PEC 2 Análisis de Datos Ómicos

Mónica Hortal Foronda

13 de junio de 2020

Repositorio GitHub: *https://github.com/monicahortal/PEC-2-An-lisis-de-Datos-micos.git*

Tabla de contenidos

1. Abstract

En este trabajo analizaremos los datos obtenidos en un estudio de RNA-seq sobre muestras de tejido de tiroides en las que se han comparado tres tipos de infiltración: *not filtered tissues* (NIT), *small focal infiltrates* (SFI) y *extensive lymphoid infiltrates* (ELI). Para realizar este análisis, utilizamos el paquete Bioconductor dentro del programa R. A partir del análisis realizado, podemos observar que el grupo ELI es el que presenta unas mayores diferencias en la expresión génica de los genes analizados con respecto a los grupos NIT y SFI, que son similares entre sí.

2. Objetivos

Los objetivos de este trabajo son elaborar y ejecutar un *pipeline* para analizar los datos de RNA-seq que se nos han proporcionado con el fin de detectar diferencias significativas en la expresión de genes entre los tres tipos de infiltración que se han realizado. Una vez obtenidos los resultados, debemos elaborar un informe técnico con la estructura tradicional para mostrar los resultados.

3. Materiales y métodos

3.1. Naturaleza de los datos, tipo de experimento y diseño experimental

Para analizar las diferencias de expresión génica con los distintos tipos de infiltración se nos han proporcionado dos ficheros csv: *targets*, que contiene la información sobre las muestras y *counts*, que contiene los resultados de los contajes obtenidos tras la secuenciación mediante RNA-seq. Estos datos proceden de muestras de un estudio obtenido del repositorio GTEx.

3.2. Software

Para realizar el análisis debemos instalar el programa R statistical software que podemos descargar desde la página de R Project (https://www.r-project.org/). Para facilitar su uso podemos descargar la interfaz RStudio desde su página web (https://www.rstudio.com/).

A la hora de realizar este análisis necesitaremos funciones que no están disponibles en la instalación básica de R, por lo que he instalado algunas librerías. Los paquetes estándar se descargan desde el repositorio CRAN con la función install.packages, mientras los paquetes de Bioconductor se descargan con la función install() del paquete BiocManager.

3.3. Procedimiento general de análisis

El orden que vamos a seguir para realizar el análisis es el siguiente:

- Selección y preparación de los datos
- Preprocesado de los datos: filtraje y normalización
- Identificación de genes diferencialmente expresados
- Anotación de los resultados
- Eliminación de efectos de los lotes
- Comparación entre distintas comparaciones
- Análisis de significación biológica de los resultados obtenidos

3.3.1. Selección y preparación de los datos

De las 292 muestras analizadas, debemos seleccionar 30 (10 de cada grupo) para realizar el análisis. Para ello, he optado por hacer esta selección de forma manual en Excel, ya que conocemos que los *Sample_Name* del archivo *targets* se corresponden con las columnas del archivo *counts*. En primer lugar, en *targets* he seleccionado las 10 primeras muestras de los grupos SFI y ELI y las primeras 10 muestras impares del grupo NIT, que es el más numeroso, y he eliminado el resto de las muestras. Después, en *counts*, he seleccionado las columnas que correspondían con las filas seleccionadas en *targets* y he eliminado el resto.

Posteriormente, eliminamos el punto y el número que va detrás de él en la primera columna del archivo *counts* para evitar problemas durante el análisis. Para ello, utilizamos la herramienta Buscar y Reemplazar de Excel.

Una vez que estos archivos están listos para ser leídos, preparamos el entorno y los leemos.

```
setwd("C:/Users/monic/Desktop/PEC2 ADO")
counts <- read.csv(file="counts.csv",sep=";",row.names = 1)
targets <- read.csv(file="targets.csv",sep=",")
```

Para preparar el *data object* que vamos a utilizar en el análisis, necesitamos una tabla con los counts de los fragmentos, que en nuestro caso va a ser *counts* y una tabla con información sobre las muestras, que es *targets*.

Para construir el objeto DESeqDataSet, hacemos lo siguiente:

```
library("DESeq2")
ddsMat <- DESeqDataSetFromMatrix(countData = counts,
                                 colData = targets,
                                 design = ~ Group)

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables
in
## design formula are characters, converting to factors
```

Una vez que hemos creado el objeto DESeqDataSet, preprocesamos los datos del estudio.

3.3.2. Preprocesado de los datos: filtraje y normalización

Para realizar el análisis, en primer lugar, haremos transformaciones de los conteos para explorar visualmente las relaciones existentes. Nuestra matriz *counts* contiene muchas filas con solo ceros. Para reducir el tamaño del objeto y aumentar la velocidad de ejecución de las funciones, podemos eliminar las filas que no tienen o casi no tienen información sobre la cantidad de expresión génica. Para ello, eliminamos los registros que no tienen recuentos o solo un recuento único en todas las muestras.

```
nrow(ddsMat)
```

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

```
dds <- ddsMat[ rowSums(counts(ddsMat)) > 1, ]
nrow(dds)
```

```
## [1] 43191
```

Al eliminar estos registros, pasamos de 56202 a 43191.

En general, los métodos estadísticos para el análisis de datos multidimensionales funcionan mejor con datos que tienen el mismo rango de varianza en diferentes rangos de los valores medios. Sin embargo, para los recuentos de RNA-seq, la varianza esperada crece con la media. Con el paquete DESeq2 tenemos dos transformaciones posibles para los datos de conteo que estabilizan la varianza a través de la media: el VST y el rlog. El VST es mucho más rápido de calcular y es menos sensible a los valores atípicos de conteo alto que el rlog.

```
vsd <- vst(dds, blind = FALSE)
head(assay(vsd), 3)
```

```
##                 GTEX.111CU.0226.SM.5GZXC GTEX.111VG.0526.SM.5N9BW
## ENSG00000223972                 4.667822                 4.742380
## ENSG00000227232                 8.571969                 9.006843
## ENSG00000243485                 4.433999                 4.092421
##                 GTEX.111YS.0726.SM.5GZY8 GTEX.1128S.0126.SM.5H12S
## ENSG00000223972                 4.771657                 4.757630
## ENSG00000227232                 8.835563                 9.443518
## ENSG00000243485                 4.275626                 3.685196
##                 GTEX.117XS.0526.SM.5987Q GTEX.117YW.0126.SM.5EGGN
## ENSG00000223972                 4.829207                 4.889587
## ENSG00000227232                 9.338791                 9.411140
## ENSG00000243485                 4.127278                 3.685196
##                 GTEX.1192W.0126.SM.5EGGS GTEX.11DXX.0226.SM.5P9HL
## ENSG00000223972                 4.765114                 4.549981
## ENSG00000227232                 9.329854                 9.196424
## ENSG00000243485                 4.101517                 4.016243
##                 GTEX.11DXY.0426.SM.5H12R GTEX.11DYG.0826.SM.5N9GH
## ENSG00000223972                 4.708410                 4.574344
## ENSG00000227232                 9.376019                 9.477219
## ENSG00000243485                 3.685196                 3.685196
```

```
##                 GTEX.11EI6.0726.SM.59866 GTEX.11EMC.0226.SM.5EGLP
## ENSG00000223972                 4.685132                 4.621621
## ENSG00000227232                 9.214604                 9.356810
## ENSG00000243485                 4.227253                 3.685196
##                 GTEX.11EQ8.0826.SM.5N9FG GTEX.11EQ9.0626.SM.5A5K1
## ENSG00000223972                 4.670706                 4.598350
## ENSG00000227232                 9.532109                 9.001445
## ENSG00000243485                 4.063881                 4.380313
##                 GTEX.11GS4.0826.SM.5986J GTEX.11I78.0526.SM.5986A
## ENSG00000223972                 4.738481                 4.743577
## ENSG00000227232                 9.157622                 9.102602
## ENSG00000243485                 4.090863                 3.685196
##                 GTEX.11NV4.0626.SM.5N9BR GTEX.11O72.2326.SM.5BC7H
## ENSG00000223972                 4.674798                 4.661108
## ENSG00000227232                10.223121                 9.175723
## ENSG00000243485                 4.065504                 4.213882
##                 GTEX.11TUW.0226.SM.5LU8X GTEX.11XUK.0226.SM.5EQLW
## ENSG00000223972                 4.808316                 4.745129
## ENSG00000227232                 9.569420                 8.845502
## ENSG00000243485                 3.685196                 3.685196
##                 GTEX.1211K.0726.SM.5FQUW GTEX.12584.0826.SM.5FQSK
## ENSG00000223972                 4.768241                 4.708028
## ENSG00000227232                 8.929773                10.036466
## ENSG00000243485                 4.102770                 4.240020
##                 GTEX.12BJ1.0426.SM.5FQSO GTEX.13NZ9.1126.SM.5MR37
## ENSG00000223972                 4.760041                 4.577242
## ENSG00000227232                 8.855723                 9.554943
## ENSG00000243485                 4.269111                 4.026955
##                 GTEX.13QJC.0826.SM.5RQKC GTEX.14ABY.0926.SM.5Q5DY
## ENSG00000223972                 4.746262                 4.611895
## ENSG00000227232                 9.786660                 9.307326
## ENSG00000243485                 4.093973                 4.186570
##                 GTEX.14AS3.0226.SM.5Q5B6 GTEX.14BMU.0226.SM.5S2QA
## ENSG00000223972                 4.770036                 4.802685
## ENSG00000227232                 9.866864                 9.010023
## ENSG00000243485                 4.103489                 3.685196
##                 GTEX.PLZ4.1226.SM.2I5FE GTEX.R55G.0726.SM.2TC6J
## ENSG00000223972                4.606721                5.465706
## ENSG00000227232                8.660304                8.815413
## ENSG00000243485                4.038567                4.394529
```

```
colData(vsd)
```

```
## DataFrame with 30 rows and 10 columns
##                            Experiment  SRA_Sample             Sample_Name
##                           <character> <character>             <character>
## GTEX.111CU.0226.SM.5GZXC   SRX567480   SRS626942 GTEX-111CU-0226-SM-5GZXC
## GTEX.111VG.0526.SM.5N9BW   SRX563960   SRS625636 GTEX-111VG-0526-SM-5N9BW
## GTEX.111YS.0726.SM.5GZY8   SRX564185   SRS625665 GTEX-111YS-0726-SM-5GZY8
## GTEX.1128S.0126.SM.5H12S   SRX561718   SRS625313 GTEX-1128S-0126-SM-5H12S
```

```
## GTEX.117XS.0526.SM.5987Q   SRX634479   SRS648886 GTEX-117XS-0526-SM-5987Q
## ...                            ...         ...                       ...
## GTEX.14ABY.0926.SM.5Q5DY   SRX575932   SRS629299 GTEX-14ABY-0926-SM-5Q5DY
## GTEX.14AS3.0226.SM.5Q5B6   SRX607358   SRS639491 GTEX-14AS3-0226-SM-5Q5B6
## GTEX.14BMU.0226.SM.5S2QA   SRX568916   SRS627158 GTEX-14BMU-0226-SM-5S2QA
## GTEX.PLZ4.1226.SM.2I5FE    SRX199272   SRS333099  GTEX-PLZ4-1226-SM-2I5FE
## GTEX.R55G.0726.SM.2TC6J    SRX204036   SRS374975  GTEX-R55G-0726-SM-2TC6J
##                          Grupo_analisis   body_site
molecular_data_type
##                              <integer> <character>
<character>
## GTEX.111CU.0226.SM.5GZXC             1     Thyroid Allele-Specific
Expression
## GTEX.111VG.0526.SM.5N9BW             3     Thyroid             RNA Seq
(NGS)
## GTEX.111YS.0726.SM.5GZY8             1     Thyroid Allele-Specific
Expression
## GTEX.1128S.0126.SM.5H12S             1     Thyroid Allele-Specific
Expression
## GTEX.117XS.0526.SM.5987Q             1     Thyroid Allele-Specific
Expression
## ...                                ...         ...
...
## GTEX.14ABY.0926.SM.5Q5DY             3     Thyroid Allele-Specific
Expression
## GTEX.14AS3.0226.SM.5Q5B6             3     Thyroid             RNA Seq
(NGS)
## GTEX.14BMU.0226.SM.5S2QA             3     Thyroid Allele-Specific
Expression
## GTEX.PLZ4.1226.SM.2I5FE              3     Thyroid             RNA Seq
(NGS)
## GTEX.R55G.0726.SM.2TC6J              3     Thyroid             RNA Seq
(NGS)
##                              sex    Group   ShortName sizeFactor
##                        <character> <factor> <character>  <numeric>
## GTEX.111CU.0226.SM.5GZXC       male      NIT   111CU_NIT   1.128734
## GTEX.111VG.0526.SM.5N9BW       male      ELI   111VG_ELI   0.969268
## GTEX.111YS.0726.SM.5GZY8       male      NIT   111YS_NIT   0.915451
## GTEX.1128S.0126.SM.5H12S     female      NIT   1128S_NIT   0.940685
## GTEX.117XS.0526.SM.5987Q       male      NIT   117XS_NIT   0.821473
## ...                             ...      ...         ...        ...
## GTEX.14ABY.0926.SM.5Q5DY       male      ELI   14ABY_ELI   1.274483
## GTEX.14AS3.0226.SM.5Q5B6     female      ELI   14AS3_ELI   0.918318
## GTEX.14BMU.0226.SM.5S2QA     female      ELI   14BMU_ELI   0.862977
## GTEX.PLZ4.1226.SM.2I5FE      female      ELI   PLZ4-_ELI   1.289324
## GTEX.R55G.0726.SM.2TC6J      female      ELI   R55G-_ELI   0.315183
```

