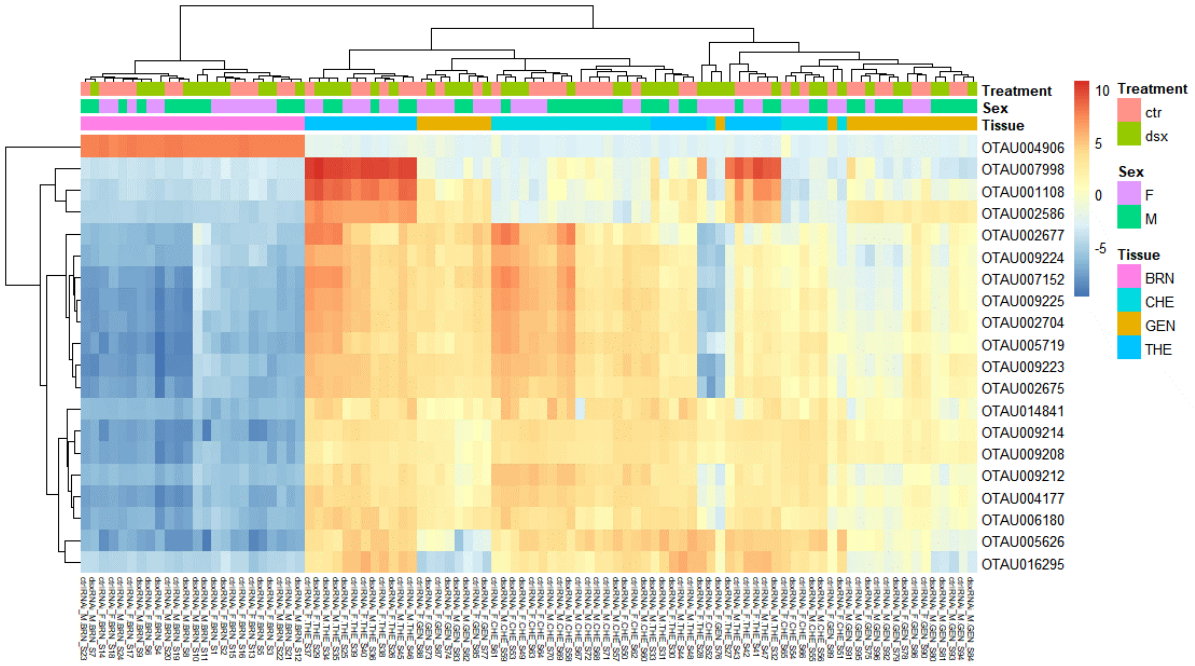
BIOINFORMÁTICA II

Análisis exploratorio de datos de RNAseq: expresión diferencial de genes



Eduardo E. Zattara

La siguiente guía es una adaptación de la viñeta:

Michael I. Love, Simon Anders, Vladislav Kim and Wolfgang Huber. 2019.

RNA-seq workflow: gene-level exploratory analysis and differential expression.

http://bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html

Versión: 1.2

Última Actualización: 25/6/2020

Contenidos

1 – Resumen

Aquí recorremos un flujo de trabajo (workflow) de expresión diferencial de RNA-seq de nivel de gen, partiendo de una tabla de conteos (para aprender cómo llegar a la tabla de conteos, ver el artículo original) utilizando paquetes de Bioconductor. Realizaremos un análisis exploratorio de datos (EDA) para evaluar la calidad y explorar la relación entre las muestras, realizar análisis de expresión genética diferencial y explorar visualmente los resultados.

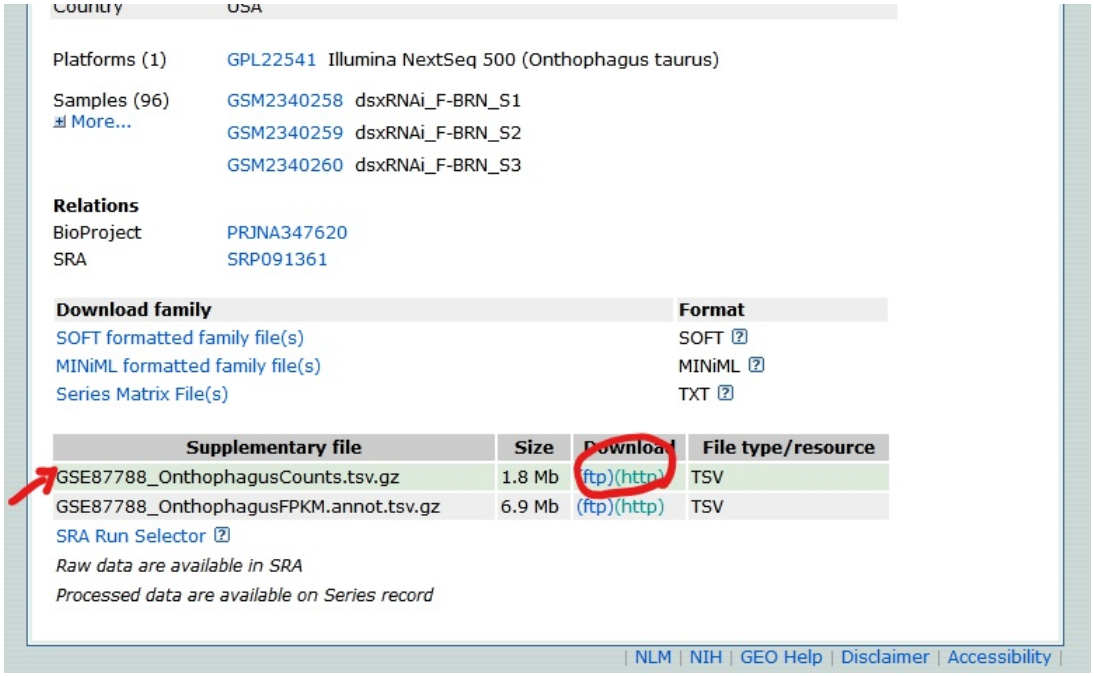
2 – Preparación de entrada de cuantificación para DESeq2

Como input, los métodos estadísticos basados en conteos, como *DESeq2* (Love, Huber y Anders 2014) , *edgeR* (Robinson, McCarthy y Smyth 2009) , *limma* con el método voom (Law et al. 2014) , *DSS* (Wu , Wang y Wu 2013) , *EBSeq* (Leng et al. 2013) y *baySeq* (Hardcastle y Kelly 2010) , esperan datos de entrada obtenidos, por ejemplo, de RNA-seq u otro experimento de secuenciación de alto rendimiento, en forma de **matriz de conteos no normalizados (counts table)**. El valor en la *i* -ésima fila y la *j*-ésima columna de la matriz indica cuántas lecturas (o fragmentos, para paired-end RNA-seq) pueden asignarse al gen *i* en la muestra *j*. Análogamente, para otros tipos de ensayos, las filas de la matriz pueden corresponder, por ejemplo, a regiones de unión (con ChIP-Seq) o secuencias de péptidos (con espectrometría de masas cuantitativa).

Los valores en la matriz deben ser recuentos o recuentos estimados de lecturas / fragmentos de secuenciación. Esto es importante para el modelo estadístico de *DESeq2*, ya que solo los recuentos permiten evaluar la precisión de la medición correctamente. Es importante *nunca* proporcionar recuentos que se hayan normalizado previamente para la profundidad de secuencia / tamaño de la biblioteca, ya que el modelo estadístico es más poderoso cuando se aplica a recuentos no normalizados, y está diseñado para tener en cuenta las diferencias de tamaño de la biblioteca internamente.

En este caso, vamos a usar la tabla de conteos generadas para el artículo de Ledón-Rettig et al. (2017) publicado en https://www.nature.com/articles/ncomms14593 . Los datos de este trabajo están depositados en el Gene Expression Omnibus bajo el identificador GSE87788. En particular, queremos la tabla de conteos que está como información suplementaria en la parte superior de la entrada.

- Descargar la tabla de conteos.
- Descomprimir y guardar en una nueva carpeta (a partir de ahora, carpeta de trabajo) como "`GSE87788_OnthophagusCounts.tsv`".
- Descargar el archivo "`dsxSamples.tsv`" con la información de las muestras y guardar en la carpeta de trabajo.
- Examinar el contenido de ambos archivos con un editor de texto (sin modificar nada). Los archivos también pueden importarse con Excel como archivos delimitados por tabulación.



Country USA

Platforms (1) GPL22541 Illumina NextSeq 500 (Onthophagus taurus)

Samples (96) GSM2340258 dsxRNAi_F-BRN_S1
More... GSM2340259 dsxRNAi_F-BRN_S2
GSM2340260 dsxRNAi_F-BRN_S3

Relations

BioProject PRJNA347620
SRA SRP091361

Download family	Format
SOFT formatted family file(s)	SOFT
MINiML formatted family file(s)	MINiML
Series Matrix File(s)	TXT

Supplementary file	Size	Download	File type/resource
GSE87788_OnthophagusCounts.tsv.gz	1.8 Mb	(ftp)(http)	TSV
GSE87788_OnthophagusFPKM.annot.tsv.gz	6.9 Mb	(ftp)(http)	TSV

SRA Run Selector

Raw data are available in SRA

Processed data are available on Series record

| NLM | NIH | GEO Help | Disclaimer | Accessibility |

3 – Instalar R, RStudio y todos los paquetes necesarios

3.1 – Instalando R

Todos los análisis que se presentan a continuación requieren el uso del entorno R. Es muy probable que ya tenga R instalado en su computadora, pero si no fuera así, o si es necesario actualizar la versión (recomendable).

Para instalar o actualizar R, ir a https://cran.r-project.org/ y seleccionar el paquete `base` para su sistema operativo.

3.2 – Instalando RStudio

RStudio es un entorno que facilita las tareas de scripting y ejecución de R. Si bien no es necesario para completar esta guía, es altamente recomendado.

Para instalar o actualizar R, ir a https://rstudio.com/products/rstudio/download/ y seleccionar RStudio Desktop, Open Source License.

3.3 – Instalando los paquetes requeridos

Si bien la instalación base de R trae numerosas capacidades, una de las características que hace más popular a este entorno es la facilidad para escribir y compartir funciones nuevas. Estas funciones se incorporan dentro de bibliotecas (*libraries*) que a su vez se agrupan en paquetes (*packages*) para facilitar su distribución e instalación. R tiene su propio administrador de paquetes, que puede emplearse directamente usando la función `install.packages` o mediante el comando de RStudio (`Tools/Install packages...`). R usa por defecto el sistema de repositorios CRAN (*Comprehensive R Archive Network*), pero es posible configurar repositorios adicionales o alternativos.

Para instalar los paquetes disponibles en CRAN, abrimos R o RStudio, y usamos el siguiente comando:

```
install.packages("dplyr",
                 "ggplot2",
                 "pheatmap",
                 "RColorBrewer",
                 "PoiClaClu",
                 "glmpca",
                 "ggbeeswarm")
```

Durante el proceso de instalación, primero se descargan los paquetes del repositorio seleccionado (y todas sus dependencias), y luego se instalan. En algunos casos, los paquetes también se compilan durante la instalación. Durante el proceso de instalación, es normal que la consola devuelva cantidades importantes de información y advertencias (*Warnings*) sin que ello implique errores. Para verificar si los paquetes se han instalado correctamente y las bibliotecas están disponibles, vamos a continuación a invocar a estas bibliotecas.

```
library("dplyr")
library("ggplot2")
library("pheatmap")
library("RColorBrewer")
library("PoiClaClu")
library("glmpca") # glmpca requiere R 3.6+
library("ggbeeswarm")
```

Además, vamos a usar paquetes instalados mediante Bioconductor, un proyecto cuya misión es "promover el análisis estadístico y la comprensión de los ensayos biológicos de alto rendimiento actuales y emergentes (http://www.bioconductor.org)". Bioconductor proporciona herramientas para el análisis y la comprensión de datos genómicos de alto rendimiento. Bioconductor utiliza el lenguaje de programación estadística R, y es de código abierto y desarrollo abierto. Tiene dos lanzamientos cada año y una comunidad de usuarios activa.

Para instalar la versión más reciente de Bioconductor (3.11) y usar el sistema de instalación `BiocManager` **se requiere la versión 4.0+ de R**. Si tenemos esa versión de R, entonces se emplean los siguientes comandos:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.11")
BiocManager::install(c("DESeq2", "vsn","apeglm","genefilter"))
```

Si contamos con una versión de R anterior, se deben usar los siguientes comandos:

```
source("https://bioconductor.org/biocLite.R")
BiocInstaller::biocLite(c("DESeq2", "vsn","apeglm","genefilter"))
```

Tras instalar los paquetes, invocamos las bibliotecas correspondientes:

```
library("DESeq2")
library("vsn")
library("apeglm")
```

3.4 – El objeto *DESeqDataSet*, información de las muestra y la fórmula de diseño

Los paquetes de software de Bioconductor a menudo definen y usan una clase personalizada para almacenar datos que garantiza que todos los espacios de datos necesarios se proporcionen de manera consistente y cumplan con los requisitos. Además, Bioconductor tiene clases de datos generales (como *SummarizedExperiment*, por "*Experimento resumido*") que se pueden usar para mover datos entre paquetes. Además, las clases principales de Bioconductor brindan una funcionalidad útil: por ejemplo, un subset o reordenamiento de las filas o columnas de un *SummarizedExperiment* subsetea o reordena automáticamente los *RowRanges* y *colData* asociados, lo que puede ayudar a prevenir intercambios accidentales de muestras que de lo contrario conducirían a resultados espurios. Con *SummarizedExperiment* todo esto se soluciona detrás de escena.

En *DESeq2*, la clase personalizada se llama *DESeqDataSet*. Está construida en base a la clase *SummarizedExperiment*, y es fácil convertir estos objetos en objetos *DESeqDataSet*, como mostramos a continuación. Una de las dos diferencias principales es que se accede a `assay` usando la función de acceso `counts`, y que la clase *DESeqDataSet exige* que los valores en esta matriz sean enteros no negativos.

Una segunda diferencia es que *DESeqDataSet* tiene una *fórmula de diseño* asociada. El diseño experimental se especifica al comienzo del análisis, ya que informará a muchas de las funciones *DESeq2* cómo tratar las muestras en el análisis (una excepción es la estimación del factor de tamaño, es decir, el ajuste para diferentes tamaños de biblioteca, que no depende de la fórmula del diseño). La fórmula de diseño indica qué columnas de la tabla de información de las muestras (`colData`) especifican el diseño experimental y cómo deben usarse estos factores en el análisis.

Vamos por lo tanto a generar el objeto *DESeqDataSet* y lo vamos a llamar `dds_dsx_all`. Para ello necesitamos

Abrir RStudio y crear un nuevo script de R. Es conveniente guardarlo en la misma carpeta donde guardamos los datos. Luego ir a `Session/Set Working Directory.../To source file location` y seleccionar dicha carpeta. Guardar en esa misma carpeta el archivo `dsxSamples.tsv` que contiene la tabla de información sobre las muestras del experimento.

