ANÁLISIS DE DATOS ÓMICOS - PEC 2

Carlos Fernández Medina

14 de junio de 2020

GitHub: https://github.com/cferna0256/Bioinformatics/tree/master/PEC_2_CarlosFdezMedina (https://github.com/cferna0256/Bioinformatics/tree/master/PEC_2_CarlosFdezMedina)

1 Abstract.

En este trabajo se analizan datos de expresión (RNA-seq) pertenecientes a un análisis del tiroides, en donde se compara tres tipos de infiltración medido en un total de 292 muestras pertenecientes a tres grupos: *Not infiltrated tissues* (NIT), *Small focal infiltrates* (SFI) y *Extensive lymphoid infiltrates* (ELI). Para ello creamos un *pipeline* utilizando el paquete *Bioconductor* en el software R. A partir del análisis realizado, se observa que las dos comparaciones que incluyen al grupo ELI presentan mayor expresión génica con respecto a la comparación restante.

2 Objetivos.

Los objetivos planteados en este trabajo son dos:

1. Crear un *pipeline* con la herramienta R para detectar diferencias significativas en la expresión de los genes del tejido de tiroides entre los tres grupos.
2. Elaborar un informe científico-técnico que muestre los resultados obtenidos.

3 Materiales, métodos y resultados.

3.1 Naturaleza de los datos y diseño experimental.

Los ficheros *targets.csv* y *counts.csv* contienen la información de las muestras de un estudio obtenido del repositorio GTEx. Este repositorio contiene datos de múltiples tipos en un total de 54 tejidos. En este estudio nos centraremos en los datos de expresión (RNA-seq) pertenecientes a un análisis del tiroides en donde se compara tres tipos de infiltración medido en un total de 292 muestras pertenecientes a tres grupos: • *Not infiltrated tissues* (NIT): 236 muestras • *Small focal infiltrates* (SFI): 42 muestras • *Extensive lymphoid infiltrates* (ELI): 14 muestras.

3.2 Procedimiento general de análisis (*Pipeline*).

1. Definición de los datos utilizados.
2. Filtraje y normalización de los datos.
3. Análisis de expresión diferencial.
4. Anotación de los resultados.
5. Comparación entre las distintas comparaciones y análisis de significación biológica.

3.2.1 Definición de los datos utilizados.

Primero, vamos a estudiar los datos y a organizarlos. En primer lugar, queremos obtener 30 muestras de manera aleatoria, 10 de cada grupo. Para ello, seguimos los siguientes pasos:

1. Preprocesamos el fichero *counts.csv* con Excel. Concretamente, eliminamos las versiones que aparecen en los identificadores de los transcritos (por ejemplo, ENSG00000223797 en vez de ENSG00000223797.**1**), ya que aparecen problemas a la hora de realizar la anotación de resultados.

2. Leemos los ficheros *targets.csv* y *counts.csv*, desde nuestro directorio de trabajo:

```
setwd("C:/Users/Carlos/Documents/MÁSTER BIOINFORMÁTICA - UOC/Análisis de Datos Ómicos/P
EC_2/") # definimos directorio de trabajo

targets <- read.csv("data/targets.csv", header = TRUE) # importamos fichero targets.csv

counts <- read.csv("data/counts.csv", sep=";",header = TRUE, row.names=1) # importamos
 fichero counts.csv
```

3. Extraemos 10 muestras de cada grupo a partir del fichero *targets.csv* utilizando para ello la librería `dplyr` . A este fichero lo llamaremos `coldata` para utilizarlo posteriormente:

```
# Construimos coldata

library(dplyr)
NIT <- filter(targets, Grupo_analisis == 1) # muestras grupo NIT
SFI <- filter(targets, Grupo_analisis == 2) # muestras grupo SFI
ELI <- filter(targets, Grupo_analisis == 3) # muestras grupo ELI
set.seed(123456) # semilla aleatoria para consistencia de datos
n <- 10 # definimos número de muestras
muestraNIT <- NIT %>% sample_n(size = n,replace=FALSE) # 10 muestras grupo NIT
muestraSFI <- SFI %>% sample_n(size = n,replace=FALSE) # 10 muestras grupo SFI
muestraELI <- ELI %>% sample_n(size = n,replace=FALSE) # 10 muestras grupo ELI
coldata <- rbind(muestraNIT, muestraSFI, muestraELI) # 30 muestras
rownames(coldata) <- coldata[,3] # nombramos a las filas según el Sample_Name
```

4. Relacionamos el filtro anterior con los datos del fichero *counts.csv* y llamamos a este fichero `countdata` :

```
# Construimos countdata
countdata <- as.matrix(counts[coldata$Sample_Name]) # relacionamos counts con coldata

# Comprobamos que los registros en coldata se corresponden con countdata
all(rownames(coldata) %in% colnames(countdata))
```

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

```
all(rownames(coldata) == colnames(countdata))
```

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

5. Por último, utilizamos la librería `DESeq2` para construir la matriz *DESeqDataSet* a partir de la matriz de *counts* filtrada (`countdata`) y de la información de las 30 muestras (`coldata`), así como el objeto *DESeqDataSet*, que utilizaremos para los siguientes pasos.

```
library("DESeq2")
```

```
ddsMat <- DESeqDataSetFromMatrix(countData = countdata,
                                 colData = coldata,
                                 design = ~ Group) # Generamos matriz DataSeq
ddsMat
```

```
## class: DESeqDataSet
## dim: 56202 30
## metadata(1): version
## assays(1): counts
## rownames(56202): ENSG00000223972 ENSG00000227232 ... ENSG00000210195
##   ENSG00000210196
## rowData names(0):
## colnames(30): GTEX.139YR.1226.SM.5IFEU GTEX.ZYW4.1126.SM.5SI99 ...
##   GTEX.TMMY.0826.SM.33HB9 GTEX.13QJC.0826.SM.5RQKC
## colData names(9): Experiment SRA_Sample ... Group ShortName
```

```
dds <- DESeq(ddsMat) # generamos objeto DESeq Set
dds
```

```
## class: DESeqDataSet
## dim: 56202 30
## metadata(1): version
## assays(6): counts mu ... replaceCounts replaceCooks
## rownames(56202): ENSG00000223972 ENSG00000227232 ... ENSG00000210195
##   ENSG00000210196
## rowData names(27): baseMean baseVar ... maxCooks replace
## colnames(30): GTEX.139YR.1226.SM.5IFEU GTEX.ZYW4.1126.SM.5SI99 ...
##   GTEX.TMMY.0826.SM.33HB9 GTEX.13QJC.0826.SM.5RQKC
## colData names(11): Experiment SRA_Sample ... sizeFactor replaceable
```

3.2.2 Filtraje y normalización de los datos.

3.2.2.1 Prefiltrado de los datos.

Realizamos un prefiltrado de datos con el objetivo de reducir el tamaño del objeto y de incrementar la rapidez de ejecución de los distintos *scripts*. Para ello, eliminamos los registros del *DESeqDataSeq* que no tengan contajes o que solo tengan un contaje:

```
nrow(dds)
```

```
## [1] 56202
```

```
dds2 <- dds[ rowSums(counts(dds)) > 1, ] # objeto de datos filtrados
nrow(dds2)
```

```
## [1] 43329
```

```
dds2
```

```
## class: DESeqDataSet
## dim: 43329 30
## metadata(1): version
## assays(6): counts mu ... replaceCounts replaceCooks
## rownames(43329): ENSG00000223972 ENSG00000227232 ... ENSG00000210195
##   ENSG00000210196
## rowData names(27): baseMean baseVar ... maxCooks replace
## colnames(30): GTEX.139YR.1226.SM.5IFEU GTEX.ZYW4.1126.SM.5SI99 ...
##   GTEX.TMMY.0826.SM.33HB9 GTEX.13QJC.0826.SM.5RQKC
## colData names(11): Experiment SRA_Sample ... sizeFactor replaceable
```

Podemos observar que con este prefiltrado, nos quedamos con 43329 registros de los 56202 iniciales.

