Análisis de datos ómicos

PEC 2

Julián David Sánchez Bautista

Tabla de contenido:

1. Resumen

La glándula tiroides tiene diferentes funciones fundamentales en el organismo. Esta glándula es susceptible de presentar patología tanto inflamatoria como tumoral. Es conocido que el infiltrado inflamatorio, tiene implicaciones pronósticas en el curso de dichas enfermedades, por lo que es necesario definir la presencia de expresión génica diferencial, dependiendo de la extensión de la infiltración linfocitaria. En el análisis realizado se encontró expresión diferencial de 199 genes, sin embargo, en la anotación génica, se encontró solo un gen que tenía implicaciones diferenciales en el tipo de infiltrado. Este gen codifica la región variable de las cadenas pesadas de las inmunoglobulinas, por lo que se encuentra implicado en procesos fisiopatológicos relacionados con el funcionamiento de linfocitos B. Estas células tienen función claramente identificada en la fisiopatología de enfermedades autoinmunes, con algunas de las enfermedades inflamatorias de la glándula tiroides (por ejemplo, tiroiditis de Hashimoto).

2. Objetivos

2.1. Objetivos de la práctica

El objetivo de esta práctica es realizar el análisis de datos de ultrasecuenciación para determinar los genes diferencialmente expresados.

2.2. Objetivos del estudio

Los datos disponibles corresponden a muestras de tejido tiroideo. El objetivo del estudio fue determinar la expresión diferencial de genes en tres tipos de muestra de tejido tiroideo con diferencias en el infiltrado: sin infiltración del tejido (NIT), con pequeños focos de infiltrado (SFI) y con extenso infiltrado linfoide (ELI)

Así mismo se busca determinar la significancia biológica de los resultados encontrados

3. Materiales y métodos

3.1. Datos usados:

Para la realización de la práctica se cuenta con datos del repositorio GTEx, de un estudio de patología tiroidea, en donde se evaluó la expresión *RNA-seq* evaluando tres diferentes tipos de infiltración en tejido tiroideo: Not infiltrated tissues (NIT), Small focal infiltrates (SFI) y Extensive lymphoid infiltrates (ELI).

3.2. Pipeline para el análisis de los datos

3.2.1. Extracción de los datos

Los datos aportados para la ejecución de la PEC estaban en formato *.cvs.* El primer paso fue convertirlos a un formato Excel para poder manipularlos de una manera más sencilla. Posteriormente se escogió el directorio en el cual se realizará la actividad.

Posteriormente se cargan los datos de los dos archivos convertidos: *targets y counts.* Una vez se cuenta con los datos en la consola, se separan los tres grupos de interés: NIT, SFI y ELI.

Cuando ya se encuentren separados los grupos, se extrajeron 10 muestras aleatorias de cada uno de los grupos. Luego estas 30 muestras se extraen de la base principal y se cargan nuevamente. Finalmente se seleccionan los grupos y se extraen las variables para unificarlas en una sola base de datos.

3.2.2. Filtrado y análisis de calidad

Una vez se tiene cargada la base unificada, se realiza limpieza de datos, mediante la eliminación de datos faltantes y de locis que tuvieran una media de 0. Lo anterior con el fin de no afectar los análisis posteriores.

Posteriormente se realiza eliminación de datos y comprobación de sobreexpresión.

3.2.3. Normalización

Luego he hacer el control de calidad de los datos, se pasa a la realización de lo normalización. Para la normalización de estos datos se usó el tipo TMM mediante comando *normalizeCounts* de la librería *tweeDEseq.*

Luego de la normalización, se graficó el resultado para comprobar que verdaderamente se haya realizado el proceso usando el comando *maPlot.*

3.2.4. Identificación de genes diferencialmente expresados

El primer paso en este apartado fue la separación de los grupos, para de esta manera poder realizar comparaciones entre ellos. Luego de esto, ya se inicia la realización de las comparaciones, se realizaron de manera separada: NIT vs SFI, SFI vs ELI y NIT vs ELI. Para esto se utilizó la librería *edgeR.* Este paquete permite la evaluación de la variabilidad entre los grupos y por lo tanto la obtención de los genes diferencialmente expresados.

Cada comparación entre dos grupos constó de la identificación de los genes diferencialmente expresados, gráfica de esta expresión y el cálculo de genes regulados al alta y a la baja.

3.2.5. Genes diferencialmente expresados comunes entre los grupos

Este apartado se representó con un diagrama de Venn, en el que se determinó la intersección entre las tres comparaciones para determinar los genes diferencialmente expresados comunes entre los grupos. Así mismo se determinó cuales genes se encontraban en la intersección.

3.2.6. Anotación de resultados

La anotación de resultados se realizó con las funciones *useMart* y *getBM* de la librería *biomaRt.*

3.2.7. Análisis de significancia biológica

Se creó un objeto *topGo*, usando la librería con el mismo nombre, que incluyó los genes diferencialmente expresados. Con esto se obtuvieron los procesos patológicos relacionados con los genes diferencialmente expresados

3.2.8. Análisis de significancia estadística

Finalmente se quiso determinar si las anotaciones GO obtenidas tenían significancia estadística. Para esto se realizó análisis con la prueba de Fisher

3.3. Software y librerías usadas

Se usó el software RStudio, con lenguaje de R versión 4.0 para MAC. Para el script del documento se usó un *RMarkDown*, que luego se exportó a Word.

En su mayoría el gestor de librerías usado fue ***Bioconductor*** y las librerías que se emplearon fueron:

- readxl
- dplyr
- lubridate
- tidyverse
- tweeDEseq
- limma
- edgeR
- VennDiagram
- RColorBrewer
- biomaRt
- topGO

4. Resultados

4.1. Extracción de datos

Luego de realizar el proceso que se mencionó en el apartado de métodos, se obtuvo una base de datos con 30 muestras y un total de 56202 datos de expresión

```
dim(base)
```

```
## [1] 56202    30
```

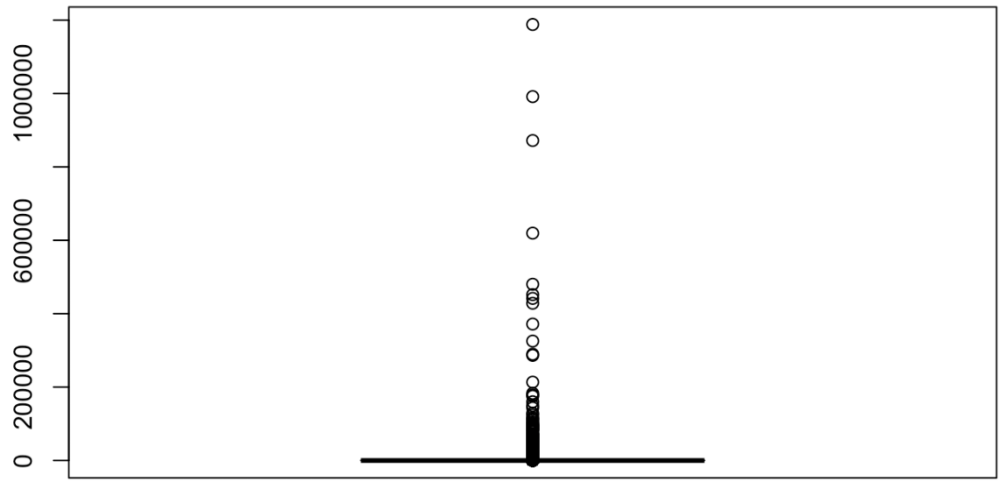
4.2. Filtrado y análisis de calidad

Cuando se realizó el filtrado de los datos, se eliminaron 9814, quedando finalmente 46388 datos de expresión, obteniendo el plot que se muestra en la gráfica 1.

```
dim(base)
```

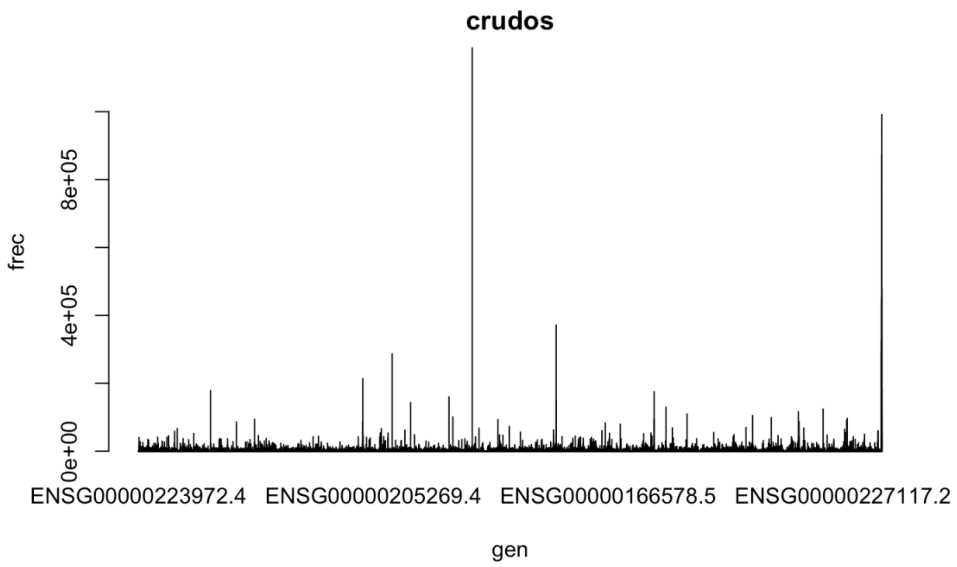
```
## [1] 46388    30
```

Gráfica 1: gráfica de resultado del filtrado de los datos de expresión



Luego se comprobó la sobreexpresión de algunos genes, para lo que se generó una gráfica de expresión cruda de los datos, lo cual se muestra en la gráfica 2.