```
#Leer tabla de conteos
dsxCts <- read.delim("GSE87788_OnthophagusCounts.tsv", row.names = "gene")

#Leer la tabla de información sobre las muestras, y seleccionar variables de interé
  s
dsxSamples <- read.delim("dsxSamples.tsv", sep="\t", row.names=1)
dsxColdata <- dsxSamples[,c("Tissue","Treatment","Sex","individual")]

#Verificar que todas las muestras de la tabla estén presentes con el mismo nombre e
  n la matriz
all(rownames(dsxColdata) %in% colnames(dsxCts)) # Debe devolver "TRUE"
```

```
all(rownames(dsxColdata) == colnames(dsxCts)) # Debe devolver "TRUE"

#Creamos el objeto DESeqDataSet
dds_dsx_all <- DESeqDataSetFromMatrix(countData = dsxCts,
                                      colData = dsxColdata,
                                      design = ~ Tissue + Sex + Treatment)
```

En la última línea, se especifica la *fórmula de diseño*. En este caso, la fórmula especifica que existen tres factores: tejido (CHE, THE, GEN, BRN), sexo (M, F) y tratamiento (ctr, dsx). El orden de los factores es importante, ya que el contraste por default es entre niveles del último factor especificado; o sea, en este caso, el análisis por default va a comparar expresión génica entre individuos tratados con dsx^{RNAi} y controles. Recuerden que los individuos dsx^{RNAi} representan los perfiles de expresión genómica cuando la expresión de *dsx* es reprimida.

Veamos los contenidos de nuestro objeto

```
dds_dsx_all
##
class: DESeqDataSet
dim: 17483 96
metadata(1): version
assays(1): counts
rownames(17483): OTAU000001 OTAU000002 ... OTAU017482 OTAU017483
rowData names(0):
colnames(96): ctrlRNAi_M.BRN_S19 ctrlRNAi_M.BRN_S20 ... dsxRNAi_F.GEN_S77
  dsxRNAi_F.GEN_S78
colData names(4): Tissue Treatment Sex individual
```

El atributo `dim` nos muestra las dimensiones de la matriz de datos: 17483 genes x 96 muestras.

Examinemos las columnas de `colData`. Podemos ver cada una de las columnas simplemente usando `$` directamente en *SummarizedExperiment* o *DESeqDataSet*.

```
colData(dds_dsx_all)
##
DataFrame with 96 rows and 4 columns
                     Tissue Treatment      Sex individual
                   <factor>  <factor> <factor>   <factor>
ctrlRNAi_M.BRN_S19      BRN       ctr        M        LM7
ctrlRNAi_M.BRN_S20      BRN       ctr        M        LM8
ctrlRNAi_M.BRN_S21      BRN       ctr        M        LM9
ctrlRNAi_M.BRN_S22      BRN       ctr        M       LM10
ctrlRNAi_M.BRN_S23      BRN       ctr        M       LM11
...                     ...       ...      ...        ...
dsxRNAi_F.GEN_S74       GEN       dsx        F        LF2
dsxRNAi_F.GEN_S75       GEN       dsx        F        LF3
dsxRNAi_F.GEN_S76       GEN       dsx        F        LF4
dsxRNAi_F.GEN_S77       GEN       dsx        F        LF5
dsxRNAi_F.GEN_S78       GEN       dsx        F        LF6
```

4 – Análisis exploratorio y visualización

Hay dos caminos separados en este flujo de trabajo; el que veremos primero involucra *transformaciones de los conteos* para explorar visualmente las relaciones de muestra. En la segunda parte, volveremos a los recuentos originales sin procesar para *las pruebas estadísticas*. Esto es crítico porque los métodos de prueba estadística se basan en datos de conteo originales (no escalados o transformados) para calcular la precisión de las mediciones.

4.1– Prefiltrando el conjunto de datos

Nuestra matriz de conteo con nuestro *DESeqDataSet* contiene muchas filas con solo ceros, y adicionalmente muchas filas con solo unos pocos fragmentos en total. Para reducir el tamaño del objeto y aumentar la velocidad de nuestras funciones, podemos eliminar las filas que no tienen o casi no tienen información sobre la cantidad de expresión génica. Aquí aplicamos la regla de filtrado más mínima: eliminar filas del *DESeqDataSet* que no tienen recuentos, o solo un recuento único en todas las muestras. La ponderación / filtrado adicional para mejorar la potencia se aplica en un paso posterior en el flujo de trabajo.

```
nrow(dds_dsx_all)
##
[1] 17483
keep <- rowSums(counts(dds_dsx_all)) > 1
dds_dsx_work <- dds_dsx_all[keep,]
nrow(dds_dsx_work)
##
[1] 14554
```

Para algunos conjuntos de datos, puede tener sentido realizar un filtrado adicional. Por ejemplo, se puede especificar que al menos 3 muestras tengan un conteo de 10 o más. Una recomendación para el número de muestras se establecería en el tamaño de grupo más pequeño. Dicha regla podría especificarse creando un vector lógico y subseteando el `dds` anterior. Aquí hay un ejemplo de otra regla que podríamos haber usado (aquí no se usa para el filtrado):

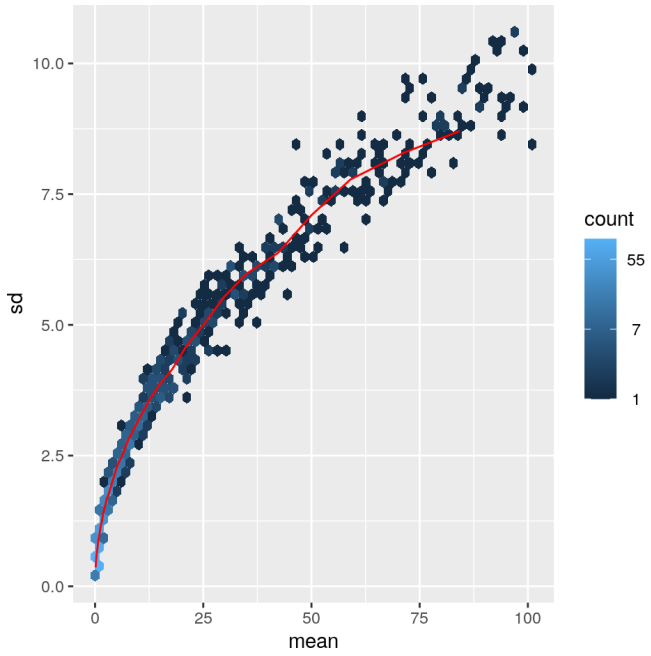
```
# al menos 3 muestras con un conteo de 10 o más
keep <- rowSums(counts(dds_dsx_all) >= 10) >= 3
```

4.2 – La transformación estabilizadora de la varianza y el rlog

Muchos métodos estadísticos comunes para el análisis exploratorio de datos multidimensionales, por ejemplo, el agrupamiento y el *análisis de componentes principales* (PCA), funcionan mejor para datos que generalmente tengan el mismo rango de varianza en diferentes rangos de los valores medios. Cuando la cantidad esperada de varianza es aproximadamente la misma en diferentes valores medios, se dice que los datos son *homoscedásticos*. Sin embargo, para los recuentos de RNA-seq, la varianza esperada crece con la media. Por ejemplo, si uno realiza un PCA directamente en una matriz

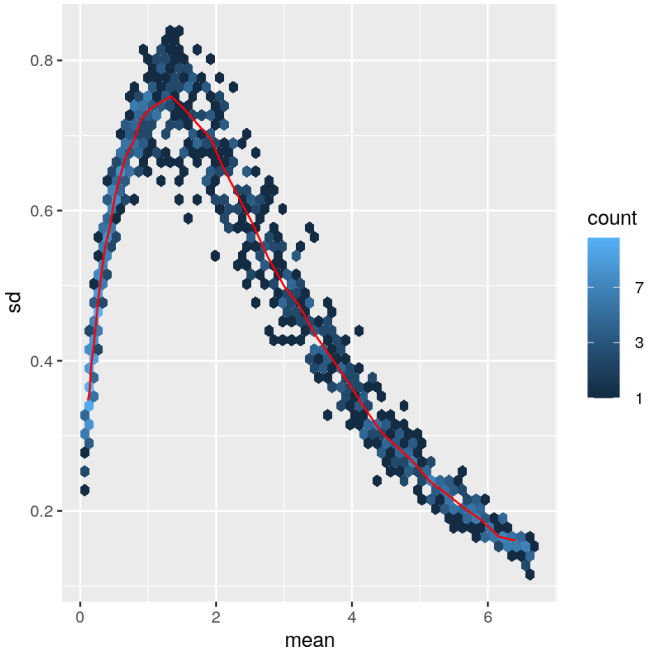
de recuentos o recuentos normalizados (por ejemplo, corrigiendo las diferencias en la profundidad de secuenciación), la gráfica resultante generalmente depende principalmente de los genes con *mayor* conteo porque muestran las mayores diferencias absolutas entre muestras. Una estrategia simple y de uso frecuente para evitar esto es tomar el logaritmo de los valores de recuento normalizados más un pseudoconteo de 1 (para evitar que el análisis de un error al intentar calcular el logaritmo de 0); sin embargo, dependiendo de la elección del pseudocuenteo, ahora los genes con los recuentos *más bajos* aportarán una gran cantidad de ruido a la trama resultante, porque tomar el logaritmo de los recuentos pequeños realmente infla su varianza. Podemos mostrar rápidamente esta propiedad de conteos con algunos datos simulados (aquí, Poisson cuenta con un rango de lambda de 0.1 a 100). Trazamos la desviación estándar de cada fila (genes) contra la media:

```
lambda <- 10^seq(from = -1, to = 2, length = 1000)
cts <- matrix(rpois(1000*100, lambda), ncol = 100)
library("vsn")
meanSdPlot(cts, ranks = FALSE)
```



Y para los recuentos transformados por logaritmo:

```
log.cts.one <- log2(cts + 1)
meanSdPlot(log.cts.one, ranks = FALSE)
```



El logaritmo con un pseudo-conteo pequeño amplifica las diferencias cuando los valores están cerca de 0. Los genes de recuento bajo con una baja relación señal / ruido contribuirán excesivamente a las distancias muestra-muestra y a los gráficos de PCA.

Como solución, *DESeq2* ofrece dos transformaciones para los datos de conteo que estabilizan la varianza a través de la media: la *transformación de estabilización de varianza* (VST) para datos binomiales negativos con una tendencia de dispersión media (Anders y Huber 2010), implementada en la función `vst`, y la *transformación de logaritmo regularizado* o `rlog` (Love, Huber y Anders 2014).

Para genes con recuentos altos, tanto el VST como el rlog darán un resultado similar a la transformación ordinaria log2 de los recuentos normalizados. Sin embargo, para los genes con recuentos más bajos, los valores se reducen a un valor medio. El VST o los datos transformados por `rlog` se vuelven aproximadamente homoscedásticos (tendencia más plana en el *meanSdPlot*), y se pueden usar directamente para calcular distancias entre muestras, hacer gráficos de PCA o como entrada para métodos posteriores que funcionan mejor con datos homoscedásticos.

¿Qué transformación elegir? El VST es mucho más rápido de calcular y es menos sensible a los valores atípicos de conteo alto que el rlog. El rlog tiende a funcionar bien en pequeños conjuntos de datos (n<30), posiblemente superando al VST cuando hay un amplio rango de profundidad de secuenciación entre muestras (un orden de diferencia de magnitud). Por lo tanto, recomendamos el VST para conjuntos de datos de medianos a grandes (n> 30). Puede realizar ambas transformaciones y comparar las `meanSdPlot` gráficas PCA o generadas, como se describe a continuación.

Tenga en cuenta que las dos transformaciones ofrecidas por DESeq2 se proporcionan para aplicaciones que *no* sean tests estadísticos de expresión diferenciales. Para estos tests, recomendamos la función *DESeq* aplicada a los recuentos sin procesar, como se describe más adelante en este flujo de trabajo, que también tiene en cuenta la dependencia de la varianza de los recuentos en el valor medio durante el paso de estimación de dispersión.

Tanto *vst* como *rlog* devuelven un objeto *DESeqTransform* que se basa en la clase *SummarizedExperiment*. Los valores transformados ya no son conteos y se almacenan en el espacio *assay*. El *colData* que se adjuntó a `dds` todavía es accesible:

```
vsd <- vst(dds_dsx_work, blind = FALSE)
head(assay(vsd)[,1:4], 3)
##
           ctrlRNAi_M.BRN_S19 ctrlRNAi_M.BRN_S20 ctrlRNAi_M.BRN_S21 ctrlRNAi_M.BRN_S22
OTAU000001           7.417811           7.387822           7.603091           7.333607
OTAU000002           9.608927           9.693236           9.645107           9.688119
OTAU000003          10.838079          10.840303          10.932025          10.860767
colData(vsd)
DataFrame with 96 rows and 5 columns
                     Tissue Treatment      Sex individual        sizeFactor
                   <factor>  <factor> <factor>   <factor>         <numeric>
ctrlRNAi_M.BRN_S19      BRN       ctr        M        LM7  1.29894343918514
ctrlRNAi_M.BRN_S20      BRN       ctr        M        LM8  1.24653773007499
ctrlRNAi_M.BRN_S21      BRN       ctr        M        LM9  1.29267006149121
ctrlRNAi_M.BRN_S22      BRN       ctr        M       LM10  1.35260475369406
ctrlRNAi_M.BRN_S23      BRN       ctr        M       LM11  1.25359976551614
...                     ...       ...      ...        ...               ...
dsxRNAi_F.GEN_S74       GEN       dsx        F        LF2  1.40144648413127
dsxRNAi_F.GEN_S75       GEN       dsx        F        LF3  1.03968475467086
dsxRNAi_F.GEN_S76       GEN       dsx        F        LF4  1.40703223735854
dsxRNAi_F.GEN_S77       GEN       dsx        F        LF5 0.984598282068786
dsxRNAi_F.GEN_S78       GEN       dsx        F        LF6  1.18491128222692
```

De nuevo, para el *rlog*:

```
rld <- rlog(dds_dsx_work, blind = FALSE)
head(assay(rld)[,1:4], 3)
##
           ctrlRNAi_M.BRN_S19 ctrlRNAi_M.BRN_S20 ctrlRNAi_M.BRN_S21 ctrlRNAi_M.BRN_S22
OTAU000001           6.984687           6.947977           7.207150           6.881132
OTAU000002           9.534876           9.619193           9.571091           9.614077
OTAU000003          10.784577          10.786732          10.875686          10.806594
```

En las llamadas a funciones anteriores, especificamos `blind = FALSE`, lo que significa que las diferencias entre los tejidos, sexo y tratamientos (las variables en el diseño) no contribuirán a la tendencia media-varianza esperada del experimento. El diseño experimental no se usa directamente en la transformación, solo para estimar la cantidad global de variabilidad en los recuentos. Para una transformación totalmente *no supervisada*, se puede establecer `blind = TRUE` (que es el valor predeterminado).

Para mostrar el efecto de la transformación, en la figura a continuación graficamos la primera muestra contra la segunda, primero simplemente usando la función *log2* (después de agregar 1, para evitar tomar el logaritmo de 0), y luego usando los valores transformados en VST y rlog. Para el enfoque *log2*, primero debemos estimar los *factores de tamaño* para tener en cuenta la profundidad de secuenciación

y luego especificar `normalized=TRUE`. La corrección de la profundidad de secuencia se realiza automáticamente para *vst* y *rlog*.

