PEC 2: Opción 1 - Análisis de datos de RNA-Seq

Análisis de Datos Ómicos

Máster de Bioinformática y Bioestadística UOC y UB

Ana M Gómez Martínez

15 de junio, 2021

Contents

Abstract

En este informe se plantean las cuestiones que deseamos responder (Objetivos) a través de nuestro estudio de RNA-seq. Se realizan los análisis de expresión diferencial necesarios (Pipeline del análisis, qué se ha hecho en cada paso y Resultados) y finalmente se expone una discusión (Discusión).

Por tanto, vamos a investigar los datos de expresión RNA-seq de un análisis del **tiroides** obtenido del repositorio GTEx, comparando tres tipos de infiltración: `Not infiltrated tissues` (**NIT**), `Small focal infiltrates` (**SFI**) y `Extensive lymphoid infiltrates` (**ELI**).

Objetivos

Nuestro objetivo en esta PEC2 es buscar **expresión diferencial** en muestras de tiroides con 3 tipos distintos de infiltración mediante un análisis de datos de RNA-seq. Para ello, compararemos los grupos **NIT vs SFI**, **NIT vs ELI** y **SFI vs ELI**, tomando 10 muestras de cada grupo (30 en total).

Las preguntas serían:

- ¿Hay diferencia de expresión de genes en tejidos de tiroides sin infiltraciones vs tejidos de tiroides con infiltraciones focales? → **NIT vs SFI**
- ¿Hay diferencia de expresión de genes en tejidos de tiroides sin infiltraciones vs tejidos de tiroides con infiltraciones extensas? → **NIT vs ELI**
- ¿Hay diferencia de expresión de genes en tejidos de tiroides con infiltraciones focales vs tejidos de tiroides con infiltraciones extensas? → **SFI vs ELI**

Materiales y métodos

Nota: El código completo se muestra en el Apéndice, ya que ahora, para facilitar la lectura, mostraremos solo la parte del código más relevante.

Naturaleza de los datos, tipo de experimento y diseño experimental

Los **datos** de las muestras se han obtenido del repositorio GTEx, teniendo un total de 54 tipos de tejidos, aunque nosotros nos hemos centrado solo en los datos de expresión de **RNA-seq** (que es nuestro **tipo de experimento**) del tiroides, donde se comparan un total de 292 muestras pertenecientes a 3 grupos, aunque nosotros trabajaremos solo con 10 muestras tomadas aleatoriamente de cada grupo:

Not infiltrated tissues (NIT) → 236 muestras

Small focal infiltrates (SFI) → 42 muestras

Extensive lymphoid infiltrates (ELI) → 14 muestras

Así pues, se nos han dado **2 ficheros**, que será con los que trabajaremos: `targets.csv` y `counts.csv`. El primero (`targets.csv`) es una tabla que contiene la información por filas (SRA_Sample, Sample_Name, Grupo_analisis, body_site, molecular_data_type, sex, Group, ShortName) de cada muestra, y por tanto, hay un total de 292 filas . El segundo (`counts.csv`), es una matriz con 292 columnas, cada una de las cuales representa a una muestra, el nombre de las filas son los nombres en ENSEMBL del transcrito y los valores de las celdas son los *counts*.

Los nombres de las columnas de `counts.csv` se corresponden con la columna `Sample_Name` de `targets.csv`.

Con respecto a la fórmula para nuestro **diseño experimental**, especificaremos `~ Group`, ya que *queremos comprobar el efecto que tienen los 3 grupos.* Aunque podríamos haber hecho distintos diseños, como por ejemplo, podríamos haber diseñado el efecto del sexo sobre los 3 grupos de infiltración (~ Group + sex). Pero nosotros, por motivos de extensión, haremos solo el simple con `~ Group`.

Herramientas y métodos: Software y pipeline

Este análisis de ha llevado a cabo con la siguiente versión de `R`: *R version 4.0.3 (2020-10-10)* en un ordenador con el siguiente SO: *Platform: x86_64-pc-linux-gnu (64-bit).*

Con respecto al **pipeline** del análisis, se ha seguido el indicado en el enunciado de la PEC:

1. Definición de los datos tal como se desribe a continuación:

Escribir un script que extraiga 10 muestras aleatorias del grupo 1 (NIT), 10 del grupo 2 (SFI) y 10 del grupo 3 (ELI). Con la información de las filas escogidas hay que "subsetear" las columnas escogidas en el archivo `counts.csv`.

2. Preprocesado de los datos: filtraje y normalización
3. Identificación de genes diferencialmente expresados
4. Anotación de los resultados
5. Búsqueda de patrones de expresión y agrupación de las muestras (comparación entre las distintas comparaciones).
6. Análisis de significación biológica ("Gene Enrichment Analysis")

Entraremos más o menos en detalle de qué se ha hecho en cada paso conforme vayamos mostrando los resultados obtenidos (Resultados), ya que considero que así es más claro de explicar.

Resultados

1. Definición de los datos tal como se desribe a continuación

Escribir un script que extraiga 10 muestras aleatorias del grupo 1 (NIT), 10 del grupo 2 (SFI) y 10 del grupo 3 (ELI). Con la información de las filas escogidas hay que "subsetear" las columnas escogidas en el archivo `counts.csv`.

Para seleccionar 30 muestras aleatoriamente, 10 de cada grupo, primero leeremos los datos de las tablas de `targets` y `counts` en R.

A continuación, crearemos 3 nuevas variables correspondientes a los grupos `NIT`, `SFI` y `ELI`, separando por la columna *Group* del dataset `targets`:

El siguiente paso, es seleccionar de forma aleatoria 10 muestras para cada dataframe (`NIT`, `SFI` y `ELI`). Para ello, utilizaremos la semilla `12345` y la función `sample`, almacenando las 10 muestras

aleatorias en 3 dataframes: `nit`, `sfi` y `eli`. Así, crearemos el dataframe `targets.total.selected` uniendo los 3 dataframes anteriores por sus filas.

Después, tomaremos la columna *Sample_Name*, ya que sus filas corresponden con el nombre de las columnas de `counts`, y renombraremos las filas del dataframe `targets.total.selected`. Además, debemos cambiar los puntos (*GTEX.111FC.1026.SM.5GZX1*) que aparecen en el nombre de las columnas de `counts` por guiones (*GTEX-111FC-1026-SM-5GZX1*), como aparece en los dataframes de `targets`.

Ahora sí, podemos tomar las columnas correspondientes del dataframe `counts`. Lo haremos creando 3 dataframes, `nit.counts`, `sfi.counts` y `eli.counts`. Luego, unimos los 3 dataframes por las columnas en un dataframe llamado `counts.total.selected`. Mostramos las 3 primeras filas y 4 primeras columnas de `counts.total.selected`:

```
##                    GTEX-QEL4-0726-SM-3GIJ5 GTEX-132AR-1126-SM-5P9GA
## ENSG00000223972.4                       4                         0
## ENSG00000227232.4                     511                       907
## ENSG00000243485.2                       2                         1
##                    GTEX-ZDYS-0626-SM-5J2N5 GTEX-ZTPG-0826-SM-5DUVC
## ENSG00000223972.4                       5                         1
## ENSG00000227232.4                     637                       524
## ENSG00000243485.2                       3                         1
```

Cada fila de la matriz de counts `counts.total.selected` representa un transcrito de ENSEMBL, cada columna una muestra específica de ARN y los valores de las celdillas son el número de fragmentos que fueron únicamente asignados al gen correspondiente en la librería.

El siguiente paso es **crear el objeto** `DESeqDataSet` de la matriz de counts y con la información correspondiente de `targets`. Pero antes, asegurémonos de que el número de columnas de `counts.total.selected` coincide con el número de filas de `targets.total.selected`:

```
ncol(counts.total.selected) == nrow(targets.total.selected)
```

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

También, no debemos olvidar pasar a **factor** la variable `Group` del dataframe `targets.total.selected`, asignando *NIT*, *SFI* y *ELI* a los niveles *1*, *2* y *3*, respectivamente.

Como ya indicamos anteriormente en la subsección diseño experimental, vamos a definir la fórmula de `DESeqDataSetFromMatrix` como `design = ~ Group`.

```
class(ddsMat)
```

```
## [1] "DESeqDataSet"
## attr(,"package")
## [1] "DESeq2"
```

```
dds <- ddsMat
dds
```

```
## class: DESeqDataSet
## dim: 56202 30
```

```
## metadata(1): version
## assays(1): counts
## rownames(56202): ENSG00000223972.4 ENSG00000227232.4 ...
##   ENSG00000210195.2 ENSG00000210196.2
## rowData names(0):
## colnames(30): GTEX-QEL4-0726-SM-3GIJ5 GTEX-132AR-1126-SM-5P9GA ...
##   GTEX-13QJC-0826-SM-5RQKC GTEX-YJ89-0726-SM-5P9F7
## colData names(8): SRA_Sample Sample_Name ... Group ShortName
```

Un aspecto importante, es que no tenemos los datos necesarios para crear un `SummarizedExperiment`, ya que por ejemplo, no tenemos las *matrices BAM*. Pero hemos podido crear un `DESeqDataSet` a partir de los *counts* y *targets*. La diferencia está, en que no podremos visualizar el *Plot de fold changes en el espacio genómico*.

2. Preprocesado de los datos: filtraje y normalización

Prefiltering

Primero, vamos a eliminar aquellas filas de nuestro dataset que no tengan counts y que por tanto, no aporten información.

Por lo que pasamos de tener 56202 filas a tener 43573 filas.

Transformación estabilizadora de la varianza y el rlog

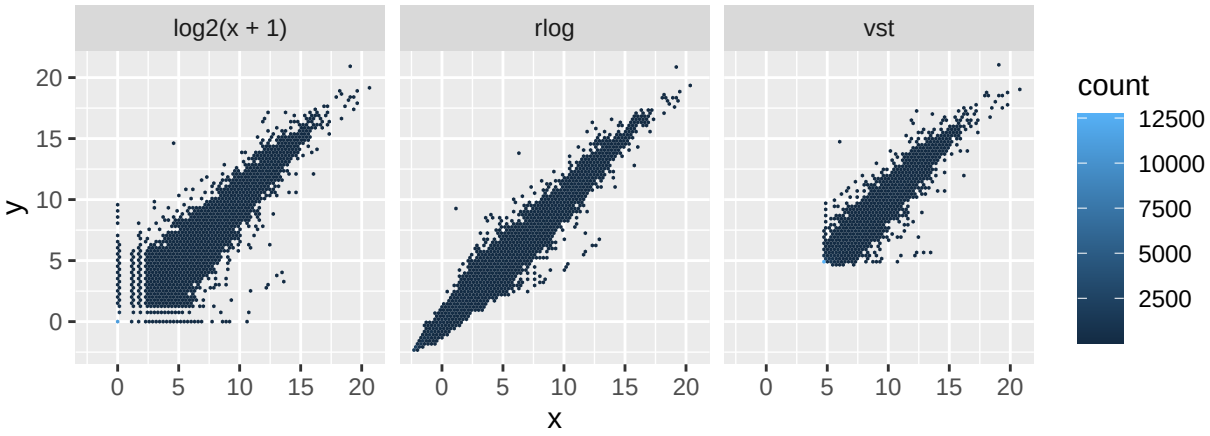
Esta transformación es importante porque muchos métodos estadísticos comunes para el análisis exploratorio de datos multidimensionales (como clustering o PCA), funcionan mejor para datos que tienen el mismo rango de varianza en diferentes rangos de los valores medios.

