Projet Statistique pour la génétique et génomique

OUOROU Rachidou

10/05/2022

```
#install.packages("ISLR")
library(tidyverse)
library(ISLR)
library(FactoMineR)
library(factoextra)
library(ade4)
library(ComplexHeatmap)
library("circlize")
library(golubEsets)
library(pls)
library(plsgenomics)
```

Introduction

Le but de ce projet est d'analyser les données génétique de trois sources différentes. Dans une première partie nous étudierons les données **NCI-60**. L'objectif de cette partie sera de classifier les différentes lignes cellulaire en différents type de cancer. Ensuite on étudiera des données portant sur le cancer de la prostate. Ici, l'objectif est d'étudier les liens entre différents gènes et le statut cancéreux des individus. Pour finir, on s'intéresse aux données **Golub** pour étudier le lien entre l'expression des gènes et le type de leucémie.

1. Données NCI-60

L'objectif de la pharmacogénétique est d'étudier les différents profils moléculaires des cellules tumorales afin de proposer des traitements plus appropriés et efficaces à ces tumeurs. Pour atteindre cet objectif on peut soit faire des essais cliniques afin d'étudier les réaction chez les patients lors des expériences ou étudier directement les lignées cellulaires cancéreuses concernée. La première est difficile à réaliser pour des raisons d'éthique, d'effets externes, et bien d'autres. La deuxième méthode est utilisé par le Developmental Therapeutics Program (DTP) du US National Cancer Institute (NCI) sun un panel de 60 lignée pour tester les agents anticancéreux potentiels. Les cellules ont été caractérisées pharmacologiquement par une exposition à plus de 100 000 composés chimiques définis (plus un grand nombre d'extraits de produits naturels), un à la fois et indépendamment.

Un petit aperçu des différentes lignées cellulaire concernées.

```
library(rvest)
```

```
##
## Attachement du package : 'rvest'

## L'objet suivant est masqué depuis 'package:readr':
##
##     guess_encoding
```

```
list = read_html("https://en.wikipedia.org/wiki/NCI-60") %>%
  html_table(fill = TRUE)

lignee = list[[1]]
t(lignee[, 1]) %>% as.vector()
```

```
##  [1] "CCRF-CEM"
##  [2] "HL-60(TB)"
##  [3] "K-562"
##  [4] "MOLT-4"
##  [5] "RPMI-8226"
##  [6] "SR-786"
##  [7] "A549/ATCC"
##  [8] "EKVX"
##  [9] "HOP-62"
## [10] "HOP-92"
## [11] "NCI-H226"
## [12] "NCI-H23"
## [13] "NCI-H322M"
## [14] "NCI-H460"
## [15] "NCI-H522"
## [16] "COLO 205"
## [17] "HCC-2998"
## [18] "HCT116"
## [19] "HCT-15"
## [20] "HT-29"
## [21] "KM12"
## [22] "SW-620"
## [23] "SF-268"
## [24] "SF-295"
## [25] "SF-539"
## [26] "SNB-19"
## [27] "SNB-75"
## [28] "U251"
## [29] "LOX IMVI"
## [30] "MALME-3M"
## [31] "M14"
## [32] "MDA-MB-435"
## [33] "SK-MEL-2"
## [34] "SK-MEL-28"
## [35] "SK-MEL-5"
## [36] "UACC-257"
## [37] "UACC-62"
## [38] "IGR-OV1"
## [39] "OVCAR-3"
## [40] "OVCAR-4"
## [41] "OVCAR-5"
## [42] "OVCAR-8"
## [43] "NCI/ADR-RES (originally MCF-7/ADR-RES)"
## [44] "SK-OV-3"
## [45] "786-O"
## [46] "A498"
## [47] "ACHN"
## [48] "CAKI-1"
## [49] "RXF 393"
## [50] "SN12C"
## [51] "TK-10"
## [52] "UO-31"
## [53] "PC-3"
## [54] "DU-145"
```

```
## [55] "MCF7"
## [56] "MDA-MB-231/ATCC"
## [57] "MDA-MB-468"
## [58] "HS 578T"
## [59] "MDA-N"
## [60] "BT-549"
## [61] "T-47D"
```

- Chargement du jeu de donnée

```
data("NCI60")
#help("NCI60")
data = NCI60$data
type = NCI60$labs
```

Le jeu de donnée contient l'expression de 6830 gènes pour 64 lignées cellulaires. Par contre, les lignées concernées ne sont pas précisées, mais plutôt le type de cancer correspondant.

On présente dans le tableau suivant les différents type de cancer.

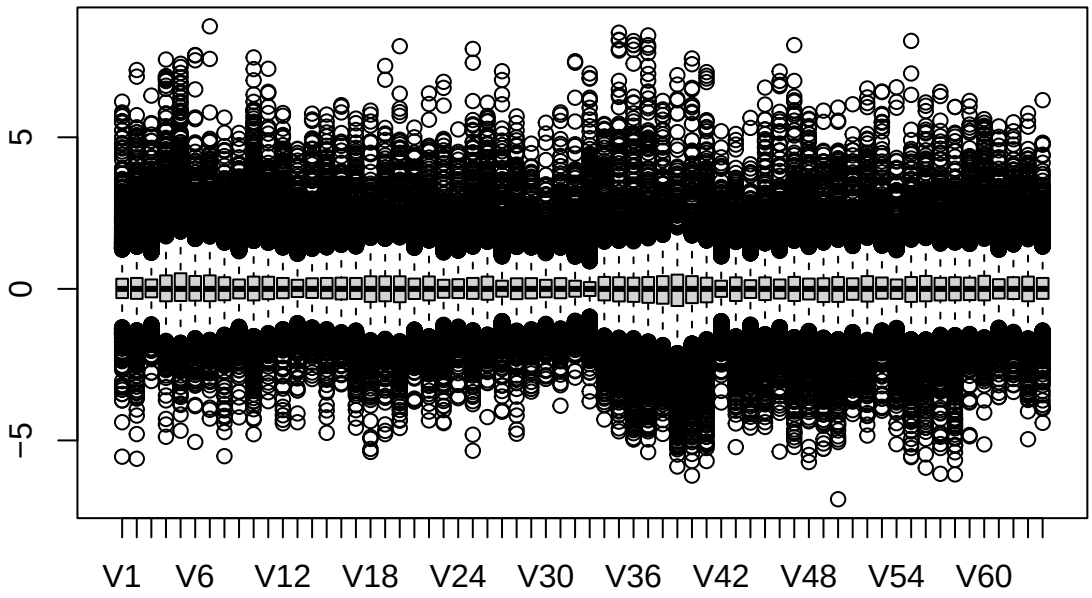
```
table(type) %>%
  sort(decreasing = T)  %>%
  addmargins() %>%
  knitr::kable() %>%
  t()
```

type	Freq
NSCLC	9
RENAL	9
MELANOMA	8
BREAST	7
COLON	7
LEUKEMIA	6
OVARIAN	6
CNS	5
PROSTATE	2
K562A-repro	1
K562B-repro	1
MCF7A-repro	1
MCF7D-repro	1
UNKNOWN	1
Sum	64

Les lignées les plus représenté parmi les 64 sont celles associé au "Non-Small Cell Lung Cancer" et au cancer Rénal.

On réalise un boxplot de l'expression des gènes pour chacune des lignées.

```
boxplot(t(data))
```



On remarque que les expressions sont plutôt normalisée et ont le même ordre de grandeur.

On aimerais savoir s'il est possible de regroupé certaines lignée cellulaire, c'est-à-dire si les gènes s'expriment de façon similaire en présences de ces celulles.

1.1 Analyse en composante principales

Mais vu que le nombre de gènes est très grand (6380), on ne peut pas visualiser ce rapprochement des lignée à travers un nuages de points. On se propose de cet fait de réduire la dimension de ces données afin de les visualiser sur un nuage de point. On réalise donc une analyse en composante principale (ACP) à cet effet.

