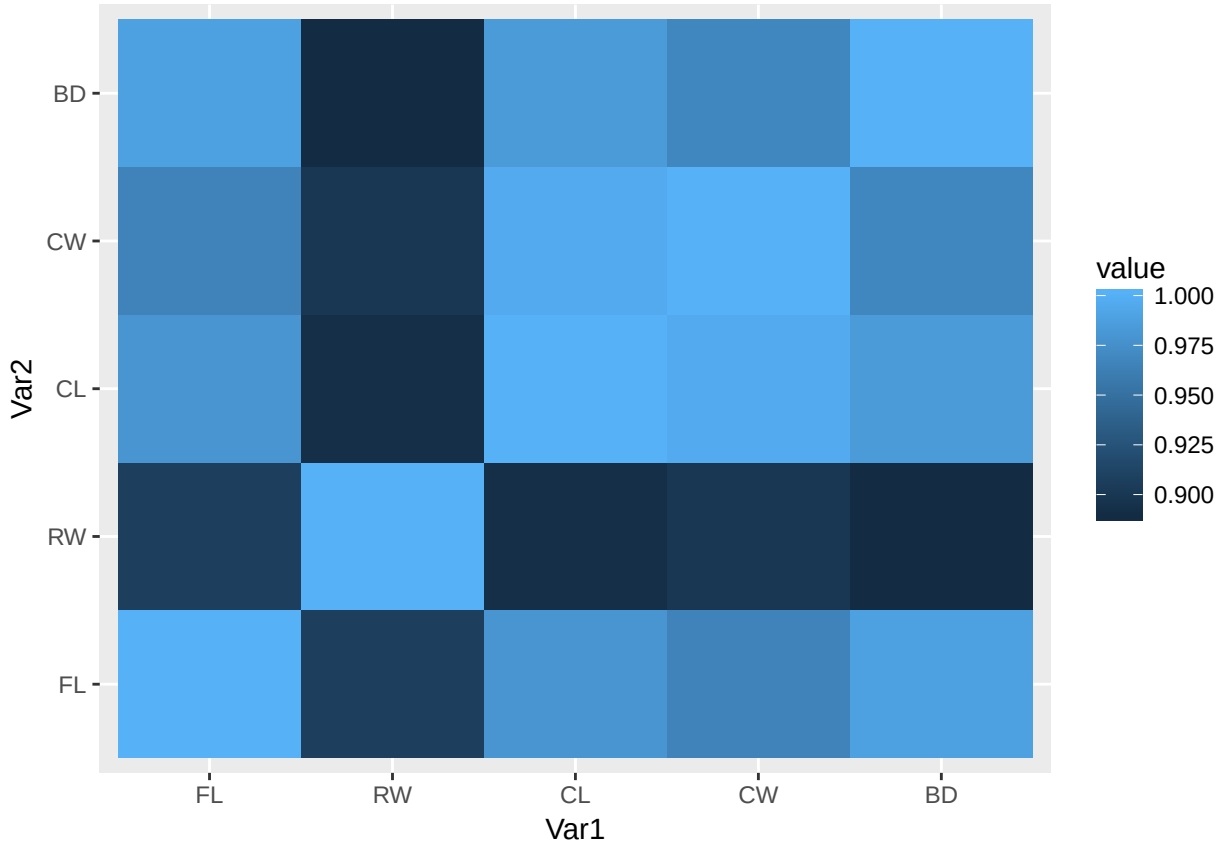
On peut calculer la matrice de corrélation.

```
corMat = cor(crabsquant)
corMat
```

```
##           FL        RW        CL        CW        BD
## FL 1.0000000 0.9069876 0.9788418 0.9649558 0.9876272
## RW 0.9069876 1.0000000 0.8927430 0.9004021 0.8892054
## CL 0.9788418 0.8927430 1.0000000 0.9950225 0.9832038
## CW 0.9649558 0.9004021 0.9950225 1.0000000 0.9678117
## BD 0.9876272 0.8892054 0.9832038 0.9678117 1.0000000
```

Et faire une *heat-map* de celle-ci:

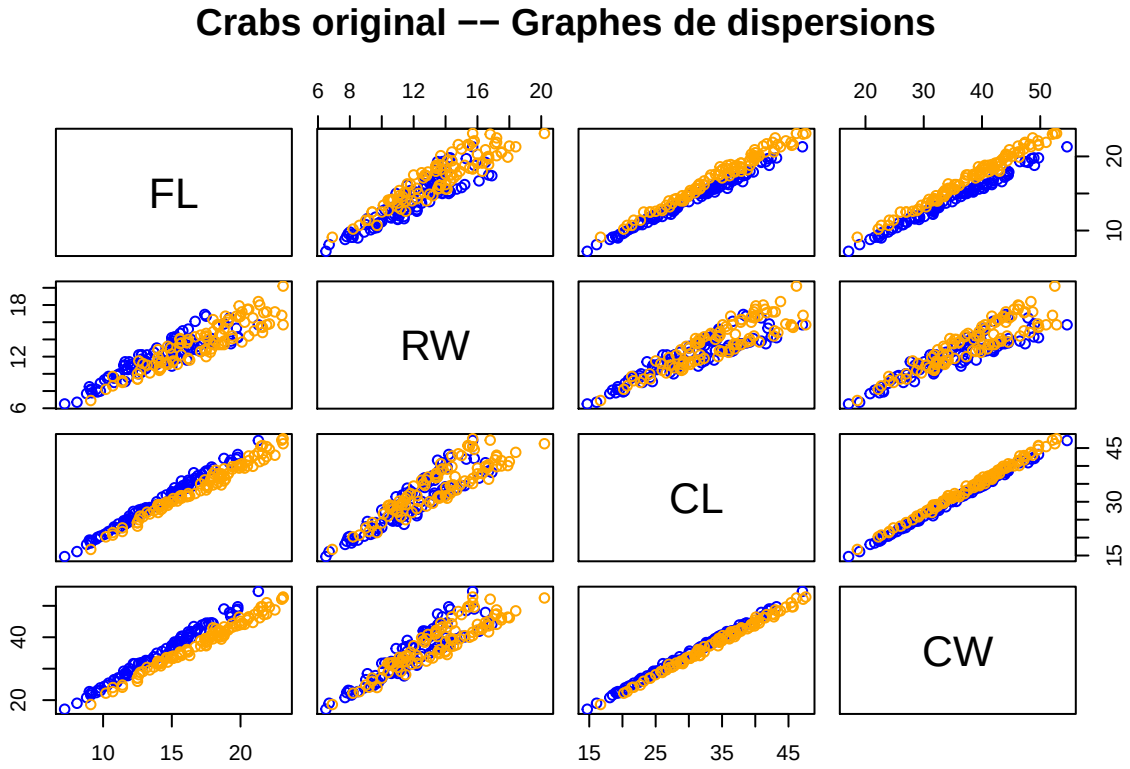
```
library(ggplot2)
library(reshape2)
meltedCormat <- melt(corMat)
ggplot(data = meltedCormat, aes(x=Var1, y=Var2, fill=value)) + geom_tile()
```



Les variables semblent être de manière générale très corrélées. Cela s'explique car les données dépendent de la taille du crabe non directement observée ici (voir `?crabs`).

