## Importación de *paquetes*

Si la instalación de todo funcionó, entonces podemos cargar los paquetes en la sesión de trabajo de R actual. A diferencia de la instalación, esta parte hay que correrla cada vez que abramos una nueva sesión en R.

# Definimos la lista de nombres de los paquetes que R va a llamar  
  
pckg.ls=c("DESeq2", "vsn", "apeglm", "genefilter", "IHW", "edgeR", # análiis de datos moleculares  
 "dplyr", "tidyr", # manipulación de datos  
 "ggplot2", # generación de gráficos  
 "pheatmap", # gráficos de mapas de calor  
 "RColorBrewer", # paletas de colores para los gráficos  
 "PoiClaClu", # cálculo de distancias de Poisson  
 "ggbeeswarm",   
 'gridExtra', # múltiplies gráficos en un único panel  
 'colorspace' # configuraciones de color  
 )  
  
# E importamos los paquetes de esa lista  
  
lapply(pckg.ls, # para cada nombre de la lista de paquetes instalados...  
 require, # aplicar la función "require" para importarlo  
 quietly =T,  
 character.only = TRUE) # (ignorar este argumento, es para las mañas de R)

## Importación y preparación de los datos

### Tabla de conteos

La tabla de conteos tiene, en cada **celda**, el número de fragmentos de RNA secuenciados de cada *muestra* que fueron identificados como provenientes de cada *gen*. Por lo tanto, tiene tantas *columnas* como muestras hayamos utilizado como fuente de RNA, y tantas *filas* como genes para los cuales se registró el numero de fragmentos de RNA secuenciados.

# importación  
cts <- read.delim("./Conteos.tsv", row.names = "gene")

Podemos explorar las primeras filas y columnas para tener una imágen de cómo se organiza esta tabla.

# filas de 1 a 6, columnas de 1 a 3  
cts[1:6, 1:3]

## ctrlRNAi\_M.BRN\_S19 ctrlRNAi\_M.BRN\_S20 ctrlRNAi\_M.BRN\_S21  
## OTAU000001 166 155 195  
## OTAU000002 955 975 976  
## OTAU000003 2318 2228 2466  
## OTAU000004 2 1 0  
## OTAU000005 29 26 38  
## OTAU000006 97 78 73

Vemos que cada **fila** tiene el nombre asociado al gen correspondiente. OTAU000001 no es el *nombre* de un gen propiamente dicho, sino que es el identificador que se le asignó a esa región codificante en el genoma que estamos utilizando. Por otra parte, en cada **columna** vemos un nombre asociado a la muestra de donde se sacó el RNA para secuenciar. Estos nombres tienen la información sobre el *diseño experimental*, es decir, el tratamiento con RNA de interferencia sobre dsx (ctrl/trat), el sexo (M/F), y el tejido (BRN/CHE/GEN/THE). ctrlRNAi\_M.BRN\_S19 es una muestra del cerebro (BRN = brain) de un macho (M = male) al que no se le aplicó interferencia de RNA dirigida a dsx (ctrl).

**¿Qué función/es de R podríamos utilizar para conocer la cantidad de genes y de muestras en estos datos? ¿Cuántos genes hay? ¿Y muestras?**

### Tabla de diseño experimental

Si bien los nombres de las columnas de la tabla de conteos tienen la información sobre el diseño experimental (tratamiento, sexo y tejido), esta es una fuente un poco incómoda para comparar los patrones de expresión entre grupos de forma programática. Para realizar análisis de comparación entre grupos, utilizamos una segunda tabla que tiene toda la información sobre las muestras de forma más ordenada.

samples <- read.csv("./Muestras\_reord.tsv", row.names=1)

Al igual que antes, podemos explorar visualmente la tabla mostrando solo algunas filas

# filas de 1 a 8, todas las columnas  
samples[1:8,]

## Treatment Sex Tissue sample lib.size  
## ctrlRNAi\_F.BRN\_S13 ctr F BRN S13 9315408  
## ctrlRNAi\_F.BRN\_S14 ctr F BRN S14 15622406  
## ctrlRNAi\_F.BRN\_S15 ctr F BRN S15 10385722  
## ctrlRNAi\_F.BRN\_S16 ctr F BRN S16 12022532  
## ctrlRNAi\_F.BRN\_S17 ctr F BRN S17 10439212  
## ctrlRNAi\_F.BRN\_S18 ctr F BRN S18 10909008  
## dsxRNAi\_F.BRN\_S1 dsx F BRN S1 11987284  
## dsxRNAi\_F.BRN\_S2 dsx F BRN S2 13156881

**¿A qué cree que hace referencia la útima columna?**

### Selección de muestras

Para seleccionar el conjunto de muestras a utilizar, vamos a generar un vector lógico (de TRUE/FALSE) de 96 elementos (uno por cada muestra). La idea es que este vector tenga valores T en las posiciones de las muestras que queremos conservar para el análisis y F para aquellas que queremos descartar.

tejidos=c('CHE', 'THE', 'BRN', 'GEN')  
sex=c('F', 'M')  
trat=c('ctr', 'dsx')  
  
idx.muestras= samples$Tissue %in% tejidos & # TRUE para las muestras pertenecientes (%in%) a los tejidos defeinidos (todos)  
 samples$Sex %in% sex & # TRUE para las muestras pertenecientes (%in%) a los sexos defeinidos (todos)  
 samples$Treatment %in% trat # TRUE para las muestras pertenecientes (%in%) a los tratamientos defeinidos (todas)

