# Importación de *paquetes*

Si la instalación de todo funcionó bien (no saltaron Errores), 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  
 #"glmpca",  
 "ggbeeswarm",   
 'gridExtra',   
 'colorspace')  
  
# 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")  
  
# dimensiones de la tabla  
dim(cts) %>% `names<-`(c('genes', 'muestras'))

## genes muestras   
## 17483 96

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

# filas de 1 a 10, columnas de 1 a 3  
cts[1:10, 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  
## OTAU000007 0 0 0  
## OTAU000008 1841 1902 1820  
## OTAU000009 730 549 617  
## OTAU000010 490 535 568

Vemos que cada **fila** tiene el nombre asociado al gen correspondiente. OTAU000001 no es el *nombre* de ese gen propiamente dicho, sino que es el identificador que se le asignó a esa región codificante en el genoma que estamos utilizando. Si queremos saber de qué gen se trata, podemos utilizar el explorador de genomas **(ver link a i5k)** para buscar esa región codificante, copiar la secuencia de la misma y buscar secuencias parecidas en una base de datos de genes (por ej. en NCBI). Esta búsqueda se realiza normalmente mediante el uso del algoritmo BLAST (*Basic Local Alignment Search Tool*). 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) **¿Se aplicó RNAi ctrl? ¿Con qué secuencia?**.

## 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), esto 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 bien ordenada.

#samples0 <- read.delim("./Muestras.tsv", sep="\t", row.names=1)  
#samples <- samples0[,c(5, 6, 4, 2, 13)]  
#samples=dplyr::rename(samples, lib.size=genome\_unique)  
  
samples <- read.csv("./Muestras\_reord.tsv", row.names=1)

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

# filas de 1 a 8  
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

# (por ahora ignoremos la última columna de lib.size)

### Ordenamiento y selección de muestras

# idx.order=order(samples$Tissue, samples$Sex, samples$Treatment)  
# samples=samples[idx.order,]  
# samples=rbind(filter(samples, Tissue!='GEN'),   
# filter(samples, Tissue=='GEN'))  
  
#   
  
tejidos=c('CHE', 'THE', 'BRN', 'GEN')  
sex=c('F', 'M')  
trat=c('ctr', 'dsx')  
  
idx.muestras= samples$Tissue %in% tejidos &  
 samples$Sex %in% sex &  
 samples$Treatment %in% trat

## 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, tenemos que chequear que cada columna de la tabla de conteos sea asociable inequívocamente a una fila de 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)

## Filtrado del dataset

Podemos descartar genes que tengan menos fragmentos secuenciados (reads) que un determinado umbral. Probar con diferentes valores.

