

Análisis de expresión diferencial y visualización de resultados con Bioconductor

Sesiones 3.1 y 3.2

Alejandro Reyes y Leonardo Collado-Torres

Octubre 26, 2016

Paquetes opcionales

- EnsDb.Hsapiens.v75 son como 50 mb

```
source("https://bioconductor.org/biocLite.R")
biocLite(c("vsn", "EnsDb.Hsapiens.v75"))
```

Análisis de expresión diferencial con *DESeq2*

- El objeto *DESeqDataSet*.
- Normalización de datos (parámetro s_j).
- Estimación de las dispersiones (parámetro α_i).
- Comparaciones sencillas (tratamiento vs control).
- Comparaciones con factores de bloqueo.
- Comparaciones con interacciones.
- Filtrado independiente.
- Transformaciones a los datos de conteo.

Análisis de expresión diferencial con *DESeq2*

- El objeto *DESeqDataSet*.
- Normalización de datos (parámetro s_j).
- Estimación de las dispersiones (parámetro α_i).
- Comparaciones sencillas (tratamiento vs control).
- Comparaciones con factores de bloqueo.
- Comparaciones con interacciones.
- Transformaciones a los datos de conteo.

Creación de un objeto *DESeqDataSet* I

Primero, vamos a especificar el directorio en el que se encuentran los archivos de conteo

```
path <- "."
countMatrixFile <-
  file.path( path, "genomeeting_counts.tsv" )
countMatrixFile
## [1] "./genomeeting_counts.tsv"
```

```

sampleAnnotationFile <-
  file.path( path, "genomeeting_sample_summary.tsv" )

sampleAnnotationFile

## [1] "./genomeeting_sample_summary.tsv"

```

Creación de un objeto *DESeqDataSet* II

Una vez especificados la dirección a los archivos, importamos los datos dentro de *R*. Comenzamos leyendo la matrix de cuentas por cada muestra.

```

countMatrix <- read.delim( countMatrixFile, row.names=1 )
head( countMatrix )

##          SRR1039508 SRR1039509 SRR1039512
## ENSG000000000003     679      448      873
## ENSG000000000005      0        0        0
## ENSG000000000419    467      515      621
## ENSG000000000457    260      211      263
## ENSG000000000460     60       55       40
## ENSG000000000938     0        0        2
##          SRR1039513 SRR1039516 SRR1039517
## ENSG000000000003    408      1138     1047
## ENSG000000000005     0        0        0
## ENSG000000000419    365      587      799
## ENSG000000000457    164      245      331
## ENSG000000000460     35       78       63
## ENSG000000000938     0        1        0
##          SRR1039520 SRR1039521
## ENSG000000000003    770      572
## ENSG000000000005     0        0
## ENSG000000000419    417      508
## ENSG000000000457    233      229
## ENSG000000000460     76       60
## ENSG000000000938     0        0

```

Creación de un objeto *DESeqDataSet* III

Después importamos el archivo con la anotación de las muestras:

```

annotation <- read.delim( sampleAnnotationFile )
head( annotation )

##           Run SampleName   cell   dex albut
## 1 SRR1039508 GSM1275862 N61311 untrt untrt
## 2 SRR1039509 GSM1275863 N61311    trt untrt
## 3 SRR1039512 GSM1275866 N052611 untrt untrt
## 4 SRR1039513 GSM1275867 N052611    trt untrt
## 5 SRR1039516 GSM1275870 N080611 untrt untrt
## 6 SRR1039517 GSM1275871 N080611    trt untrt
##   avgLength Experiment   Sample   BioSample
## 1         126   SRX384345  SRS508568 SAMN02422669
## 2         126   SRX384346  SRS508567 SAMN02422675

```

```

## 3      126  SRX384349 SRS508571 SAMN02422678
## 4      87   SRX384350 SRS508572 SAMN02422670
## 5     120  SRX384353 SRS508575 SAMN02422682
## 6     126  SRX384354 SRS508576 SAMN02422673

```

Creación de un objeto *DESeqDataSet* IV

Es importante verificar que el orden de las columnas de la matriz de cuentas esté en el mismo orden que las filas el *data.frame* de anotación:

```

all( colnames( countMatrix ) == annotation$Run )

## [1] TRUE

```

Creación de un objeto *DESeqDataSet* V

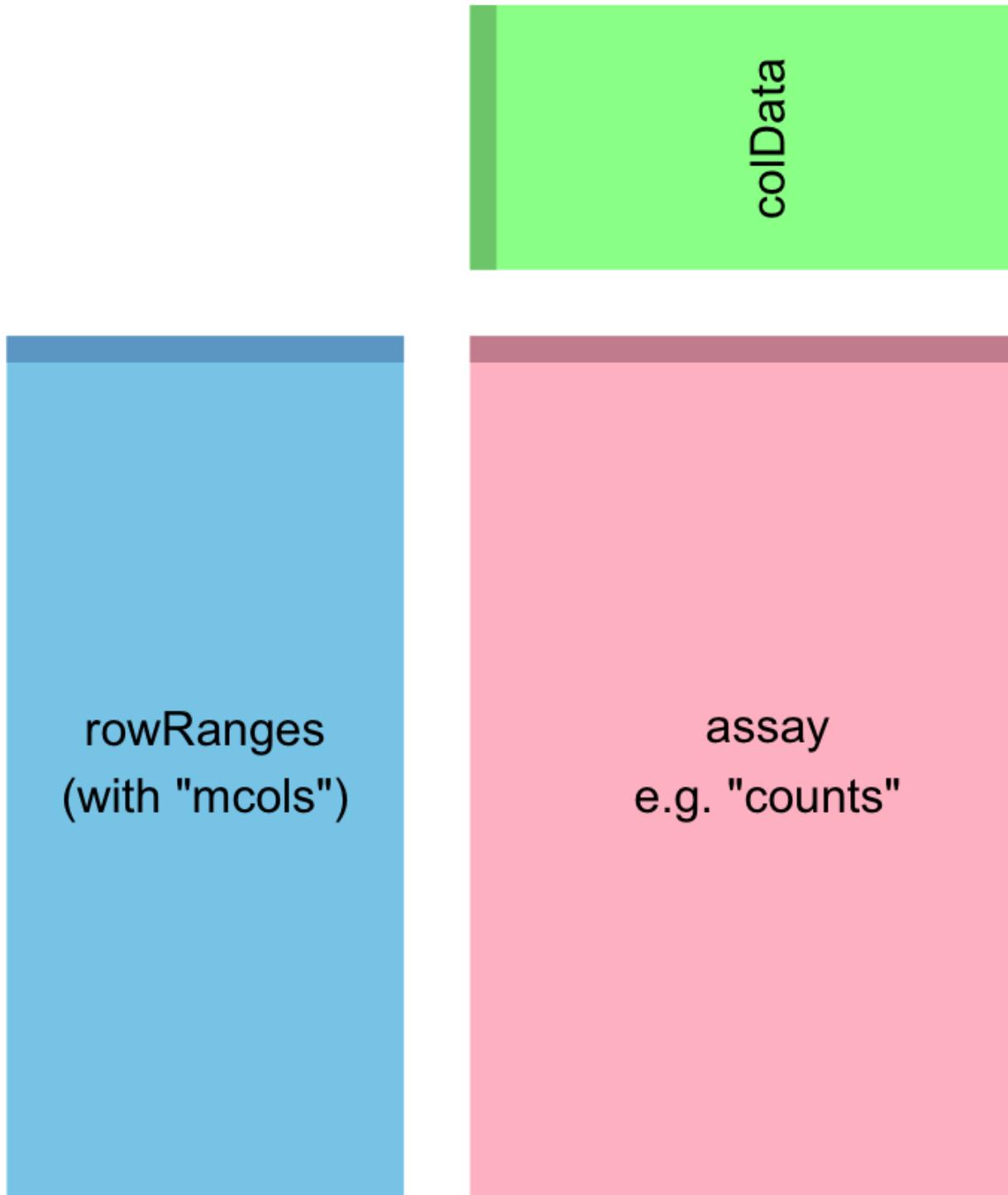
Ya preparados nuestros datos, llamamos a la función que genera el *DESeqDataSet* a partir de nuestra matriz de cuentas y el *data.frame* con la anotación de cada muestra.

```

suppressMessages( library('DESeq2') )
dsd <- DESeqDataSetFromMatrix(
  countData = countMatrix, colData = annotation,
  design = ~ dex,
)
dsd

## class: DESeqDataSet
## dim: 64102 8
## metadata(1): version
## assays(1): counts
## rownames(64102): ENSG00000000003
##   ENSG00000000005 ... LRG_98 LRG_99
## rowData names(0):
## colnames(8): SRR1039508 SRR1039509 ...
##   SRR1039520 SRR1039521
## colData names(9): Run SampleName ... Sample
##   BioSample