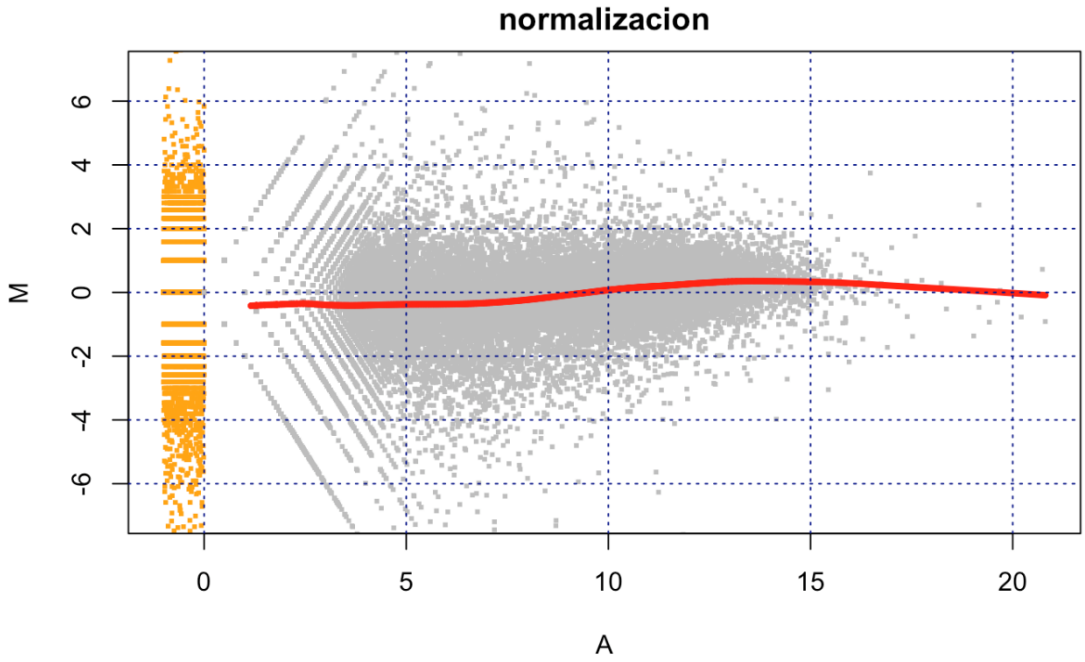
Gráfica 2: Expresión cruda de los datos



4.3. Normalización

Luego de la normalización de los datos, se graficó un MA plot, en donde se observa que después de realizar este proceso los datos tenían menos variabilidad y el log-ratio está cercana a cero. En la gráfica 3, se muestra el MA plot

Gráfica 3: MA plot de los datos normalizados.



4.4. Identificación de genes diferencialmente expresados

4.4.1. Comparación entre NIT y SFI

Inicialmente se calculó la distancia de los genes diferencialmente expresados con MDS plot (gráfica 4), luego se calculó el coeficiente de variación biológica y se presentó en un BCV plot (gráfica 5), finalmente se realizó la clasificación de la expresión diferencial como al alta, a la baja o no signficativa, graficado en un smear plot (gráfica 6).

Numéricamente se encontraron 148 genes diferencialmente expresados entre los dos grupos, de los cuales 122 estaban expresados a la baja y 26 expresados al alta.

4.4.2. Comparación entre SFI y ELI

Se realizaron los cálculos previamente mencionados y se hicieron los mismos gráficos, los cuales se muestran en las gráficas 4, 5 y 6.

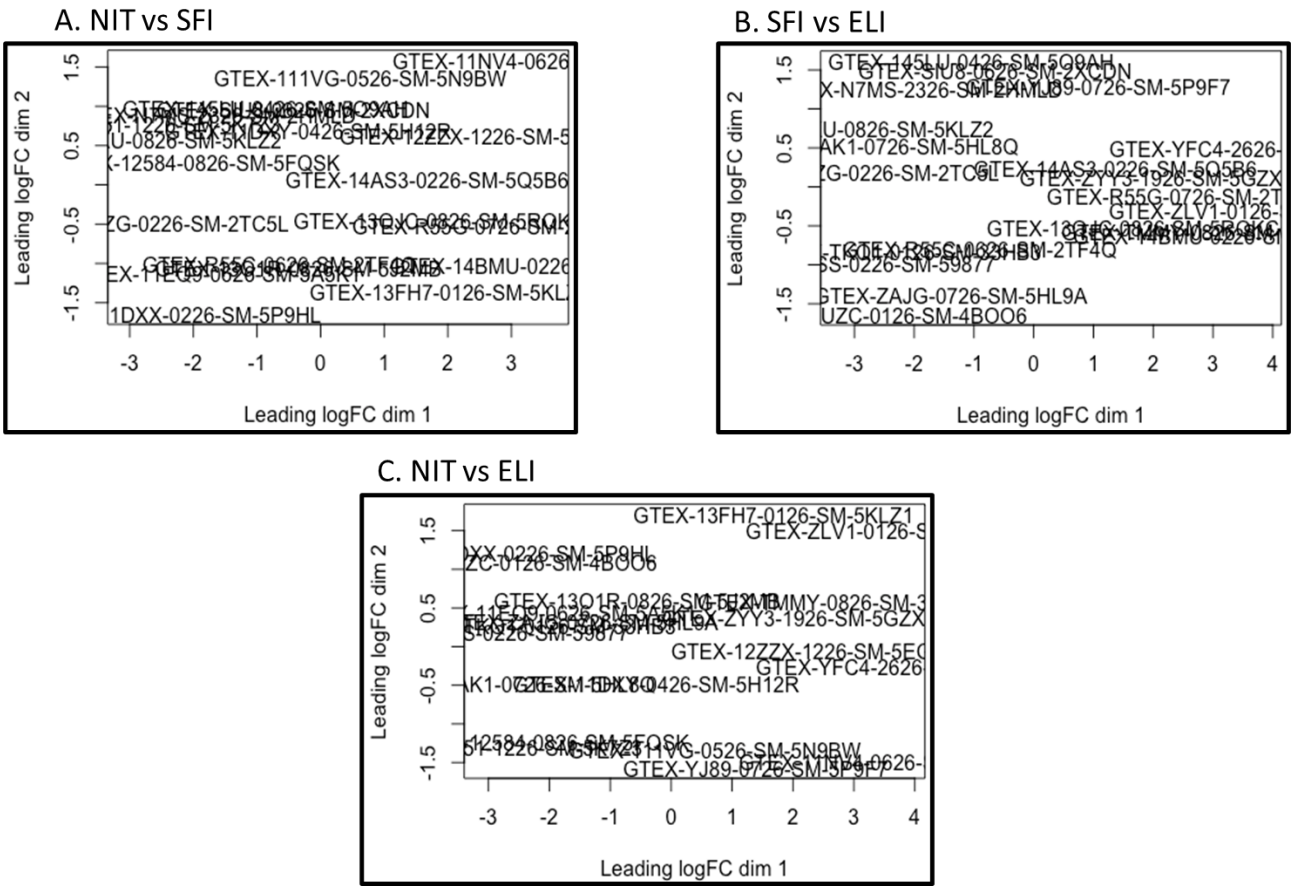
La cantidad de genes diferencialmente expresados que se encontraron entre los dos grupos fue de 147: 59 con regulación a la baja y 88 con regulación al alta

4.4.3. Comparación entre NIT y ELI

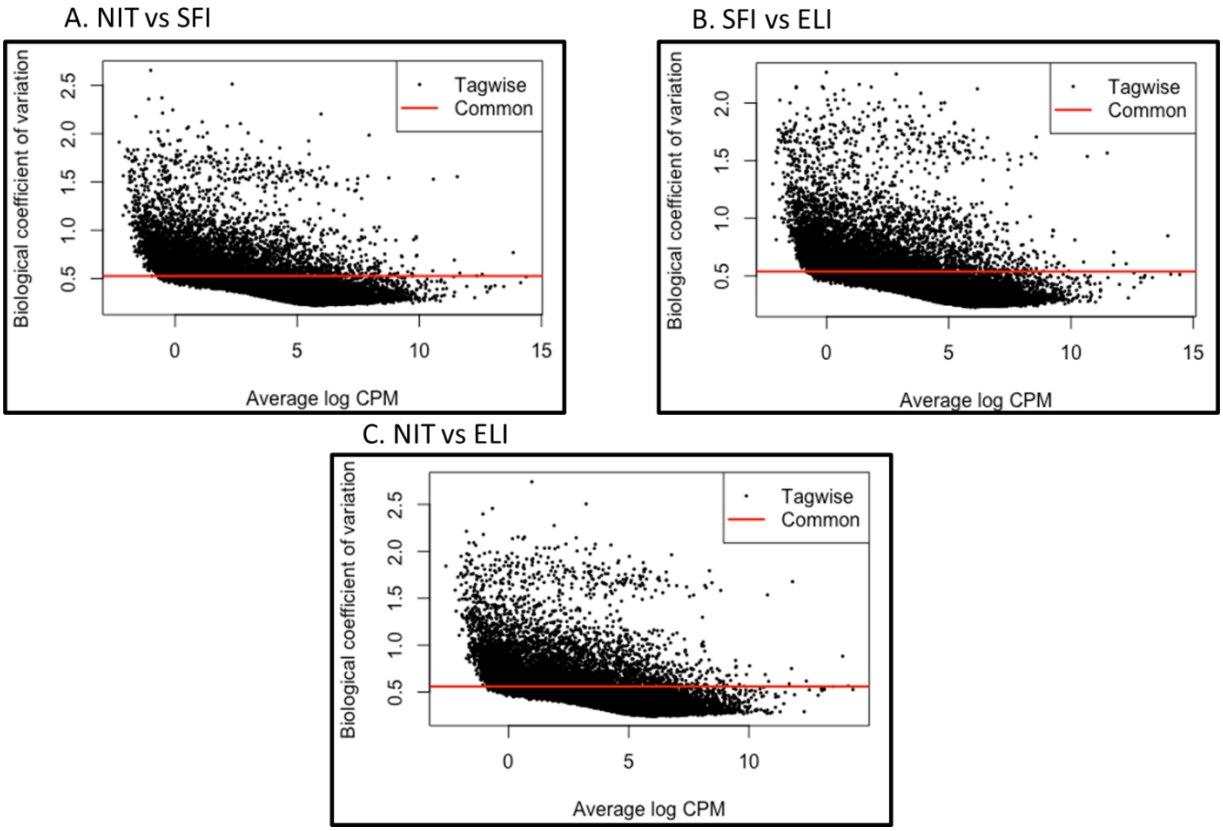
De manera similar se realizó la misma comparación y se realizaron los mismos gráficos para esta comparación (gráficas 4, 5 y 6).

