

Apartado2_DEGtask

Marcos Rubio Fernandez

2023-April

Para comenzar con el análisis de genes diferencialmente expresados de este apartado cargaremos en primer lugar el ambiente conda `ISCIII_GSE` utilizado durante las sesiones prácticas y abriremos el proyecto `Apartado2_DEGtask`. Una vez abierto activaremos los paquetes necesarios para el análisis diferencial.

```
#if (!require("BiocManager", quietly = TRUE))
#   install.packages("BiocManager")

#BiocManager::install("vsn")

library("DESeq2")
library("tidyverse")
library("pheatmap")
library("RColorBrewer")
library("vsn")
```

Pre-procesado de los datos

Partimos de una matriz de cuentas crudas (“`rawcounts.tsv`”) y una tabla con los metadatos del experimento (“`metadata.tsv`”). Para generar nuestro objeto `DESeq2` necesitamos preprocesar estos datos.

Utilizando la función `read.csv` cargamos la matriz de cuentas (`rawcounts.tsv`) en el objeto `counts_data`, estableciendo como nombre de las filas los datos contenidos en la primera columna del archivo (`rownames = 1`). Para la matriz experimental usamos el mismo comando cargando los datos del archivo (`metadata.tsv`) en el objeto `experiment_data`. En este caso, y ya que es condición necesaria, establecemos como nombres de filas el mismo nombre que tienen las columnas del objeto `counts_data`. Además debemos procesar el objeto para eliminar una columna que se crea (con nombre `X`) y establecer como factores las tres variables del experimento: pacientes (`patient`), tratamiento (`agent`) y tiempo de tratamiento (`time`).

Por último, comprobamos que los nombres de las columnas en `counts_data` se encuentran en el objeto `experiment_data` y que además están en el mismo orden.

```
counts_data <- read.csv(file = "./input/rawcounts.tsv", sep = "\t", row.names = 1)
colnames(counts_data)

experiment_data <- read.csv(file = "./input/metadata.tsv", sep = "\t")
rownames(experiment_data) <- colnames(counts_data)
experiment_data <- mutate(.data=experiment_data,
                         X=NULL,
                         patient = as.factor(patient),
```

```

            agent = as.factor(agent),
            time = as.factor(time))

all(colnames(counts_data) %in% rownames(experiment_data))
all(colnames(counts_data) == rownames(experiment_data))

```

Adicionalmente crearemos una nueva variable del experimento, la variable **group** que consistirá en la unión de las variables **agent** y la **time**. Esto lo hacemos para utilizarlo en diseño que veremos en el siguiente apartado.

```

experiment_data$group <- as.factor(paste0(experiment_data$agent, experiment_data$time))
experiment_data$group

## [1] Control24h Control48h DPN24h      DPN48h      OHT24h      OHT48h
## [7] Control24h Control48h DPN24h      DPN48h      OHT24h      OHT48h
## [13] Control24h Control48h DPN24h      DPN48h      OHT24h      OHT48h
## [19] Control48h DPN24h      DPN48h      OHT24h      OHT48h      Control24h
## Levels: Control24h Control48h DPN24h DPN48h OHT24h OHT48h

```

A continuación crearemos el objeto DESeq. Para este objeto, utilizaremos como diseño la modelización por la variable **patienty** la variable **group**. La generación de esta variable nos permite analizar a la vez el tratamiento y el tiempo, en vez de estar utilizando en el diseño la formulación correspondiente la interacción. Se ha decidido realizar esta aproximación siguiendo la recomendación de los autores de DESeq2 en su apartado Interacciones.

‘

```

dds <- DESeqDataSetFromMatrix(countData = counts_data,
                               colData = experiment_data,
                               design = ~ patient + group)

dds

## class: DESeqDataSet
## dim: 53160 24
## metadata(1): version
## assays(1): counts
## rownames(53160): ENSG00000000003 ENSG00000000005 ... ENSG00000257111
##   ENSG00000257112
## rowData names(0):
## colnames(24): GSM913873 GSM913874 ... GSM913895 GSM913896
## colData names(4): patient agent time group

```

Eliminaremos aquellos genes que tengan un número de lecturas menor que 10, ya que este filtrado no supone alteración en el estudio estadístico.

```

keep <- rowSums(counts(dds)) >= 10
dds2 <- dds[keep, ]

```

Comprobamos que el número de genes ha disminuido.

```

## [1] "El número de genes incialmente es: 53160"
## [1] "El número de genes tras filtrar aquellos con un número de lecturas > 10 es: 24416"