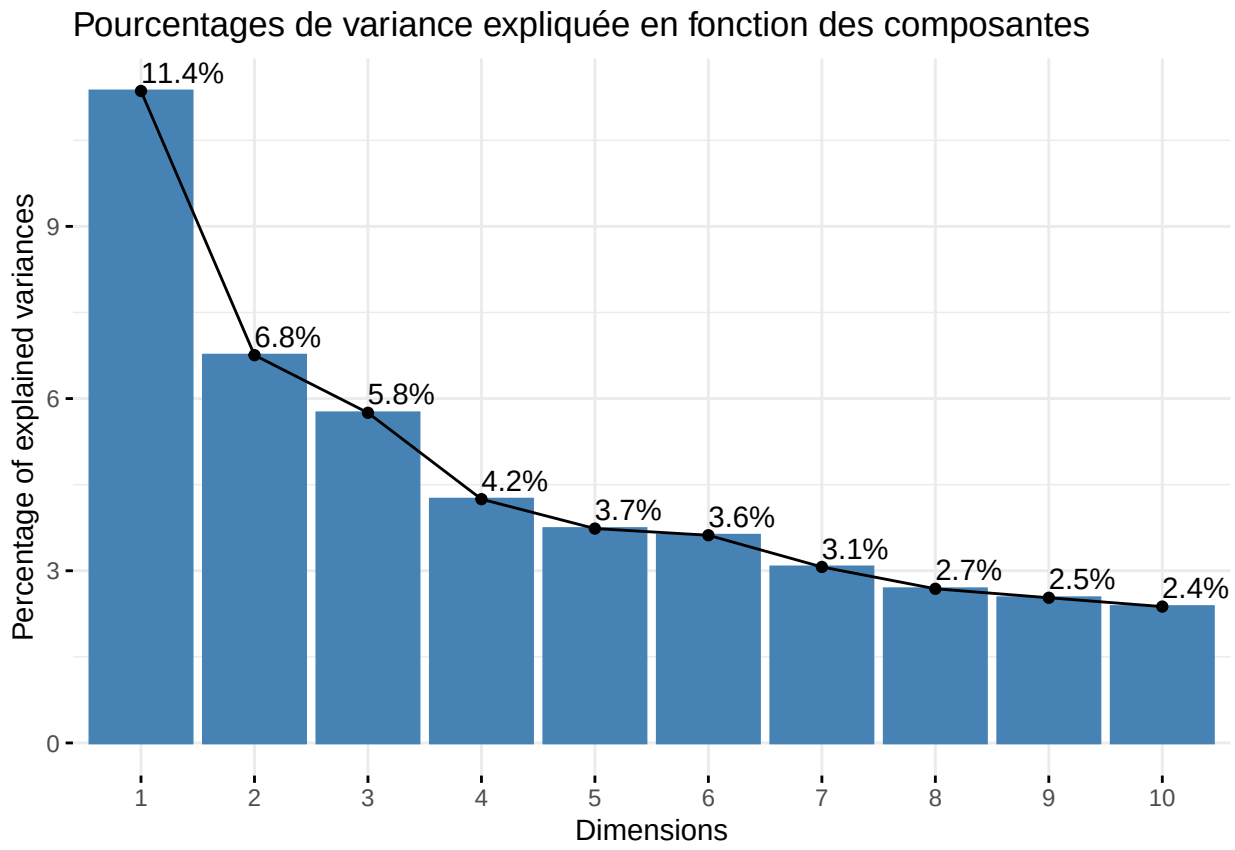
1.1.1. Principe ACP

Cette méthode consiste à projeté un jeu de donnée de grande dimension dans un espace à dimension plus réduite tout en conservant un maximum de variance des données initiales. Ainsi on part d'un nuage à $n * p$ dimensions à un nuage de points à $n * r$ dimension (avec $r \leq p$). En génétique, généralement $p >> n$. Le nombre de gènes étudié est très grand pour peu d'individus (grande dimension). On utilise donc cette méthode afin d'avoir moins de dimensions avec le plus d'information possible.

1.1.2. Application

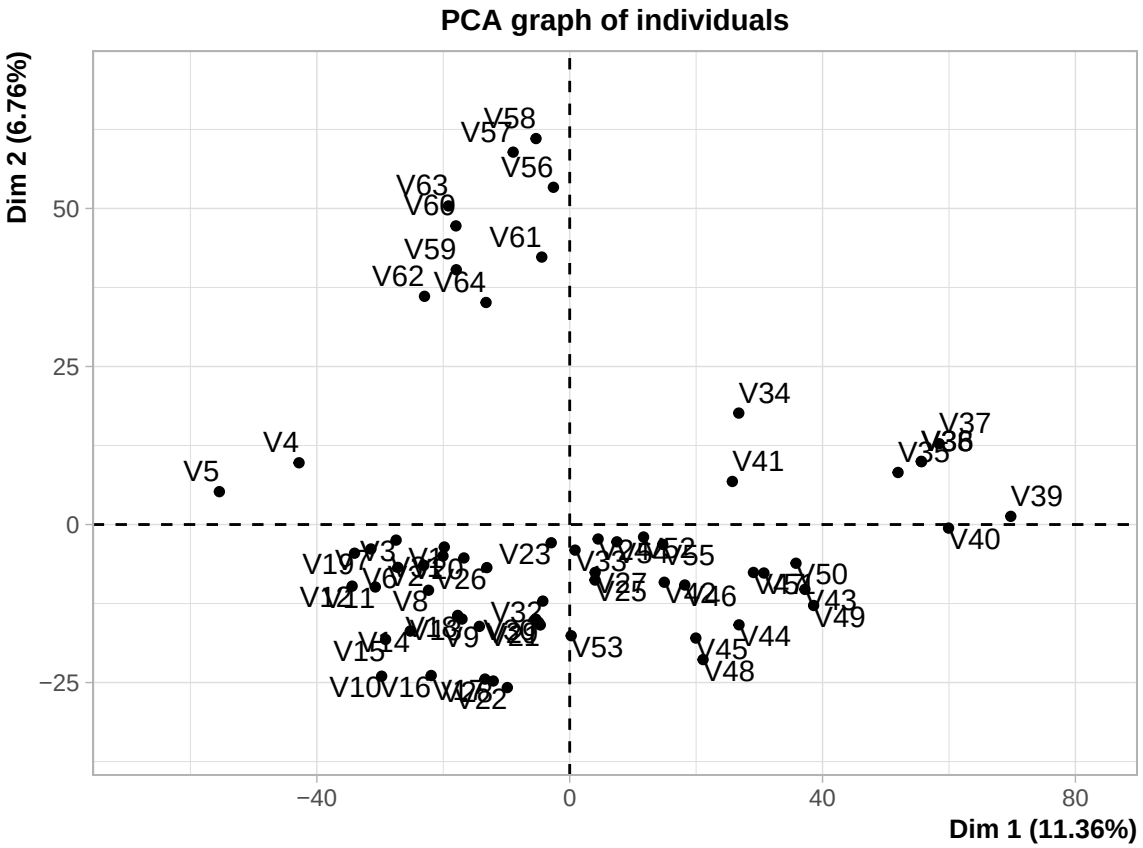
```
data1 = data.frame(type, data)
acp = PCA(data1[, -1], graph = F)
```

```
fviz_eig(acp, addlabels=TRUE) +
  labs(title = "Pourcentages de variance expliquée en fonction des composantes")
```



Ce graphique présente le pourcentage de variances contenu dans les dimensions de l'ACP. On remarque qu'avec la première composante, on on a 11.4 de la variance de départ. On projette les données sur les deux premières dimensions nous permettant de retenir 18.2 de la variance.