Corra la suigiente linea en la consola para ver cómo quedan asignados los valores T/F a cada muestra.

cbind(samples, idx.muestras)

### Combinación en un único dataset

Para usar la mayoría de herramientas de análisis de este TP es necesario combinar estas dos tablas en un único *objeto*. Para esto, primero tenemos que chequear que cada columna de la tabla de conteos sea asociable inequívocamente a la fila correspondiente en la tabla de muestras.

cts=cts[,rownames(samples)] # para corregir diferencias en el orden de las columnas-filas entre tablas  
  
all(rownames(samples) == colnames(cts)) # debe devolver TRUE

## [1] TRUE

Una vez que chequeamos esto podemos unir ambas tablas en un tipo de objeto utilizado por las herramientas de análisis de expresión diferencial (DE) incluidas en el paquete DESeq:

#objeto DESeqDataSet  
data\_0 <- DESeqDataSetFromMatrix(countData = cts[,idx.muestras],  
 colData = samples[idx.muestras, c("Sex", "Tissue", "Treatment")],  
 design = ~ Sex + Tissue + Treatment)

Podemos explorar la información general de este objeto si corremos su nombre en la consola.

data\_0

## class: DESeqDataSet   
## dim: 17483 96   
## metadata(1): version  
## assays(1): counts  
## rownames(17483): OTAU000001 OTAU000002 ... OTAU017482 OTAU017483  
## rowData names(0):  
## colnames(96): ctrlRNAi\_F.BRN\_S13 ctrlRNAi\_F.BRN\_S14 ...  
## dsxRNAi\_M.GEN\_S83 dsxRNAi\_M.GEN\_S84  
## colData names(3): Sex Tissue Treatment

## Filtrado del dataset

Podemos descartar genes que tengan una cantidad de fragmentos secuenciados (reads) por debajo de un determinado umbral.

# Probar con diferentes valores  
reads.minimos=100  
  
# genes sobre el umbral (suficientes reads)  
c(rowSums(counts(data\_0)) >= reads.minimos ) %>% sum()

## [1] 12472

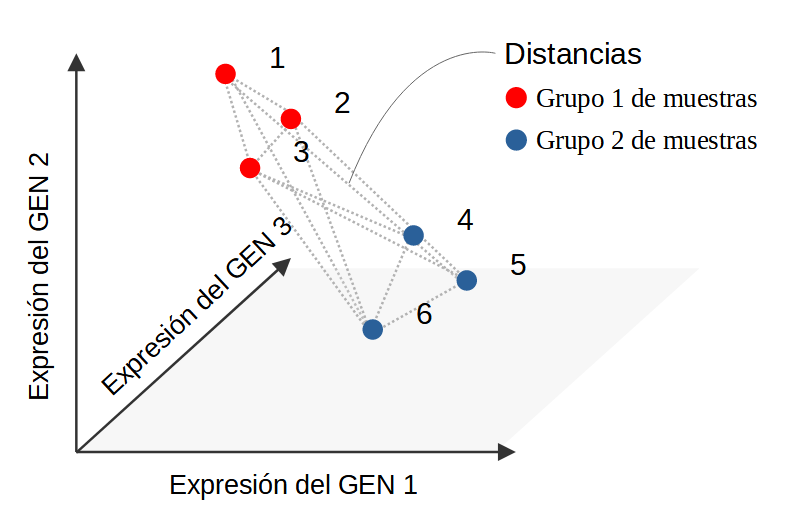
Para el resto del trabajo vamos a utilizar un umbral de 10 lecturas para considerar genes con suficiente información, pero sin descartar demasiados.

reads.minimos=10  
  
data\_filtered <- data\_0[rowSums(counts(data\_0)) >= reads.minimos,]  
  
# Cantidad de genes finales  
n.genes=nrow(data\_filtered)

## Exploración y análisis

### Comparación de transcriptomas entre muestras

A continuación, vamos a realizar una comparación general de los transcriptomas de todas las muestras. Recuerde que cada muestra cuenta, para cada gen, con un nivel de expresión determinado (el cual aproximamos con el número de lecturas de RNAseq que mapearon sobre ese gen). Por lo tanto, podemos considerar que cada muestra está en una determinada posición en cada uno de los 14554 ejes definidos por el conjunto de genes.