```

Análisis exploratorio

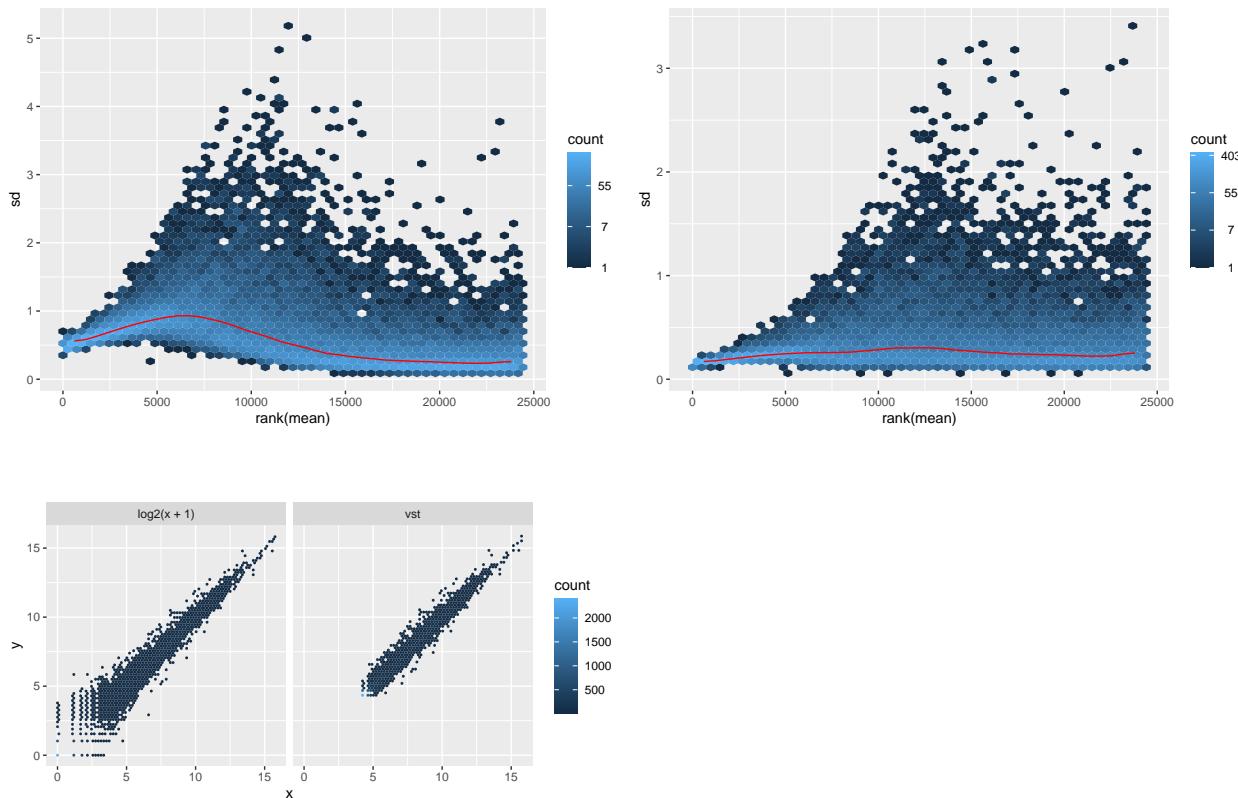
Antes de pasar a realizar la expresión diferencial tenemos que revisar nuestros y datos y la aproximación.

Transformación estabilizadora de la varianza (VST)

Esta función normaliza y estabiliza la varianza de las cuentas (utilizando los size factors), y consiguiendo una matriz que es aproximadamente homocedastica. También tiene en cuenta el tamaño de la librería.

```
vsd <- vst(dds2, blind = TRUE)
```

En los siguientes gráficos podemos ver que la estabilización ha ido correctamente ya que la curva (línea roja) de las cuentas (panel izquierdo) se ha aplanado al transformarse (panel derecho). La tercera figura es una alternativa de ver el resultado de VST con respecto a los valores más pequeños.

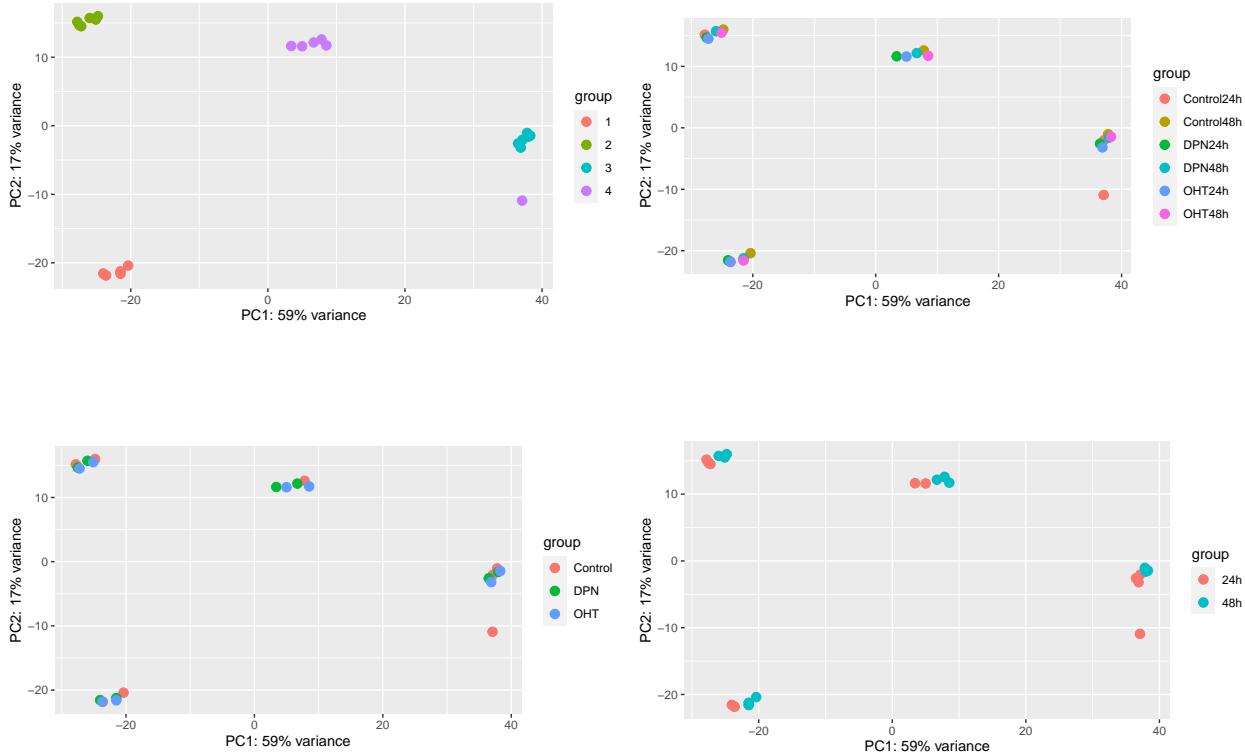


Análisis de componentes principales (PCA)