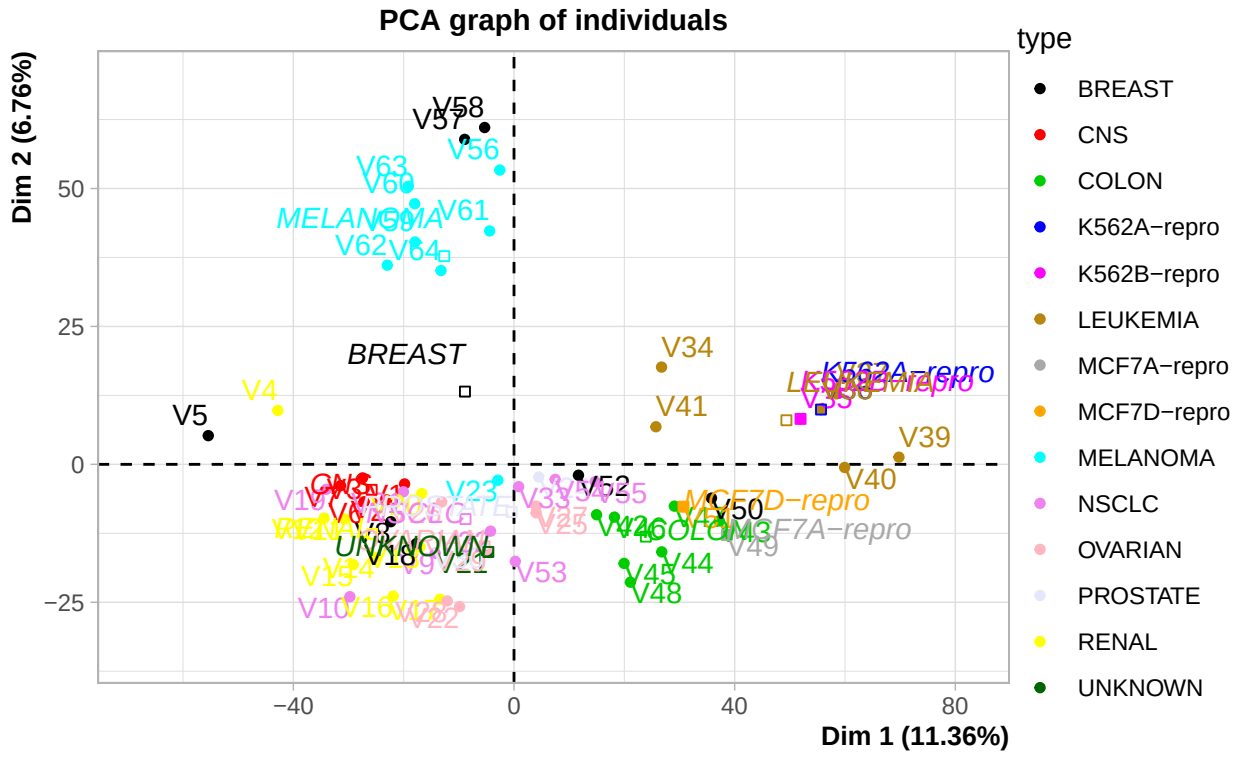
```
plot(acp, axes = c(1, 2), choix = "ind")
```



Sur ce graphique, on remarque un groupe qui se distingue dans le cadran en haut à gauche. On a trois autres groupe concentrée dans chaque cadran même si la distinction n'est pas très nette.

On rajoute a ce graphique les type de cancer afin de vérifier sa pertinence.

```
acp = PCA(data1, graph = F, quali.sup = 1 )
plot(acp, axes = c(1, 2), choix = "ind", habillage = 1)
```



On remarque bien que les lignées cellulaire du cadran (-, +) correspond essentiellement au cancer Melanoma. En général, les lignés cellulaire proche sur le graphique sont du même type de cancer. Bien qu'on ait perdu de l'information après réduction de dimension, les résultats sont plutôt pertinent dans la mesure les lignées cellulaire de même type sont assez proche sur le graphique.

1.2 Classification Ascendante Hiérachique (CAH)

On souhaite maintenant classifier les lignées cellulaires en utilisant une CAH.

1.2.1 Principe CAH

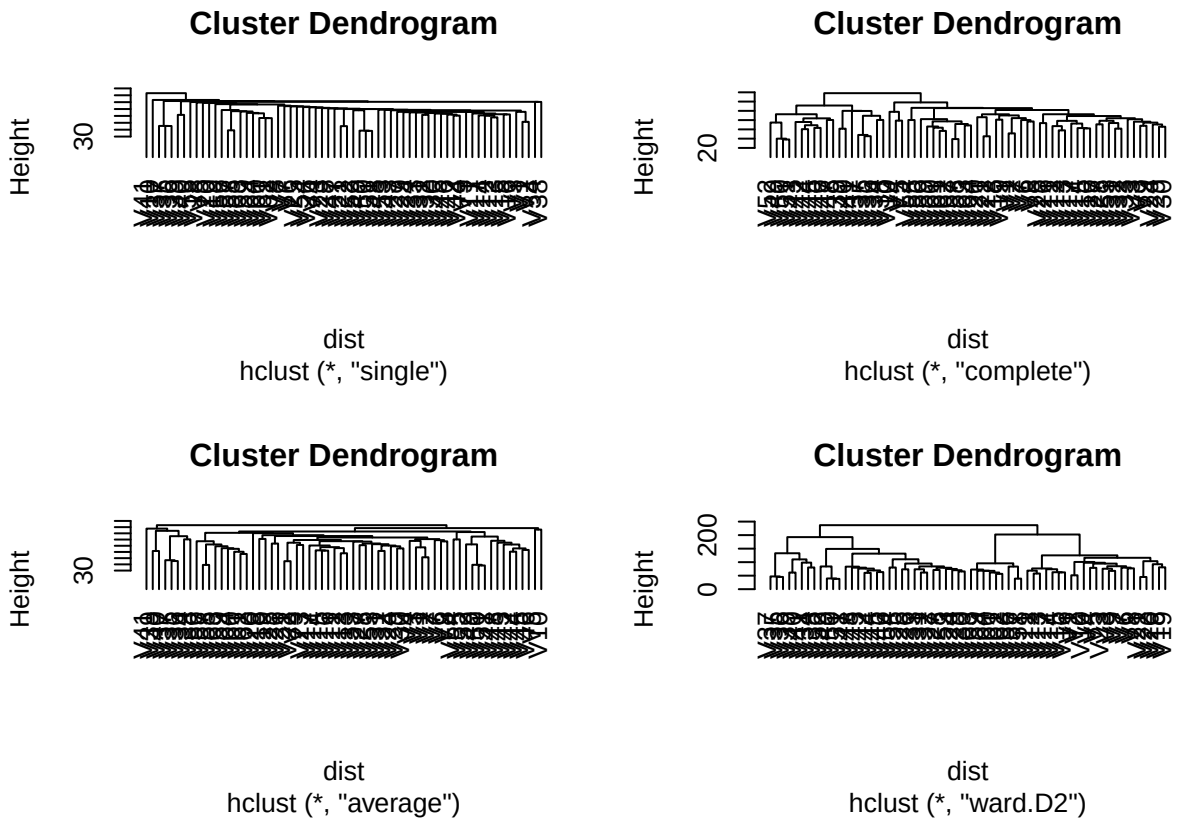
Le pricipe de la CAH consiste à calculer la distance entre les individus et de regrouper les individus les plus proche au sens de la distance choisie (euclidienne, Mahalanobis, etc. . .). Il convient aussi de choisir le critère permettant de calculé la distance entre deux groupes. Il existe les méthodes minimium, maximum, ward, etc.

Au début, chaque individu est dans un classe. Ils sont succècivement aggréger a chaque étape jusqu'a ce que tout le monde ne forme qu'une classe.

Afin de décider de la meilleur stratégie d'agréggation on réalisera la classification pour chacune de ces méthodes en choisissant la meilleure méthode aux vues des dendrogramme.

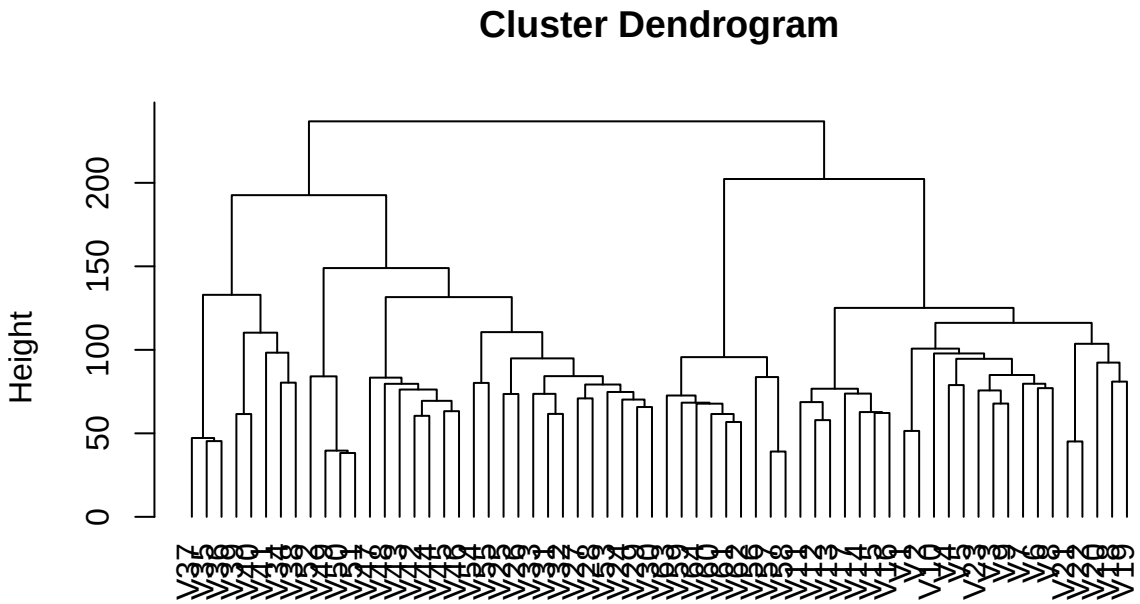
Le boxplot des données réalisée plus haut montre que les données étaient plutôt normalisée. On teste donc les différentes stratégies avec la distance euclidienne.

```
library(ade4)
par(mfrow = c(2, 2))
dist = dist.quant(data, method = 1)
for (j in c("single", "complete", "average", "ward.D2")) {
  res = hclust(dist, method = j)
  plot(res, hang = -1)
}
```