Como nuestro dataset es de `n=30` porque estamos trabajando con 30 muestras, podríamos utilizar en realidad tanto `vst` como `rlog`. Nos quedaremos con el primero porque es más rápido. Tando `vst` como `rlog` devuelven un objeto `DESeqTransform` que se basa en la clase `SummarizedExperiment.` Por otro lado, en la función `vst()`, especificamos `blind = FALSE` para que las diferencias entre los 3 grupos de infiltración no contribuyan a la tendencia media de la varianza esperada del experimento.

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

```
## class: DESeqTransform
## dim: 43573 30
## metadata(1): version
## assays(1): ''
## rownames(43573): ENSG00000223972.4 ENSG00000227232.4 ...
##   ENSG00000210195.2 ENSG00000210196.2
## rowData names(4): baseMean baseVar allZero dispFit
## colnames(30): GTEX-QEL4-0726-SM-3GIJ5 GTEX-132AR-1126-SM-5P9GA ...
##   GTEX-13QJC-0826-SM-5RQKC GTEX-YJ89-0726-SM-5P9F7
## colData names(9): SRA_Sample Sample_Name ... ShortName sizeFactor
```

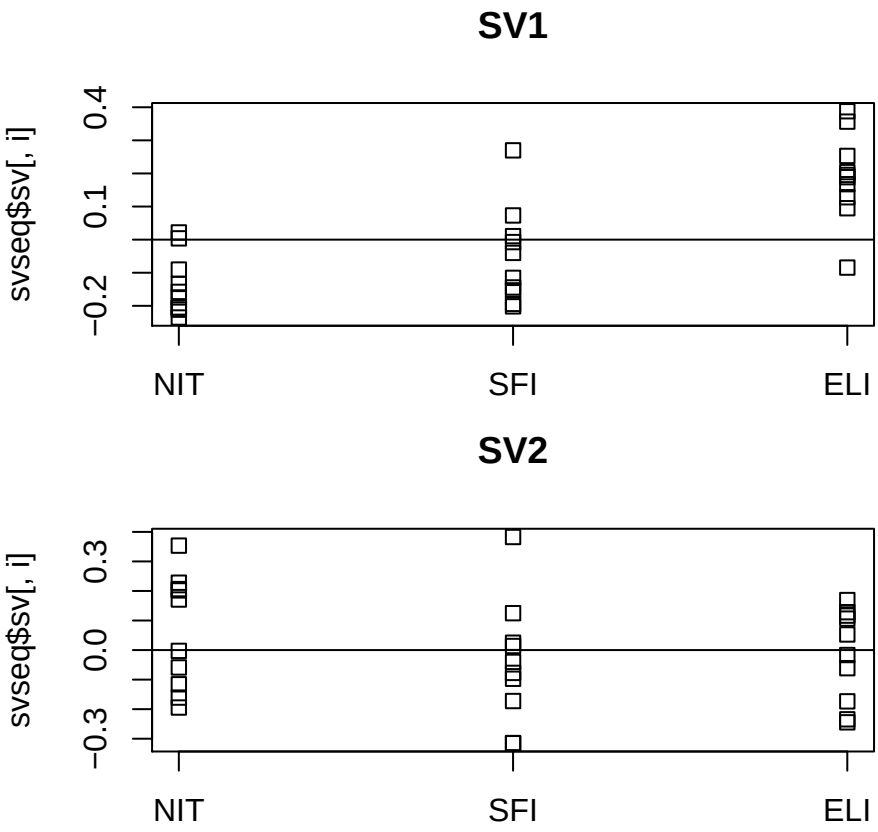
A continuación, mostramos un plot con 3 figuras, de izquierda a derecha: transformación log normal, transformación rlog y transformacuón vst. Las muestras que se representan son la primera (perteneciente al grupo *NIT*) vs la número 11 (perteneciente al grupo *SFI*).



Podemos ver cómo los genes con recuentos bajos (esquina inferior izquierda) parecen ser excesivamente variables en la escala *logarítmica*, mientras que en *rlog* y *vst*, los genes que tienen entre 5 y 13 counts se ven con más dispersión. Esto puede ser un indicio sobre la expresión diferencial que hay entre estas dos muestras de grupos distintos.

Eliminación de efectos Batch ocultos

Utilizaremos la función `SVA` de `DESeq2`:



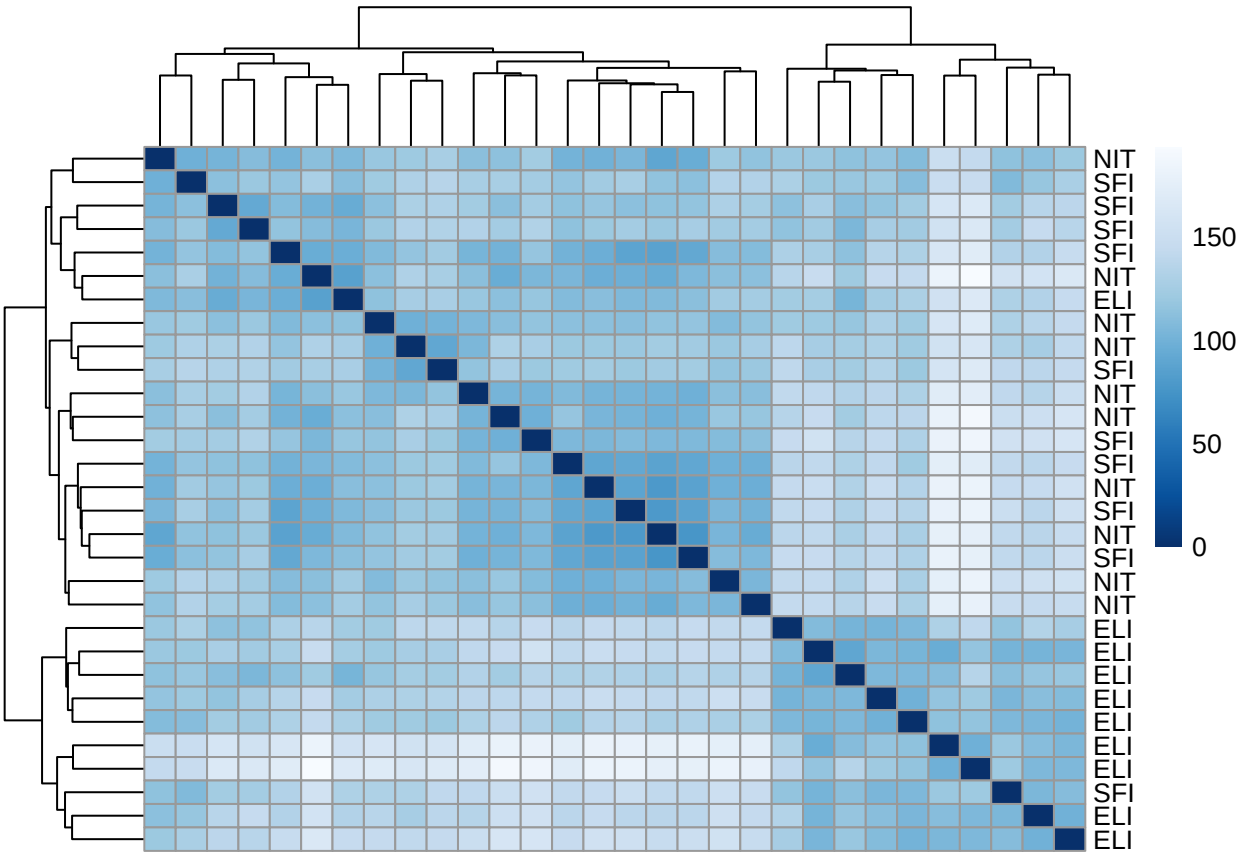
Estos gráficos muestran los resultados de la función SVA, el primero se corresponde con la primera variable surrogada, y el segundo gráfico, con la segunda. En realidad, no tendremos en cuenta estos efectos Batch para el resto del análisis, solo los hemoos realizado para mostrar que también se pueden analizar y remover estos efectos.

3. Identificación de genes diferencialmente expresados

Previsualización de plots

Distancia entre las muestras

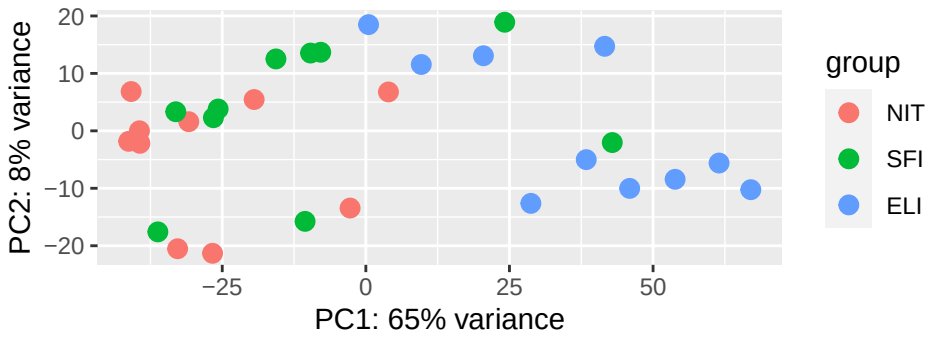
Evaluar la similitud general entre las muestras es un primer paso útil en los análisis de RNA-seq. Por lo tanto, vamos a visualizar la distancia entre las muestras en el siguiente heatmap, utilizando la función `pheatmap`:



En el plot podemos ver que mientras que `NIT` y `SFI` aparecen más o menos intercaladas, `ELI` parece estar un poco más alejada. Entonces se intuye que hay como 2 clústeres principales: uno con las muestras de `NIT` y `SFI`, y otro con las de `ELI`.