Este objeto vsd nos permite trabajar con unos datos preparados para el análisis de componentes principales y clustering de distancias. A continuación mostramos los resultados gráficos del análisis de componentes principales (PCA), en donde podemos observar que los datos se agrupan por su similitud en relación a los pacientes. El tratamiento y el tiempo no separan los grupos. En los gráficos se aprecia la existencia de un **outlier** correspondiente al *paciente 4*, tratamiento *control* y tiempo *24 horas*.

```
plotPCA(vsd, intgroup = "patient")
plotPCA(vsd, intgroup = "group")
```

```
plotPCA(vsd, intgroup = "agent")
plotPCA(vsd, intgroup = "time")
```

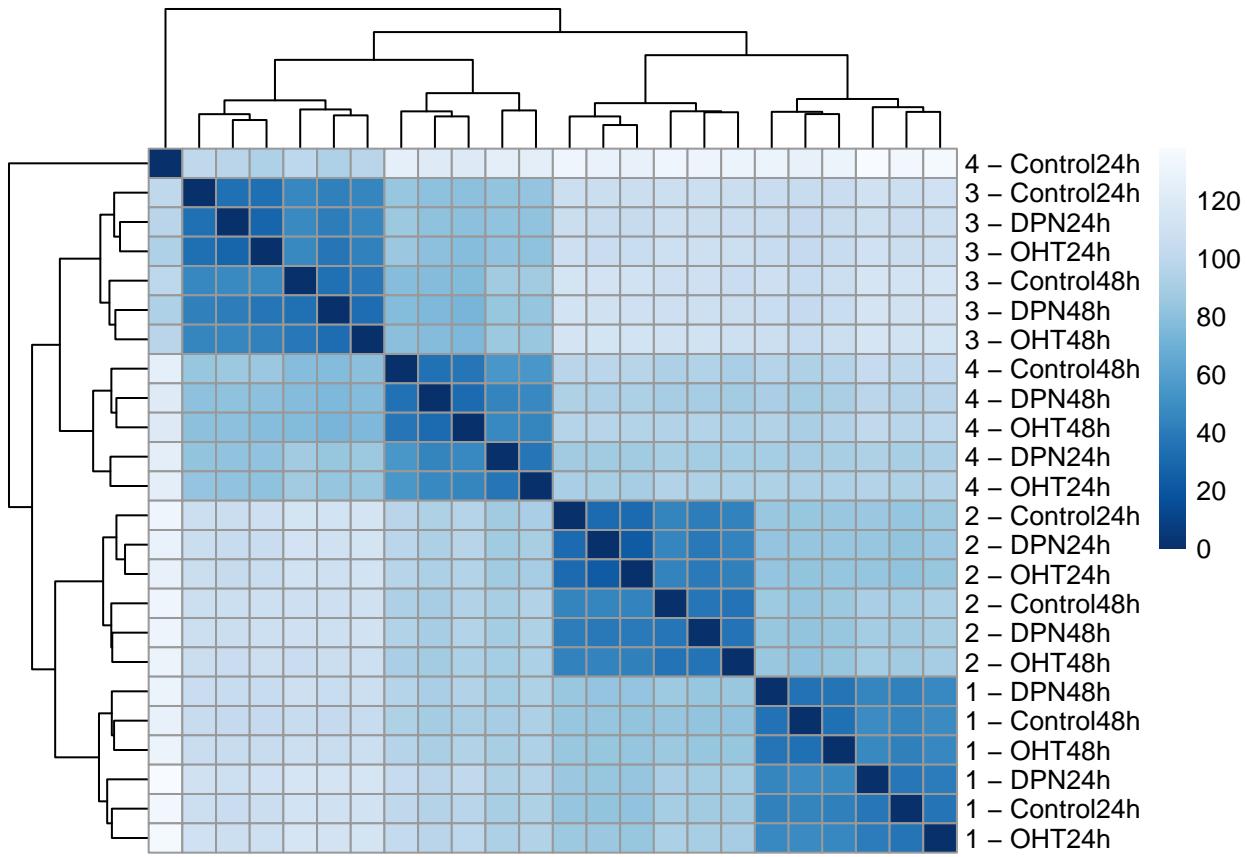


Matriz de distancias

Alternativamente, podemos comprobar los resultados del PCA mediante una matriz de distancias. En ella podemos ver que los pacientes se agrupan bien entre ellos. En cada paciente los tiempos se agrupan mejor entre ellos (24h vs 48h). Entre los pacientes parece que los pacientes 1 y 2 serían más cercanos entre sí, y por otro lado el 3 y el 4. En relación con el outlier, comprobamos su existencia también por este método y vemos que, al igual que demostraba el PCA, se encuentra más cercano a los datos del paciente 3 que a los de su propio grupo.

```
sampleDists <- dist(t(assay(vsd)))
sampleDistMatrix <- as.matrix( sampleDists )
rownames(sampleDistMatrix) <- paste( vsd$patient, vsd$group, sep = " - " )
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)

pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)
```



Análisis de expresión diferencial

Manejo de outlier

Inicialmente vamos a mantener el outlier detectado en el análisis exploratorio. No suele ser una buena práctica mantener este tipo de datos ya que pueden perturbar el resultado en cuanto a los cálculos de expresión diferencial pero creemos, que en una primera aproximación, podría ser más correcto no perder ningún dato ya que este outlier corresponde al único punto que tenemos para esta medición (Paciente 4 - Control - 24 horas). Posteriormente, si se estima necesario repetiremos el análisis eliminando este punto para comprobar si y como varían los resultados.

