TFM

Juan Alberto Gomez Sanchez

28 de marzo de 2018

Preparación del documento

En el presente documento se resume el código utilizado para la realización del proyecto, incluyendo la descripción de cada proceso realizado y la explicación de los resultados obtenidos.

El primer paso es indicar el directorio de trabajo que se va a utilizar, en el deben de estar incluidos los diferentes grupos de datos que se van a utilizar, tanto la matriz de expresión en formato .txt como la carpeta con todos los archivos en formato .CEL.

```
setwd("C:/Users/Juanal/Desktop/Bioinformatica/TFM/Datos")
```

Ahora se van a instalar los paquetes necesarios para realizar todas las funciones del proyecto, en primer lugar se instalan los paquetes correspondientes al repositorio CRAN y posteriormente los correspondientes a Bioconductor.

```
chooseCRANmirror(graphics = getOption("menu.graphics"), ind = 1, local.only = FALSE)

install.packages("factoextra")
```

```
## Installing package into 'C:/Users/Juanal/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'factoextra' is in use and will not be installed
```

```
install.packages("ggplot2")
```

```
## Installing package into 'C:/Users/Juanal/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'ggplot2' is in use and will not be installed
```

```
install.packages("cluster")
```

```
## Installing package into 'C:/Users/Juanal/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'cluster' is in use and will not be installed
```

```
install.packages("FactoMineR")
```

```
## Installing package into 'C:/Users/Juanal/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'FactoMineR' is in use and will not be installed
```

```
install.packages("clValid")
```

```
## Installing package into 'C:/Users/Juanal/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## package 'clValid' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Juanal\AppData\Local\Temp\RtmpoD9YKk\downloaded_packages
```

```
install.packages("bindrcpp")
```

```
## Installing package into 'C:/Users/Juanal/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'bindrcpp' is in use and will not be installed
```

```
install.packages("mclust")
```

```
## Installing package into 'C:/Users/Juanal/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'mclust' is in use and will not be installed
```

```
install.packages("optCluster")
```

```
## Installing package into 'C:/Users/Juanal/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'optCluster' is in use and will not be installed
```

```
library(factoextra)
library(cluster)
library(FactoMineR)
```

```
source("https://bioconductor.org/biocLite.R")
```

```
## Bioconductor version 3.7 (BiocInstaller 1.30.0), ?biocLite for help
```

```
biocLite("GEOquery")
```

```
## BioC_mirror: https://bioconductor.org
```

```
## Using Bioconductor 3.7 (BiocInstaller 1.30.0), R 3.5.0 (2018-04-23).
```

```
## Installing package(s) 'GEOquery'
```

```
## Warning: package 'GEOquery' is in use and will not be installed
```

```
## installation path not writeable, unable to update packages: MASS, survival
```

```
## Old packages: 'data.table', 'kohonen', 'openxlsx', 'pillar', 'stringi',
##   'utf8'
```

```
biocLite("preprocessCore")
```

```
## BioC_mirror: https://bioconductor.org
```

```
## Using Bioconductor 3.7 (BiocInstaller 1.30.0), R 3.5.0 (2018-04-23).
```

```
## Installing package(s) 'preprocessCore'
```

```
## Warning: package 'preprocessCore' is in use and will not be installed
```

```
## installation path not writeable, unable to update packages: MASS, survival
```

```
## Old packages: 'data.table', 'kohonen', 'openxlsx', 'pillar', 'stringi',
##   'utf8'
```

```
biocLite("genefilter")
```

```
## BioC_mirror: https://bioconductor.org
```

```
## Using Bioconductor 3.7 (BiocInstaller 1.30.0), R 3.5.0 (2018-04-23).
```

```
## Installing package(s) 'genefilter'
```

```
## Warning: package 'genefilter' is in use and will not be installed
## installation path not writeable, unable to update packages: MASS, survival
## Old packages: 'data.table', 'kohonen', 'openxlsx', 'pillar', 'stringi',
##   'utf8'
```

```
biocLite("hgu133plus2")
```

```
## BioC_mirror: https://bioconductor.org
## Using Bioconductor 3.7 (BiocInstaller 1.30.0), R 3.5.0 (2018-04-23).
## Installing package(s) 'hgu133plus2'
## Warning: package 'hgu133plus2' is not available (for R version 3.5.0)
## installation path not writeable, unable to update packages: MASS, survival
## Old packages: 'data.table', 'kohonen', 'openxlsx', 'pillar', 'stringi',
##   'utf8'
```

```
biocLite("affy")
```

```
## BioC_mirror: https://bioconductor.org
## Using Bioconductor 3.7 (BiocInstaller 1.30.0), R 3.5.0 (2018-04-23).
## Installing package(s) 'affy'
## Warning: package 'affy' is in use and will not be installed
## installation path not writeable, unable to update packages: MASS, survival
## Old packages: 'data.table', 'kohonen', 'openxlsx', 'pillar', 'stringi',
##   'utf8'
```

```
biocLite("annotate")
```

```
## BioC_mirror: https://bioconductor.org
## Using Bioconductor 3.7 (BiocInstaller 1.30.0), R 3.5.0 (2018-04-23).
## Installing package(s) 'annotate'
## Warning: package 'annotate' is in use and will not be installed
## installation path not writeable, unable to update packages: MASS, survival
## Old packages: 'data.table', 'kohonen', 'openxlsx', 'pillar', 'stringi',
##   'utf8'
```

```
biocLite("GO.db")
```

```
## BioC_mirror: https://bioconductor.org
## Using Bioconductor 3.7 (BiocInstaller 1.30.0), R 3.5.0 (2018-04-23).
## Installing package(s) 'GO.db'
## installing the source package 'GO.db'
## installation path not writeable, unable to update packages: MASS, survival
## Old packages: 'data.table', 'kohonen', 'openxlsx', 'pillar', 'stringi',
##   'utf8'
```

```
biocLite("Biobase")
```

```
## BioC_mirror: https://bioconductor.org

## Using Bioconductor 3.7 (BiocInstaller 1.30.0), R 3.5.0 (2018-04-23).

## Installing package(s) 'Biobase'

## Warning: package 'Biobase' is in use and will not be installed

## installation path not writeable, unable to update packages: MASS, survival

## Old packages: 'data.table', 'kohonen', 'openxlsx', 'pillar', 'stringi',
##   'utf8'
```

```
library(Biobase)
library(preprocessCore)
library(GEOquery)
library(tidyr)
library(genefilter)
library(affy)
library(hgu133plus2.db)
```

Introducción

Obtención de los datos de expresión

Los datos se encuentran en un archivo tipo GSE proveniente de los servidores del ncbi, por lo cuál es necesario utilizar la función `getGEO` para obtenerlos.
Ahora a partir del set de datos se extraen los resultados fenotípicos de los pacientes con las funciones `phenoData` y `pData`. Estos se utilizarán para obtener las clases de cada muestra, que se utilizarán para realizar diferentes análisis de los datos.

```
gse <- getGEO(filename = "GSE5764_series_matrix.txt.gz")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   ID_REF = col_character()
## )

## See spec(...) for full column specifications.

## File stored at:

## C:\Users\Juanal\AppData\Local\Temp\RtmpoD9YKk/GPL570.soft
```

```
patient <- pData(phenoData(gse))
```

El tratamiento de la información fenotípica de los pacientes se lleva a cabo extrayendo la información que indica el tipo celular y el estado mediante la obtención de la primera columna del archivo `patient`.

A continuación se muestran las primeras líneas del archivo `patient` para poder observar como se almacena este tipo de información.

```
head(patient[1])
```

```
##                          title
## GSM134584  6ILC_normal_ductal
## GSM134586 6ILC_normal_lobular
## GSM134587  6ILC_tumor_lobular
```

```
## GSM134588  9ILC_normal_ductal
## GSM134589 9ILC_normal_lobular
## GSM134591  9ILC_tumor_lobular
```

Para darle uniformidad a las clases se eliminan los elementos identificativos de cada paciente para dejar sólo la información de clase.
Se generan dos archivos, uno con las clases en forma numérica y otro con las clases en forma de caracteres resumidos. Para poder utilizarlo después se guardan en formato .csv.

```
clase <- patient[, 1]
clase <- as.character(clase)
clase.a <- substr(clase, 6, 20)
clase.a <- gsub("_", "", clase.a)
clase.a <- gsub(" ", "", clase.a)
clase.a <- as.matrix(clase.a)
row.names(clase.a) = row.names(patient)
colnames(clase.a) = "Clase"
clase.f <- clase.a
clase.f[clase.f == "normalductal"] = 1
clase.f[clase.f == "normallobular"] = 2
clase.f[clase.f == "tumorlobular"] = 3
clase.f[clase.f == "tumorductal"] = 4
clase.f <- as.vector(clase.f)
clase.a <- factor(clase.a, levels = c("normalductal", "normallobular", "tumorlobular",
    "tumorductal"))
clase <- as.data.frame(clase.a)
rownames(clase) <- rownames(patient)
write.csv(clase, file = "clase.csv")
```

Para facilitar la interpretación de los gráficos se van a crear unas abreviaturas para cada muestra, estas se incluyen en el siguiente data frame, donde se asocia cada muestra con su estado y tipo celular y su abreviatura asignada.

```
estadocel <- clase.f
estadocel[estadocel == 1] = "normal"
estadocel[estadocel == 2] = "normal"
estadocel[estadocel == 3] = "tumoral"
estadocel[estadocel == 4] = "tumoral"
tipocel <- clase.f
tipocel[tipocel == 1] = "ductal"
tipocel[tipocel == 4] = "ductal"
tipocel[tipocel == 2] = "lobular"
tipocel[tipocel == 3] = "lobular"
abreviatura <- c("1ND", "1NL", "1TL", "2ND", "2NL", "2TL", "3ND", "3NL", "3TL",
    "4ND", "4NL", "4TL", "5ND", "5NL", "5TL", "6ND", "6NL", "1TD", "7ND", "7NL",
    "2TD", "8ND", "8NL", "3TD", "9ND", "9NL", "4TD", "10ND", "10NL", "5TD")

resclases <- cbind(estadocel, tipocel, abreviatura)
rownames(resclases) <- rownames(clase)
as.data.frame(resclases)
```

```
##           estadocel tipocel abreviatura
## GSM134584    normal  ductal         1ND
## GSM134586    normal lobular         1NL
## GSM134587   tumoral lobular         1TL
## GSM134588    normal  ductal         2ND
```

```
## GSM134589    normal lobular         2NL
## GSM134591   tumoral lobular         2TL
## GSM134687    normal  ductal         3ND
## GSM134688    normal lobular         3NL
## GSM134689   tumoral lobular         3TL
## GSM134690    normal  ductal         4ND
## GSM134691    normal lobular         4NL
## GSM134692   tumoral lobular         4TL
## GSM134693    normal  ductal         5ND
## GSM134694    normal lobular         5NL
## GSM134695   tumoral lobular         5TL
## GSM134696    normal  ductal         6ND
## GSM134697    normal lobular         6NL
## GSM134698   tumoral  ductal         1TD
## GSM134699    normal  ductal         7ND
## GSM134700    normal lobular         7NL
## GSM134701   tumoral  ductal         2TD
## GSM134702    normal  ductal         8ND
## GSM134703    normal lobular         8NL
## GSM134704   tumoral  ductal         3TD
## GSM134705    normal  ductal         9ND
## GSM134706    normal lobular         9NL
## GSM134707   tumoral  ductal         4TD
## GSM134708    normal  ductal        10ND
## GSM134709    normal lobular        10NL
## GSM134710   tumoral  ductal         5TD
```

En este punto se van a utilizar los archivos en formato .CEL, que se van a cargar con la función `read.affybatch`. El formato .CEL es el utilizado por Affymetrix para guardar la información de los estudios de intensidad de expresión genética, de forma que se puedan utilizar para extraer estos datos en colaboración con los archivos con la información fenotípica de las muestras. Para la información fenotípica que requiere esta función se carga el archivo .csv obtenido con la función `read.AnnotatedDataFrame` anteriormente con las clases.

```
celFiles <- list.celfiles("./GSE5764_RAW", full.names = TRUE)
my.targets <- read.AnnotatedDataFrame(file.path(getwd(), "clase.csv"), header = TRUE,
    row.names = 1, sep = ",")
rawData <- read.affybatch(filename = celFiles, phenoData = my.targets)
```

Debido a que la forma de los datos no es uniforme se debe escalar para que las medidas sean similares, para ello se utiliza la función `rma`. El resultado de este proceso se mejora implementando la anotación para este tipo de microarray.
El volumen de los datos es demasiado grande para ser procesado de manera eficiente por las funciones que se van a utilizar, por ello se plantea un filtro con la función `genefilter` que recogerá los datos de las probes más importantes, eliminando el resto.
El último paso de la preparación de los datos es generar un archivo con la matriz transpuesta, ya que algunas funciones utilizan esta forma.

```
eset_rma <- rma(rawData)
```

```
## Background correcting
## Normalizing
## Calculating Expression
```

```
annotation(eset_rma) <- "hgu133plus2.db"
eset_filter <- nsFilter(eset_rma, var.cutoff = 0.99)
eset_filter <- eset_filter$eset
```

```
data <- exprs(eset_filter)
data.t <- t(data)
data.m <- data.t
rownames(data.t) <- abreviatura
```

Análisis de los datos

En este momento se tienen los datos preparados para realizar los diferentes procesos de clustering, pero antes de esto es necesario realizar una revisión del estado de los datos de forma que se pueda ver si alguno de los procesos preparatorios no se ha realizado correctamente o puede ser mejorado.