La stratégie du minim et celle des centroïdes ne fournit de dendrogramme assez lisible contrairement a la stratégie du maximum et de ward. On retient celle de ward qui parait plus nette.

```
res = hclust(dist, method = "ward.D2")
plot(res, hang = -1)
```



A cette étape, il faut décider a quel niveau couper l'arbre afin d'obtenir les classe. On utlise le critères du saut maximal à cet effet.

```
## Critère du saut de maximal
which.max(diff(res$height))
```

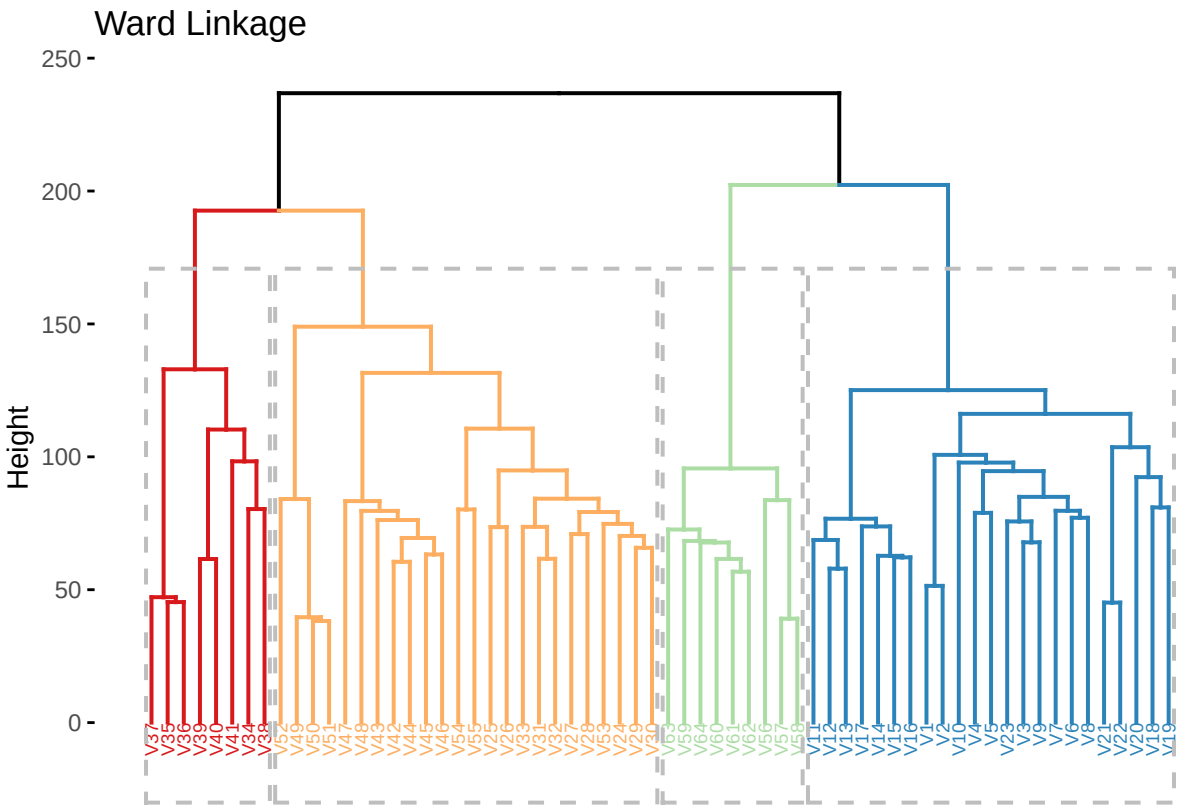
```
## [1] 60
```

```
res$height[60:61]
```

```
## [1] 148.9644 192.6257
```

Le maximal est observé à lors du passage de l'étape 60 à 61. On décide donc de couper l'arbre entre ces deux passages.

```
fviz_dend(res, main="Ward Linkage", cex=.5,
          k = 4, # cut in four groups
          rect=TRUE,
          palette="Spectral")
```



Remarquons déjà que le critère du saut maximal nous permet de retenir un classement en 4 classes comme on l'observait sur le nuages de poins des deux premières composantes de l'ACP.

On compare également le classement obtenue des lignées cellulaire par CAH avec les vraie classement

```
cl_cah = cutree(res, k = 4)
table(type, cl_cah)
```

```
##              cl_cah
## type          1 2 3 4
##   BREAST      3 2 0 2
##   CNS         5 0 0 0
##   COLON       0 7 0 0
##   K562A-repro 0 0 1 0
##   K562B-repro 0 0 1 0
##   LEUKEMIA    0 0 6 0
##   MCF7A-repro 0 1 0 0
##   MCF7D-repro 0 1 0 0
##   MELANOMA    1 0 0 7
##   NSCLC       3 6 0 0
##   OVARIAN     1 5 0 0
##   PROSTATE    0 2 0 0
##   RENAL       9 0 0 0
##   UNKNOWN     1 0 0 0
```

On remarque que la méthode est assez efficace. Par exemple, le 9 lignée cellulaire du cancer rénal sont dans la même classe (1). De même pour le Colon, leukemia, prostate, etc... Seul les lignées du cancer dunsein et

NSCLC sont plus ou moins mal classé. En général, les lignées celullaire du même cancer sont dans le même groupe et nous permet de confirmer la pertinence de la classification.

Pour résumer cette première partie, on a utiliser l'analyse en composante principales afin d'afficher les lignées cellulaire sur un graphique. Ce graphique était plus ou moins en accord avec les type de cancer. Ensuite on réalise une classification ascendante hiérachique qui donne aussi de résultat pertinent dans le classenment des lignées, confirmé par le tableau de contingence.

2. Donnée cancer prostate

L'objectif de cette partie, est d'étudier le lien entre l'expression des gènes et le statut cancéreux des individus. Etant confronté au problème de grande dimension, on essaie dans une première partie de réduire la dimension grâce à l'ACP avant de proposer un autre alternative d'étude de ce lien.

2.1 Réduction de dimension

2.1.1 Description des donnée

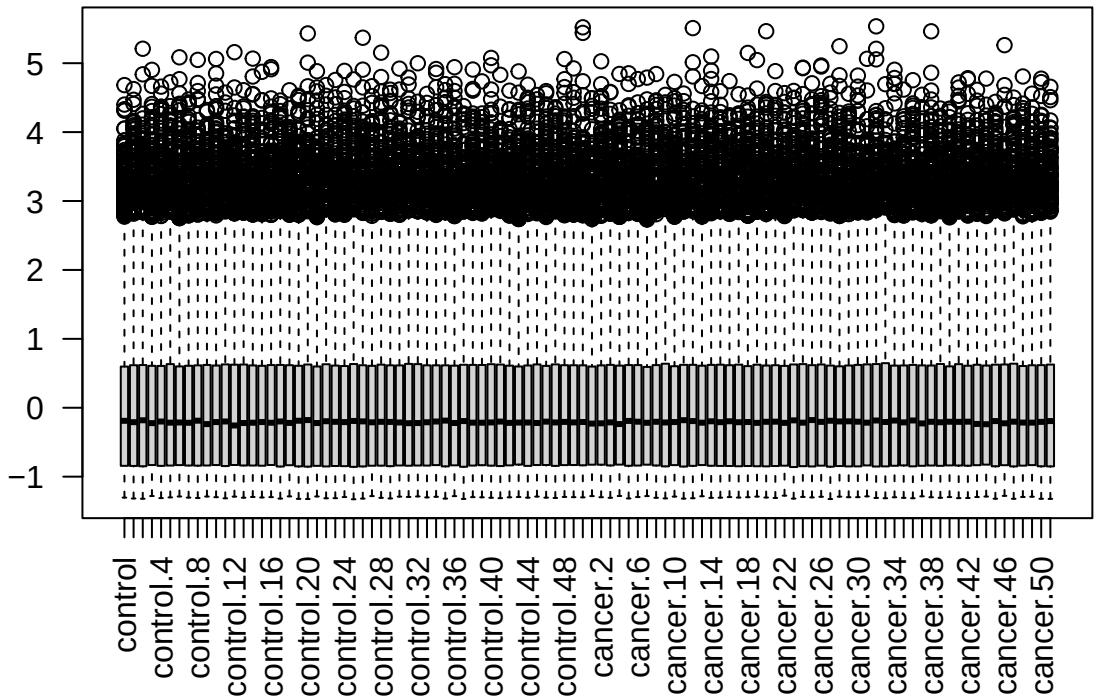
```
prostate = read.csv("http://web.stanford.edu/~hastie/CASI_files/DATA/prostmat.csv")


dim(prostate)
```

```
## [1] 6033  102
```