Análisis de expresión diferencial

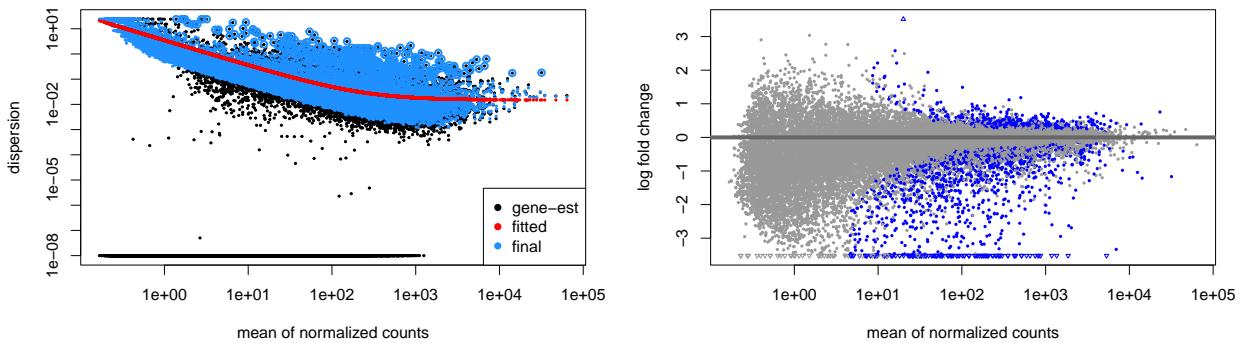
A continuación creamos un nuevo objeto con la función `DESeq`, la cual lleva a cabo el análisis a través de:

- Estimación de los *size factors*
- Estimación de la dispersión
- Ajuste del modelo de GLM - Binomial negativa, y en este caso el estadístico de Wald.

```
dds3 <- DESeq(dds2, test = "Wald")
```

Ahora podemos graficar los resultados obtenidos de la dispersión de los datos (gráfico izquierda) mediante el método de la máxima probabilidad. Y también podemos graficar los resultados de la expresión diferencial (MA-plot, grafico derecha), el cual nos muestra la variación en log fold de cada gen.

```
plotDispEsts(dds3)
plotMA(dds3)
```



Efecto del tratamiento con DPN a 24 horas

Para comprobar si el efecto del tratamiento con DPN a 24 horas tiene efecto en la expresión de algunos genes utilizaremos la función `results` del paquete de DESeq2. En esta función establecemos los diferentes parámetros:

- `object`: identifica el objeto DESeq donde está el modelo ajustado
- `contrast`: las variables que queremos contrastar. En nuestro caso queremos que dentro de la variable `group` compare los datos del `Control24h` con los de `DPN24h`.
- `alpha`: es el valor del nivel de confianza. Lo establecemos en 0,05 % (es decir el 95%).
- `pAdjustMethod`: este análisis nos ofrece el valor de los p-valor para cada gen pero como estamos haciendo múltiples test necesitamos ajustar el p-valor. En este caso aplicamos la corrección de *Benjamini-Hochberg* (BH).

```
res_DPN <- results(object = dds3,
                     contrast = c("group", "Control24h", "DPN24h"),
                     alpha = 0.05,
                     pAdjustMethod = "BH"
)
summary(res_DPN)
res_DPN$padj
```

De estos resultados podemos ver que el número de genes diferencialmente expresados dado el valor p-valor ajustado < 0.05 es de 87 (74-up, 13-down).

Podríamos buscar ahora aquellos con un `logFoldChange` (LFC) superior a 1 (incremento de 2 veces) o de 0.585 (incremento de 1,5 veces) que suelen ser los valores más representativos de un cambio en la expresión. Lo que sucede es que este tipo de análisis *post-hoc* no son una buena práctica. Los autores de DESeq2 describen en su artículo esta situación de la siguiente manera:

For well-powered experiments, however, a statistical test against the conventional null hypothesis of zero LFC may report genes with statistically significant changes that are so weak in effect strength that they could be considered irrelevant or distracting. A common procedure is to disregard genes whose estimated LFC is below some threshold (0). However, this approach loses the benefit of an easily interpretable FDR, as the reported P value and adjusted P value still correspond to the test of zero LFC. It is therefore desirable to include the threshold in the statistical testing procedure directly, i.e., not to filter post hoc on a reported fold-change estimate, but rather to evaluate statistically directly whether there is sufficient evidence that the LFC is above the chosen threshold.

Por lo tanto lo correcto sería incluir el valor de LFC que queremos testar en el análisis de `results`, utilizando la opción `lfcThreshold = 1` y dejando el siguiente código:

```

` 

res_DPN_lfc1 <- results(object = dds3,
                         contrast = c("group", "Control24h", "DPN24h"),
                         alpha = 0.05,
                         lfcThreshold = 1,
                         pAdjustMethod = "BH"
)
summary(res_DPN_lfc1)

## 
## out of 24416 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 1.00 (up) : 0, 0%
## LFC < -1.00 (down) : 0, 0%
## outliers [1] : 0, 0%
## low counts [2] : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results

```

Por lo tanto y teniendo en cuenta estos resultados no obtendríamos ningún gen diferencialmente expresado manteniendo un umbral superior a $LFC > 1$.

Por otra parte, los mismos autores también refieren los siguiente en el artículo:

For small-scale experiments, statistical significance is often a much stricter requirement than biological significance, thereby relieving the researcher from the need to decide on a threshold for biological significance.

Por lo que consideraremos nuestro experimento como de pequeña escala (bajo número de muestras y bajo número de réplicas por cada una) y continuaremos el análisis con los datos `sin` el filtro de `lfcThreshold`. Una forma que tenemos para representar los resultados son los heatmap, en este caso representaremos los 30 genes con menor p-valor ajustado.

```

mat_DPN <- assay(vsd)[head(order(res_DPN$padj), 30), ]
mat_DPN <- mat_DPN - rowMeans(mat_DPN)
anno <- as.data.frame(colData(vsd)[ , c("patient", "agent", "time")])
ann_colors <- list(
  patient = c("1" = "red", "2" = "yellow", "3" = "blue", "4" = "green"),
  agent = c(Control = "gold1", DPN = "turquoise2", OHT = "purple"),
  time = c("24h" = "red2", "48h" = "aquamarine")
)
pheatmap(mat = mat_DPN, annotation_col = anno, show_colnames = F, annotation_colors = ann_colors)

```

De estos resultados podemos decir:

- Hay un grupo de genes diferencialmente expresado en el paciente 4 con respecto al resto de pacientes.
- Existe un segundo grupo de genes que está diferencialmente expresado entre los pacientes 1 y 2 con respecto al paciente 4.
- Por su parte el **outlier** presenta una expresión génica siempre upregulado.