Cela se voit aussi graphiquement avec les graphes de dispersions:

```
pairs(crabsquant[,1:4],
     main = "Crabs original -- Graphes de dispersions",
     col = c("blue","orange")[crabs$sp])
```



On peut, pour s'affranchir de ces corrélations effectuer une ACP.

3.2 ACP

On effectue l'ACP comme suit:

```
# Normalisation par centrage et réduction
Xcrabs = scale(crabsquant, scale = TRUE)
covarMatCrabs = cov(Xcrabs)
Ucrabs = eigen(covarMatCrabs)$vectors
Ucrabs
```

```
##            [,1]       [,2]        [,3]         [,4]        [,5]
## [1,] -0.4520437 -0.1375813  0.53076841  0.696923372  0.09649156
## [2,] -0.4280774  0.8981307 -0.01197915 -0.083703203 -0.05441759
## [3,] -0.4531910 -0.2682381 -0.30968155 -0.001444633 -0.79168267
## [4,] -0.4511127 -0.1805959 -0.65256956  0.089187816  0.57452672
## [5,] -0.4511336 -0.2643219  0.44316103 -0.706636423  0.17574331
```

et changer de base :

```
XcrabsACP = Xcrabs %*% Ucrabs
head(XcrabsACP)
```

```
##       [,1]        [,2]        [,3]         [,4]         [,5]
## 1 4.915239 -0.26777335  0.12195173 -0.039045942  0.069295183
## 2 4.375197 -0.09383811  0.03913369  0.005453536 -0.003044597
## 3 4.118329 -0.16845321 -0.03355942  0.038001540  0.037965533
```

```
## 4 3.873960 -0.24539253 -0.01446472  0.019045937  0.001311681
## 5 3.824458 -0.22360515  0.01502960  0.054497120 -0.024821663
## 6 2.945563 -0.21946999 -0.03833196 -0.069665684  0.018926413
```

On peut vérifier que les variables ne sont plus corrélées dans cette nouvelle base :

```
round(cor(XcrabsACP),5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

On utilise le résumé pour avoir des informations sur l'ACP, et l'on se rend compte que le nouvel axe explique la quasi-intégralité de la variance observée.

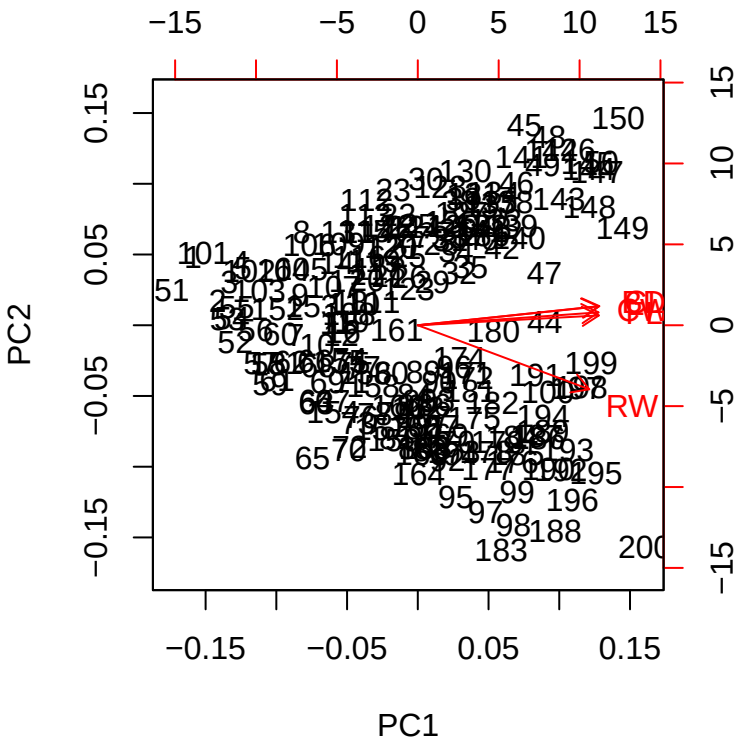
```
ACPcrabs = prcomp(Xcrabs)
summary(ACPcrabs)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5
## Standard deviation     2.1883 0.38947 0.21595 0.10552 0.04137
## Proportion of Variance 0.9578 0.03034 0.00933 0.00223 0.00034
## Cumulative Proportion  0.9578 0.98810 0.99743 0.99966 1.00000
```