3.2.2.2 Transformación estabilizadora de la varianza (VST) y *rlog*.

En el análisis de secuencias de RNA la varianza esperada crece al mismo tiempo que la media, por lo que se hace necesaria una transformación de los datos de la matriz *counts* para estabilizar la varianza. El paquete `DESeq2` ofrece dos transformaciones en este sentido: Transformación Estabilidazadora de la Varianza (VST, del inglés *Variance Stabilizing Transformation*) y la transformación *rlog*. Realizamos ambas transformaciones para comentar diferencias:

```
# VST
vsd <- vst(dds2, blind = FALSE)
head(assay(vsd), 3)
```

```
##                 GTEX.139YR.1226.SM.5IFEU GTEX.ZYW4.1126.SM.5SI99
## ENSG00000223972                 4.640210                4.336738
## ENSG00000227232                 7.898041                9.311691
## ENSG00000243485                 4.305638                3.966490
##                 GTEX.12WSL.0626.SM.5GCOY GTEX.147GR.0726.SM.5S2PL
## ENSG00000223972                 4.500976                 4.544113
## ENSG00000227232                 8.622673                 9.490165
## ENSG00000243485                 3.966490                 4.376285
##                 GTEX.OIZG.0226.SM.2TC5L GTEX.YEC3.0826.SM.4WWFP
## ENSG00000223972                4.957043                4.657747
## ENSG00000227232                8.749811                9.400117
## ENSG00000243485                4.377480                4.457597
##                 GTEX.XBEW.0126.SM.4AT66 GTEX.ZXG5.0926.SM.5NQ8H
## ENSG00000223972                4.383585                4.363659
## ENSG00000227232                9.351482                9.834911
## ENSG00000243485                4.383585                4.650166
##                 GTEX.13NYB.0726.SM.5MR4J GTEX.111YS.0726.SM.5GZY8
## ENSG00000223972                 4.298474                 4.726235
## ENSG00000227232                10.202681                 8.872956
## ENSG00000243485                 4.298474                 4.506777
##                 GTEX.11EQ8.0826.SM.5N9FG GTEX.RM2N.0526.SM.2TF4N
## ENSG00000223972                 4.312446                4.616220
## ENSG00000227232                 9.560987                8.867710
## ENSG00000243485                 4.312446                4.714677
##                 GTEX.13O1R.0826.SM.5J2MB GTEX.ZYVF.1126.SM.5E458
## ENSG00000223972                 4.570927                4.495394
## ENSG00000227232                 8.838166                9.847441
## ENSG00000243485                 3.966490                4.341522
##                 GTEX.R55C.0626.SM.2TF4Q GTEX.11DXY.0426.SM.5H12R
## ENSG00000223972                5.176368                 3.96649
## ENSG00000227232                8.720210                 9.41538
## ENSG00000243485                4.785863                 3.96649
##                 GTEX.131YS.0726.SM.5P9G9 GTEX.13NYC.2426.SM.5MR3K
## ENSG00000223972                 4.274563                 3.966490
## ENSG00000227232                10.096808                 9.493169
## ENSG00000243485                 4.274563                 3.966490
##                 GTEX.117YW.0126.SM.5EGGN GTEX.OXRP.0326.SM.33HBJ
## ENSG00000223972                 4.393888                4.469968
## ENSG00000227232                 9.448003                9.650131
## ENSG00000243485                 3.966490                4.546889
##                 GTEX.11NV4.0626.SM.5N9BR GTEX.YJ89.0726.SM.5P9F7
## ENSG00000223972                 4.565208                4.543902
## ENSG00000227232                10.247323                9.767536
## ENSG00000243485                 4.313806                4.256638
##                 GTEX.11XUK.0226.SM.5EQLW GTEX.14ABY.0926.SM.5Q5DY
## ENSG00000223972                 3.966490                 4.291355
## ENSG00000227232                 8.885394                 9.339412
## ENSG00000243485                 3.966490                 4.424958
##                 GTEX.14AS3.0226.SM.5Q5B6 GTEX.YFC4.2626.SM.5P9FQ
## ENSG00000223972                 3.966490                4.243306
## ENSG00000227232                 9.899991                9.782615
## ENSG00000243485                 4.349520                4.243306
##                 GTEX.ZYY3.1926.SM.5GZXS GTEX.R55G.0726.SM.2TC6J
## ENSG00000223972                4.883608                5.072338
## ENSG00000227232               10.133326                8.850121
## ENSG00000243485                4.346142                4.615217
##                 GTEX.TMMY.0826.SM.33HB9 GTEX.13QJC.0826.SM.5RQKC
## ENSG00000223972                 4.478779                 3.966490
## ENSG00000227232                 9.411378                 9.814173
## ENSG00000243485                 4.478779                 4.339991
```

```
colData(vsd)
```

```
## DataFrame with 30 rows and 11 columns
##                          Experiment SRA_Sample             Sample_Name
##                            <factor>   <factor>                <factor>
## GTEX.139YR.1226.SM.5IFEU  SRX560623  SRS624912 GTEX.139YR.1226.SM.5IFEU
## GTEX.ZYW4.1126.SM.5SI99   SRX617567  SRS644468  GTEX.ZYW4.1126.SM.5SI99
## GTEX.12WSL.0626.SM.5GCOY  SRX641141  SRS650189 GTEX.12WSL.0626.SM.5GCOY
## GTEX.147GR.0726.SM.5S2PL  SRX627705  SRS648114 GTEX.147GR.0726.SM.5S2PL
## GTEX.OIZG.0226.SM.2TC5L   SRX203686  SRS374813  GTEX.OIZG.0226.SM.2TC5L
## ...                             ...        ...                     ...
## GTEX.YFC4.2626.SM.5P9FQ   SRX615373  SRS644099  GTEX.YFC4.2626.SM.5P9FQ
## GTEX.ZYY3.1926.SM.5GZXS   SRX568364  SRS627095  GTEX.ZYY3.1926.SM.5GZXS
## GTEX.R55G.0726.SM.2TC6J   SRX204036  SRS374975  GTEX.R55G.0726.SM.2TC6J
## GTEX.TMMY.0826.SM.33HB9   SRX222429  SRS389623  GTEX.TMMY.0826.SM.33HB9
## GTEX.13QJC.0826.SM.5RQKC  SRX601511  SRS638114 GTEX.13QJC.0826.SM.5RQKC
##                          Grupo_analisis body_site        molecular_data_type
##                               <integer>  <factor>                  <factor>
## GTEX.139YR.1226.SM.5IFEU              1   Thyroid Allele-Specific Expression
## GTEX.ZYW4.1126.SM.5SI99               1   Thyroid              RNA Seq (NGS)
## GTEX.12WSL.0626.SM.5GCOY              1   Thyroid              RNA Seq (NGS)
## GTEX.147GR.0726.SM.5S2PL              1   Thyroid              RNA Seq (NGS)
## GTEX.OIZG.0226.SM.2TC5L               1   Thyroid              RNA Seq (NGS)
## ...                                 ...       ...                       ...
## GTEX.YFC4.2626.SM.5P9FQ               3   Thyroid Allele-Specific Expression
## GTEX.ZYY3.1926.SM.5GZXS               3   Thyroid Allele-Specific Expression
## GTEX.R55G.0726.SM.2TC6J               3   Thyroid              RNA Seq (NGS)
## GTEX.TMMY.0826.SM.33HB9               3   Thyroid Allele-Specific Expression
## GTEX.13QJC.0826.SM.5RQKC              3   Thyroid Allele-Specific Expression
##                              sex    Group ShortName        sizeFactor
##                          <factor> <factor>  <factor>         <numeric>
## GTEX.139YR.1226.SM.5IFEU     male      NIT 139YR_NIT  1.15223105996581
## GTEX.ZYW4.1126.SM.5SI99      male      NIT ZYW4._NIT 0.965937645086996
## GTEX.12WSL.0626.SM.5GCOY     male      NIT 12WSL_NIT 0.921533506208229
## GTEX.147GR.0726.SM.5S2PL     male      NIT 147GR_NIT 0.787524747223083
## GTEX.OIZG.0226.SM.2TC5L      male      NIT OIZG._NIT 0.782921072556282
## ...                           ...      ...       ...               ...
## GTEX.YFC4.2626.SM.5P9FQ    female      ELI YFC4._ELI  1.73221716541755
## GTEX.ZYY3.1926.SM.5GZXS    female      ELI ZYY3._ELI 0.918418128425997
## GTEX.R55G.0726.SM.2TC6J    female      ELI R55G._ELI 0.311090238552623
## GTEX.TMMY.0826.SM.33HB9    female      ELI TMMY._ELI  1.50608226625364
## GTEX.13QJC.0826.SM.5RQKC   female      ELI 13QJC_ELI 0.949093181406718
##                          replaceable
##                            <logical>
## GTEX.139YR.1226.SM.5IFEU        TRUE
## GTEX.ZYW4.1126.SM.5SI99         TRUE
## GTEX.12WSL.0626.SM.5GCOY        TRUE
## GTEX.147GR.0726.SM.5S2PL        TRUE
## GTEX.OIZG.0226.SM.2TC5L         TRUE
## ...                              ...
## GTEX.YFC4.2626.SM.5P9FQ         TRUE
## GTEX.ZYY3.1926.SM.5GZXS         TRUE
## GTEX.R55G.0726.SM.2TC6J         TRUE
## GTEX.TMMY.0826.SM.33HB9         TRUE
## GTEX.13QJC.0826.SM.5RQKC        TRUE
```

```
# rlog
rld <- rlog(dds2, blind = FALSE)
```

```
## rlog() may take a few minutes with 30 or more samples,
## vst() is a much faster transformation
```

```
head(assay(vsd), 3)
```

```
##                 GTEX.139YR.1226.SM.5IFEU GTEX.ZYW4.1126.SM.5SI99
## ENSG00000223972                 4.640210                4.336738
## ENSG00000227232                 7.898041                9.311691
## ENSG00000243485                 4.305638                3.966490
##                 GTEX.12WSL.0626.SM.5GCOY GTEX.147GR.0726.SM.5S2PL
## ENSG00000223972                 4.500976                 4.544113
## ENSG00000227232                 8.622673                 9.490165
## ENSG00000243485                 3.966490                 4.376285
##                 GTEX.OIZG.0226.SM.2TC5L GTEX.YEC3.0826.SM.4WWFP
## ENSG00000223972                4.957043                4.657747
## ENSG00000227232                8.749811                9.400117
## ENSG00000243485                4.377480                4.457597
##                 GTEX.XBEW.0126.SM.4AT66 GTEX.ZXG5.0926.SM.5NQ8H
## ENSG00000223972                4.383585                4.363659
## ENSG00000227232                9.351482                9.834911
## ENSG00000243485                4.383585                4.650166
##                 GTEX.13NYB.0726.SM.5MR4J GTEX.111YS.0726.SM.5GZY8
## ENSG00000223972                 4.298474                 4.726235
## ENSG00000227232                10.202681                 8.872956
## ENSG00000243485                 4.298474                 4.506777
##                 GTEX.11EQ8.0826.SM.5N9FG GTEX.RM2N.0526.SM.2TF4N
## ENSG00000223972                 4.312446                4.616220
## ENSG00000227232                 9.560987                8.867710
## ENSG00000243485                 4.312446                4.714677
##                 GTEX.13O1R.0826.SM.5J2MB GTEX.ZYVF.1126.SM.5E458
## ENSG00000223972                 4.570927                4.495394
## ENSG00000227232                 8.838166                9.847441
## ENSG00000243485                 3.966490                4.341522
##                 GTEX.R55C.0626.SM.2TF4Q GTEX.11DXY.0426.SM.5H12R
## ENSG00000223972                5.176368                 3.96649
## ENSG00000227232                8.720210                 9.41538
## ENSG00000243485                4.785863                 3.96649
##                 GTEX.131YS.0726.SM.5P9G9 GTEX.13NYC.2426.SM.5MR3K
## ENSG00000223972                 4.274563                 3.966490
## ENSG00000227232                10.096808                 9.493169
## ENSG00000243485                 4.274563                 3.966490
##                 GTEX.117YW.0126.SM.5EGGN GTEX.OXRP.0326.SM.33HBJ
## ENSG00000223972                 4.393888                4.469968
## ENSG00000227232                 9.448003                9.650131
## ENSG00000243485                 3.966490                4.546889
##                 GTEX.11NV4.0626.SM.5N9BR GTEX.YJ89.0726.SM.5P9F7
## ENSG00000223972                 4.565208                4.543902
## ENSG00000227232                10.247323                9.767536
## ENSG00000243485                 4.313806                4.256638
##                 GTEX.11XUK.0226.SM.5EQLW GTEX.14ABY.0926.SM.5Q5DY
## ENSG00000223972                 3.966490                 4.291355
## ENSG00000227232                 8.885394                 9.339412
## ENSG00000243485                 3.966490                 4.424958
##                 GTEX.14AS3.0226.SM.5Q5B6 GTEX.YFC4.2626.SM.5P9FQ
## ENSG00000223972                 3.966490                4.243306
## ENSG00000227232                 9.899991                9.782615
## ENSG00000243485                 4.349520                4.243306
##                 GTEX.ZYY3.1926.SM.5GZXS GTEX.R55G.0726.SM.2TC6J
## ENSG00000223972                4.883608                5.072338
## ENSG00000227232               10.133326                8.850121
## ENSG00000243485                4.346142                4.615217
##                 GTEX.TMMY.0826.SM.33HB9 GTEX.13QJC.0826.SM.5RQKC
## ENSG00000223972                 4.478779                 3.966490
## ENSG00000227232                 9.411378                 9.814173
## ENSG00000243485                 4.478779                 4.339991
```