```
library("dplyr")
library("ggplot2")
dds_dsx_work <- estimateSizeFactors(dds_dsx_work)
df <- bind_rows(
  as_data_frame(log2(counts(dds_dsx_work, normalized=TRUE)[, 1:2]+1)) %>%
         mutate(transformation = "log2(x + 1)"),
  as_data_frame(assay(vsd)[, 1:2]) %>% mutate(transformation = "vst"),
  as_data_frame(assay(rld)[, 1:2]) %>% mutate(transformation = "rlog"))
colnames(df)[1:2] <- c("x", "y")
lvls <- c("log2(x + 1)", "vst", "rlog")
df$transformation <- factor(df$transformation, levels=lvls)
ggplot(df, aes(x = x, y = y)) + geom_hex(bins = 80) +
  coord_fixed() + facet_grid( . ~ transformation)
```

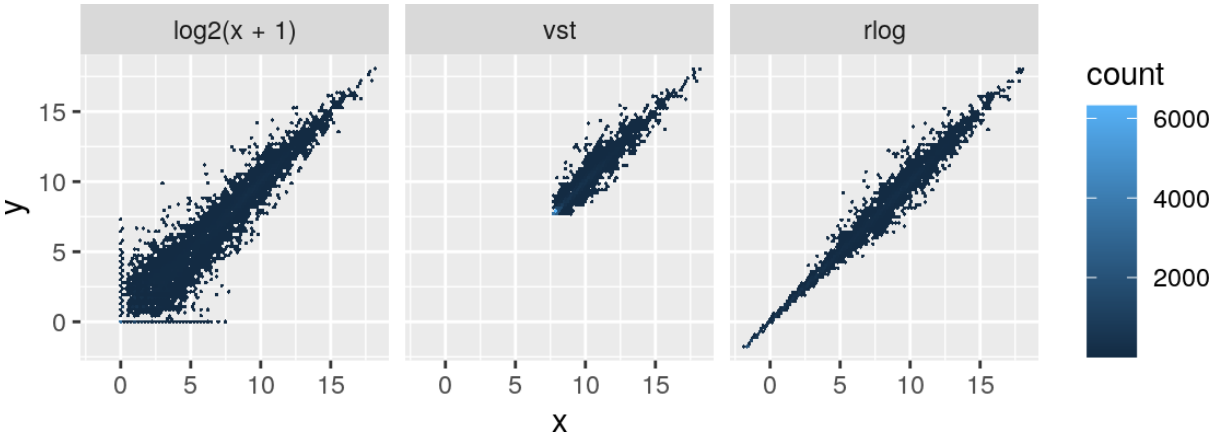


Diagrama de dispersión de conteos transformados de dos muestras. Se muestran diagramas de dispersión utilizando la transformación log2 de conteos normalizados (izquierda), utilizando el VST (centro) y el rlog (derecha). Mientras que el rlog está aproximadamente en la misma escala que el log2(conteo), el VST tiene un desplazamiento hacia arriba para los valores más pequeños. Son las diferencias entre las muestras (desviación de y = x en estos diagramas de dispersión) lo que contribuirá a los cálculos de distancia y al diagrama de PCA.

Podemos ver cómo los genes con conteos bajos (esquina inferior izquierda) parecen ser excesivamente variables en la escala logarítmica ordinaria, mientras que VST y rlog comprimen las diferencias para los genes de conteo bajo para los cuales los datos proporcionan poca información sobre la expresión diferencial.

4.3 – Distancias entre muestras

Un primer paso en un análisis de secuencia de ARN es evaluar la similitud general entre muestras: ¿Qué muestras son similares entre sí? ¿Cuáles son diferentes? ¿Se ajusta esto a las expectativas del diseño del experimento?

Usamos la función *dist* para calcular la distancia euclidiana entre muestras. Para garantizar que tengamos una contribución aproximadamente igual de todos los genes, la usamos en los datos VST. Necesitamos transponer la matriz de valores usando *t*, porque la función *dist* espera que las diferentes muestras sean filas de su argumento, y diferentes dimensiones (aquí, genes) sean columnas.

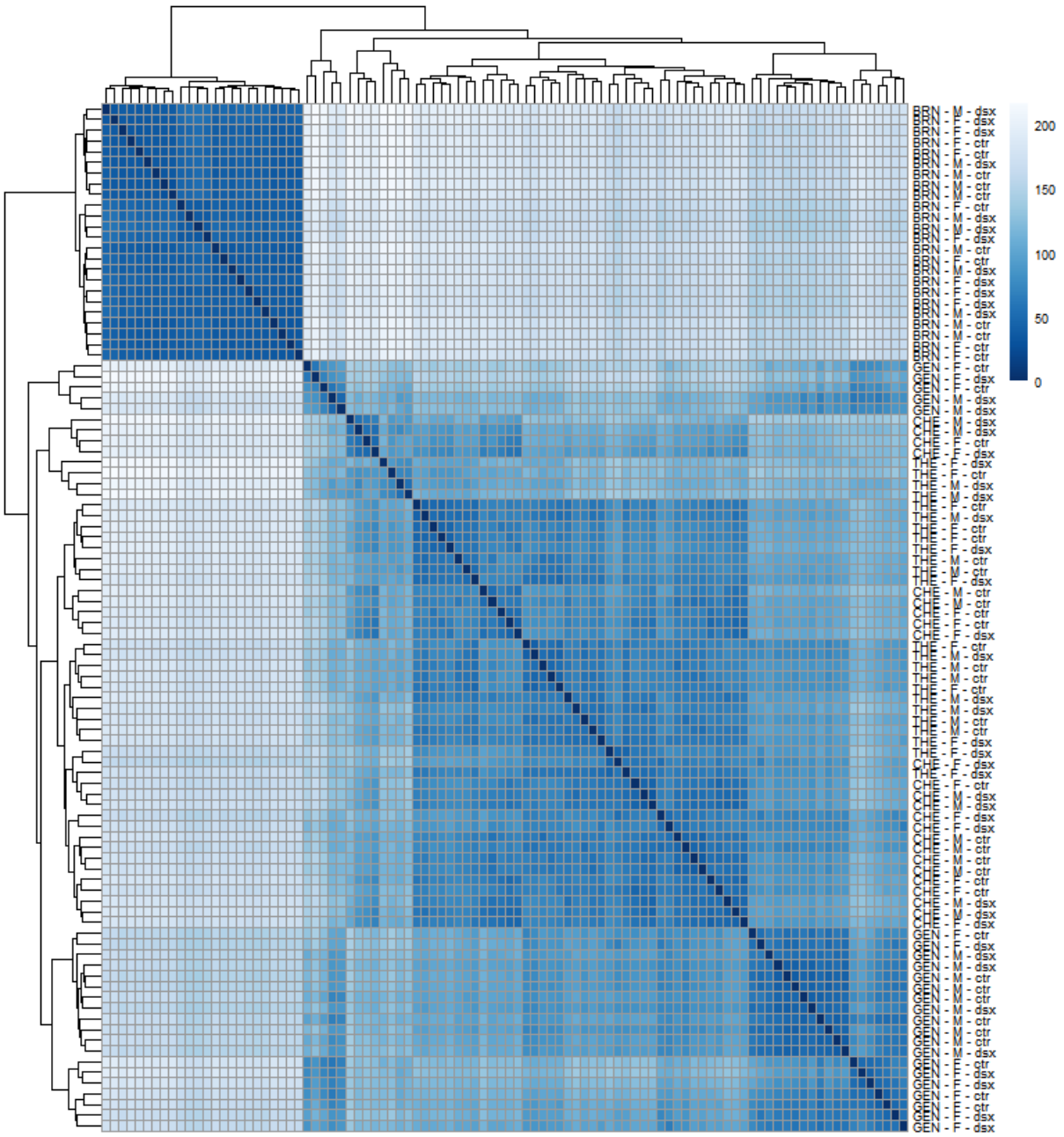
```
sampleDists <- dist(t(assay(vsd)))
```

Visualizamos las distancias en un mapa de calor (heatmap) en una figura a continuación, utilizando la función *pheatmap* del paquete *pheatmap* .

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

Con el fin de trazar la matriz de distancia de muestra con las filas / columnas ordenadas por las distancias en nuestra matriz de distancia, necesitamos ingresar manualmente el objeto `sampleDists` como argumento `clustering_distance` para la función *pheatmap*. De lo contrario, la función *pheatmap* supondría que la matriz contiene los valores de los datos en sí mismos y calcularía las distancias entre las filas / columnas de la matriz de distancia, lo que no se desea. También especificamos manualmente una paleta de color azul usando el *colorRampPalette* función del paquete *RColorBrewer* .

```
sampleDistMatrix <- as.matrix( sampleDists )
rownames(sampleDistMatrix) <- paste( (vsd)$Tissue, (vsd)$Sex,  (vsd)$Treatment, sep
  = " - " )
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors , fontsize = 7)
```



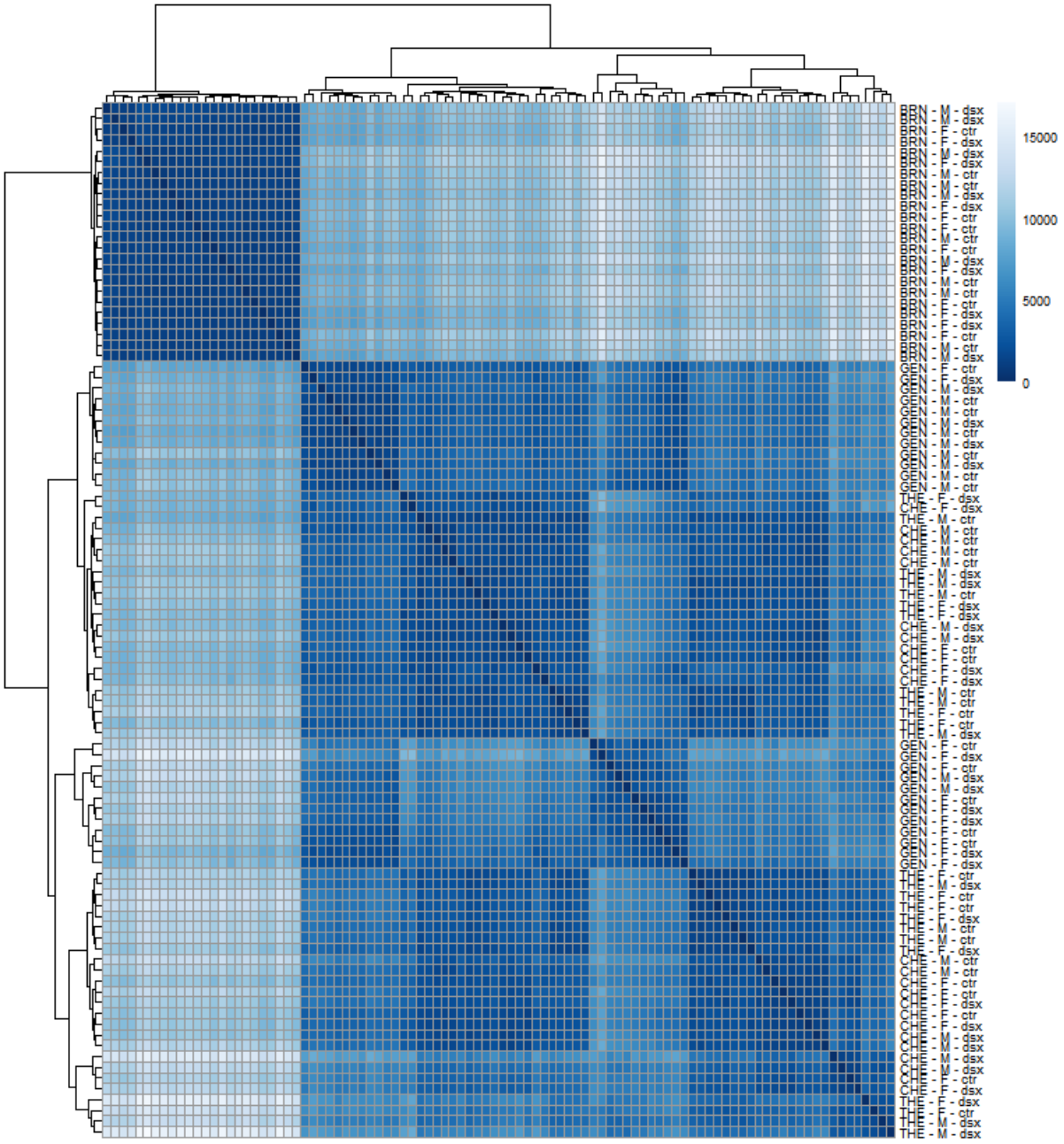
Tenga en cuenta que hemos cambiado los nombres de fila de la matriz de distancia para contener el tejido, sexo y tipo de tratamiento en lugar de la ID de la muestra, de modo que tengamos toda esta información a la vista al mirar el mapa de calor.

Otra opción para calcular distancias de muestra es utilizar la *Distancia de Poisson* (Witten 2011), implementada en el paquete *PoiClaClu* . Esta medida de disimilitud entre conteos también tiene en cuenta la estructura de varianza inherente de los conteos al calcular las distancias entre muestras. La función *PoissonDistance* toma la matriz de conteo original (no normalizada) con muestras como filas en lugar de columnas, por lo que debemos transponer los conteos `dds_dsx_work`.

```
library("PoiClaClu")
poisd <- PoissonDistance(t(counts(dds_dsx_work)))
```

Graficamos el mapa de calor en una figura a continuación.

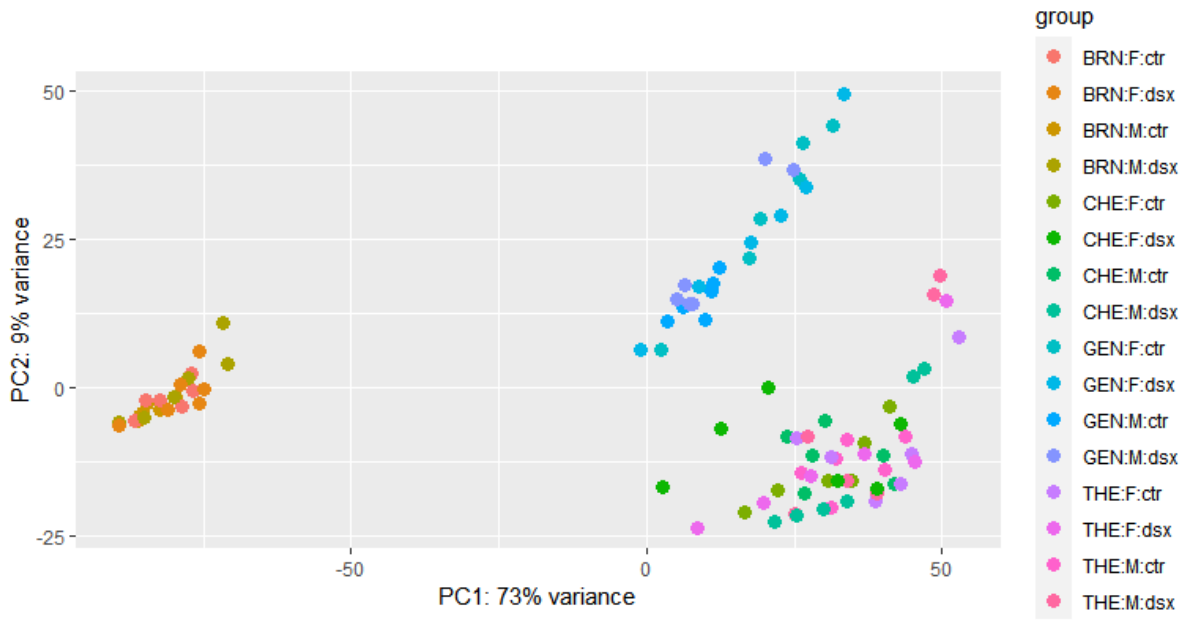
```
samplePoisDistMatrix <- as.matrix( poisd$dd )
rownames(samplePoisDistMatrix) <- paste((vsd)$Tissue, (vsd)$Sex,  (vsd)$Treatment, s
   ep = " - " )
colnames(samplePoisDistMatrix) <- NULL
pheatmap(samplePoisDistMatrix,
         clustering_distance_rows = poisd$dd,
         clustering_distance_cols = poisd$dd,
         col = colors , fontsize = 7)
```



4.4 – Gráficos PCA

Otra forma de visualizar distancias entre muestras es un análisis de componentes principales (PCA). En este método de ordenación, los puntos de datos (aquí, las muestras) se proyectan en el plano 2D de manera que se extienden en las dos direcciones que explican la mayoría de las diferencias (figura a continuación). El eje x es la dirección que separa más los puntos de datos. Los valores de las muestras en esta dirección se escriben como *PC1*. El eje y es una dirección (debe ser *ortogonal* a la primera dirección) que separa más los datos. Los valores de las muestras en esta dirección se escriben *PC2*. El porcentaje de la varianza total contenida en la dirección se imprime en la etiqueta del eje. Tenga en cuenta que estos porcentajes no se suman al 100%, porque hay más dimensiones que contienen la varianza restante (aunque cada una de estas dimensiones restantes explicará menos que las dos que vemos).