### Estabilización de la varianza (marco teórico **opcional**)

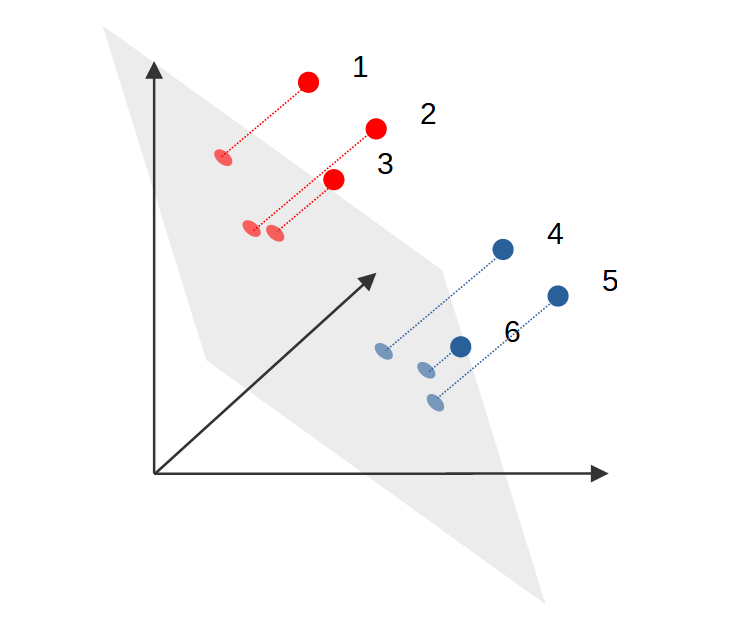
Más allá de las diferencias de expresión de cada gen *entre muestras*, es esperable que los niveles basales de expresión de cada gen sean diferentes entre sí. Es decir, que haya genes que tengan, en todas las muestras, mayores niveles de expresión que otros genes. Un problema de esto es que genes con mayor expresión tienen a su vez mayor *varianza* entre muestras. Imaginemos que tenemos un gen **A** con un nivel de expresión de 100 y un gen **B** con un nivel de 10.000. En este caso, variaciones en el nivel de expresión (entre muestras) del 10 % generarían diferencias de +/- 10 para el gen A y +/- 1000 para el gen B. Por lo tanto, a la hora de calcular *distancias* entre muestras (ver figura anterior), estas estarían gobernadas por los genes con mayores niveles de expresión, despreciando la información contenida en las diferencias de expresión de genes con niveles basales muy bajos. Para sortear este problema, y poder comparar muestras en función de todo su transcriptoma de forma representativa, se suelen realizar transformaciones de los datos que *estabilicen* la varianza. Esto significa que los valores de conteos ya no son los reales, pero conservan la información de las diferencias entre muestras como para hacer las comparaciones de transcriptomas completos. En este caso vamos a utilizar la transformación propuesta por Anders y Huber (2010, [*https://doi.org/10.1186/gb-2010-11-10-r106*](https://doi.org/10.1186/gb-2010-11-10-r106)) implementada en la función vst.

data\_var.st=vst(data\_filtered, blind = FALSE)

**Nota:** *como los conteos de cada gen han sido modificados, este dataset transformado no puede ser utilizado para comparar la expresión de cada gen particular entre muestras. Sólo lo vamos a utilizar para comparar entre transcriptomas completos.*

#### Análisis de Componentes Principales (PCA)

A continuación, vamos a visualizar la configuración de las muestras en el *espacio* de variables definido por los niveles de expresión de cada gen. Como este espacio tiene demasiadas dimensiones (14554) vamos a realizar una reducción dimensional mediante la proyección de la “sombra” de cada punto (correspondiente al transcriptoma de una muestra) sobre un plano de 2 dimensiones (Componentes Principales).



Primero usamos la función plotPCA para generar el nuevo plano 2D sobre el que proyectar los puntos, y guardamos los datos de las nuevas coordenadas en el objeto pcaData.