En relación con el número de genes diferencialmente expresados, en el análisis se encontró un total de 117 genes con expresión diferencial, de los cuales 47 tuvieron regulación a la baja y 70 regulación al alta

Gráfica 4: MDS plot de las tres comparaciones



Gráfica 5: BCV plot de las tres comparaciones

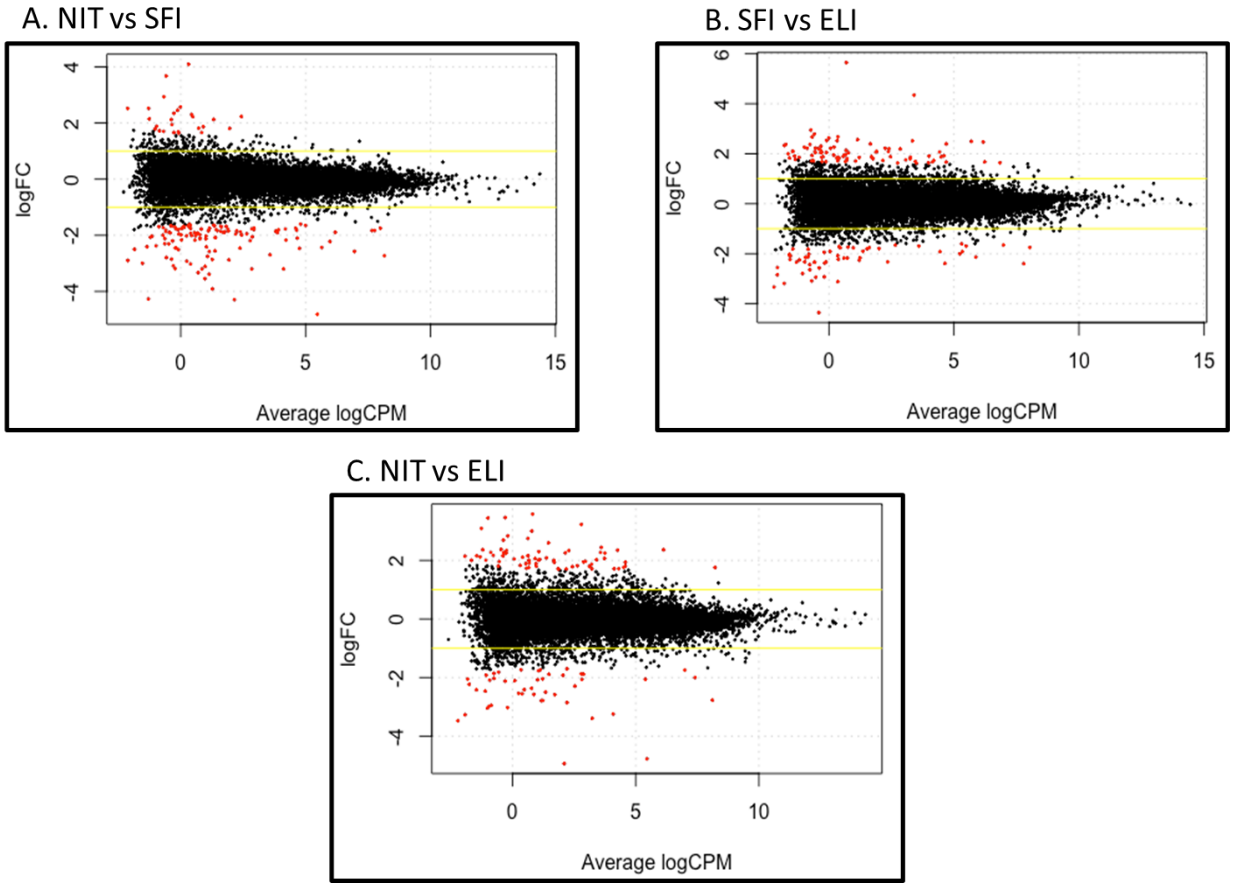


4.5. Genes diferencialmente expresados comunes entre los grupos

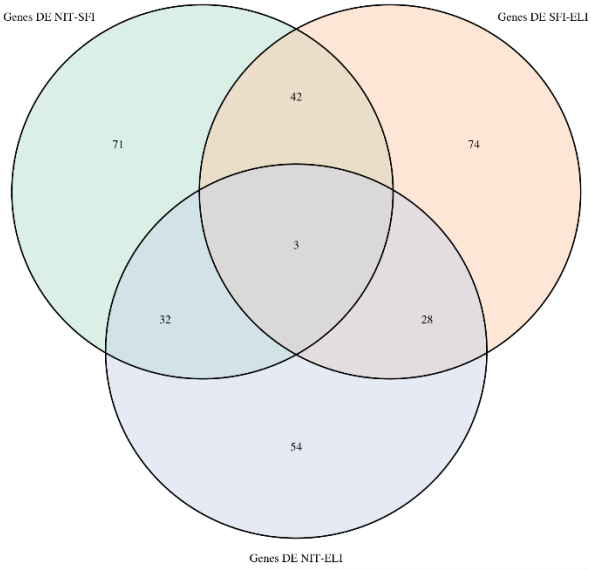
Al realizar comparación de las tres comparaciones se encontró que entre la comparación NIT-SFI y SFI-ELI había 42 genes diferencialmente expresados comunes entre los dos grupos. Así mismo, al comparar SFI-ELI y NIT-ELI se encontraron 28 genes diferencialmente expresados comunes entre estas dos comparaciones. En la intersección de NIT-ELI y NIT-SFI se encontraron 32 genes diferencialmente expresados comunes entre los dos grupos mencionados.

Finalmente, en la intersección de las tres comparaciones, se encontraron 3 genes diferencialmente expresados, que se encontraban en las otras tres comparaciones. El resultado de este análisis se grafica en el diagrama de Venn (gráfica 7).

Gráfica 6: Smear plot de las tres comparaciones



Gráfica 7: Diagrama de Venn



4.6. Anotación de genes

Usando la librería biomaRt se realizó la anotación de genes. En los resultados se obtuvo el nombre de un gen

```
## [1] "IGHV4OR15-8"
```

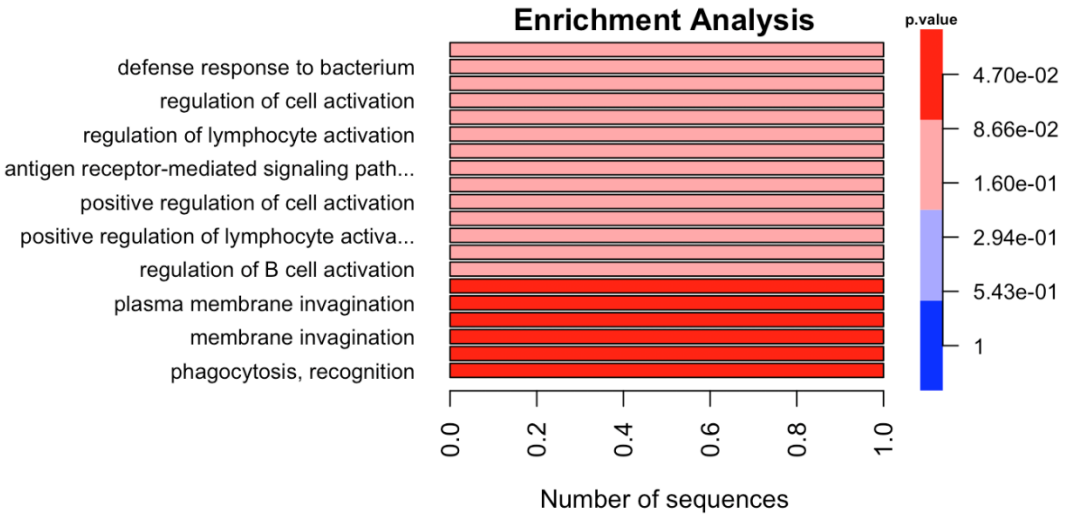
4.7. Análisis de significancia biológica

Usando la librería topGo se realizó este paso. Luego se realizó una prueba de Fisher para determinar aquellas anotaciones que tuvieran significancia estadística (p< 0,01)

Se obtuvo que, de las 4420 anotaciones, se analizaron 665 correspondientes a un gen diferencialmente expresado. De las anotaciones analizadas ninguna tuvo significancia estadística.

Finalmente, se realizó un plot de enriquecimiento (gráfica 8), en donde se muestran los procesos fisiológicos o patológicos involucrados o regulados por los genes diferencialmente expresados.

Gráfica 8: Plot de enriquecimiento donde se muestran los resultados de significancia biológica



5. Discusión

La glándula tiroides hace parte del sistema endocrinológico y se encuentra localizada en la parte anterior del cuello. Tiene funciones fundamentales en el metabolismo y la homeóstasis del cuerpo humano(1).

Así mismo, esta glándula es susceptible de tener patologías, dentro de las cuales hay patologías inflamatorias (por ejemplo, tiroiditis de Hashimoto) o patologías tumorales (por ejemplo, carcinoma papilar de tiroides)(2).

Muchas de estas patologías tienen alguna mediación inmunológica, principalmente las tiroiditis inflamatorias, en la que los linfocitos aportan gran parte de la fisiopatología de la enfermedad. Es por eso que se ha descrito ampliamente que, tanto en la patología inflamatoria como tumoral de la tiroides, los infiltrados inflamatorios, especialmente de linfocitos, tienen significancia en cuanto a evolución y pronóstico de la enfermedad(3–5).

Por lo anterior se hace necesario determinar la expresión diferencial de genes en diferentes tipos de infiltrado teniendo en cuenta la extensión de este para mejorar el entendimiento de la patología, de tal manera que en el futuro puedan darse intervenciones más precisas. En el repositorio GTEx se encuentran las muestras usadas para este análisis. La mayoría de las muestras de tejido tiroideo usadas, correspondían a patologías inflamatorias(6).

Luego del análisis realizado, se encontraron genes diferencialmente expresados, lo cual muestra que cada tipo de extensión del infiltrado puede tener una regulación génica única, sin embargo, al realizar la anotación y la evaluación de significancia biológica, solo hubo un gen identificado: IGHV4OR15-8, gen que codifica la región variable de la cadena pesada de las inmunoglobulinas, lo cual está altamente relacionado con el reconocimiento antigénico y por lo tanto tiene plausibilidad biológica en la presencia de infiltrado linfocitario en patología tiroidea inflamatoria autoinmune(7).

En el plot de enriquecimiento se encontraron procesos fisiopatológicos relacionados con esta expresión diferencial, y en su mayoría, tienen relación con la función de los linfocitos: reconocimiento antigénico, activación celular, fagocitosis, entre otros. Esto está directamente relacionado con el gen identificado, por lo que se puede concluir que tanto la expresión génica, como la presentación clínica, los procesos fisiopatológicos y los hallazgos histopatológicos, en la patología tiroidea, tienen concordancia con los resultados encontrados.

6. Conclusiones

Con el análisis realizado se puede concluir que se encontró expresión diferencial entre los diferentes tipos de infiltrado linfocitario en tejido tiroideo (principalmente enfermedades inflamatorias de la tiroides) del gen que regula la codificación de la región variable de la cadena pesada de las inmunoglobulinas. La expresión de este gen está involucrado en el funcionamiento de los linfocitos B, células encargadas en gran medida de las respuestas inflamatorias, tanto de defensa contra microorganismos externos, como de procesos autoinmunes, los cuales pueden afectar la glándula tiroides.