```
plotPCA (vsd, intgroup = c("Tissue", "Sex", "Treatment"))
```

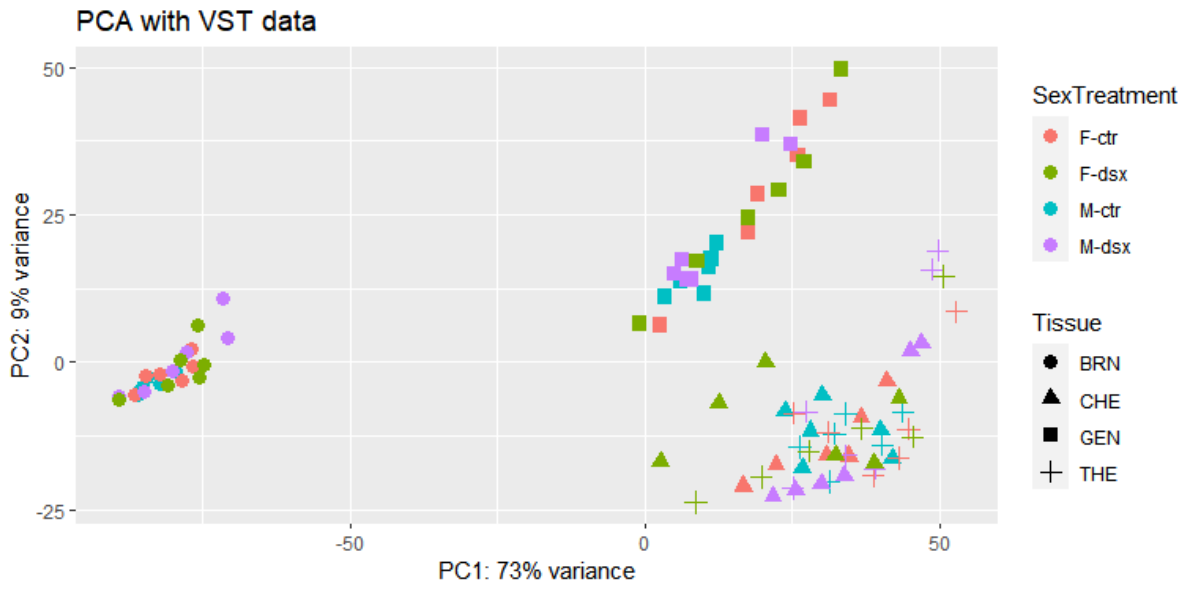


Aquí, hemos utilizado la función *plotPCA* que viene con *DESeq2*. Los tres términos especificados por `intgroup` son los grupos interesantes para etiquetar las muestras; le dicen a la función que los use para elegir colores. También podemos construir el diagrama de PCA desde cero usando el paquete *ggplot2* (Wickham 2009). Esto se hace pidiendo a la función *plotPCA* que devuelva los datos utilizados para el trazado en lugar de construir el trazado. Consulte la documentación de *ggplot2* para obtener más detalles sobre el uso de *ggplot*.

```
pcaData <- plotPCA(vsd, intgroup = c( "dex", "cell"), returnData = TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
SexTreatment <-with(pcaData, paste(Sex,Treatment,sep="-"))
```

Luego podemos usar estos datos para construir un segundo gráfico en una figura a continuación, especificando que el color de los puntos debe reflejar el tratamiento con dexametasona y la forma debe reflejar la línea celular.

```
ggplot(pcaData, aes(x = PC1, y = PC2, color = SexTreatment, shape = Tissue)) +
  geom_point(size =3) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  coord_fixed() +
  ggtitle("PCA with VST data")
```



En el gráfico de PCA, vemos que las diferencias entre los tejidos (símbolos) son considerables, al punto de oscurecer diferencias entre sexos o tratamientos. Esto muestra por qué será importante tener esto en cuenta en los contrastes.

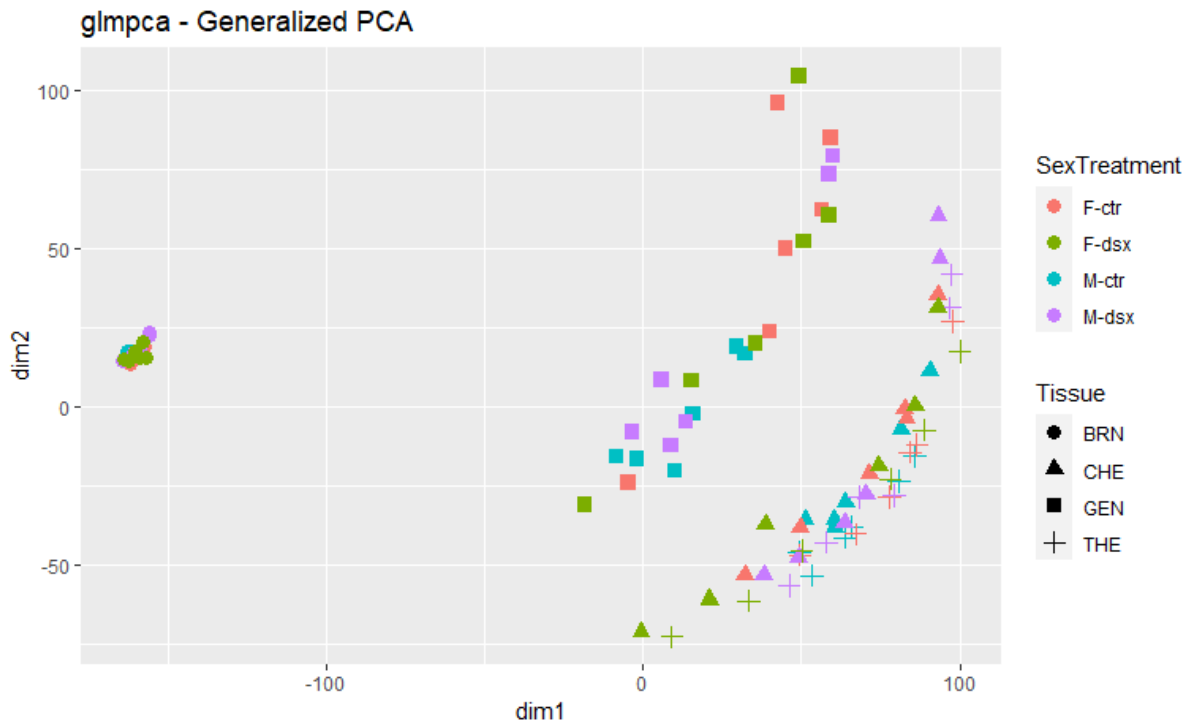
4.5 – Gráficos de PCA utilizando PCA generalizada

Otra técnica para realizar la reducción dimensional en datos que no se distribuyen normalmente (por ejemplo, datos de conteos sobredispersados) es el *análisis generalizado de componentes principales* , o GLM-PCA, (Townes et *al.2019*) como se implementa en el paquete *glmpca de* CRAN . Este paquete toma como entrada la matriz de conteo, así como el número de dimensiones latentes para ajustar (aquí, especificamos 2). Según lo indicado por Townes et al. (2019):

> *… Proponemos el uso de GLM-PCA, una generalización de PCA a las probabilidades familiares exponenciales. GLM-PCA opera en conteos sin procesar, evitando las trampas de la normalización. También demostramos que la aplicación de PCA a la desviación o los residuos de Pearson proporciona una aproximación útil y rápida a GLM-PCA.*

```
library("glmpca")# requiere R versión 3.6+
```

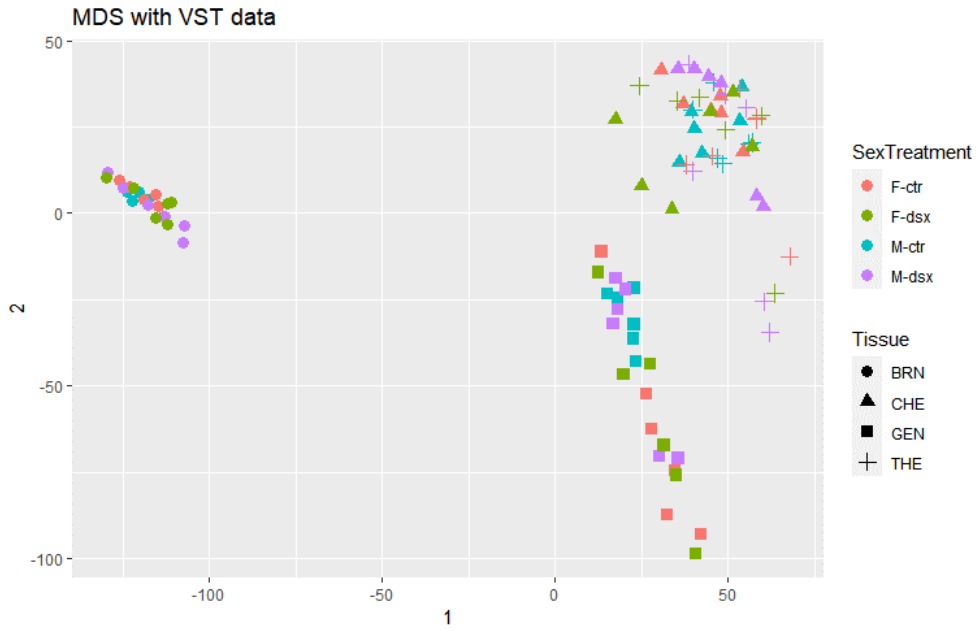
```
gpca <- glmpca(counts(dds_dsx_work), L=2)
gpca.dat <- gpca$factors
gpca.dat$Sex <- dds_dsx_work$Sex
gpca.dat$Tissue <- dds_dsx_work$Tissue
gpca.dat$Treatment <- dds_dsx_work$Treatment
gpca.dat$SexTreatment <- paste(dds_dsx_work$Sex,dds_dsx_work$Treatment,sep="-")
ggplot(gpca.dat, aes(x = dim1, y = dim2, color = SexTreatment, shape = Tissue)) +
  geom_point(size =3) + coord_fixed() + ggtitle("glmpca - Generalized PCA")
```



4.6 - Gráfico MDS

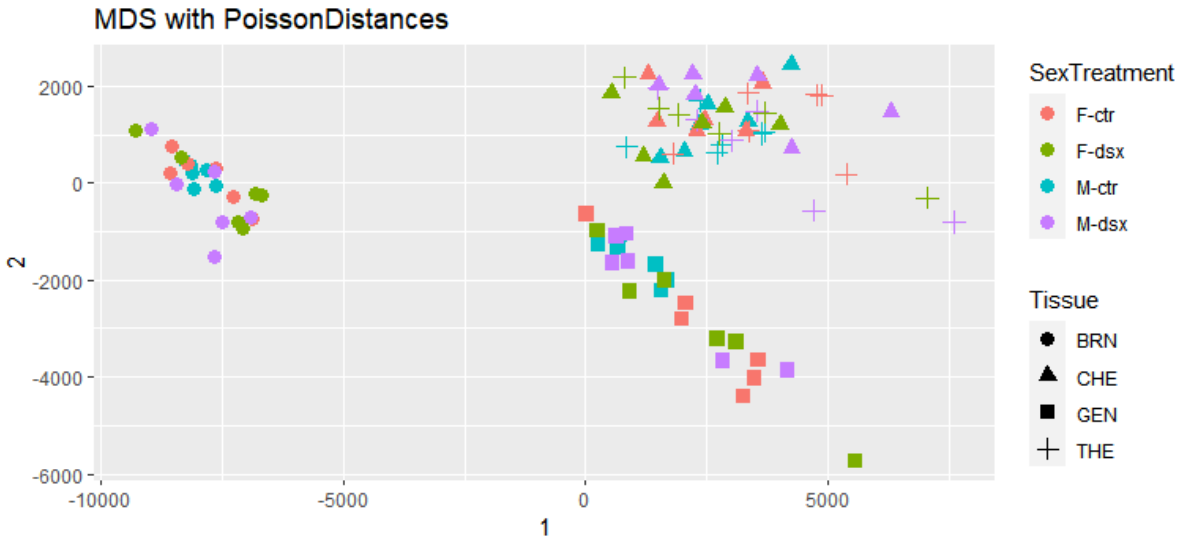
Se puede hacer otra gráfica, muy similar a la gráfica de PCA, usando la función de *escalamiento multidimensional* (MDS) en la base R. Esto es útil cuando no tenemos una matriz de datos, sino solo una matriz de distancias. Aquí calculamos el MDS para las distancias calculadas a partir de los datos *VST* y los graficamos en la siguiente figura.