En primer lugar se muestran las dimensiones de los datos originales, antes de ser filtrados.

```
dim(eset_rma)
```

```
## Features  Samples
##    54675       30
```

Debido al gran volumen de los datos muchos de los algoritmos no funcionaban de forma correcta, por lo cual se ha elegido un método de filtrado para reducir el volumen a una cantidad que pueda ser manejada de forma eficiente. El filtraje se realizó a un nivel de 0.99, lo que dejó un total de variables como las que se muestran.

```
dim(eset_filter)
```

```
## Features  Samples
##      202       30
```

Ahora se realiza un sumario (`summary`) de las muestras, de esta forma se puede comprobar como los datos estan escalados de tal forma que las medias de las muestras son similares y se distribuyen alrededor de valores cercanos.
También se analiza la distribución de las clases, lo que revela que dos tercios de las observaciones pertenecen a células no tumorales y la mitad son de cada tipo celular.

```
x <- summary(data)
x[, 1:30]
```

```
##   "GSM134584"      "GSM134586"      "GSM134587"      "GSM134588"
##  Min.   : 2.477   Min.   : 2.671   Min.   : 2.845   Min.   : 2.555
##  1st Qu.: 3.615   1st Qu.: 4.341   1st Qu.: 4.523   1st Qu.: 5.152
##  Median : 5.095   Median : 7.086   Median : 7.017   Median : 6.920
##  Mean   : 6.096   Mean   : 6.788   Mean   : 6.766   Mean   : 6.775
##  3rd Qu.: 8.581   3rd Qu.: 8.709   3rd Qu.: 8.231   3rd Qu.: 8.310
##  Max.   :13.126   Max.   :13.107   Max.   :12.618   Max.   :11.739
##   "GSM134589"      "GSM134591"      "GSM134687"      "GSM134688"
##  Min.   : 2.700   Min.   : 2.672   Min.   : 2.681   Min.   : 2.583
##  1st Qu.: 5.113   1st Qu.: 5.215   1st Qu.: 5.614   1st Qu.: 4.060
##  Median : 6.675   Median : 6.568   Median : 7.030   Median : 6.401
##  Mean   : 6.728   Mean   : 6.762   Mean   : 7.126   Mean   : 6.518
##  3rd Qu.: 8.156   3rd Qu.: 8.325   3rd Qu.: 8.524   3rd Qu.: 8.691
##  Max.   :12.003   Max.   :13.060   Max.   :13.461   Max.   :13.347
##   "GSM134689"      "GSM134690"      "GSM134691"      "GSM134692"
##  Min.   : 2.431   Min.   : 2.554   Min.   : 2.473   Min.   : 2.625
##  1st Qu.: 4.316   1st Qu.: 6.172   1st Qu.: 5.595   1st Qu.: 3.482
##  Median : 6.617   Median : 7.079   Median : 7.492   Median : 4.248
##  Mean   : 6.623   Mean   : 7.214   Mean   : 7.223   Mean   : 5.009
##  3rd Qu.: 8.346   3rd Qu.: 8.307   3rd Qu.: 8.740   3rd Qu.: 5.685
```

```
##  Max.   :13.146   Max.   :13.488   Max.   :13.446   Max.   :12.203
##   "GSM134693"      "GSM134694"      "GSM134695"      "GSM134696"
##  Min.   : 3.008   Min.   : 2.721   Min.   : 2.462   Min.   : 2.432
##  1st Qu.: 4.839   1st Qu.: 4.880   1st Qu.: 4.304   1st Qu.: 4.581
##  Median : 6.318   Median : 6.413   Median : 5.786   Median : 6.890
##  Mean   : 6.597   Mean   : 6.506   Mean   : 6.100   Mean   : 6.676
##  3rd Qu.: 7.760   3rd Qu.: 7.897   3rd Qu.: 7.296   3rd Qu.: 8.437
##  Max.   :11.898   Max.   :11.795   Max.   :12.327   Max.   :12.884
##   "GSM134697"      "GSM134698"      "GSM134699"      "GSM134700"
##  Min.   : 2.741   Min.   : 2.594   Min.   : 2.623   Min.   : 2.720
##  1st Qu.: 4.990   1st Qu.: 4.128   1st Qu.: 5.846   1st Qu.: 6.271
##  Median : 7.013   Median : 5.948   Median : 7.128   Median : 7.574
##  Mean   : 6.917   Mean   : 6.020   Mean   : 7.101   Mean   : 7.372
##  3rd Qu.: 8.698   3rd Qu.: 7.501   3rd Qu.: 8.552   3rd Qu.: 8.690
##  Max.   :13.088   Max.   :12.340   Max.   :12.437   Max.   :12.095
##   "GSM134701"      "GSM134702"      "GSM134703"      "GSM134704"
##  Min.   : 2.820   Min.   : 2.865   Min.   : 2.792   Min.   : 2.661
##  1st Qu.: 5.256   1st Qu.: 5.857   1st Qu.: 4.424   1st Qu.: 3.978
##  Median : 7.187   Median : 6.987   Median : 5.816   Median : 5.947
##  Mean   : 6.973   Mean   : 7.139   Mean   : 6.194   Mean   : 6.038
##  3rd Qu.: 8.556   3rd Qu.: 8.247   3rd Qu.: 7.467   3rd Qu.: 7.497
##  Max.   :12.835   Max.   :12.949   Max.   :12.967   Max.   :12.613
##   "GSM134705"      "GSM134706"      "GSM134707"      "GSM134708"
##  Min.   : 2.923   Min.   : 2.845   Min.   : 2.767   Min.   : 2.522
##  1st Qu.: 5.623   1st Qu.: 4.342   1st Qu.: 4.294   1st Qu.: 5.302
##  Median : 6.620   Median : 5.540   Median : 5.442   Median : 7.233
##  Mean   : 6.745   Mean   : 5.796   Mean   : 5.947   Mean   : 7.063
##  3rd Qu.: 7.912   3rd Qu.: 6.708   3rd Qu.: 7.116   3rd Qu.: 8.621
##  Max.   :12.112   Max.   :12.007   Max.   :13.155   Max.   :13.214
##   "GSM134709"      "GSM134710"
##  Min.   : 2.529   Min.   : 2.837
##  1st Qu.: 5.395   1st Qu.: 4.510
##  Median : 7.424   Median : 6.094
##  Mean   : 7.143   Mean   : 6.208
##  3rd Qu.: 8.523   3rd Qu.: 7.430
##  Max.   :13.152   Max.   :12.863
```

```
table(clase.a)
```

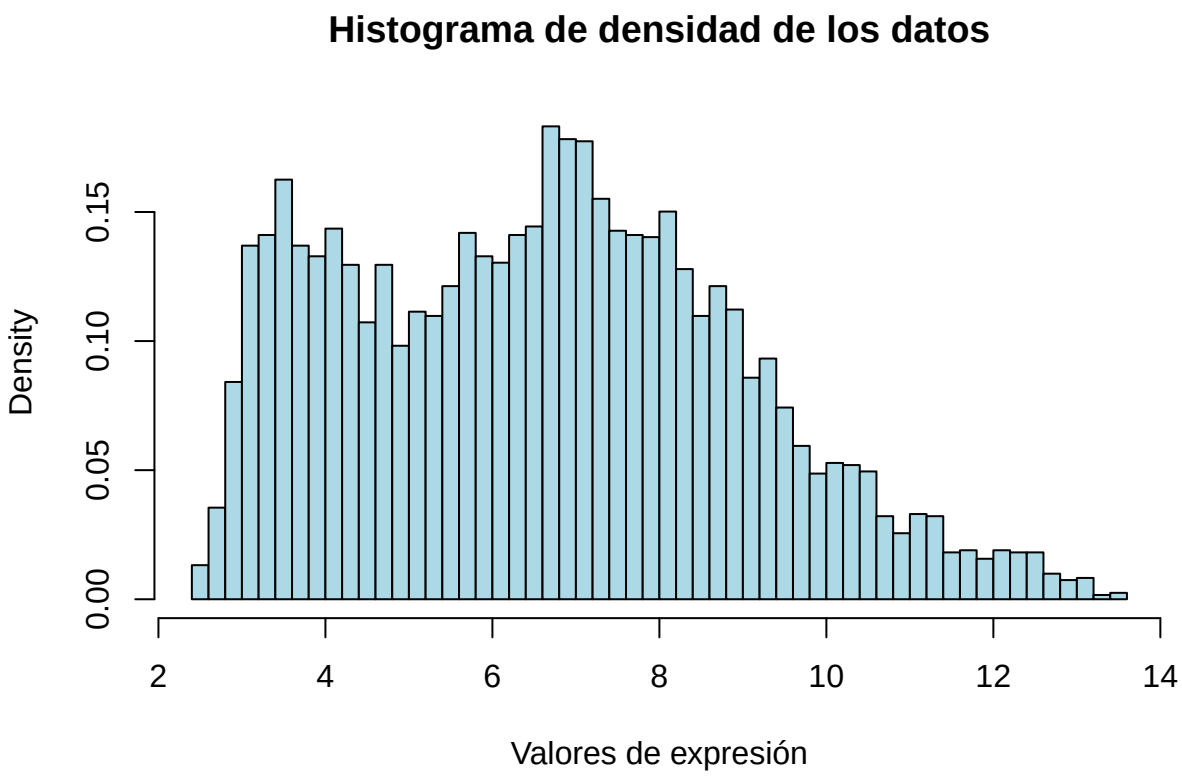
```
## clase.a
##  normalductal normallobular  tumorlobular   tumorductal
##            10            10             5             5
```

```
prop.table(table(clase.a))
```

```
## clase.a
##  normalductal normallobular  tumorlobular   tumorductal
##     0.3333333     0.3333333     0.1666667     0.1666667
```

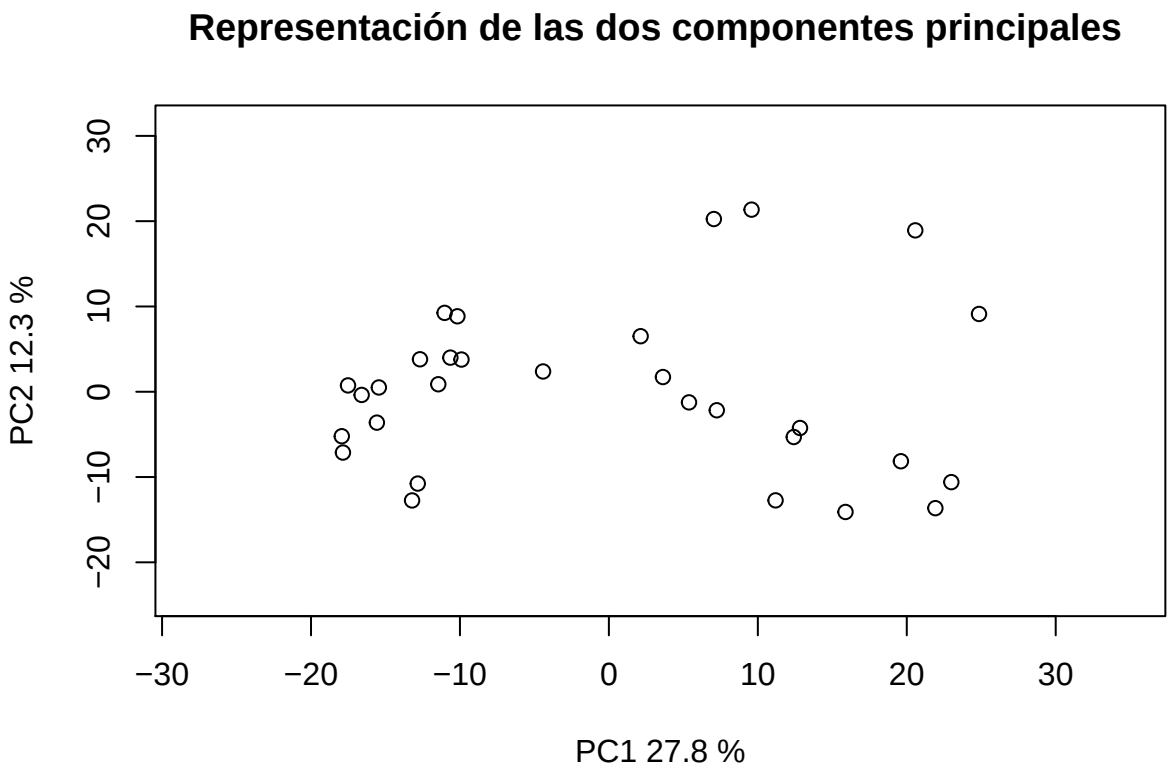
En este punto se pueden realizar varias representaciones gráficas que aporten una idea general de los datos. En primer lugar se realiza un histograma (`hist`) de densidad, cuyo resultado indica que la mayoría de las variables se distribuyen en la zona de valores entre 3 y 10 . Esto puede indicar un buen escalado de los datos.

```
hist(data, col = "lightblue", freq = F, breaks = 50, main = "Histograma de densidad de los datos",
    xlab = "Valores de expresión")
```



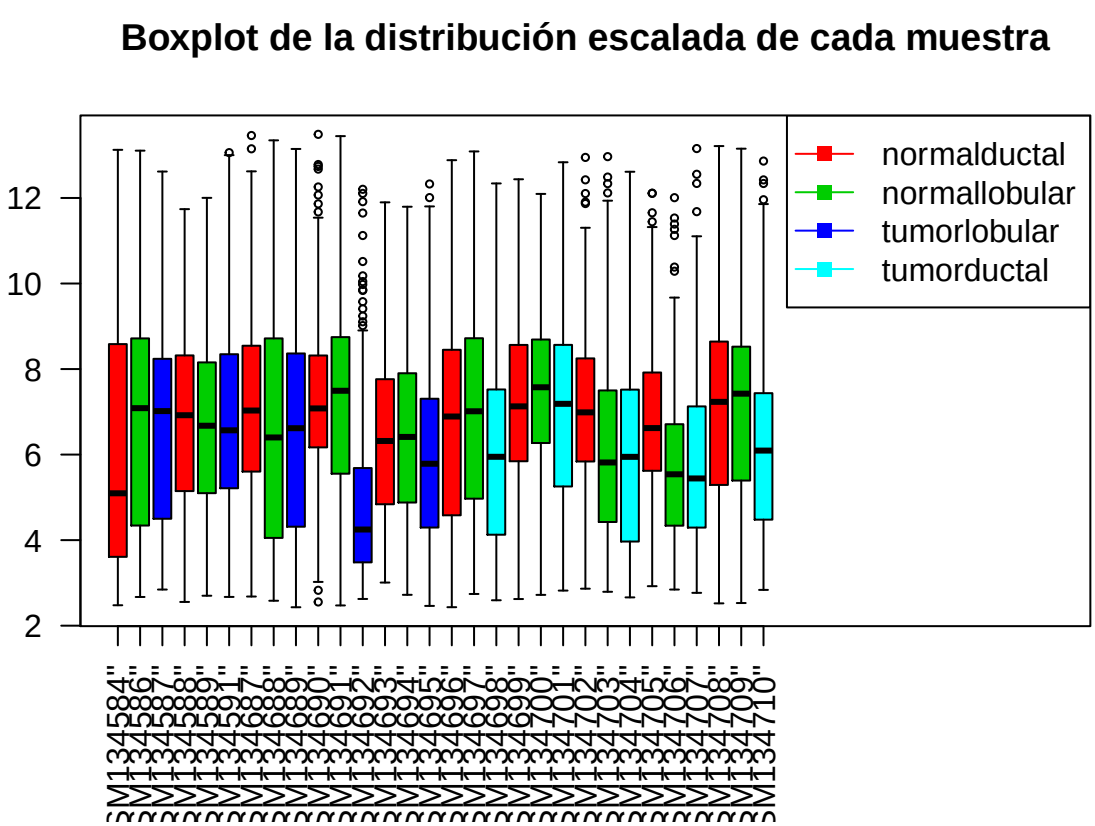
El segundo tipo de representación que se puede realizar es un gráfico de PCA, para ello es necesario crear una función donde se haga un análisis de las componentes principales de los datos y se representen en un gráfico de puntos las observaciones en función de estas componentes principales.
El resultado es una nube de puntos donde la mayoría se agrupan entre el -20% y el 25% de la componente 1 y -20% y 20% de la compontente 2.

```
plotPCA <- function(X, labels = NULL, colors = NULL, dataDesc = "", scale = TRUE) {
    pcX <- prcomp(X)
    loads <- round(pcX$sdev^2/sum(pcX$sdev^2) * 100, 1)
    xlab <- c(paste("PC1", loads[1], "%"))
    ylab <- c(paste("PC2", loads[2], "%"))
    if (is.null(colors))
        colors = 1
    plot(pcX$x[, 1:2], xlab = xlab, ylab = ylab, col = colors, xlim = c(min(pcX$x[,
        1]) - 10, max(pcX$x[, 1]) + 10), ylim = c(min(pcX$x[, 2]) - 10, max(pcX$x[,
        2]) + 10))
    text(pcX$x[, 1], pcX$x[, 2], labels, pos = 3, cex = 0.8)
    title("Representación de las dos componentes principales ", cex = 0.8)
}
plotPCA(data.m)
```