Le jeu de données contient l'expression de 6033 gènes collecté sur 102 individus

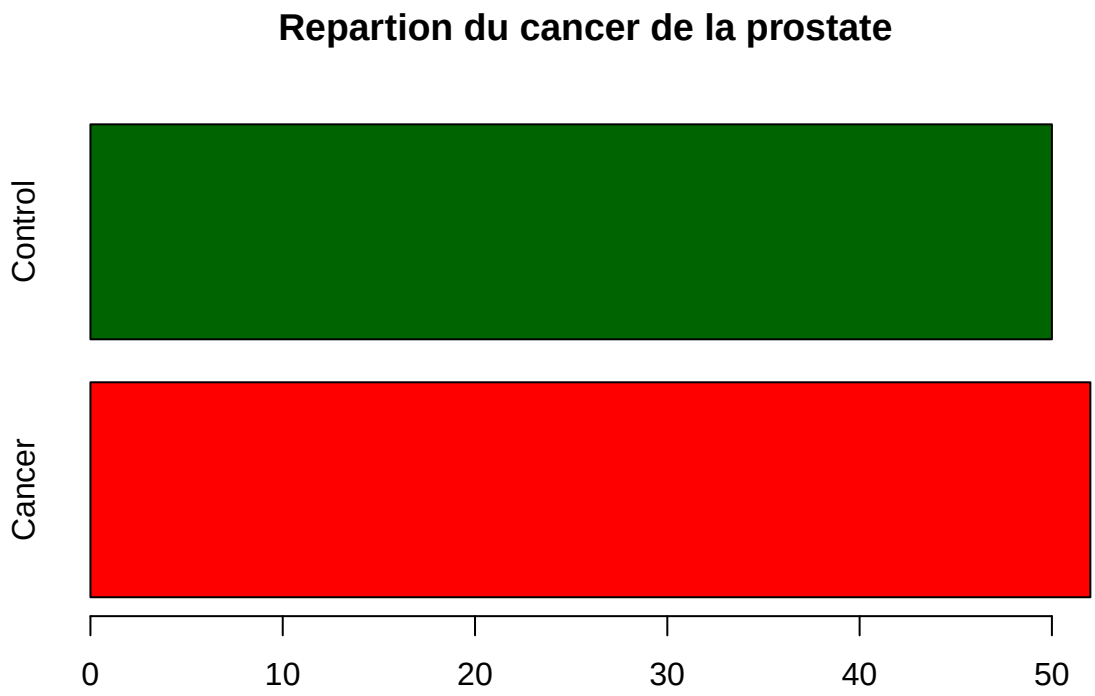
On réalise un boxplot pour vérifié si les données ont normalisées.

```
boxplot((prostate), las = 2)
```



Elles paraissent bien normalisées.

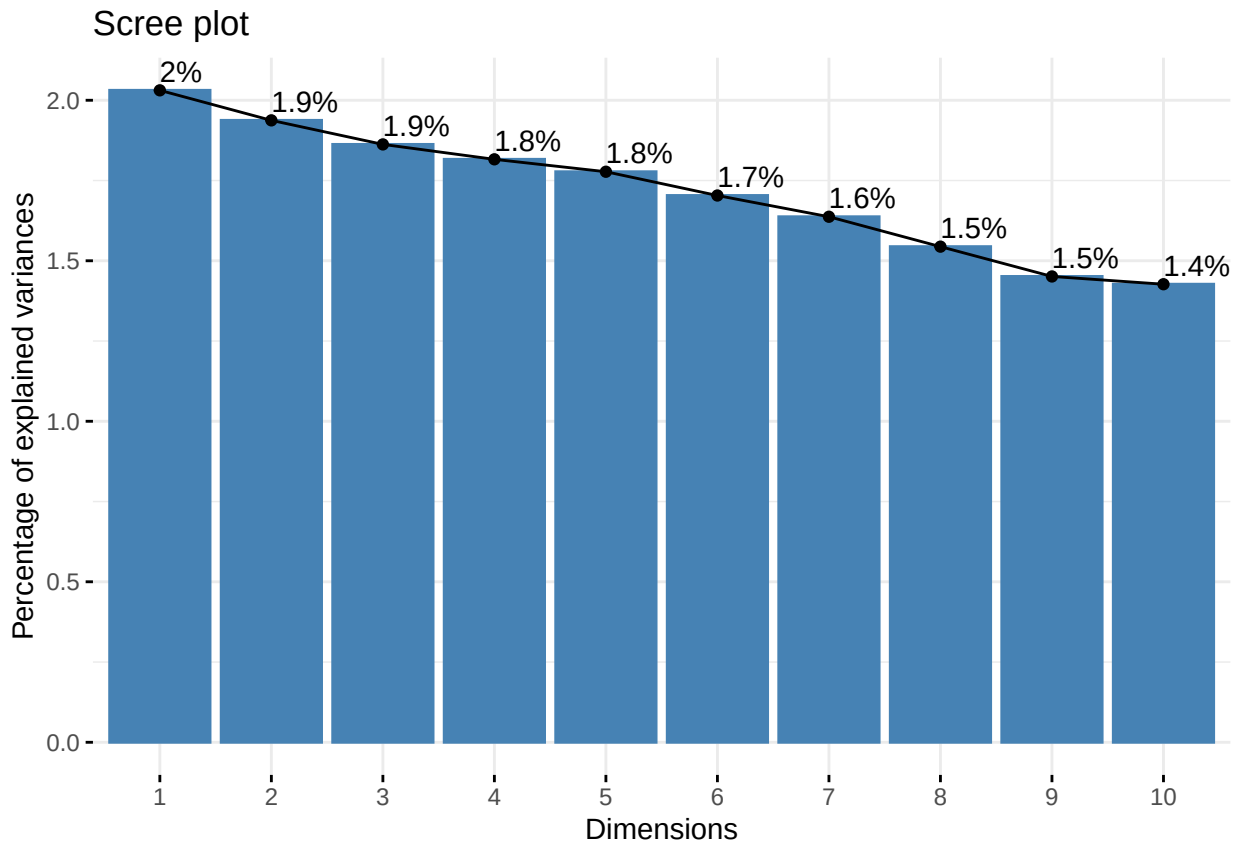
```
statut = c(rep("Control", 50), rep("Cancer", 52))
barplot(table(statut),col = c("red", "darkgreen"), horiz=TRUE, main = "Repartion du cancer de la prosta
```



On a 50 individus contrôle (n'ayant pas le cancer) et 52 atteintes.

2.1.2 ACP

```
prostate2 = data.frame(t(prostate), statut)
acp2 = PCA(prostate2, quali.sup = 6034, graph = F)
fviz_eig(acp2, addlabels = TRUE)
```



Remarquons que les deux premières dimensions ne permettent d'avoir que 4% de la variance initiales. Ce qui parait très peut si ces composantes devait servir de base pour toute analyse car elles ne refléterait pas la variance du jeu de données initiales.

```
which(as.vector(acp2$eig[, 3]) >80)
```

```
##  [1]  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
## [20]  90  91  92  93  94  95  96  97  98  99 100 101
```

En effet pour avoir 80% de la variance expliqué par les axes principaux, il faudrait prendre en compte les 71 premiers axes. Puisque l'on a que 102, individus dans le jeu de données, un modèle à 70 coefficient parait peut raisonnable.

2.2 Expression différentielle

La réduction de dimension n'étant pas efficace dans ce cas, on propose donc de faire des test de student pour chacun des 6033 gènes du jeu de donné. Le test de student permet de tester l'égalité de deux moyenne. Ici on teste si l'expression moyenne des gènes chez les individus atteints de cancer est la même que chez les individus sains. On teste,

- Hypothèse nulle : $H_0 : \mu_{cancer} = \mu_{control}$ contre
- Hypothèse alternative : $H_0 : \mu_{cancer} \neq \mu_{control}$.

