Examen de biostatistiques

Barras Clément & Delebarre Bertille

April 1, 2019

Chargement du jeu de données `prostate.txt`

```
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 3.5.3
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.5.3
```

```
## Warning: package 'Matrix' was built under R version 3.5.3
```

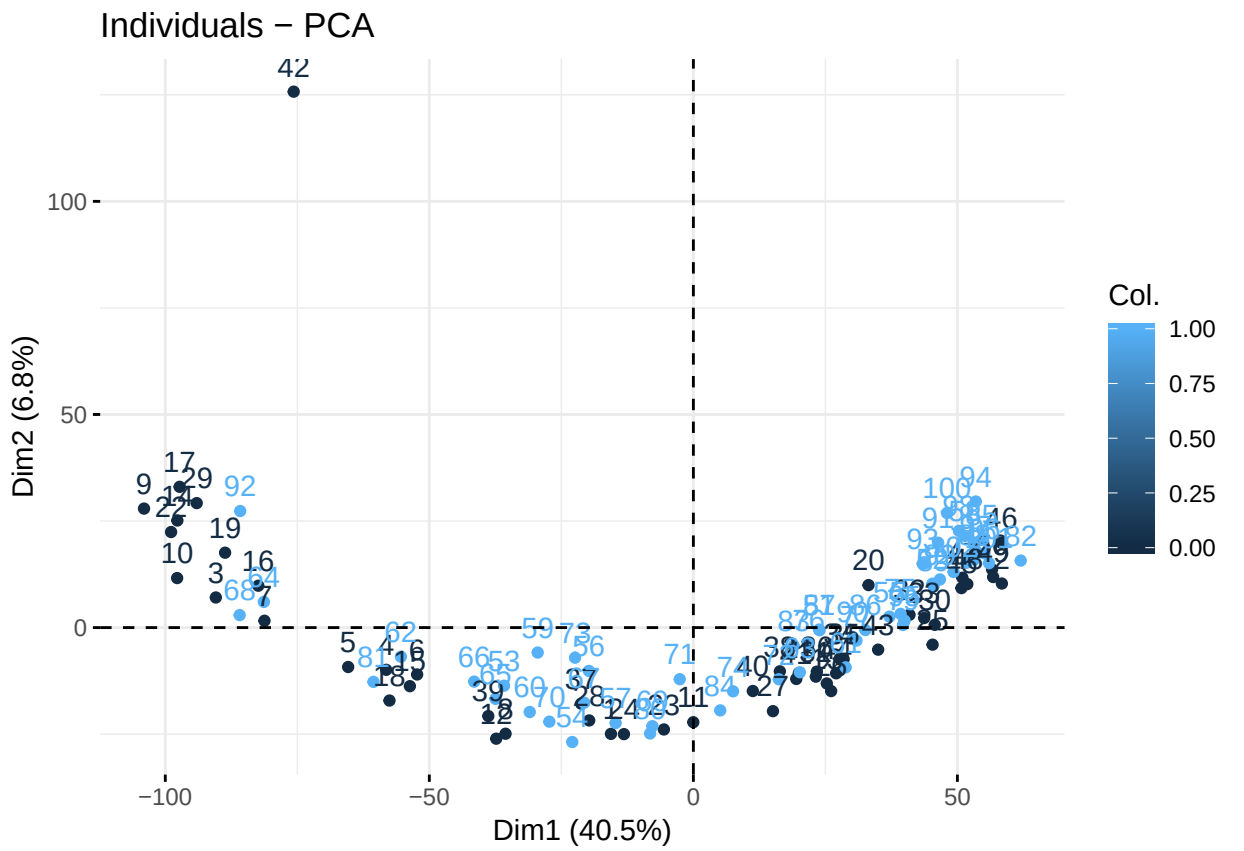
```
## Warning: package 'foreach' was built under R version 3.5.3
```

```
prostate <- read.delim("../data/prostate.txt", sep=" ")
X = as.matrix(prostate[, -1])
y = as.matrix(prostate$y)
n <- nrow(X)
p <- ncol(X)
```

Question 1.