La siguiente representación es un boxplot (`boxplot`), que muestra la distribución de valores de cada observación. En este caso se ha indicado la clase de cada observación con un color diferente para facilitar la observación de diferencias. A primera vista no se observan diferencias tangibles entre clases, con la excepción de las muestras de la clase tumoral ductal que parecen tener menores valores.

```
col <- paste((as.numeric(clase.f) + 1), sep = ",")
boxplot(data, main = "Boxplot de la distribución escalada de cada muestra",
    col = c(col), lty = 1, las = 2, xlim = c(1, 43), cex = 0.5)
legend(x = "topright", legend = levels(clase.a), col = c(2, 3, 4, 5), lty = 1,
    pch = 15)
```



Por último se va a utilizar el paquete `factoextra` para representar la distancia euclidea entre las diferentes observaciones. Como se observa la distancia es bastante baja entre la mayoría de las muestras.

```
mat_dist <- dist(x = data.m, method = "euclidean")
round(as.matrix(mat_dist)[1:30, 1:30], 2)
```

```
##             "GSM134584" "GSM134586" "GSM134587" "GSM134588" "GSM134589"
## "GSM134584"        0.00       37.54       39.83       45.59       43.97
## "GSM134586"       37.54        0.00       36.95       42.25       39.17
## "GSM134587"       39.83       36.95        0.00       41.62       39.79
## "GSM134588"       45.59       42.25       41.62        0.00       16.17
## "GSM134589"       43.97       39.17       39.79       16.17        0.00
## "GSM134591"       44.86       42.26       33.79       39.02       36.70
## "GSM134687"       34.50       29.04       35.08       38.55       36.89
## "GSM134688"       35.91       34.39       39.12       42.07       39.97
## "GSM134689"       45.35       45.17       38.18       44.69       43.69
## "GSM134690"       38.42       33.11       35.99       36.36       33.49
## "GSM134691"       40.87       34.40       39.32       37.74       34.81
## "GSM134692"       45.70       49.59       45.36       51.04       50.30
## "GSM134693"       39.78       34.41       29.86       44.77       42.52
## "GSM134694"       41.68       37.02       31.55       46.40       44.31
## "GSM134695"       47.11       45.47       38.26       50.39       48.40
## "GSM134696"       44.52       39.21       40.52       32.14       30.79
## "GSM134697"       43.62       36.42       39.16       31.61       29.58
## "GSM134698"       51.83       49.01       41.03       50.17       48.35
## "GSM134699"       39.05       33.25       33.07       34.81       33.40
## "GSM134700"       41.56       35.47       33.77       36.17       34.07
```

```
## "GSM134701"       40.73       33.64       31.85       36.06       34.11
## "GSM134702"       42.97       37.41       29.12       38.13       35.92
## "GSM134703"       40.49       39.48       30.54       42.24       40.34
## "GSM134704"       44.50       46.39       37.81       43.99       42.80
## "GSM134705"       39.10       37.10       32.56       38.61       35.43
## "GSM134706"       40.83       41.39       37.63       41.27       38.54
## "GSM134707"       46.76       48.73       41.50       46.83       45.04
## "GSM134708"       41.69       39.64       40.67       32.73       32.31
## "GSM134709"       38.90       35.90       36.31       33.64       30.36
## "GSM134710"       45.38       46.01       37.98       43.63       42.39
##             "GSM134591" "GSM134687" "GSM134688" "GSM134689" "GSM134690"
## "GSM134584"       44.86       34.50       35.91       45.35       38.42
## "GSM134586"       42.26       29.04       34.39       45.17       33.11
## "GSM134587"       33.79       35.08       39.12       38.18       35.99
## "GSM134588"       39.02       38.55       42.07       44.69       36.36
## "GSM134589"       36.70       36.89       39.97       43.69       33.49
## "GSM134591"        0.00       40.14       43.23       33.09       41.28
## "GSM134687"       40.14        0.00       29.38       41.29       27.13
## "GSM134688"       43.23       29.38        0.00       45.41       33.14
## "GSM134689"       33.09       41.29       45.41        0.00       44.04
## "GSM134690"       41.28       27.13       33.14       44.04        0.00
## "GSM134691"       43.15       32.21       36.03       47.62       25.72
## "GSM134692"       43.12       49.68       49.84       42.27       51.20
## "GSM134693"       34.42       33.62       39.15       42.77       37.53
## "GSM134694"       35.41       36.68       41.74       43.62       40.18
## "GSM134695"       39.45       46.67       50.14       46.13       48.36
## "GSM134696"       43.51       34.19       37.77       47.56       33.84
## "GSM134697"       43.61       29.69       36.73       46.14       29.59
## "GSM134698"       38.54       48.82       51.35       44.13       51.00
## "GSM134699"       38.13       28.99       34.97       43.90       29.53
## "GSM134700"       36.81       30.81       38.84       44.96       30.42
## "GSM134701"       36.89       31.23       37.10       40.75       30.68
## "GSM134702"       30.73       33.76       39.67       32.72       33.75
## "GSM134703"       32.86       37.96       40.50       32.70       38.40
## "GSM134704"       35.07       43.04       45.37       33.13       44.28
## "GSM134705"       32.83       32.94       36.13       40.59       32.85
## "GSM134706"       36.97       40.34       40.53       41.44       39.84
## "GSM134707"       39.93       45.49       46.64       36.14       45.84
## "GSM134708"       39.34       31.88       36.58       44.44       33.16
## "GSM134709"       37.68       30.48       36.26       45.22       29.88
## "GSM134710"       35.05       42.15       46.33       33.08       44.58
##             "GSM134691" "GSM134692" "GSM134693" "GSM134694" "GSM134695"
## "GSM134584"       40.87       45.70       39.78       41.68       47.11
## "GSM134586"       34.40       49.59       34.41       37.02       45.47
## "GSM134587"       39.32       45.36       29.86       31.55       38.26
## "GSM134588"       37.74       51.04       44.77       46.40       50.39
## "GSM134589"       34.81       50.30       42.52       44.31       48.40
## "GSM134591"       43.15       43.12       34.42       35.41       39.45
## "GSM134687"       32.21       49.68       33.62       36.68       46.67
## "GSM134688"       36.03       49.84       39.15       41.74       50.14
## "GSM134689"       47.62       42.27       42.77       43.62       46.13
## "GSM134690"       25.72       51.20       37.53       40.18       48.36
## "GSM134691"        0.00       53.61       40.52       42.91       50.87
## "GSM134692"       53.61        0.00       44.36       44.34       41.95
```

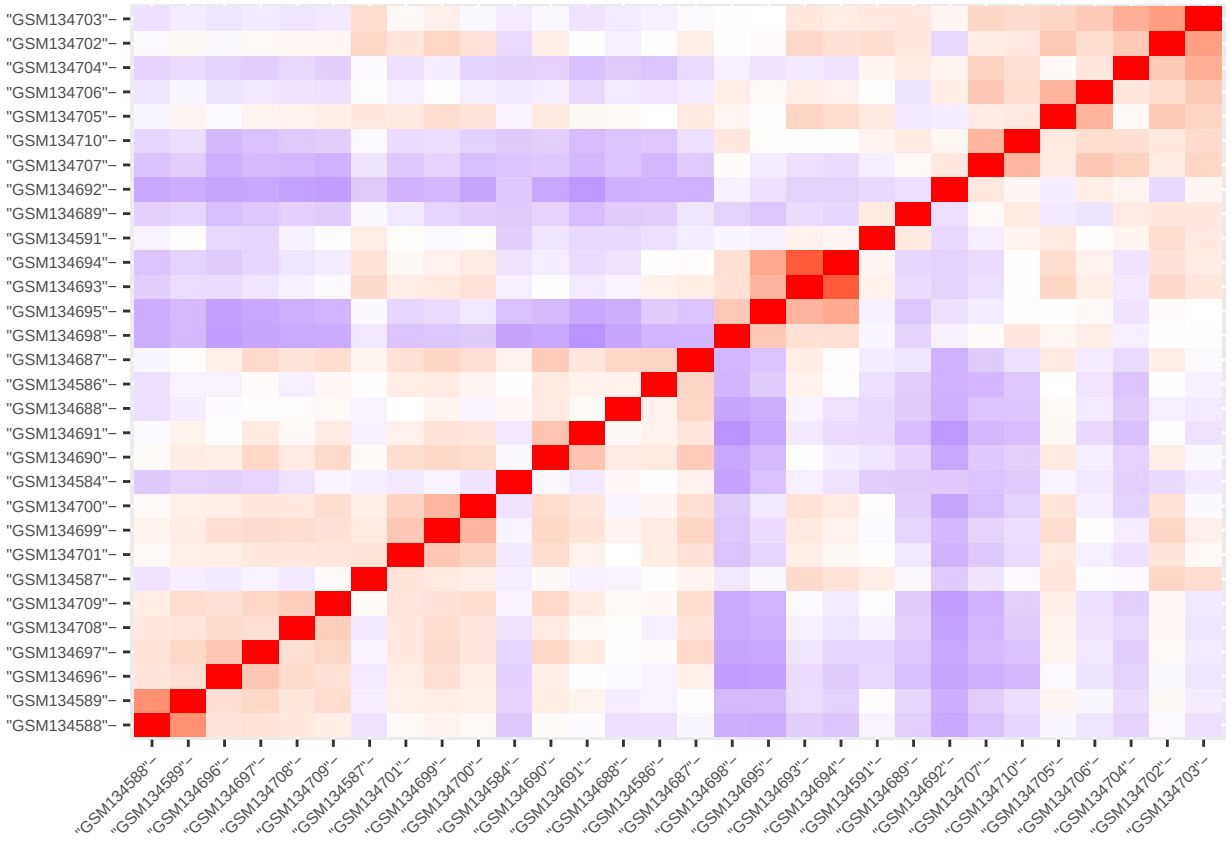
```
## "GSM134693"       40.52       44.36        0.00        7.45       22.78
## "GSM134694"       42.91       44.34        7.45        0.00       20.53
## "GSM134695"       50.87       41.95       22.78       20.53        0.00
## "GSM134696"       36.99       51.56       42.95       45.36       52.16
## "GSM134697"       33.08       51.16       41.30       43.72       50.93
## "GSM134698"       54.41       39.53       31.25       31.17       26.75
## "GSM134699"       31.60       48.63       32.96       34.71       42.84
## "GSM134700"       32.33       51.50       31.49       33.32       40.92
## "GSM134701"       34.57       49.43       33.93       35.97       43.69
## "GSM134702"       37.45       43.21       29.63       31.26       36.50
## "GSM134703"       41.81       35.31       32.48       33.44       37.11
## "GSM134704"       47.40       34.95       40.80       41.56       41.62
## "GSM134705"       35.94       40.00       28.96       30.46       37.00
## "GSM134706"       43.16       33.68       33.77       34.73       36.06
## "GSM134707"       48.41       32.61       42.04       42.69       40.13
## "GSM134708"       36.00       52.03       39.49       41.31       49.95
## "GSM134709"       33.24       52.48       37.93       40.35       49.08
## "GSM134710"       47.96       35.62       37.30       37.41       36.80
##             "GSM134696" "GSM134697" "GSM134698" "GSM134699" "GSM134700"
## "GSM134584"       44.52       43.62       51.83       39.05       41.56
## "GSM134586"       39.21       36.42       49.01       33.25       35.47
## "GSM134587"       40.52       39.16       41.03       33.07       33.77
## "GSM134588"       32.14       31.61       50.17       34.81       36.17
## "GSM134589"       30.79       29.58       48.35       33.40       34.07
## "GSM134591"       43.51       43.61       38.54       38.13       36.81
## "GSM134687"       34.19       29.69       48.82       28.99       30.81
## "GSM134688"       37.77       36.73       51.35       34.97       38.84
## "GSM134689"       47.56       46.14       44.13       43.90       44.96
## "GSM134690"       33.84       29.59       51.00       29.53       30.42
## "GSM134691"       36.99       33.08       54.41       31.60       32.33
## "GSM134692"       51.56       51.16       39.53       48.63       51.50
## "GSM134693"       42.95       41.30       31.25       32.96       31.49
## "GSM134694"       45.36       43.72       31.17       34.71       33.32
## "GSM134695"       52.16       50.93       26.75       42.84       40.92
## "GSM134696"        0.00       26.28       52.75       30.88       33.87
## "GSM134697"       26.28        0.00       51.33       30.24       32.50
## "GSM134698"       52.75       51.33        0.00       46.13       45.40
## "GSM134699"       30.88       30.24       46.13        0.00       22.92
## "GSM134700"       33.87       32.50       45.40       22.92        0.00
## "GSM134701"       33.76       32.59       46.68       26.49       28.52
## "GSM134702"       38.32       36.14       37.09       29.19       31.49
## "GSM134703"       41.22       40.44       37.47       34.14       38.24
## "GSM134704"       44.15       44.79       39.67       40.06       44.12
## "GSM134705"       37.90       35.02       35.61       30.39       31.94
## "GSM134706"       41.48       40.94       33.86       37.06       39.89
## "GSM134707"       49.93       48.23       36.41       44.01       47.57
## "GSM134708"       29.92       30.79       50.71       30.50       32.65
## "GSM134709"       31.26       29.30       50.61       31.29       30.71
## "GSM134710"       48.51       47.02       32.58       42.36       44.21
##             "GSM134701" "GSM134702" "GSM134703" "GSM134704" "GSM134705"
## "GSM134584"       40.73       42.97       40.49       44.50       39.10
## "GSM134586"       33.64       37.41       39.48       46.39       37.10
## "GSM134587"       31.85       29.12       30.54       37.81       32.56
## "GSM134588"       36.06       38.13       42.24       43.99       38.61
```

```
## "GSM134589"       34.11       35.92       40.34       42.80       35.43
## "GSM134591"       36.89       30.73       32.86       35.07       32.83
## "GSM134687"       31.23       33.76       37.96       43.04       32.94
## "GSM134688"       37.10       39.67       40.50       45.37       36.13
## "GSM134689"       40.75       32.72       32.70       33.13       40.59
## "GSM134690"       30.68       33.75       38.40       44.28       32.85
## "GSM134691"       34.57       37.45       41.81       47.40       35.94
## "GSM134692"       49.43       43.21       35.31       34.95       40.00
## "GSM134693"       33.93       29.63       32.48       40.80       28.96
## "GSM134694"       35.97       31.26       33.44       41.56       30.46
## "GSM134695"       43.69       36.50       37.11       41.62       37.00
## "GSM134696"       33.76       38.32       41.22       44.15       37.90
## "GSM134697"       32.59       36.14       40.44       44.79       35.02
## "GSM134698"       46.68       37.09       37.47       39.67       35.61
## "GSM134699"       26.49       29.19       34.14       40.06       30.39
## "GSM134700"       28.52       31.49       38.24       44.12       31.94
## "GSM134701"        0.00       32.17       36.16       41.91       32.86
## "GSM134702"       32.17        0.00       18.89       26.93       26.66
## "GSM134703"       36.16       18.89        0.00       21.69       29.07
## "GSM134704"       41.91       26.93       21.69        0.00       36.08
## "GSM134705"       32.86       26.66       29.07       36.08        0.00
## "GSM134706"       39.72       30.72       26.84       32.51       22.59
## "GSM134707"       45.65       33.52       29.25       28.33       33.21
## "GSM134708"       32.71       35.76       41.30       43.52       34.53
## "GSM134709"       32.26       35.64       40.83       44.64       33.85
## "GSM134710"       42.77       32.89       30.25       30.94       32.86
##             "GSM134706" "GSM134707" "GSM134708" "GSM134709" "GSM134710"
## "GSM134584"       40.83       46.76       41.69       38.90       45.38
## "GSM134586"       41.39       48.73       39.64       35.90       46.01
## "GSM134587"       37.63       41.50       40.67       36.31       37.98
## "GSM134588"       41.27       46.83       32.73       33.64       43.63
## "GSM134589"       38.54       45.04       32.31       30.36       42.39
## "GSM134591"       36.97       39.93       39.34       37.68       35.05
## "GSM134687"       40.34       45.49       31.88       30.48       42.15
## "GSM134688"       40.53       46.64       36.58       36.26       46.33
## "GSM134689"       41.44       36.14       44.44       45.22       33.08
## "GSM134690"       39.84       45.84       33.16       29.88       44.58
## "GSM134691"       43.16       48.41       36.00       33.24       47.96
## "GSM134692"       33.68       32.61       52.03       52.48       35.62
## "GSM134693"       33.77       42.04       39.49       37.93       37.30
## "GSM134694"       34.73       42.69       41.31       40.35       37.41
## "GSM134695"       36.06       40.13       49.95       49.08       36.80
## "GSM134696"       41.48       49.93       29.92       31.26       48.51
## "GSM134697"       40.94       48.23       30.79       29.30       47.02
## "GSM134698"       33.86       36.41       50.71       50.61       32.58
## "GSM134699"       37.06       44.01       30.50       31.29       42.36
## "GSM134700"       39.89       47.57       32.65       30.71       44.21
## "GSM134701"       39.72       45.65       32.71       32.26       42.77
## "GSM134702"       30.72       33.52       35.76       35.64       32.89
## "GSM134703"       26.84       29.25       41.30       40.83       30.25
## "GSM134704"       32.51       28.33       43.52       44.64       30.94
## "GSM134705"       22.59       33.21       34.53       33.85       32.86
## "GSM134706"        0.00       26.53       41.65       41.95       30.53
## "GSM134707"       26.53        0.00       48.73       49.45       23.04
```

```
## "GSM134708"       41.65       48.73        0.00       27.68       45.40
## "GSM134709"       41.95       49.45       27.68        0.00       44.64
## "GSM134710"       30.53       23.04       45.40       44.64        0.00
```