7. Link a github:

https://github.com/juliansanchez8/PEC2_ADO

8. Bibliografía

1. Stathatos N. Anatomy and physiology of the thyroid gland clinical correlates to thyroid cancer. In: Thyroid Cancer (Second Edition): A Comprehensive Guide to Clinical Management. Humana Press; 2006. p. 3–7.
2. Chiasera JM. Back to the Basics: Thyroid Gland Structure, Function and Pathology [Internet]. Vol. 26, CLINICAL LABORATORY SCIENCE. 2013 [cited 2020 Jun 14]. Available from: http://hwmaint.clsjournal.ascls.org/
3. Kuo C-Y, Liu T-P, Yang P-S, Cheng S-P. Characteristics of lymphocyte-infiltrating papillary thyroid cancer. J Cancer Res Pract. 2017 Sep 1;4(3):95–9.
4. Scopsi L, Collini P, Sampietro G, Boracchi P, Pilotti S. Prognostic impact of thyroid lymphocytic infiltration in patients with medullary thyroid carcinoma. Thyroid. 1996;6(6):613–7.
5. Bay BH, Sit KH, Pang AS. Lymphocytic infiltration in focal thyroiditis: An ultrastructural case study. Immunobiology. 1994;190(3):290–4.
6. GTEx Portal [Internet]. [cited 2020 Jun 14]. Available from: https://www.gtexportal.org/home/histologyPage
7. Gene: IGHV4OR15-8 (ENSG00000259261) - Summary - Homo sapiens - GRCh37 Archive browser 100 [Internet]. [cited 2020 Jun 14]. Available from:

http://grch37.ensembl.org/Homo_sapiens/Gene/Summary?db=core;g=ENSG00000259261;r=15:22472918-22473353;t=ENST00000557788

8. Apéndice: códigos usados

Sanchez_Julian_ADO_PEC2

Julian Sanchez Bautista

6/13/2020

1. EXTRACCION DE LOS DATOS

Primero se cargan los datos que # Plots

```
plotEnrich = function(allRes, title){
  # Plotting!
  layout(t(1:2), widths = c(8,1))
  par(mar=c(4, .5, .7, .7), oma = c(3, 15, 3, 4), las = 1)

  rbPal = colorRampPalette(c('red', 'white', 'blue'))
  pvalue = as.numeric(gsub("<", "", allRes$classicFisher))
  max_value = as.integer(max(-log(pvalue))) + 1
  pv_range = exp(-seq(max_value, 0, -1))
  allRes$Color = rbPal(max_value) [cut(pvalue, pv_range)]

  o = order(allRes$Significant, decreasing = T)
  barplot(allRes$Significant[o], names.arg = allRes$Term[o], las = 2, horiz = T, col = allRes$Color[o],
        xlab = "Number of sequences", main = title, cex.names = 0.85)

  image(0, seq(1, max_value), t(seq_along(seq(1, max_value))), col = rev(rbPal(max_value)), axes = F, ann = F)
```

```
  pv_label = exp(-seq(log(1), -log(min(pvalue)), l = 6))
  pv_label = formatC(pv_label, format = "e", digits = 2)
  axis(4, at = seq(1, max_value, length = 6), labels = c(1, pv_label[2:6]), cex.axis = 0.85)
  title("p.value", cex.main = 0.6)
}
```

plotEnrich(allRes = allRes, title = 'Enrichment Analysis')usaremos para el analisis.

Elijo el directorio en el cual trabajar

```
setwd("~/Downloads/Datosomicos/PEC3")

library(readxl)
```

Cargo targets

```
target <- read.csv("~/Downloads/Datosomicos/PEC3/targets.csv")
```

Luego counts

```
counts2 <- read_excel("counts2.xlsm")

## New names:
## * `` -> ...1
```

Los separo segun los grupos

```
NIT <- subset(target,grepl("^(NIT)", target$Group))
SFI <- subset(target,grepl("^(SFI)", target$Group))
ELI <- subset(target,grepl("^(ELI)", target$Group))
```

Extraigo 10 de cada uno

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

set.seed(12)

NIT.10 <- sample_n(NIT, 10)
```

```
SFI.10 <- sample_n(SFI, 10)
ELI.10 <- sample_n(ELI, 10)
```

Observandolos en los datos generados los extraigo de la base principal y los cargo

```
ELI_10 <- read_excel("~/Downloads/Datosomicos/PEC3/ELI_10.xlsm")

NIT_10 <- read_excel("~/Downloads/Datosomicos/PEC3/NIT_10.xlsm")

SFI_10 <- read_excel("~/Downloads/Datosomicos/PEC3/SFI_10.xlsm")
```

Seleccionamos los grupos

```
colum <- rbind(NIT_10, SFI_10, ELI_10)
grupos <- as.factor(colum$Group)
colNIT_SFI <- rbind(NIT_10,SFI_10)
g1 <- factor(colNIT_SFI$Group)
colSFI_ELI <- rbind(SFI_10,ELI_10)
g2 <- factor(colSFI_ELI$Group)
colNIT_ELI <- rbind(NIT_10,ELI_10)
g3 <- factor(colNIT_ELI$Group)
```

Extraemos los datos de las variables y las unimos en una sola base

```
library(readxl)
basejdsb <- read_excel("basejdsb.xlsm")
```

A partir de esa base de datos colocamos la primera columna como los terminos GO

```
library(lubridate)

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(tidyverse)

## -- Attaching packages ---------------------------------------------
---------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v stringr 1.4.0
## v tidyr   1.1.0     v forcats 0.5.0
## v readr   1.3.1

## -- Conflicts ------------------------------------------------------
---------------------- tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
```

```
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

```
base <- basejdsb %>%
     remove_rownames() %>%
     column_to_rownames(var = 'GO')
```

Comprobamos

```
dim(base)
```

```
## [1] 56202    30
```

FILTRADO Y ANALISIS DE CALIDAD

HACEMOS LIMPIEZA DE LOS DATOS, ELIMINANDO DATOS FALTANTES Y LOCIS CON MEDIA DE 0 PARA NO AFECTAR EL ANALISIS

```
table(is.na(base))
```

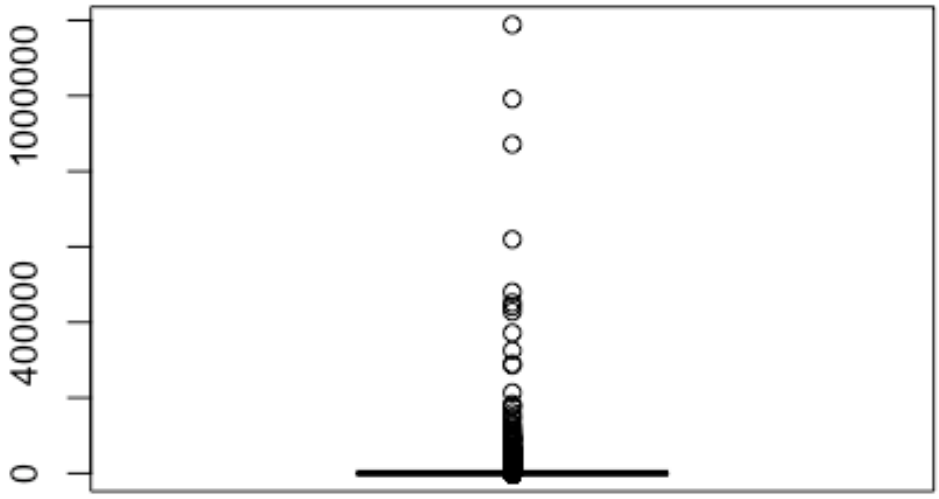
```
##
##   FALSE
## 1686060
```

```
mediagen<-apply(base, 1, mean)
table(mediagen == 0)
```

```
##
## FALSE  TRUE
## 46388  9814
```

Graficamos el resultado

```
boxplot(mediagen)
```



Eliminacion de datos

```
delete<-base[which(mediagen ==0),]
i<-intersect(rownames(delete), rownames(base))
base<-base[!rownames(base)%in% i,]
```

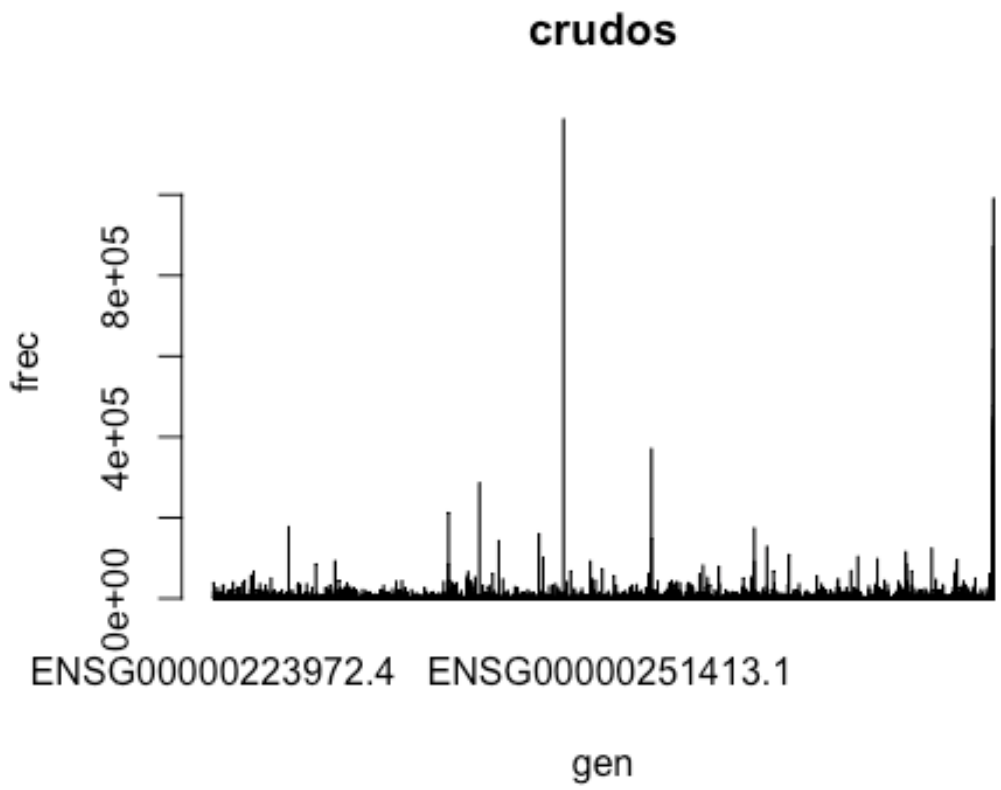
Comprobamos

```
dim(base)

## [1] 46388    30
```

Vamos que disminuyeron de 56202 a 46388 datos de expresion

Comprobamos la sobrexpresion

```
mediagendelet<-apply(base, 1, mean)
barplot(mediagendelet,main = 'crudos', xlim=NULL, xlab = 'gen', ylab='fre
c')
```