Efecto del tratamiento con OHT a 24 horas

A continuación repetiremos el contraste pero esta vez utilizando el segundo tratamiento a 24 horas.

```
res_OHT <- results(object = dds3,
                      contrast = c("group", "Control24h", "OHT24h"),
                      alpha = 0.05,
                      pAdjustMethod = "BH"
)
summary(res_OHT)

##
## out of 24416 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)      : 44, 0.18%
## LFC < 0 (down)    : 4, 0.016%
## outliers [1]       : 0, 0%
## low counts [2]     : 5681, 23%
## (mean count < 2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

En este caso podemos ver que el número de genes (48) es menor que para DPN, siendo 44 up y 4 down para el LFC. A continuación evaluaremos el mismo contraste pero estableciendo el umbral para LFC a priori.

```
res_OHT_lfc1 <- results(object = dds3,
                           contrast = c("group", "Control24h", "OHT24h"),
                           alpha = 0.05,
                           lfcThreshold = 1,
                           pAdjustMethod = "BH"
)
summary(res_OHT_lfc1)

##
## out of 24416 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 1.00 (up)    : 1, 0.0041%
## LFC < -1.00 (down) : 0, 0%
## outliers [1]        : 0, 0%
## low counts [2]      : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
gene <- res_OHT_lfc1@rownames[which(res_OHT_lfc1$padj < 0.05)]
paste("El gen que sale significativo es:", gene)

## [1] "El gen que sale significativo es: ENSG00000240864"
```

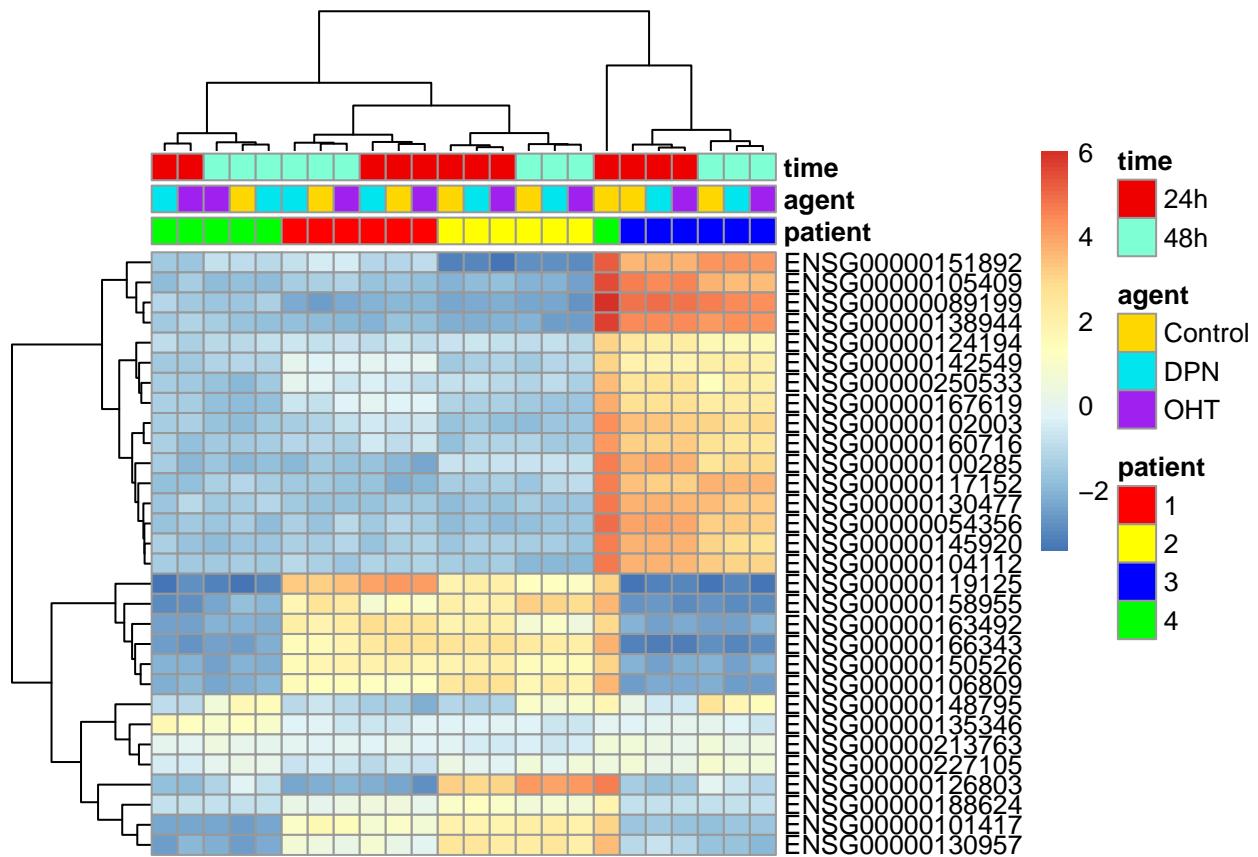
Ya que solo hemos obtenido un gen (ENSG00000240864), vamos a actuar de la misma forma que con DPN y utilizaremos para el heatmap los datos del contraste **sin** filtrar por LFC.