Aprovechando las distancias obtenidas en el paso anterior se realiza un mapa de calor donde los colores mas rojos indican menor distancia, mientras los colores mas azules indican mayor distancia. Como se puede ver, hay algunas observaciones en la zona intermedia que tienen colores mas azules con la mayoría de las otras observaciones, también se observan zonas de mayor concentración roja en las últimas muestras.

```
fviz_dist(dist.obj = mat_dist, lab_size = 6) + theme(legend.position = "none")
```



Una vez analizados los datos y dando por hecho que están correctamente preparados se va a proceder a llevar a cabo la aplicación de los algoritmos de agrupamiento elegidos.

Aplicación de los algoritmos

El proceso a seguir para aplicar cada algoritmo sobre los datos es el siguiente.
En primer lugar se utilizará la función `optCluster`, indicando un intervalo de valores para k, con lo que se obtendrá una lista con los valores de k ordenados por su eficiencia, de mayor a menor.
Utilizando los valores obtenidos anteriormente se realizará una representación de cada algoritmo con sus dos mejores opciones en cuanto a valor de k. Para mantener una estabilidad en los procesos se indicará una semilla fija (123) para cada vez que se utilice la función `optCluster`.
Para determinar la eficiencia se utilizarán todos los medidores de internalidad, estabilidad y variación biológica que plantea dicha función.
Con el fin de que el resultado sea significativo se realizarán 1000 repeticiones de cada proceso.

Hierarchical

Obtención de el número óptimo de grupos

Para obtener los grupos óptimos para el algoritmo de hierarchical clustering es necesario indicar que el método de agrupamiento es completo.

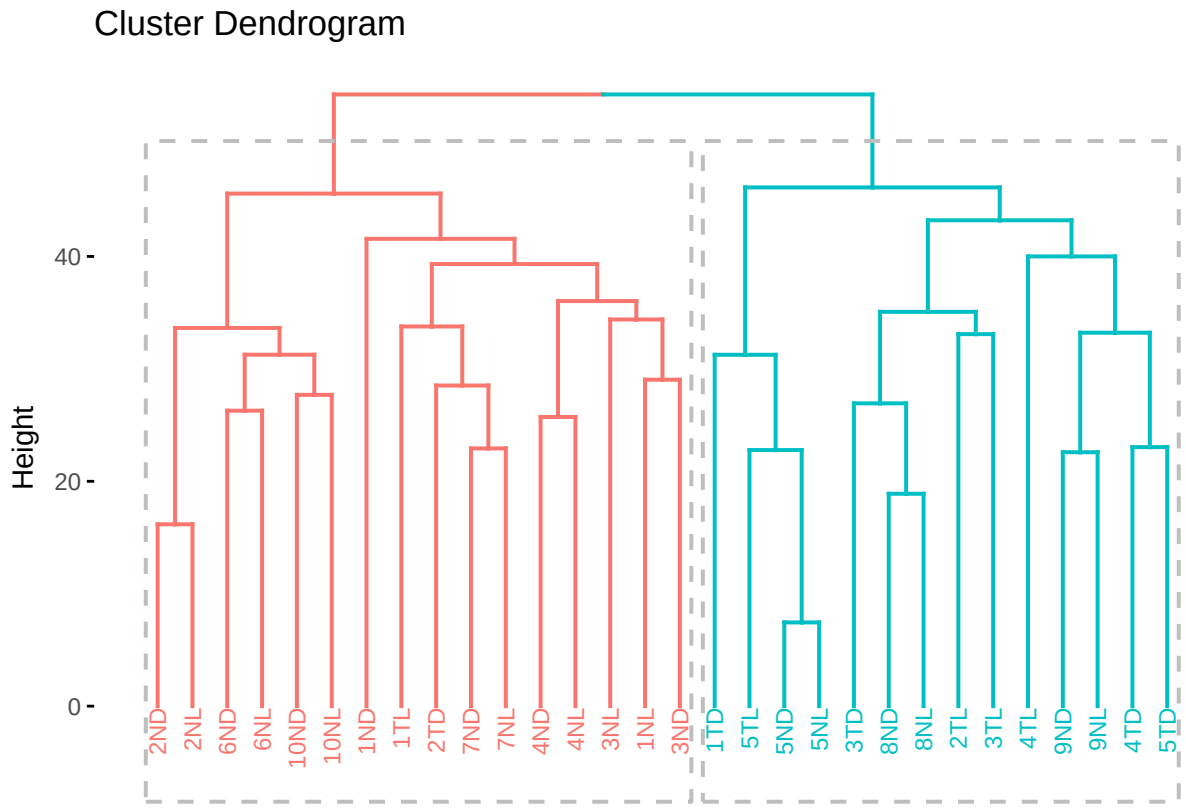
```
library(optCluster)
optresult <- optCluster(data[, ], 2:5, clMethods = c("hierarchical"), validation = "all",
    annotation = "hgu133plus2.db", seed = 123, maxIter = 1000, hierMethod = "complete")
summary(optresult)
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                   2       3       4       5
##
## hierarchical APN            0.1417  0.2613  0.3927  0.4254
##              AD            15.8531 15.4282 15.0461 14.4532
##              ADM            2.0677  3.9373  4.5007  4.5357
##              FOM            2.0421  1.9810  1.9223  1.8586
##              Connectivity  26.1869 90.5337 91.2266 91.8960
##              Dunn           0.2168  0.2354  0.2382  0.2529
##              Silhouette     0.3040  0.1200  0.1235  0.1037
##              BHI            0.3514  0.3313  0.3556  0.3374
##              BSI            0.7196  0.5716  0.4291  0.3388
##
## Optimal Scores:
##
##                Score       Method Clusters
## APN           0.1417 hierarchical        2
## AD           14.4532 hierarchical        5
## ADM           2.0677 hierarchical        2
## FOM           1.8586 hierarchical        5
## Connectivity 26.1869 hierarchical        2
## Dunn          0.2529 hierarchical        5
## Silhouette    0.3040 hierarchical        2
## BHI           0.3556 hierarchical        4
## BSI           0.7196 hierarchical        2
##
## The overall optimal clustering method and number of clusters is:
##      hierarchical-2
##
## The optimal list is:
##         hierarchical-2 hierarchical-4 hierarchical-5 hierarchical-3
##
##   Algorithm:   CE
##   Distance:    Spearman
##   Score:       2.079421
##   Iterations:  7
```

Para este algoritmo el número óptimo de grupos es 2, mientras el segundo mejor es 4.
En estas representaciones se utiliza un dendrograma, la separación por grupos se lleva a cabo por colores en las ramas de dicho dendrograma.

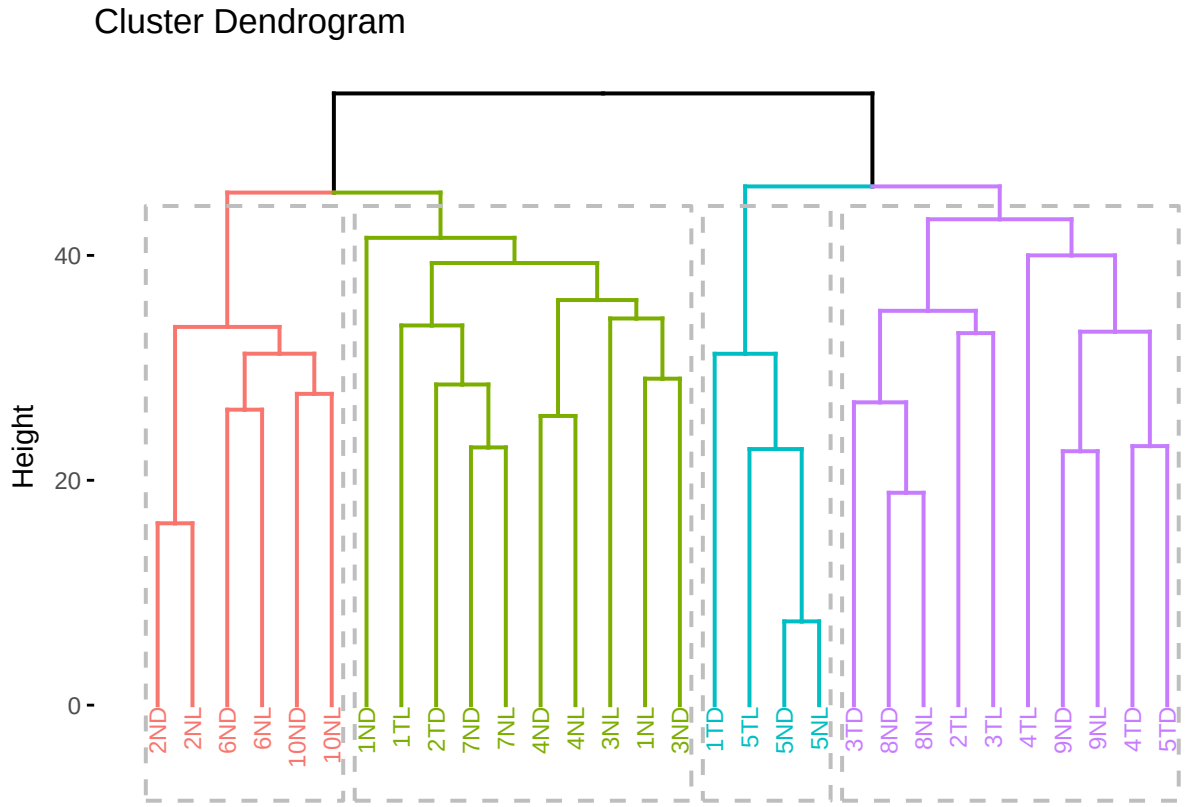
k=2

```
data.dis <- dist(as.data.frame(data.t), method = "euclidean")
hc <- hclust(data.dis, method = "complete")
fviz_dend(x = hc, k = 2, cex = 0.6, rect = TRUE, labels_track_height = 8)
```



k=4

```
data.dis <- dist(as.data.frame(data.t), method = "euclidean")
hc <- hclust(data.dis, method = "complete")
fviz_dend(x = hc, k = 4, cex = 0.6, rect = TRUE, labels_track_height = 8)
```



AGNES

Obtención de el número óptimo de grupos

```
optresult <- optCluster(data[, ], 2:5, clMethods = c("agnes"), validation = "all",
    annotation = "hgu133plus2.db", seed = 123, maxIter = 1000, hierMethod = "complete")
summary(optresult)
```