```



Exploración del *DESeqDataSet* I

```
head( counts( dsd ) )

##          SRR1039508 SRR1039509 SRR1039512
## ENSG00000000003     679      448      873
## ENSG00000000005      0       0       0
## ENSG00000000419    467      515      621
## ENSG00000000457    260      211      263
## ENSG00000000460     60       55       40
## ENSG00000000938     0       0       2
##          SRR1039513 SRR1039516 SRR1039517
## ENSG00000000003    408      1138     1047
## ENSG00000000005     0       0       0
## ENSG00000000419    365      587      799
## ENSG00000000457    164      245      331
## ENSG00000000460     35       78       63
## ENSG00000000938     0       1       0
##          SRR1039520 SRR1039521
## ENSG00000000003    770      572
## ENSG00000000005     0       0
## ENSG00000000419    417      508
## ENSG00000000457    233      229
## ENSG00000000460     76       60
## ENSG00000000938     0       0
```

Exploración del *DESeqDataSet* II

```
colData( dsd )[c("Run", "Sample", "dex")]

## DataFrame with 8 rows and 3 columns
##           Run   Sample   dex
##           <factor> <factor> <factor>
## SRR1039508 SRR1039508 SRS508568   untrt
## SRR1039509 SRR1039509 SRS508567     trt
## SRR1039512 SRR1039512 SRS508571   untrt
## SRR1039513 SRR1039513 SRS508572     trt
## SRR1039516 SRR1039516 SRS508575   untrt
## SRR1039517 SRR1039517 SRS508576     trt
## SRR1039520 SRR1039520 SRS508579   untrt
## SRR1039521 SRR1039521 SRS508580     trt
```

Exploración del *DESeqDataSet* III

```
rowRanges( dsd )

## GRangesList object of length 64102:
## $ENSG00000000003
## GRanges object with 0 ranges and 0 metadata columns:
##   seqnames    ranges strand
##   <Rle> <IRanges> <Rle>
```

```

## 
## $ENSG00000000005
## GRanges object with 0 ranges and 0 metadata columns:
##   seqnames ranges strand
## 
## $ENSG00000000419
## GRanges object with 0 ranges and 0 metadata columns:
##   seqnames ranges strand
## 
## ...
## <64099 more elements>
## -----
## seqinfo: no sequences

```

Exploración del *DESeqDataSet* IV

```

mcols( dsd )
## DataFrame with 64102 rows and 0 columns

```

Generando sub sets del *DESeqDataSet*

```

dsdSubset <- dsd[,1:4]
dim( counts( dsdSubset ) )

## [1] 64102      4
dim( colData( dsdSubset ) )

## [1] 4 9

```

Ejercicio 1

- ¿Cuántos fragmentos fueron secuenciados del gen ENSG00000003137 en la muestra SRR1039521?
- ¿Cuál es el identificador del experimento de la muestra SRR1039512?

Solución 1

```

counts(dsd)[ 'ENSG00000003137' , 'SRR1039521' ]

## [1] 68
colData(dsd)$Experiment[ colData(dsd)$Run == 'SRR1039512' ]

## [1] SRX384349
## 8 Levels: SRX384345 SRX384346 ... SRX384358

```

Análisis de expresión diferencial con *DESeq2*

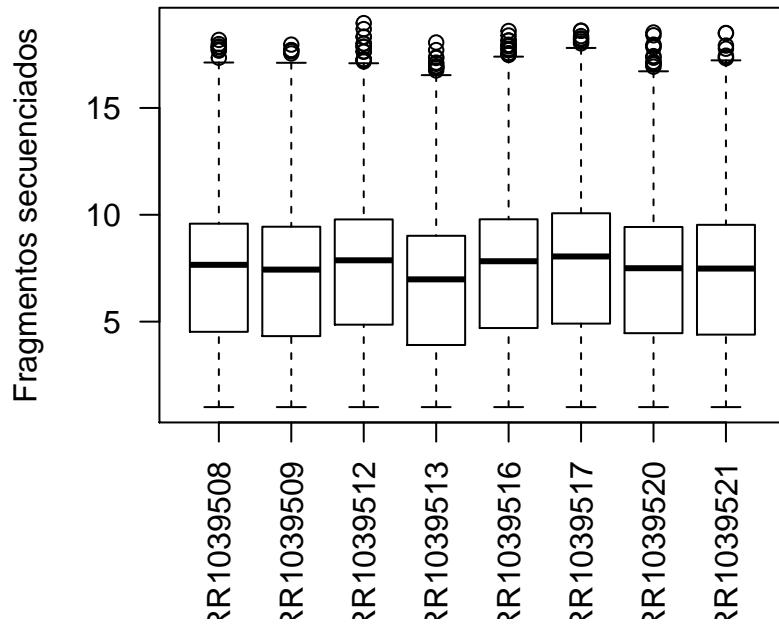
- El objeto *DESeqDataSet*.
- Normalización de datos (parámetro s_j).