```
mds <- as.data.frame(colData(vsd))  %>% cbind(cmdscale(sampleDistMatrix))
SexTreatment <-with(mds, paste(Sex,Treatment,sep="-"))
ggplot(mds, aes(x = `1`, y = `2`, color = SexTreatment, shape = Tissue)) +
  geom_point(size = 3) + coord_fixed() + ggtitle("MDS with VST data")
```



En la figura a continuación mostramos la misma gráfica para *PoissonDistance* :

```
mdsPois <- as.data.frame(colData(dds_dsx_work)) %>% cbind(cmdscale(samplePoisDistMa
   trix))
SexTreatment <-with(mdsPois, paste(Sex,Treatment,sep="-"))
ggplot(mdsPois, aes(x = `1`, y = `2`, color = SexTreatment, shape = Tissue)) +
  geom_point(size = 3) + coord_fixed() + ggtitle("MDS with PoissonDistances")
```



5 – Análisis de expresión diferencial

5.1 – Ejecutar el pipeline de expresión diferencial

Como ya hemos especificado un diseño experimental cuando creamos el *DESeqDataSet* , podemos ejecutar el pipeline de expresión diferencial en los conteos sin *formato* con una sola llamada a la función *DESeq* :

```
dds_dsx_work <- DESeq(dds_dsx_work)
```

Esta función imprimirá un mensaje para los diversos pasos que realiza. Estos se describen con más detalle en la página del manual de *DESeq*, a la que se puede acceder escribiendo `?DESeq`. Brevemente, estos son: la estimación de los factores de tamaño (control de las diferencias en la profundidad de secuenciación de las muestras), la estimación de los valores de dispersión para cada gen y el ajuste de un modelo lineal generalizado.

Se *devuelve* un *DESeqDataSet* que contiene todos los parámetros ajustados dentro de él, y la siguiente sección describe cómo extraer tablas de resultados de interés de este objeto.

5.2 – Construyendo la tabla de resultados

Llamar a *resultados* sin ningún argumento extraerá los cambios estimados del log2fold y los valores de *p* para la última variable en la fórmula de diseño. Si hay más de 2 niveles para esta variable, los *resultados* extraerán la tabla de resultados para una comparación del último nivel sobre el primer nivel. La comparación se imprime en la parte superior de la salida: `Treatment dsx vs ctr`.

```
res <- results(dds_dsx_work)
res
##
log2 fold change (MLE): Treatment dsx vs ctr
Wald test p-value: Treatment dsx vs ctr
DataFrame with 16376 rows and 6 columns
                    baseMean      log2FoldChange             lfcSE
                   <numeric>           <numeric>         <numeric>
OTAU000001   115.269191809465 0.00429633210295244 0.0477614460827956
OTAU000002   872.366298391412 -0.0886047526689091 0.0331813494161848
OTAU000003   1476.28466132297 -0.0207623959175764 0.0466573680575139
OTAU000004  0.478913293424159  0.0895468866802047  0.468502034929671
OTAU000005   20.1373870783668   0.104013767175372 0.0799060877312879

                        stat              pvalue              padj
                   <numeric>           <numeric>         <numeric>
OTAU000001  0.0899539786861696   0.928323786187127 0.978646253345714
OTAU000002   -2.67031794149067 0.00757794539754188 0.143915352636434
OTAU000003  -0.444997152260773   0.656321812099588 0.884482016080553
OTAU000004   0.191134466883686   0.848420244725349 0.955442391282774
OTAU000005    1.30170015988211   0.193018904919718 0.588482634750759
```

Podríamos haber producido de manera equivalente esta tabla de resultados con el siguiente comando más específico. Debido a que `Treatment` es la última variable en el diseño, opcionalmente podríamos dejar de lado el argumento `contrast` para extraer la comparación de los dos niveles de `Treatment`.

```
res <- results(dds_dsx_work, contrast=c("Treatment","dsx","ctr"))
```

Como `res` es un objeto *DataFrame*, contiene metadatos con información sobre el significado de las columnas:

```
mcols(res, use.names = TRUE)
##
DataFrame with 6 rows and 2 columns
                     type                                 description
                 <character>                                 <character>
baseMean       intermediate    mean of normalized counts for all samples
log2FoldChange      results log2 fold change (MLE): Treatment dsx vs ctr
lfcSE               results         standard error: Treatment dsx vs ctr
stat                results         Wald statistic: Treatment dsx vs ctr
pvalue              results      Wald test p-value: Treatment dsx vs ctr
padj                results                         BH adjusted p-values
```

La primera columna, `baseMean` es solo el promedio de los valores de conteo normalizados, dividido por los factores de tamaño, tomados sobre todas las muestras en el *DESeqDataSet*. Las cuatro columnas restantes se refieren a un contraste específico, es decir, la comparación del nivel `dsx` sobre el nivel `ctr` para la variable del factor `Treatment`. A continuación, veremos cómo obtener otros contrastes.

La columna `log2FoldChange` es la estimación del tamaño del efecto. Nos dice cuánto parece haber cambiado la expresión del gen debido al tratamiento *dsx* dsRNA en comparación con las muestras control. Este valor se informa en una escala logarítmica en base 2: por ejemplo, un cambio log2fold de 1.5 significa que la expresión del gen se incrementa por un factor multiplicativo de $2^{1,5} \approx 2.82$.

Por supuesto, esta estimación tiene una incertidumbre asociada, que está disponible en la columna `lfcSE`, la estimación de error estándar para la estimación de cambio log2 fold. También podemos expresar la incertidumbre de una estimación de tamaño de efecto particular como resultado de un test estadístico. El **propósito de un test de expresión diferencial es probar si los datos proporcionan evidencia suficiente para concluir que este valor es realmente diferente de cero**. *DESeq2* realiza para cada gen una *prueba de hipótesis* para ver si la evidencia es suficiente para decidir contra la *hipótesis nula* (que postula que el tratamiento tiene un efecto cero en el gen) y que la diferencia observada entre el tratamiento y el control fue simplemente causada por la variabilidad experimental (es decir, el tipo de variabilidad que puede esperar entre diferentes muestras en el mismo grupo de tratamiento). Como es habitual en las estadísticas, el resultado de esta prueba se informa como un valor *p*, y se encuentra en la columna `pvalue`. Recuerde que un valor *p* indica la probabilidad de que un cambio tan fuerte como el observado, o incluso más fuerte, se vea en la situación descrita por la hipótesis nula.

También podemos resumir los resultados con la siguiente línea de código, que informa cierta información adicional, que se tratará en secciones posteriores.

```
summary(res)
##
out of 16376 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 332, 2%
LFC < 0 (down)     : 172, 1.1%
outliers [1]       : 43, 0.26%
low counts [2]     : 3172, 19%
(mean count < 0)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Sólo 332 genes (2%) muestran un aumento significativo en su expresión en individuos *dsx*^{RNAi} relativo a los controles, y 172 genes (1.1%) muestran una reducción significativa para el mismo contraste. Esto es un número bastante bajo, pero hay que tener en cuenta de que este contraste combina datos de tejidos muy distintos, y en ambos sexos. Sin embargo, hay dos formas de ser aún más estrictos acerca de qué conjunto de genes se consideran significativos:

- baje el umbral de la tasa de descubrimiento falso (el umbral para `padj` en la tabla de resultados)
- elevar el umbral para cambio log2 fold por encima de 0 usando el argumento `lfcThreshold` de *results*

Si bajamos el umbral de la tasa de descubrimiento falso, también deberíamos informar a la función `results()`al respecto, para que la función pueda usar este umbral para el filtrado independiente óptimo que realiza:

```
res.05 <- results(dds_dsx_work, alpha = 0.05)
table(res.05$padj < 0.05)
##
## FALSE  TRUE
## 12847   314
```

Si queremos aumentar el umbral de cambio log2 fold, de modo que analicemos los genes que muestran cambios más sustanciales debido al tratamiento, simplemente proporcionamos un valor en la escala log2. Por ejemplo, al especificar `lfcThreshold = 1`, probamos los genes que muestran efectos significativos del tratamiento en los recuentos de genes más del doble o menos de la mitad, porque 2^1=2.

```
resLFC1 <- results(dds_dsx_work, lfcThreshold=1)
table(resLFC1$padj < 0.1)
##
## FALSE  TRUE
## 16327     6
```

Algunas veces un subconjunto de *p-values* será `NA` ("no disponible"). Esta es la forma en que *DESeq* informa que todos los recuentos de este gen fueron cero y, por lo tanto, no se aplicó ninguna prueba. Además, a los valores de *p* se les puede asignar `NA` si el gen se excluyó del análisis porque contenía un recuento extremo atípico. Para obtener más información, consulte la sección de detección de valores atípicos de la viñeta *DESeq2*.

Si utiliza los resultados de un paquete de análisis R en una investigación publicada, puede encontrar la cita adecuada para el software escribiendo `citation("pkgName")`, donde sustituirá el nombre del paquete `pkgName`. Citar documentos sobre métodos ayuda a apoyar y recompensar a las personas que dedican tiempo al software de código abierto para el análisis de datos genómicos.

5.3 – Otras comparaciones

En general, los resultados para una comparación de cualquiera de los dos niveles de una variable se pueden extraer utilizando el argumento `contrast` de *results*. El usuario debe especificar tres valores: el nombre de la variable, el nombre del nivel para el numerador y el nombre del nivel para el denominador. Aquí extraemos resultados para el cambio log2 fold entre cerebro (BRN) y epitelio cefálico dorsal (CHE):

```
results(dds_dsx_work, contrast = c("Tissue", "BRN", "CHE"))
##
log2 fold change (MLE): Tissue BRN vs CHE
Wald test p-value: Tissue BRN vs CHE
DataFrame with 16376 rows and 6 columns
                    baseMean     log2FoldChange             lfcSE                stat
                   <numeric>          <numeric>         <numeric>           <numeric>
OTAU000001   115.269191809465 0.0788637149224524 0.0661250136003748   1.19264572706274
OTAU000002   872.366298391412  -0.22758402315813 0.0469283639983164   -4.8496048821624
OTAU000003   1476.28466132297  0.542079382638965 0.0658985683970324   8.22596599326696
OTAU000004  0.478913293424159  -1.11007363367487  0.693345455882796  -1.60103974758365
OTAU000005   20.1373870783668  0.396810733736405  0.110810583741356   3.58098225222426
...                       ...                ...                ...                ...
                      pvalue                padj
                   <numeric>           <numeric>
OTAU000001    0.233008149321211    0.296798938678064
OTAU000002 1.23707645408512e-06 2.57554676472441e-06
OTAU000003 1.93623949559107e-16 6.14925455478495e-16
OTAU000004    0.109368115856598    0.147846263991286
OTAU000005 0.000342304908190801 0.000606431141377317

summary(results(dds_dsx_work, contrast = c("Tissue", "BRN", "CHE")))
##
out of 16376 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)       : 5223, 32%
LFC < 0 (down)     : 5125, 31%
outliers [1]       : 43, 0.26%
low counts [2]     : 1905, 12%
```

```
(mean count < 0)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

Ahora la cantidad de genes que muestran aumento o reducción significativa es mucho más elevado: un total del 63% del total de genes se expresa en niveles significativamente distintos entre estos dos tejidos. Esto concuerda con las diferencias que visualizamos en los gráficos de PCA y MDS de la sección anterior.

Hay formas adicionales de crear tablas de resultados para ciertas comparaciones después de ejecutar *DESeq* una vez. Si se desean resultados para un término de interacción, se debe utilizar el argumento `name` de *results*. Consulte la página de ayuda para la función de *results* para obtener detalles sobre las formas adicionales de crear tablas de resultados. En particular, la sección de **Ejemplos** de la página de ayuda para *results* ofrece algunos ejemplos pertinentes.

5.4 – Testeos múltiples

Cuando se realizan baterías de tests estadísticos, algo habitual en análisis a nivel genómico, debemos tener cuidado de no usar los valores de *p* directamente como evidencia contra el valor nulo, sino corregirlo por *los testeos múltiples* que se realizan. ¿Qué pasaría si simplemente limitáramos los valores de *p* a un valor bajo, digamos 0.05? Hay 1868 genes con un valor *p* inferior a 0.05 entre los 16333 genes para los cuales la prueba logró informar un valor *p*:

```
sum(res$pvalue < 0.05, na.rm=TRUE)
## [1] 1868
sum(!is.na(res$pvalue))
## [1] 16333
```

Ahora, suponga por un momento que la hipótesis nula es verdadera para todos los genes, es decir, el tratamiento con *dsx* RNAi no afecta a ningún gen. Luego, según la definición del valor *p*, esperamos que hasta el 5% de los genes tengan un valor *p* inferior a 0,05. Esto equivale a 817 genes. Si acabamos de considerar la lista de genes con un valor de *p* inferior a 0,05 como expresada diferencialmente, esta lista debería contener hasta 817 /1868 = 44% de falsos positivos.

DESeq2 utiliza el ajuste Benjamini-Hochberg (BH) (Benjamini y Hochberg 1995) tal como se implementa en la función base *p.adjust* . Este método calcula para cada gen un valor *p* ajustado que responde a la siguiente pregunta: si uno llamara significativo a todos los genes con un valor *p* ajustado menor o igual al umbral del valor *p* ajustado de este gen , ¿cuál sería la fracción de falsos positivos (*tasa de descubrimiento falso*, FDR) entre ellos, en el sentido del cálculo descrito anteriormente? Estos valores, llamados valores *p* ajustados por BH, se dan en la columna `padj` del objeto `res`.

El FDR es una estadística útil para muchos experimentos de alto rendimiento, ya que a menudo estamos interesados en informar o centrarnos en un conjunto de genes interesantes, y nos gustaría poner un límite superior en el porcentaje de falsos positivos en este conjunto.

Por lo tanto, si consideramos que una fracción del 10% de falsos positivos es aceptable, podemos considerar todos los genes con un valor de *p* ajustado por debajo del 10% = 0.1 como significativos. ¿Cuántos de esos genes hay?

```
sum(res$padj < 0.1, na.rm=TRUE)
## [1] 504
```

Subseteamos la tabla de resultados a estos genes y luego los clasificamos por la estimación del cambio log2 fold para obtener los genes significativos con mayor reducción en la expresión:

```
resSig <- subset(res, padj < 0.1)
head(resSig[ order(resSig$log2FoldChange), ])
##
log2 fold change (MLE): Treatment dsx vs ctr
Wald test p-value: Treatment dsx vs ctr
DataFrame with 6 rows and 6 columns
                  baseMean    log2FoldChange             lfcSE              stat
                 <numeric>         <numeric>         <numeric>         <numeric>
OTAU011847 12.4854514878699  -4.3781490937408 0.624994783274993 -7.00509701984896
OTAU012036 16.2087620890212 -3.27697820183207 0.750593716432895 -4.36584816804158
OTAU012035 56.8385260912509  -3.2480797877935 0.531990806810374 -6.10551864094761
OTAU014218 4.32830060994666 -3.20525237437465 0.534013712540693 -6.00219114060747
OTAU014747 7.21168549847224 -2.77907569897671  0.52996831531311 -5.24385254491074
OTAU015509 10.1069483096921 -2.59571018732055 0.657428920032607 -3.94827502749925
                      pvalue                 padj
                   <numeric>            <numeric>
OTAU011847 2.46814741133027e-12 1.62416440402588e-08
OTAU012036 1.26630420977987e-05  0.00326780974606135
OTAU012035 1.02467394097202e-09 1.98143864749023e-06
OTAU014218 1.94672334482243e-09 2.32916599465527e-06
OTAU014747 1.57257894367652e-07 0.000121745361633686
OTAU015509 7.87163417204048e-05  0.00851994491227535
```

... y con mayor incremento en la expresión:

```
head(resSig[ order(resSig$log2FoldChange, decreasing = TRUE), ])
##
log2 fold change (MLE): Treatment dsx vs ctr
Wald test p-value: Treatment dsx vs ctr
DataFrame with 6 rows and 6 columns
                  baseMean   log2FoldChange             lfcSE             stat
                 <numeric>        <numeric>         <numeric>        <numeric>
OTAU000477 78.6833839437277  4.4809561505535  1.13038482789031 3.96409792487795
OTAU005345  9.2965613156511 4.27952848734207  1.21957744450357 3.50902561098451
OTAU000898 813.265950596662 3.58483258296331 0.591704053889672 6.05848913726006
OTAU007153 6539.73916957956 3.51369957549055 0.581277648846234 6.04478700060947
OTAU016735 2.42219496075558 3.38449008739917 0.971701928695525 3.48305379196141
OTAU007147   8214.332599796    3.33173729165  0.62517542150543 5.32928387304021
                      pvalue                 padj
                   <numeric>            <numeric>
OTAU000477 7.36740073894164e-05  0.00828738129275307
```

```
OTAU005345 0.000449751594082152   0.0288740523400741
OTAU000898 1.37406025107099e-09 1.98143864749023e-06
OTAU007153 1.49607516179483e-09 1.98143864749023e-06
OTAU016735 0.000495728745943115   0.0310680286921778
OTAU007147 9.86007784326983e-08 8.65123229968495e-05
```

5.5 – Generando una tabla de conteos normalizados

Si bien DESeq2 emplea la tabla de conteos sin normalizar como base para todos los análisis, a menudo es útil contar con una tabla de valores normalizados, a fin de poder hacer comparaciones directas con otros resultados experimentales distintos (aunque tales comparaciones siempre deben hacerse con cuidado, ya que no siempre se dan las condiciones para que sean válidas). Existen distintos criterios y métodos de normalización. Para este ejemplo, vamos a usar un método conocido como normalización TMM (trimmed mean of M-values, o media recortada de valores M). Este método, propuesto por Robinson & Oshlack (2010), consiste en calcular una media recortada y ponderada del logaritmo de los cocientes de expresión.

Primero invocamos la biblioteca edgeR y eliminamos de la tabla de conteos original las filas sin ningún conteo.