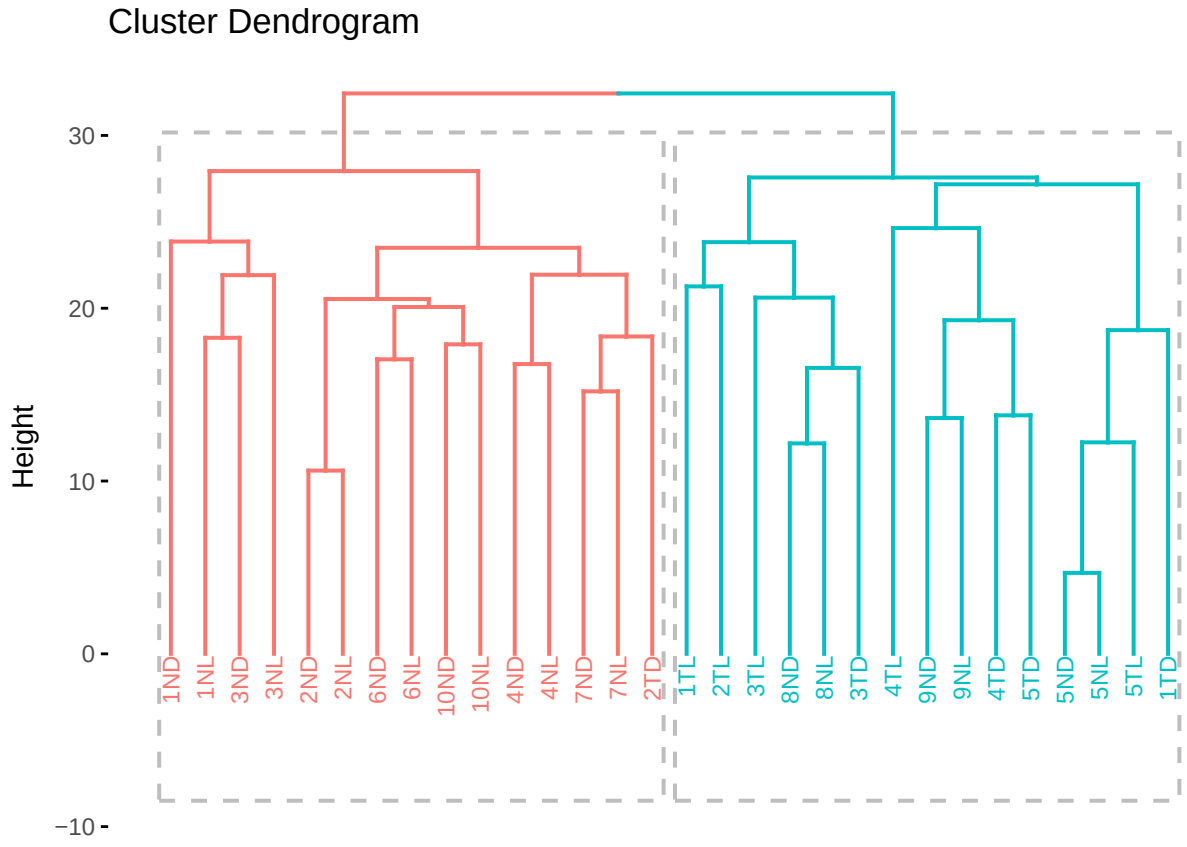
```
##
## Clustering Methods:
##  agnes
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                             2       3       4       5
##
## agnes APN            0.1417  0.2613  0.3927  0.4254
##       AD            15.8531 15.4282 15.0461 14.4532
##       ADM            2.0677  3.9373  4.5007  4.5357
##       FOM            2.0421  1.9810  1.9223  1.8586
##       Connectivity  26.1869 90.5337 91.2266 91.8960
##       Dunn           0.2168  0.2354  0.2382  0.2529
```

```
##       Silhouette     0.3040  0.1200  0.1235  0.1037
##       BHI            0.3514  0.3313  0.3556  0.3374
##       BSI            0.7196  0.5716  0.4291  0.3388
##
## Optimal Scores:
##
##                Score Method Clusters
## APN           0.1417  agnes        2
## AD           14.4532  agnes        5
## ADM           2.0677  agnes        2
## FOM           1.8586  agnes        5
## Connectivity 26.1869  agnes        2
## Dunn          0.2529  agnes        5
## Silhouette    0.3040  agnes        2
## BHI           0.3556  agnes        4
## BSI           0.7196  agnes        2
##
## The overall optimal clustering method and number of clusters is:
##      agnes-2
##
## The optimal list is:
##         agnes-2 agnes-4 agnes-5 agnes-3
##
##   Algorithm:   CE
##   Distance:    Spearman
##   Score:       2.079421
##   Iterations:  7
```

Para este algoritmo se obtienen valores de 2 y 4, en ese orden como valores óptimos. La función se configura de forma similar al algoritmo hierarchical clustering y su representación es en forma de dendrograma también.

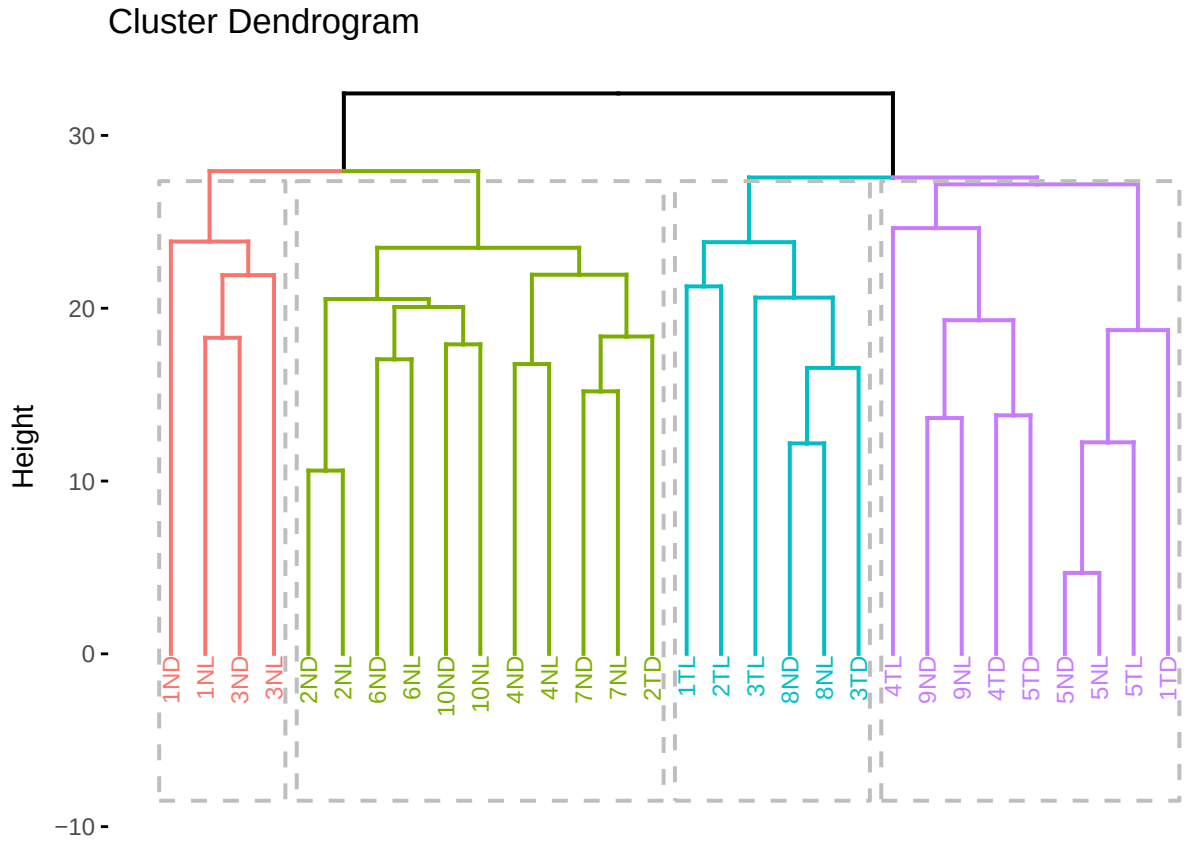
k=2

```
agnes1 <- agnes(as.data.frame(data.t), diss = FALSE, metric = "euclidean", stand = TRUE,
    method = "complete")
fviz_dend(x = agnes1, k = 2, cex = 0.6, rect = TRUE, labels_track_height = 8)
```



k=4

```
agnes1 <- agnes(as.data.frame(data.t), diss = FALSE, metric = "euclidean", stand = TRUE,
    method = "complete")
fviz_dend(x = agnes1, k = 4, cex = 0.6, rect = TRUE, labels_track_height = 8)
```



DIANA

Obtención de el número óptimo de grupos

```
optresult <- optCluster(data[, ], 2:5, clMethods = c("diana"), validation = "all",
    annotation = "hgu133plus2.db", seed = 123, maxIter = 1000, hierMethod = "complete")
summary(optresult)
```

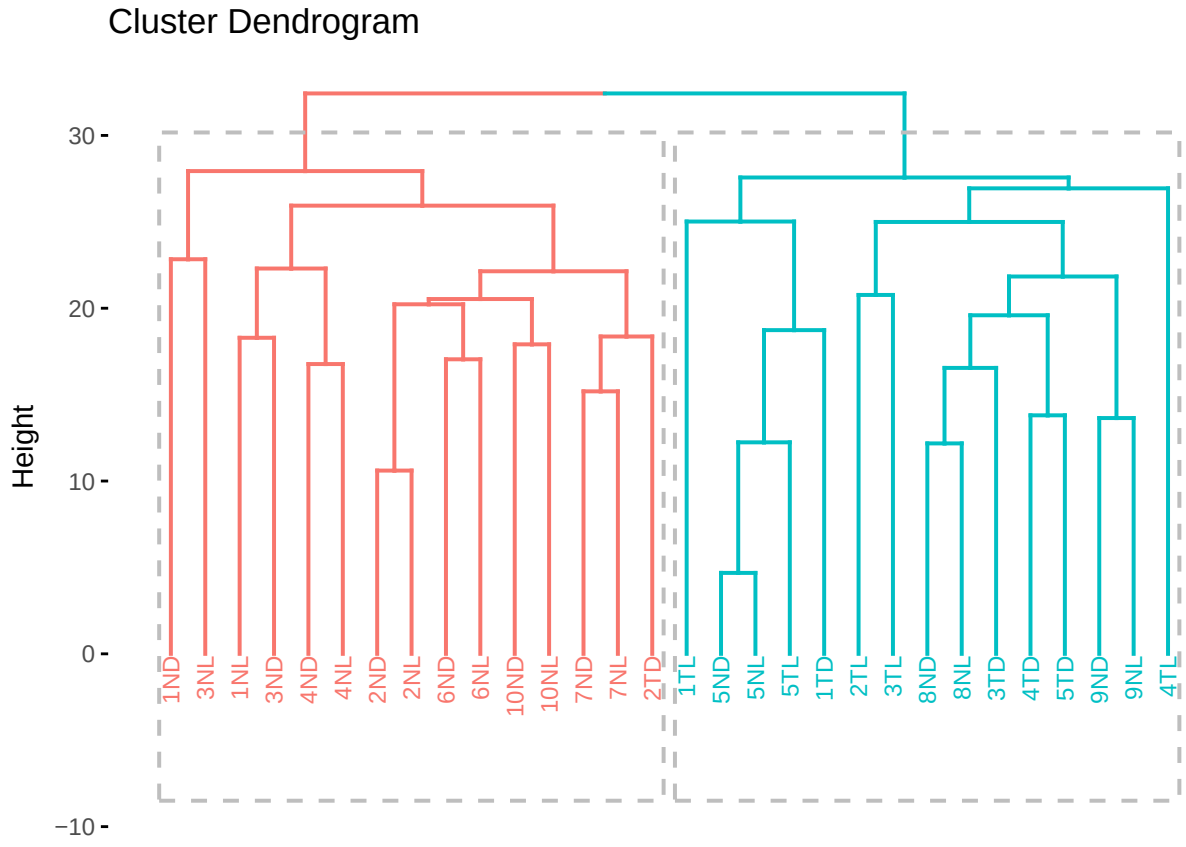
```
##
## Clustering Methods:
##  diana
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                             2        3        4        5
##
## diana APN             0.0126   0.0473   0.0577   0.0975
##       AD             14.9668  14.1683  13.8255  13.5816
##       ADM             0.2382   0.6111   0.8002   1.4599
##       FOM             2.0070   1.9222   1.8878   1.8532
##       Connectivity   36.8310  88.9401  93.6119 102.6702
##       Dunn            0.2525   0.2589   0.2880   0.2699
```

```
##       Silhouette      0.2502   0.1530   0.1515   0.1282
##       BHI             0.2924   0.2889   0.3350   0.3296
##       BSI             0.6682   0.4343   0.4161   0.3929
##
## Optimal Scores:
##
##                Score Method Clusters
## APN           0.0126  diana        2
## AD           13.5816  diana        5
## ADM           0.2382  diana        2
## FOM           1.8532  diana        5
## Connectivity 36.8310  diana        2
## Dunn          0.2880  diana        4
## Silhouette    0.2502  diana        2
## BHI           0.3350  diana        4
## BSI           0.6682  diana        2
##
## The overall optimal clustering method and number of clusters is:
##      diana-2
##
## The optimal list is:
##         diana-2 diana-4 diana-3 diana-5
##
##   Algorithm:   CE
##   Distance:    Spearman
##   Score:       2.34971
##   Iterations:  7
```

El algoritmo DIANA obtiene como óptimos los valores 2 y 4, su representación es en dendrograma también.

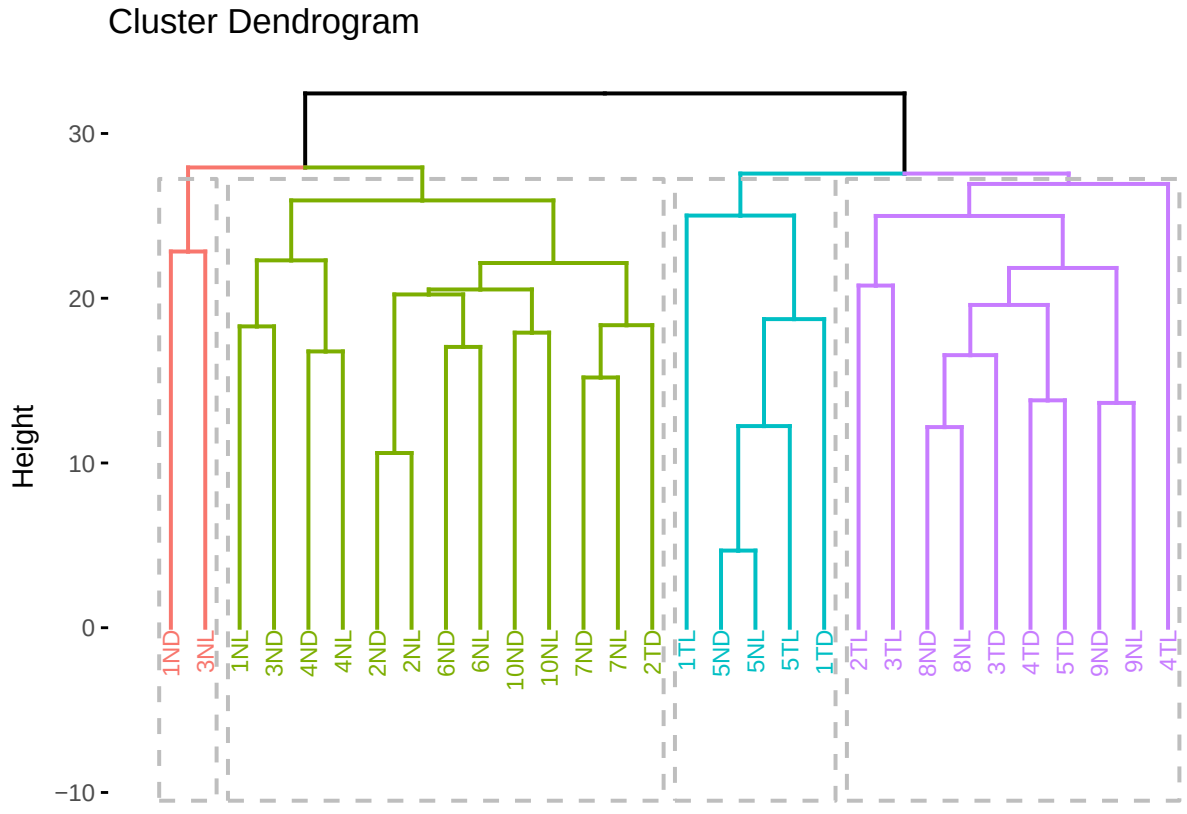
k=2

```
diana1 <- diana(as.data.frame(data.t), diss = FALSE, stand = TRUE, metric = "euclidean")
fviz_dend(x = diana1, k = 2, cex = 0.6, rect = TRUE, labels_track_height = 8)
```



k=4

```
diana2 <- diana(as.data.frame(data.t), diss = FALSE, stand = TRUE, metric = "euclidean")
fviz_dend(x = diana2, k = 4, cex = 0.6, rect = TRUE, labels_track_height = 10)
```