Por último, con el objetivo de ver el efecto de la transformación, creamos una figura con las distintas transformaciones. En la imagen de la izquierda, utilizamos la función `log2` y los comparamos con la transformación *vst* y *rlog* (figura 1).

```
library("dplyr")
library("ggplot2")
```

```
dds3 <- estimateSizeFactors(dds2)

df <- bind_rows(
  as_data_frame(log2(counts(dds3, normalized=TRUE)[, 1:2]+1)) %>%
         mutate(transformation = "log2(x + 1)"),
  as_data_frame(assay(rld)[, 1:2]) %>% mutate(transformation = "rlog"),
  as_data_frame(assay(vsd)[, 1:2]) %>% mutate(transformation = "vst"))
colnames(df)[1:2] <- c("x", "y")
ggplot(df, aes(x = x, y = y)) + geom_hex(bins = 80) +
  coord_fixed() + facet_grid( . ~ transformation)
```

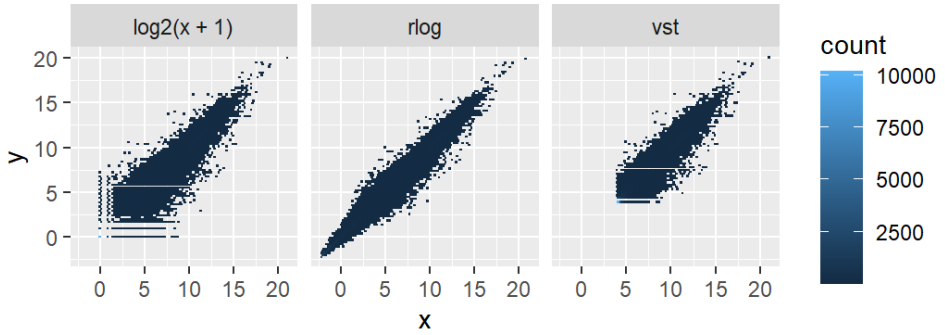

Figura 1. Diagramas de dispersión utilizando la transformación log2 de recuentos normalizados (izquierda), rlog (centro) y VST (derecha).