Especificamos blind = FALSE para que las diferencias causadas por las variables en el diseño no contribuyan a la tendencia esperada de varianza media del experimento.

```
rld <- rlog(dds, blind = FALSE)

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

head(assay(rld), 3)
```

```
##                 GTEX.111CU.0226.SM.5GZXC GTEX.111VG.0526.SM.5N9BW
## ENSG00000223972                2.7471214               2.82990062
## ENSG00000227232                8.6779356               9.00765308
## ENSG00000243485                0.3188247               0.00848186
##                 GTEX.111YS.0726.SM.5GZY8 GTEX.1128S.0126.SM.5H12S
## ENSG00000223972                2.8617660                 2.846543
## ENSG00000227232                8.8775266                 9.342058
## ENSG00000243485                0.1584116                -0.149579
##                 GTEX.117XS.0526.SM.5987Q GTEX.117YW.0126.SM.5EGGN
## ENSG00000223972               2.92337310                2.9865768
## ENSG00000227232               9.26151573                9.3170681
## ENSG00000243485               0.03350278               -0.1399076
##                 GTEX.1192W.0126.SM.5EGGS GTEX.11DXX.0226.SM.5P9HL
## ENSG00000223972               2.85467488               2.61147429
## ENSG00000227232               9.25467218               9.15235006
## ENSG00000243485               0.01503769              -0.04656893
##                 GTEX.11DXY.0426.SM.5H12R GTEX.11DYG.0826.SM.5N9GH
## ENSG00000223972                2.7924762                2.6399997
## ENSG00000227232                9.2901543                9.3681325
## ENSG00000243485               -0.1531724               -0.1628009
##                 GTEX.11EI6.0726.SM.59866 GTEX.11EMC.0226.SM.5EGLP
## ENSG00000223972                2.7665510                2.6946413
## ENSG00000227232                9.1663041                9.2754164
## ENSG00000243485                0.1190663               -0.1594416
##                 GTEX.11EQ8.0826.SM.5N9FG GTEX.11EQ9.0626.SM.5A5K1
## ENSG00000223972                2.7503683                 2.667863
## ENSG00000227232                9.4104073                 9.003378
## ENSG00000243485               -0.0121606                 0.269939
##                 GTEX.11GS4.0826.SM.5986J GTEX.11I78.0526.SM.5986A
## ENSG00000223972              2.825629121                2.8312102
## ENSG00000227232              9.122718495                9.0806836
## ENSG00000243485              0.007357343               -0.1506068
##                 GTEX.11NV4.0626.SM.5N9BR GTEX.11O72.2326.SM.5BC7H
## ENSG00000223972               2.75496691                2.7395517
## ENSG00000227232               9.94792643                9.1365329
## ENSG00000243485              -0.01098461                0.1081253
##                 GTEX.11TUW.0226.SM.5LU8X GTEX.11XUK.0226.SM.5EQLW
## ENSG00000223972                2.9011654                2.8329079
## ENSG00000227232                9.4390845                8.8850053
## ENSG00000243485               -0.1458649               -0.1504933
##                 GTEX.1211K.0726.SM.5FQUW GTEX.12584.0826.SM.5FQSK
## ENSG00000223972               2.85806602                2.7920526
## ENSG00000227232               8.94904479                9.8018747
```

```
## ENSG00000243485               0.01593947                0.1294908
##                  GTEX.12BJ1.0426.SM.5FQSO GTEX.13NZ9.1126.SM.5MR37
## ENSG00000223972                2.8491658               2.64337536
## ENSG00000227232                8.8927920               9.42810989
## ENSG00000243485                0.1531403              -0.03886429
##                  GTEX.13QJC.0826.SM.5RQKC GTEX.14ABY.0926.SM.5Q5DY
## ENSG00000223972              2.834145901               2.68347707
## ENSG00000227232              9.607351185               9.23740368
## ENSG00000243485              0.009601189               0.08573122
##                  GTEX.14AS3.0226.SM.5Q5B6 GTEX.14BMU.0226.SM.5S2QA
## ENSG00000223972               2.86001068                2.8951500
## ENSG00000227232               9.66962966                9.0101498
## ENSG00000243485               0.01645707               -0.1462779
##                  GTEX.PLZ4.1226.SM.2I5FE GTEX.R55G.0726.SM.2TC6J
## ENSG00000223972              2.67752239               3.5268739
## ENSG00000227232              8.74439756               8.8646375
## ENSG00000243485             -0.03048287               0.1799795
```

Para ver el efecto de la transformación, hacemos una gráfica comparando la primera muestra frente a la segunda, primero simplemente usando la función log2 (después de agregar 1, para evitar tomar el registro de cero), y los valores transformados con VST y rlog.

En el caso de log2, en primer lugar, debemos estimar los factores de tamaño para tener en cuenta la profundidad de secuenciación y luego especificar normalized=TRUE. La corrección de la profundidad de secuencia se realiza automáticamente para vst y rlog.