- Estimación de las dispersiones (parámetro α_i).
- Comparaciones sencillas (tratamiento vs control).
- Comparaciones con factores de bloqueo.
- Comparaciones con interacciones.
- Transformaciones a los datos de conteo.

Normalización de datos I

¿Porqué debemos normalizar los datos?

```
hasCounts <- rowSums( counts(dsd) > 0 ) == ncol(dsd)
boxplot( log2( counts( dsd )[hasCounts,] + 1 ), las=2,
         ylab="Fragmentos secuenciados", srt = 45)
```



Normalización de datos II

$$\hat{s}_j = \text{median}_i \frac{k_{ij}}{\left(\prod_{v=1}^m k_{iv}\right)^{\frac{1}{m}}}$$

Normalización de datos III

Llamamos a la función `estimateSizeFactors`:

```
dsd <- estimateSizeFactors( dsd )

sizeFactors( dsd )

## SRR1039508 SRR1039509 SRR1039512 SRR1039513
## 1.0236476 0.8961667 1.1794861 0.6700538
## SRR1039516 SRR1039517 SRR1039520 SRR1039521
## 1.1776714 1.3990365 0.9207787 0.9445141

colData( dsd )[ , "sizeFactor"]
```

```

## SRR1039508 SRR1039509 SRR1039512 SRR1039513
## 1.0236476 0.8961667 1.1794861 0.6700538
## SRR1039516 SRR1039517 SRR1039520 SRR1039521
## 1.1776714 1.3990365 0.9207787 0.9445141

```

Ejercicio II

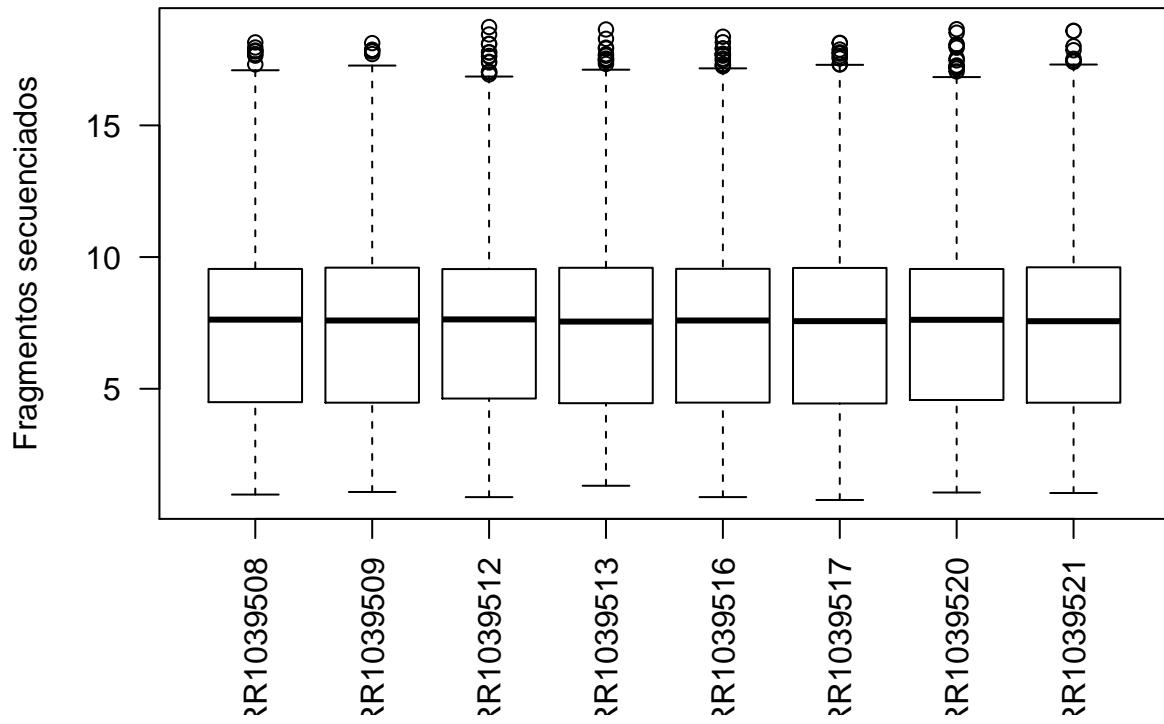
Repetir el boxplot anterior, pero ahora con las cuentas normalizadas. Pista: el código es casi idéntico, cambia sólo un parametro de la function *counts*.

Solución II

```

boxplot( log2(
  counts( dsd, normalized = TRUE )[hasCounts,] + 1 ) , las=2,
  ylab="Fragmentos secuenciados", srt = 45)

```



Análisis de expresión diferencial con *DESeq2*

- El objeto *DESeqDataSet*
- Normalización de datos (parámetro s_j).
- **Estimación de las dispersiones (parámetro α_i)**.
- Comparaciones sencillas (tratamiento vs control).
- Comparaciones con factores de bloqueo.
- Comparaciones con interacciones.
- Transformaciones a los datos de conteo.

Estimación de las dispersiones I

¿Porqué no podemos estimar la varianza con métodos comunes de máxima verosimilitud?

Estimación de las dispersiones II

Tres pasos importantes:

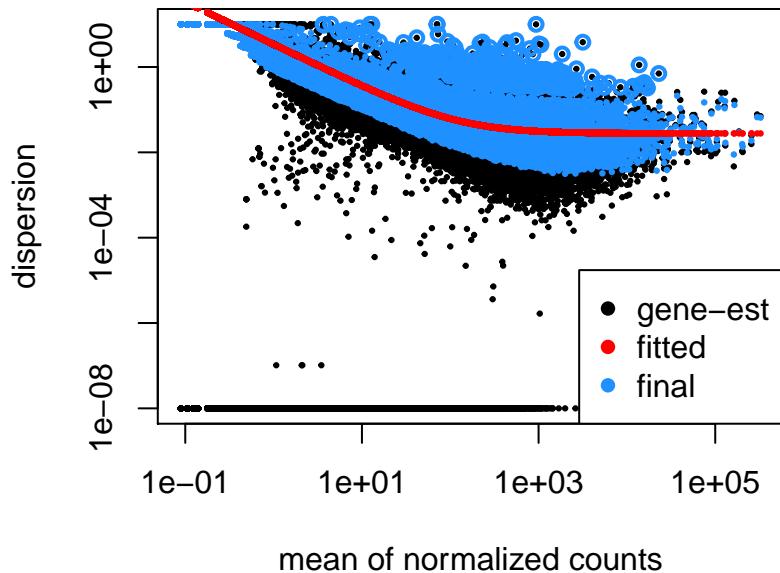
1. Estimar la dispersión por cada gen.
2. Modelar la relación entre media y varianza.
 - Asumiendo que: genes con medias similares tendrán varianzas similares.
 - Compartiendo información entre genes.
3. Calcular un compromiso entre dispersiones individuales y la dispersión compartida.