La transformación VST es mucho más rápida computacionalmente, pero menos sensible con los datos "*outliers*" que *rlog*. El método *rlog* suele funcionar mejor con conjuntos de datos pequeños (con una $n < 30$), mientras que VST se recomienda para conjuntos de datos mayores ($n > 30$). En este caso, como tenemos 30 muestras, usamos los dos métodos y observamos que el *rlog* está aproximadamente en la misma escala que el *log2*, mientras que el VST tiene un desplazamiento hacia arriba para los valores más pequeños.

3.2.2.3 Similitud entre muestras.

Un paso útil en el análisis de RNA-seq es evaluar la similitud general entre muestras, es decir, qué muestras son similares entre sí, cuáles son diferentes y si esto se ajusta las expectativas del diseño del experimento.

Para ello, el paquete *DESeq2* incluye una función llamada `dist` que realiza este estudio. Los resultados se pueden ver de una mejor manera mediante un mapa de calor (figura 2).

```
library("pheatmap")
library("RColorBrewer")
```

```
sampleDists <- dist(t(assay(vsd))) # Similitud entre muestras
head(sampleDists)
```

```
## [1] 146.5265 105.1612 122.2497 114.5824 133.1921 134.2941
```

```
sampleDistMatrix <- as.matrix(sampleDists) # Convertimos en matriz el objeto
rownames(sampleDistMatrix) <- paste(vsd$Group, sep = " - ") # Nombramos las filas
colnames(sampleDistMatrix) <- NULL # No queremos nombres en las columnas
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255) # Establecemos colores
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors) # Generamos mapa de calor de similitud entre muestras
```

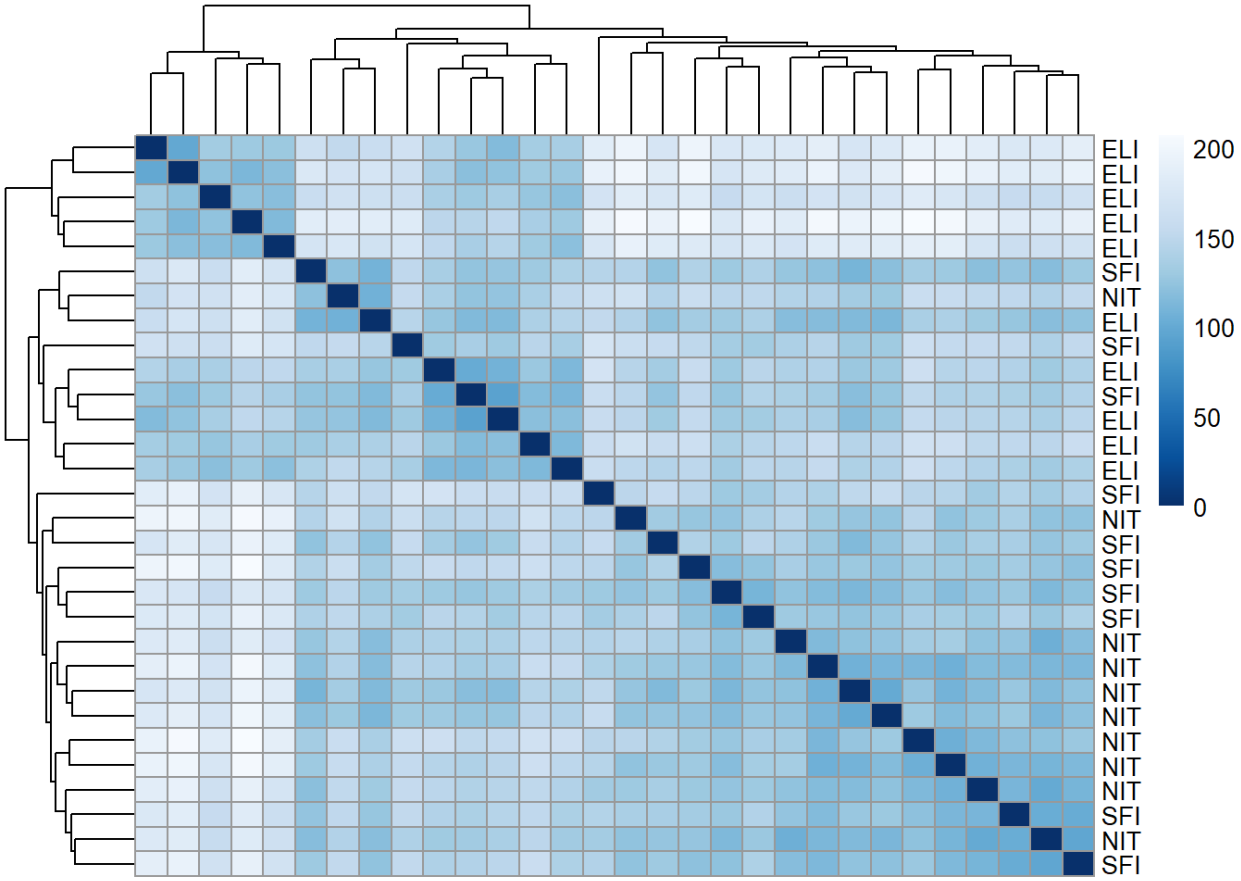

Figura 2. Mapa de calor de distancias entre muestras utilizando los valores transformados por rlog.

3.2.2.4 Gráfico PCA para el control de calidad.

Visualizamos los datos filtrados mediante un gráfico PCA (*Principal Component Analysis*, figura 3) con el objetivo de observar la calidad de los mismos.

```
plotPCA(vsd, intgroup = c("Group"))
```