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

dds <- estimateSizeFactors(dds)

df <- bind_rows(
  as_data_frame(log2(counts(dds, normalized=TRUE)[, 1:2]+1)) %>%
         mutate(transformation = "log2(x + 1)"),
  as_data_frame(assay(vsd)[, 1:2]) %>% mutate(transformation = "vst"),
  as_data_frame(assay(rld)[, 1:2]) %>% mutate(transformation = "rlog"))

## Warning: `as_data_frame()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.

colnames(df)[1:2] <- c("x", "y")

ggplot(df, aes(x = x, y = y)) + geom_hex(bins = 80) +
  coord_fixed() + facet_grid( . ~ transformation)
```

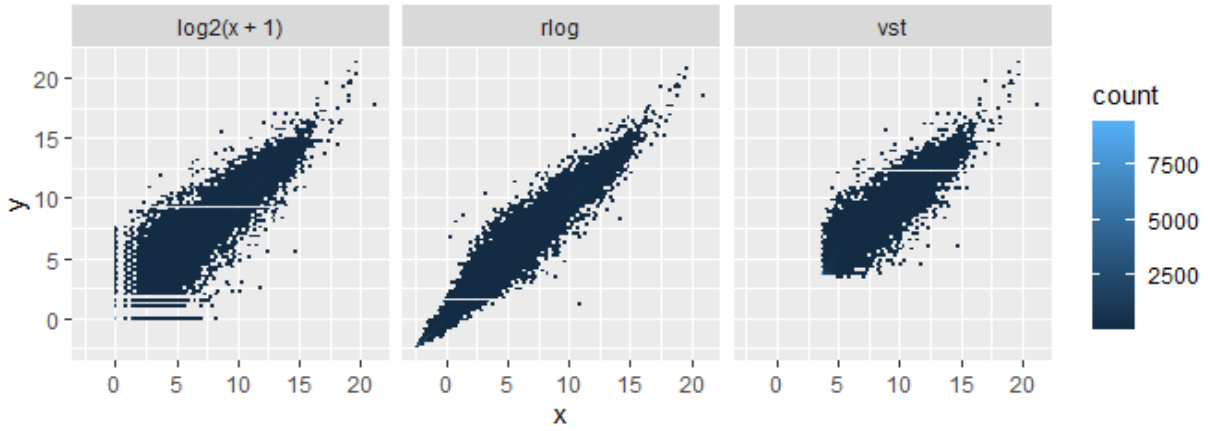


Gráfico 1. Diagramas de dispersión utilizando la transformación log2 de recuentos normalizados (izquierda), rlog (centro) y VST (derecha). Mientras que los counts del rlog está aproximadamente en la misma escala que el log2, el VST tiene un desplazamiento hacia arriba para los valores más pequeños. Podemos ver cómo los genes con recuentos bajos (esquina inferior izquierda) parecen ser excesivamente variables en la escala logarítmica ordinaria, mientras que VST y rlog comprimen las diferencias para los genes de recuento bajo, para los cuales los datos proporcionan poca información sobre la expresión diferencial.

Un paso útil en un análisis de RNA-seq suele ser 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.

```
sampleDists <- dist(t(assay(vsd)))
sampleDists
```

```
##                          GTEX.111CU.0226.SM.5GZXC GTEX.111VG.0526.SM.5N9BW
## GTEX.111VG.0526.SM.5N9BW                165.21921
## GTEX.111YS.0726.SM.5GZY8                112.76755                153.99089
## GTEX.1128S.0126.SM.5H12S                115.86245                148.52906
## GTEX.117XS.0526.SM.5987Q                134.50108                140.77062
## GTEX.117YW.0126.SM.5EGGN                150.31007                147.35927
## GTEX.1192W.0126.SM.5EGGS                140.52121                146.41575
## GTEX.11DXX.0226.SM.5P9HL                102.29216                160.34577
## GTEX.11DXY.0426.SM.5H12R                133.77405                134.00313
## GTEX.11DYG.0826.SM.5N9GH                127.11697                154.01352
## GTEX.11EI6.0726.SM.59866                129.53220                141.13136
## GTEX.11EMC.0226.SM.5EGLP                130.59029                140.25837
## GTEX.11EQ8.0826.SM.5N9FG                161.55944                150.06422
## GTEX.11EQ9.0626.SM.5A5K1                 96.37599                152.97819
## GTEX.11GS4.0826.SM.5986J                128.68221                148.34797
## GTEX.11I78.0526.SM.5986A                103.63997                157.46684


… continuación cortada
```

Para visualizar estas distancias, creamos un mapa de calor con la función *pheatmap*. Para trazar la matriz de distancia entre muestras con las filas y columnas ordenadas por las distancias en nuestra matriz de distancia, proporcionamos manualmente SampleDists al argumento clustering_distance de la función *pheatmap*.

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

sampleDistMatrix <- as.matrix( sampleDists )
rownames(sampleDistMatrix) <- paste( vsd$dex, vsd$cell, sep = " - " )
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)
```

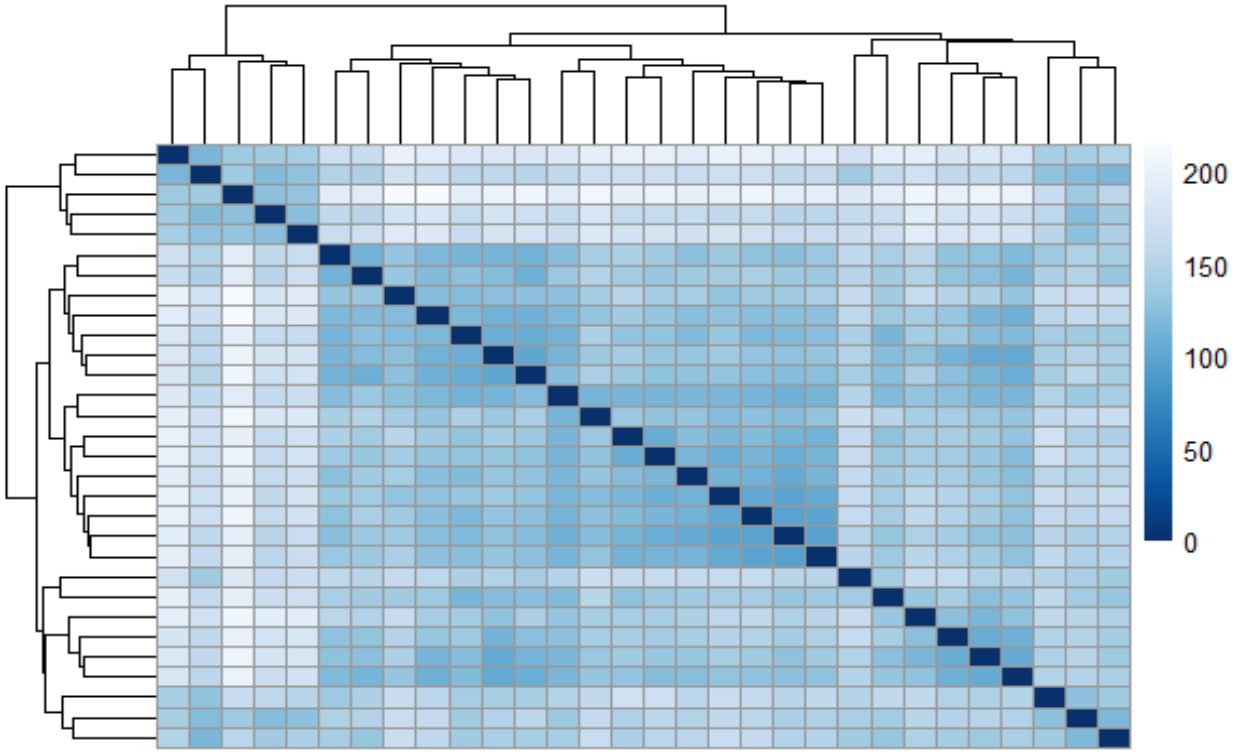


Gráfico 2. Mapa de calor de distancias entre muestras utilizando los valores transformados por rlog.

A continuación, realizamos un análisis de las componentes principales.

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

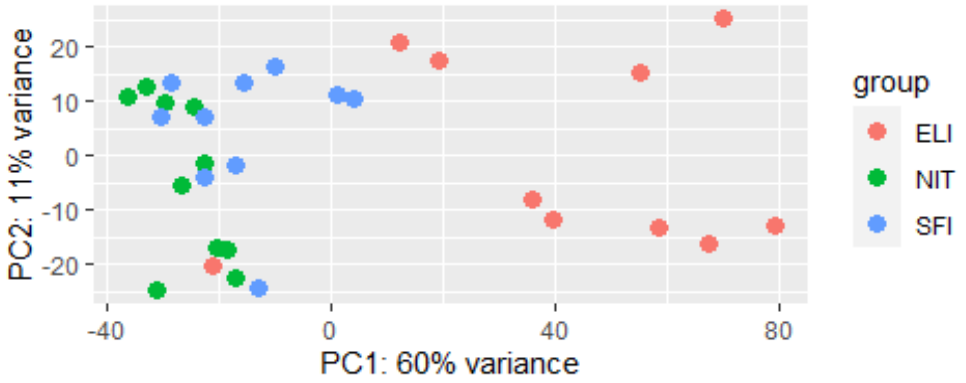


Gráfico 3. Análisis de componentes principales. Las 10 muestras de cada grupo están representadas del mismo color. Observamos que las muestras del grupo ELI se agrupan mayoritariamente a la derecha del gráfico mientras las de los grupos NIT y SFI se sitúan a la izquierda. La primera componente principal tiene una influencia muy superior a la segunda (60% vs 11% de la variación).