K-means

Obtención de el número óptimo de grupos

```
optresult <- optCluster(data[, ], 2:5, clMethods = c("kmeans"), validation = "all",
    annotation = "hgu133plus2.db", seed = 123, maxIter = 1000)
summary(optresult)
```

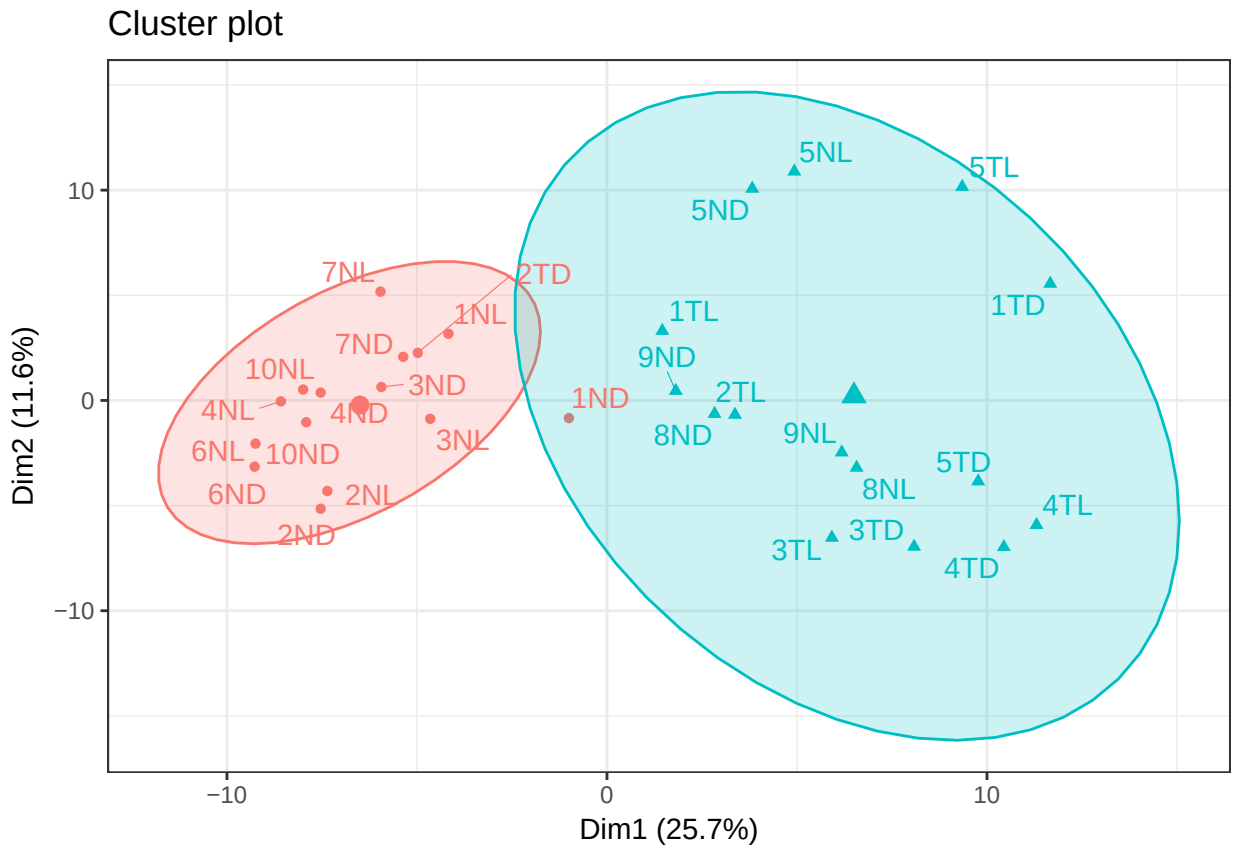
```
##
## Clustering Methods:
##  kmeans
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                              2        3        4        5
##
## kmeans APN             0.0282   0.0467   0.0819   0.2775
##        AD              14.9935  14.1376  13.4948  13.5666
##        ADM              0.4979   0.6971   1.2267   3.0983
##        FOM              2.0234   1.9301   1.8281   1.7934
##        Connectivity    50.4683  64.4766 103.2385 137.1218
##        Dunn             0.2614   0.2616   0.2340   0.2673
```

```
##        Silhouette      0.2441   0.1887   0.1502   0.1305
##        BHI             0.2925   0.2987   0.3000   0.3198
##        BSI             0.6482   0.5222   0.3497   0.2897
##
## Optimal Scores:
##
##                Score Method Clusters
## APN           0.0282 kmeans        2
## AD           13.4948 kmeans        4
## ADM           0.4979 kmeans        2
## FOM           1.7934 kmeans        5
## Connectivity 50.4683 kmeans        2
## Dunn          0.2673 kmeans        5
## Silhouette    0.2441 kmeans        2
## BHI           0.3198 kmeans        5
## BSI           0.6482 kmeans        2
##
## The overall optimal clustering method and number of clusters is:
##      kmeans-2
##
## The optimal list is:
##         kmeans-2 kmeans-3 kmeans-4 kmeans-5
##
##   Algorithm:   CE
##   Distance:    Spearman
##   Score:       2.58949
##   Iterations:  7
```

El resultado de la función para k-means es que el número óptimo de grupos es 2, mientras el segundo mejor es 3.
Las representaciones de estos dos algoritmos se presentan a continuación, estas son en forma de gráfico de dispersión sobre las componentes principales. Los puntos se agrupan por colores y mediante una elipse que recoge los grupos.

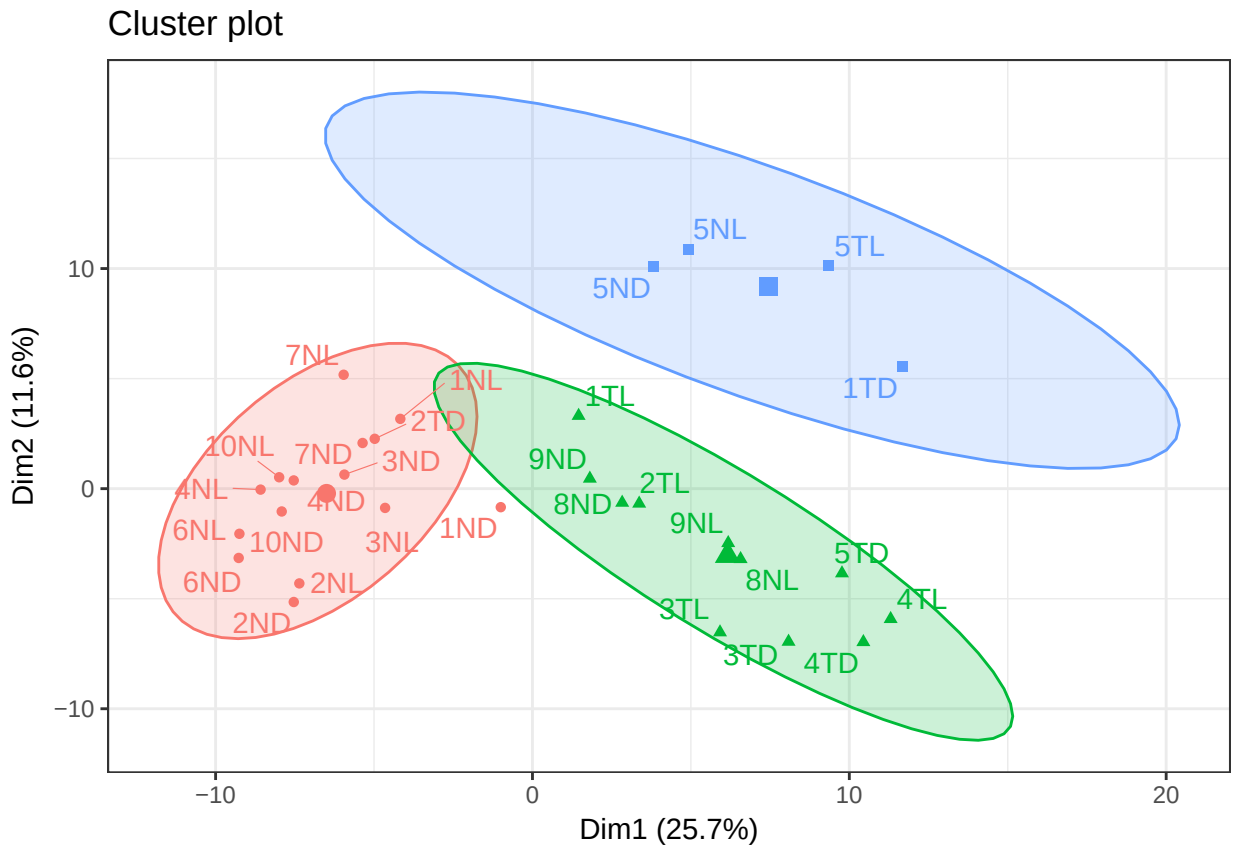
k=2

```
model1 <- kmeans(as.data.frame(data.t), 2, nstart = 1000)

fviz_cluster(object = model1, data = as.data.frame(data.t), ellipse.type = "t",
    repel = TRUE) + theme_bw() + theme(legend.position = "none")
```



k=3

```
model2 <- kmeans(as.data.frame(data.t), 3, nstart = 1000)

fviz_cluster(object = model2, data = as.data.frame(data.t), ellipse.type = "t",
    repel = TRUE) + theme_bw() + theme(legend.position = "none")
```



CLARA

Obtención de el número óptimo de grupos

```
optresult <- optCluster(data[, ], 2:5, clMethods = c("clara"), validation = "all",
    annotation = "hgu133plus2.db", seed = 123, maxIter = 1000)
summary(optresult)
```

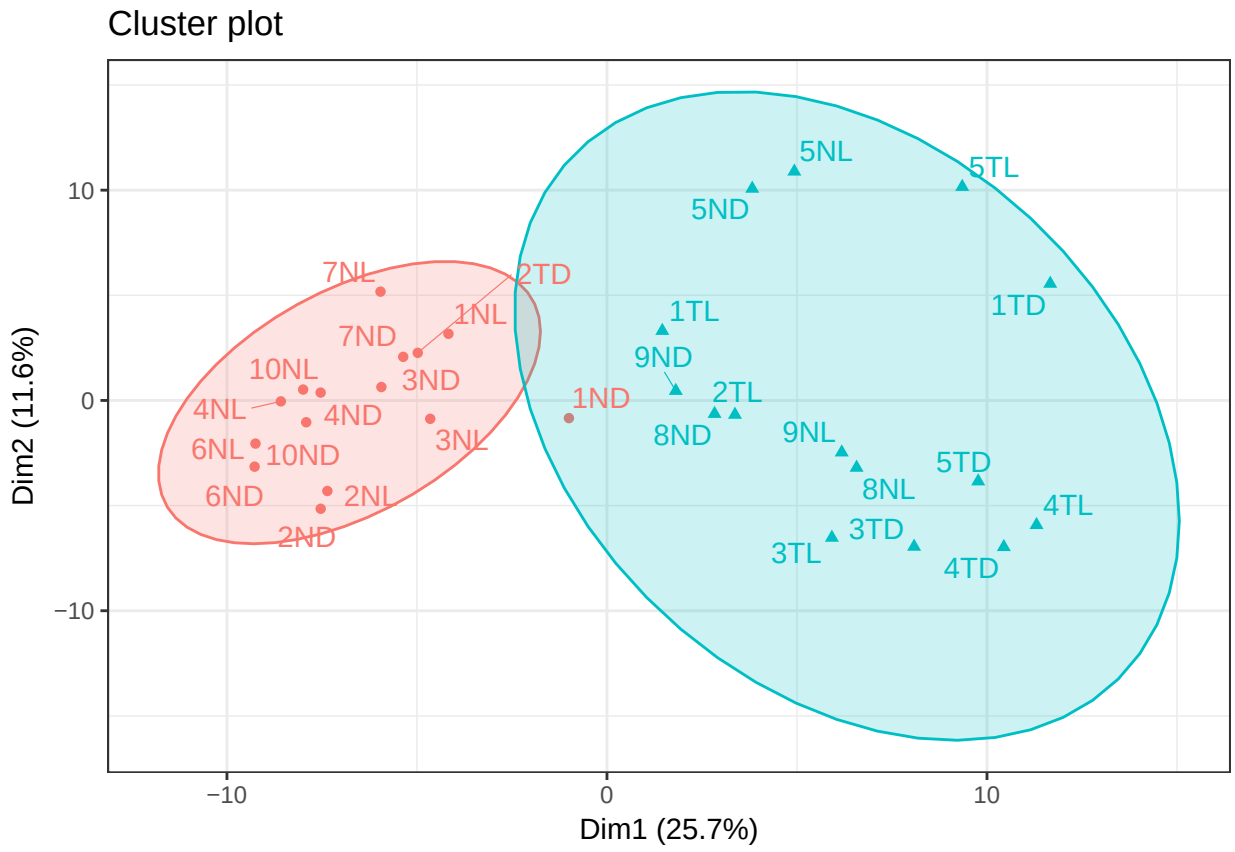
```
##
## Clustering Methods:
##  clara
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                              2        3        4        5
##
## clara APN             0.0976   0.1244   0.1567   0.2666
##       AD             15.2201  14.5017  14.0634  14.0022
##       ADM             1.3460   1.6016   1.9604   3.2442
##       FOM             2.0192   1.9459   1.9120   1.8651
##       Connectivity   49.8504 108.1881 146.6877 157.2016
##       Dunn            0.2614   0.2221   0.2313   0.2454
```

```
##       Silhouette      0.2469   0.1558   0.1089   0.0947
##       BHI             0.2947   0.2989   0.2957   0.2927
##       BSI             0.6355   0.4462   0.3524   0.2987
##
## Optimal Scores:
##
##                Score Method Clusters
## APN           0.0976  clara        2
## AD           14.0022  clara        5
## ADM           1.3460  clara        2
## FOM           1.8651  clara        5
## Connectivity 49.8504  clara        2
## Dunn          0.2614  clara        2
## Silhouette    0.2469  clara        2
## BHI           0.2989  clara        3
## BSI           0.6355  clara        2
##
## The overall optimal clustering method and number of clusters is:
##      clara-2
##
## The optimal list is:
##         clara-2 clara-3 clara-4 clara-5
##
##   Algorithm:   CE
##   Distance:    Spearman
##   Score:       1.951488
##   Iterations:  7
```

La mejores eficiencias de este algoritmos se consiguen a valores de 2 y 3 grupos. La representación es similar a la realizada en k-means.