PCA plot

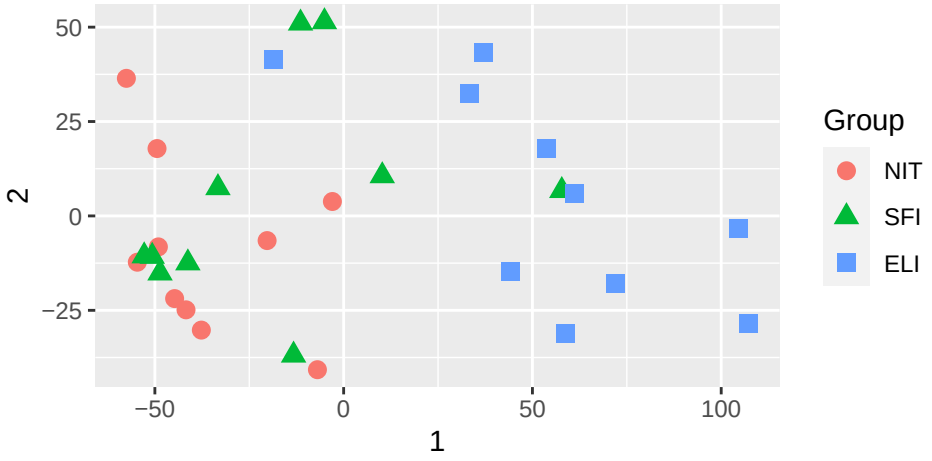
Vamos a utilizar los datos obtenidos con `vst`.



Mediante este gráfico del análisis de componentes principales, se sugiere lo mismo que hemos visto con el *heatmap*: hay menor diferencia entre las muestras de los grupos de `NIT` y `SFI`, que con el grupo `ELI`.

MDS plot

Este plot es útil cuando no tenemos una matriz de datos, pero sí una matriz de distancias. Vamos a hacer el plot con los datos obtenidos con `vst`:



Así pues, vemos lo mismo que mostraban los dos plots anteriores: mayor similitud entre `NIT` y `SFI`, que con el grupo `ELI`.

Análisis de expresión diferencial

Para ejecutar este análisis, utilizamos la función `DESeq` sobre el objeto *DESeqDataSet* `dds`.

Como queremos comparar los grupos **NIT vs SFI**, **NIT vs ELI** y **SFI vs ELI**, vamos a crear 3 variables: `res1`, `res2` y `res3` (`res1` para el grupo 1 **NIT vs SFI**, `res2` para el grupo 2 **NIT vs ELI** y `res3` para el 3 **SFI vs ELI**), con los resultados de los contrastes de estas comparaciones.

Veamos los metadata para `res1`, `res2` y `res3`, así como el resumen de los mismos:

- Para `res1`:

```
mcols(res1, use.names = TRUE)
```

```
## DataFrame with 6 rows and 2 columns
##                         type            description
##                  <character>            <character>
## baseMean       intermediate mean of normalized c..
## log2FoldChange      results log2 fold change (ML..
## lfcSE               results standard error: Grou..
## stat                results Wald statistic: Grou..
## pvalue              results Wald test p-value: G..
## padj                results   BH adjusted p-values
```

```
summary(res1)
```

```
##
## out of 43573 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 4, 0.0092%
## LFC < 0 (down)     : 90, 0.21%
## outliers [1]       : 0, 0%
## low counts [2]     : 15206, 35%
## (mean count < 2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

- Para res2:

```
mcols(res2, use.names = TRUE)
```

```
## DataFrame with 6 rows and 2 columns
##                         type            description
##                  <character>            <character>
## baseMean       intermediate mean of normalized c..
## log2FoldChange      results log2 fold change (ML..
## lfcSE               results standard error: Grou..
## stat                results Wald statistic: Grou..
## pvalue              results Wald test p-value: G..
## padj                results   BH adjusted p-values
```

```
summary(res2)
```

```
##
## out of 43573 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 2425, 5.6%
## LFC < 0 (down)     : 4210, 9.7%
## outliers [1]       : 0, 0%
```

```
## low counts [2]     : 12672, 29%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

- Para `res3`:

```
mcols(res3, use.names = TRUE)
```

```
## DataFrame with 6 rows and 2 columns
##                         type            description
##                  <character>            <character>
## baseMean       intermediate mean of normalized c..
## log2FoldChange      results log2 fold change (ML..
## lfcSE               results standard error: Grou..
## stat                results Wald statistic: Grou..
## pvalue              results Wald test p-value: G..
## padj                results   BH adjusted p-values
```

```
summary(res3)
```

```
##
## out of 43573 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 1757, 4%
## LFC < 0 (down)     : 3008, 6.9%
## outliers [1]       : 0, 0%
## low counts [2]     : 14361, 33%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Si aceptamos una fracción del 10% de falsos positivos, podemos considerar significativos todos los genes con un adj p-valor por debajo del 10% (es decir, 0.1). Por lo tanto, para `res1` habrá 94 genes significativos, para `res2`, 6635 y para `res3`, 4765.

Por otro lado, también podemos ordenar los genes según el *log2 fold change* y ver cuáles son los que están más fuertemente regulados *aguas abajo*:

```
## log2 fold change (MLE): Group NIT vs SFI
##
## DataFrame with 6 rows and 3 columns
##                     baseMean log2FoldChange       padj
##                    <numeric>      <numeric>  <numeric>
## ENSG00000110680.8 1724.6567       -6.46454 0.00531105
## ENSG00000211672.2   33.5237       -5.97867 0.03968745
## ENSG00000270999.1   12.5949       -5.49015 0.00833464
## ENSG00000253998.2   85.1844       -5.45443 0.00531105
## ENSG00000211685.2  996.1638       -5.32229 0.02491453
## ENSG00000254709.2   95.1695       -5.26934 0.01292136
```

```
## log2 fold change (MLE): Group NIT vs ELI
##
## DataFrame with 6 rows and 3 columns
##                     baseMean log2FoldChange        padj
##                    <numeric>      <numeric>   <numeric>
## ENSG00000170054.10   37.6228       -8.61348 2.72501e-10
## ENSG00000181617.5   237.7016       -7.99629 3.09257e-11
## ENSG00000211672.2    33.5237       -7.79819 2.19449e-06
## ENSG00000254029.1    16.6763       -7.74534 7.69570e-03
## ENSG00000253441.1    16.4033       -7.45019 9.68652e-06
## ENSG00000235304.1    34.4027       -7.36358 5.88892e-12
```

```
## log2 fold change (MLE): Group SFI vs ELI
##
## DataFrame with 6 rows and 3 columns
##                     baseMean log2FoldChange        padj
##                    <numeric>      <numeric>   <numeric>
## ENSG00000170054.10   37.6228       -8.48284 1.44219e-09
## ENSG00000257275.2    17.2702       -7.93142 2.72405e-06
## ENSG00000100721.6   292.8806       -7.32754 8.08111e-09
## ENSG00000250850.2    11.4487       -6.74841 8.59662e-12
## ENSG00000160505.11   13.3855       -6.41921 2.72905e-06
## ENSG00000196092.8   397.7885       -6.40486 9.10867e-09
```

Por ejemplo, vemos que el gen `ENSG00000170054.10` coincide para los grupos `NIT vs ELI` y `SFI vs ELI`.

Y los mayores *aguas arriba*:

```
## log2 fold change (MLE): Group NIT vs SFI
##
## DataFrame with 6 rows and 3 columns
##                     baseMean log2FoldChange      padj
##                    <numeric>      <numeric> <numeric>
## ENSG00000143552.5    85.6165       3.156950 0.0206594
## ENSG00000261857.2   117.1133       2.624284 0.0377188
## ENSG00000173432.6   155.9475       2.491966 0.0495067
## ENSG00000234617.1   130.7233       0.821906 0.0860351
## ENSG00000128578.5   113.8785      -0.973630 0.0670929
## ENSG00000029534.15  133.4000      -1.153664 0.0950242
```

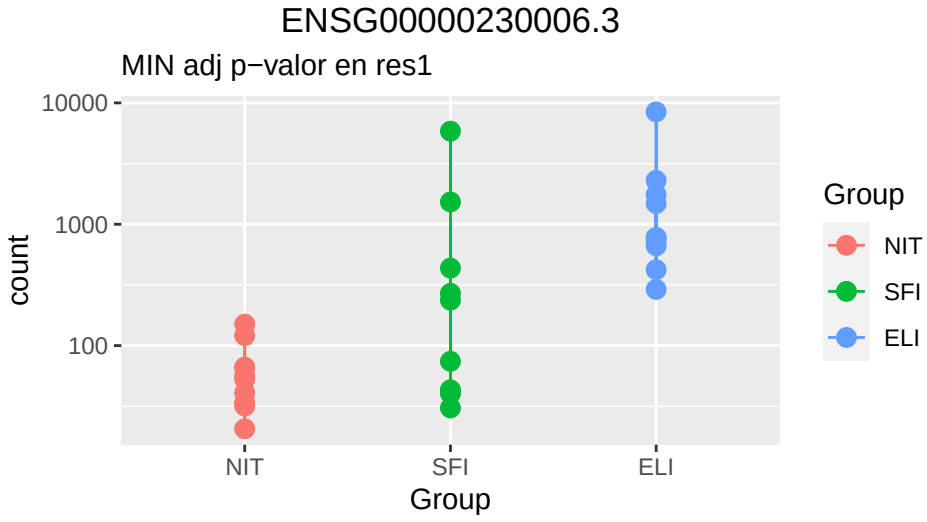
```
## log2 fold change (MLE): Group NIT vs ELI
##
## DataFrame with 6 rows and 3 columns
##                     baseMean log2FoldChange        padj
##                    <numeric>      <numeric>   <numeric>
## ENSG00000253301.1   5.76323        4.87524 7.19797e-03
## ENSG00000243584.1   4.64279        4.83889 6.96753e-05
## ENSG00000170419.6  23.09641        4.64157 9.37796e-04
## ENSG00000236848.2  14.96493        4.15753 1.31032e-03
## ENSG00000215873.2   1.49505        4.08412 3.41144e-03
```