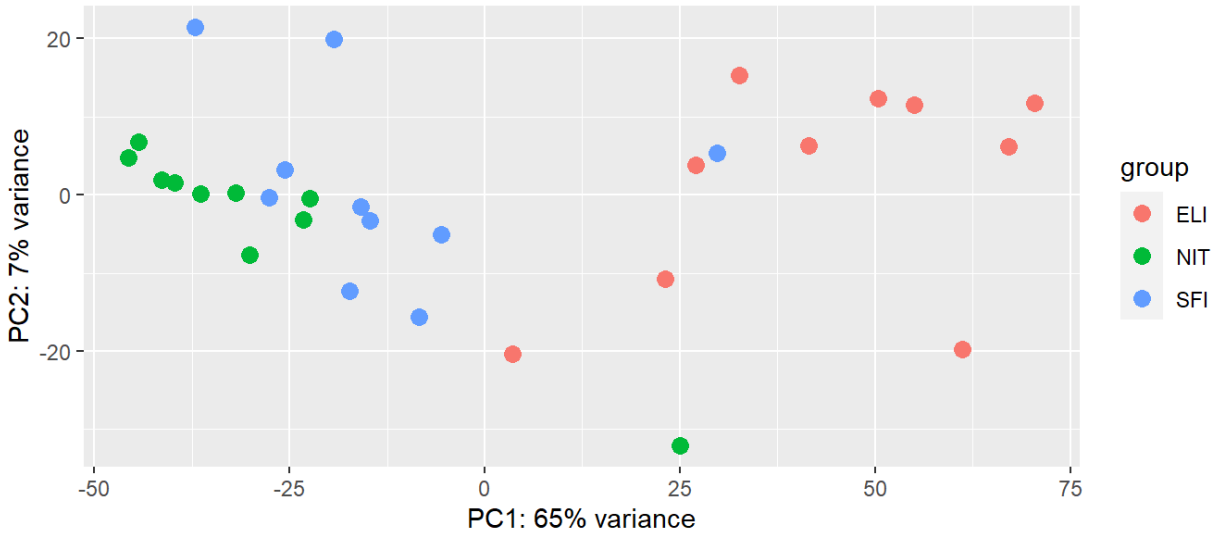

Figura 3. Análisis de componentes principales.

Observamos que las muestras del grupo ELI se agrupan mayoritariamente a la derecha del gráfico mientras las de los grupos NIT y SFI se sitúan a la izquierda.Podemos ver que el primer componente de los datos filtrados presenta el 65% de la variabilidad de las muestras, mientras que la segunda componente solo presenta por el 7%.

3.2.3 Análisis de expresión diferencial.

Procedemos a realizar el análisis de expresión diferencial. Para ello, utilizamos objeto `DESeq` creado en los pasos anteriores y realizamos comparaciones entre los tres grupos: NIT vs. ELI, SFI vs. ELI y NIT vs. SFI. El objeto DESeq contiene distintos parámetros ajustados, por lo que podemos utilizar los que consideremos para realizar el análisis. Utilizamos la función `results` con este objetivo:

```
# Comparación NIT vs. ELI
resNITvsELI <- results(dds, contrast = c("Group","NIT","ELI"))

# Comparación SFI vs. ELI
resSFIvsELI <- results(dds, contrast = c("Group","SFI","ELI"))

# Comparación NIT vs. SFI
resNITvsSFI <- results(dds, contrast = c("Group","NIT","SFI"))
```

Estas nuevas variables creadas con la función `results` contienen metadatos con información sobre cada columna:

```
mcols(resNITvsELI, use.names = TRUE)
```

```
## DataFrame with 6 rows and 2 columns
##                        type                                description
##                 <character>                                <character>
## baseMean       intermediate mean of normalized counts for all samples
## log2FoldChange      results  log2 fold change (MLE): Group NIT vs ELI
## lfcSE               results          standard error: Group NIT vs ELI
## stat                results          Wald statistic: Group NIT vs ELI
## pvalue              results       Wald test p-value: Group NIT vs ELI
## padj                results                         BH adjusted p-values
```

Realizamos gráficos de tipo MA-plot, mediante la función `plotMA` (figura 4). Estos gráficos nos permiten visualizar genes que se encuentran diferencialmente expresados mediante cambios de log2 atribuibles a una variable dada sobre la media de los recuentos normalizados para todas las muestras del objeto *DESeqDataSet*.

```
par(mfrow=c(1,3), mar=c(4,4,2,1))
plotMA(resNITvsELI, ylim=c(-3,3), main = "resNITvsELI")
plotMA(resSFIvsELI, ylim=c(-3,3), main = "resSFIvsELI")
plotMA(resNITvsSFI, ylim=c(-3,3), main = "resNITvsSFI")
```

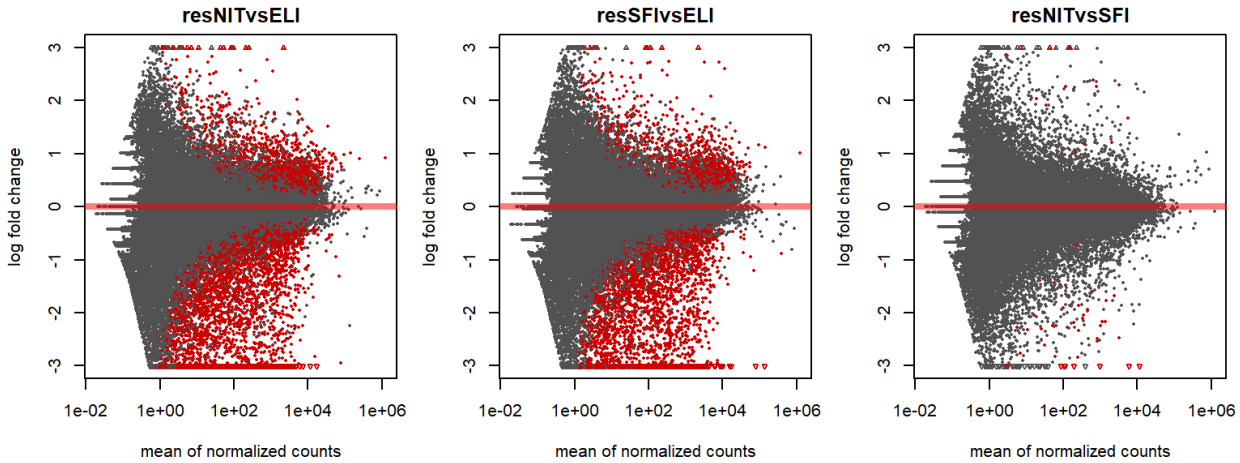

Figura 4: Gráficas de MA. Los puntos de color rojo representan los p-valores ajustado menores de 0.1 y los puntos que se representan por triángulos abiertos que apuntan hacia arriba o hacia abajo son valores que se salen de la gráfica.

Podemos ordenar los *p-valores* obtenidos de menor a mayor valor y realizar un resumen de los principales estadísticos:

```
resOrd_NITvsELI <- resNITvsELI[order(resNITvsELI$pvalue),]
summary(resNITvsELI)
```

```
##
## out of 46151 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1891, 4.1%
## LFC < 0 (down)     : 3841, 8.3%
## outliers [1]       : 0, 0%
## low counts [2]     : 14155, 31%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
resOrd_SFIvsELI <- resSFIvsELI[order(resNITvsELI$pvalue),]
summary(resSFIvsELI)
```

```
##
## out of 46151 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1429, 3.1%
## LFC < 0 (down)     : 2980, 6.5%
## outliers [1]       : 0, 0%
## low counts [2]     : 15924, 35%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
resOrd_NITvsSFI <- resNITvsSFI[order(resNITvsSFI$pvalue),]
summary(resNITvsSFI)
```

```
##
## out of 46151 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 15, 0.033%
## LFC < 0 (down)     : 46, 0.1%
## outliers [1]       : 0, 0%
## low counts [2]     : 17693, 38%
## (mean count < 2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Podemos ser más o menos estrictos para determinar los genes cuyas diferencias en la expresión son estadísticamente significativas. En este primer cálculo, ponemos el umbral más bajo:

```
res.05 <- results(dds2, alpha = 0.05)
table(res.05$padj < 0.05)
```

```
##
## FALSE  TRUE
## 26630  3256
```

Ahora, elevamos el umbral:

```
resLFC1 <- results(dds, lfcThreshold=1)
table(resLFC1$padj < 0.1)
```

```
##
## FALSE  TRUE
## 27160   416
```

Por lo tanto, si consideramos que una tasa del 10% de falsos positivos es aceptable, podemos considerar todos los genes con un p-valor ajustado por debajo del 10% = 0.1 como expresados diferencialmente de forma significativa:

```
sum(resNITvsELI$padj < 0.1, na.rm=TRUE)
```

```
## [1] 5732
```

```
sum(resSFIvsELI$padj < 0.1, na.rm=TRUE)
```

```
## [1] 4409
```

```
sum(resNITvsSFI$padj < 0.1, na.rm=TRUE)
```

```
## [1] 61
```

A continuación, hacemos subconjuntos con la función subset y clasificamos los genes. Con ello, obtenemos los genes infraexpresados (*down-regulated*) con una expresión diferencial más fuerte estadísticamente y los genes con una sobreexpresión más fuerte (*up-regulated*):

```
resSigNITvsELI <- subset(resNITvsELI, padj < 0.1)
head(resSigNITvsELI[ order(resSigNITvsELI$log2FoldChange), ]) # down-regulated
```

```
## log2 fold change (MLE): Group NIT vs ELI
## Wald test p-value: Group NIT vs ELI
## DataFrame with 6 rows and 6 columns
##                          baseMean    log2FoldChange             lfcSE
##                         <numeric>         <numeric>         <numeric>
## ENSG00000229807  11639.8014525624 -10.0852837894159  1.19635341128736
## ENSG00000255760  85.0925249874743 -9.10153697069768    1.087057911937
## ENSG00000100721  392.330758120949 -9.06378424916742  1.09808904913321
## ENSG000001700540 52.4940352250393  -8.7426763273314  1.31919219824988
## ENSG00000257275  27.8982167952771 -8.70999383733212   1.6258532001492
## ENSG000001635340 981.332670745295  -8.2257554502604 0.854592129796747
##                             stat               pvalue                padj
##                        <numeric>            <numeric>           <numeric>
## ENSG00000229807  -8.43002050586654 3.45604652202388e-17 1.18914013611013e-14
## ENSG00000255760  -8.37263302235655 5.63448633371919e-17  1.8029792819268e-14
## ENSG00000100721  -8.25414319204991 1.52995834103305e-16 4.07976141289305e-14
## ENSG000001700540  -6.6272953546344 3.41893130190114e-11 2.96483422031259e-09
## ENSG00000257275  -5.35718343853726 8.45292893139547e-08 3.74115176868221e-06
## ENSG000001635340 -9.62535830070982 6.24887607051447e-22 7.69068405309202e-19
```

```
head(resSigNITvsELI[ order(resSigNITvsELI$log2FoldChange, decreasing = TRUE), ]) # up-r
egulated
```

```
## log2 fold change (MLE): Group NIT vs ELI
## Wald test p-value: Group NIT vs ELI
## DataFrame with 6 rows and 6 columns
##                          baseMean   log2FoldChange             lfcSE
##                         <numeric>        <numeric>         <numeric>
## ENSG00000110680  2163.57508183693 5.54561128906044  1.47480917345449
## ENSG00000079689  81.4056732458653 5.14605879079007 0.857978398559983
## ENSG00000224714  3.21841489354298 4.55085957436397  1.46142640190076
## ENSG00000228411  1.26344615571461 3.75335397481996  1.41513403281953
## ENSG00000157005  11.0386231097852 3.74274671290751  1.15674189383497
## ENSG000000892255 208.805466125787  3.5634314761145 0.783387646464628
##                             stat               pvalue                padj
##                        <numeric>            <numeric>           <numeric>
## ENSG00000110680  3.76022294197614 0.000169761997714649  0.00287324864188609
## ENSG00000079689  5.99788852426487 1.99899664751291e-09 1.21839797569077e-07
## ENSG00000224714  3.11398478120077  0.00184579010212019   0.0193144007448476
## ENSG00000228411  2.65229574568406  0.00799464820735127   0.0566727399173756
## ENSG00000157005  3.23559363835185  0.00121390062735866   0.0140280267875948
## ENSG000000892255 4.54874606741121 5.39665151222895e-06 0.000152013601883639
```

```
resSigSFIvsELI <- subset(resSFIvsELI, padj < 0.1)
head(resSigSFIvsELI[ order(resSigSFIvsELI$log2FoldChange), ]) # down-regulated
```

```
## log2 fold change (MLE): Group SFI vs ELI
## Wald test p-value: Group SFI vs ELI
## DataFrame with 6 rows and 6 columns
##                          baseMean    log2FoldChange            lfcSE
##                         <numeric>         <numeric>        <numeric>
## ENSG00001628970 69.2232228257621 -8.31002937949534 1.48668688634475
## ENSG00000254029  18.9598435531979 -8.16693893486635 2.53056550394058
## ENSG00000211979  17.6061172744022 -7.08179874849105 1.24019950404907
## ENSG00000257275  27.8982167952771 -6.72949728781579 1.57577514668928
## ENSG00000242580  62.6749878800144 -6.49653310990557 1.64137763874238
## ENSG00000253202   8.1714750716581 -6.40660553794855 1.76972445931027
##                             stat               pvalue                padj
##                         <numeric>            <numeric>           <numeric>
## ENSG00001628970 -5.58962983787853 2.27554166987585e-08 2.40523163217996e-06
## ENSG00000254029  -3.22731773674653  0.00124956619789331   0.0169620054612998
## ENSG00000211979  -5.71020930533356 1.12837315640895e-08  1.3699084545479e-06
## ENSG00000257275  -4.27059488909602 1.94952251797655e-05 0.000649769192044446
## ENSG00000242580  -3.95797588352867 7.55875872931097e-05  0.00193034857248092
## ENSG00000253202  -3.62011470443566 0.000294472445092875  0.00557763284157745
```

```
head(resSigSFIvsELI[ order(resSigSFIvsELI$log2FoldChange, decreasing = TRUE), ]) # up-r
egulated
```

```
## log2 fold change (MLE): Group SFI vs ELI
## Wald test p-value: Group SFI vs ELI
## DataFrame with 6 rows and 6 columns
##                          baseMean   log2FoldChange             lfcSE
##                         <numeric>        <numeric>         <numeric>
## ENSG00000110680 2163.57508183693 7.31134343966858  1.47477051272508
## ENSG00000128564  86.559494693927 4.82887396027247  1.02270034337421
## ENSG00000224714 3.21841489354298 4.31838947086168  1.46210879027292
## ENSG00000232893 2.59233043832856 4.18955811120693  1.37952778535326
## ENSG00000108342 114.092776092716 3.82703074175467 0.909969521914989
## ENSG00000079689 81.4056732458653 3.77639401162312 0.858975086297236
##                             stat               pvalue                padj
##                         <numeric>            <numeric>           <numeric>
## ENSG00000110680 4.95761433835469  7.1364044852617e-07 4.05514112010266e-05
## ENSG00000128564 4.72168997649935 2.33893017897088e-06 0.000107259103462435
## ENSG00000224714 2.95353498972918  0.00314156992342064   0.0336835689657363
## ENSG00000232893 3.03695087238428  0.00238984402319886   0.0277225574909062
## ENSG00000108342 4.20566914560046 2.60310815191673e-05 0.000818427947244646
## ENSG00000079689 4.39639527602824 1.10063424938081e-05 0.000398469142021339
```

```
resSigNITvsSFI <- subset(resNITvsSFI, padj < 0.1)
head(resSigNITvsSFI[ order(resSigNITvsSFI$log2FoldChange), ]) # down-regulated
```