```
Ucrabs
```

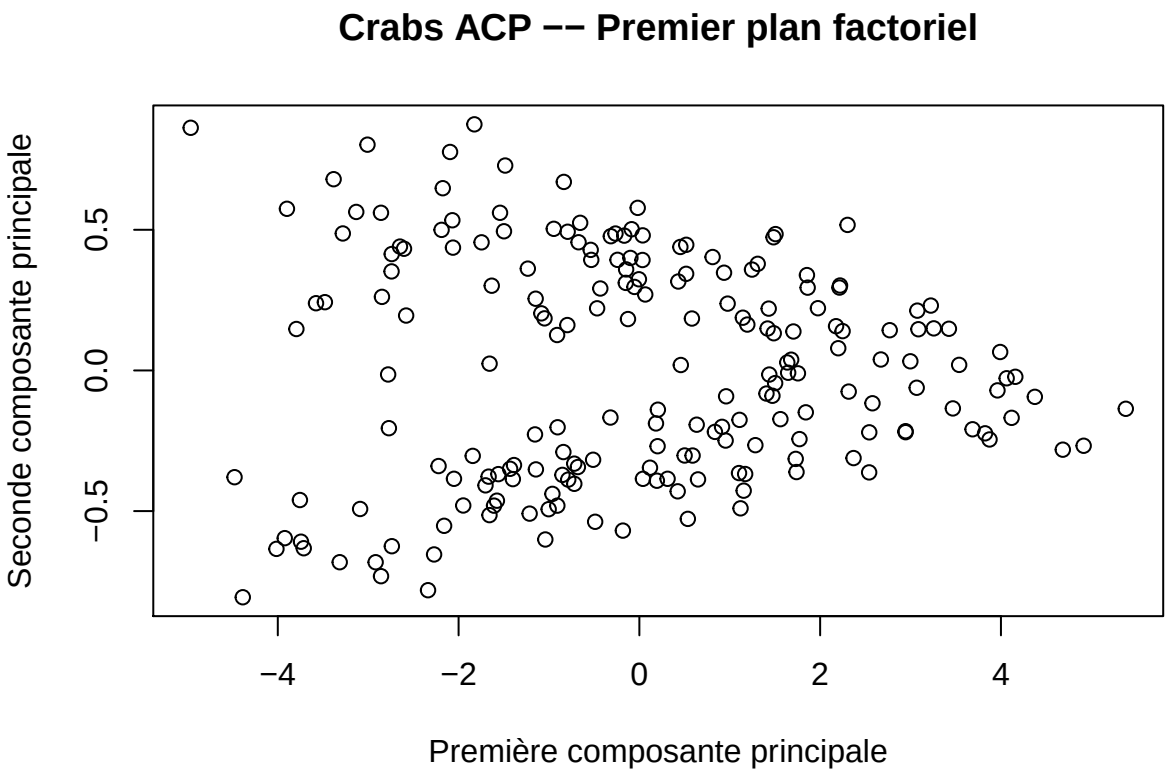
```
##            [,1]       [,2]        [,3]          [,4]        [,5]
## [1,] -0.4520437 -0.1375813  0.53076841  0.696923372  0.09649156
## [2,] -0.4280774  0.8981307 -0.01197915 -0.083703203 -0.05441759
## [3,] -0.4531910 -0.2682381 -0.30968155 -0.001444633 -0.79168267
## [4,] -0.4511127 -0.1805959 -0.65256956  0.089187816  0.57452672
## [5,] -0.4511336 -0.2643219  0.44316103 -0.706636423  0.17574331
```

```
biplot(ACPcrabs)
```



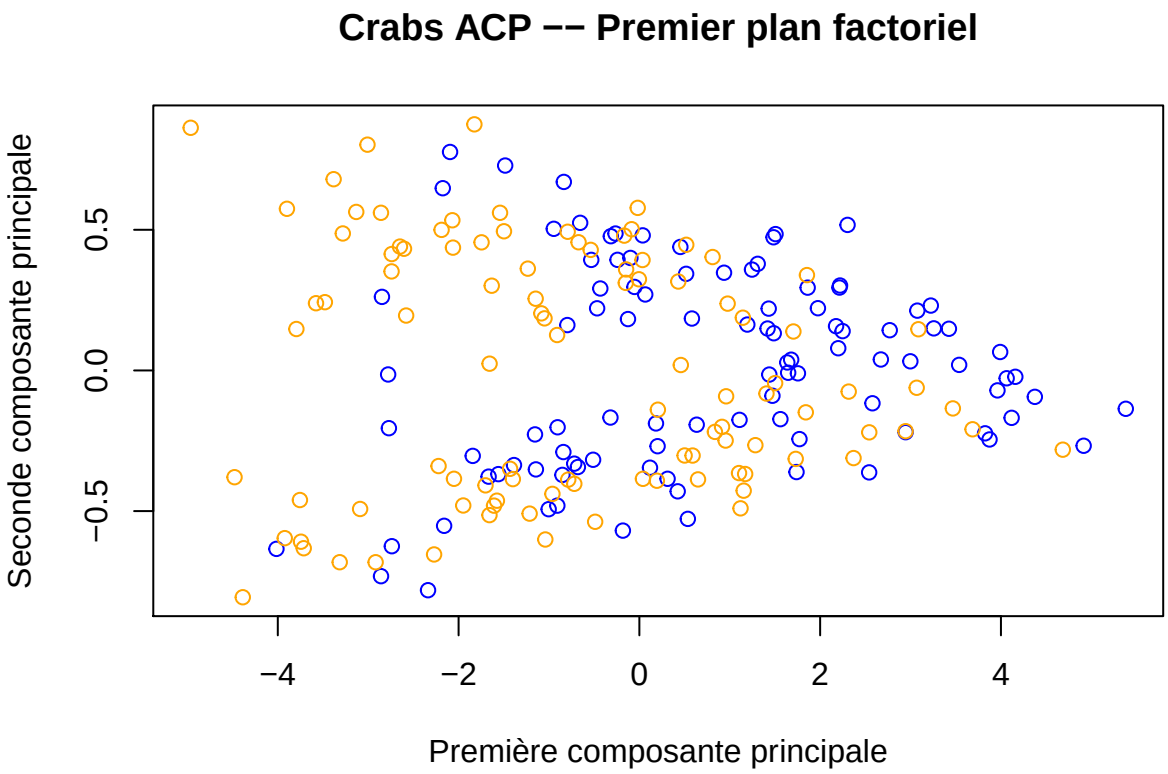
On peut visualiser la projection de nos individus sur le premier plan factoriel:

```
plot(XcrabsACP[,1],
     XcrabsACP[,2],
     xlab = "Première composante principale",
     ylab = "Seconde composante principale",
     main = "Crabs ACP -- Premier plan factoriel")
```



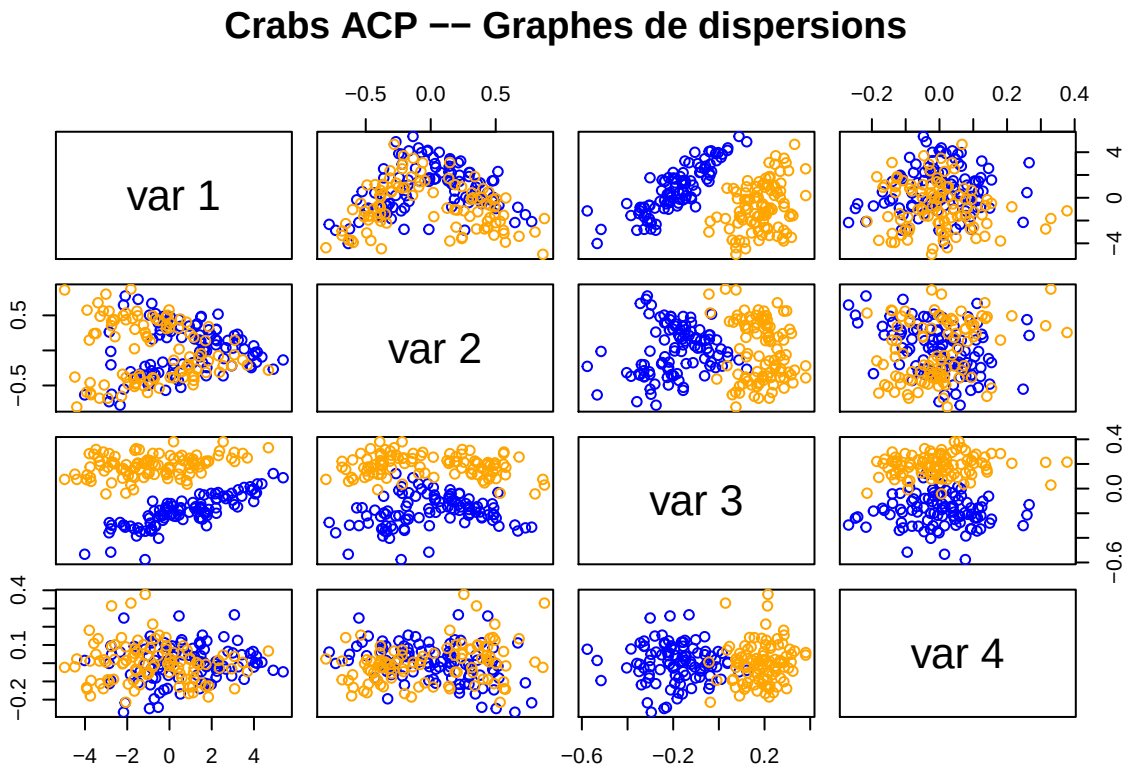
Pour visualiser les données en fonction des classes, on peut rajouter des couleurs pour différencier les individus dans leurs classes. On réutilise pour cela les valeurs de `crabs$sp` pour discriminer.

```
plot(XcrabsACP[,1],
     XcrabsACP[,2],
     xlab = "Première composante principale",
     ylab = "Seconde composante principale",
     main = "Crabs ACP -- Premier plan factoriel",
     col = c("blue","orange")[crabs$sp])
```



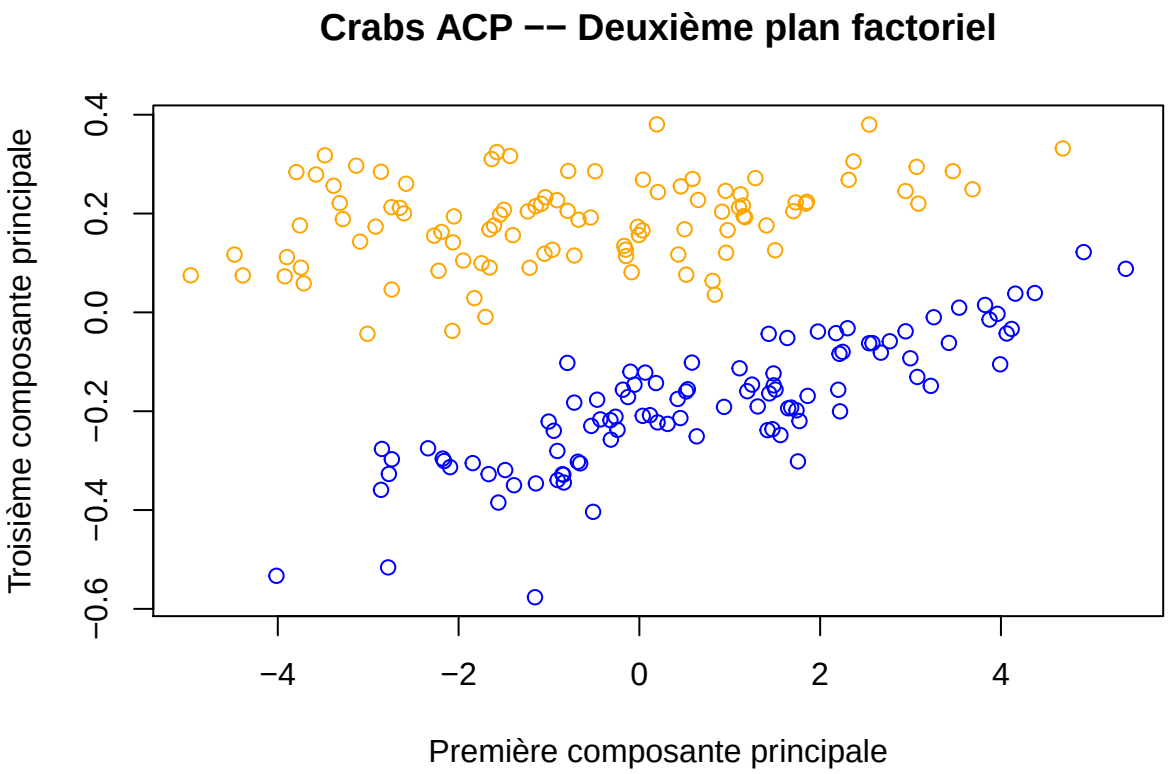
On peut aussi générer les graphes de dispersions pour les variables:

```
pairs(XcrabsACP[,1:4],
      main = "Crabs ACP -- Graphes de dispersions",
      col = c("blue","orange")[crabs$sp])
```



On voit que les données semblent être *linéairement séparables* sur les projections sur les plans où l'axe 3 est considéré. Voyez plutôt:

```
plot(XcrabsACP[,1],
     XcrabsACP[,3],
     xlab = "Première composante principale",
     ylab = "Troisième composante principale",
     main = "Crabs ACP -- Deuxième plan factoriel",
     col = c("blue","orange")[crabs$sp])
```



4 Données Pima

Dans cette dernière partie, on va étudier le jeu de données Pima disponible dans le fichier `Pima.csv`. Celui-ci contient 532 enregistrements d'individus de sexe féminin décrits par huit variables, une étant qualitative:

- nombre de grossesses, `npreg`;
- taux plasmatique de glucose, `glu`;
- pression artérielle diastolique, `bp`;
- épaisseur du pli cutané au niveau du triceps, `skin`;
- indice de masse corporelle, `bmi`;
- fonction de pedigree du diabète, `ped`, la mesure de l'influence génétique espérée des proches, affectés ou non par le diabète, sur le risque éventuel du sujet,
- âge, `age`;
- catégorie,`z`, diabétique si , `z = 2`.

On charge le jeu de données ainsi:

```
pima = read.csv("donnees/Pima.csv", header=T)
pima$z = factor(pima$z)
head(pima)
```

```
##   npreg glu bp skin  bmi   ped age z
## 1     5  86 68   28 30.2 0.364  24 1
## 2     5  77 82   41 35.8 0.156  35 1
## 3     0 165 76   43 47.9 0.259  26 1
## 4     0 107 60   25 26.4 0.133  23 1
```

```
## 5     3  83 58   31 34.3 0.336  25 1
## 6     1 193 50   16 25.9 0.655  24 1
```

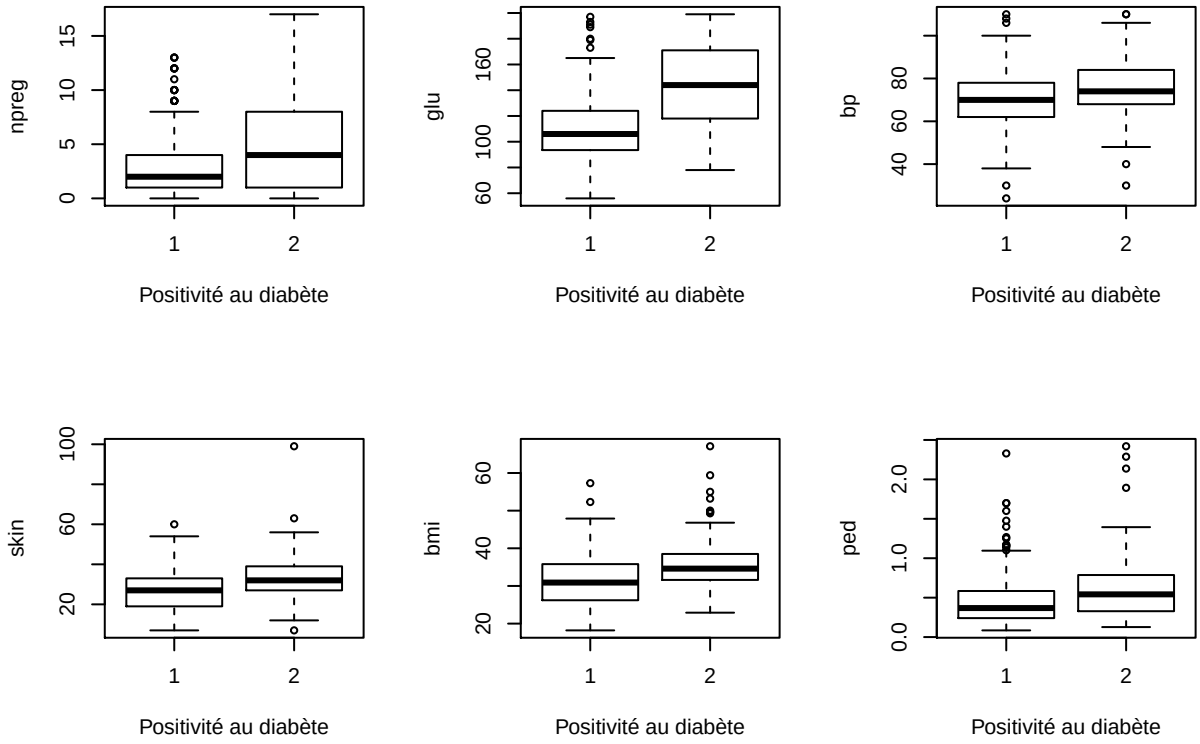
4.1 Analyse exploratoire

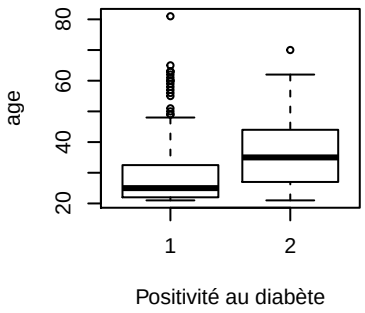
On connait la procédure.

```
summary(pima)
```

```
##      npreg             glu              bp              skin
##  Min.   : 0.000   Min.   : 56.00   Min.   : 24.00   Min.   : 7.00
##  1st Qu.: 1.000   1st Qu.: 98.75   1st Qu.: 64.00   1st Qu.:22.00
##  Median : 2.000   Median :115.00   Median : 72.00   Median :29.00
##  Mean   : 3.517   Mean   :121.03   Mean   : 71.51   Mean   :29.18
##  3rd Qu.: 5.000   3rd Qu.:141.25   3rd Qu.: 80.00   3rd Qu.:36.00
##  Max.   :17.000   Max.   :199.00   Max.   :110.00   Max.   :99.00
##       bmi              ped              age        z
##  Min.   :18.20   Min.   :0.0850   Min.   :21.00   1:355
##  1st Qu.:27.88   1st Qu.:0.2587   1st Qu.:23.00   2:177
##  Median :32.80   Median :0.4160   Median :28.00
##  Mean   :32.89   Mean   :0.5030   Mean   :31.61
##  3rd Qu.:36.90   3rd Qu.:0.6585   3rd Qu.:38.00
##  Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

```
cols = names(pima)
def.par <- par(no.readonly=T)
par(mfrow=c(2,3))
for(i in 1:7) {
  plot(pima[,i]~pima$z, xlab = "Positivité au diabète", ylab = cols[i])
}
```


npreg
0
5
10
15
1
2
Positivité au diabète
glu
60
100
160
1
2
Positivité au diabète
bp
40
60
80
1
2
Positivité au diabète
skin
20
60
100
1
2
Positivité au diabète
bmi
20
40
60
1
2
Positivité au diabète
ped
0.0
1.0
2.0
1
2
Positivité au diabète