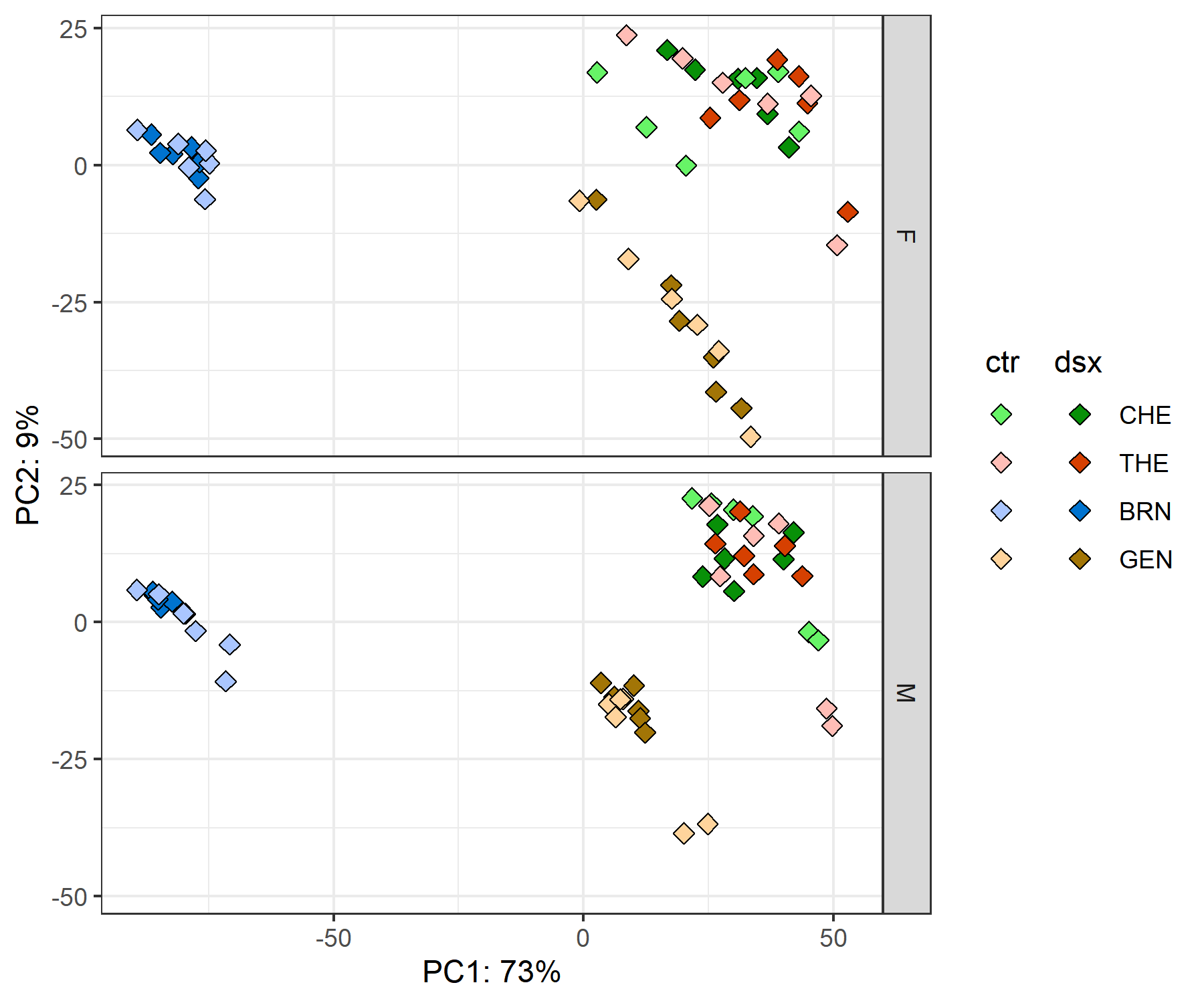
# cálculo de las nuevas dimensiones (proyección sobre dos componentes principales)  
pcaData <- plotPCA(data\_var.st,   
 intgroup = c("Tissue", "Sex", "Treatment"),   
 returnData = TRUE)  
percentVar <- round(100 \* attr(pcaData, "percentVar"))  
  
pcaData$tt=paste0(pcaData$Tissue, '-', pcaData$Treatment)

Ahora generamos un gráfico de puntos donde se vean las posiciones de cada muestra en este nuevo sistema de coordenadas. Primero hay que dar un poco de vueltas para configurar los colores de los puntos para cada grupo de muestras. Esto es medio enroscado así que puede saltar al siguiente bloque de código.

# Configuración de los colores de los puntos   
colores=c(BRN='dodgerblue',   
 CHE='limegreen',   
 GEN='darkgoldenrod2',   
 THE='coral1')[tejidos]  
  
colores.trat=colorspace::lighten(colores, 0.5)  
names(colores.trat)=paste0(tejidos, '-dsx')  
  
colores.ctrl=colorspace::darken(colores.trat, 0.4)  
names(colores.ctrl)=paste0(tejidos, '-ctr')  
  
colores2=c(colores.trat, colores.ctrl)

Una vez configurados los colores, podemos hacer el gráfico de PCA. Para esto vamos a utilizar el paquete ggplot2, que contiene una gran cantidad de funciones que nos permiten controlar de forma precisa los distintos aspectos de un gráfico. La comprensión de la sintaxis de ggplot va más allá del alcance de este TP pero, de forma **opcional**, puede tratar de comprender cómo se relacionan las diferentes partes en el siguiente código. Nótese que no hay una única función que contenga todo el código dentro de sus paréntesis, sino que los objetos y/o configuraciones se van agregando con diferentes funciones de forma secuencial mediante el operador +.

# Generación inicial del gráfico  
ggplot(pcaData, # datos de entrada  
   
 # aspectos del gráfico a ser mapeados a valores de variables presentes en los datos  
 aes(x = PC1, # posición en X  
 y = PC2, # posición en Y  
 fill = tt)) + # color de relleno  
   
 # Recién acá le indicamos que queremos dibujar puntos sobre el gráfico.  
 geom\_point(size =2.5, shape=23) +   
 # Sus coordenadas y colores van a ser definidos por el mapeo especificado en aes().  
 # Tamaño y forma son especificados fuera de aes() ya que, en este caso,   
 # no dependen de variables de los datos  
   
 # Especificación del conjunto de colores y su leyenda  
 scale\_fill\_manual(values = colores2,   
 name=' ctr dsx',   
 labels=c(rep('', length(tejidos)), tejidos))+  
   
 guides(fill=guide\_legend(ncol=2)) +  
   
 # titulos de ejes  
 xlab(paste0("PC1: ", percentVar[1], "%")) +  
 ylab(paste0("PC2: ", percentVar[2], "%")) +  
   
 facet\_grid(Sex ~ .)+ # ¿?  
   
 theme\_bw() # estética



**Opcional: ¿Qué cree que hace la función facet\_grid()? ¿Qué cree que pasaría si en vez de Sex ~ . el argumento fuera . ~ Sex?**

**¿Qué patrón de similitud/diferencia entre grupos de muestras observa?**

#### Heatmap

A continuación vamos a utilizar otra técnica de visualización para comparar muestras en cuanto a sus transcriptomas: los mapas de calor (o *Heatmaps*). Estos son básicamente representaciones gráficas de *matrices de distancias*. Una matriz de distancias es una tabla en la que cada elemento corresponde a la distancia entre la observación i y la observación j. En este caso, utilizamos la acepción más literal de distancia (distancia “en linea recta”), por lo que ; es decir, la matriz es simétrica sobre su diagonal. Para generar nuestro mapa de calor, primero vamos a calcular la matriz de distancias con la función dist.