```
mat_OHT <- assay(vsd)[head(order(res_OHT$padj), 30), ]
mat_OHT <- mat_OHT - rowMeans(mat_OHT)
anno <- as.data.frame(colData(vsd)[ , c("patient", "agent", "time")])
ann_colors <- list(
  patient = c("1" = "red", "2" = "yellow", "3" = "blue", "4" = "green"),
  agent = c(Control = "gold1", DPN = "turquoise2", OHT = "purple"),
```

```

    time = c("24h" = "red2", "48h" ="aquamarine")
)
pheatmap(mat = mat_OHT, annotation_col = anno, show_colnames = F, annotation_colors = ann_colors)

```



Outlier

Dada la importancia que parece tener este dato, con valores de expresión (diferencial) de los más altos hemos decidido tratar los datos para eliminarlo. Para este tratamiento de datos podríamos optar por dos aproximaciones:

1. Eliminar todos los datos relativos al paciente 4: quizás esta aproximación sería la más conservadora y correcta viendo el diseño experimental donde no tenemos más que una réplica por condición. Por lo que haciendo esto no dejariamos “descolgadas” las comparaciones de los tratamientos.
2. Eliminar únicamente el dato *outlier*, correspondiente con el paciente 4, tratamiento control y tiempo 24 horas.

Para este ejercicio hemos decidido realizar esta última, aunque como comentamos puede que no sea lo idóneo pero así podemos evaluar (o eso creo) la influencia de este dato sobre el resto del análisis mientras que si eliminamos el grupo entero no podríamos evaluarlo de manera tan concreta.

Lo primero que hacemos es localizar nuestro dato dentro de los objetos `experiment_data` y `counts_data`, y crear nuevos objetos sin él.

```

patient_outlier <- which(experiment_data$patient == 4)
agent_outlier <- which(experiment_data$agent == "Control")
time_outlier <- which(experiment_data$time == "24h")
pat_agent <- intersect(patient_outlier, agent_outlier)

```

```

pat_time <- intersect(patient_outlier,time_outlier)
index_outlier <- intersect(pat_agent,pat_time)

experiment_data_2 <- experiment_data[-index_outlier,]
counts_data_2 <- counts_data[ , -index_outlier]

```

Creamos nuestro DESeqDataSet y filtramos las cuentas con valores menores de 10.

```

dds_outlier <- DESeqDataSetFromMatrix(countData = counts_data_2,
                                         colData = experiment_data_2,
                                         design = ~ patient + group)

dds_outlier

keep <- rowSums(counts(dds_outlier)) >= 10
dds2_outlier <- dds_outlier[keep, ]

## class: DESeqDataSet
## dim: 53160 23
## metadata(1): version
## assays(1): counts
## rownames(53160): ENSG00000000003 ENSG00000000005 ... ENSG00000257111
##   ENSG00000257112
## rowData names(0):
## colnames(23): GSM913873 GSM913874 ... GSM913894 GSM913895
## colData names(4): patient agent time group

```

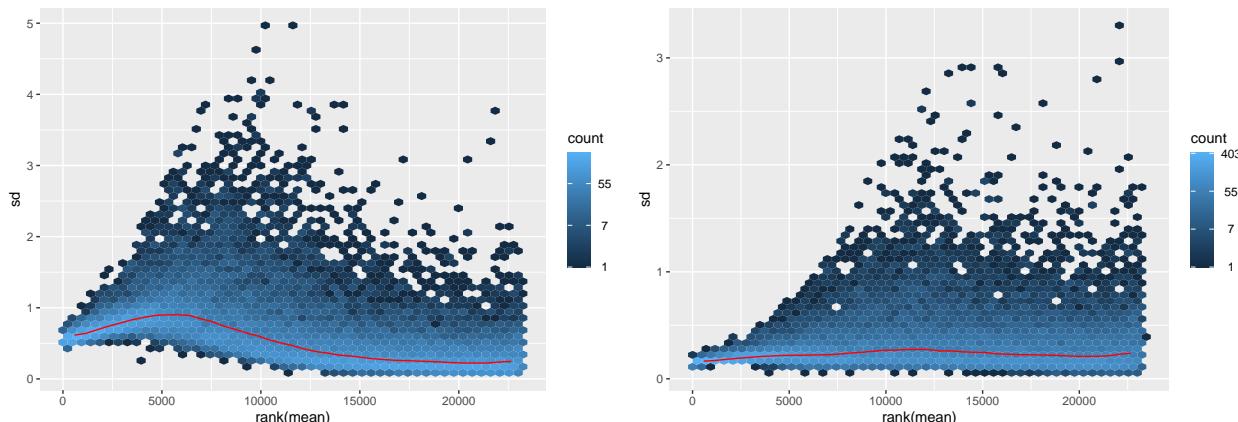
Como en el anterior análisis vemos la reducción (casi a la mitad) en el número de genes.

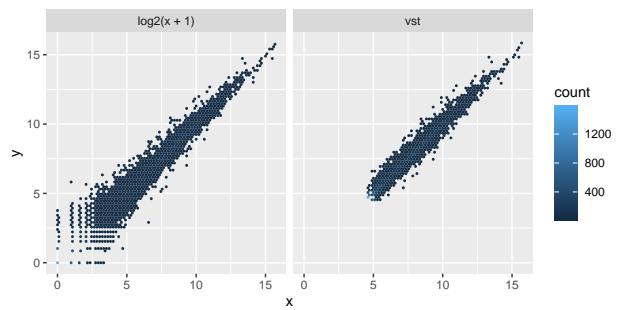
```

## [1] "El número de genes incialmente es: 53160"
## [1] "El número de genes tras filtrar aquellos con un número de lecturas > 10 es: 23233"