Il semblarait bien qu'il existe un lien entre les variables et la positivité au diabète:

```
for(i in 1:7) {
  model = lm(pima[,i] ~ 1 + z, data = pima)
  print(anova(model))
}
```

```
## Analysis of Variance Table
##
## Response: pima[, i]
##            Df Sum Sq Mean Sq F value    Pr(>F)
## z           1  371.6  371.62  36.118 3.457e-09 ***
## Residuals 530 5453.2   10.29
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: pima[, i]
##            Df Sum Sq Mean Sq F value    Pr(>F)
## z           1 129417  129417   180.1 < 2.2e-16 ***
## Residuals 530 380848     719
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: pima[, i]
##            Df Sum Sq Mean Sq F value    Pr(>F)
```

```
## z           1   2708 2707.56  18.454 2.071e-05 ***
## Residuals 530  77761  146.72
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: pima[, i]
##            Df Sum Sq Mean Sq F value    Pr(>F)
## z           1   3820  3820.3  36.821 2.467e-09 ***
## Residuals 530  54989   103.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: pima[, i]
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## z           1  2276.4 2276.45  52.764 1.352e-12 ***
## Residuals 530 22866.2   43.14
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: pima[, i]
##            Df Sum Sq Mean Sq F value    Pr(>F)
## z           1  3.424  3.4243  30.445 5.379e-08 ***
## Residuals 530 59.612  0.1125
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Analysis of Variance Table
##
## Response: pima[, i]
##            Df Sum Sq Mean Sq F value   Pr(>F)
## z           1   6106  6105.7  58.422 9.99e-14 ***
## Residuals 530  55390   104.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2 ACP

Regardons si l'ACP nous permet d'obtenir une représentation visuelle qui permet de discriminer les individus.

```
# Normalisation par centrage et réduction
Xpima = scale(pima[,-8],scale = TRUE)
covarMat = cov(Xpima)
Upima = eigen(covarMat)$vectors
Upima
```

```
##            [,1]        [,2]         [,3]        [,4]        [,5]
## [1,] -0.3457261  0.55503349  0.005562932  0.36110259  0.15530959
## [2,] -0.3668726 -0.04859651 -0.327275843 -0.71779145  0.46953185
## [3,] -0.4038932  0.05827409  0.302422878 -0.38508615 -0.73891450
## [4,] -0.4328994 -0.40977680  0.188714763  0.33473503  0.24383968
## [5,] -0.4212959 -0.49130840  0.180006770  0.16085861  0.02380542
```

```
## [6,] -0.1618285 -0.18173488 -0.854735845  0.24276077 -0.38406416
## [7,] -0.4377986  0.49381352 -0.053018077  0.09608948  0.04352443
##              [,6]        [,7]
## [1,] -0.499198431  0.41068856
## [2,] -0.079834810  0.11779240
## [3,]  0.018522767  0.21768266
## [4,]  0.510436950  0.42073343
## [5,] -0.531539804 -0.48963929
## [6,]  0.006822988  0.06104082
## [7,]  0.448282545 -0.59125703
```

```
XpimaACP = Xpima %*% Upima
head(XpimaACP)
```

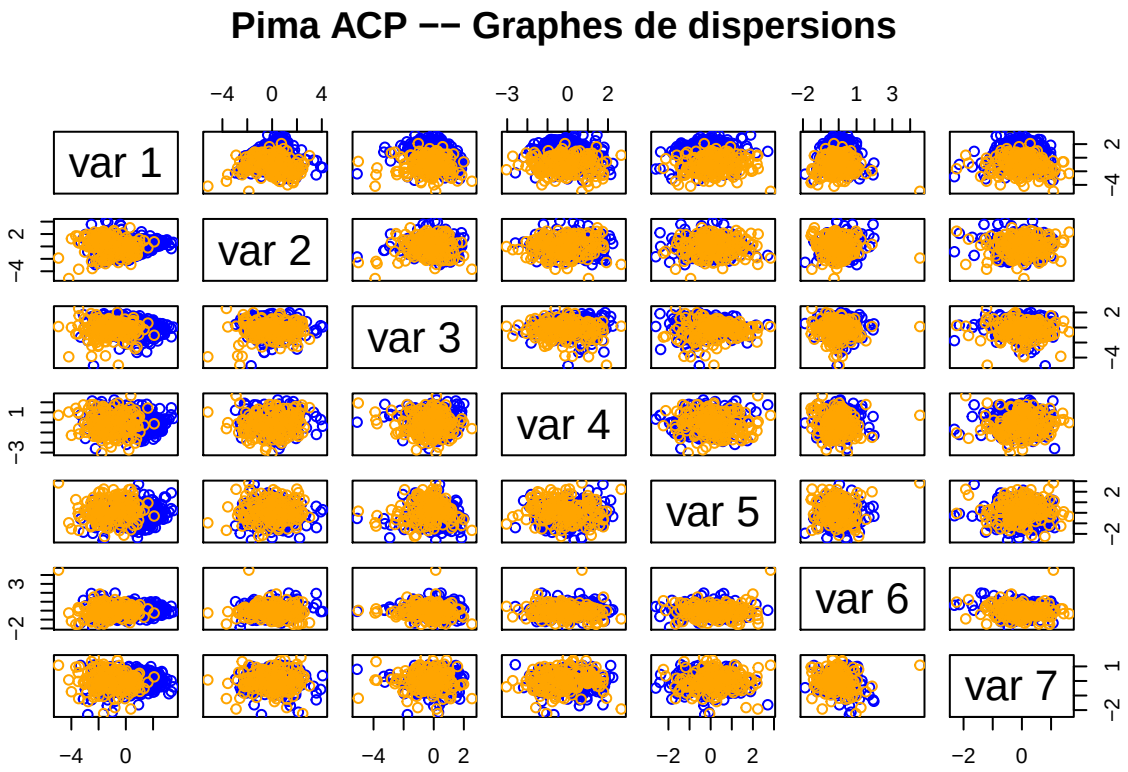
```
##            [,1]        [,2]       [,3]       [,4]       [,5]       [,6]
## [1,]  0.9631757  0.24886278  0.5768797  0.8160861 -0.1632100 -0.3080763
## [2,] -0.6170609  0.03768109  1.8572473  1.0826073 -0.5429390  0.3882190
## [3,] -1.4450865 -2.37569981  0.9136006 -0.9738060  0.8526295 -0.3043665
## [4,]  2.0042817 -0.19574353  0.5750183 -0.3209956  0.5713935  0.4812133
## [5,]  1.1335871 -0.47782976  0.5851106  1.1607815  0.4167588 -0.1440518
## [6,]  1.3251603 -0.05362819 -2.0512692 -1.8117259  1.7330463 -0.2519473
##              [,7]
## [1,]  0.526702206
## [2,]  0.220106990
## [3,] -0.439919982
## [4,]  0.009517358
## [5,] -0.141234404
## [6,] -0.003221412
```

```
summary(prcomp(Xpima))
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5     PC6     PC7
## Standard deviation     1.5220 1.2249 1.0035 0.8945 0.8467 0.59064 0.55700
## Proportion of Variance 0.3309 0.2143 0.1439 0.1143 0.1024 0.04984 0.04432
## Cumulative Proportion  0.3309 0.5453 0.6891 0.8034 0.9058 0.95568 1.00000
```

Même si le premier axe explique la majorité de la variance cette fois ci, la variance est dispersée sur tous les axes qui faut quasiement tous considérer.