```
## ENSG00000198414.5   1.25629        3.83139 3.62785e-02
```

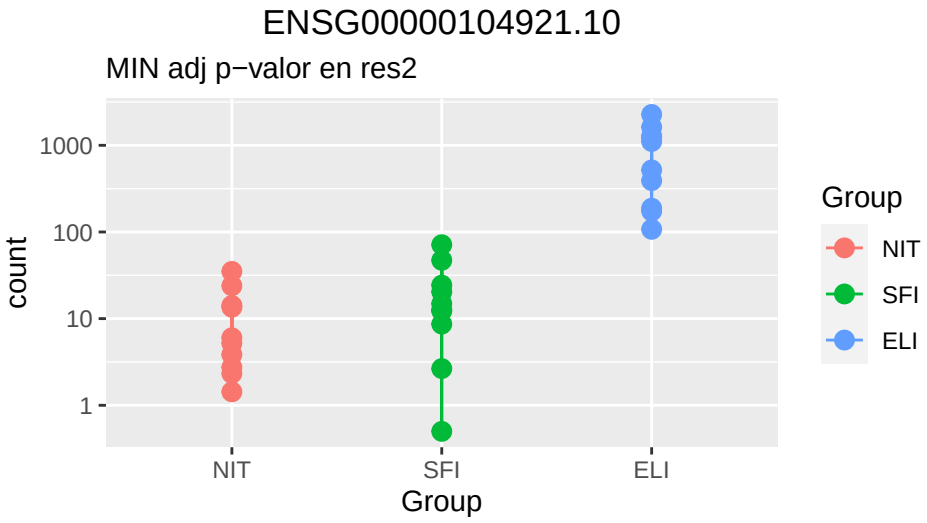
```
## log2 fold change (MLE): Group SFI vs ELI
##
## DataFrame with 6 rows and 3 columns
##                     baseMean log2FoldChange        padj
##                    <numeric>      <numeric>   <numeric>
## ENSG00000110680.8 1724.65666        8.17771 5.63731e-08
## ENSG00000157005.3   14.55340        4.70182 5.56465e-03
## ENSG00000253301.1    5.76323        4.51005 1.94559e-02
## ENSG00000145808.4    2.26807        4.41983 7.36914e-03
## ENSG00000100604.8   61.92715        4.28814 5.87808e-05
## ENSG00000128564.5   34.60913        3.91980 4.17845e-05
```

Counts plot

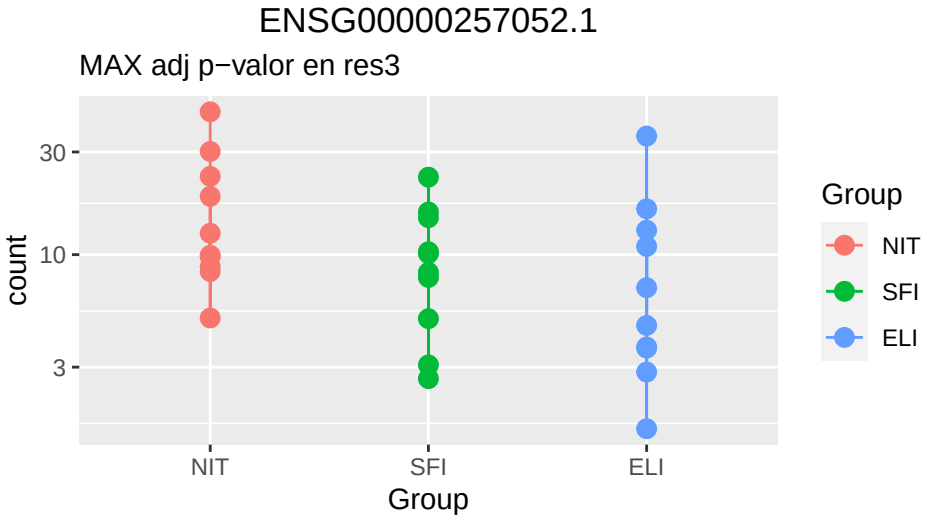
La función `plotCounts` es una forma rápida de visualizar los recuentos de un gen en particular.



En este caso, estamos visualizando los counts del gen ENSG00000230006.3 para los 3 grupos. Este gen es el que tiene el adj p-valor menor dentro de la comparación `res1` (*NIT vs SFI*). De hecho, vemos que hay una gran diferencia en un orden de casi 8.000 entre los counts de ese gen para los grupos `NIT` y `SFI`.

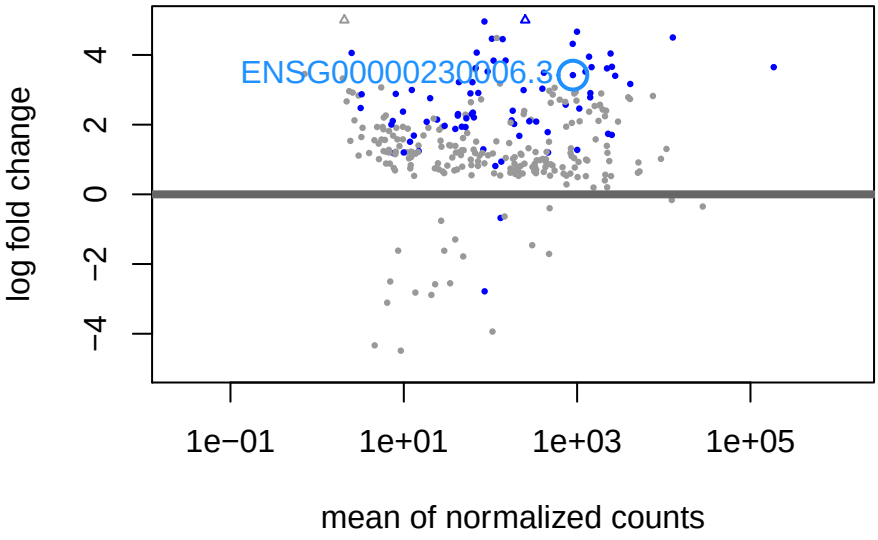


Ahora vemos los counts del gen ENSG00000104921.10 para los 3 grupos. Este gen es el que tiene el adj p-valor menor dentro de la comparación `res2` (*NIT vs ELI*). Y en efecto, vemos que hay bastante diferencia en un orden de casi 1.000 entre los counts de ese gen para los grupos `NIT` y `ELI`.



En este gráfico se representan los counts del gen ENSG00000257052.1 para los 3 grupos. Este gen es el que tiene el adj p-valor *mayor* dentro de la comparación `res3` (*SFI vs ELI*). Así pues, vemos que no hay mucha diferencia en un orden de aproximadamente 5 entre los counts de ese gen para los grupos `SFI` y `ELI`.

MA-Plot



Este plot proporciona una descripción general útil para la distribución de los coeficientes estimados en el modelo, como se describe en los ejes del gráfico. Hemos utilizado el método `apeglm` para reducir los coeficientes del grupo *Group_SFI_vs_NIT*. Además, en el gráfico se señala el transcrito con el adj p-valor más pequeño. De hecho, coincide con el primer gráfico del counts plot, cuyo gen con menor adj p-valor es el mismo que el que nos muestra este MA-plot 5370.

4. Anotación de los resultados

Utilizaremos `org.Hs.eg.db` (Genome wide annotation for Human link).

Un detalle importante, es que nuestros transcritos (genes) aparecen con un punto, que corresponde a la versión, por poner un ejemplo: *ENSG00000104921.10*. Entonces, debemos quitar el `.` y lo que haya detrás de él para que funcione el KEY con ENSEMBL (es decir, que el transcrito se quede así: *ENSG00000104921*).

Mostremos las 6 primeras filas y 5 columnas para `res1`:

```
## log2 fold change (MLE): Group NIT vs SFI
##
## DataFrame with 6 rows and 5 columns
##                     baseMean log2FoldChange       padj      symbol      entrez
##                    <numeric>      <numeric>  <numeric> <character> <character>
## ENSG00000230006.3    892.9027       -3.75311 0.00531105  ANKRD36BP2      645784
## ENSG00000253998.2     85.1844       -5.45443 0.00531105    IGKV2-29       28920
## ENSG00000110680.8   1724.6567       -6.46454 0.00531105       CALCA         796
## ENSG00000211892.2  12713.5842       -4.93746 0.00531105       IGHG4        3503
## ENSG00000088340.11  1415.7075       -3.06424 0.00546962      FER1L4       80307
## ENSG00000251039.2    103.7949       -4.95094 0.00814208   IGKV2D-40       28878
```

Ahora aparecen las columnas `symbol` y `entrez`.

Para `res2`:

```
## log2 fold change (MLE): Group NIT vs ELI
```

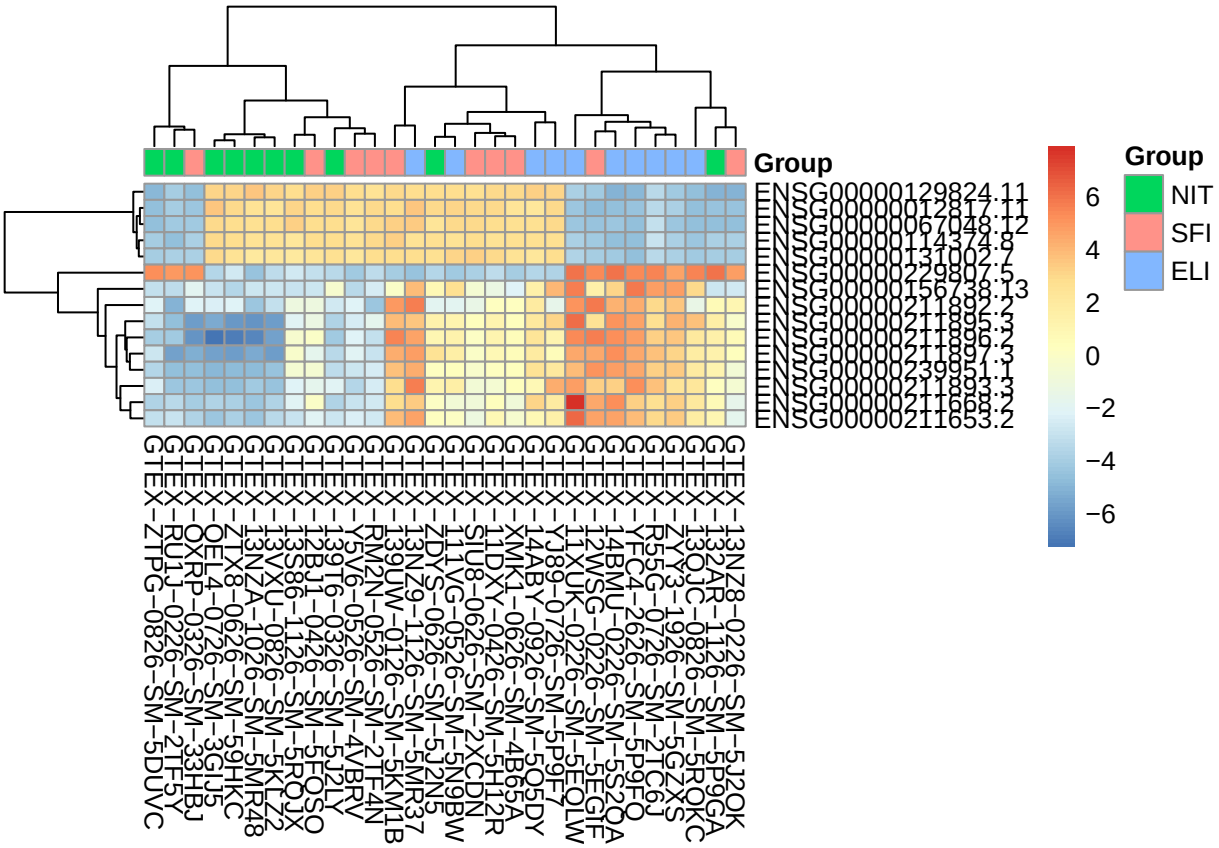
```
##
## DataFrame with 6 rows and 5 columns
##                     baseMean log2FoldChange        padj      symbol      entrez
##                    <numeric>      <numeric>   <numeric> <character> <character>
## ENSG00000104921.10   305.846       -6.40286 1.86891e-20       FCER2        2208
## ENSG00000174123.6    407.421       -6.02227 5.42531e-19       TLR10       81793
## ENSG00000247982.2   1531.019       -3.59794 1.16054e-18   LINC00926      283663
## ENSG00000175857.4    170.051       -4.31479 1.27463e-18        GAPT      202309
## ENSG00000172794.15   412.612       -3.98666 1.35593e-18       RAB37      326624
## ENSG00000245164.2    908.374       -4.76939 1.47264e-18   LINC00861   100130231
```

Para `res3`:

```
## log2 fold change (MLE): Group SFI vs ELI
##
## DataFrame with 6 rows and 5 columns
##                     baseMean log2FoldChange        padj      symbol      entrez
##                    <numeric>      <numeric>   <numeric> <character> <character>
## ENSG00000175857.4    170.051       -4.23265 1.25482e-17        GAPT      202309
## ENSG00000269404.2    312.440       -5.26454 1.25482e-17        SPIB        6689
## ENSG00000247982.2   1531.019       -3.42627 9.53196e-17   LINC00926      283663
## ENSG00000167483.13   761.903       -5.27346 5.03891e-16      NIBAN3      199786
## ENSG00000068831.14  3445.527       -3.15292 1.05828e-15     RASGRP2       10235
## ENSG00000104921.10   305.846       -5.40543 4.23989e-15       FCER2        2208
```

5. Búsqueda de patrones de expresión y agrupación de las muestras (comparación entre las distintas comparaciones)

En realidad, con respecto a este apartado, ya hemos comparado en la sección anterior entre distintas comparaciones entre genes diferencialmente expresados, pero lo que aún no hemos hecho es un clúster con los genes con mayor variabilidad. Por tanto, representemos el clúster con la función `pheatmap` con los 15 genes más altamente variables. Volveremos a trabajar con los datos de `vst`.



Por tanto, a la derecha vemos el nombre de los 15 genes (transcritos) con mayor variabilidad. Arriba, aparecen en verde los pertenecientes al grupo *NIT*, los del grupo *SFI* aparecen en color salmón y los del grupo *ELI*, en azul. Abajo, vemos el nombre de las 30 muestras. La representación de los genes en las muestras aparecen en colores entre naranja y azul. Así, los genes pueden dividirse en dos grupos: el primero abarca los 5 primeros genes, y el segundo grupo, los 10 genes restantes. Con respecto a los 3 grupos de infiltración, vemos que para los 5 primeros genes, los grupos *NIT* y *SFI* coinciden bastante con respecto a la representación de la variabilidad de estos genes (color naranja), mientras que el grupo *ELI*, sigue un patrón diferente (color azul). A partir del gen número 6 y en adelante, esta similitud entre los grupos *NIT* y *SFI* no se ve tan clara, pero para el grupo *ELI* sí que se observa un cambio de patrón (color rojo-naranja-amarillo).

6. Análisis de significación biológica ("Gene Enrichment Analysis")

Como ya vimos en la PEC anterior, el análisis de significación biológica estudia las funciones de los genes buscando sus anotaciones en bases de datos de anotación funcional.

El primer paso es preparar la lista de genes que analizaremos. Para ello, tomaremos los 35 primeros genes con el adj p-valor significativo más alto de las comparaciones `res1`, `res2` y `res3`.

El siguiente paso será crear el universo, para el cual definiremos que contenga todos los genes que tengan al menos una anotación en Gene Ontology. Estamos utilizando los paquetes de anotaciones de organismos de Gene Ontology (`org.Hs.egGO`) y de rutas metabólicas (`org.Hs.egPATH`).

Ya podemos usar el paquete `ReactomePA` para hacer el análisis de significación biológica sobre las 3 comparaciones **NIT vs SFI**, **NIT vs ELI** y **SFI vs ELI**.

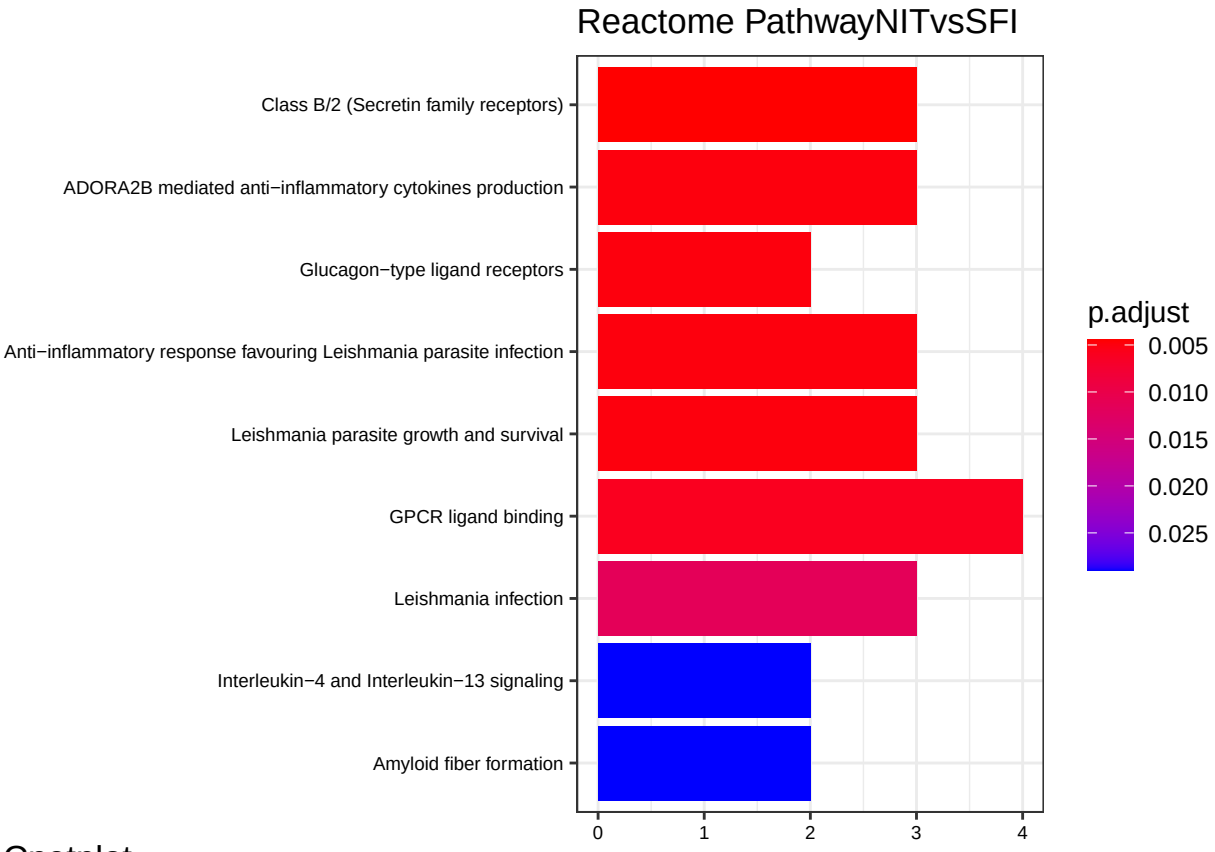
- Primer grupo **NIT vs SFI**. Vamos a mostrar el *head* de todos los resultados obtenidos con `enrichPathway` solo para esta comparación, para las demás, por motivos de extensión, mostraremos solo la *descripción* y el *geneID*.

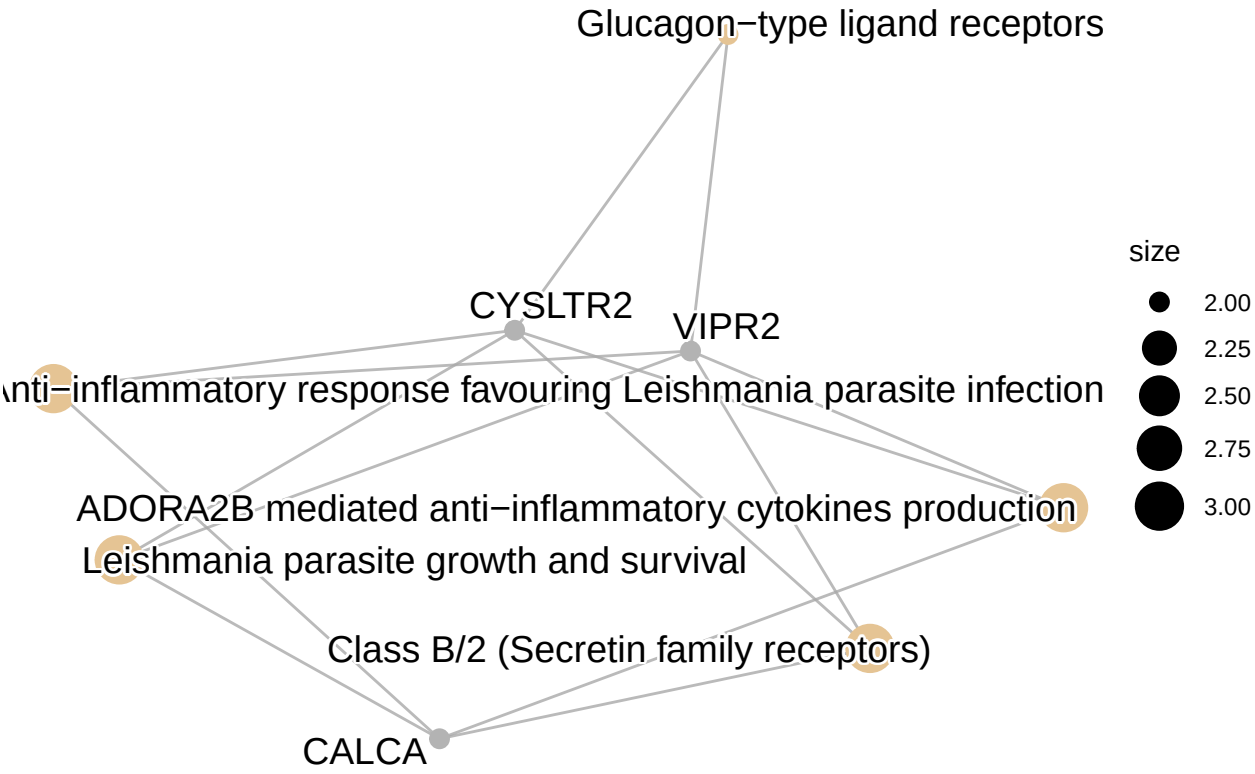
```
## ####################################