Para completar el análisis exploratorio, realizamos un gráfico de MDS.

```
mds <- as.data.frame(colData(vsd))  %>%
         cbind(cmdscale(sampleDistMatrix))
ggplot(mds, aes(x = `1`, y = `2`, color = Group, shape = sex)) +
  geom_point(size = 3) + coord_fixed()
```

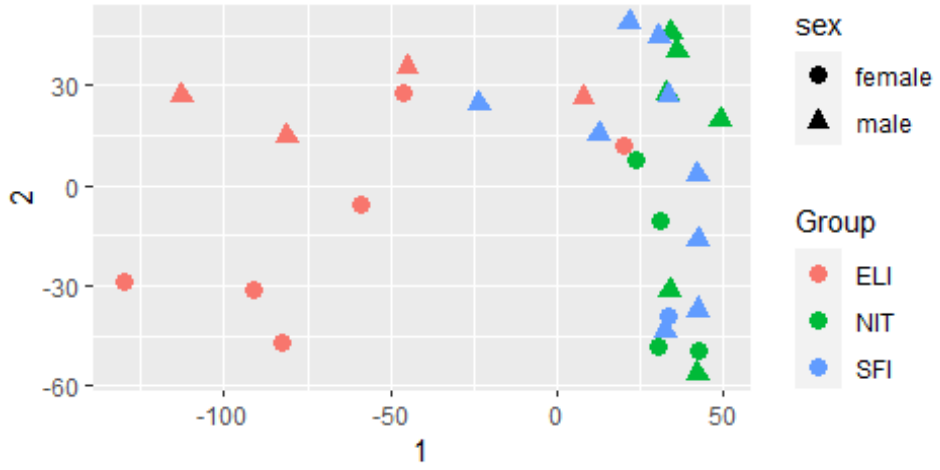


Gráfico 4. MDS. Las muestras de cada grupo se representan de un color y, también, se representan con distinta forma aquellas que corresponden a hombres (triángulo) y mujeres (círculo).

3.3.3. Identificación de genes diferencialmente expresados

Para identificar los genes diferencialmente expresados, utilizaremos el objeto DESeqDataSet creado previamente y lo llamaremos con la función DESeq.

```
dds <- DESeq(dds, parallel =TRUE)

## using pre-existing size factors

## estimating dispersions

## gene-wise dispersion estimates: 2 workers

## mean-dispersion relationship

## final dispersion estimates, fitting model and testing: 2 workers

## -- replacing outliers and refitting for 279 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

En el DESeqDataSet que nos devuelve tenemos todos los parámetros ajustados y podemos extraer aquellos que necesitemos para cada análisis. En este experimento contamos con 3 grupos de muestras: *not infiltrated tissues* (NIT), *small focal inflitrates* (SFI) y *extensive lymphoid infiltrates* (ELI). Por ello, realizaremos 3 comparaciones: NIT-SFI, NIT-ELI y SFI-ELI.

Comparación NIT-SFI

A continuación, extraemos los resultados que necesitamos para hacer una comparación, indicando que compare las muestras con NIT y SFI en la variable Group.

```
res <- results(dds, contrast=c("Group","NIT","SFI"))
res

## log2 fold change (MLE): Group NIT vs SFI
## Wald test p-value: Group NIT vs SFI
## DataFrame with 43191 rows and 6 columns
##                     baseMean log2FoldChange     lfcSE       stat    pvalue
##                    <numeric>      <numeric> <numeric>  <numeric> <numeric>
## ENSG00000223972    7.318224      -0.106453  0.294222  -0.361812  0.717492
## ENSG00000227232  613.726874      -0.141465  0.193877  -0.729665  0.465595
## ENSG00000243485    1.089936      -0.270180  0.785312  -0.344042  0.730815
## ENSG00000237613    1.070880       0.796808  1.099133   0.724942  0.468488
## ENSG00000268020    0.489658      -0.946868  1.410579  -0.671262  0.502054
## ...                     ...            ...       ...        ...       ...
## ENSG00000198695 5.99527e+04       0.339903  0.714078  0.4760022  0.634073
```

```
## ENSG00000210194 2.36582e+01       0.425349  0.992825  0.4284234  0.668343
## ENSG00000198727 4.14582e+05       0.110952  0.355543  0.3120647  0.754991
## ENSG00000210195 6.52466e-01      -0.951350  1.070130 -0.8890047  0.374001
## ENSG00000210196 1.86903e+00       0.089186  0.922505  0.0966781  0.922982
##                      padj
##                 <numeric>
## ENSG00000223972        NA
## ENSG00000227232  0.999716
## ENSG00000243485        NA
## ENSG00000237613        NA
## ENSG00000268020        NA
## ...                   ...
## ENSG00000198695  0.999716
## ENSG00000210194  0.999716
## ENSG00000198727  0.999716
## ENSG00000210195        NA
## ENSG00000210196        NA
```

Al ser un objeto DataFrame, lleva metadatos con información sobre el significado de las columnas:

```
mcols(res, 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 SFI
## lfcSE               results          standard error: Group NIT vs SFI
## stat                results          Wald statistic: Group NIT vs SFI
## pvalue              results       Wald test p-value: Group NIT vs SFI
## padj                results                        BH adjusted p-values
```

Dónde:

- *baseMean* es el promedio de los *counts* normalizados dividido entre el tamaño, tomados sobre todas las muestras en el DESeqDataSet.
- *log2FoldChange* es la estimación del tamaño del efecto. Nos dice cuánto parece haber cambiado la expresión del gen entre los dos grupos analizados.
- *lfcSE* es la estimación de error estándar.
- *padj* son los p-valor ajustados por el método de Benjamini-Hochbergtal.

También podemos resumir los resultados con la siguiente línea de código:

```
summary(res)

##
## out of 43190 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 2, 0.0046%
## LFC < 0 (down)     : 39, 0.09%
## outliers [1]       : 0, 0%
## low counts [2]     : 22610, 52%
## (mean count < 14)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

En este punto, 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(dds, alpha = 0.05)
table(res.05$padj < 0.05)

##
## FALSE  TRUE
## 26158  3634
```

Ahora, elevamos el umbral:

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

##
## FALSE  TRUE
## 28239   716
```

Vemos que el número de genes diferencialmente expresados en los dos grupos desciende de 3634 a 716 al ser más estrictos con el umbral.

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.

Para calcular cuántos genes cumplen este criterio, hacemos lo siguiente:

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

## [1] 41
```

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.

```
resSig <- subset(res, padj < 0.1)
head(resSig[ order(resSig$log2FoldChange), ])
```

```
## 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      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000242534   45.8368       -9.64550  1.540679  -6.26055 3.83616e-10
## ENSG00000211942   57.8043       -5.28095  0.989222  -5.33849 9.37229e-08
## ENSG00000211625  157.9147       -4.94683  1.049398  -4.71397 2.42936e-06
## ENSG00000241244   38.2830       -4.81216  1.271708  -3.78402 1.54318e-04
## ENSG00000224607   38.5159       -4.66303  1.186367  -3.93051 8.47649e-05
## ENSG00000240382  451.4738       -4.34686  1.041051  -4.17545 2.97394e-05
##                       padj
##                  <numeric>
## ENSG00000242534 7.89521e-06
## ENSG00000211942 6.42970e-04
## ENSG00000211625 7.09791e-03
## ENSG00000241244 8.35792e-02
## ENSG00000224607 5.62757e-02
## ENSG00000240382 2.78212e-02
```

...y los genes con una sobreexpresión más fuerte (*up-regulated*):

```
head(resSig[ order(resSig$log2FoldChange, decreasing = TRUE), ])
```

```
## 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      stat      pvalue
##                   <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000116690    305.360        2.19278  0.521415   4.20545 2.60567e-05
## ENSG00000198734    219.327        1.68831  0.436109   3.87130 1.08258e-04
## ENSG00000130766   1058.496       -1.47354  0.384900  -3.82838 1.28989e-04
## ENSG00000211677    982.692       -2.29453  0.604392  -3.79643 1.46796e-04
## ENSG00000132465   6108.137       -2.39408  0.613882  -3.89990 9.62334e-05
## ENSG000001432974  1127.173       -2.50327  0.654179  -3.82658 1.29936e-04
##                       padj
##                  <numeric>
## ENSG00000116690  0.0268137
## ENSG00000198734  0.0675168
## ENSG00000130766  0.0764058
## ENSG00000211677  0.0816545
## ENSG00000132465  0.0618931
## ENSG000001432974 0.0764058
```

A continuación, realizamos las dos comparaciones restantes entre grupos. Se muestran únicamente los comandos, ya que las explicaciones serían las mismas que en la comparación ya explicada.

Comparación NIT-ELI