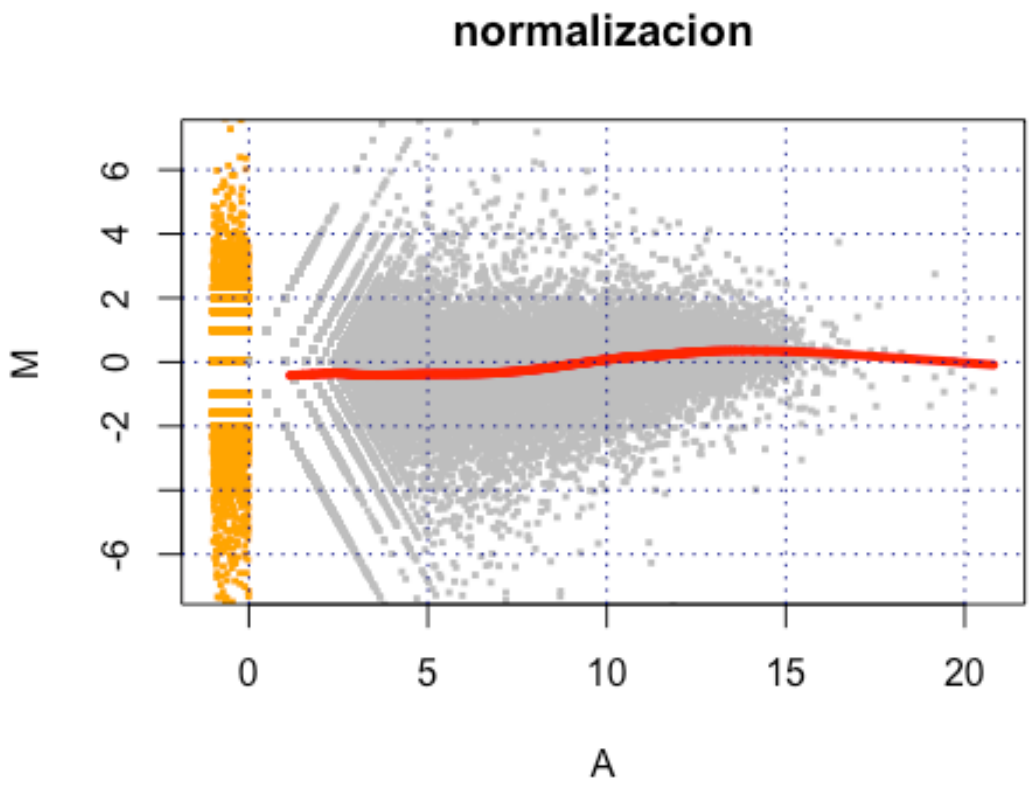
NORMALIZACION DE DATOS TTM

```
library(tweeDEseq)
normales <- normalizeCounts(base)

## Using edgeR-TMM normalization.

## Calculating normalization factors with the TMM method.

## Estimating common dispersion.

## Estimating tagwise dispersions.

## Calculating effective library sizes.

## Adjusting counts to effective library sizes using tagwise dispersions.
```

Grafico de normalizacion

```
library(edgeR)

## Loading required package: limma

library(limma)
maPlot(normales[,1], normales[,2],
       pch=15, cex=.4, ylim=c(-7,7),
```

```
       allCol="grey", lowess=TRUE)
grid(col="darkblue")
title("normalizacion")
```



IDENTIFICACION DE GENES DIFERENCIALMENTE EXPRESADOS

Primero saparamos los grupos para poder hacer las comparaciones entre ellos.

```
grupo1 <- cbind(normales[,1:10],normales[,11:20])
grupo2 <- cbind(normales[,11:20],normales[,21:30])
grupo3 <- cbind(normales[,1:10],normales[,21:30])
```

- Comparacion NIT y SFI

```
d <- DGEList(counts = grupo1, group = g1)
d <- calcNormFactors(d)

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced
```

```
## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
```

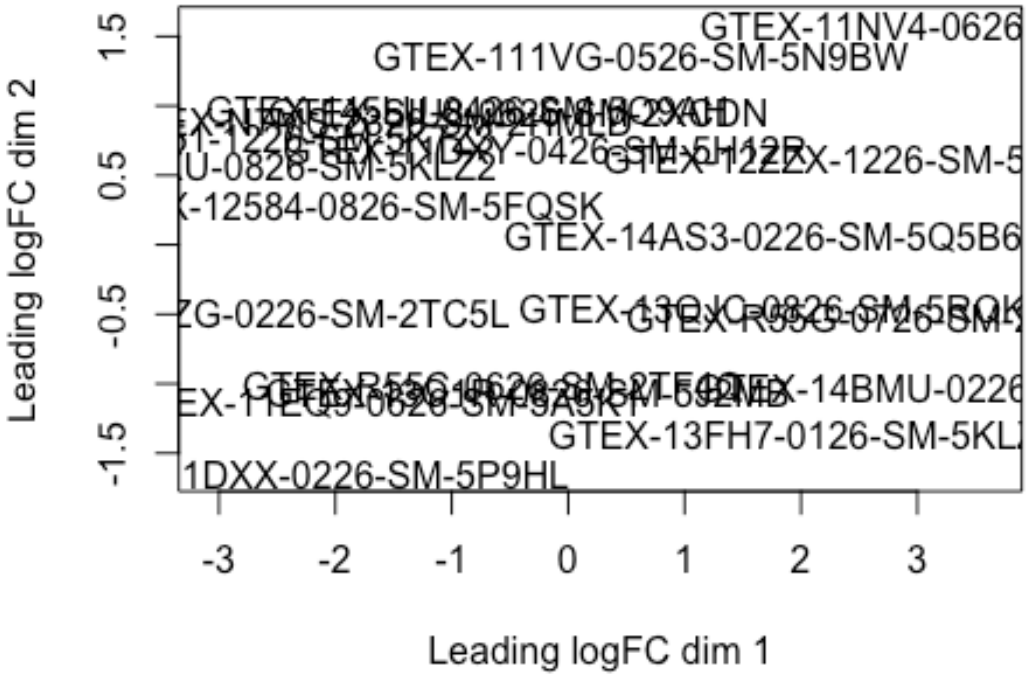
```
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced
```

```
m <- sweep(d$counts, 2, 1e6 / d$samples$lib.size, '*')
ridx_grupo1 <- rowSums(m>1) >= 2
table(ridx_grupo1)
```

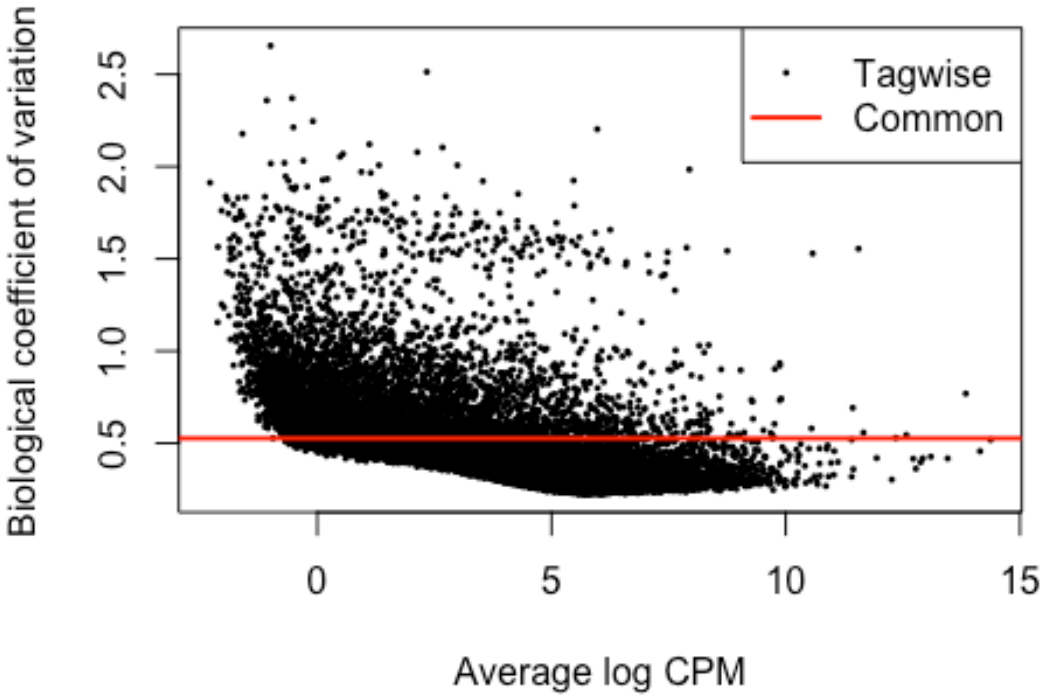
```
## ridx_grupo1
## FALSE  TRUE
## 27287 19101
```

Graficamos

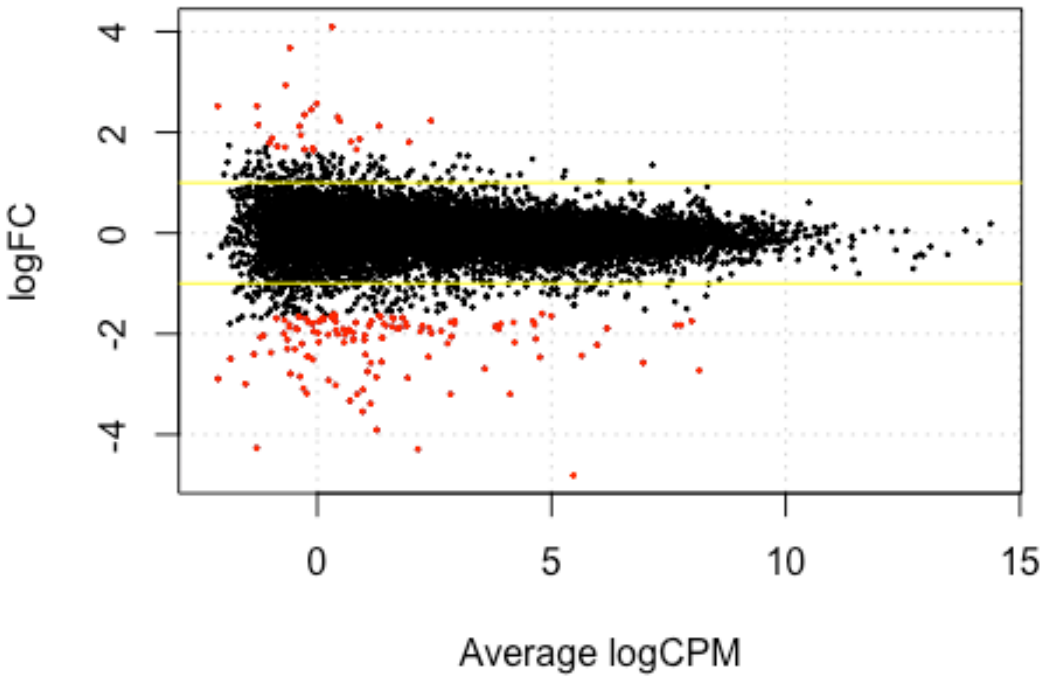
```
d <- d[ridx_grupo1,]
plotMDS(d)
```




```
d1 <- estimateCommonDisp(d)
dtag1 <- estimateTagwiseDisp(d1)
res.common1  <- exactTest(d1, pair=c("NIT", "SFI"), dispersion="common")
res.tagwise1 <- exactTest(dtag1, pair=c("NIT", "SFI"), dispersion="tagwis
e")

plotBCV(dtag1, cex=0.4)
```




```
dec1 <- decideTestsDGE(res.common1,p=0.001, adjust="BH")
dtag_grupo1 <- rownames(d1)[as.logical(dec1)]
plotSmear(res.common1, de.tags = dtag_grupo1)
abline(h=c(-1,1),col="yellow")
```