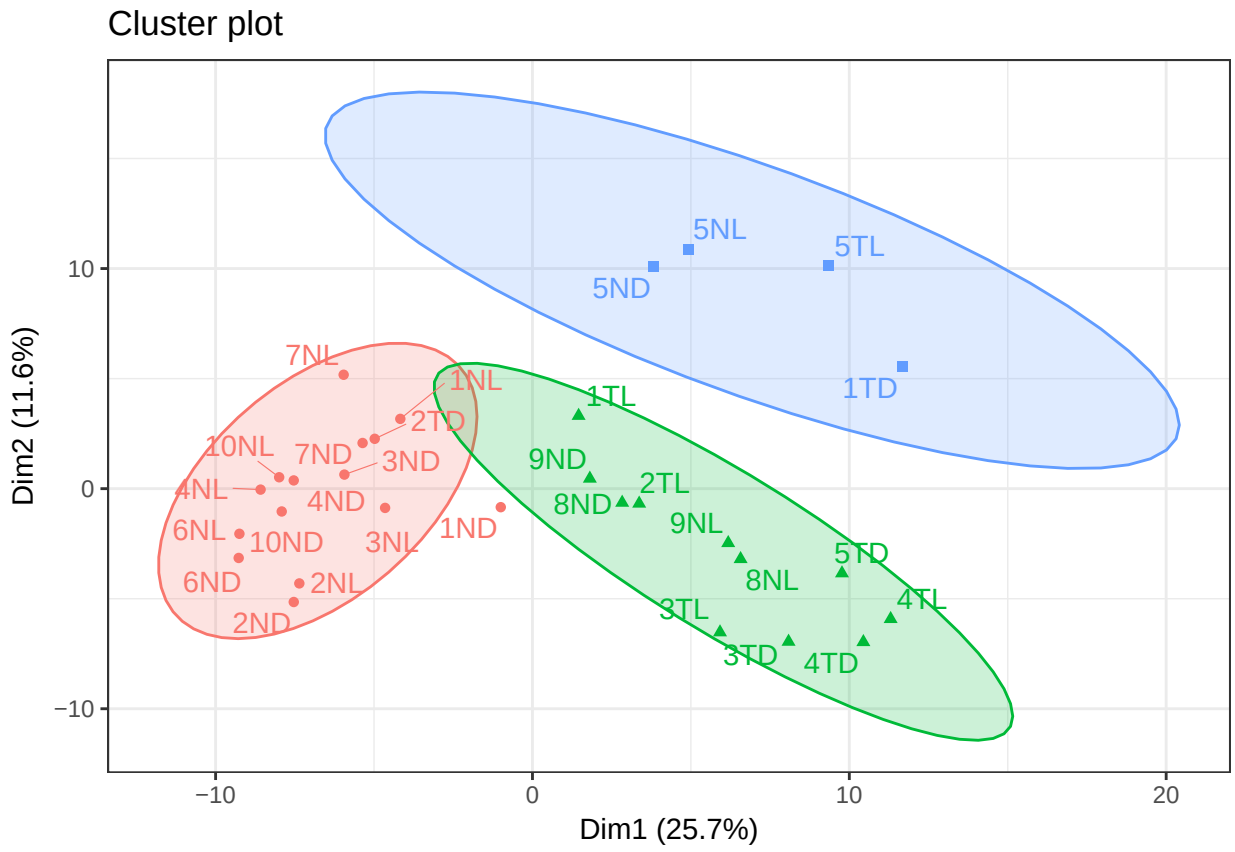
k=2

```
clara1 <- clara(as.data.frame(data.t), k = 2, metric = "euclidean", samples = 1000,
    pamLike = TRUE, stand = TRUE)
fviz_cluster(object = clara1, data = as.data.frame(data.t), ellipse.type = "t",
    repel = TRUE) + theme_bw() + theme(legend.position = "none")
```



k=3

```
clara2 <- clara(as.data.frame(data.t), k = 3, metric = "euclidean", samples = 1000,
    pamLike = TRUE, stand = TRUE)
fviz_cluster(object = clara2, data = as.data.frame(data.t), ellipse.type = "t",
    repel = TRUE) + theme_bw() + theme(legend.position = "none")
```



PAM

Obtención del número óptimo de grupos

```
optresult <- optCluster(data[, ], 2:5, clMethods = c("pam"), validation = "all",
    annotation = "hgu133plus2.db", seed = 123, maxIter = 1000)
summary(optresult)
```

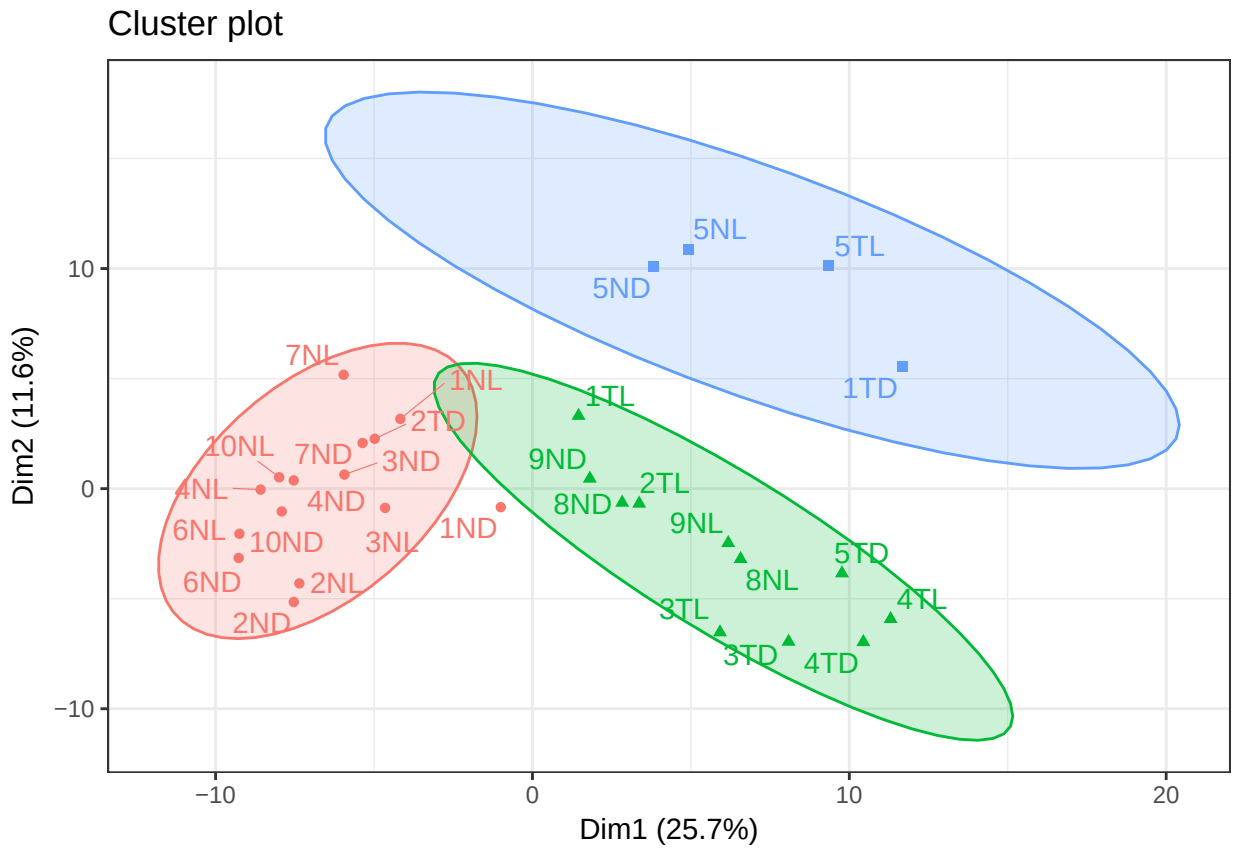
```
##
## Clustering Methods:
##  pam
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                            2        3        4        5
##
## pam APN              0.3060   0.0587   0.0751   0.2126
##     AD              16.3082  14.3226  13.6458  13.6166
##     ADM              5.0172   0.6924   0.9642   2.3409
##     FOM              2.1775   1.9375   1.8650   1.8423
##     Connectivity    75.9627 119.0417 133.8817 146.6389
##     Dunn             0.1592   0.2024   0.2046   0.2203
```

```
##     Silhouette      0.2093   0.1401   0.1293   0.1255
##     BHI             0.2887   0.2915   0.2870   0.2932
##     BSI             0.5880   0.4612   0.3466   0.2809
##
## Optimal Scores:
##
##                Score Method Clusters
## APN           0.0587    pam        3
## AD           13.6166    pam        5
## ADM           0.6924    pam        3
## FOM           1.8423    pam        5
## Connectivity 75.9627    pam        2
## Dunn          0.2203    pam        5
## Silhouette    0.2093    pam        2
## BHI           0.2932    pam        5
## BSI           0.5880    pam        2
##
## The overall optimal clustering method and number of clusters is:
##      pam-3
##
## The optimal list is:
##         pam-3 pam-5 pam-4 pam-2
##
##   Algorithm:   CE
##   Distance:    Spearman
##   Score:       2.056702
##   Iterations:  7
```

Para el algoritmo PAM los valores óptimos de grupos son 3 y 5. Su representación es en forma de gráfico de dispersión.

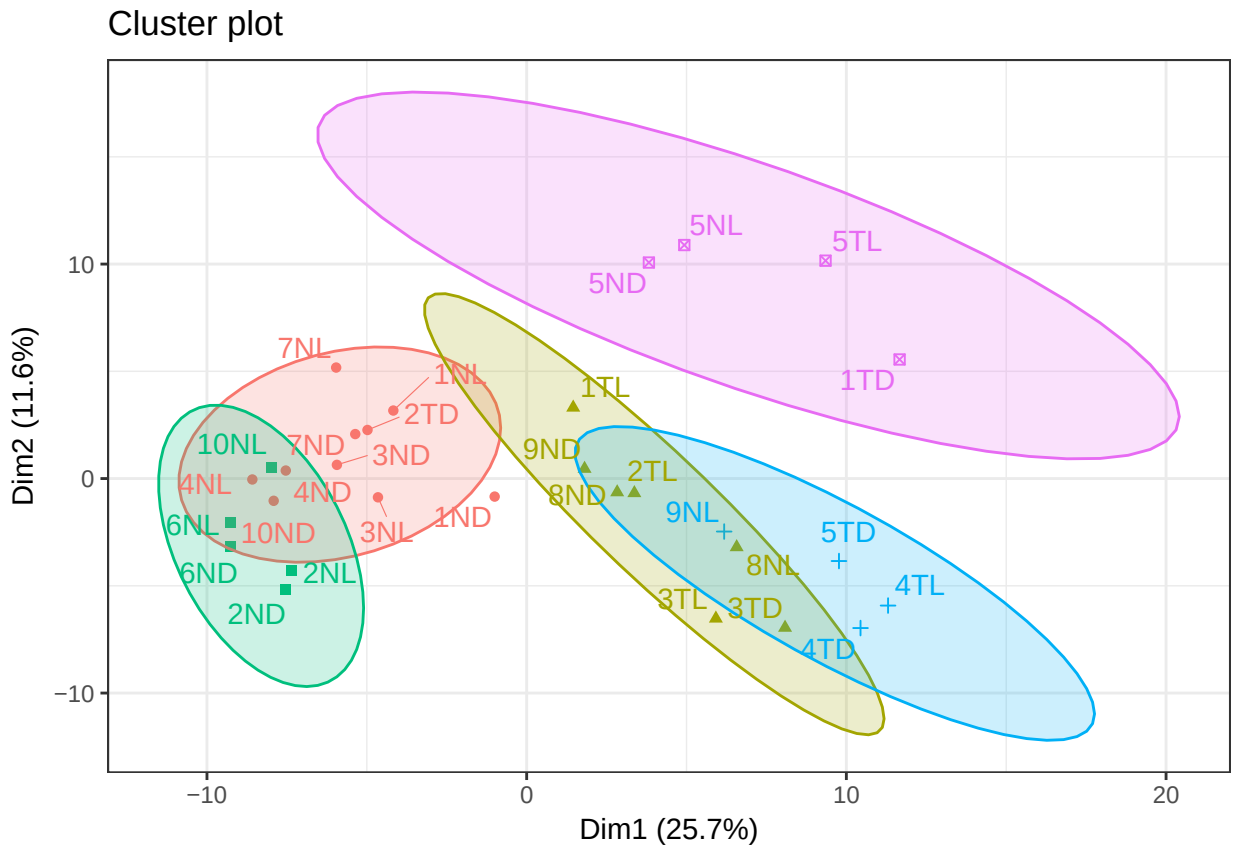
k=3

```
pam1 <- pam(as.data.frame(data.t), 3, diss = FALSE, metric = "euclidean")
fviz_cluster(object = pam1, data = as.data.frame(data.t), ellipse.type = "t",
    repel = TRUE) + theme_bw() + theme(legend.position = "none")
```



k=5

```
pam2 <- pam(as.data.frame(data.t), 5, diss = FALSE, metric = "euclidean")
fviz_cluster(object = pam2, data = as.data.frame(data.t), ellipse.type = "t",
    repel = TRUE) + theme_bw() + theme(legend.position = "none")
```



Model-based

Obtención de el número óptimo de grupos

```
optresult <- optCluster(data[, ], 2:5, clMethods = c("model"), validation = "all",
    annotation = "hgu133plus2.db", seed = 123, maxIter = 1000)
summary(optresult)
```

```
##
## Clustering Methods:
##  model
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                             2        3        4        5
##
## model APN             0.1747   0.2489   0.3190   0.3820
##       AD             15.8937  15.1637  14.9657  14.8716
##       ADM             2.1353   2.8552   3.7156   4.4088
##       FOM             2.0919   1.9953   1.9827   1.9594
##       Connectivity   72.6036 103.7829 105.3833 150.8587
##       Dunn            0.1553   0.1767   0.2343   0.2051
```

```
##       Silhouette      0.1664   0.1406   0.1237   0.0979
##       BHI             0.2801   0.2981   0.3337   0.3179
##       BSI             0.6589   0.4603   0.3791   0.3336
##
## Optimal Scores:
##
##                Score Method Clusters
## APN           0.1747  model        2
## AD           14.8716  model        5
## ADM           2.1353  model        2
## FOM           1.9594  model        5
## Connectivity 72.6036  model        2
## Dunn          0.2343  model        4
## Silhouette    0.1664  model        2
## BHI           0.3337  model        4
## BSI           0.6589  model        2
##
## The overall optimal clustering method and number of clusters is:
##      model-2
##
## The optimal list is:
##         model-2 model-4 model-3 model-5
##
##   Algorithm:   CE
##   Distance:    Spearman
##   Score:       2.644534
##   Iterations:  7
```