On rejetera l'hypothèse nulle si la pvaleur du teste est plus petite que le seuil fixée arbitrairement ici à 10%. Dans lequel cas le gène serait exprimé différentiellement.

```r
p.values = apply(prostate, 1, function(x){t.test(x~ factor(statut))$p.value})


sum(p.values < 0.1, na.rm = TRUE)
```

```
## [1] 797
```

On remarque que pour 797 gènes, la p.valeur était inférieur à 10%. On serait donc tenté de dire que ces gènes sont associé au statut cancéreux. Cependant, les test étant réalisé indépendamment, cette erreur de 10% s'est acummulé sur les 6833 tests réalisé. On a donc un taux d'erreur très élevé qui fausse de ce fait cette conclusion.

On décide de ce fait d'apporté une correction a ces p-valeur afin d'ajusté le seuil de décision. On utilise a cet effet la correction de bonferroni et celle de Benjamini-Hochberg.

La correction de bonferroni consiste à divisé le seuil de décision par le nombre de test réalisé

```r
sum(p.values < 0.1/length(p.values))
```

```
## [1] 6
```

```r
# ou
sum(p.adjust(p.values, method = "bonferroni") < 0.1)
```

```
## [1] 6
```

Ainsi en appliquant la correction de bonferroni, on ne retient que 6 gènes différentiellement exprimés au seuil de 10%. Cette méthode est assez sévère et rejette beaucoup de variable. On décide de faire la correction de benjamini-Hochberg.

Cette méthode consiste à contrôler la proportions de fausses découvertes, c'est à dire la fréquence à la quelle on décide H_1 (alors que la vérité est H_0) parmi toutes les décisions H_1.

```r
p.values.BH=p.adjust(p.values,method="BH")

sum(p.values.BH < 0.1, na.rm = T)
```

```
## [1] 57
```

On remarque qu'avec la correction de BH est moins péssimiste et nous permet de retenir 57 gènes différentiellement exprimés

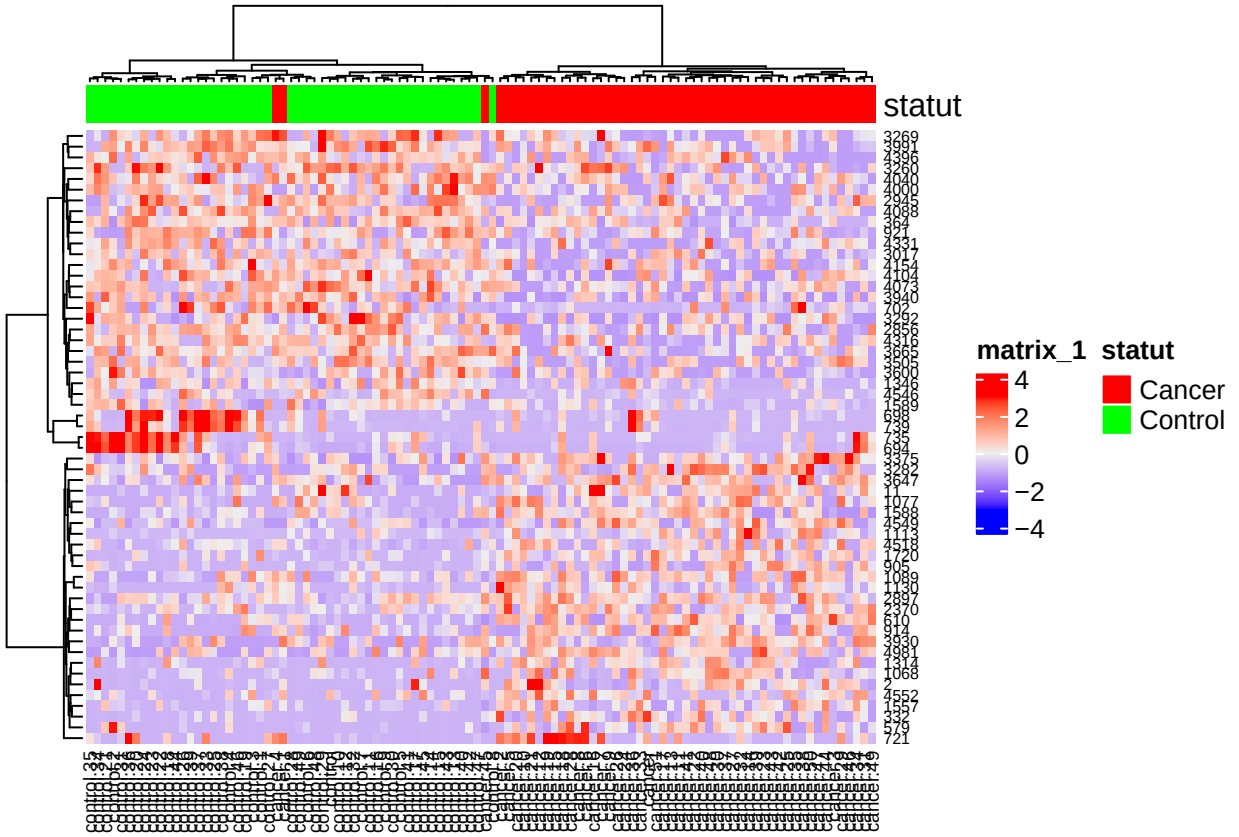
Pour terminer cette partie, on décide de visualiser ces gènes différentiellement exprimés sur une heatmap.

```r
selecDE = which(p.values.BH <= 0.1)
selecDE
```

```
##  [1]    2   11  332  364  579  610  694  698  702  721  735  739  905  914  921
## [16] 1068 1077 1089 1113 1130 1314 1346 1557 1588 1589 1720 2370 2856 2897 2945
## [31] 3017 3260 3269 3282 3292 3375 3505 3600 3647 3665 3930 3940 3991 4000 4040
## [46] 4073 4088 4104 4154 4316 4331 4396 4518 4546 4549 4552 4981
```

```
dataH = prostate[selecDE,]
```

```
ha = HeatmapAnnotation(statut = prostate2[, 6034] ,
col = list("statut"= c("Cancer" = "red", "Control" = "green")))

Heatmap(as.matrix(dataH), top_annotation = ha,
              clustering_method_rows = "ward.D",
              clustering_method_columns = "ward.D",
              clustering_distance_columns = "pearson",
              row_names_gp = gpar(fontsize = 6),
              column_names_gp = gpar(fontsize = 7)
              )
```



Sur cette heatmap on peut observer 4 carrés assez homogènes en couleur. Pour la moitié des premiers gènes (3269 à 735), on remarque que ces gènes sont sur-exprimés chez les personnes saines et sous exprimés chez les personnes atteintes de cancer de la prostate. Mais les gènes en partant de 735, sont plutôt sur-exprimés chez les personnes atteintes de cancer et sous-exprimés chez les personnes saines.

La suite de cette étape serait de faire une analyse d'enrichissement des différente fonctions de ces gènes, mais que nous n'aborderont pas dans ce projet.

3. Données Golub

L'objectif de cette partie est d'étudier le lien entre l'expression des gènes et le type de leucémie à savoir la leucémie aiguë lymphoblastique (ALL) et la leucémie aiguë myéloïde (AML). Comme précédemment on

procédera à une réduction de dimension avant d'appliquer la régression logistique pour étudier le dit lien. On utilisera une deuxième méthode de réduction de dimension qu'est la PLS (Principal Least Square).

3.1. Chargement et transformation des données

```
data(Golub_Merge)
x <- exprs(Golub_Merge)
dim(x)
```

```
## [1] 7129   72
```

Le jeu de donnée contient l'expression de 7129 gènes collecté chez 72 individus.

On souhaite ensuite enlever les gènes dont l'expression n'est pas assez importante ou dont la variance n'est pas élevé.

```
x[x < 100] <- 100
x[x > 16000] <- 16000
emax <- apply(x, 1, max)
emin <- apply(x, 1, min)
x <- x[emax/emin > 5 & emax - emin > 500, ]
x <- log10(x)
x <- t(scale(t(x)))
x = data.frame(x)
```

```
statut = as.numeric(Golub_Merge$ALL.AML) - 1

donnee = data.frame(X = I(t(x)), Y = statut)
table(Golub_Merge$ALL.AML) %>% addmargins()
```

```
##
## ALL AML Sum
##  47  25  72
```

Parmi les 72 individus, 47 ont la leucémie ALL et 25 la leucémie AML.