Calculo de los genes regulados a la alta y a la baja:

```
d1_df <- as.data.frame(dec1)
d1_df[,2] <- rownames(res.common1)
up1 <- d1_df[which(d1_df$`SFI-NIT`==1),]
down1 <- d1_df[which(d1_df$`SFI-NIT`==-1),]
summary(dec1)
```

```
##        SFI-NIT
## Down       122
## NotSig   18953
## Up          26
```

```
genes_grupo1 <- c(up1$V2,down1$V2)
```

REALIZAMOS EL MISMO PROCESO PARA LAS DEMAS COMPARACIONES

- Comparacion SFI y ELI

```
d <- DGEList(counts = grupo2, group = g2)
d <- calcNormFactors(d)
```

```
## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced
```

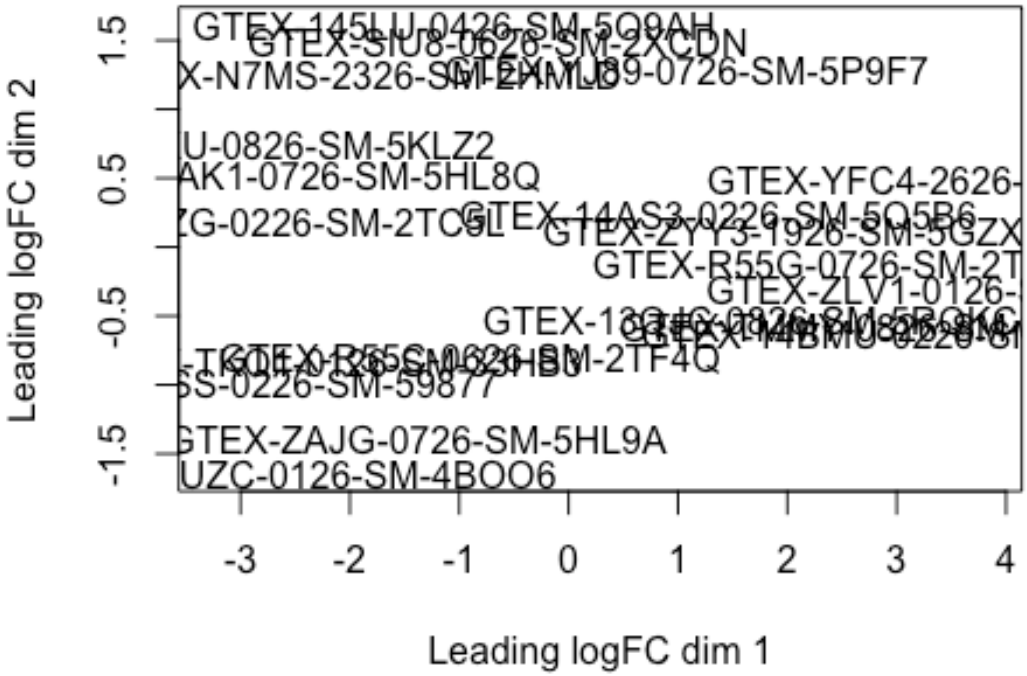
```
## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
```

```
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced
```

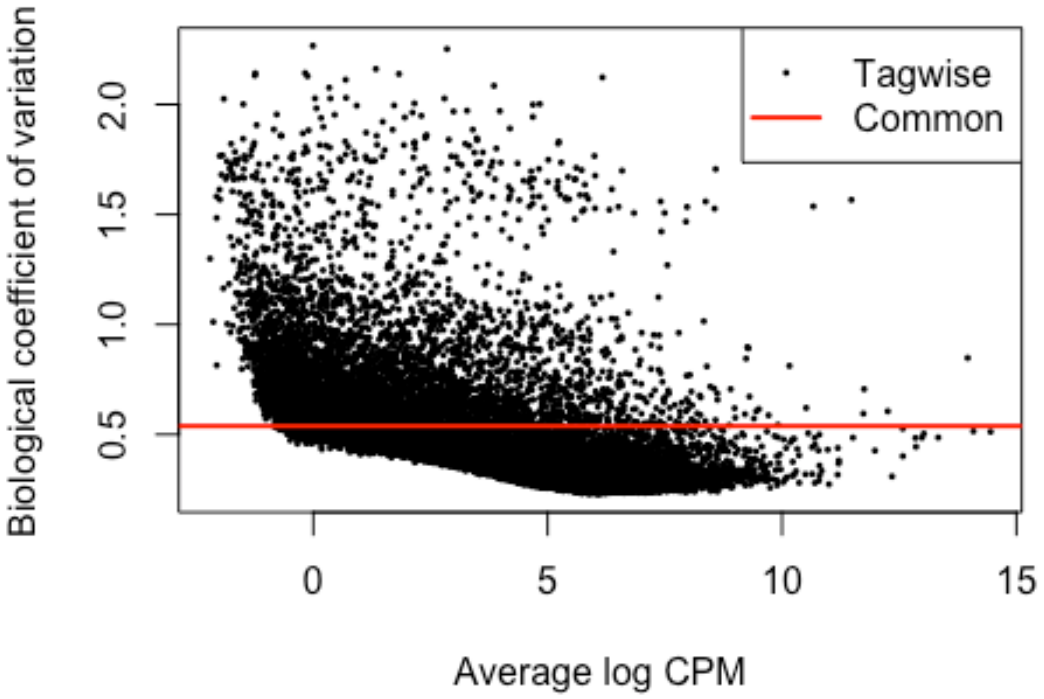
```
m <- sweep(d$counts, 2, 1e6 / d$samples$lib.size, '*')
ridx_grupo2 <- rowSums(m>1) >= 2
table(ridx_grupo2)
```

```
## ridx_grupo2
## FALSE  TRUE
## 27279 19109
```

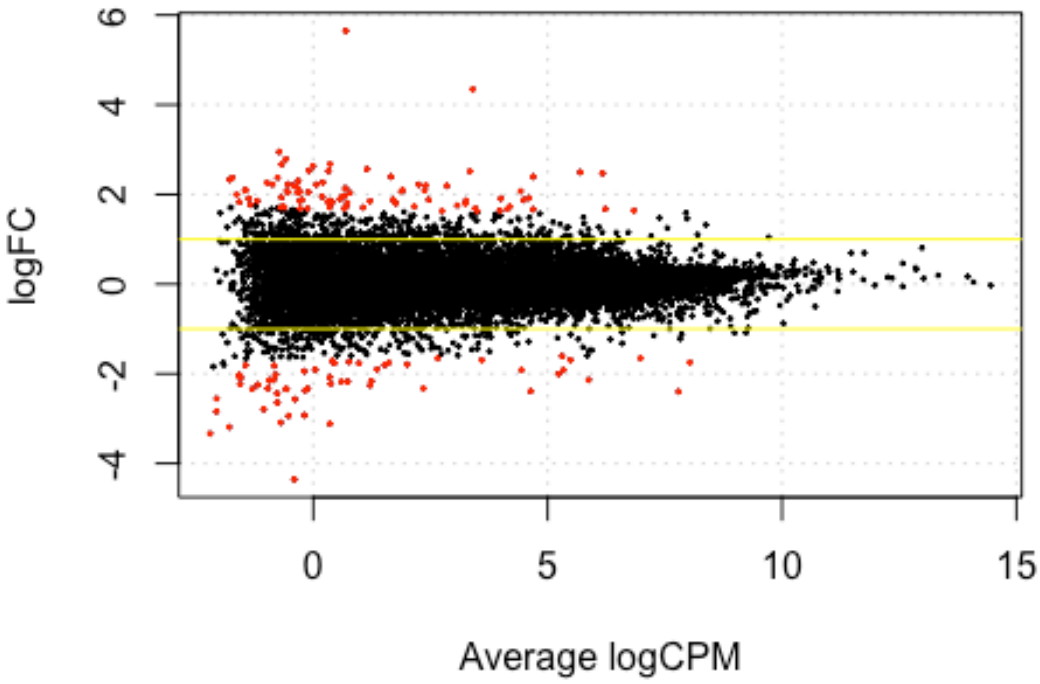
```
d <- d[ridx_grupo2,]
plotMDS(d)
```




```
d2 <- estimateCommonDisp(d)
dtag2 <- estimateTagwiseDisp(d2)
res.common2  <- exactTest(d2, pair=c("SFI", "ELI"), dispersion="common")
res.tagwise2 <- exactTest(dtag2, pair=c("SFI", "ELI"), dispersion="tagwis
e")

plotBCV(dtag2, cex=0.4)
```




```
dec2 <- decideTestsDGE(res.common2,p=0.001, adjust="BH")
dtag_segun <- rownames(d2)[as.logical(dec2)]
plotSmear(res.common2, de.tags = dtag_segun)
abline(h=c(-1,1),col="yellow")
```




```
d2_df <- as.data.frame(dec2)
d2_df[,2] <- rownames(res.common2)
up2 <- d2_df[which(d2_df$`ELI-SFI`==1),]
down2 <- d2_df[which(d2_df$`ELI-SFI`==-1),]
summary(dec2)
```

```
##        ELI-SFI
## Down        59
## NotSig   18962
## Up          88
```

```
genes_grupo2 <- c(up2$V2,down2$V2)
```

- Comparacion NIT y ELI

```
d <- DGEList(counts = grupo3, group = g3)
d <- calcNormFactors(d)
```

```
## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced
```

```
## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced
```