Estimación de las dispersiones III

```
dsd <- estimateDispersions( dsd )  
  
## gene-wise dispersion estimates  
## mean-dispersion relationship  
## final dispersion estimates
```

Estimación de las dispersiones IV

```
plotDispEsts( dsd )
```



Ejercicio III

¿En qué campo del *DESeqDataSet* se guarda la información de las dispersiones?

Solución III

```
rowData(dsd)[,
  grep('disp', colnames(rowData(dsd)))
]

## DataFrame with 64102 rows and 6 columns
##      dispGeneEst    dispFit dispersion
##      <numeric>    <numeric>   <numeric>
## 1    0.025075298 0.03251273 0.027334343
## 2        NA        NA        NA
## 3    0.001329435 0.03423796 0.007780134
## 4    0.000000010 0.04198854 0.011068811
## 5    0.061685923 0.08605230 0.069624032
## ...
## 64098       NA        NA        NA
## 64099       NA        NA        NA
## 64100       NA        NA        NA
## 64101       NA        NA        NA
## 64102       NA        NA        NA
##      dispIter dispOutlier    dispMAP
##      <numeric>  <logical>   <numeric>
## 1          9     FALSE 0.027334343
## 2         NA        NA        NA
## 3          7     FALSE 0.007780134
## 4          8     FALSE 0.011068811
## 5         10     FALSE 0.069624032
## ...
## 64098       NA        NA        NA
## 64099       NA        NA        NA
## 64100       NA        NA        NA
## 64101       NA        NA        NA
## 64102       NA        NA        NA
```

Análisis de expresión diferencial con *DESeq2*

- El objeto *DESeqDataSet*
- Normalización de datos (parámetro s_j).
- Estimación de las dispersiones (parámetro α_i).
- **Comparaciones sencillas (tratamiento vs control)**
 - Comparaciones con factores de bloqueo.
 - Comparaciones con interacciones.
 - Transformaciones a los datos de conteo.

Comparaciones sencillas (tratamiento vs control) I

```
dsd <- nbinomWaldTest( dsd )
res1 <- results( dsd )
```

Comparaciones sencillas (tratamiento vs control) II

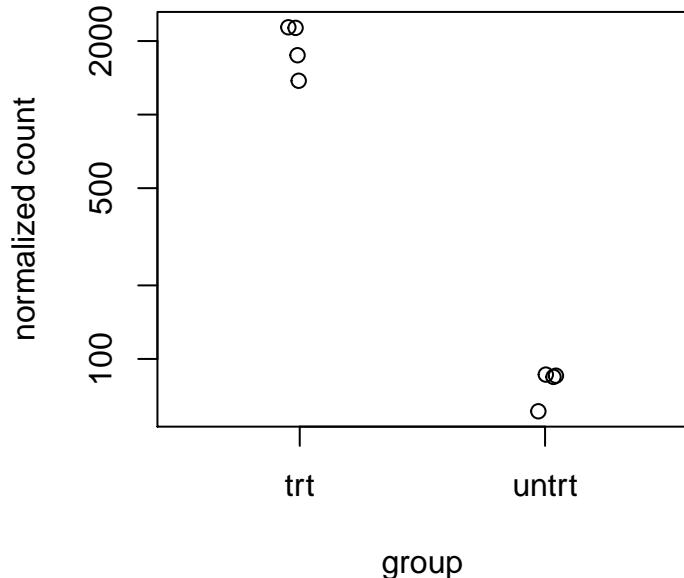
```
summary( res1 )

##
## out of 33469 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 1505, 4.5%
## LFC < 0 (down)    : 1877, 5.6%
## outliers [1]       : 98, 0.29%
## low counts [2]     : 14805, 44%
## (mean count < 4)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

Comparaciones sencillas (tratamiento vs control) III

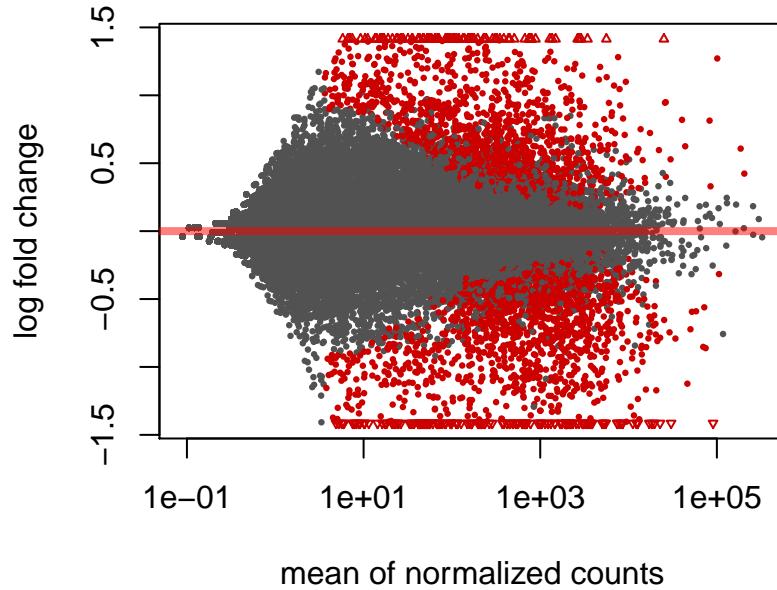
```
plotCounts(dsd,
            gene=rownames(res1)[which.min(res1$padj)],  
intgroup="dex")
```

ENSG00000152583



Comparaciones sencillas (tratamiento vs control) IV

```
plotMA( res1 )
```



Análisis de expresión diferencial con *DESeq2*

- El objeto *DESeqDataSet*
- Normalización de datos (parámetro s_j).
- Estimación de las dispersiones (parámetro α_i).
- Comparaciones sencillas (tratamiento vs control).
- **Comparaciones con factores de bloqueo.**
- Comparaciones con interacciones.
- Transformaciones a los datos de conteo.

Comparaciones con factores de bloqueo I

¿Cuándo se usa un factor de bloqueo? Ejemplos: muestras pareadas, variables técnicas o biológicas adicionales.

```
colData( dsd )[, c( "cell", "dex" ) ]
```

```
## DataFrame with 8 rows and 2 columns
##           cell      dex
##           <factor> <factor>
## SRR1039508  N61311    untrt
## SRR1039509  N61311      trt
## SRR1039512  N052611    untrt
## SRR1039513  N052611      trt
## SRR1039516  N080611    untrt
## SRR1039517  N080611      trt
## SRR1039520  N061011    untrt
## SRR1039521  N061011      trt
```

Comparaciones con factores de bloqueo II

```

dsdSimple <- dsd
design( dsd ) <- ~ cell + dex
dsd <- estimateDispersions( dsd )

## found already estimated dispersions, replacing these
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
dsd <- nbinomWaldTest( dsd )

```

Comparaciones con factores de bloqueo III

¿Hace alguna diferencia introducir la línea celular al modelo?