3.2. ACP + regression logistique

On se propose aussi de subdiviser le jeu de données en 2 parties: une partie servira à l'entraînement du modèle et une seconde partie pour la validation du modèle.

```
set.seed(1)
train <- rbinom(length(donnee$Y),1,0.8)
donnee.train <- c()
donnee.test <- c()
donnee.train$X <- donnee$X[train==1,] # donnee.train est une liste avec deux éléments
donnee.train$Y <- donnee$Y[train==1]
donnee.test$X <- donnee$X[train==0,] # donnee.test est une liste avec deux éléments
donnee.test$Y <- donnee$Y[train==0]
```

On réalise l'ACP.

```
pcrdonnee <- pcr(Y ~ X, data = donnee.train
                 , scale = FALSE,
                validation="none")
summary(pcrdonnee)
```

```
## Data:    X dimension: 60 3571
##  Y dimension: 60 1
## Fit method: svdpc
## Number of components considered: 59
## TRAINING: % variance explained
##     1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X  16.75236    28.38    34.13    39.31    43.30    46.93    49.93    52.33
## Y   0.04582    42.00    61.71    73.08    75.16    77.46    81.02    81.34
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    54.55     56.66     58.63     60.48     62.16      63.7     65.17
## Y    81.40     81.64     85.50     86.45     88.69      89.1     90.75
##    16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X     66.58     67.93     69.22     70.49     71.72     72.86     73.92
## Y     90.82     90.84     90.91     91.62     91.67     92.02     92.43
##    23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X     74.96     75.99     76.98     77.97     78.91     79.83     80.72
## Y     94.57     95.04     95.21     95.58     95.59     95.97     96.01
##    30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X     81.60     82.45     83.28     84.10     84.89     85.65     86.40
## Y     96.36     96.69     97.15     97.42     97.46     97.46     97.56
##    37 comps  38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X     87.14     87.86     88.57     89.28     89.97     90.65     91.33
## Y     97.58     97.75     98.04     98.04     98.04     98.05     98.45
##    44 comps  45 comps  46 comps  47 comps  48 comps  49 comps  50 comps
## X     91.99     92.64     93.26     93.87     94.47     95.06     95.63
## Y     98.51     98.53     98.55     98.55     98.62     98.96     99.02
##    51 comps  52 comps  53 comps  54 comps  55 comps  56 comps  57 comps
## X     96.19     96.72     97.24     97.75     98.24     98.72     99.18
## Y     99.30     99.30     99.30     99.31     99.39     99.64     99.73
##    58 comps  59 comps
## X     99.60       100
## Y     99.96       100
```

On remarque que pour conserver 70% de la variance des données initiales, il nous faut retenir les 19 premières composantes. On décide donc dans un premier temp de récupérer les 19 premières composantes principales pour la régression logistique

```
ncomponents <- 19
donnee.train$reducedX <- pcrdonnee$scores[, 1:ncomponents]
pca.model <- glm(Y ~ reducedX, data = donnee.train, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(pca.model)
```

```
##
## Call:
## glm(formula = Y ~ reducedX, family = binomial, data = donnee.train)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -8.801e-06  -8.905e-07  -2.110e-08   2.110e-08   1.100e-05
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.213e+01  9.448e+04       0        1
## reducedXComp 1  -1.742e-01  1.749e+03       0        1
## reducedXComp 2  -1.247e+00  4.224e+03       0        1
## reducedXComp 3  -1.277e+00  4.287e+03       0        1
## reducedXComp 4   1.237e+00  4.752e+03       0        1
## reducedXComp 5  -7.526e-01  4.046e+03       0        1
## reducedXComp 6   5.137e-01  3.195e+04       0        1
## reducedXComp 7  -1.035e+00  7.195e+03       0        1
## reducedXComp 8   8.799e-02  1.202e+04       0        1
## reducedXComp 9  -1.920e-01  1.890e+04       0        1
## reducedXComp 10  3.148e-01  1.067e+04       0        1
## reducedXComp 11  9.799e-01  7.546e+03       0        1
## reducedXComp 12 -5.580e-01  1.102e+04       0        1
## reducedXComp 13 -9.200e-01  1.171e+04       0        1
## reducedXComp 14 -3.213e-01  1.706e+04       0        1
## reducedXComp 15  1.443e+00  8.513e+03       0        1
## reducedXComp 16  5.614e-01  1.460e+04       0        1
## reducedXComp 17 -3.642e-01  1.371e+04       0        1
## reducedXComp 18 -7.308e-01  1.462e+04       0        1
## reducedXComp 19  9.948e-01  1.484e+04       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7.7694e+01  on 59  degrees of freedom
## Residual deviance: 8.7146e-10  on 40  degrees of freedom
## AIC: 40
##
## Number of Fisher Scoring iterations: 25
```

On remarque déjà que l'algorithme de calcul des coefficient n'a pas convergé, les variances sont énormes et les statistique de test sont nulles. Le nombre de paramètres est assez important pour ce nombre d'individu.

On se restreint de ce fait aux trois premiers axes principaux qui nous permettent de conserver 27% de la variance des données.

```
ncomponents <- 3
donnee.train$reducedX <- pcrdonnee$scores[, 1:ncomponents]
pca.model <- glm(Y ~ reducedX, data = donnee.train, family = binomial)
summary(pca.model)
```

```
##
```

```
## Call:
## glm(formula = Y ~ reducedX, family = binomial, data = donnee.train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.59733  -0.25766  -0.03385   0.08312   2.16739
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.85739    0.76462  -2.429 0.015133 *
## reducedXComp 1  0.01759    0.02681   0.656 0.511755
## reducedXComp 2 -0.22511    0.06763  -3.329 0.000873 ***
## reducedXComp 3 -0.18126    0.06522  -2.779 0.005451 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 77.694  on 59  degrees of freedom
## Residual deviance: 22.486  on 56  degrees of freedom
## AIC: 30.486
##
## Number of Fisher Scoring iterations: 7
```

D'après les résultat, le premier axe principal n'est pas significativement lié à la variable réponse (type de leucémie). On la retire du modèle en conservant les deux autres

```
ncomponents <- 3
donnee.train$reducedX <- pcrdonnee$scores[, 2:ncomponents]
pca.model <- glm(Y ~ reducedX, data = donnee.train, family = binomial)
summary(pca.model)
```

```
##
## Call:
## glm(formula = Y ~ reducedX, family = binomial, data = donnee.train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.33804  -0.25930  -0.03904   0.06418   2.29535
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.63350    0.64768  -2.522  0.01167 *
## reducedXComp 2 -0.21807    0.06654  -3.278  0.00105 **
## reducedXComp 3 -0.18518    0.06483  -2.856  0.00429 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 77.694  on 59  degrees of freedom
## Residual deviance: 22.934  on 57  degrees of freedom
## AIC: 28.934
```

```
##
## Number of Fisher Scoring iterations: 7
```

```
anova(pca.model, test = "LRT")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Y
##
## Terms added sequentially (first to last)
##
##
##          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                        59     77.694
## reducedX  2   54.759        57     22.934 1.286e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On réalise un test de significativité global du modèle. On teste si le modèle proposé est meilleur que le modèle nulle, c'est à dire le modèle ne contenant que la constante. On rejette ici l'hypothèse nulle et donc le modèle proposé est globalement significatif.

On utilise donc ce modèle final pour prédire le type de leucémie des individus de la base test. On projette de ce fait ces individu sur le plan formé par les dimension 2 et 3.

```
gen_moy = apply(donnee.train$X, 2, mean) # moyenne d'expression par gènes
bar_donnee_train <- matrix(rep(gen_moy, each = length(donnee.test$Y)), ncol = ncol(donnee.train$X)) # O

reduction.matrix <- pcrdonnee$loadings[,2:3] # Matrice de loadings pour la projection
x_test_centered <- donnee.test$X - bar_donnee_train
donnee.test$reducedX <- x_test_centered %*% reduction.matrix # Projections
```

Maintenant que la projection est faites, test le modèles sur ces données

```
test.prediction <- predict(pca.model, newdata = donnee.test, type = "response")
# Matrice de confusion
table(true_value = donnee.test$Y, predictions = (test.prediction >0.5)*1)
```

```
##           predictions
## true_value 0 1
##          0 8 0
##          1 0 4
```

La matrice de confusion, nous permet de comparé les prédiction réalisé aux vraies valeur du type de leucémie. On remarque que notre modèle ne fait aucune erreur de classement.

3.3. PLS + regression

On décide maintenant de réduire la dimension des données en utilisant la PLS comme méthode de réduction. Rappelons que le principe, est le même: Partir d'un jeu de données à $n * p$ dimension pour un jeu de donnée

à $n * r$ dimension avec r plus petit ou égale à n. Contrairement à l'acp qui cherche les composantes sorte à maximer la variances conteu dans les données initiales, la PLS cherche plutôt à maximiser la covariance au carré entre la variable d'intérêt Y et les axes principaux.

On met cela en pratique dans le code suivant.