##
## Comparison:  NITvsSFI

## ####################################

##                          ID
## R-HSA-373080   R-HSA-373080
## R-HSA-9660821 R-HSA-9660821
## R-HSA-420092   R-HSA-420092
## R-HSA-9662851 R-HSA-9662851
## R-HSA-9664433 R-HSA-9664433
## R-HSA-500792   R-HSA-500792
##                                                                  Description
## R-HSA-373080                                 Class B/2 (Secretin family receptors)
## R-HSA-9660821            ADORA2B mediated anti-inflammatory cytokines production
## R-HSA-420092                                      Glucagon-type ligand receptors
## R-HSA-9662851 Anti-inflammatory response favouring Leishmania parasite infection
## R-HSA-9664433                              Leishmania parasite growth and survival
## R-HSA-500792                                                  GPCR ligand binding
##               GeneRatio   BgRatio       pvalue    p.adjust      qvalue
## R-HSA-373080       3/10  94/10668 7.602505e-05 0.004485478 0.002720896
## R-HSA-9660821      3/10 133/10668 2.132439e-04 0.005008010 0.003037865
## R-HSA-420092       2/10  33/10668 4.111655e-04 0.005008010 0.003037865
## R-HSA-9662851      3/10 168/10668 4.244076e-04 0.005008010 0.003037865
## R-HSA-9664433      3/10 168/10668 4.244076e-04 0.005008010 0.003037865
## R-HSA-500792       4/10 466/10668 6.116321e-04 0.006014383 0.003648332
##                                  geneID Count
## R-HSA-373080       CALCA/VIPR2/CYSLTR2     3
## R-HSA-9660821      CALCA/VIPR2/CYSLTR2     3
## R-HSA-420092             VIPR2/CYSLTR2     2
## R-HSA-9662851      CALCA/VIPR2/CYSLTR2     3
## R-HSA-9664433      CALCA/VIPR2/CYSLTR2     3
## R-HSA-500792  CALCA/VIPR2/CYSLTR2/SAA1     4
```


Reactome PathwayNITvsSFI
Class B/2 (Secretin family receptors)
ADORA2B mediated anti−inflammatory cytokines production
Glucagon−type ligand receptors
Anti−inflammatory response favouring Leishmania parasite infection
Leishmania parasite growth and survival
GPCR ligand binding
Leishmania infection
Interleukin−4 and Interleukin−13 signaling
Amyloid fiber formation
0
1
2
3
4
p.adjust
0.005
0.010
0.015
0.020
0.025


Cnetplot
NITvsSFI
Glucagon−type ligand receptors
CYSLTR2
VIPR2
nti−inflammatory response favouring Leishmania parasite infection
ADORA2B mediated anti−inflammatory cytokines production
Leishmania parasite growth and survival
Class B/2 (Secretin family receptors)
CALCA
size
2.00
2.25
2.50
2.75
3.00

Para esta primera comparación **NIT vs SFI** hay 3 genes que se repiten, que son: `CALCA`, `VIPR2` y `CYSLTR2`. Estos genes están implicados en las siguientes funciones:

Receptores de ligandos de tipo glucagón

Respuesta antiinflamatoria que favorece la infección por parásitos Leishmania

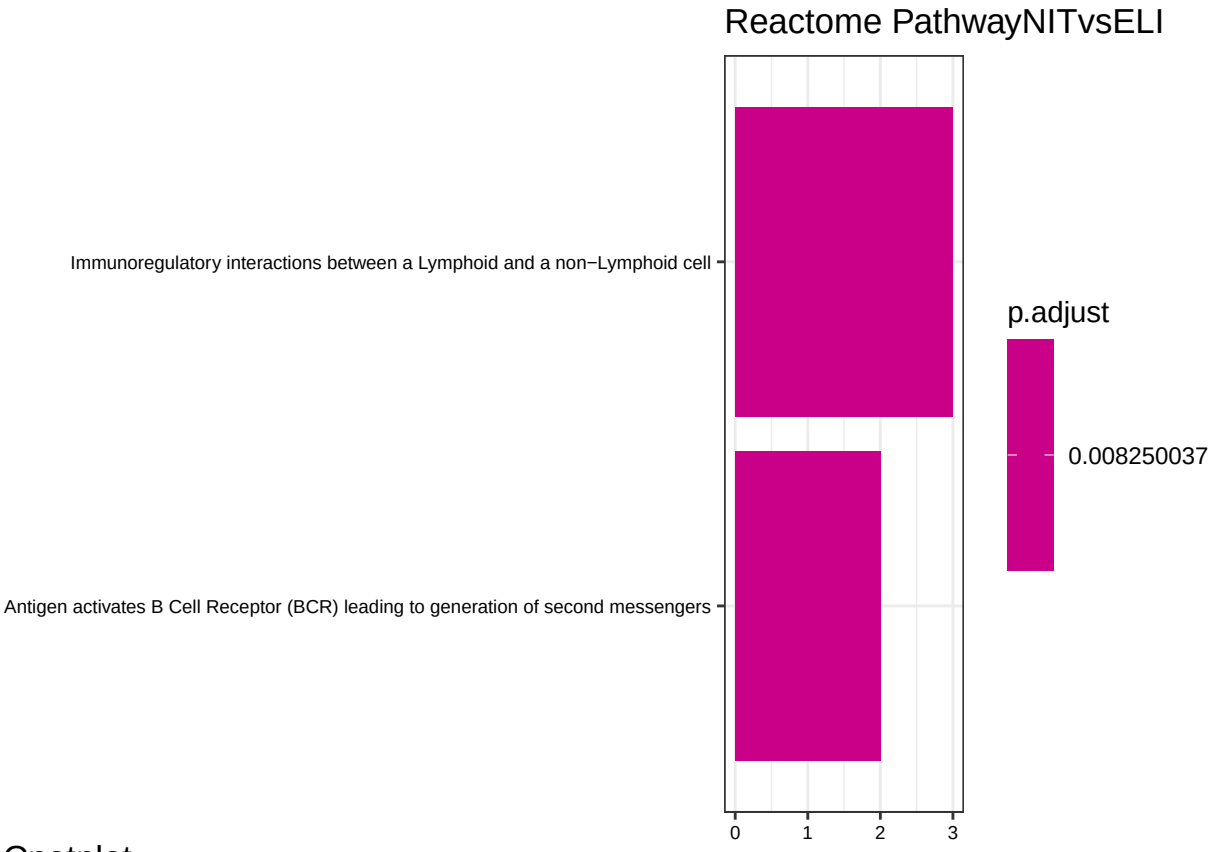
Producción de citocinas antiinflamatorias mediada por ADORA2B

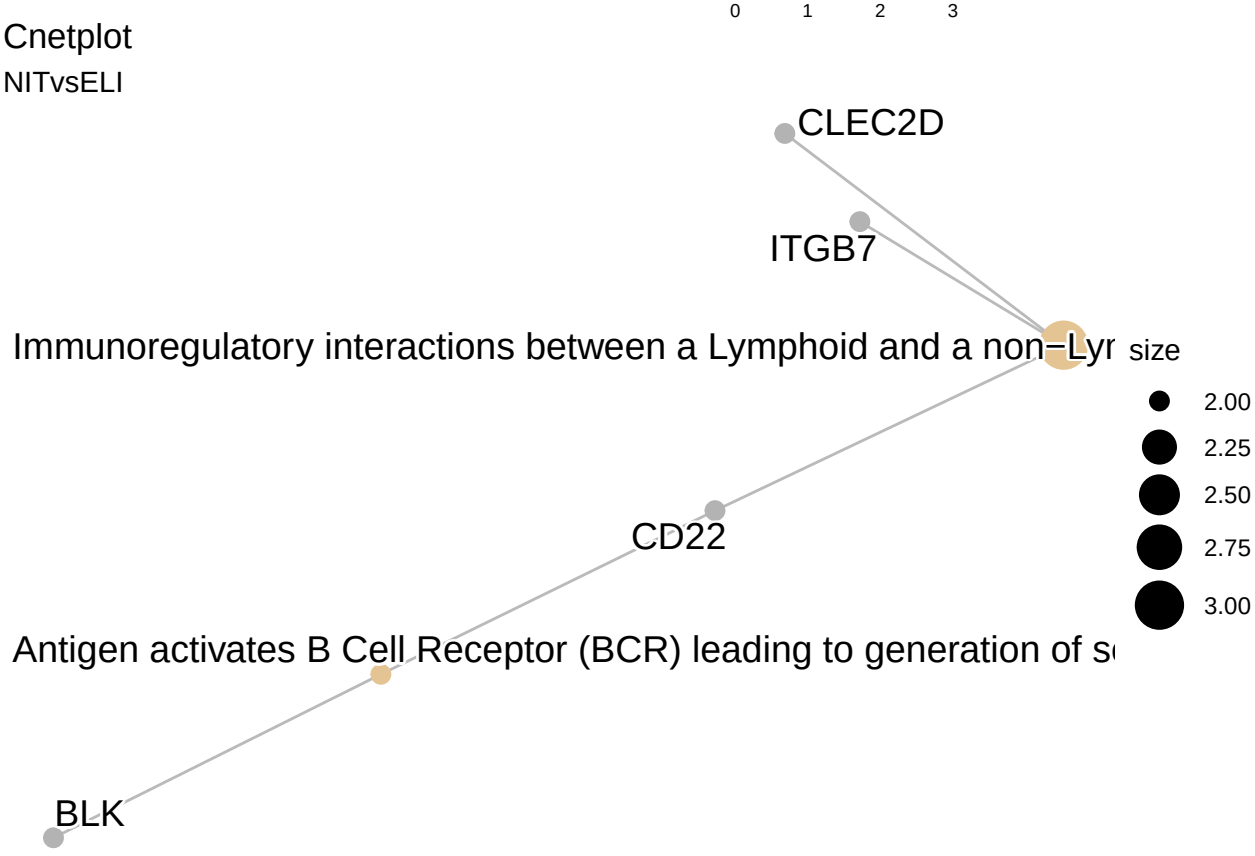
Crecimiento y supervivencia del parásito Leishmania

Clase B/2 (receptores de la familia de la secretina)

- Segundo grupo **NIT vs ELI**. Ahora, por motivos de extensión, mostraremos solo el *head* de la *descripción* y el *geneID*:

```
## ###################################
##
## Comparison:  NITvsELI
## ###################################
## [1] "Immunoregulatory interactions between a Lymphoid and a non-Lymphoid cell"
## [2] "Antigen activates B Cell Receptor (BCR) leading to generation of second messengers"
## [3] "Signaling by the B Cell Receptor (BCR)"
## [4] "RUNX1 and FOXP3 control the development of regulatory T lymphocytes (Tregs)"
## [5] "NOTCH2 intracellular domain regulates transcription"
## [6] "Rap1 signalling"
## [1] "CLEC2D/CD22/ITGB7" "CD22/BLK"          "CD22/BLK"
## [4] "CTLA4"             "FCER2"             "RASGRP2"
```





Esta vez se comparan los grupos **NIT vs ELI**, donde los genes que aparecen implicados en el

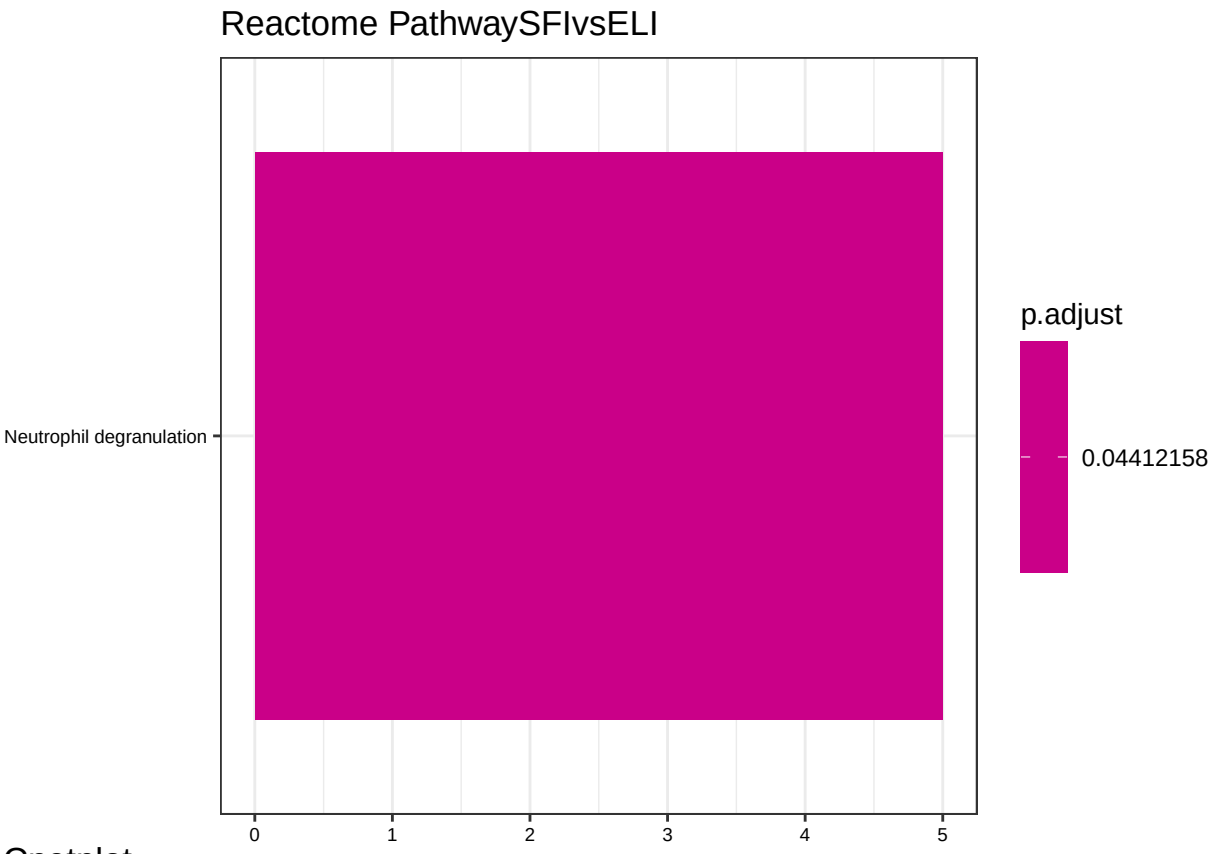
cnetplot son `CLEC2D`, `CD22`, `ITGB7` y `BLK`. Estos genes están relacionados con:

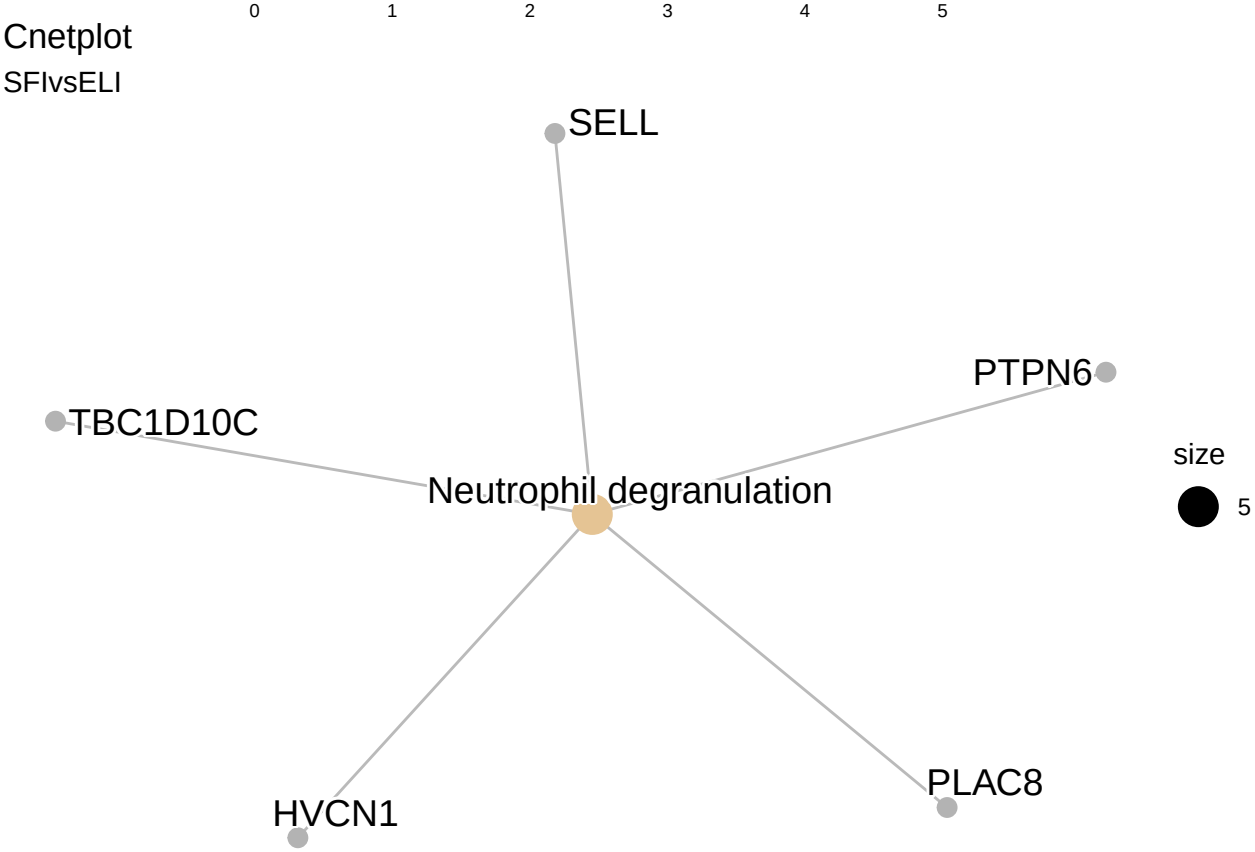
Interacciones inmunorreguladoras entre una célula linfoide y una no linfoide

El antígeno que activa el receptor de células B (BCR), generando segundos mensajeros

- Tercer grupo **SFI vs ELI**:

```
## ####################################
##
## Comparison:  SFIvsELI
## ####################################
##                                                          Description
## R-HSA-6798695                              Neutrophil degranulation
## R-HSA-9007101                          Rab regulation of trafficking
## R-HSA-202733       Cell surface interactions at the vascular wall
## R-HSA-210990                                     PECAM1 interactions
## R-HSA-2197563 NOTCH2 intracellular domain regulates transcription
## R-HSA-8851680              Butyrophilin (BTN) family interactions
##                                        geneID
## R-HSA-6798695 PTPN6/SELL/TBC1D10C/HVCN1/PLAC8
## R-HSA-9007101                 TBC1D10C/DENND1C
## R-HSA-202733                        PTPN6/SELL
## R-HSA-210990                             PTPN6
## R-HSA-2197563                            FCER2
## R-HSA-8851680                           BTN3A1
```





Finalmente, con respecto a la comparación de los grupos **SFI vs ELI**, los genes `PTPN6`, `SELL`,

`TBC1D10C`, `HVCN1` y `PLAC8` están implicados en la *desgranulación de neutrófilos.*

Discusión

Como a lo largo del informe hemos ido comentando los resultados, los resumiremos brevemente y responderemos a nuestras preguntas planteadas en los Objetivos.

En general, hemos visto que los grupos de infiltración `NIT` y `SFI` tienden a tener una menor distancia entre las muestras, según se aprecia en el primer heatmap. Por tanto, entre esos dos grupos hay menos diferencia entre las muestras, como señalan los plots PCA y el MDS. El clúster muestra las ditancias, similitudes y diferencias en la expresión de los 15 genes significativos con mayor variabilidad. Por otro lado, con respecto al análisis de expresión diferencial de genes, vimos distintos genes entre los grupos, como en los counts plots o en el MA-plots. Finalmente, aprovecharemos los resultados del análisis de significación biológica, para el cual utilizamos los **35 genes con mayor adj p-valor significativo expresados diferencialmente entre los grupos de infiltración correspondientes**, para responder a las siguientes cuestiones:

- ¿Hay diferencia de expresión de genes en tejidos de tiroides sin infiltraciones vs tejidos de tiroides con infiltraciones focales? → **NIT vs SFI**

Hay expresión diferencial entre en los genes `CALCA`, `VIPR2` y `CYSLTR2` cuando comparamos **NIT vs SFI** (tejidos sin infiltración vs tejidos con infiltración focal). Además, llama la atención que algunos de los genes están relacionados con la respuesta inflamatoria del parásito Leishmania. Otras funciones de los genes serían la producción de citoquinas antiinflamatorias, receptores de ligandos de tipo glucagón y receptores de la familia de la secretina.

- ¿Hay diferencia de expresión de genes en tejidos de tiroides sin infiltraciones vs tejidos de tiroides con infiltraciones extensas? → **NIT vs ELI**

Hay expresión diferencial entre en los genes `CLEC2D`, `CD22`, `ITGB7` y `BLK` cuando comparamos **NIT vs ELI** (tejidos sin infiltración vs tejidos con infiltraciones extensas). Estos genes están relacionados con interacciones inmunorreguladoras entre células linfoides y no linfoides y con el antígeno que activa el receptor de células B.

- ¿Hay diferencia de expresión de genes en tejidos de tiroides con infiltraciones focales vs tejidos de tiroides con infiltraciones extensas? → **SFI vs ELI**

Hay expresión diferencial entre los genes `PTPN6`, `SELL`, `TBC1D10C`, `HVCN1` y `PLAC8` con comparación **SFI vs ELI** (tejidos con infiltración focal vs tejidos con infiltraciones extensas). Estos genes están implicados en la desgranulación de neutrófilos.

Apéndice

A continuación, se muestra el **código** empleado.

1. Definición de los datos tal como se ha descrito en la PEC