# cálculo de distancias  
sampleDists <- dist(t(assay(data\_var.st)))  
  
# transformación del tipo de objeto R (class: dist -> matrix)  
sampleDistMatrix <- as.matrix( sampleDists )  
  
# homologar nombres de filas y columnas  
rownames(sampleDistMatrix) <- rownames(samples)[idx.muestras]  
colnames(sampleDistMatrix) <- rownames(samples)[idx.muestras]  
  
sampleDistMatrix[1:4, 1:4] # ver primeras 4 filas y 4 columnas

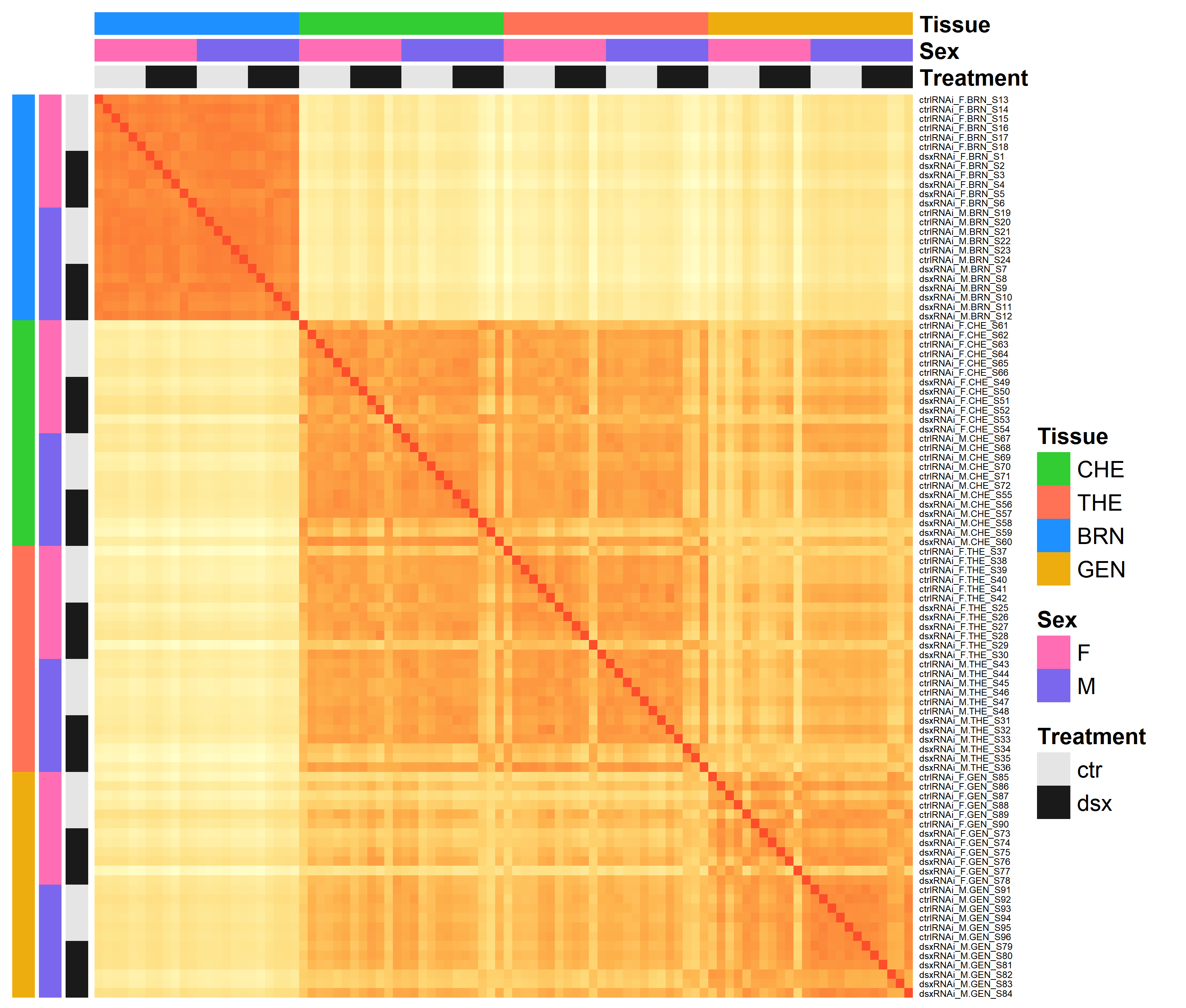
## ctrlRNAi\_F.BRN\_S13 ctrlRNAi\_F.BRN\_S14 ctrlRNAi\_F.BRN\_S15  
## ctrlRNAi\_F.BRN\_S13 0.00000 36.24175 38.32422  
## ctrlRNAi\_F.BRN\_S14 36.24175 0.00000 35.40467  
## ctrlRNAi\_F.BRN\_S15 38.32422 35.40467 0.00000  
## ctrlRNAi\_F.BRN\_S16 35.92918 31.42541 34.32071  
## ctrlRNAi\_F.BRN\_S16  
## ctrlRNAi\_F.BRN\_S13 35.92918  
## ctrlRNAi\_F.BRN\_S14 31.42541  
## ctrlRNAi\_F.BRN\_S15 34.32071  
## ctrlRNAi\_F.BRN\_S16 0.00000

**¿Qué dimensiones espera que tenga esta matriz de distancias?**

**¿Cómo se explican los valores de la diagonal?**

Ahora, con esta matriz de distancias vamos a generar un mapa de calor con la función pheatmap para poder interpretarla más facilmente de forma visual.