```
library(edgeR)
dsxCts_nozeros <- dsxCts[which(rowSums(dsxCts)>0),]
```

Luego convertimos la tabla en un objeto matrix, y generamos una lista de datos usando DGEList, para luego calcular los factores de normalización con calcNormFactors. Finalmente, usamos la función cpm para generar la tabla TMM-normalizada.

```
rnaseqMatrix = as.matrix(dsxCts_nozeros)
exp_study = DGEList(counts=rnaseqMatrix, group=factor(colnames(rnaseqMatrix)))
exp_study = calcNormFactors(exp_study)
exp_study$samples$eff.lib.size = exp_study$samples$lib.size * exp_study$samples$nor
  m.factors
dsx_TMM_matrix <- cpm(rnaseqMatrix)
```

Podemos exportar la tabla como archivo de texto delimitado por tabulador para poder usarla con otros programas (por ejemplo, Excel).

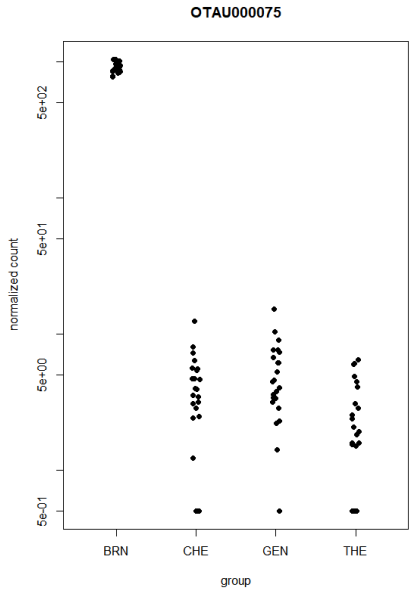
```
write.table(dsx_TMM_matrix, file="dsx.TMM_table.tsv",quote = F,sep = "\t",row.names
  = T, col.names = T)
```

6 – Graficando los resultados

6.1 - Gráfico de conteos

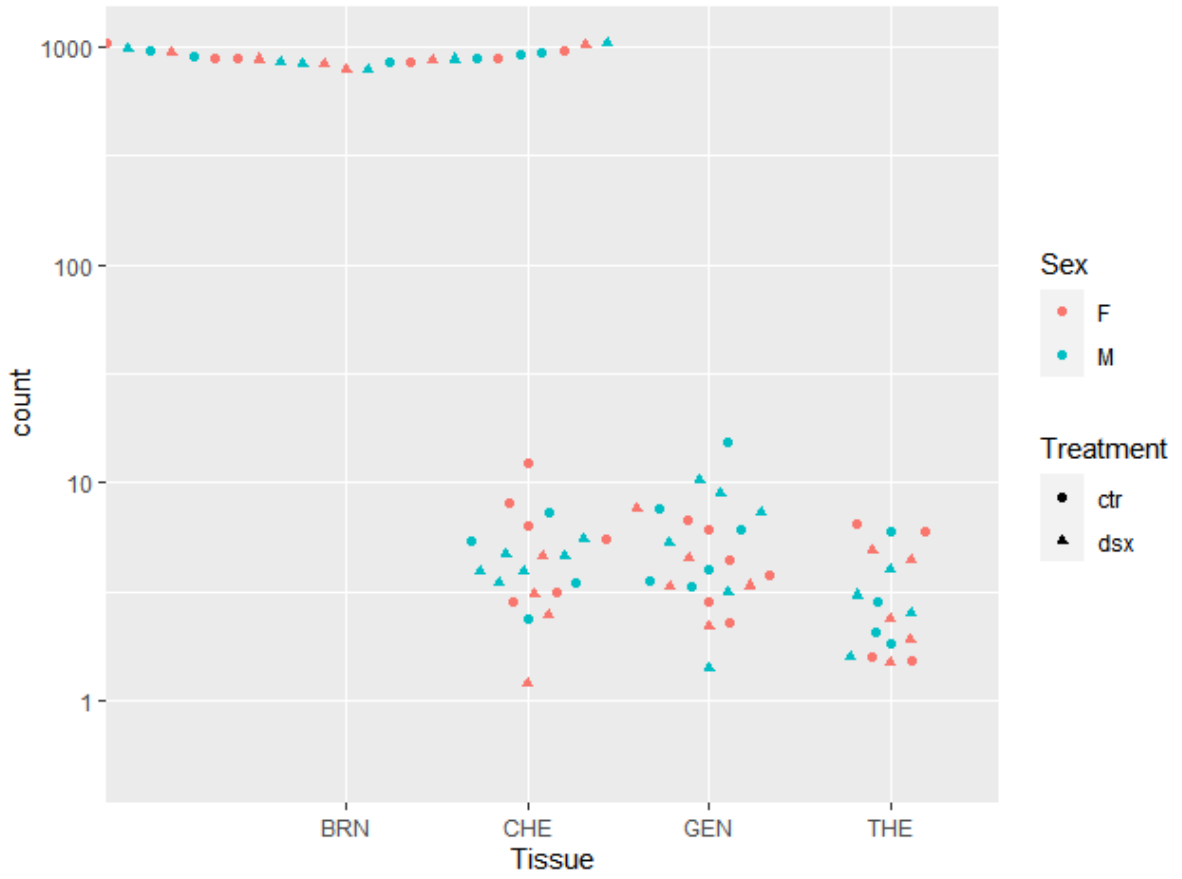
Una forma rápida de visualizar los conteos de un gen en particular es usar la función *plotCounts* que toma como argumentos el *DESeqDataSet* , un nombre de gen y el grupo sobre el cual trazar los recuentos (figura a continuación).

```
brn_che_res <- results(dds_dsx_work, contrast = c("Tissue", "BRN", "CHE"))
topGene <- rownames(brn_che_res)[which.min(brn_che_res$padj)]
plotCounts(dds_dsx_work, gene = topGene, intgroup=c("Tissue"), pch=16)
```

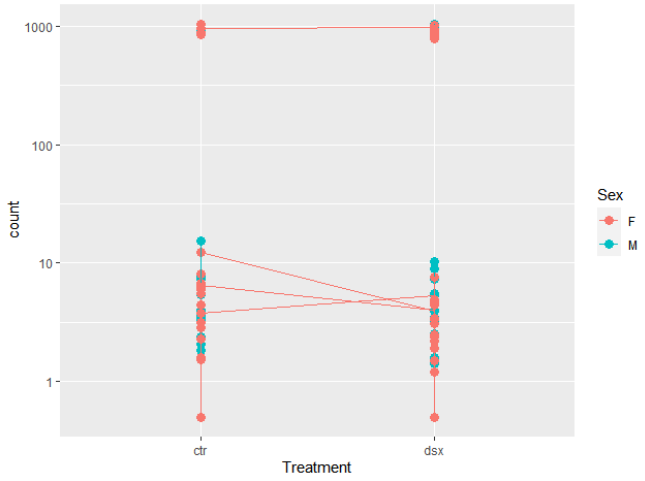


También podemos hacer gráficos personalizados utilizando la *ggplot* función del paquete *ggplot2*.

```
library("ggbeeswarm")
geneCounts <- plotCounts(dds_dsx_work, gene = topGene,
   intgroup = c("Tissue","Sex","Treatment"),returnData = TRUE)
ggplot(geneCounts, aes(x = Tissue, y = count, color = Sex, shape = Treatment)) +
  geom_beeswarm(cex = 3) +   scale_y_log10()
```




```
ggplot(geneCounts, aes(x = Treatment, y = count, color = Sex, group = Tissue)) +
  scale_y_log10() + geom_point(size = 3) + geom_line()
```



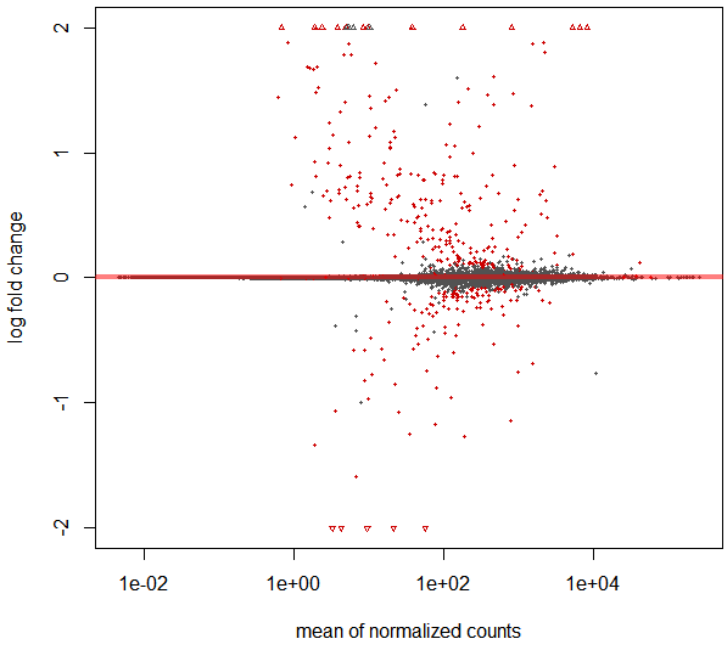
6.2 – Gráfico MA

Un *gráfico MA* (Dudoit et al. 2002) proporciona una visión general útil para la distribución de los coeficientes estimados en el modelo, por ejemplo, las comparaciones de interés, en todos los genes. En el eje y, la "M" significa "menos" (la resta del log de los valores es equivalente al log del cociente) y en el eje x, la "A" significa "promedio" (*average*). Es posible ver este gráfico denominado gráfico de diferencia-promedio o gráfico de Bland-Altman.

Antes de hacer el gráfico MA, usamos la función *lfcShrink* para reducir los cambios log2 fold para la comparación entre muestras tratadas con *dsx dsRNA* y controles. Hay tres tipos de estimadores de contracción en *DESeq2*, que están cubiertos en la viñeta DESeq2. Aquí especificamos el método *apeglm* para reducir los coeficientes, lo cual es bueno para reducir las estimaciones ruidosas de LFC a la vez que brinda estimaciones de LFC de bajo sesgo para verdaderas diferencias grandes (Zhu, Ibrahim y Love 2018). Para usar *apeglm*, especificamos un coeficiente del modelo para reducir, ya sea por nombre o número como aparece el coeficiente `resultsNames(dds_dsx_work)`.

```
library("apeglm")
resultsNames(dds_dsx_work)
##
[1] "Intercept"            "Tissue_CHE_vs_BRN"    "Tissue_GEN_vs_BRN"    "Tissue_THE
  _vs_BRN"    "Sex_M_vs_F"
[6] "Treatment_dsx_vs_ctr"
res <- lfcShrink(dds_dsx_work, coef="Treatment_dsx_vs_ctr", type="apeglm")
plotMA(res, ylim = c(-2, 2))
```

Si es necesario especificar un contraste no representado en `resultsNames(dds_dsx_work)`, cualquiera de los otros dos métodos de contracción puede ser utilizado, o en algunos casos, re-factorizar las variables relevantes y funcionando `nbinomWaldTest` seguido por `lfcShrink` es suficiente. Vea la viñeta DESeq2 para más detalles.

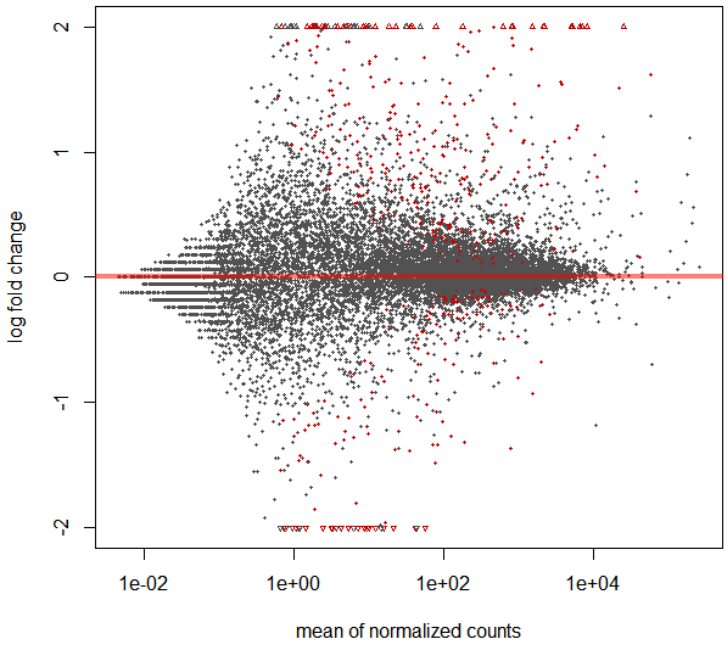


Un diagrama MA de cambios inducidos por el tratamiento. El cambio log2 fold para una comparación particular se traza en el eje y y el promedio de los recuentos normalizados por factor de tamaño se muestra en el eje x. Cada gen está representado con un punto. Los genes con un valor *p* ajustado por debajo de un umbral (aquí 0.1, el valor predeterminado) se muestran en rojo.

El paquete *DESeq2* utiliza un procedimiento bayesiano para moderar (o "reducir") los cambios de log2 fold de genes con recuentos muy bajos y recuentos muy variables, como se puede ver por el estrechamiento de la extensión vertical de puntos en el lado izquierdo del gráfico MA. Como se muestra arriba, la función *lfcShrink* realiza esta operación. Para obtener una explicación detallada de la justificación de los cambios moderados de pliegue, consulte el documento *DESeq2* (Love, Huber y Anders 2014).

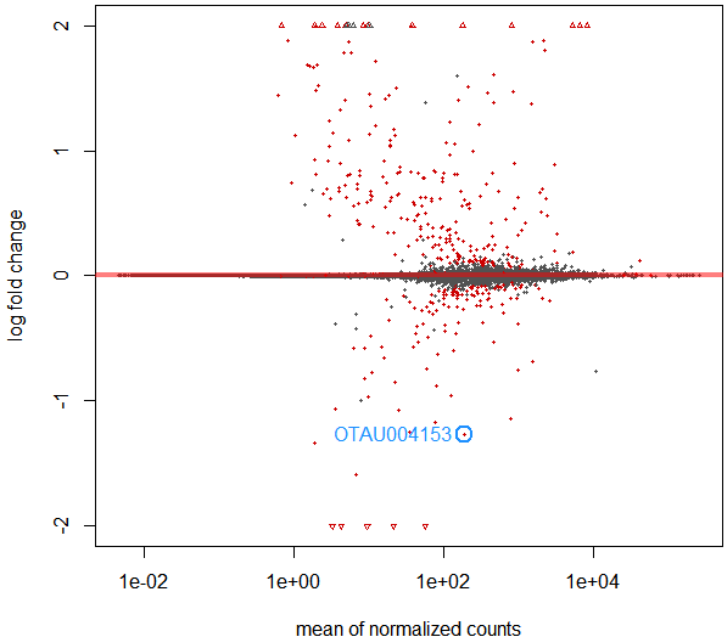
Si no hubiéramos usado la moderación estadística para reducir los ruidosos cambios de log2 veces, habríamos visto el siguiente diagrama:

```
res.noshr <- results(dds_dsx_work, name="Treatment_dsx_vs_ctr")
plotMA(res.noshr, ylim = c(-2, 2))
```



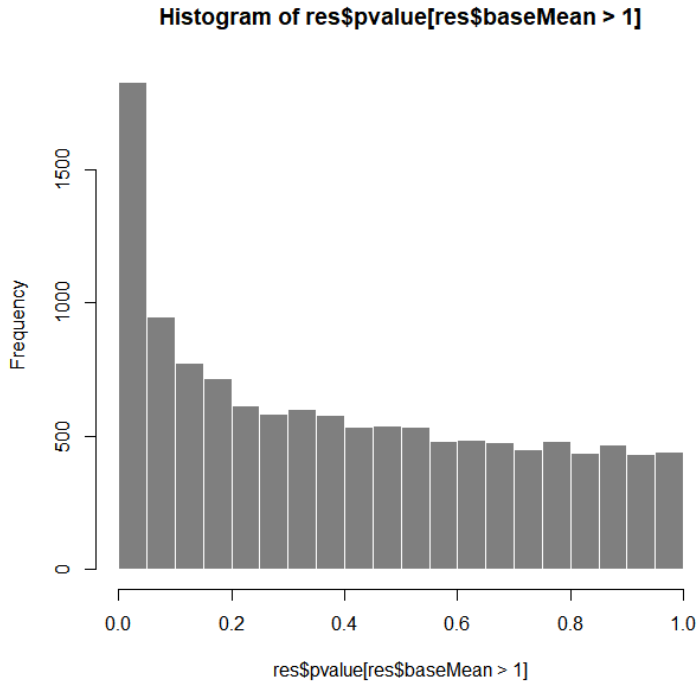
También podemos etiquetar puntos individuales en el diagrama MA. Aquí usamos la función *with* para trazar un círculo y texto para una fila seleccionada del objeto de resultados. Dentro de la función *with*, solo se utilizan los valores `baseMean` y `log2FoldChange` para las filas seleccionadas de `res`.

```
plotMA(res, ylim = c(-2,2))
topGene <- rownames(res)[which.min(res$padj)]
with(res[topGene, ], {
  points(baseMean, log2FoldChange, col="dodgerblue", cex=2, lwd=2)
  text(baseMean, log2FoldChange, topGene, pos=2, col="dodgerblue")
})
```



Otro gráfico de diagnóstico útil es el histograma de los valores de *p* (figura a continuación). Este gráfico se forma mejor excluyendo genes con recuentos muy pequeños, que de lo contrario generan picos en el histograma.

```
hist(res$pvalue[res$baseMean > 1], breaks = 0:20/20,
     col = "grey50", border = "white")
```



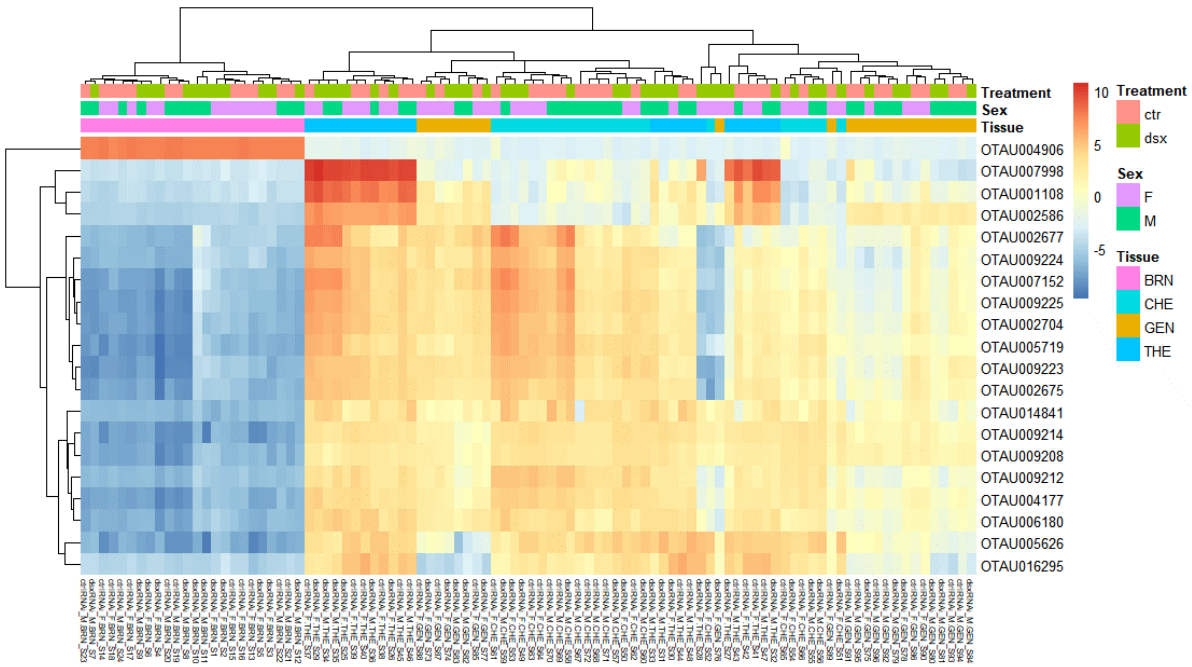
6.3 - **Agrupación de genes (clustering)**

En el mapa de calor de distancia entre muestras realizado previamente, el dendrograma al costado nos muestra un agrupamiento jerárquico de las muestras. Tal agrupación también se puede realizar para los genes. Dado que la agrupación solo es relevante para los genes que realmente tienen una señal, generalmente solo se agruparía un subconjunto de los genes más altamente variables. Aquí, para demostración, seleccionemos los 20 genes con la mayor varianza entre las muestras. Trabajaremos con los datos VST.

```
library("genefilter")
topVarGenes <- head(order(rowVars(assay(vsd)), decreasing = TRUE), 20)
```

El mapa de calor se vuelve más interesante si no observamos la fuerza de expresión absoluta, sino más bien la cantidad en que cada gen se desvía en una muestra específica del promedio del gen en todas las muestras. Por lo tanto, centramos los valores de cada gen en las muestras y graficamos un mapa de calor (figura a continuación). Proporcionamos un *data.frame* que instruye a la función *pheatmap* sobre cómo etiquetar las columnas.

```
mat  <- assay(vsd)[ topVarGenes, ]
mat  <- mat - rowMeans(mat)
anno <- as.data.frame(colData(vsd)[, c("Tissue","Sex","Treatment")])
pheatmap(mat, annotation_col = anno, fontsize_col = 6)
```



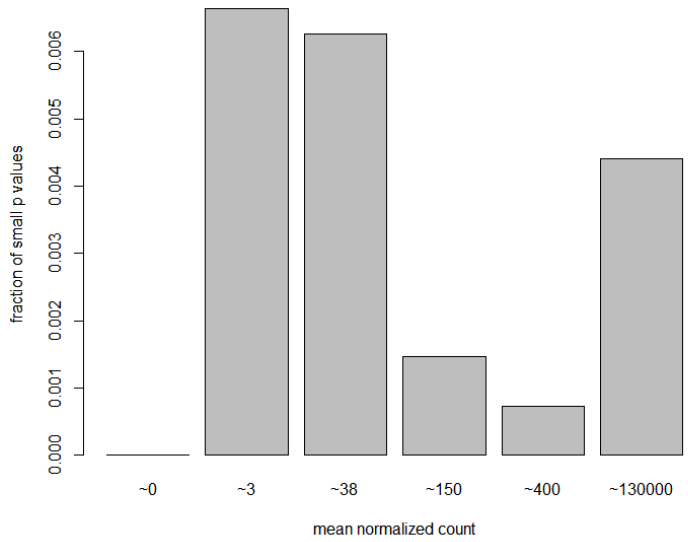
Mapa de calor de los valores relativos transformados de VST de las muestras. El tejido, sexo y tratamiento se muestran con barras de colores en la parte superior del mapa de calor.

6.4 - Filtrado independiente

El diagrama MA destaca una propiedad importante de los datos de RNA-seq. Para los genes débilmente expresados, no tenemos posibilidad de ver una expresión diferencial, porque los recuentos bajos de lectura sufren de un ruido de Poisson tan alto que cualquier efecto biológico se ahoga en las incertidumbres de un tamaño de muestreo bajo. También podemos mostrar esto examinando la proporción de valores *p* pequeños (digamos, menos de 0.05) para genes agrupados por conteo normalizado promedio. Usaremos la tabla de resultados sujeta al umbral para mostrar cómo se ve esto en un caso cuando hay pocas pruebas con un valor *p* pequeño.

En el siguiente fragmento de código, creamos bins usando la función *cuantil*, agrupamos los genes por el promedio base usando *cut*, cambiamos el nombre de los niveles de los bins usando el punto medio, calculamos la proporción de valores de *p* menores a 0.05 para cada bin, y finalmente graficamos estas proporciones (figura a continuación).

```
qs <- c(0, quantile(resLFC1$baseMean[resLFC1$baseMean > 0], 0:6/6))
bins <- cut(resLFC1$baseMean, qs)
levels(bins) <- paste0("~", round(signif((qs[-1] + qs[-length(qs)])/2, 2)))
fractionSig <- tapply(resLFC1$pvalue, bins, function(p)
                          mean(p < .05, na.rm = TRUE))
barplot(fractionSig, xlab = "mean normalized count",
                     ylab = "fraction of small p values")
```



La proporción de valores *p* pequeños para genes agrupados por recuento normalizado medio. Los valores de *p* derivan de testear por un cambio log2 fold mayor que 1 o menor que -1. Este gráfico demuestra que los genes con un conteo promedio muy bajo tienen poca o ninguna potencia, y es mejor excluirlos de las pruebas.

A primera vista, puede parecer que hay pocos beneficios en el filtrado de estos genes. Después de todo, la prueba los encontró no significativos de todos modos. Sin embargo, estos genes influyen en el ajuste de las pruebas múltiples, cuyo rendimiento mejora si se eliminan dichos genes. Al eliminar los genes de recuento bajo de la entrada al procedimiento FDR, podemos encontrar que más genes son significativos entre los que conservamos, y así mejoramos el poder de nuestra prueba. Este enfoque se conoce como *filtrado independiente*.

El software *DESeq2* realiza automáticamente un filtrado independiente que maximiza el número de genes con un valor *p* ajustado menor que un valor crítico (por defecto, `alpha` se establece en 0.1). Este filtrado automático independiente se realiza y puede controlarse mediante la función de *results*.

El término *independiente* destaca una advertencia importante. Tal filtrado es permisible solo si el estadístico mediante el cual filtramos (en este ejemplo, la media de conteos normalizados en todas las muestras) es independiente del estadístico del test (el valor *p*) bajo la hipótesis nula. De lo contrario, el filtrado invalidaría la prueba y, en consecuencia, los supuestos del procedimiento BH. El software de filtrado independiente utilizado dentro de *DESeq2* proviene del paquete de *genefilter* , que contiene una referencia a un documento que describe la base estadística para el filtrado independiente (Bourgon, Gentleman y Huber 2010) .

6.5 – Ponderación de hipótesis independiente

Una generalización de la idea del filtrado del valor *p* es *ponderar las* hipótesis para optimizar la potencia. Se encuentra disponible un paquete de *bioconductor*, *IHW* , que implementa el método de *ponderación de hipótesis independiente* (Ignatiadis et al.2016) . Consulte la *viñeta del* paquete *DESeq2* para ver un ejemplo del uso de *IHW* en combinación con *DESeq2*. En particular, el siguiente fragmento de código (aquí no evaluado) se puede utilizar para realizar IHW en lugar del filtrado independiente descrito anteriormente.

```
library("IHW")
res.ihw <- results(dds_dsx_work, filterFun=ihw)
```

7 – Análisis pormenorizado de datos

Planteo de hipótesis

Antes de seguir en el análisis, conviene plantear hipótesis concretas buscando responder a preguntas específicas. Estas preguntas conllevan a plantear contrastes específicos. A su vez, permiten definir la escala de resolución de la pregunta.

Por ejemplo, uno podría preguntarse si el tamaño del repertorio de genes asociado a un dimorfismo sexual es proporcional a la exageración (diferencia morfológica entre morfos) de dicho dimorfismo. En este caso, se plantean contrastes paralelos para tejidos que varían en su nivel de exageración: en el caso de *Onthophagus taurus*, es posible comparar entre sexos las diferencias de repertorio de genes asociados al desarrollo de un tejido fuertemente dimórfico (como el dorso de la cabeza donde los machos forman cuernos y las hembras no) con un tejido moderadamente dimórfico (como es el dorso del protórax, donde los machos forman una prominencia levemente más marcada que las hembras). Bajo la hipótesis de que hay una correlación entre el nivel de exageración y la cantidad de genes diferencialmente expresados, se espera que la cantidad de genes diferencialmente expresados entre machos y hembras sea mayor en la epidermis dorsal de la cabeza que en la epidermis dorsal del protórax.

Otro tipo de hipótesis reflejan mejor los mecanismos putativos que podrían explicar esos dimorfismos. Usando el mismo ejemplo, se puede plantear que el crecimiento exagerado de los cuernos en machos se debe a una sobrerregulación de genes que dirigen procesos de división celular y crecimiento de los tejidos. Bajo esta hipótesis, uno espera que genes asociados a estos procesos se encuentren “más” diferencialmente expresados (ya sea en valores absolutos de fold-change mayores, o en enriquecimiento del repertorio de genes) en el contraste macho vs. hembra del tejido de la cabeza que en el protórax.

Del mismo modo es posible plantear hipótesis sobre regulación por un factor “maestro” de transcripción que controla/modula una gran cantidad de procesos de desarrollo. *Doublesex* (*dsx*) es un candidato a ese rol. Es posible plantear que gran parte de los procesos de desarrollo asociados a los dimorfismos sexuales están controlados por un único gen “maestro” que traduce la información sobre el sexo del individuo a los tejidos y células encargados de manifestar este dimorfismo. Bajo esta hipótesis, se espera que, si se inhibe la expresión de *dsx* mediante RNAi, una fracción considerable de los genes que en condiciones normales se expresa diferencialmente ahora dejarán de estarlo. Al igual que en el caso anterior, es posible plantear el análisis a nivel más mecanístico, individualizando genes o vías genéticas putativas y testeando su respuesta ante la inhibición de *dsx*.

Subconjuntos para ejercicios

Para practicar el planteo de hipótesis y el análisis de los datos de RNAseq, vamos a dividir en distintas preguntas el set de datos. En otras palabras, vamos a reducir en un factor el conjunto disponible de datos para plantear tres subconjuntos con solo dos factores de variación. Cada uno de los subconjuntos responde a preguntas específicas. Los tres subconjuntos son:

1 – Transcriptómica del cerebro:

Si bien la morfología externa del cerebro de *Onthophagus taurus* no muestra dimorfismos sexuales obvios, es muy probable que las diferencias de comportamiento entre sexos estén mediadas, al menos en parte, por diferencias neurales resultantes de patrones de expresión diferencial durante el desarrollo de este tejido durante la metamorfosis. ¿Cuántas de estas diferencias entre sexos están potencialmente bajo control por *dsx*? ¿Qué genes o familias de genes están asociados a estas diferencias entre sexos?

Para poder contestar estas preguntas, se pueden usar los datos específicos de cerebro (BRN) del conjunto de datos de RNAseq.