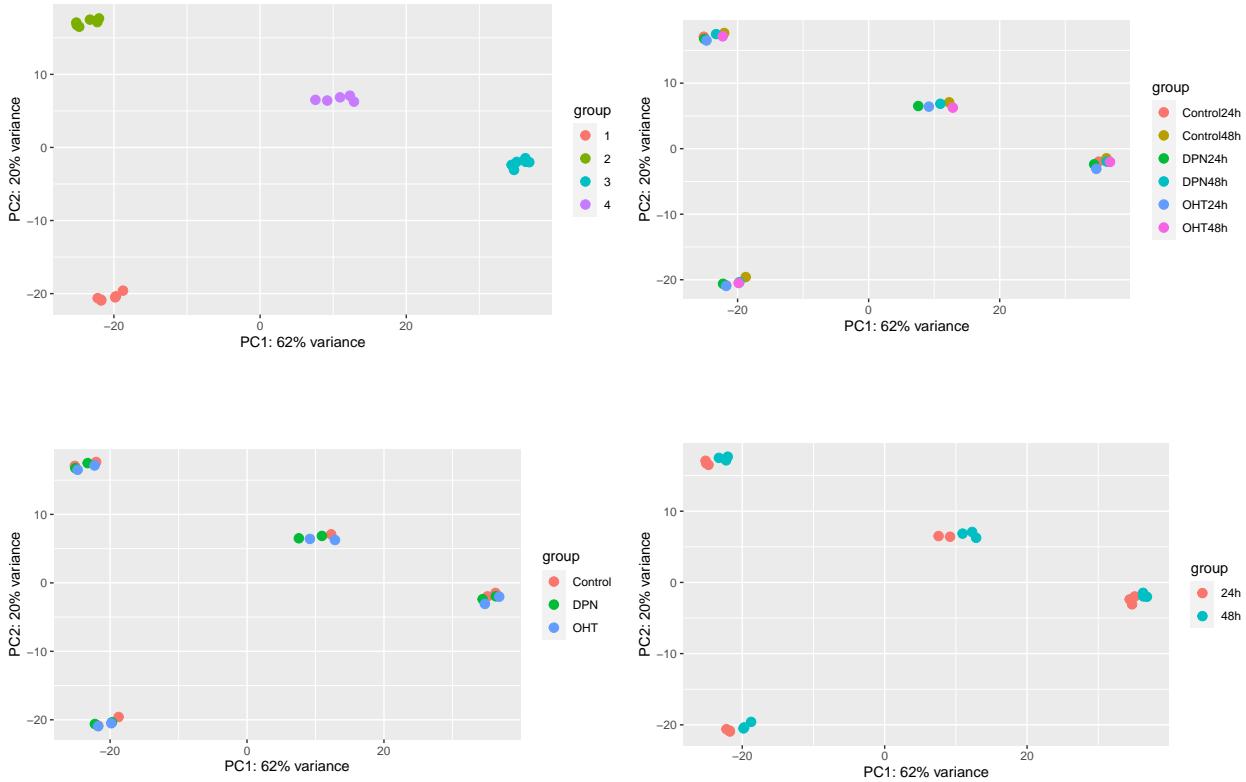
```

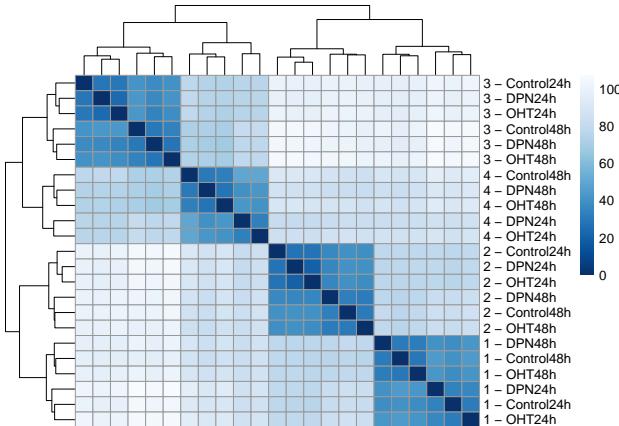
Procedemos con la **transformación estabilizadora de la varianza (VST)**, y mediante los gráficos construidos comprobamos que ha sucedido correctamente.





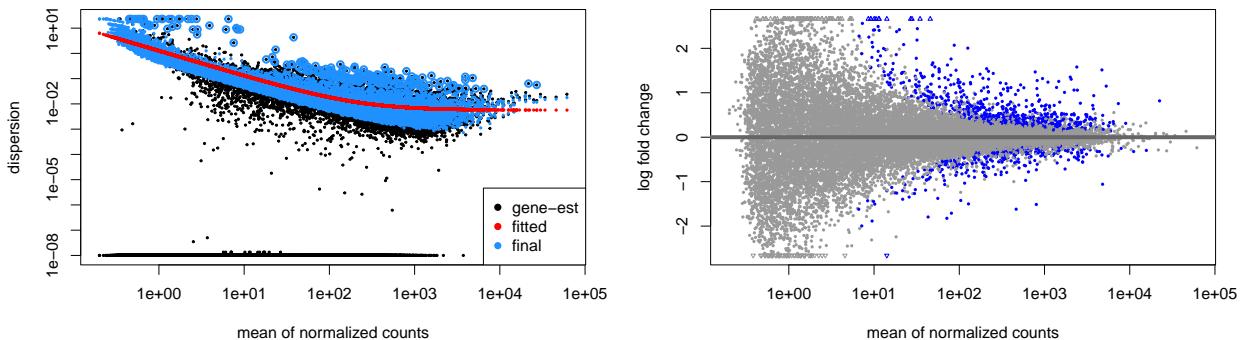
Una vez realizado esto comprobamos de nuevo el **análisis de componentes principales (PCA)** y la **matriz de distancias**, en los cuales vemos que la separación se sigue manteniendo por pacientes pero ya no existe el outlier.





El siguiente paso que haremos es crear nuestro objeto DESeq y comprobar graficamente la distribución de la estimación de la dispersión así como los resultados de la expresión diferencial en el **MA-plot**. De este último podemos resaltar que es diferente con respecto al primer MA-plot obtenido (ver resultados anteriores). En el anterior análisis, muchos de los genes con pvalores significativos se localizan en rangos de logFoldChange negativos, además de ser mucho más abundantes, en este nuevo análisis la mayoría de esos genes con LFC negativo han desaparecido. Esto puede ser indicativo de la influencia que tenia este outlier dentro del análisis de expresión diferencial.