# çonfiguración de la paleta de colores  
heatmap.1.colors <- colorRampPalette( brewer.pal(9, "YlOrRd")[6:1] )(255)  
  
pheatmap(sampleDistMatrix,  
 col = heatmap.1.colors, border\_color = NA,   
 fontsize\_row = 4.5, fontsize = 11, # tamaño de las letras  
 cluster\_cols = F, cluster\_rows = F, # mantener el orden de filas y columnas  
 annotation\_col = samples[idx.muestras,1:3], # información para el código de anotación por colores de columnas  
 annotation\_row = samples[idx.muestras,1:3], # información para el código de anotación por colores de filas   
 show\_colnames=F, show\_rownames=T, # nombres de muestras en filas  
 annotation\_names\_row=F, legend = F, # mostrar qué significa cada color  
   
 # configuración de los colores de cada grupo de muestras  
 annotation\_colors = list(Sex=c(M='slateblue2',   
 F='hotpink1')[sex],   
   
 Treatment=c(ctr='gray90',   
 dsx='gray10')[trat],   
   
 Tissue=c(BRN='dodgerblue',   
 CHE='limegreen',   
 GEN='darkgoldenrod2',   
 THE='coral1')[tejidos]  
 )  
 )



**¿Son coherentes los patrones de similitud y diferencia entre las muestras observados en el PCA y el Heatmap?**

**¿Qué patrón prominente observa? ¿Hay algún factor (tejido, sexo, tratamiento) que determine más el perfil transcriptómico de una muestra que los otros factores?**

### Análisis de expresión génica diferencial

#### Análisis estadístico utilizando DESeq

A continuación vamos a realizar un análisis estadístico de expresión diferencial. Este evalúa el grado de diferencia en el nivel de expresión de cada gen entre los diferentes niveles de cada factor especificado (tejido, sexo, tratamiento, etc.). Para este análisis, vamos a trabajar sólo con las muestras del tejido cefálico sin tratamiento con RNA de interferencia de *dsx*. El filtrado lo hacemos utilizando un vector lógico (idx.muestras) igual que antes.

# subconjunto de muestras  
tejidos=c('CHE')  
sex=c('F', 'M')  
trat=c('ctr')  
  
# índice T/F de muestras que pertenecen al subconjunto especificado  
idx.muestras= samples$Tissue %in% tejidos &  
 samples$Sex %in% sex &  
 samples$Treatment %in% trat  
  
# objeto DESeqDataSet (conteos + información de muestras)  
data.DE\_0 <- DESeqDataSetFromMatrix(countData = cts[,idx.muestras],  
 colData = samples[idx.muestras,   
 c("Sex", "Tissue", "Treatment")],  
 design = ~ Sex)  
  
  
reads.minimos=10  
  
data\_DE <- data.DE\_0[rowSums(counts(data.DE\_0)) >= reads.minimos,]  
  
# Cantidad de genes incluidos en el análisis  
nrow(data\_DE)

## [1] 11867

Con estos datos filtrados utilizamos la función DESeq para estimar el nivel de expresión de cada gen a lo largo de las diferentes condiciones (puede tardar varios segundos en correr).

de.an=DESeq(data\_DE)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

Con el resultado de esta estimación, podemos generar una tabla que muestre el grado de expresción diferencial para cada gen entre diferentes condiciones. Por defecto, la función results utiliza la fómula definida al crear el objeto DESeqDataSet para hacer las comparaciones, pero nosotros podemos especificar la comparación que deseemos mediante el argumento contrast.

res = results(de.an, contrast = c('Sex', 'M', 'F')) %>%   
 as.data.frame() %>%   
 select(-c(4:5)) # borrar algunas columans que no son de interés  
  
head(res, 5)

## baseMean log2FoldChange lfcSE padj  
## OTAU000001 98.84266 -0.24766932 0.10910713 0.13351501  
## OTAU000002 734.63890 0.08395974 0.07340131 0.54500459  
## OTAU000003 1068.10593 0.25464177 0.10794724 0.11439624  
## OTAU000005 15.21843 -0.06968228 0.28740102 0.92689274  
## OTAU000006 10.77057 1.35944475 0.53609925 0.08219862

Para cuantificar la magnitud del cambio en el nivel de expresión entre condiciones (Macho y Hembra en este caso, como se especifica en contrast) se suele utilizar el estadístico del *Log2FoldChange* (LFC). El valor del LFC representa cuántas veces se duplicó (o se redujo a la mitad) el conteo de reads mapeados a cada gen (nuestro estimador del nivel de expresión). Éste se calcula como

Donde es el nivel de expresión en Machos y es el nivel de expresión en Hembras. De esta fórmula puede deducirse que, en este caso, el LFC toma valores positivos cuando y negativos cuando . La relación está especificado en este sentido por el orden de M y F en el argumento contrast utilizado al generar la tabla de resutlados; si se invirtiera este orden entonces se invertiría el cociente de la ecuación anterior y, por lo tanto, el signo de los LFC. Además del signo, el valor específico del LFC nos dá información sobre la cantidad de cambio, correspondiendo cada unidad a un factor 2. Es decir, un significa que la expresión es el *doble* en machos que en hembras; un indica que la expresión es *8 veces mayor* en hembras que en machos.