```
#### Seleccionando las muestras de cerebro y generando el objeto dds de trabajo----
dsxCts_BRN <- dsxCts %>% select(rownames(dsxColdata %>% filter(Tissue=="BRN")))
dds_dsx_BRN <- DESeqDataSetFromMatrix(countData = dsxCts_BRN,
                                      colData = dsxColdata %>% filter(Tissue=="BRN"),
                                      design = ~ Treatment + Sex)
```

2 – Sesgo sexual y nivel de dimorfismo:

En *Onthophagus taurus*, la cabeza de las pupas masculinas y femeninas se diferencia por la presencia de enormes cuernos en las primeras, y su ausencia en las segundas. En cambio, ambos sexos muestran un cuerno torácico con un dimorfismo mucho más moderado. ¿Correlaciona esta diferencia entre niveles de dimorfismo con el tamaño y composición de los repertorios de genes expresados diferencialmente entre sexos en cada tejido? ¿Hay genes que presentan un sesgo sexual consistente entre ambos tejidos? ¿Qué genes o familias de genes están asociados a estas diferencias entre sexos en cada tejido?

Para poder contestar estas preguntas, se pueden usar los datos específicos de los individuos con tratamiento control (ctr) y tejidos dorsal cefálico (CHE) y dorsal protorácico (THE) del conjunto de datos de RNAseq.

```
#### Seleccionando las muestras necesarias y generando el objeto dds de trabajo----
dsxCts_dorsal <- dsxCts %>% select(rownames(dsxColdata
                 %>% filter(Treatment == "ctr", Tissue=="CHE", Tissue=="THE")))
dds_dsx_ dorsal <- DESeqDataSetFromMatrix(countData = dsxCts_ dorsal,
                                      colData = dsxColdata
                 %>% filter((Treatment == "ctr", Tissue=="CHE", Tissue=="THE")),
                                      design = ~ Sex + Tissue)
```

3 – Bases transcriptómicas del dimorfismo cefálico

Los machos de *Onthophagus taurus* compiten por las hembras en la entrada de los túneles, y han evolucionado cuernos exageradamente grandes. Las hembras en cambio pasan buena parte de su vida excavando túneles en el suelo bajo los parches de estiércol; los cuernos resultan muy imprácticos para esta tarea, y por tanto carecen de cuernos. ¿Cuántos genes hacen falta para generar tan marcada diferencia entre sexos? ¿Cuantos de estos genes están potencialmente bajo el control de *dsx*? ¿Qué genes o familias de genes están implicados en la formación de cuernos?

Para poder contestar estas preguntas, se pueden usar los datos específicos de cerebro (BRN) del conjunto de datos de RNAseq.

```
#### Seleccionando las muestras de epidermis dorsal y generando el objeto dds ----
dsxCts_CHE <- dsxCts %>% select(rownames(dsxColdata %>% filter(Tissue=="CHE")))
dds_dsx_CHE <- DESeqDataSetFromMatrix(countData = dsxCts_CHE,
                                      colData = dsxColdata %>% filter(Tissue=="CHE"),
                                      design = ~ Treatment + Sex)
```

8 – Referencias

Anders, Simon, and Wolfgang Huber. 2010. “Differential expression analysis for sequence count data.” *Genome Biology* 11 (10):R106+. https://doi.org/10.1186/gb-2010-11-10-r106.

Anders, Simon, Paul T. Pyl, and Wolfgang Huber. 2015. “HTSeq – a Python framework to work with high-throughput sequencing data.” *Bioinformatics* 31 (2):166–69. https://doi.org/10.1093/bioinformatics/btu638.

Benjamini, Yoav, and Yosef Hochberg. 1995. “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing.” *Journal of the Royal Statistical Society. Series B (Methodological)* 57 (1):289–300. http://www.jstor.org/stable/2346101.

Bourgon, R., R. Gentleman, and W. Huber. 2010. “Independent filtering increases detection power for high-throughput experiments.” *Proceedings of the National Academy of Sciences* 107 (21):9546–51. https://doi.org/10.1073/pnas.0914005107.

Bray, Nicolas, Harold Pimentel, Pall Melsted, and Lior Pachter. 2016. “Near-Optimal Probabilistic Rna-Seq Quantification.” *Nature Biotechnology* 34:525–27. http://dx.doi.org/10.1038/nbt.3519.

Casasa, A. Sofía, Eduardo E. Zattara, and Armin P. Moczek. 2020. “Nutrition-responsive gene expression and the developmental evolution of insect polyphenism.” Nature Ecology & Evolution. https://doi.org /10.1038/s41559-020-1202-x.

Dudoit, Rine, Yee H. Yang, Matthew J. Callow, and Terence P. Speed. 2002. “Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments.” *Statistica Sinica*, 111–39.

Frankish, Adam, Alexandra Bignell, Andrew Berry, Andrew Yates, Anne Parker, Bianca M Schmitt, Bronwen Aken, et al. 2018. “GENCODE reference annotation for the human and mouse genomes.” *Nucleic Acids Research* 47 (D1):D766–D773.

Hardcastle, Thomas, and Krystyna Kelly. 2010. “baySeq: Empirical Bayesian methods for identifying differential expression in sequence count data.” *BMC Bioinformatics* 11 (1):422+. https://doi.org/10.1186/1471-2105-11-422.

Himes, Blanca E., Xiaofeng Jiang, Peter Wagner, Ruoxi Hu, Qiyu Wang, Barbara Klanderman, Reid M. Whitaker, et al. 2014. “RNA-Seq transcriptome profiling identifies CRISPLD2 as a glucocorticoid responsive gene that modulates cytokine function in airway smooth muscle cells.” *PloS One* 9 (6). https://doi.org/10.1371/journal.pone.0099625.

Huber, Wolfgang, Vincent J. Carey, Robert Gentleman, Simon Anders, Marc Carlson, Benilton S. Carvalho, Hector Corrada C. Bravo, et al. 2015. “Orchestrating high-throughput genomic analysis with Bioconductor.” *Nature Methods* 12 (2). Nature Publishing Group:115–21. https://doi.org/10.1038/nmeth.3252.

Huntley, Melanie A., Jessica L. Larson, Christina Chaivorapol, Gabriel Becker, Michael Lawrence, Jason A. Hackney, and Joshua S. Kaminker. 2013. “ReportingTools: an automated result processing and presentation toolkit for high-throughput genomic analyses.” *Bioinformatics* 29 (24). Oxford University Press:3220–1. https://doi.org/10.1093/bioinformatics/btt551.

Ignatiadis, Nikolaos, Bernd Klaus, Judith Zaugg, and Wolfgang Huber. 2016. “Data-Driven Hypothesis Weighting Increases Detection Power in Genome-Scale Multiple Testing.” *Nature Methods*. http://dx.doi.org/10.1038/nmeth.3885.

Köster, Johannes, and Sven Rahmann. 2012. “Snakemake - A scalable bioinformatics workflow engine.” *Bioinformatics*. https://doi.org/10.1093/bioinformatics/bts480.

Law, Charity W., Yunshun Chen, Wei Shi, and Gordon K. Smyth. 2014. “Voom: precision weights unlock linear model analysis tools for RNA-seq read counts.” *Genome Biology* 15 (2). BioMed Central Ltd:R29+. https://doi.org/10.1186/gb-2014-15-2-r29.

Lawrence, Michael, Wolfgang Huber, Hervé Pagès, Patrick Aboyoun, Marc Carlson, Robert Gentleman, Martin T. Morgan, and Vincent J. Carey. 2013. “Software for Computing and Annotating Genomic

Ranges." Edited by Andreas Prlic. *PLoS Computational Biology* 9 (8). Public Library of Science:e1003118+. https://doi.org/10.1371/journal.pcbi.1003118.

Ledón-Rettig, C.C., E.E. Zattara, and A.P. Moczek. 2017. Asymmetric interactions between doublesex and tissue- and sex-specific target genes mediate sexual dimorphism in beetles 8 (February 27): 14593. https://doi.org/10.1038/ncomms14593.

Leek, Jeffrey T. 2014. "svaseq: removing batch effects and other unwanted noise from sequencing data." *Nucleic Acids Research* 42 (21). Oxford University Press:000. https://doi.org/10.1093/nar/gku864.

Leng, N., J. A. Dawson, J. A. Thomson, V. Ruotti, A. I. Rissman, B. M. G. Smits, J. D. Haag, M. N. Gould, R. M. Stewart, and C. Kendziorski. 2013. "EBSeq: an empirical Bayes hierarchical model for inference in RNA-seq experiments." *Bioinformatics* 29 (8). Oxford University Press:1035–43. https://doi.org/10.1093/bioinformatics/btt087.

Leong, Hui S., Keren Dawson, Chris Wirth, Yaoyong Li, Yvonne Connolly, Duncan L. Smith, Caroline R. Wilkinson, and Crispin J. Miller. 2014. "A global non-coding RNA system modulates fission yeast protein levels in response to stress." *Nature Communications* 5. https://doi.org/10.1038/ncomms4947.

Li, Bo, and Colin N. Dewey. 2011. "RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome." *BMC Bioinformatics* 12:323+. https://doi.org/10.1186/1471-2105-12-3231.

Liao, Y., G. K. Smyth, and W. Shi. 2014. "featureCounts: an efficient general purpose program for assigning sequence reads to genomic features." *Bioinformatics* 30 (7). Oxford University Press:923–30. https://doi.org/10.1093/bioinformatics/btt656.

Love, Michael I., John B. Hogenesch, and Rafael A. Irizarry. 2016. "Modeling of Rna-Seq Fragment Sequence Bias Reduces Systematic Errors in Transcript Abundance Estimation." *Nature Biotechnology* 34 (12):1287–91. http://dx.doi.org/10.1038/nbt.3682.

Love, Michael I., Wolfgang Huber, and Simon Anders. 2014. "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2." *Genome Biology* 15 (12). BioMed Central Ltd:550+. https://doi.org/10.1186/s13059-014-0550-8.

Love, Michael I., Charlotte Soneson, Peter F. Hickey, Lisa K. Johnson, N. Tessa Pierce, Lori Shepherd, Martin Morgan, and Rob Patro. 2020. "Tximeta: Reference sequence checksums for provenance identification in RNA-seq." *PLOS Computational Biology*. https://doi.org/10.1371/journal.pcbi.1007664.

Patro, Rob, Geet Duggal, Michael I. Love, Rafael A. Irizarry, and Carl Kingsford. 2017. "Salmon Provides Fast and Bias-Aware Quantification of Transcript Expression." *Nature Methods*. http://dx.doi.org/10.1038/nmeth.4197.

Risso, Davide, John Ngai, Terence P. Speed, and Sandrine Dudoit. 2014. "Normalization of RNA-seq data using factor analysis of control genes or samples." *Nature Biotechnology* 32 (9). Nature Publishing Group:896–902. https://doi.org/10.1038/nbt.2931.

Robert, Christelle, and Mick Watson. 2015. "Errors in RNA-Seq quantification affect genes of relevance to human disease." *Genome Biology*. https://doi.org/10.1186/s13059-015-0734-x.

Robinson, M. D., D. J. McCarthy, and G. K. Smyth. 2009. "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." *Bioinformatics* 26 (1). Oxford University Press:139–40. https://doi.org/10.1093/bioinformatics/btp616.

Robinson, M.D., and A. Oshlack. 2010. "A scaling normalization method for differential expression analysis of RNA-seq data." *Genome Biology* 11: R25. https://doi.org/10.1186/gb-2010-11-3-r25.

Schurch, Nicholas J., Pieta Schofield, Marek Gierlinski, Christian Cole, Alexander Sherstnev, Vijender Singh, Nicola Wrobel, et al. 2016. "How Many Biological Replicates Are Needed in an Rna-Seq Experiment and Which Differential Expression Tool Should You Use?" 22 (6):839–51. https://doi.org/10.1261/rna.053959.115.

Soneson, Charlotte, Michael I. Love, and Mark Robinson. 2015. “Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences.” *F1000Research* 4 (1521). https://doi.org/10.12688/f1000research.7563.1.

Tonner, Peter D, Cynthia L Darnell, Barbara E Engelhardt, and Amy K Schmid. 2017. “Detecting differential growth of microbial populations with Gaussian process regression.” *Genome Research* 27:320–33. https://doi.org/10.1101/gr.210286.116.

Townes, F. William, Stephanie C. Hicks, Martin J. Aryee, and Rafael A. Irizarry. 2019. “Feature Selection and Dimension Reduction for Single Cell Rna-Seq Based on a Multinomial Model.” *bioRxiv*. Cold Spring Harbor Laboratory. https://doi.org/10.1101/574574.

Trapnell, Cole, David G Hendrickson, Martin Sauvageau, Loyal Goff, John L Rinn, and Lior Pachter. 2013. “Differential analysis of gene regulation at transcript resolution with RNA-seq.” *Nature Biotechnology*. https://doi.org/10.1038/nbt.2450.

Wickham, Hadley. 2009. *ggplot2*. New York, NY: Springer New York. https://doi.org/10.1007/978-0-387-98141-3.

Witten, Daniela M. 2011. “Classification and clustering of sequencing data using a Poisson model.” *The Annals of Applied Statistics* 5 (4):2493–2518. https://doi.org/10.1214/11-AOAS493.

Wu, Hao, Chi Wang, and Zhijin Wu. 2013. “A new shrinkage estimator for dispersion improves differential expression detection in RNA-seq data.” *Biostatistics* 14 (2). Oxford University Press:232–43. https://doi.org/10.1093/biostatistics/kxs033.

Zhu, Anqi, Joseph G. Ibrahim, and Michael I. Love. 2018. “Heavy-Tailed Prior Distributions for Sequence Count Data: Removing the Noise and Preserving Large Differences.” *bioRxiv*. https://doi.org/10.1101/303255.