```
## log2 fold change (MLE): Group NIT vs SFI
## Wald test p-value: Group NIT vs SFI
## DataFrame with 6 rows and 6 columns
##                         baseMean    log2FoldChange             lfcSE
##                        <numeric>         <numeric>         <numeric>
## ENSG00000229807  11639.8014525624 -8.94722348451969  1.19635883539822
## ENSG00000255760  85.0925249874743 -4.74018690750406  1.09444069130549
## ENSG000001888481  88.185068440339 -3.88369344838237 0.870656259852124
## ENSG00000235532  87.0507584846438 -3.85023028832202  1.02172402185554
## ENSG000001635340 981.332670745295 -3.71550366742302  0.85550262099262
## ENSG00000240535  5.47106508253582 -3.54055112519324 0.920553794521039
##                             stat               pvalue                 padj
##                        <numeric>            <numeric>            <numeric>
## ENSG00000229807  -7.47871225571007  7.5054403686228e-14 2.13612338331374e-09
## ENSG00000255760  -4.33115009809236 1.48332518935597e-05   0.0332740898068956
## ENSG000001888481 -4.46065069243514  8.1711179647332e-06   0.0290697735492839
## ENSG00000235532  -3.76836621823736 0.000164319511070342   0.0880651493538489
## ENSG000001635340 -4.34306520664076 1.40508407312555e-05   0.0332740898068956
## ENSG00000240535  -3.84610996800614 0.000120007918594329    0.079431287700307
```

```
head(resSigNITvsSFI[ order(resSigNITvsSFI$log2FoldChange, decreasing = TRUE), ]) # up-r
egulated
```

```
## log2 fold change (MLE): Group NIT vs SFI
## Wald test p-value: Group NIT vs SFI
## DataFrame with 6 rows and 6 columns
##                         baseMean   log2FoldChange             lfcSE
##                        <numeric>        <numeric>         <numeric>
## ENSG00000159763  42.2151262105706 4.65712005233626  1.02592944227259
## ENSG00000162078  153.247880954693 4.61868411137652 0.883944481217323
## ENSG00000230567  7.02805721511094 3.36256697890004 0.896226131321802
## ENSG00000227160  2.97886564655239 2.85586624838007  0.70990760971106
## ENSG00000178445  656.674467810871  2.3838815795946  0.51984250690012
## ENSG000001244911 3287.87045771214 2.30038290697918 0.606081420718207
##                             stat               pvalue                 padj
##                        <numeric>            <numeric>            <numeric>
## ENSG00000159763  4.53941553916226 5.64103702203649e-06  0.0229356506691687
## ENSG00000162078  5.22508393854771 1.74076123454787e-07 0.00247719027482334
## ENSG00000230567  3.75191802758612 0.000175486847367249  0.0891880564807013
## ENSG00000227160  4.02287031342352 5.74931572992248e-05  0.0511347734341636
## ENSG00000178445  4.58577655338336 4.52302132789578e-06  0.0229356506691687
## ENSG000001244911 3.79550144311176 0.000147345337971703    0.08558358497985
```

En resumen, podemos observar una cantidad de 5732 genes expresados de la comparación NIT vs. ELI, 4409 de la comparación SFI vs. ELI y 61 de la comparación NIT vs. SFI. Esto indica que existe una mayor expresión de genes en las comparaciones que incluyen al grupo ELI (NIT vs. ELI y SFI vs. ELI) que en la otra comparación (SFI vs. NIT), aparte de que se observa que las dos primeros presentan valores de expresión similares entre sí.

3.2.4 Anotación de los resultados.

Desde nuestra matriz importada *countData* tenemos solo los identificadores *Ensembl*, pero es mucho más útil tener una herramienta para poder saber con qué genes se relacionan.

Para ello, existe la librería `AnnotationDbi` , que permite mapear los identificadores con los genes que correspondan. También necesitaremos una base de datos que nos aporte esta información. En nuestro caso, utilizamos la base de datos *org.Hs.eg.db*, que se corresponde a anotaciones genómicas del ser

humano. Tras importarla, utilizamos la función `mapIds` para realizar dicho mapeo y ordenamos los datos obtenidos, relacionando el identificador *Ensembl* con los genes que corresponden.

```
library("org.Hs.eg.db")
library("AnnotationDbi")
```

```
# Mapeamos genes de los resultados de NIT vs ELI
resNITvsELI$symbol <- mapIds(org.Hs.eg.db,
                     keys=rownames(resNITvsELI),
                     column="SYMBOL",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
resNITvsELI$entrez <- mapIds(org.Hs.eg.db,
                     keys=rownames(resNITvsELI),
                     column="ENTREZID",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
resOrdered_NITvsELI <- resNITvsELI[order(resNITvsELI$pvalue),]
head(resOrdered_NITvsELI)
```

```
## log2 fold change (MLE): Group NIT vs ELI
## Wald test p-value: Group NIT vs ELI
## DataFrame with 6 rows and 8 columns
##                         baseMean    log2FoldChange             lfcSE
##                        <numeric>         <numeric>         <numeric>
## ENSG00000136573  1231.34057071827 -6.77657271639884 0.590607936972313
## ENSG000000834547 1138.65703089593 -6.30556343250846 0.571472030317519
## ENSG000000097900 1474.06470724423 -4.76164847294204 0.444771521480733
## ENSG000001054091 61.0125073747512  -3.0855911850552 0.291984199884919
## ENSG00000264198  1414.32992676752 -5.28423276752989 0.508453938390725
## ENSG00000186265  211.668221452204 -5.61257943485723 0.545924244530422
##                             stat               pvalue                 padj
##                        <numeric>            <numeric>            <numeric>
## ENSG00000136573  -11.4738937494443 1.78445212090191e-30 5.71006834167403e-26
## ENSG000000834547 -11.0338968453189 2.62249939267469e-28 4.19586790330987e-24
## ENSG000000097900 -10.7058303937482 9.55699333909855e-27 1.01938076619271e-22
## ENSG000001054091 -10.5676649156747 4.20842696881267e-26 3.36663636437592e-22
## ENSG00000264198  -10.3927462618436 2.67536839399508e-25 1.71218226478897e-21
## ENSG00000186265  -10.2808759476966 8.59429238769955e-25 4.58347936856663e-21
##                      symbol      entrez
##                 <character> <character>
## ENSG00000136573         BLK         640
## ENSG000000834547         NA          NA
## ENSG000000097900         NA          NA
## ENSG000001054091         NA          NA
## ENSG00000264198          NA          NA
## ENSG00000186265        BTLA      151888
```

```
# Mapeamos genes de los resultados de SFI vs ELI
resSFIvsELI$symbol <- mapIds(org.Hs.eg.db,
                     keys=rownames(resSFIvsELI),
                     column="SYMBOL",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
resSFIvsELI$entrez <- mapIds(org.Hs.eg.db,
                     keys=rownames(resSFIvsELI),
                     column="ENTREZID",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
resOrdered_SFIvsELI <- resSFIvsELI[order(resSFIvsELI$pvalue),]
head(resOrdered_SFIvsELI)
```