```
dds3_outlier <- DESeq(dds2_outlier, test = "Wald")
```

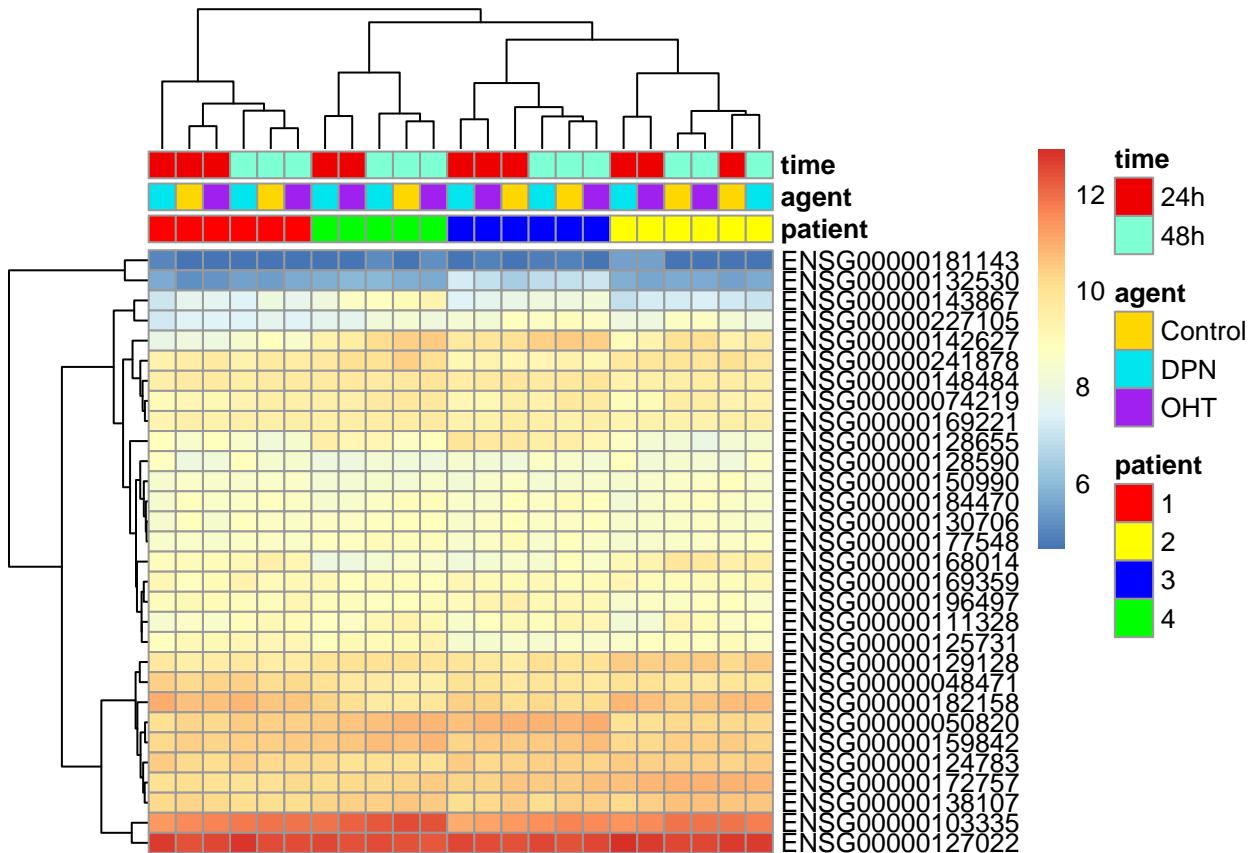


```
## [1] "Summary Control-DPN24h sin lfcThreshold"
##
## out of 23233 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)      : 3, 0.013%
## LFC < 0 (down)    : 2, 0.0086%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
## [1] "Summary Control-DPN24h, lfcThreshold = 1"
##
## out of 23233 with nonzero total read count
```

```

## adjusted p-value < 0.05
## LFC > 1.00 (up)      : 0, 0%
## LFC < -1.00 (down)   : 0, 0%
## outliers [1]          : 0, 0%
## low counts [2]         : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results

```



```

## [1] "Heatmap"
res_OHT_outlier <- results(object = dds3_outlier,
                           contrast = c("group", "Control24h", "OHT24h"),
                           alpha = 0.05,
                           pAdjustMethod = "BH")
summary(res_OHT_outlier)

##
## out of 23233 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0 (up)      : 0, 0%
## LFC < 0 (down)    : 0, 0%
## outliers [1]        : 0, 0%
## low counts [2]      : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results

```

```

## [2] see 'independentFiltering' argument of ?results
res_OHT_lfc1_outlier <- results(object = dds3_outlier,
                                contrast = c("group", "Control24h", "OHT24h"),
                                alpha = 0.05,
                                lfcThreshold = 1,
                                pAdjustMethod = "BH"
)
summary(res_DPN_lfc1_outlier)

##
## out of 23233 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 1.00 (up)      : 0, 0%
## LFC < -1.00 (down)   : 0, 0%
## outliers [1]          : 0, 0%
## low counts [2]         : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
mat_OHT_outlier <- assay(vsd_outlier)[head(order(res_OHT_outlier$padj), 30), ]
mat_OHT <- mat_OHT_outlier - rowMeans(mat_OHT_outlier)
anno <- as.data.frame(colData(vsd_outlier)[ , c("patient", "agent", "time")])
ann_colors <- list(
  patient = c("1" = "red", "2" = "yellow", "3" = "blue", "4" = "green"),
  agent = c(Control = "gold1", DPN = "turquoise2", OHT = "purple"),
  time = c("24h" = "red2", "48h" = "aquamarine")
)
pheatmap(mat = mat_OHT_outlier, annotation_col = anno, show_colnames = F, annotation_colors = ann_color

```