Para identificar los genes con mayor diferencias en su expresión vamos a reordenar por |LFC| y filtrar por significancia estadística de estas diferencias.

res2=res[order(abs(res$log2FoldChange), decreasing = T),] %>% # re-ordenamiento  
 as.data.frame() %>% # llevar a formato (class) data.frame  
 filter(padj<=0.05) %>% # filtrar por significancia (p-valor < 0.05)  
 head()  
  
res2

## baseMean log2FoldChange lfcSE padj  
## OTAU000477 55.357960 8.621074 2.6225529 1.515777e-02  
## OTAU003371 538.693051 -8.051160 0.7289697 8.769848e-25  
## OTAU012035 119.552901 7.944221 1.6402324 7.036039e-05  
## OTAU012036 31.398590 7.804147 2.7256404 4.272054e-02  
## OTAU015750 79.819884 6.477372 0.8566532 2.050096e-11  
## OTAU014526 7.733296 6.268248 2.1828129 4.190321e-02

#### Comparación gráfica utilizando ggplot

También podemos hacer una comparación gráfica de los niveles de expresión de cada gen entre condiciones. Para esto, lo más correcto sería corregir/normalizar los valores de conteos por *tamaño de la librería*. Esto se debe a que durante la extrracción y preparación del RNA para la secuenciación, hay errores de procedimiento que generan variaciones en la cantidad de RNA entre muestras que nada tienen que ver con diferencias en el nivel de expresión. Por lo tanto, conviene relativizar el conteo de cada gen de una muestra respecto de la cantidad total de reads que se obtuvieron de la misma. Las siguientes lineas hacen esa normalización.

cts.trans=cts %>%   
 as.matrix() %>%   
 t()   
  
cts.trans.norm=cts.trans/(samples$lib.size/1e7)  
  
cts.norm=t(cts.trans.norm) %>%   
 as.data.frame()

Ahora, con esta tabla de conteos normalizados podemos graficar los niveles de expresion de los genes con mayores diferencias entre sexos. Para ello primero reorganizamos un poco las tablas.

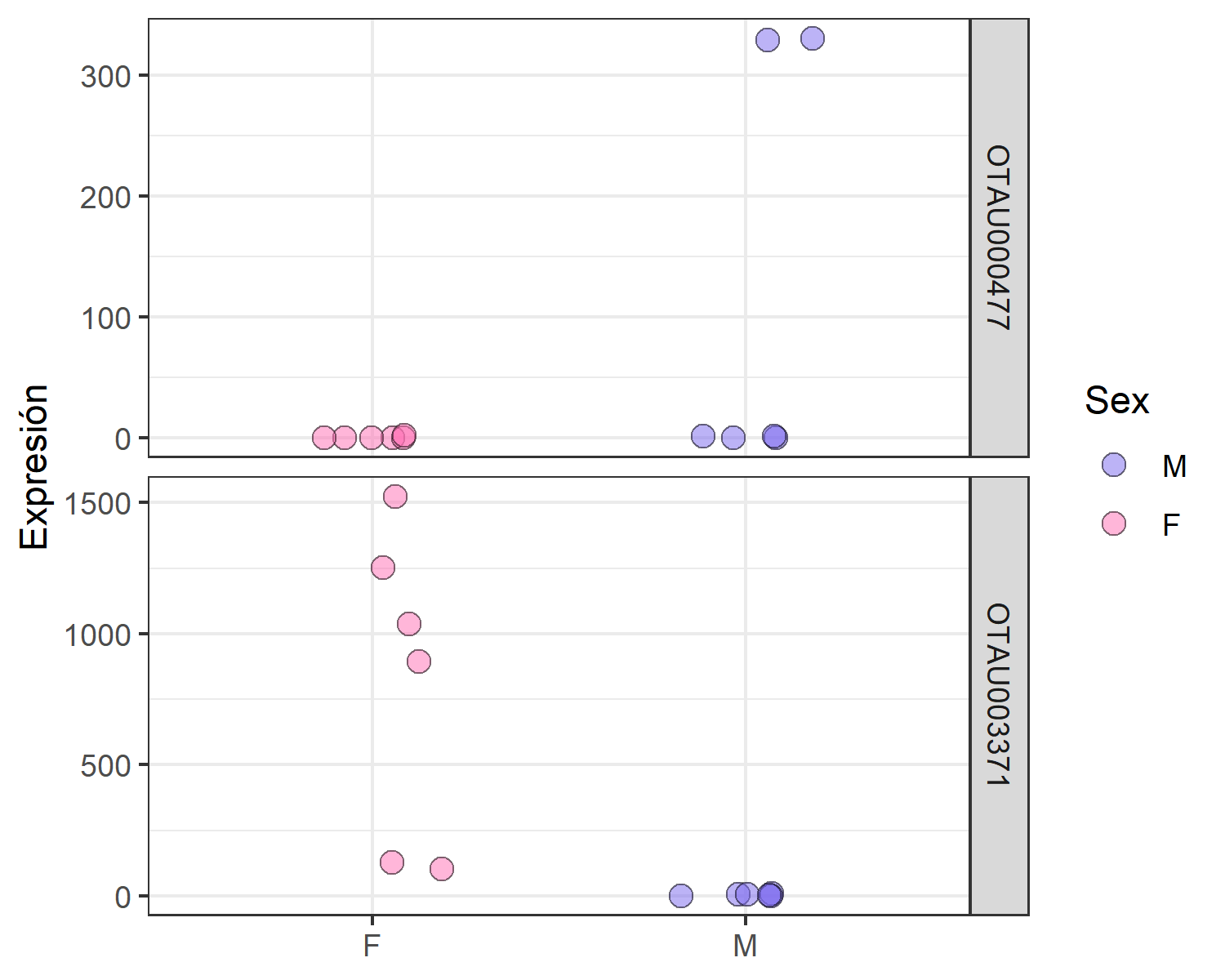
topgenes.n=2 # cantidad de genes a considerar  
  