```
targets <- read.csv("targets.csv", sep=",", row.names = 1)
counts <- read.csv("counts.csv", sep=";", row.names = 1)
targets.groups <- split(targets, targets$Group)
NIT <- targets.groups$NIT
SFI <- targets.groups$SFI
ELI <- targets.groups$ELI
set.seed(12345)
nit <- NIT[sample(nrow(NIT), 10), ]
sfi <- SFI[sample(nrow(SFI), 10), ]
eli <- ELI[sample(nrow(ELI), 10), ]
targets.total.selected <- rbind(nit,sfi,eli)
rownames(targets.total.selected) <- targets.total.selected$Sample_Name
names(counts) <- gsub(x = names(counts), pattern = "\\.", replacement = "-")
library(dplyr)
library(tibble)
nit.counts <- counts %>% select(matches(paste(nit$Sample_Name)))
sfi.counts <- counts %>% select(matches(paste(sfi$Sample_Name)))
eli.counts <- counts %>% select(matches(paste(eli$Sample_Name)))
counts.total.selected <- cbind(nit.counts,sfi.counts,eli.counts)
ncol(counts.total.selected) == nrow(targets.total.selected)
library("magrittr")
targets.total.selected$Group <- factor(targets.total.selected$Group,
                                       levels=c("NIT","SFI","ELI"),
                                       labels=c("NIT","SFI","ELI"))
library(DESeq2)
ddsMat <- DESeqDataSetFromMatrix(countData = counts.total.selected,
                                 colData = targets.total.selected,
                                 design = ~ Group)
class(ddsMat)
dds <- ddsMat
```

2. Preprocesado de los datos: filtraje y normalización

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

```
#Transformación estabilizadora de la varianza y el rlog
#########################################################
vsd <- vst(dds, blind = FALSE)
head(assay(vsd), 3)
colData(vsd)
rld <- rlog(dds, blind = FALSE)
# head(assay(rld), 3)
library("dplyr")
library("ggplot2")
dds <- estimateSizeFactors(dds)
df <- bind_rows(
  as_data_frame(log2(counts(dds, normalized=TRUE)[, c(1,11)]+1)) %>%
         mutate(transformation = "log2(x + 1)"),
  as_data_frame(assay(vsd)[, c(1,11)]) %>% mutate(transformation = "vst"),
  as_data_frame(assay(rld)[, c(1,11)]) %>% mutate(transformation = "rlog"))
colnames(df)[1:2] <- c("x", "y")
ggplot(df, aes(x = x, y = y)) + geom_hex(bins = 80) +
  coord_fixed() + facet_grid( . ~ transformation)
#########################################################
#Eliminación de efectos Batch ocultos
#########################################################
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)
svseq$sv
for (i in 1:2) {
  stripchart(svseq$sv[, i] ~ dds$Group, vertical = TRUE, main = paste0("SV", i))
  abline(h = 0)
 }
```

3. Identificación de genes diferencialmente expresados

Previsualización de plots

```
#########################################################
#Distancia entre las muestras
#########################################################
sampleDists <- dist(t(assay(vsd)))
library("pheatmap")
library("RColorBrewer")
sampleDistMatrix <- as.matrix( sampleDists )
rownames(sampleDistMatrix) <- paste(vsd$Group)
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette(rev(brewer.pal(9, "Blues")) )(255)
```

```
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)
#########################################################
#PCA plot
#########################################################
plotPCA(vsd, intgroup = c("Group"))
#########################################################
#MDS plot
#########################################################
mds <- as.data.frame(colData(vsd))  %>%
         cbind(cmdscale(sampleDistMatrix))
ggplot(mds, aes(x = `1`, y = `2`, color =Group, shape = Group)) +
  geom_point(size = 3) + coord_fixed()
```

Análisis de expresión diferencial

```
dds <- DESeq(dds, parallel =TRUE)
res1 <- results(dds, contrast=c("Group", "NIT","SFI"))
res2 <- results(dds, contrast=c("Group", "NIT","ELI"))
res3 <- results(dds, contrast=c("Group", "SFI", "ELI"))
mcols(res1, use.names = TRUE)
summary(res1)
mcols(res2, use.names = TRUE)
summary(res2)
mcols(res3, use.names = TRUE)
summary(res3)
sum(res1$padj < 0.1, na.rm=TRUE)
sum(res2$padj < 0.1, na.rm=TRUE)
sum(res3$padj < 0.1, na.rm=TRUE)
#Regulación aguas abajo:
res1Sig <- subset(res1, padj < 0.1)
head(res1Sig[ order(res1Sig$log2FoldChange),c(1:2,6)])
res2Sig <- subset(res2, padj < 0.1)
head(res2Sig[ order(res2Sig$log2FoldChange),c(1:2,6)])
res3Sig <- subset(res3, padj < 0.1)
head(res3Sig[ order(res3Sig$log2FoldChange),c(1:2,6)])
#Regulación aguas arriba:
head(res1Sig[ order(res1Sig$log2FoldChange, decreasing = TRUE),c(1:2,6)])
head(res2Sig[ order(res2Sig$log2FoldChange, decreasing = TRUE),c(1:2,6)])
head(res3Sig[ order(res3Sig$log2FoldChange, decreasing = TRUE),c(1:2,6)])
#########################################################
#Counts plot
#########################################################
topGene <- rownames(res1)[which.min(res1$padj)]
library("ggbeeswarm")
```

```
geneCounts <- plotCounts(dds, gene = topGene, intgroup = c("Group"),
                         returnData = TRUE)
ggplot(geneCounts, aes(x = Group, y = count, color = Group, group = Group)) +
  ggtitle(topGene, "MIN adj p-valor en res1") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_log10() + geom_point(size = 3) + geom_line()
#########################################################
topGene <- rownames(res2)[which.min(res2$padj)]
library("ggbeeswarm")
geneCounts <- plotCounts(dds, gene = topGene, intgroup = c("Group"),
                         returnData = TRUE)
ggplot(geneCounts, aes(x = Group, y = count, color = Group, group = Group)) +
  ggtitle(topGene, "MIN adj p-valor en res2") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_log10() + geom_point(size = 3) + geom_line()
#########################################################
topGene <- rownames(res3)[which.max(res3$padj)]
library("ggbeeswarm")
geneCounts <- plotCounts(dds, gene = topGene, intgroup = c("Group"),
                         returnData = TRUE)
ggplot(geneCounts, aes(x = Group, y = count, color = Group, group = Group)) +
  ggtitle(topGene, "MAX adj p-valor en res3") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_log10() + geom_point(size = 3) + geom_line()
#########################################################
#MA-plot
#########################################################
library("apeglm")
resultsNames(dds)
res.1 <- lfcShrink(dds, coef="Group_SFI_vs_NIT", type="apeglm")
plotMA(res.1, ylim = c(-5,5))
topGene <- rownames(res.1)[which.min(res.1$padj)]
with(res.1[topGene, ], {
  points(baseMean, log2FoldChange, col="dodgerblue", cex=2, lwd=2)
  text(baseMean, log2FoldChange, topGene, pos=2, col="dodgerblue")
})
```

4. Anotación de los resultados

```
library("AnnotationDbi")
library("org.Hs.eg.db")
#columms(org.Hs.eg.db)
#res1:
tmp1=gsub("\\..*","",row.names(res1))
res1$symbol <- mapIds(org.Hs.eg.db,
                      keys=tmp1,
                      column="SYMBOL",
```

```
                      keytype="ENSEMBL",
                      multiVals="first")
res1$entrez <- mapIds(org.Hs.eg.db,
                      keys=tmp1,
                      column="ENTREZID",
                      keytype="ENSEMBL",
                      multiVals="first")
res1Ordered <- res1[order(res1$padj),]
#head(res1Ordered)
head(res1Ordered[,c(1:2,6:8)],6)
#res2:
tmp2=gsub("\\..*","",row.names(res2))
res2$symbol <- mapIds(org.Hs.eg.db,
                      keys=tmp2,
                      column="SYMBOL",
                      keytype="ENSEMBL",
                      multiVals="first")
res2$entrez <- mapIds(org.Hs.eg.db,
                      keys=tmp2,
                      column="ENTREZID",
                      keytype="ENSEMBL",
                      multiVals="first")
res2Ordered <- res2[order(res2$padj),]
head(res2Ordered[,c(1:2,6:8)],6)
# res3:
tmp3=gsub("\\..*","",row.names(res3))
res3$symbol <- mapIds(org.Hs.eg.db,
                      keys=tmp3,
                      column="SYMBOL",
                      keytype="ENSEMBL",
                      multiVals="first")
res3$entrez <- mapIds(org.Hs.eg.db,
                      keys=tmp3,
                      column="ENTREZID",
                      keytype="ENSEMBL",
                      multiVals="first")
res3Ordered <- res3[order(res3$padj),]
head(res3Ordered[,c(1:2,6:8)],6)
```

5. Búsqueda de patrones de expresión y agrupación de las muestras (comparación entre las distintas comparaciones).

```
library("genefilter")
topVarGenes <- head(order(rowVars(assay(vsd)), decreasing = TRUE), 15)
mat  <- assay(vsd)[topVarGenes, ]
mat  <- mat - rowMeans(mat)
anno <- as.data.frame(colData(vsd)[, c("Sample_Name","Group")])
```

```
anno$Sample_Name <- NULL
pheatmap(mat, annotation_col = anno)
```

6. Análisis de significación biológica ("Gene Enrichment Analysis")

```
NITvsSFI = head(res1Ordered,35)
NITvsSFI$padj < 0.05
NITvsELI  = head(res2Ordered,35)
NITvsELI$padj < 0.05
SFIvsELI = head(res3Ordered,35)
SFIvsELI$padj < 0.05
listOfTables <- list(NITvsSFI = NITvsSFI,
                     NITvsELI  = NITvsELI,
                     SFIvsELI = SFIvsELI)
mapped_genes2GO <- mappedkeys(org.Hs.egGO)
mapped_genes2KEGG <- mappedkeys(org.Hs.egPATH)
mapped_genes <- union(mapped_genes2GO , mapped_genes2KEGG)
library(ReactomePA)
listOfData <- listOfTables
comparisonsNames <- names(listOfData)
universe <- mapped_genes
for (i in 1:length(listOfData)){
   genesIn <- listOfData[[i]]
   comparison <- comparisonsNames[i]
   enrich.result <- enrichPathway(gene = genesIn$entrez,
                                  pvalueCutoff = 0.05,
                                  readable = T,
                                  pAdjustMethod = "BH",
                                  organism = "human",
                                  universe = universe)
   cat("####################################")
   cat("\nComparison: ", comparison, "\n")
   cat("####################################\n")
   print(head(enrich.result))
   print(barplot(enrich.result, showCategory = 10, font.size = 7,
             title = paste0("Reactome Pathway ", comparison)))
   print(cnetplot(enrich.result, categorySize = "geneNum", schowCategory = 2,
                  vertex.label.cex = 0.75))
}
```