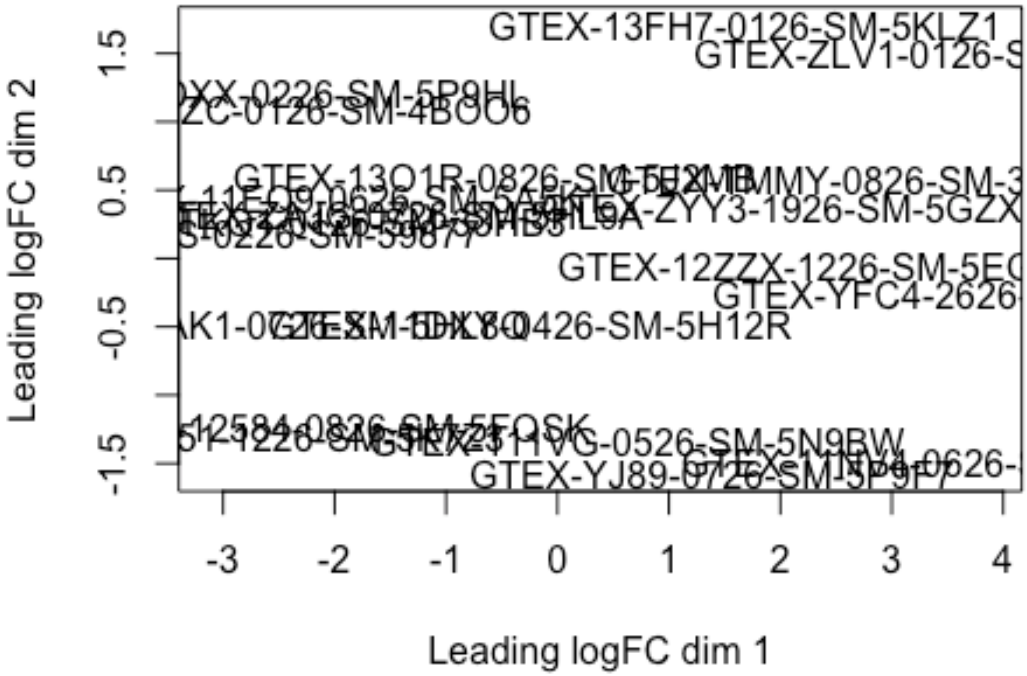
```
## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
```

```
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced

## Warning in .calcFactorTMM(obs = x[, i], ref = x[, refColumn], libsize.
obs =
## lib.size[i], : NaNs produced
```

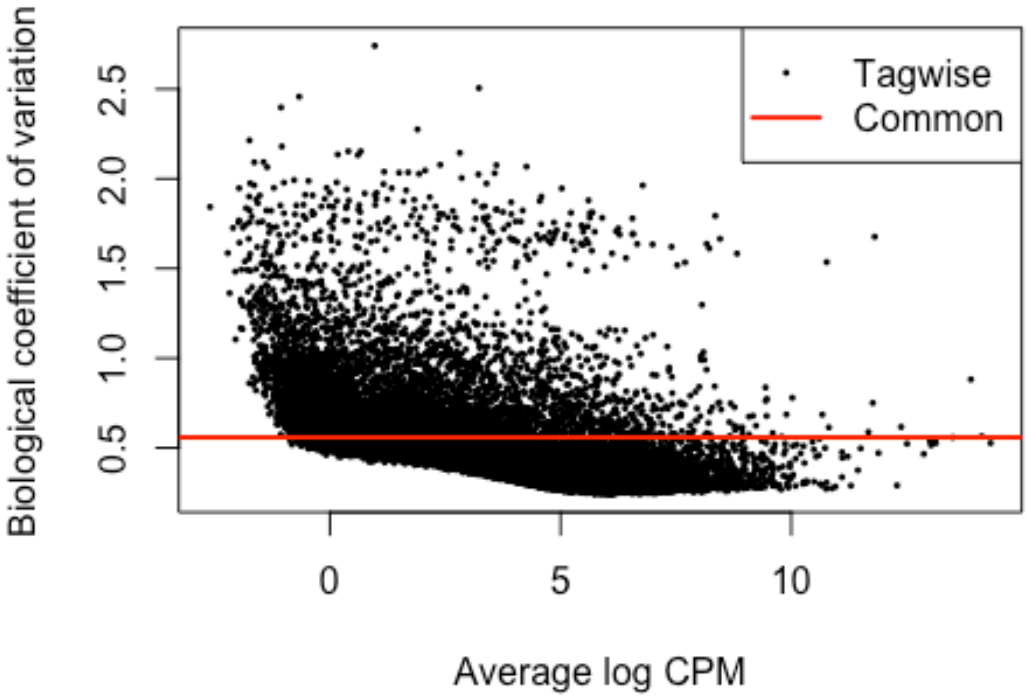
```
m <- sweep(d$counts, 2, 1e6 / d$samples$lib.size, '*')
ridx_grupo3 <- rowSums(m>1) >= 2
table(ridx_grupo3)
```

```
## ridx_grupo3
## FALSE  TRUE
## 27358 19030
```

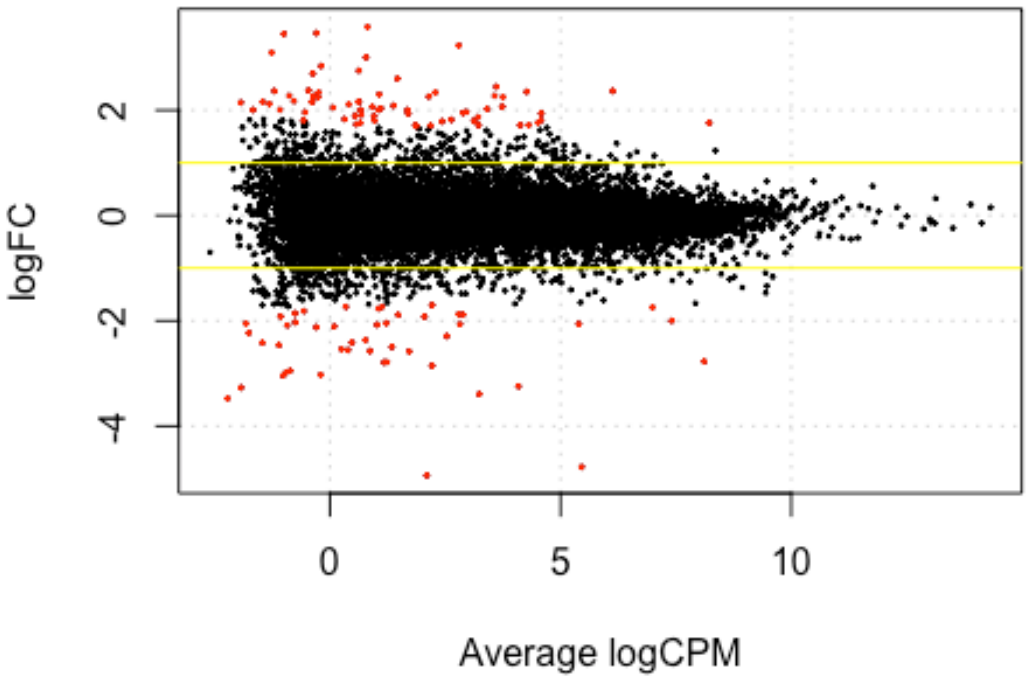
```
d <- d[ridx_grupo3,]
plotMDS(d)
```




```
d3 <- estimateCommonDisp(d)
dtag3 <- estimateTagwiseDisp(d3)
res.common3  <- exactTest(d3, pair=c("NIT", "ELI"), dispersion="common")
res.tagwise3 <- exactTest(dtag3, pair=c("NIT", "ELI"), dispersion="tagwis
e")

plotBCV(dtag3, cex=0.4)
```




```
dec3 <- decideTestsDGE(res.common3,p=0.001, adjust="BH")
dtag_terc <- rownames(d3)[as.logical(dec3)]
plotSmear(res.common3, de.tags = dtag_terc)
abline(h=c(-1,1),col="yellow")
```




```
d3_df <- as.data.frame(dec3)
d3_df[,2] <- rownames(res.common3)
up3 <- d3_df[which(d3_df$`ELI-NIT`==1),]
down3 <- d3_df[which(d3_df$`ELI-NIT`==-1),]
summary(dec3)
```

```
##        ELI-NIT
## Down        47
## NotSig   18913
## Up          70
```

```
genes_grupo3 <- c(up3$V2,down3$V2)
```

GENES DIFERENCIALMENTE EXPRESADOS EN DIAGRAMA DE VENN - COMUNES ENTRE LOS GRUPOS

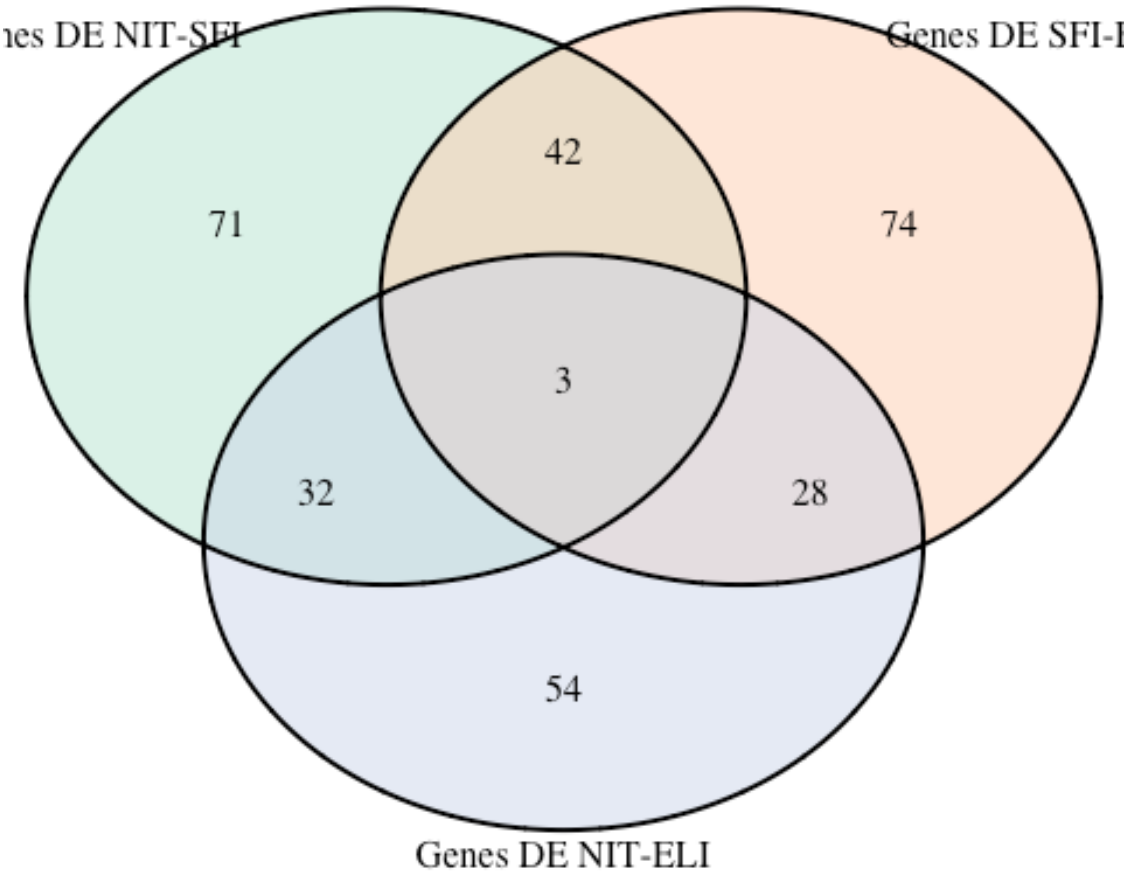
```
library(VennDiagram)
```

```
## Loading required package: grid
```

```
## Loading required package: futile.logger
```

```
library(RColorBrewer)
diagrama <- venn.diagram(x = list("Genes DE NIT-SFI" = genes_grupo1, "Gen
es DE SFI-ELI" = genes_grupo2,"Genes DE NIT-ELI" = genes_grupo3), fill =
```

```
brewer.pal(3, "Pastel2"), filename = NULL)
grid.draw(diagrama)
```



La interseccion de genes DE de los tres grupos es de 3.