```
## log2 fold change (MLE): Group SFI vs ELI
## Wald test p-value: Group SFI vs ELI
## DataFrame with 6 rows and 8 columns
##                         baseMean    log2FoldChange             lfcSE
##                        <numeric>         <numeric>         <numeric>
## ENSG00000170006  368.888237487614 -2.67153275939965 0.318916209123075
## ENSG00000136573  1231.34057071827 -4.83064454362784  0.58601984628659
## ENSG00000245904  90.7003251507727 -3.03537565261272 0.369459157212701
## ENSG000001054091 61.0125073747512 -2.23486633825312 0.279604319009508
## ENSG000000106711 581.532395499199 -3.42333202591325 0.429045361683367
## ENSG00000268027  214.104853747565 -3.51478075965168 0.442377739821277
##                            stat               pvalue                 padj
##                        <numeric>            <numeric>            <numeric>
## ENSG00000170006   -8.3769111853724 5.43348173803344e-17 1.64254152940751e-12
## ENSG00000136573   -8.2431415492803 1.67746927917394e-16 2.12499110235815e-12
## ENSG00000245904  -8.21572721464643  2.1088234558632e-16 2.12499110235815e-12
## ENSG000001054091  -7.9929607173812 1.31736114132584e-15 8.92279816951732e-12
## ENSG000000106711 -7.97895125233786 1.47581842036343e-15 8.92279816951732e-12
## ENSG00000268027  -7.94520257974931  1.9387368565573e-15 9.76800252895454e-12
##                      symbol      entrez
##                 <character> <character>
## ENSG00000170006      TMEM154      201799
## ENSG00000136573          BLK         640
## ENSG00000245904           NA          NA
## ENSG000001054091          NA          NA
## ENSG000000106711          NA          NA
## ENSG00000268027           NA          NA
```

```
# Mapeamos genes de los resultados de NIT vs SFI
resNITvsSFI$symbol <- mapIds(org.Hs.eg.db,
                     keys=rownames(resNITvsSFI),
                     column="SYMBOL",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
resNITvsSFI$entrez <- mapIds(org.Hs.eg.db,
                     keys=rownames(resNITvsSFI),
                     column="ENTREZID",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
resOrdered_NITvsSFI <- resNITvsSFI[order(resNITvsSFI$pvalue),]
head(resOrdered_NITvsSFI)
```

```
## log2 fold change (MLE): Group NIT vs SFI
## Wald test p-value: Group NIT vs SFI
## DataFrame with 6 rows and 8 columns
##                         baseMean     log2FoldChange             lfcSE
##                        <numeric>          <numeric>         <numeric>
## ENSG00000229807 11639.8014525624  -8.94722348451969  1.19635883539822
## ENSG00000162078 153.247880954693   4.61868411137652 0.883944481217323
## ENSG00000153930 19.3009966783901   2.02915420406717 0.426461799540424
## ENSG00000178445 656.674467810871    2.3838815795946  0.51984250690012
## ENSG00000105376 360.074130280622  -2.04521834348565 0.447412035432337
## ENSG00000139514 1756.50190918809 -0.955411908866977 0.209576154025949
##                             stat               pvalue                 padj
##                        <numeric>            <numeric>            <numeric>
## ENSG00000229807 -7.47871225571007  7.5054403686228e-14 2.13612338331374e-09
## ENSG00000162078  5.22508393854771 1.74076123454787e-07  0.00247719027482334
## ENSG00000153930  4.75811480947153 1.95409351939048e-06   0.0185384852184575
## ENSG00000178445  4.58577655338336 4.52302132789578e-06   0.0229356506691687
## ENSG00000105376 -4.57121887995109 4.84895471931221e-06   0.0229356506691687
## ENSG00000139514 -4.55878157182273 5.14512489727402e-06   0.0229356506691687
##                      symbol      entrez
##                 <character> <character>
## ENSG00000229807        XIST        7503
## ENSG00000162078       ZG16B      124220
## ENSG00000153930      ANKFN1      162282
## ENSG00000178445        GLDC        2731
## ENSG00000105376       ICAM5        7087
## ENSG00000139514      SLC7A1        6541
```

Previo a estos pasos, hemos tenido que realizar una modificación en el fichero *counts.csv*, ya que los identificadores venían con versión (identificada con punto y número de versión). Al realizar el mapeo, daba error, por lo que hemos tenido que eliminar estos caracteres.

Tras la obtención de los resultados de las anotaciones, los exportamos a ficheros CSV:

```
resOrdered_NITvsELIexp <- as.data.frame(resOrdered_NITvsELI)
write.csv(resOrdered_NITvsELIexp, file = "resultados/results_NITvsELI.csv")

resOrdered_NITvsSFIexp <- as.data.frame(resOrdered_NITvsSFI)
write.csv(resOrdered_NITvsSFIexp, file = "resultados/results_NITvsSFI.csv")

resOrdered_SFIvsELIexp <- as.data.frame(resOrdered_SFIvsELI)
write.csv(resOrdered_NITvsSFIexp, file = "resultados/results_SFIvsELI.csv")
```

Podemos observar una asociación de genes con cada uno de los identificadores *Ensembl* del fichero, así como datos de los p-valores y otros estadísticos. Vemos también una cantidad considerable de registros indeterminados ("NA"), lo cual indica que el mapeo no se ha realizado correctamente para algunas de las muestras.

3.2.5 Comparación entre las distintas comparaciones y análisis de significación biológica.

En este estudio no ha sido posible realizar una comparación entre las distintas comparaciones ni un análisis de significación biológica. Como se ha comentado en el apartado anterior de la anotación de genes, tenemos muchos valores indeterminados ("NA") y el hecho de no poder conocer el origen de los datos nos hace imposible determinar estos dos apartados.

4 Resumen de resultados y discusión.

En este estudio hemos utilizado el paquete `DESeq` de Bioconductor en el software R para el análisis de expresión de R. Existen otras alternativas dentro de R, tales como utilizar los paquetes `edgeR` o `limma` o con otras herramientas distintas de R, como Galaxy. El hecho de utilizar `DESeq` se debe a que presenta una serie de funciones muy útiles y bastante fáciles de utilizar, a la vez que intuitivas y robustas, además de que existe una gran cantidad de manuales y documentación disponibles.

Hemos podido establecer un *pipeline* con este paquete en R. Para poder realizarlo, primero hemos tenido que preprocesar los datos aportados en el fichero *counts.csv* con el objetivo de eliminar la versión de cada muestra, ya que daba errores a la hora de realizar el posterior análisis. Esto puede influir a la hora de la interpretación de los datos, pero no podemos realizarlo de otra forma debido a que no tenemos acceso al origen de datos. Tras ello, hemos podido aplicar los distintos paquetes de visualización y tratamiento de datos, as´como el análisis estadístico propio de datos de secuenciación RNA.

En cuanto a los resultados obtenidos, hemos podido comprobar que existe una mayor expresión de genes en las comparaciones que incluyen al grupo ELI (NIT vs. ELI y SFI vs. ELI) que en el otro grupo (SFI vs. NIT), aparte de que se observa que los dos primeros presentan valores de expresión similares entre sí. Sin embargo, no podemos sacar conclusiones debido a que este grupo solo está representado por 14 muestras, mientras que los otros dos están representados por 42 y 236 muestras respectivamente, por lo que es probable que el tamaño muestral no sea el adecuado.

Por otro lado, al realizar el mapeo con la base de datos hemos encontrado una gran cantidad de registros NA, lo cual quiere decir que muchos registros de expresión no se corresponen con ningún gen. En consecuencia, no hemos podido realizar ni la comparación entre comparaciones ni el análisis de significación biológica, por lo que no podemos obtener resultados robustos.

Como conclusión, podemos decir que, aunque disponemos de bastantes datos se hace necesario revisar la representatividad de cada grupo y revisar el origen de los mismos para poder realizar un análisis más exhaustivo y concluyente.

5 Referencias.

- Gonzalo Sanz, Ricardo & Sánchez Pla, Alexandre. *RNAseq pipeline - Bioconductor*. Mayo, 2020.
- Love, Michael I.; Anders, Simon; Kim, Vladislav & Huber, Wolfgang. *RNA-seq workflow: gene-level exploratory analysis and differential expression.* Octubre, 2019. https://www.bioconductor.org/packages/devel/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html (https://www.bioconductor.org/packages/devel/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html).