```
res1 <- results(dds, contrast=c("Group","NIT","ELI"))
res1
```

```
## log2 fold change (MLE): Group NIT vs ELI
## Wald test p-value: Group NIT vs ELI
## DataFrame with 43191 rows and 6 columns
##                     baseMean log2FoldChange     lfcSE       stat    pvalue
##                    <numeric>      <numeric> <numeric>  <numeric> <numeric>
## ENSG00000223972    7.318224      -0.143189  0.294222  -0.486669  0.626493
## ENSG00000227232  613.726874      -0.196502  0.193967  -1.013069  0.311027
## ENSG00000243485    1.089936       0.179628  0.822085   0.218503  0.827037
## ENSG00000237613    1.070880       0.818001  1.098570   0.744605  0.456510
## ENSG00000268020    0.489658      -1.352379  1.380135  -0.979889  0.327141
## ...                     ...            ...       ...        ...       ...
## ENSG00000198695 5.99527e+04      0.3827127  0.714078  0.5359538  0.591990
## ENSG00000210194 2.36582e+01      0.5912155  0.994204  0.5946623  0.552069
## ENSG00000198727 4.14582e+05      0.2871608  0.355543  0.8076677  0.419282
## ENSG00000210195 6.52466e-01      0.0975266  1.145842  0.0851135  0.932171
## ENSG00000210196 1.86903e+00     -0.9781233  0.880753 -1.1105532  0.266761
##                       padj
##                  <numeric>
## ENSG00000223972  0.820208
## ENSG00000227232  0.582807
## ENSG00000243485  0.928503
## ENSG00000237613  0.708734
## ENSG00000268020        NA
## ...                   ...
## ENSG00000198695  0.798361
## ENSG00000210194  0.772810
## ENSG00000198727  0.679274
## ENSG00000210195        NA
## ENSG00000210196  0.537474
```

```
mcols(res1, 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
```

```
summary(res1)
```

```
##
## out of 43190 with nonzero total read count
```

```
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1654, 3.8%
## LFC < 0 (down)     : 3681, 8.5%
## outliers [1]       : 0, 0%
## low counts [2]     : 11724, 27%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
res1.05 <- results(dds, alpha = 0.05)
table(res1.05$padj < 0.05)
```

```
##
## FALSE  TRUE
## 26158  3634
```

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

```
##
## FALSE  TRUE
## 28239   716
```

```
sum(res1.05$padj < 0.1, na.rm=TRUE)
```

```
## [1] 4828
```

Genes *down-regulated*:

```
resSig <- subset(res1, padj < 0.1)
head(resSig[ order(resSig$log2FoldChange), ])
```

```
## 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      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000211619   59.5263      -10.00076   1.30096  -7.68723 1.50350e-14
## ENSG00000222037  690.3915       -9.73724   1.02326  -9.51594 1.80086e-21
## ENSG00000242534   45.8368       -9.71727   1.54077  -6.30674 2.84966e-10
## ENSG00000253818   72.6830       -9.69324   1.60317  -6.04628 1.48230e-09
## ENSG00000211625  157.9147       -9.68406   1.04375  -9.27813 1.72485e-20
## ENSG00000235896   39.1231       -9.40643   1.27639  -7.36957 1.71177e-13
##                        padj
##                   <numeric>
## ENSG00000211619 2.16030e-12
## ENSG00000222037 1.08976e-18
## ENSG00000242534 1.86425e-08
## ENSG00000253818 8.46525e-08
## ENSG00000211625 8.35015e-18
## ENSG00000235896 2.04031e-11
```

Genes *up-regulated*:

```
head(resSig[ order(resSig$log2FoldChange, decreasing = TRUE), ])
```

```
## 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      stat      pvalue
##                    <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000110680  8021.19841       10.36824  1.481376   6.99906 2.57692e-12
## ENSG00000134443    82.94040        8.26272  1.943622   4.25120 2.12630e-05
## ENSG00000157005    22.92310        5.64353  1.320266   4.27454 1.91536e-05
## ENSG00000179914    66.31481        4.86271  0.874842   5.55839 2.72275e-08
## ENSG00000157219     1.96393        4.55570  1.733221   2.62846 8.57720e-03
## ENSG000001324373   14.39274        4.45535  0.916128   4.86324 1.15477e-06
##                         padj
##                    <numeric>
## ENSG00000110680  2.46468e-10
## ENSG00000134443  5.11531e-04
## ENSG00000157005  4.68252e-04
## ENSG00000179914  1.22395e-06
## ENSG00000157219  6.32674e-02
## ENSG000001324373 3.64466e-05
```

Comparación SFI-ELI

```
res2 <- results(dds, contrast=c("Group","SFI","ELI"))
res2
```

```
## log2 fold change (MLE): Group SFI vs ELI
## Wald test p-value: Group SFI vs ELI
## DataFrame with 43191 rows and 6 columns
##                     baseMean log2FoldChange     lfcSE       stat    pvalue
##                    <numeric>      <numeric> <numeric>  <numeric> <numeric>
## ENSG00000223972    7.318224     -0.0367355  0.294222 -0.1248565  0.900637
## ENSG00000227232  613.726874     -0.0550367  0.193940 -0.2837820  0.776577
## ENSG00000243485    1.089936      0.4498083  0.809685  0.5555352  0.578529
## ENSG00000237613    1.070880      0.0211933  1.154493  0.0183572  0.985354
## ENSG00000268020    0.489658     -0.4055117  1.372328 -0.2954918  0.767618
## ...                     ...            ...       ...        ...       ...
## ENSG00000198695 5.99527e+04      0.0428102  0.714079  0.0599516  0.952194
## ENSG00000210194 2.36582e+01      0.1658661  0.995681  0.1665857  0.867696
## ENSG00000198727 4.14582e+05      0.1762084  0.355543  0.4956030  0.620175
## ENSG00000210195 6.52466e-01      1.0488770  1.077300  0.9736160  0.330247
## ENSG00000210196 1.86903e+00     -1.0673093  0.891680 -1.1969649  0.231320
##                       padj
##                  <numeric>
## ENSG00000223972  0.959926
## ENSG00000227232  0.902620
## ENSG00000243485  0.796501
```

```
## ENSG00000237613  0.993594
## ENSG00000268020        NA
## ...                   ...
## ENSG00000198695  0.979814
## ENSG00000210194  0.945381
## ENSG00000198727  0.820765
## ENSG00000210195        NA
## ENSG00000210196  0.509097
```

```
mcols(res2, 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 SFI vs ELI
## lfcSE               results          standard error: Group SFI vs ELI
## stat                results          Wald statistic: Group SFI vs ELI
## pvalue              results       Wald test p-value: Group SFI vs ELI
## padj                results                       BH adjusted p-values
```

```
summary(res2)
```

```
##
## out of 43190 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1626, 3.8%
## LFC < 0 (down)     : 3186, 7.4%
## outliers [1]       : 0, 0%
## low counts [2]     : 12562, 29%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
res2.05 <- results(dds, alpha = 0.05)
table(res2.05$padj < 0.05)
```

```
##
## FALSE  TRUE
## 26158  3634
```

```
resLFC1.2 <- results(dds, lfcThreshold=1)
table(resLFC1.2$padj < 0.1)
```

```
##
## FALSE  TRUE
## 28239   716
```

```
sum(res2.05$padj < 0.1, na.rm=TRUE)
```

```
## [1] 4828
```

Genes *down-regulated*:

```
resSig <- subset(res2, padj < 0.1)
head(resSig[ order(resSig$log2FoldChange), ])
```

```
## 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      stat      pvalue
##                  <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000196092  376.3652       -8.34459   0.86925  -9.59976 8.01301e-22
## ENSG00000100721  314.6597       -8.01093   1.01271  -7.91042 2.56531e-15
## ENSG00000181617  207.0494       -7.84935   1.27433  -6.15958 7.29366e-10
## ENSG00000254029   14.9976       -7.63698   2.61546  -2.91994 3.50098e-03
## ENSG00000260303   18.1727       -7.62165   1.91534  -3.97927 6.91277e-05
## ENSG00000250850   11.8943       -7.58993   1.10005  -6.89962 5.21409e-12
##                       padj
##                  <numeric>
## ENSG00000196092 1.63620e-18
## ENSG00000100721 1.04764e-12
## ENSG00000181617 6.76963e-08
## ENSG00000254029 3.40201e-02
## ENSG00000260303 1.49317e-03
## ENSG00000250850 9.02273e-10
```

Genes *up-regulated*:

```
head(resSig[ order(resSig$log2FoldChange, decreasing = TRUE), ])
```

```
## 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      stat      pvalue
##                   <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000110680  8021.19841        8.14988   1.48139   5.50151 3.76547e-08
## ENSG00000134443    82.94040        6.45980   1.94434   3.32236 8.92602e-04
## ENSG000001268903    5.98935        6.11730   2.06290   2.96540 3.02294e-03
## ENSG00000230663     4.02900        4.26923   1.45713   2.92990 3.39074e-03
## ENSG00000204787     4.51693        4.26089   1.29440   3.29178 9.95564e-04
## ENSG00000197978     5.95125        4.25525   1.01102   4.20886 2.56666e-05
##                        padj
##                   <numeric>
## ENSG00000110680  2.10078e-06
## ENSG00000134443  1.20442e-02
## ENSG000001268903 3.05274e-02
## ENSG00000230663  3.31549e-02
## ENSG00000204787  1.31426e-02
## ENSG00000197978  6.47030e-04
```

3.3.4. Anotación de los resultados

Nuestra tabla de resultados hasta ahora solo contiene las ID de los genes de Ensembl, pero otros nombres alternativos son más útiles a la hora de realizar la interpretación. Los paquetes de anotación de Bioconductor ayudan a mapear varios esquemas de identificación entre sí:

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

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"
"ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"
"GENENAME"
## [11] "GO"           "GOALL"        "IPI"          "MAP"          "OMIM"
## [16] "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"         "PMID"
## [21] "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"       "UNIGENE"
## [26] "UNIPROT"
```