```

res2 <- results( dsd,
                  contrast=c( "dex", "trt", "untrt" ) )
table( 'complex'=res2$padj < 0.1, 'simple'=res1$padj < 0.1 )

##      simple
## complex FALSE  TRUE
##   FALSE 13058   100
##   TRUE   1529  3265

```

Ejercicio IV

¿Cuántos genes están diferencialmente expresados entre las líneas celulares N061011 y N080611?

Análisis de expresión diferencial con *DESeq2*

- El objeto *DESeqDataSet*
- Normalización de datos (parámetro s_j).
- Estimación de las dispersiones (parámetro α_i).
- Comparaciones sencillas (tratamiento vs control).
- Comparaciones con factores de bloqueo.
- **Comparaciones con interacciones.**
- Transformaciones a los datos de conteo.

Comparaciones con interacciones I

¿Qué genes responden al tratamiento de manera diferente entre las líneas celulares N61311 y N080611?

```

design( dsd ) <- ~ cell * dex
dsd <- estimateDispersions( dsd )

## found already estimated dispersions, replacing these
## Warning in checkForExperimentalReplicates(object, modelMatrix): same number of samples and coefficients
##   estimating dispersion by treating samples as replicates.
##   read the ?DESeq section on 'Experiments without replicates'

## gene-wise dispersion estimates

```

```

## mean-dispersion relationship
## final dispersion estimates
dsd <- nbinomWaldTest( dsd )

```

Comparaciones con interacciones II

¿Porqué el código de arriba muestra ‘warnings’?

```

colData( dsd )[,c( "cell", "dex" )]

## DataFrame with 8 rows and 2 columns
##           cell      dex
##           <factor> <factor>
## SRR1039508   N61311    untrt
## SRR1039509   N61311      trt
## SRR1039512   N052611    untrt
## SRR1039513   N052611      trt
## SRR1039516   N080611    untrt
## SRR1039517   N080611      trt
## SRR1039520   N061011    untrt
## SRR1039521   N061011      trt

```

Comparaciones con interacciones III

```

resultsNames( dsd )

## [1] "Intercept"
## [2] "cell_N061011_vs_N052611"
## [3] "cell_N080611_vs_N052611"
## [4] "cell_N61311_vs_N052611"
## [5] "dex_untrt_vs_trt"
## [6] "cellN061011.dexuntrt"
## [7] "cellN080611.dexuntrt"
## [8] "cellN61311.dexuntrt"
res4 <- results( dsd, list(
  "cellN61311.dexuntrt",
  "cellN080611.dexuntrt" ) )

```

Comparaciones con interacciones III

```

summary( res4 )

##
## out of 33469 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 0, 0%
## LFC < 0 (down)    : 0, 0%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 0)

```

```
## [1] see 'cooksCutoff' argument of ?results  
## [2] see 'independentFiltering' argument of ?results
```

Expresión diferencial en una sola línea de código

```
design(dsd) <- ~ cell + dex  
dsd <- DESeq(dsd)  
  
## using pre-existing size factors  
## estimating dispersions  
## found already estimated dispersions, replacing these  
## gene-wise dispersion estimates  
## mean-dispersion relationship  
## final dispersion estimates  
## fitting model and testing
```

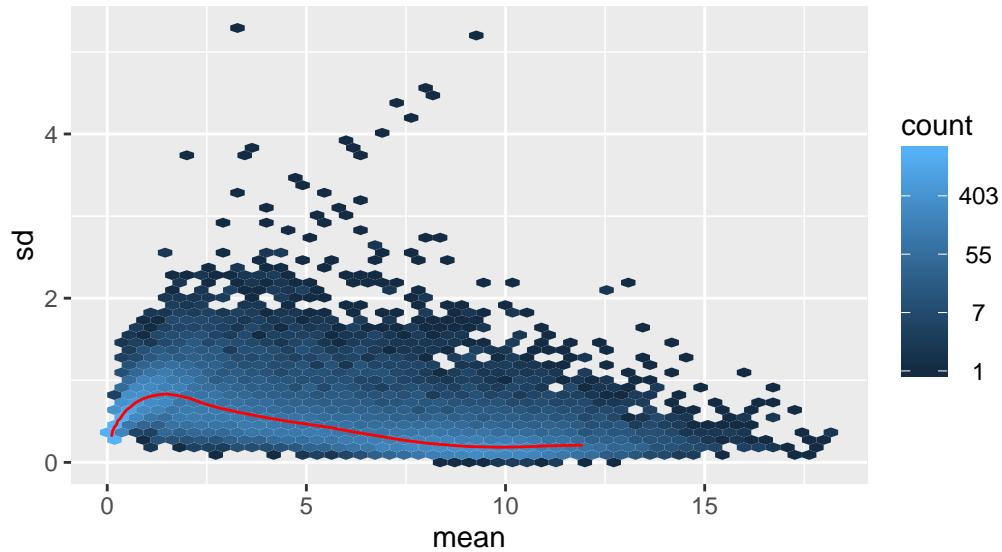
Análisis de expresión diferencial con *DESeq2*

- El objeto *DESeqDataSet*.
- Normalización de datos (parámetro s_j).
- Estimación de las dispersiones (parámetro α_i).
- Comparaciones sencillas (tratamiento vs control).
- Comparaciones con factores de bloqueo.
- Comparaciones con interacciones.
- Transformaciones a los datos de conteo.

Transformaciones a los datos de conteo I

Problema: dependencia de la varianza en la media ocasiona ruido