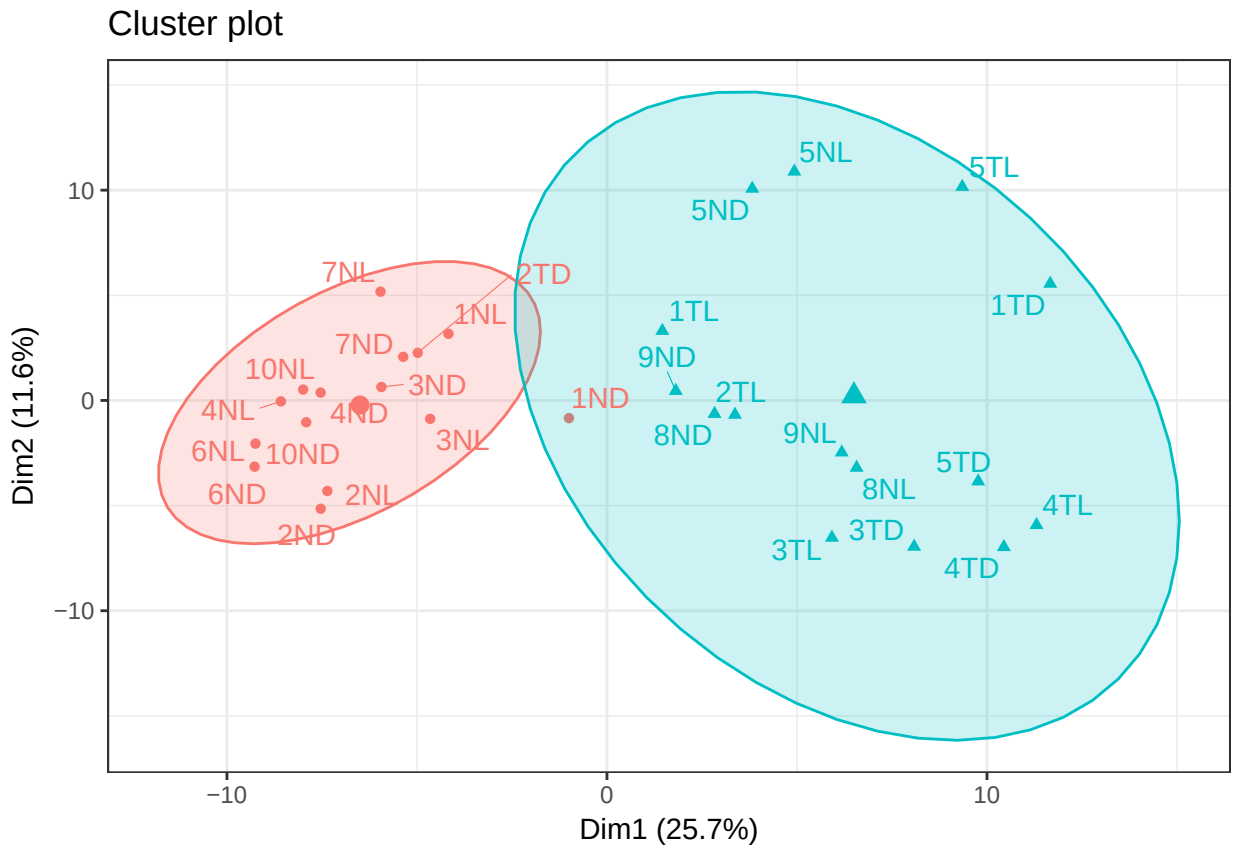
El algoritmo model-based tiene varias peculiaridades, ya que tiene diversos modelos a seguir, pero en este caso la función sólo elige los mejores valores de númnero de grupos, en este caso 2 y 4. En la representación la función mclust elige para cada k el mejor modelo, por lo que no habrá problemas con estas peculiaridades.

k=2

```
library(mclust)
model1 <- Mclust(data.t, G = 2)
model1
```

```
## 'Mclust' model object:
##  best model: spherical, equal volume (EII) with 2 components
```

```
fviz_cluster(object = model1, repel = TRUE, ellipse.type = "t", pallete = "jco") +
    theme_bw() + theme(legend.position = "none")
```



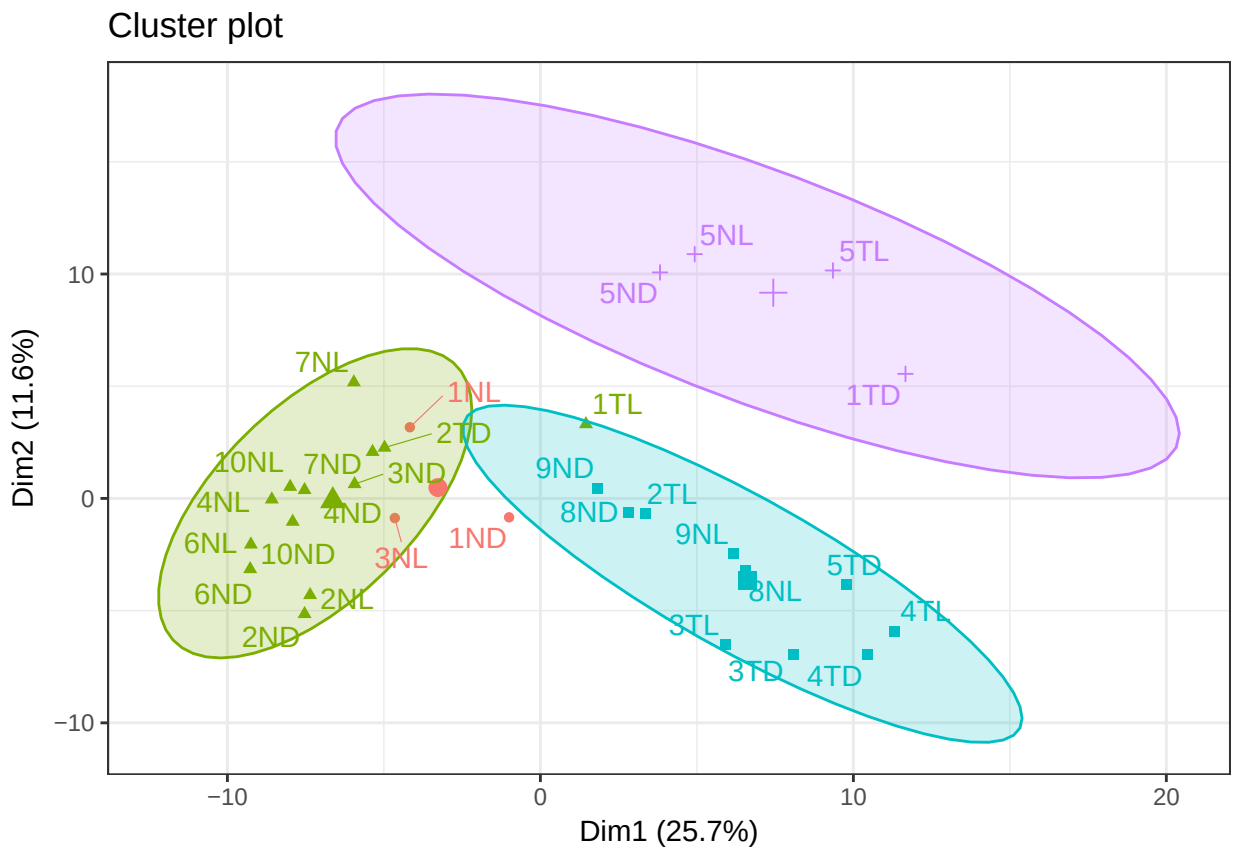
k=4

```
model1 <- Mclust(data.t, G = 4)
model1
```

```
## 'Mclust' model object:
##  best model: spherical, varying volume (VII) with 4 components
```

```
fviz_cluster(object = model1, repel = TRUE, ellipse.type = "t", pallete = "jco") +
    theme_bw() + theme(legend.position = "none")
```

```
## Too few points to calculate an ellipse
```



Comparativa de todos los algoritmos

Finalmente se utiliza de nuevo la función optCluster para comparar todos los algoritmos entre si. Para ello se indica en las opciones de la función los algoritmos elegidos, además como se han obtenido los grupos óptimos para cada uno de ellos se indica un valor de k entre el mínimo y el máximo elegido de todos ellos.

```
optresult <- optCluster(data[, ], 2:5, clMethods = c("model", "kmeans", "pam",
    "agnes", "diana", "hierarchical", "clara"), validation = "all", annotation = "hgu133plus2.db",
    seed = 123, maxIter = 1000, hierMethod = "complete")
summary(optresult)
```

```
##
## Clustering Methods:
##  model kmeans pam agnes diana hierarchical clara
##
## Cluster sizes:
##  2 3 4 5
##
## Validation Measures:
##                                   2        3        4        5
##
## model        APN             0.1747   0.2489   0.3190   0.3820
##              AD             15.8937  15.1637  14.9657  14.8716
##              ADM             2.1353   2.8552   3.7156   4.4088
##              FOM             2.0919   1.9953   1.9827   1.9594
##              Connectivity   72.6036 103.7829 105.3833 150.8587
```

```
##              Dunn            0.1553   0.1767   0.2343   0.2051
##              Silhouette      0.1664   0.1406   0.1237   0.0979
##              BHI             0.2801   0.2981   0.3337   0.3179
##              BSI             0.6589   0.4603   0.3791   0.3336
## kmeans       APN             0.0294   0.2513   0.1033   0.0914
##              AD             14.9964  14.8622  13.5890  13.0879
##              ADM             0.5145   3.8155   1.5546   1.0384
##              FOM             2.0256   1.9101   1.8250   1.7743
##              Connectivity   50.4683 101.7373  98.2528 107.9480
##              Dunn            0.2614   0.1927   0.2419   0.2673
##              Silhouette      0.2441   0.1276   0.1601   0.1529
##              BHI             0.2925   0.2904   0.3132   0.3277
##              BSI             0.6471   0.4463   0.3518   0.3089
## pam          APN             0.3060   0.0587   0.0751   0.2126
##              AD             16.3082  14.3226  13.6458  13.6166
##              ADM             5.0172   0.6924   0.9642   2.3409
##              FOM             2.1775   1.9375   1.8650   1.8423
##              Connectivity   75.9627 119.0417 133.8817 146.6389
##              Dunn            0.1592   0.2024   0.2046   0.2203
##              Silhouette      0.2093   0.1401   0.1293   0.1255
##              BHI             0.2887   0.2915   0.2870   0.2932
##              BSI             0.5880   0.4612   0.3466   0.2809
## agnes        APN             0.1417   0.2613   0.3927   0.4254
##              AD             15.8531  15.4282  15.0461  14.4532
##              ADM             2.0677   3.9373   4.5007   4.5357
##              FOM             2.0421   1.9810   1.9223   1.8586
##              Connectivity   26.1869  90.5337  91.2266  91.8960
##              Dunn            0.2168   0.2354   0.2382   0.2529
##              Silhouette      0.3040   0.1200   0.1235   0.1037
##              BHI             0.3514   0.3313   0.3556   0.3374
##              BSI             0.7196   0.5716   0.4291   0.3388
## diana        APN             0.0126   0.0473   0.0577   0.0975
##              AD             14.9668  14.1683  13.8255  13.5816
##              ADM             0.2382   0.6111   0.8002   1.4599
##              FOM             2.0070   1.9222   1.8878   1.8532
##              Connectivity   36.8310  88.9401  93.6119 102.6702
##              Dunn            0.2525   0.2589   0.2880   0.2699
##              Silhouette      0.2502   0.1530   0.1515   0.1282
##              BHI             0.2924   0.2889   0.3350   0.3296
##              BSI             0.6682   0.4343   0.4161   0.3929
## hierarchical APN             0.1417   0.2613   0.3927   0.4254
##              AD             15.8531  15.4282  15.0461  14.4532
##              ADM             2.0677   3.9373   4.5007   4.5357
##              FOM             2.0421   1.9810   1.9223   1.8586
##              Connectivity   26.1869  90.5337  91.2266  91.8960
##              Dunn            0.2168   0.2354   0.2382   0.2529
##              Silhouette      0.3040   0.1200   0.1235   0.1037
##              BHI             0.3514   0.3313   0.3556   0.3374
##              BSI             0.7196   0.5716   0.4291   0.3388
## clara        APN             0.0976   0.1244   0.1567   0.2666
##              AD             15.2201  14.5017  14.0634  14.0022
##              ADM             1.3460   1.6016   1.9604   3.2442
##              FOM             2.0192   1.9459   1.9120   1.8651
##              Connectivity   49.8504 108.1881 146.6877 157.2016
```

```
##              Dunn            0.2614   0.2221   0.2313   0.2454
##              Silhouette      0.2469   0.1558   0.1089   0.0947
##              BHI             0.2947   0.2989   0.2957   0.2927
##              BSI             0.6355   0.4462   0.3524   0.2987
##
## Optimal Scores:
##
##                 Score       Method Clusters
## APN            0.0126        diana        2
## AD            13.0879       kmeans        5
## ADM            0.2382        diana        2
## FOM            1.7743       kmeans        5
## Connectivity 26.1869        agnes        2
## Dunn           0.2880        diana        4
## Silhouette     0.3040        agnes        2
## BHI            0.3556 hierarchical        4
## BSI            0.7196 hierarchical        2
##
## The overall optimal clustering method and number of clusters is:
##      diana-2
##
## The optimal list is:
##         diana-2 agnes-2 diana-4 hierarchical-2 kmeans-2 clara-2 diana-3 diana-5
##         kmeans-5 pam-3 kmeans-4 clara-3 pam-4 agnes-3 clara-4 hierarchical-3 agnes-4
##         hierarchical-4 agnes-5 pam-5 hierarchical-5 kmeans-3 model-4 clara-5
##         model-2 model-5 pam-2
##
##   Algorithm:   CE
##   Distance:    Spearman
##   Score:       63.1634
##   Iterations:  187
```

El resultado de la función deja claro que el mejor algoritmo es DIANA, ya que el algoritmo con k=2 es el más eficiente y el que utiliza k=4 es el tercero mejor, por lo cual queda definido como el mejor algoritmo para agrupar los datos de expresión génica de este tipo. También sería valorable utilizar el algoritmo AGNES con k=2 ya que se encuentra como el segundo mas eficiente.

En cuanto a los demás algoritmos, los primeros puestos de la lista de eficiencia están ocupados por la agrupación con k=2 de la mayoría de los otros algoritmos.