Podemos utilizar la función mapIds para añadir columnas a nuestra tabla de resultados. Proporcionamos los nombres de fila de nuestra tabla de resultados como una clave y especificamos que keytype = ENSEMBL. El argumento de la columna le dice a la función mapIds qué información queremos, y el argumento multiVals le dice a la función qué hacer si hay múltiples valores posibles para un solo valor de entrada. Aquí, le pedimos que nos devuelva el primero que encuentre en la base de datos. Para agregar el símbolo del gen y la identificación de Entrez, llamamos a mapIds dos veces.

```
res$symbol <- mapIds(org.Hs.eg.db,
                     keys=row.names(res),
                     column="SYMBOL",
                     keytype="ENSEMBL",
                     multiVals="first")
```

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

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

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

```
resOrdered <- res[order(res$pvalue),]
head(resOrdered)
```

```
## 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      stat      pvalue
```

```
##                   <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000242534     45.8368       -9.64550  1.540679  -6.26055 3.83616e-10
## ENSG00000244116    523.8542       -4.17390  0.755210  -5.52681 3.26098e-08
## ENSG00000211942     57.8043       -5.28095  0.989222  -5.33849 9.37229e-08
## ENSG00000211896 118694.9781       -3.58874  0.721527  -4.97381 6.56488e-07
## ENSG00000242371   1328.1033       -3.96038  0.821167  -4.82286 1.41511e-06
## ENSG00000239951   5307.1606       -3.03859  0.642369  -4.73029 2.24199e-06
##                        padj      symbol      entrez
##                   <numeric> <character> <character>
## ENSG00000242534 7.89521e-06          NA          NA
## ENSG00000244116 3.35571e-04          NA          NA
## ENSG00000211942 6.42970e-04          NA          NA
## ENSG00000211896 3.37780e-03          NA          NA
## ENSG00000242371 5.82490e-03          NA          NA
## ENSG00000239951 7.09791e-03          NA          NA
```

Repetimos el proceso en las otras dos comparaciones:

```
res1$symbol <- mapIds(org.Hs.eg.db,
                      keys=row.names(res1),
                      column="SYMBOL",
                      keytype="ENSEMBL",
                      multiVals="first")
```

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

```
res1$entrez <- mapIds(org.Hs.eg.db,
                      keys=row.names(res1),
                      column="ENTREZID",
                      keytype="ENSEMBL",
                      multiVals="first")
```

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

```
res1Ordered <- res1[order(res1$pvalue),]
head(res1Ordered)
```

```
## 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      stat      pvalue
##                   <numeric>      <numeric> <numeric> <numeric>   <numeric>
## ENSG00000211659     640.370       -8.68982  0.672388  -12.9238 3.30283e-38
## ENSG00000001704761  1098.334       -7.36781  0.611878  -12.0413 2.15493e-33
## ENSG00000239951     5307.161       -7.72518  0.642103  -12.0311 2.44005e-33
## ENSG00000211644     2108.216       -7.08294  0.613426  -11.5465 7.68577e-31
## ENSG00000001635340   716.172       -6.97479  0.607811  -11.4753 1.75641e-30
## ENSG00000211896  118694.978       -8.07000  0.721518  -11.1848 4.84253e-29
##                         padj      symbol      entrez
##                    <numeric> <character> <character>
## ENSG00000211659  1.03930e-33          NA          NA
## ENSG00000001704761 2.55936e-29          NA          NA
```

```
## ENSG00000239951  2.55936e-29          NA          NA
## ENSG00000211644  6.04620e-27          NA          NA
## ENSG000001635340 1.10538e-26          NA          NA
## ENSG00000211896  2.53967e-25          NA          NA
```

```
res2$symbol <- mapIds(org.Hs.eg.db,
                      keys=row.names(res2),
                      column="SYMBOL",
                      keytype="ENSEMBL",
                      multiVals="first")
```

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

```
res2$entrez <- mapIds(org.Hs.eg.db,
                      keys=row.names(res2),
                      column="ENTREZID",
                      keytype="ENSEMBL",
                      multiVals="first")
```

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

```
res2Ordered <- res2[order(res$pvalue),]
head(res2Ordered)
```

```
## 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       stat
pvalue
##                   <numeric>      <numeric> <numeric>  <numeric>
<numeric>
## ENSG00000242534     45.8368     -0.0717676  1.400310 -0.0512512 9.59125e-
01
## ENSG00000244116    523.8542     -3.7768751  0.732890 -5.1534002 2.55805e-
07
## ENSG00000211942     57.8043     -3.7250323  0.750865 -4.9609881 7.01355e-
07
## ENSG00000211896 118694.9781     -4.4812557  0.721419 -6.2117246 5.24062e-
10
## ENSG00000242371   1328.1033     -4.1041283  0.812589 -5.0506805 4.40239e-
07
## ENSG00000239951   5307.1606     -4.6865914  0.640245 -7.3199922 2.47985e-
13
##                        padj      symbol      entrez
##                   <numeric> <character> <character>
## ENSG00000242534 9.82674e-01          NA          NA
## ENSG00000244116 1.14047e-05          NA          NA
## ENSG00000211942 2.77543e-05          NA          NA
## ENSG00000211896 5.11194e-08          NA          NA
## ENSG00000242371 1.84461e-05          NA          NA
## ENSG00000239951 5.98074e-11          NA          NA
```

A continuación, guardamos los resultados obtenidos en archivos csv, que se crean en el *working directory*.

```
resOrdered_NIT_SFI <- as.data.frame(resOrdered)
write.csv(resOrdered, file = "results_NIT_SFI.csv")

resOrdered_NIT_ELI <- as.data.frame(res1Ordered)
write.csv(res1Ordered, file = "results_NIT_ELI.csv")

resOrdered_SFI_ELI <- as.data.frame(res2Ordered)
write.csv(res2Ordered, file = "results_SFI_ELI.csv")
```

En las visualizaciones de las primeras posiciones de la tabla no observemos ningún símbolo de los genes ni ningún código *entrez*. Esto puede deberse a que las filas de la tabla son transcritos en lugar de genes. Sin embargo, al abrir los archivos csv exportados, sí podemos observar que estos datos están completos en algunos registros.

3.3.5. Eliminación de efectos de los lotes

Podemos usar métodos estadísticos diseñados para detectar efectos de los lotes en RNA-seq con el paquete sva o RUVSeq y después agregarlos al diseño del DESeqDataSet. Primero usamos SVA para encontrar efectos de lote ocultos y luego RUV.

A continuación, obtenemos una matriz de recuentos normalizados para los cuales el recuento promedio en las muestras es mayor que 1. En este apartado, tratamos de detectar cualquier efecto oculto por lotes, suponiendo que no tenemos información previa sobre los grupos. Por lo tanto, utilizamos una matriz de modelo completo con la variable *Group*, y una matriz de modelo reducida o nula con solo un término de intercepción.

```
library("sva")

dat  <- counts(dds, normalized = TRUE)
idx  <- rowMeans(dat) > 1
dat  <- dat[idx, ]
mod  <- model.matrix(~ Group, colData(dds))
mod0 <- model.matrix(~   1, colData(dds))
svseq <- svaseq(dat, mod, mod0, n.sv = 2)
```

```
## Number of significant surrogate variables is:  2
## Iteration (out of 5 ):1  2  3  4  5
```