reads.minimos=100  
  
# bajo el umbral (muy pocos reads)  
c(rowSums(counts(data\_0)) < reads.minimos ) %>% sum()

## [1] 5011

# 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  
nrow(data\_filtered)

## [1] 14554

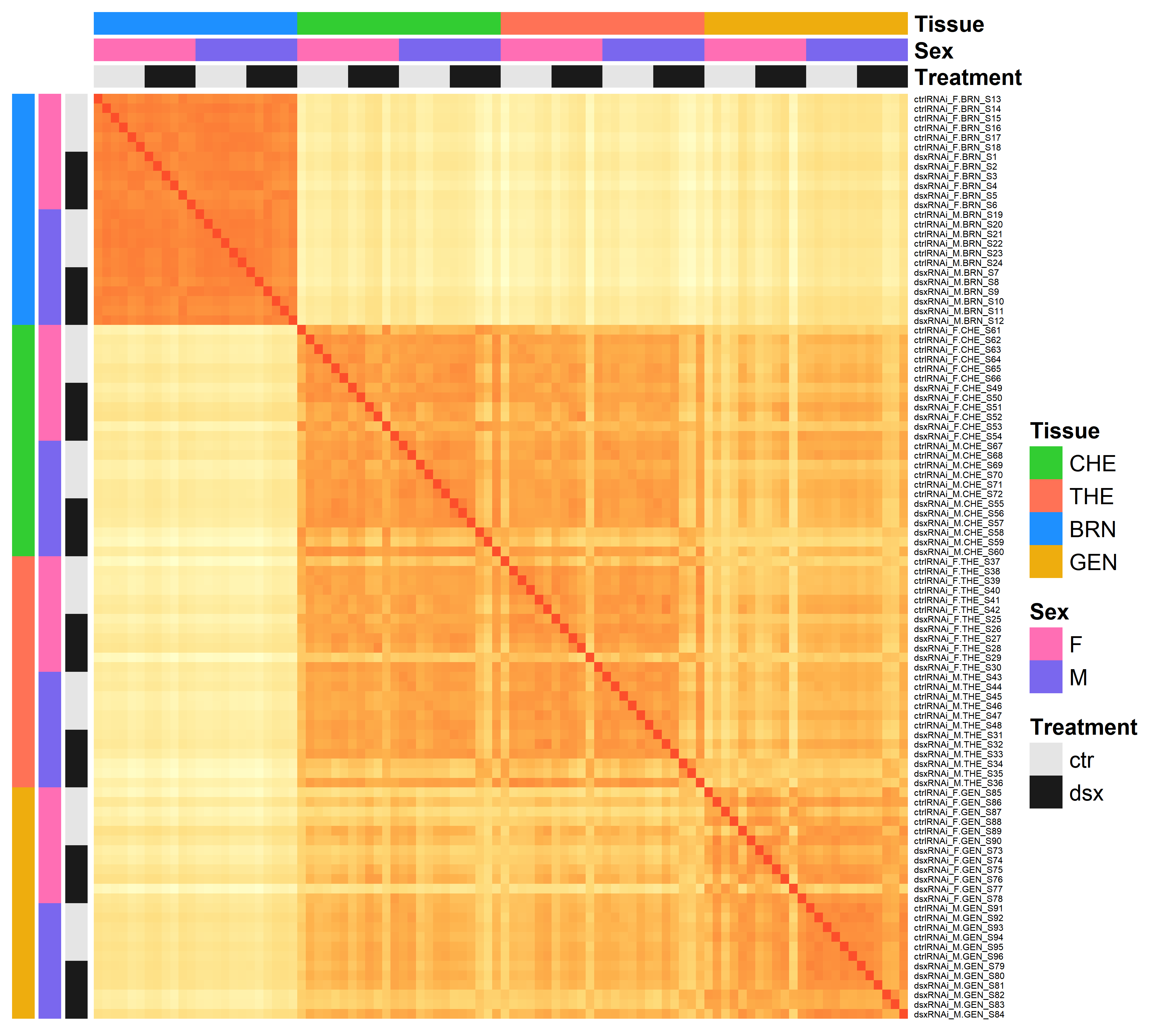
# Exploración y análisis

## Comparación entre conjuntos de muestras

### Heatmap

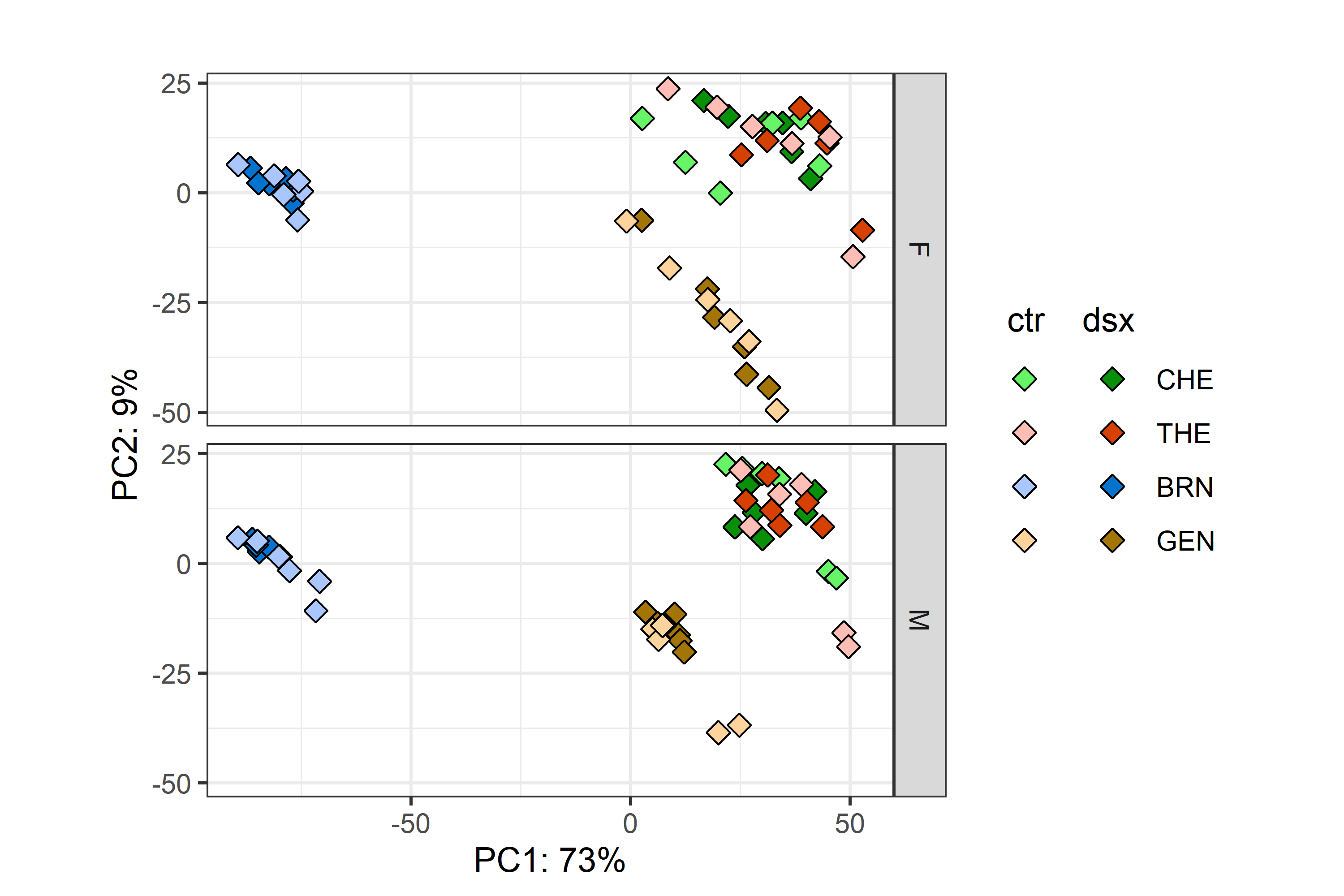
#### Estabilización de la varianza

data\_var.st=vst(data\_filtered, blind = FALSE)  
  
heatmap.1.colors <- colorRampPalette( brewer.pal(9, "YlOrRd")[6:1] )(255)  
  
sampleDists <- dist(t(assay(data\_var.st)))  
  
sampleDistMatrix <- as.matrix( sampleDists )  
  
rownames(sampleDistMatrix) <- rownames(samples)[idx.muestras]  
colnames(sampleDistMatrix) <- rownames(samples)[idx.muestras]  
  
pheatmap(sampleDistMatrix,  
 col = heatmap.1.colors, border\_color = NA,   
 fontsize\_row = 4.5, fontsize = 11,  
 cluster\_cols = F, cluster\_rows = F,   
 annotation\_col = samples[idx.muestras,1:3],   
 annotation\_row = samples[idx.muestras,1:3],   
 show\_colnames=F, show\_rownames=T,   
 annotation\_names\_row=F, legend = F,   
 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]  
 ))



### PCA

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)  
  
  
# colores para los puntos (es bastante vueltero esto así que ignorar)  
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)  
  
# gráfico de PCA  
ggplot(pcaData,   
 aes(x = PC1, y = PC2, fill = tt)) +  
 geom\_point(size =2.5, shape=23) +  
 scale\_fill\_manual(values = colores2,   
 name=' ctr dsx',   
 labels=c(rep('', length(tejidos)), tejidos))+  
 xlab(paste0("PC1: ", percentVar[1], "%")) +  
 ylab(paste0("PC2: ", percentVar[2], "%")) +  
 coord\_fixed() +  
 ggtitle("")+  
 facet\_grid(Sex~.)+  
 theme\_bw()+  
 guides(fill=guide\_legend(ncol=2))



# Análisis de expresión diferencial

## Estabilización de la varianza

El tipo de herramientas estadísticas que vamos a utilizar para buscar patrones de expresión génica diferencial requiere que los datos cumplan ciertas propiedades. En particular,