On effectue une ACP sur les données, et on les représente sur le premier plan principal :

```
fit.pca = prcomp(X, scale = TRUE, center = TRUE)
fviz_pca_ind(fit.pca, col.ind = y)
```



Le premier axe principal explique plus de 40% de la variance, ce qui est remarquable au vu du nombre de variables. Cependant, les deux catégories d'individus ne sont pas séparable avec cette projection.

La variable `fit.pca$rotation` contient les vecteurs propres de $\bar{X}^T\bar{X}$ où $\bar{X}$ désigne la matrice des échantillons centrée et réduite.

Afin de représenter quels gènes contribuent le plus aux première et deuxième composantes, on trie les coordonnées des colonnes correspondantes par ordre décroissant en valeur absolue :

```
sort(abs(fit.pca$rotation[,1]), decreasing = TRUE)[1:10]
```

```
##      X5435      X1686      X4784      X5365      X1783      X1321
## 0.01959220 0.01958715 0.01951524 0.01942688 0.01942369 0.01939736
##      X4705      X4813      X2248      X1202
## 0.01938836 0.01938645 0.01938349 0.01937350
```

```
sort(abs(fit.pca$rotation[,2]), decreasing = TRUE)[1:10]
```

```
##       X500       X580      X3712       X881      X5659      X1416
## 0.03921846 0.03803741 0.03797690 0.03782114 0.03767390 0.03754320
##      X5143       X786      X1154      X5616
## 0.03749617 0.03736217 0.03729699 0.03716413
```

Question 2.

Dans cette partie, nous séparons le jeu de données en deux parties : les individus sains et les individus malades.

```
Xsain = X[1:50,]
Xmalade = X[51:102,]
```

Nous effectuons un test de Student sur chacune des variables afin de comparer par leur moyenne les échantillons sains et les échantillons malades sur cette variable. Selon l'hypothèse $\mathcal{H}_0$, la variable correspond à un gène qui ne s'exprime pas différentiellement entre patients et contrôles. Selon l'hypothèse $\mathcal{H}_1$, le gène s'exprime différentiellement, c'est à dire que les échantillons ont une distribution différente sur cette variable.

Nous fixons une p-value $p = 0.05$, sans correction en premier lieu :

```
pvalue = 0.05
ttests <- list()
selected.variables <- list()
for (i in 1:p)
{
  ttests[[i]] <- t.test(Xsain[, i], Xmalade[, i])
  if (ttests[[i]]$p.value<pvalue)
  {
    selected.variables <- append(selected.variables, i)
  }
}
sprintf("Number of selected variables before correction: %d", length(selected.variables))
```

```
## [1] "Number of selected variables before correction: 2326"
```

Le nombre de variables sélectionnées comme ayant une influence sur le statut de l'échantillon est assez élevé : il y a sûrement un certain nombre de faux positifs étant donné le grand nombre de tests. On applique donc une correction de Bonferroni en divisant notre pvalue par le nombre de tests que l'on effectue :

```
pvalue = 0.05/p
selected.variables <- list()
for (i in 1:p)
{
  if (ttests[[i]]$p.value<pvalue)
  {
    selected.variables <- append(selected.variables, i)
  }
}
sprintf("Number of selected variables after correction: %d", length(selected.variables))
```

```
## [1] "Number of selected variables after correction: 163"
```

On a un nombre de variables sélectionnées beaucoup plus restreint, au prix d'avoir généré sans doute davantage de faux négatifs.

Question 3.

Question 8

On construit un modèle de régression logistique parcimonieuse via le package `glmnet`. La fonction `cv.glmnet` permet de détermniner la meilleure valeur du coefficient `lambda` par validation croisée (plus précisément, selon la procédure "leave one out" afin d'éviter tout effet aléatoire). On choisit pour `lambda` la valeur `lambda.min`, qui minimise l'erreur sur la validation croisée

```
fit.logreg = cv.glmnet(X, y, family = "binomial", alpha = 1, nfolds = n, grouped = FALSE)
best_lambda = fit.logreg$lambda.min
best_lambda_index = findInterval(1-best_lambda, 1-fit.logreg$lambda)
```

Le nombre de variables sélectionnées est obtenu via l'attribut `glmnet.fit$df` :

```
fit.logreg$glmnet.fit$df[best_lambda_index]
```

```
## [1] 27
```