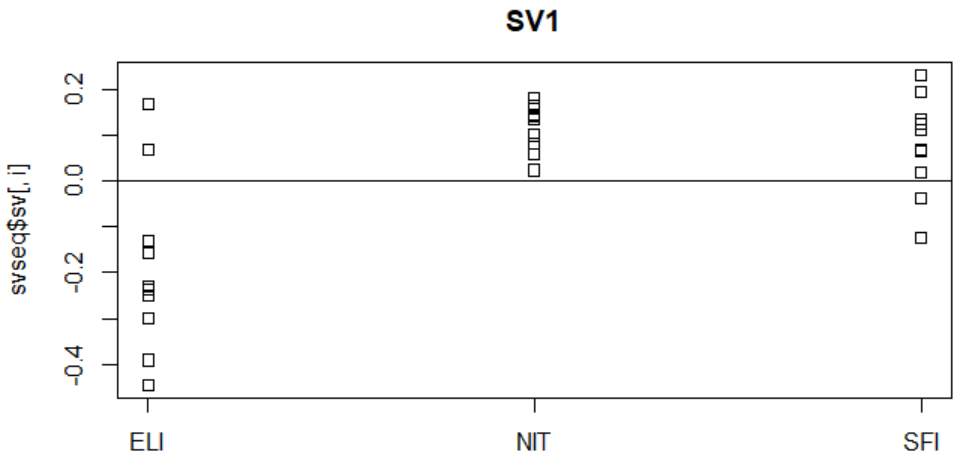
```
svseq$sv
```

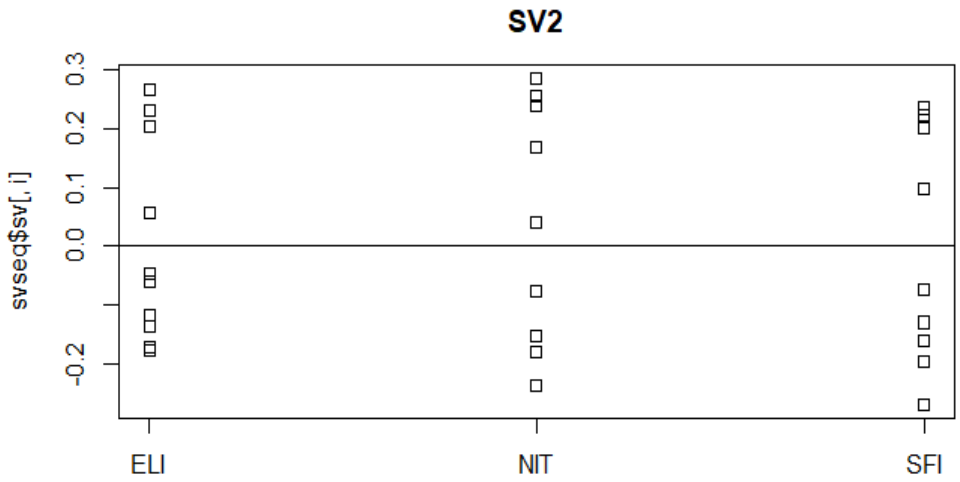
```
##              [,1]        [,2]
##  [1,]  0.18067344  0.28640367
##  [2,] -0.23670466 -0.17226953
##  [3,]  0.10032463  0.16785034
##  [4,]  0.05857486  0.04056339
##  [5,]  0.08335751 -0.23823527
```

```
##  [6,]  0.01809581 -0.27105922
##  [7,]  0.02430688 -0.23848689
##  [8,]  0.14070302  0.24002471
##  [9,]  0.06800603 -0.12995128
## [10,]  0.16357300 -0.15234409
## [11,]  0.14519809 -0.18008769
## [12,]  0.08212840 -0.07751139
## [13,] -0.12321029 -0.07429063
## [14,]  0.11247327  0.23538534
## [15,]  0.12605100 -0.16063357
## [16,]  0.13404651  0.25551445
## [17,] -0.39108832 -0.13711176
## [18,]  0.23243617 -0.07335665
## [19,]  0.06679541  0.09718903
## [20,] -0.44678376  0.20443468
## [21,]  0.13423889  0.22306671
## [22,] -0.03870764 -0.19687656
## [23,]  0.19627101  0.20032818
## [24,] -0.24976405 -0.06058297
## [25,] -0.23109224  0.05587452
## [26,]  0.16800287 -0.17817052
## [27,] -0.12945959 -0.11732273
## [28,] -0.15750313  0.26763883
## [29,]  0.06882619 -0.04632878
## [30,] -0.29976928  0.23034567
```

Como realmente sí conocemos los grupos, podemos comprobar si el método SVA logró recuperar estas variables.

```
par(mfrow = c(2, 1), mar = c(3,5,3,1))
for (i in 1:2) {
  stripchart(svseq$sv[, i] ~ dds$Group, vertical = TRUE, main = paste0("SV",
i))
  abline(h = 0)
 }
```





Gráficos 5 y 6. Análisis del efecto de los lotes con SVA.

Finalmente, para usar SVA para eliminar cualquier efecto sobre los recuentos de nuestras variables sustitutas, simplemente añadimos estas dos variables como columnas al DESeqDataSet y luego las agregamos al diseño:

```
ddssva <- dds
ddssva$SV1 <- svseq$sv[,1]
ddssva$SV2 <- svseq$sv[,2]
design(ddssva) <- ~ SV1 + SV2 + Group
```

Ahora ya podríamos obtener resultados controlando estas variables al utilizar DESeq con el nuevo diseño.

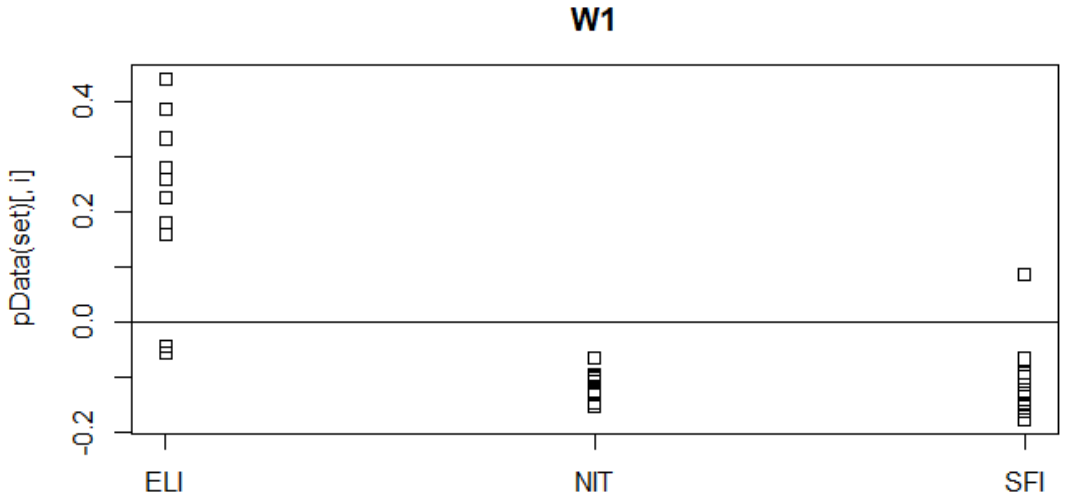
Además, podemos usar la función RUVg para estimar factores de variación no deseada, análogos a las variables sustitutas de SVA.

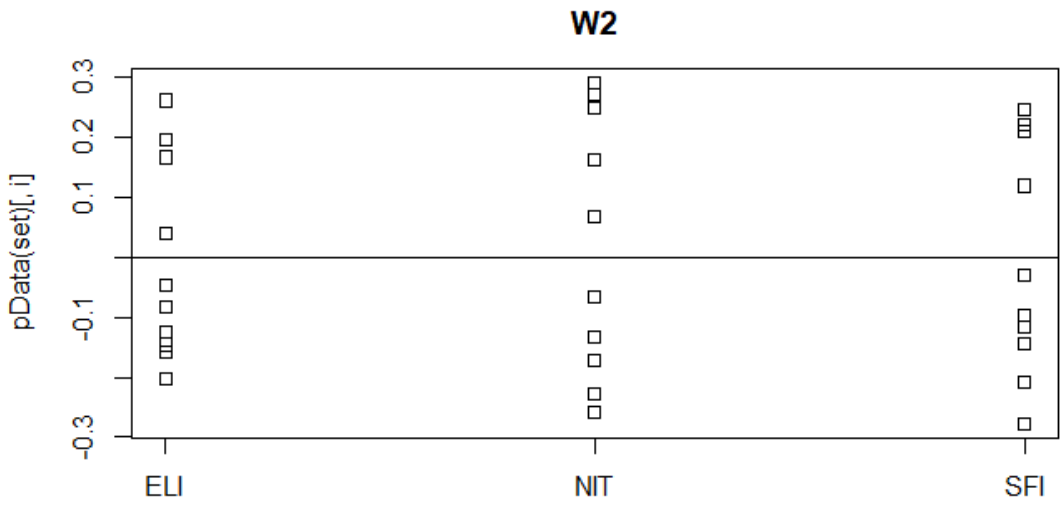
```
library("RUVSeq")

set <- newSeqExpressionSet(counts(dds))
idx  <- rowSums(counts(set) > 5) >= 2
set  <- set[idx, ]
set <- betweenLaneNormalization(set, which="upper")
not.sig <- rownames(res)[which(res$pvalue > .1)]
empirical <- rownames(set)[ rownames(set) %in% not.sig ]
set <- RUVg(set, empirical, k=2)
pData(set)

##                                  W_1         W_2
## GTEX.111CU.0226.SM.5GZXC -0.14502070  0.29042853
## GTEX.111VG.0526.SM.5N9BW  0.17934395 -0.20308986
## GTEX.111YS.0726.SM.5GZY8 -0.12657429  0.16221070
## GTEX.1128S.0126.SM.5H12S -0.09821277  0.06678032
## GTEX.117XS.0526.SM.5987Q -0.10582820 -0.22727096
## GTEX.117YW.0126.SM.5EGGN -0.09086053 -0.27852053
```

```
## GTEX.1192W.0126.SM.5EGGS -0.09455113 -0.25879536
## GTEX.11DXX.0226.SM.5P9HL -0.13043664  0.24796972
## GTEX.11DXY.0426.SM.5H12R -0.06575332 -0.11694077
## GTEX.11DYG.0826.SM.5N9GH -0.15251911 -0.13281789
## GTEX.11EI6.0726.SM.59866 -0.10063636 -0.17246758
## GTEX.11EMC.0226.SM.5EGLP -0.06363675 -0.06733891
## GTEX.11EQ8.0826.SM.5N9FG  0.08688819 -0.09661319
## GTEX.11EQ9.0626.SM.5A5K1 -0.11392627  0.22099853
## GTEX.11GS4.0826.SM.5986J -0.12912780 -0.14437443
## GTEX.11I78.0526.SM.5986A -0.09763117  0.27129828
## GTEX.11NV4.0626.SM.5N9BR  0.38423512 -0.15812327
## GTEX.11O72.2326.SM.5BC7H -0.16074462 -0.02985737
## GTEX.11TUW.0226.SM.5LU8X -0.14877202  0.11877431
## GTEX.11XUK.0226.SM.5EQLW  0.43997850  0.16577229
## GTEX.1211K.0726.SM.5FQUW -0.14496169  0.24455011
## GTEX.12584.0826.SM.5FQSK -0.09902392 -0.20814817
## GTEX.12BJ1.0426.SM.5FQSO -0.17674470  0.20891391
## GTEX.13NZ9.1126.SM.5MR37  0.27896621 -0.08380447
## GTEX.13QJC.0826.SM.5RQKC  0.22583299  0.03993550
## GTEX.14ABY.0926.SM.5Q5DY -0.05645032 -0.14559097
## GTEX.14AS3.0226.SM.5Q5B6  0.15866823 -0.12378919
## GTEX.14BMU.0226.SM.5S2QA  0.25782217  0.26052520
## GTEX.PLZ4.1226.SM.2I5FE  -0.04273820 -0.04634430
## GTEX.R55G.0726.SM.2TC6J   0.33241519  0.19572983
```