```
pairs(XpimaACP[,1:7],
     main = "Pima ACP -- Graphes de dispersions",
     col = c("blue","orange")[pima$z])
```



Ici, il ne semble cette-fois ci pas y avoir une représentation simple qui permette de distinguer les deux catégories de patientes.

5 Données Mutations

Lorsque l'on est en présence d'un tableau invidus variable X, on peut utiliser l'ACP pour étudier ce même jeu de données dans un autre point de vue qui rend son étude plus pertinent ou plus simple.

Néanmoins, on peut être amené à travailler avec *un jeu de données* Δ^2 *qui représente une mesure de dissimilarité (ou de similarité)* à la place des valeurs de leur attributs. À partir d'un tel tableau, *on souhaiterait se ramener dans une représentation euclidienne des individus* : l'analyse factorielle en tableau de distance permet de réaliser cela.

Chargons les données et procédons à ces transformations.

```
mutOriginal = read.csv("donnees/mutations2.csv", header = TRUE, row.names = 1)
head(mutOriginal)
```

```
##         Man Monkey Dog Horse Donkey Pig Rabbit Kangaroo Pekin.Duck Pigeon
## Man       0      1  13    17     16  13     12       12         17     16
## Monkey    1      0  12    16     15  12     11       13         16     15
## Dog      13     12   0    10      8   4      6        7         12     12
## Horse    17     16  10     0      1   5     11       11         16     16
## Donkey   16     15   8     1      0   4     10       12         15     15
## Pig      13     12   4     5      4   0      6        7         13     13
##         Chicken King.Penguin Snapping.Turtle Rattlesnake Tuna Screwworm.Fly
## Man          18           18              19          20   31            33
```

```
## Monkey      17           17              18          21   32            32
## Dog         14           14              13          30   29            24
## Horse       16           17              16          32   27            24
## Donkey      15           16              15          31   26            25
## Pig         13           14              13          30   25            26
##         Moth Bakers.Mould Bread.Yeast Skin.Fungus
## Man       36           63          56          66
## Monkey    35           62          57          65
## Dog       28           64          61          66
## Horse     33           64          60          68
## Donkey    32           64          59          67
## Pig       31           64          59          67
```

On peut s'assurer que le donnée représentent bien des dissimilarités (voire à des distances) en procédant à cette transformation:

```
mut = as.dist(mutOriginal, diag = TRUE, upper = TRUE)
```

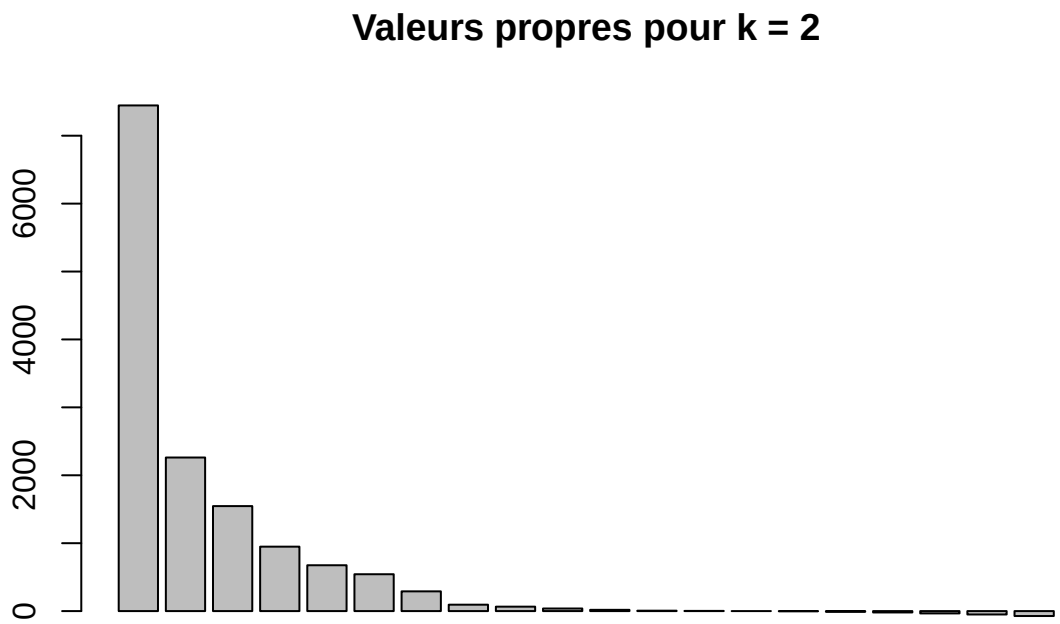
On peut ensuite procéder à un transformation par l'AFTD pour se rammener à une représentation euclidienne des données. On va pour commencer se rammener à une representation en deux dimensions, $k = 2$.