genes=rownames(res2)[1:topgenes.n] # nombres de estos genes  
  
genes.exp=cts.norm[genes,] %>% # extracción de estos genes de la tabla de conteos  
 t() # verticalización para poder juntar con la tabla del diseño experimental  
  
exp.table=cbind(samples, genes.exp) # unión con la tabla de diseño  
  
exp.table.long=pivot\_longer(exp.table, cols = -c(1:5),   
 names\_to = 'gene', values\_to = 'exp') # verticalización

Y filtramos sólo aquellos datos que nos interesa graficar. Por ahora sólo queresmo comparar la expresión de los dos genes de interés entre los tejidos cefálicos de machos y hembras control.

data.plot= as.data.frame(exp.table.long) %>% # cambiamo a CLASS data.frame para trabajar con filter()  
 filter(Treatment %in% trat) %>% # filtramos por tratamiento (sólo ctr)  
 filter(Tissue %in% tejidos) # filtramos por tejido (sólo CHR))

Ahora le pasamos esta tabla a ggplot, una función del paquete ggplot2 que permite generar gráficos controlando de forma precisa, y relativamente simple, muchos parámetros gráficos. Notesé que la generación del gráfico comienza con el llamado de la función ggplot(data.plot), seguida de múltiples funciones que se van sumando con + para agregar elementos o configuraciones al gráfico.

ggplot(data= data.plot) + # datos en los que ggplot buscará lo que le ordenemos en la progrmación del gráfico  
   
 # generamos un elemento (geom) de puntos  
 geom\_jitter(aes(x=Sex, y=exp, # la posición de cada punto codificará el nivel de expresión en función del sexo  
 fill=Sex), # el relleno de los puntos también codificará el sexo  
 width = 0.2, # ancho de la dispersión horizontal aleatoria (para que no se solapen tanto)  
 pch=21, # estilo de punto  
 size=3, # tamaño de los puntos  
 alpha=0.5)+ # transparencia (0 - 1)  
   
 scale\_fill\_manual(values = c(M='slateblue2', F='hotpink1'))+ # colores de cada sexo  
   
 facet\_grid(gene~., # desdoblamos el gráfico verticalmente, uno para cada gen  
 scales = 'free')+ # cada panel puede tener su propia escala  
   
 labs(x=NULL, y='Expresión')+ # títulos de ejes  
   
 theme\_bw() # estética general del gráfico



**¿Es este gráfico coherente con los sentidos de cambio de expresión indicados por el signo de los valores de LFC? Justifique**

#### Investigación sobre genes diferencialmente expressados

Busque los dos genes con mayor diferencias en su expresión (entre las regiones de los cuernos cefálicos de machos y hembras) en el *Apollo Browser* (<https://i5k.nal.usda.gov/node/739225>). Para esto copie los códigos (OTAU0...) de los primeros dos nombres de fila de la tabla res2 (recordemos que esta contiene lo mismo que res pero está ordenada por LFC y filtrada por significancia estadística). Para buscar el modelo correspondiente a cada gen, pegue el código en el buscador del browser Apollo y complete con -RA al final. Esto se debe a que la base de datos de Apollo almacena los modelos de genes de *O. taurus* con el sufijo -RA. En el panel de la izquierda, active los tracks O. taurus M PD1 BRN+CHE+THE+GEN transcripts y O. taurus F PD1 BRN+CHE+THE+GEN transcripts para ver los perfiles de *coverage* **¿Son las diferencias de covergae coherentes con los valores de LFC y el gráfico de expresión? Justifique**

Cuando identifique el modelo correspondiente, selecciónelo clickeando en un intron. Luego *click derecho* -> *View details*. Scrollee hasta la primer secuencia, selecciónela con soble click y cópiela en un archivo de texto. Esta corresopnde a la secuencia **genómica** del modelo, es decir, contiene los intrones.