```
suppressMessages(library('vsn'))  
notAllZero <- (rowSums(counts(dsd))>0)  
meanSdPlot(  
  log2( counts( dsd, normalized=TRUE )[notAllZero,] + 1),  
  ranks=FALSE)
```



Transformaciones a los datos de conteo II

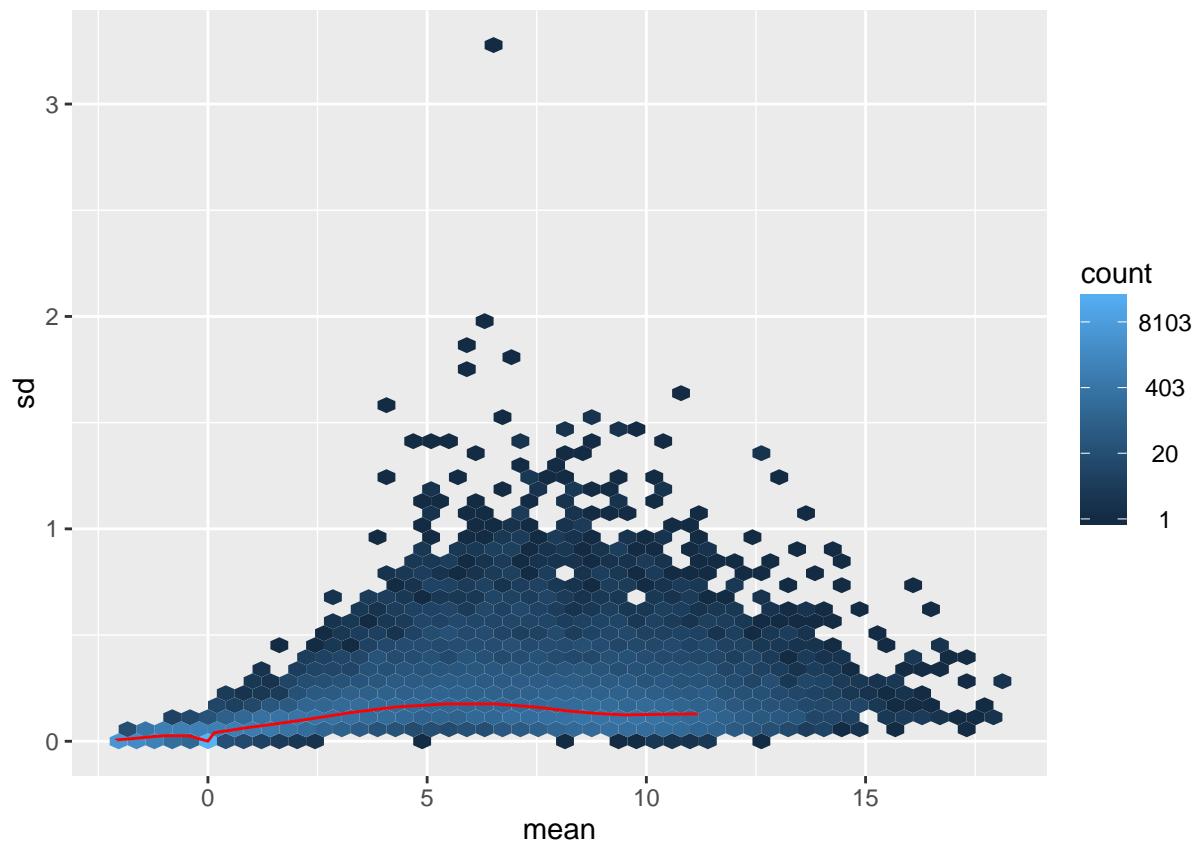
```
rld <- rlog( dsd )
vsd <- varianceStabilizingTransformation(dsd)
```

Ejercicio V

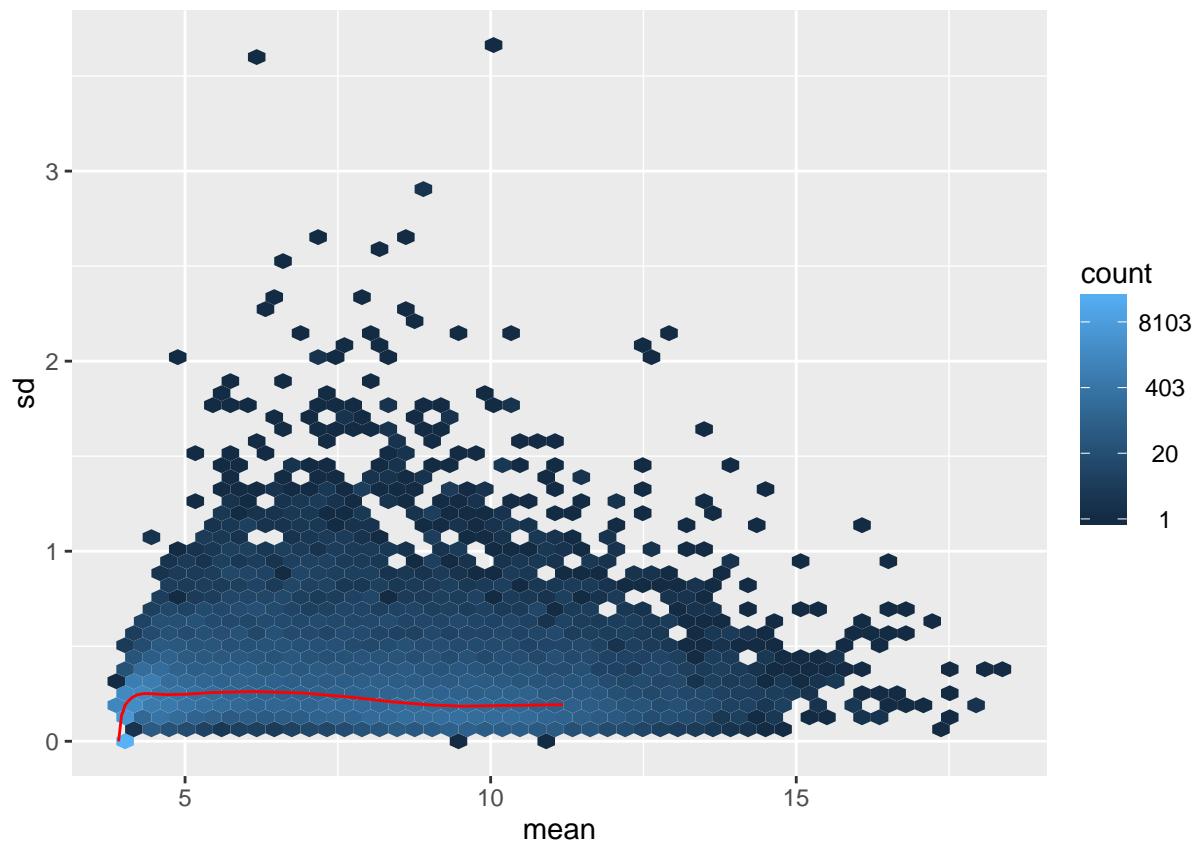
Repetir gráfico de la media y varianza, ahora con los datos transformados.

Solución V

```
meanSdPlot(assays(rld)[[1]], ranks = FALSE)
```



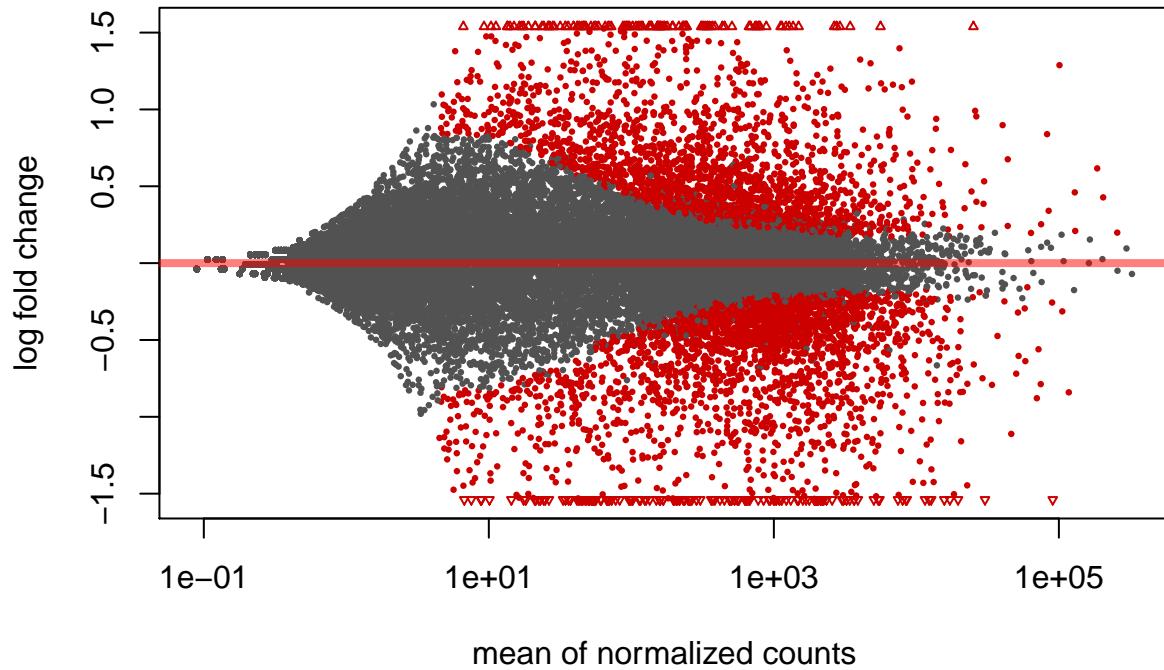
```
meanSdPlot(assays(vsd)[[1]], ranks = FALSE)
```



Visualizando resultados

Mean-Average