Y podemos representar los factores estimados por RUV:

```
par(mfrow = c(2, 1), mar = c(3,5,3,1))
for (i in 1:2) {
  stripchart(pData(set)[, i] ~ dds$Group, vertical = TRUE, main = paste0("W",
i))
  abline(h = 0)
 }
```





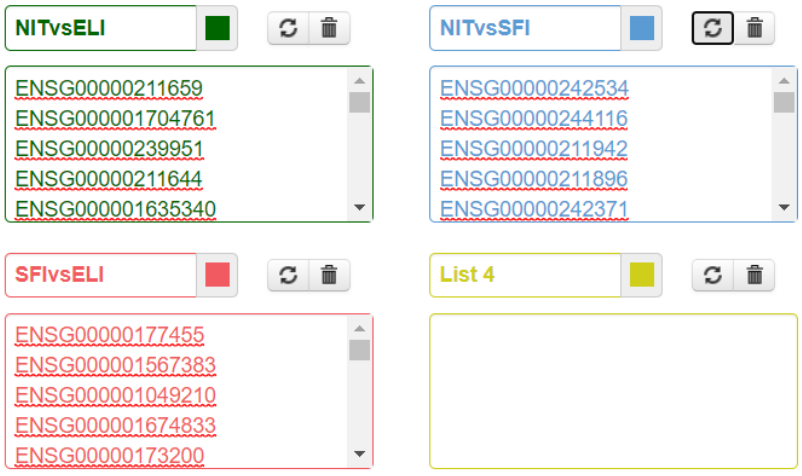
Gráficos 7 y 8. Análisis del efecto de los lotes con RUVg.

3.3.6. Comparación entre distintas comparaciones

En este punto del estudio es interesante conocer si algunos genes se han seleccionado como diferencialmente expresados en más de una comparación. Para ello, una de las formas más visuales de hacerlo es elaborar un diagrama de Venn, en el que se representa el número de genes que están diferencialmente expresados en una, dos o las tres comparaciones.

Al tener problemas para elaborar el diagrama con R, he decidido utilizar la herramienta web jvenn (Bardou, 2014), en la que se pueden representar diagramas de Venn de forma interactiva.

Para ello, he seleccionado de los archivos csv exportados previamente (punto 3.3.4.) los identificadores de aquellos transcritos que tienen un p-valor ajustado inferior a 0.15.



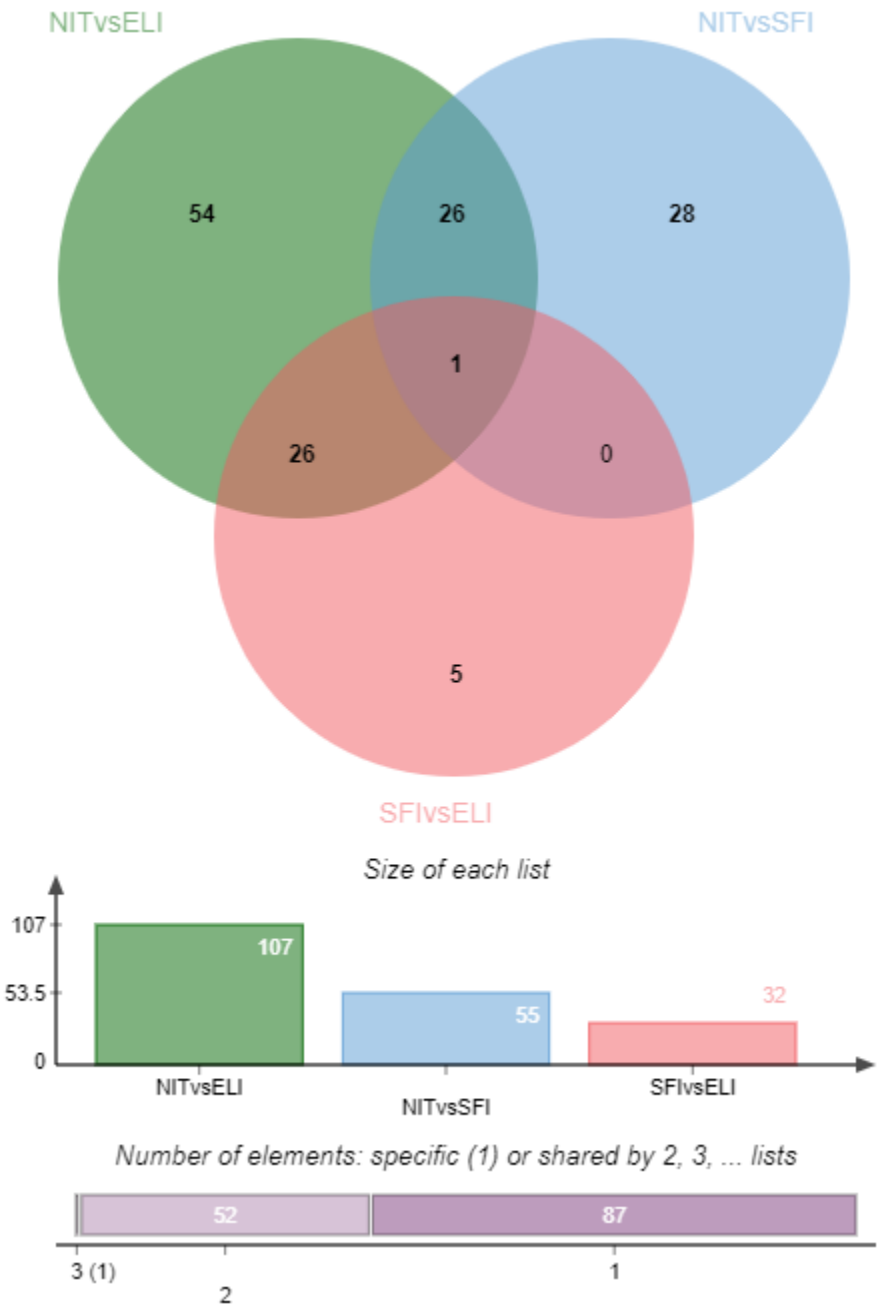


Gráfico 9. Diagrama de Venn en el que se muestra cuántos genes diferencialmente expresados son compartidos entre varias comparaciones.

Como podemos observar, las comparaciones que más genes comparten son las dos en las que está presente ELI y las dos en las que tenemos el grupo control (NIT), con 26 en cada caso. Vemos que solamente uno de los transcritos está diferencialmente expresado con un p-valor ajustado inferior a 0.15 en las tres comparaciones realizadas.

3.3.7. Análisis de significación biológica de los resultados obtenidos

En este apartado del análisis, el objetivo es conocer si hay rutas moleculares, funciones, procesos biológicos, etc. que estén relacionados con los genes que están diferencialmente expresados.

En este caso, en el paso de la anotación, solo en un 23% de los transcritos se han completado las columnas *symbol* y *entrez*, por lo que tratar de determinar cuáles son las rutas moleculares o procesos biológicos más relacionados con los genes diferencialmente expresados teniendo en cuenta sólo una pequeña parte de ellos no nos daría un resultado fiable.

En el apartado anterior, con la herramienta jvenn, hemos podido utilizar los identificadores ya que la herramienta busca que el código que le damos, sea del tipo que sea, esté en las tres listas que le proporcionamos, pero en este caso no sería suficiente.

4. Resultados

En este estudio hemos encontrado que el número de genes diferencialmente expresados es muy inferior en la comparación NIT-SFI con respecto a las comparaciones NIT-ELI y SFI-ELI. Este hecho, junto lo observado en los gráficos 3 y 4, en los que las muestras del grupo NIT y SFI se agrupaban mayoritariamente en un lado y las de ELI en otro, nos hace pensar que los resultados del tratamiento con pequeños infiltrados focales (SFI) son similares al grupo control sin infiltración (NIT), mientras que los infiltrados linfoides extensos (ELI) sí presentan diferencias con respecto a los otros dos grupos.

5. Discusión y conclusiones

La principal limitación que he encontrado a la hora de realizar este estudio ha sido la anotación de genes, ya que la mayoría no han sido detectados. Este hecho no nos ha permitido hacer un análisis de significación biológica que fuera realmente representativo.

En este análisis hemos analizado solo 30 muestras de las 292 que tiene el estudio original, ya que el objetivo no era tener unas conclusiones sino aprender a hacer el análisis y que éste se pudiera ejecutar de forma fluida. Sin embargo, al no conocer el estudio original, no podemos saber si nuestras conclusiones son las mismas.

6. Apéndice

En el repositorio de GitHub *https://github.com/monicahortal/PEC-2-An-lisis-de-Datos-micos.git* se puede encontrar el archivo rmd que contiene el código de este análisis y los archivos generados en el desarrollo del trabajo.

7. Referencias

Michael I. Love, Charlotte Soneson, Simon Anders, Vladislav Kim and Wolfgang Huber. RNA-seq workflow: gene-level exploratory analysis and differential expression. CSAMA 2017 version.

Philippe Bardou, Jérôme Mariette, Frédéric Escudié, Christophe Djemiel and Christophe Klopp. jvenn: an interactive Venn diagram viewer. BMC Bioinformatics 2014, 15:293 doi:10.1186/1471-2105-15-293.