```
plsdonnee <- plsr(Y ~ X, data = donnee.train, scale = FALSE,
                  validation = "LOO")
```

Le choix du nombre d'axe dans ce cas ne se fait pas par maximisation de la variance mais par validation croisé ou leave one out. Ici on utlisera la validation leaveoneout qui consiste a retirer un individu de la base de donnée, à estimer le modèle et ensuite prédire la réponse de cet individu. Ce procédé est réalisé pour chaque individu et la fin l'erreur de prédiction du modèle sera la fréquence de mal classé des individu.

Cette méthode est donc réalisé en incluant au fur et a mesure des composantes principales dans le modèles. On choisira donc le nombre de composante en fonction de ces erreurs.

```
summary(plsdonnee)
```

```
## Data:    X dimension: 60 3571
##  Y dimension: 60 1
## Fit method: kernelpls
## Number of components considered: 58
##
## VALIDATION: RMSEP
## Cross-validated using 60 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.4851   0.2915   0.2218   0.2260   0.2067   0.2076   0.2077
## adjCV       0.4851   0.2909   0.2202   0.2231   0.2055   0.2061   0.2061
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.2075   0.2078   0.2082    0.2083    0.2083    0.2084    0.2084
## adjCV   0.2058   0.2061   0.2065    0.2065    0.2066    0.2067    0.2067
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.2084    0.2084    0.2084    0.2084    0.2084    0.2084    0.2084
## adjCV    0.2067    0.2067    0.2067    0.2067    0.2067    0.2067    0.2067
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV       0.2084    0.2084    0.2084    0.2084    0.2084    0.2084    0.2084
## adjCV    0.2067    0.2067    0.2067    0.2067    0.2067    0.2067    0.2067
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV       0.2084    0.2084    0.2084    0.2084    0.2084    0.2084    0.2084
## adjCV    0.2067    0.2067    0.2067    0.2067    0.2067    0.2067    0.2067
##        35 comps  36 comps  37 comps  38 comps  39 comps  40 comps  41 comps
## CV       0.2084    0.2084    0.2084    0.2084    0.2084    0.2084    0.2084
## adjCV    0.2067    0.2067    0.2067    0.2067    0.2067    0.2067    0.2067
##        42 comps  43 comps  44 comps  45 comps  46 comps  47 comps  48 comps
## CV       0.2084    0.2084    0.2084    0.2084    0.2081    0.4424     24.58
## adjCV    0.2067    0.2067    0.2067    0.2067    0.2064    0.4387     24.38
##        49 comps  50 comps  51 comps   52 comps   53 comps   54 comps   55 comps
## CV         1450    109834   9345477  682233248  1.712e+10  6.515e+10  5.905e+10
## adjCV      1438    108915   9267271  676524083  1.697e+10  6.461e+10  5.856e+10
##         56 comps  57 comps   58 comps
## CV     5.936e+10  5.94e+10  5.961e+10
## adjCV  5.887e+10  5.89e+10  5.911e+10
##
```

```
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    10.61    16.64    24.74    35.75    39.07    42.36    45.33    47.82
## Y    75.24    91.70    96.16    98.33    99.44    99.80    99.94    99.98
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    49.86      52.2     54.54        56     57.68     59.01     60.31
## Y   100.00     100.0    100.00       100    100.00    100.00    100.00
##    16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X      61.7     63.04     64.32     65.66     66.97      68.4     69.55
## Y     100.0    100.00    100.00    100.00    100.00     100.0    100.00
##    23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X     70.55     71.68      72.7      73.6     74.58     75.53     76.39
## Y    100.00    100.00     100.0     100.0    100.00    100.00    100.00
##    30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X     77.34     78.35      79.3     80.24     81.04     81.91     82.87
## Y    100.00    100.00     100.0    100.00    100.00    100.00    100.00
##    37 comps  38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X     83.79     84.62     85.32     86.11     86.94     87.68     88.42
## Y    100.00    100.00    100.00    100.00    100.00    100.00    100.00
##    44 comps  45 comps  46 comps  47 comps  48 comps  49 comps  50 comps
## X     89.19     89.97     90.71     91.53     92.28     93.03     93.78
## Y    100.00    100.00    100.00    100.00    100.00    100.00    100.00
##    51 comps  52 comps  53 comps  54 comps  55 comps  56 comps  57 comps
## X     94.52     95.21     96.21     97.18     98.15     99.13     100.1
## Y    100.00    100.00    100.00    100.00    100.00    100.00     100.0
##    58 comps
## X     101.1
## Y     100.0
```

```
#selectNcomp(plsdonnee, method = "randomization", plot = F)
# plot(RMSEP(plsdonnee), legendpos = "topright", ylim = c(0, 0.5))
```

On s'intéresse à la sortie `VALIDATION: RMSEP` qui renseigne sur l'erreur quadratique moyenne de prédiction. On remarque qu'avec que la constante dans le modèle l'erreur est de 0.48. En rajoutant une nouvelle composante, on note une baisse signifiactive des erreur qui passe à 0.29. Un nouvelle ajout permet de diminuer l'erreur à 0.22. Mais à partir de l'ajout de la troisièmes composantes, l'erreur quadratique ne change plus significativement. On se propose donc de ne prendre que le deux premières composantes pour la réalisation du modèle.

```
ncomponents = 2
pls.reduction.matrix <- plsdonnee$loadings[, 1:ncomponents]
donnee.train$pls.reducedX <- plsdonnee$scores[, 1:ncomponents]
pls.model <- glm(Y ~ pls.reducedX, data = donnee.train, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(pls.model)
```

```
##
## Call:
```

```
## glm(formula = Y ~ pls.reducedX, family = binomial, data = donnee.train)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -3.036e-05  -2.110e-08  -2.110e-08   2.110e-08   1.945e-05
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -19.078  27241.940  -0.001    0.999
## pls.reducedXComp 1     2.810   2818.355   0.001    0.999
## pls.reducedXComp 2     1.733   2204.449   0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7.7694e+01  on 59  degrees of freedom
## Residual deviance: 1.6022e-09  on 57  degrees of freedom
## AIC: 6
##
## Number of Fisher Scoring iterations: 25
```

Par contre avec la régression logistique, l'algorithme ne converge pas pour les deux premières composante. On prend que la première.

```
donnee.train$pls.reducedX <- as.matrix(plsdonnee$scores[, 1])
pls.model1 <- glm(Y ~ pls.reducedX, data = donnee.train, family = binomial)
summary(pls.model)
```

```
##
## Call:
## glm(formula = Y ~ pls.reducedX, family = binomial, data = donnee.train)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -3.036e-05  -2.110e-08  -2.110e-08   2.110e-08   1.945e-05
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -19.078  27241.940  -0.001    0.999
## pls.reducedXComp 1     2.810   2818.355   0.001    0.999
## pls.reducedXComp 2     1.733   2204.449   0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7.7694e+01  on 59  degrees of freedom
## Residual deviance: 1.6022e-09  on 57  degrees of freedom
## AIC: 6
##
## Number of Fisher Scoring iterations: 25
```

```
anova(pls.model1,pls.model,  test = "LRT")
```

```
## Analysis of Deviance Table
```

```
##
## Model 1: Y ~ pls.reducedX
## Model 2: Y ~ pls.reducedX
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1        58     10.995
## 2        57      0.000  1   10.995 0.0009134 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Estimons l'erreur quadratique moyen de notre modèle à partir des données de test.

```
# projection des individus
pls.reduction.matrix <- as.matrix(plsdonnee$loadings[, 1:2])
donnee.test$pls.reducedX <- x_test_centered %*% pls.reduction.matrix
test.prediction2 <- predict(pls.model, newdata = donnee.test, type="response")
mean((test.prediction2-donnee.test$Y)^2)
```

```
## [1] 2.914299e-14
```

```
table(test.prediction2> 0.5, donnee.test$Y)
```

```
##
##         0 1
##   FALSE 8 0
##   TRUE  0 4
```