```
plotMA(dsd)
```

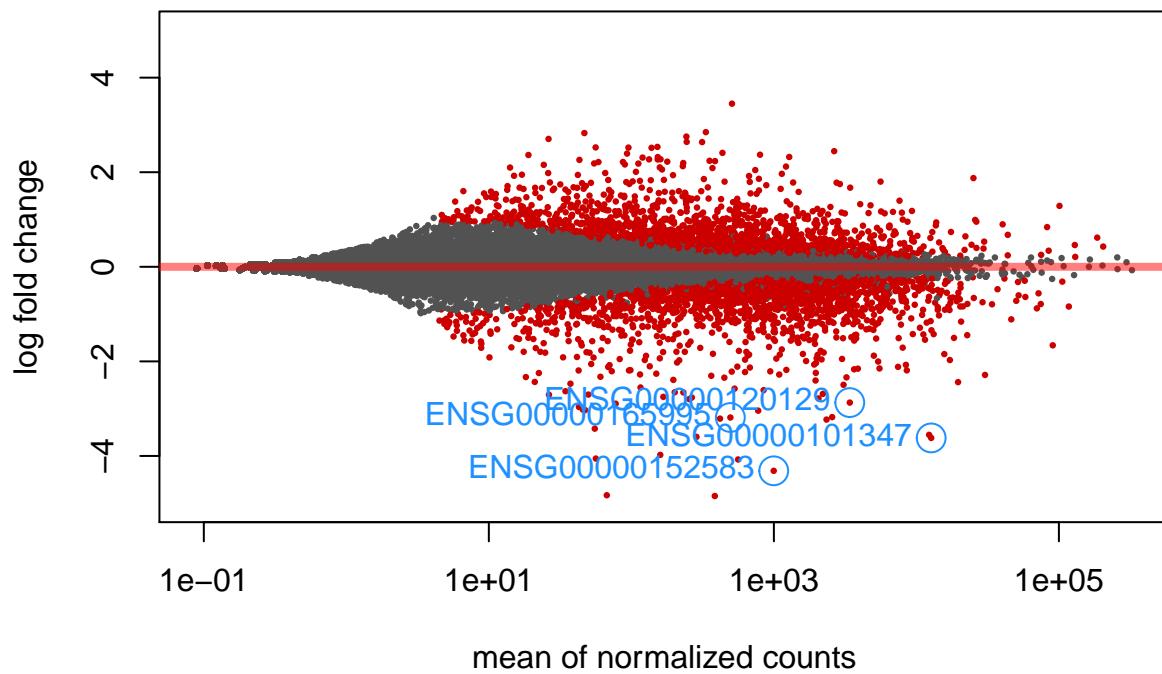


```
?plotMA
```

```
## S4 method for signature 'DESeqDataSet'
plotMA(object, alpha = 0.1, main = "",
       xlab = "mean of normalized counts", ylim, MLE = FALSE, ...)
```

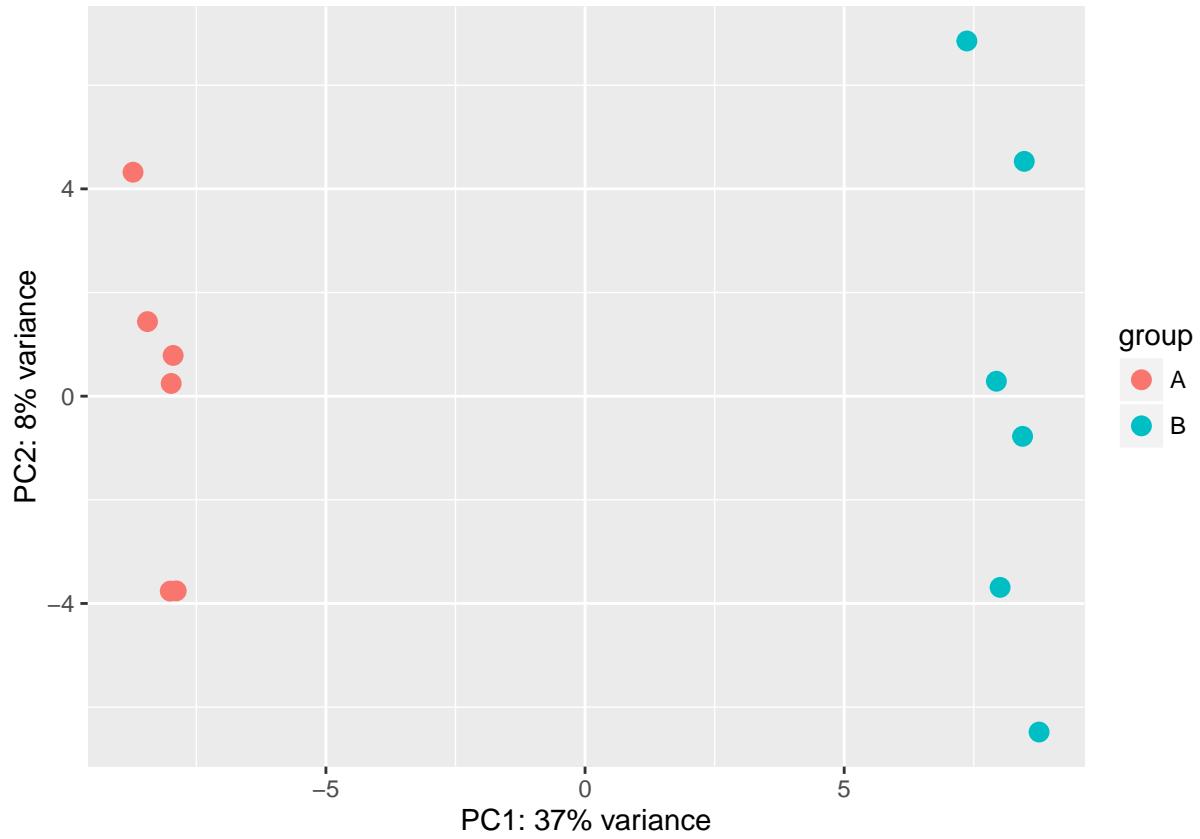
```
res <- results(dsd, alpha = 0.05)
top <- head(order(res$padj), n = 4)
plotMA(dsd, alpha=0.05, main='4 genes', ylim = c(-5, 5))