```
mds = cmdscale(mut, eig = TRUE, k = 2)
summary(mds)
```

```
##        Length Class  Mode
## points 40     -none- numeric
## eig    20     -none- numeric
## x       0     -none- NULL
## ac      1     -none- numeric
## GOF     2     -none- numeric
```

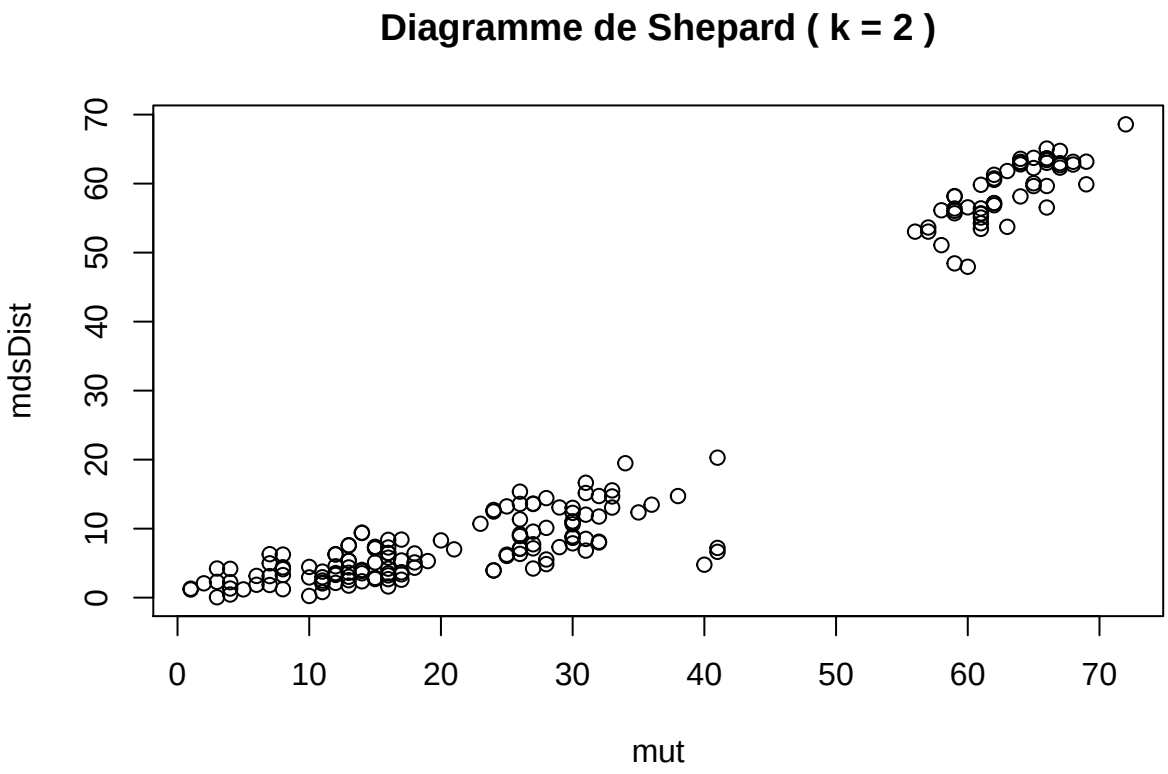
On peut représenter les valeurs propres en fonction de leur valeurs. On peut voir ici que les dernières valeurs propres sont nulles. Cela arrive lorsque la dissimilarité n'est pas une distance euclidienne. Néanmoins, on utilisera quand même cette méthode si leurs valeurs ne sont pas trop élevées.

```
barplot(mds$eig,main = "Valeurs propres pour k = 2")
```



Pour se faire une meilleure idée de la pertinence de k et surtout des résultats, on peut réaliser un *diagramme de Shepard*. Celui-ci représente la distance du positionement multidimensionnel (en ordonné) en fonction de la dissimilarité initiale. Plus les points se rapprochent de la première birectrice, plus les résultats sont bons.

```
mdsDist = dist(mds$points, method = "euclidian")
plot(x = mut,y = mdsDist,main = "Diagramme de Shepard ( k = 2 )")
```



On peut afficher les diagrammes pour plusieurs valeurs de k. Ici on fera varier k dans $\{2, 4, 6, 8\}$

```
def.par <- par(no.readonly=T)
par(mfrow=c(2,2))
for (l in 1:4) {
  mds = cmdscale(mut, eig = TRUE, k = 2 * l)
  mdsDist = dist(mds$points, method = "euclidian")
  plot(x = mut,y = mdsDist,main = paste("Diagramme de Shepard ( k =",2 * l,")"))
}
```

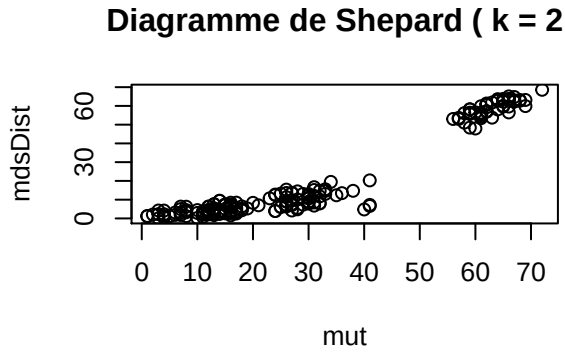


Diagramme de Shepard (k = 2)
mdsDist
0
30
60
0
10
20
30
40
50
60
70
mut

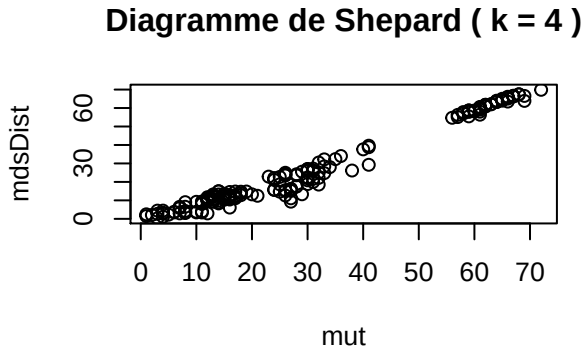


Diagramme de Shepard (k = 4)
mdsDist
0
30
60
0
10
20
30
40
50
60
70
mut

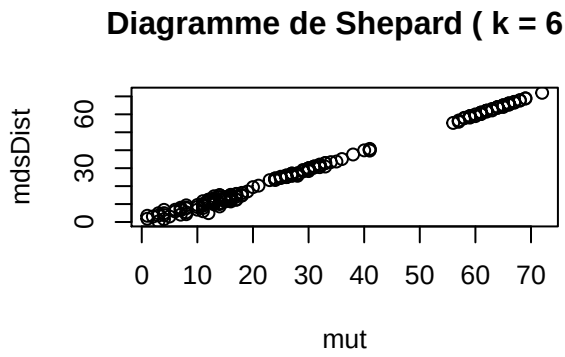


Diagramme de Shepard (k = 6)
mdsDist
0
30
60
0
10
20
30
40
50
60
70
mut

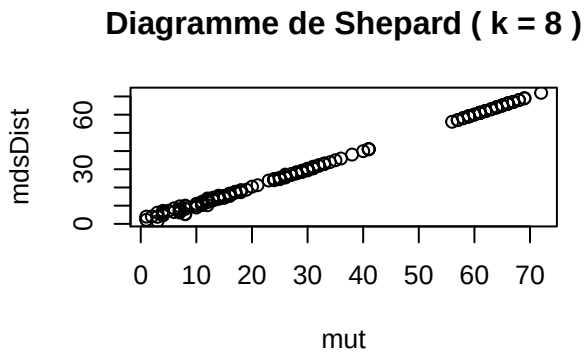


Diagramme de Shepard (k = 8)
mdsDist
0
30
60
0
10
20
30
40
50
60
70
mut

.