Para mostrar cuales son usamos el siguiente codigo:

```
encomun <- intersect(intersect(genes_grupo1,genes_grupo2),genes_grupo3)
encomun

## [1] "ENSG00000100604.8" "ENSG00000259261.2" "ENSG00000215644.5"

length(encomun)

## [1] 3
```

ANOTACION DE LOS RESULTADOS

```
library(biomaRt)
mart <- useMart(biomart = "ensembl", dataset = "hsapiens_gene_ensembl")

commongenes<-getBM(attributes =  c("hgnc_symbol"), filters = "ensembl_gen
e_id_version", values =encomun, mart = mart)
```

```
commongenes <- commongenes[,1]
commongenes

## [1] "IGHV4OR15-8"
```

ANALISIS DE SIGNIFICANCIA BIOLOGICA

```
totales<-getBM(attributes =  c("hgnc_symbol","go_id"), filters = "ensembl
_gene_id_version", values =rownames(normales), mart = mart)

head(totales,10)

##    hgnc_symbol go_id
## 1       SNORA77
## 2    RNU6-1265P
## 3     RNU6-880P
## 4     RNU6-828P
## 5       SNORD46
## 6      RNU4-61P
## 7     RNU1-155P
## 8      RN7SKP19
## 9    RNU6-1199P
## 10   RNU6-1062P
```

Eliminamos los que no cuentan con terminos GO

```
elim <- which(totales$go_id=="")
dim(totales)

## [1] 17504     2

totales <- totales[-elim,]
dim(totales)

## [1] 8729    2
```

Conversion para calculo de objeto TOPGO

```
list_genes <- unique(totales$hgnc_symbol)
lista <- list()
for (i in list_genes) {
  lista[[i]] = totales[which(totales$hgnc_symbol==i),]$go_id
}
head(lista,2)

## $MIR215
## [1] "GO:0005615" "GO:1903231" "GO:0035195" "GO:1903561" "GO:0045391"
##
## $MIR429
## [1] "GO:1903231" "GO:0035195"
```

```r
gen2 <- names(lista)
compar <- factor(as.integer(gen2 %in% commongenes))
table(compar)
```

```
## compar
##   0   1
## 747   1
```

```r
library(topGO)
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following object is masked from 'package:limma':
##
##     plotMA
```

```
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
```

```
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##     as.data.frame, basename, cbind, colnames, dirname, do.call,
##     duplicated, eval, evalq, get, grep, grepl, intersect, is.unsorted,
##     lapply, mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##     pmin.int, rank, rbind, rownames, sapply, setdiff, sort, table,
##     tapply, union, unique, unsplit, which, which.max, which.min
```

```
## Loading required package: graph
```

```
##
## Attaching package: 'graph'
```

```
## The following object is masked from 'package:stringr':
##
##     boundary

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: GO.db

## Loading required package: AnnotationDbi

## Loading required package: stats4

## Loading required package: IRanges

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:tidyr':
##
##     expand

## The following objects are masked from 'package:lubridate':
##
##     second, second<-

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:base':
##
##     expand.grid

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:purrr':
##
##     reduce

## The following object is masked from 'package:lubridate':
##
##     %within%
```

```
## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

##
## Attaching package: 'AnnotationDbi'

## The following object is masked from 'package:dplyr':
##
##     select

##

## Loading required package: SparseM

##
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':
##
##     backsolve

##
## groupGOTerms:    GOBPTerm, GOMFTerm, GOCCTerm environments built.

##
## Attaching package: 'topGO'

## The following object is masked from 'package:IRanges':
##
##     members

## The following object is masked from 'package:grid':
##
##     depth

names(compar) <- gen2

GO_data <- new("topGOdata", ontology="BP", allGenes=compar,annot = annFUN
.gene2GO, gene2GO = lista)

##
## Building most specific GOs .....

##  ( 1514 GO terms found. )

##
## Build GO DAG topology ..........

##  ( 4420 GO terms and 10181 relations. )

##
## Annotating nodes ...............
```

```
##  ( 665 genes annotated to the GO terms. )
```

Aplicacion del TEST DE FISHER

```
resFisher = runTest(GO_data, algorithm = 'classic', statistic = 'fisher')

##
##           -- Classic Algorithm --
##
##       the algorithm is scoring 76 nontrivial nodes
##       parameters:
##           test statistic: fisher

resFisher

##
## Description:
## Ontology: BP
## 'classic' algorithm with the 'fisher' test
## 4420 GO terms scored: 0 terms with p < 0.01
## Annotation data:
##     Annotated genes: 665
##     Significant genes: 1
##     Min. no. of genes annotated to a GO: 1
##     Nontrivial nodes: 76
```

De las 4420 anotaciones GO totales, 665 de 1 genes DE fueron analizados. De estos GO, 0 obtienen una significaci??n de p < 0.01

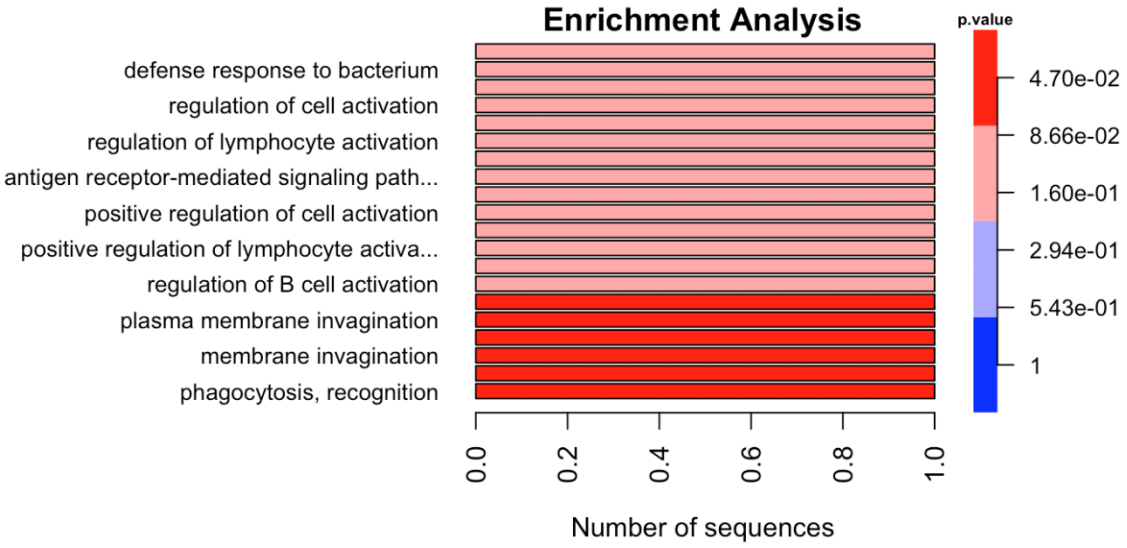
Corremos los primeros 20 en terminos practicos de interpretacion

```
Nodes = 20
allRes = GenTable(GO_data, classicFisher = resFisher, topNodes = Nodes)
head(allRes)

##        GO.ID                                     Term Annotated Signif
icant
## 1 GO:0006910                phagocytosis, recognition        31
1
## 2 GO:0006911                 phagocytosis, engulfment        31
1
## 3 GO:0010324                    membrane invagination        31
1
## 4 GO:0050853        B cell receptor signaling pathway        31
1
## 5 GO:0099024             plasma membrane invagination        31
1
## 6 GO:0050871 positive regulation of B cell activation        32
1
##   Expected classicFisher
## 1     0.05         0.047
## 2     0.05         0.047
```

```
## 3     0.05         0.047
## 4     0.05         0.047
## 5     0.05         0.047
## 6     0.05         0.048
```

Y finalmente graficamos

```
plotEnrich = function(allRes, title){

  # Plotting!

  layout(t(1:2), widths = c(8,1))

  par(mar=c(4, .5, .7, .7), oma = c(3, 15, 3, 4), las = 1)

  rbPal = colorRampPalette(c('red', 'white', 'blue'))

  pvalue = as.numeric(gsub("<", "", allRes$classicFisher))

  max_value = as.integer(max(-log(pvalue))) + 1

  pv_range = exp(-seq(max_value, 0, -1))

  allRes$Color = rbPal(max_value) [cut(pvalue, pv_range)]

  o = order(allRes$Significant, decreasing = T)

  barplot(allRes$Significant[o], names.arg = allRes$Term[o], las = 2, horiz = T, col =
allRes$Color[o],

        xlab = "Number of sequences", main = title, cex.names = 0.85)

  image(0, seq(1, max_value), t(seq_along(seq(1, max_value))), col =
rev(rbPal(max_value)), axes = F, ann = F)

  pv_label = exp(-seq(log(1), -log(min(pvalue)), l = 6))

  pv_label = formatC(pv_label, format = "e", digits = 2)

  axis(4, at = seq(1, max_value, length = 6), labels = c(1, pv_label[2:6]), cex.axis = 0.85)

  title("p.value", cex.main = 0.6)

}

plotEnrich(allRes = allRes, title = 'Enrichment Analysis')
```


Enrichment Analysis
defense response to bacterium
regulation of cell activation
regulation of lymphocyte activation
antigen receptor-mediated signaling path...
positive regulation of cell activation
positive regulation of lymphocyte activa...
regulation of B cell activation
plasma membrane invagination
membrane invagination
phagocytosis, recognition
0.0
0.2
0.4
0.6
0.8
1.0
Number of sequences
p.value
4.70e-02
8.66e-02
1.60e-01
2.94e-01
5.43e-01
1