with(res[top, ], {
  points(baseMean, log2FoldChange, col="dodgerblue", cex=2)
  text(baseMean, log2FoldChange, rownames(res)[top], pos=2,
       col="dodgerblue")
})
```

4 genes



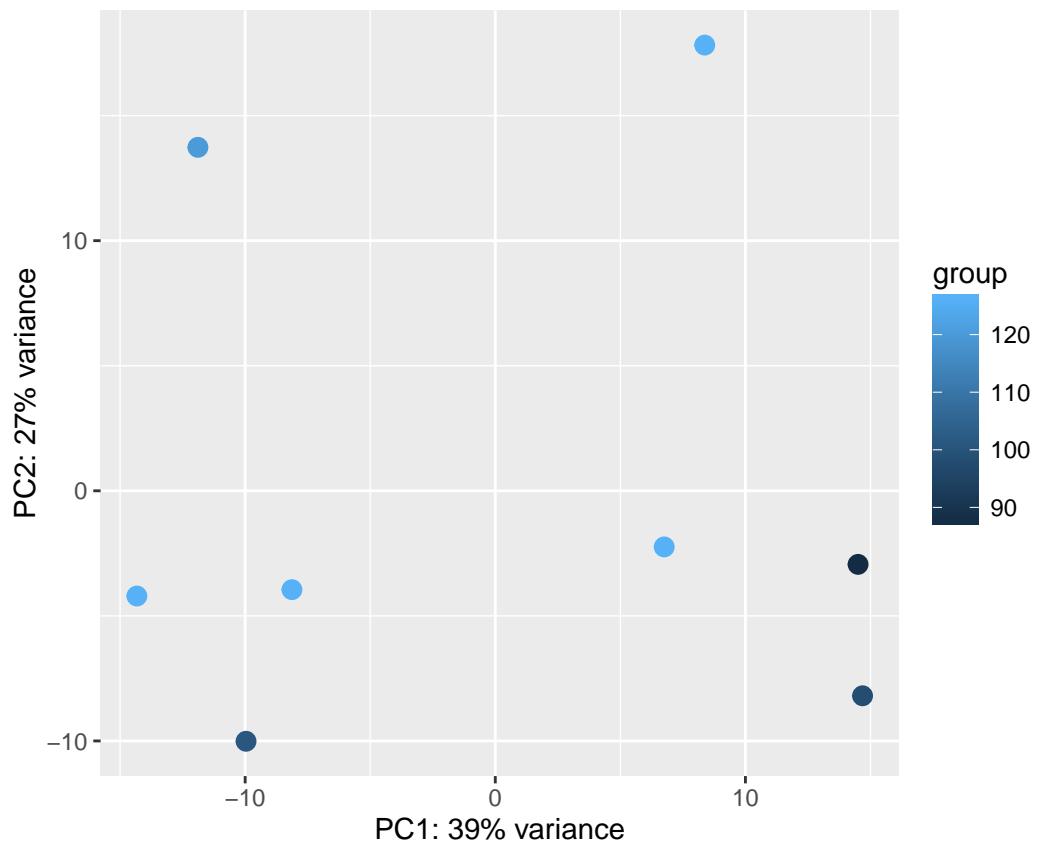
Grupos de muestras

```
prueba <- rlog(makeExampleDESeqDataSet(betaSD=1))
plotPCA(prueba)
```

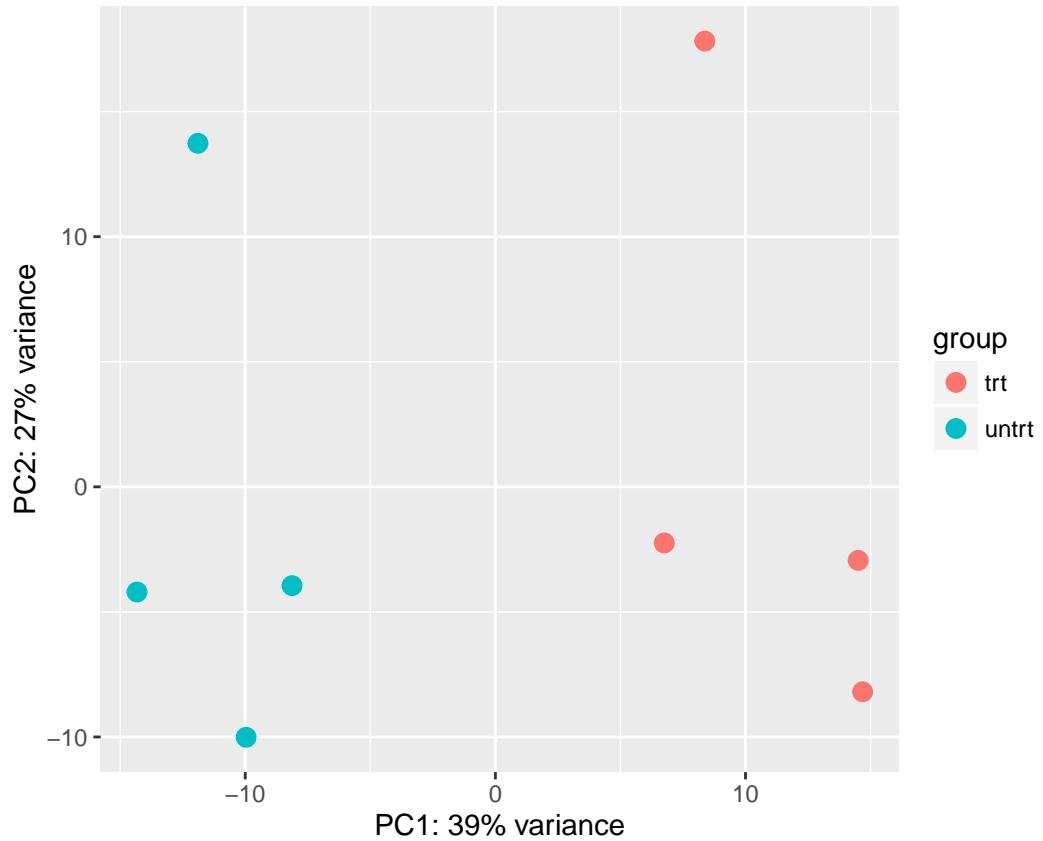


Con nuestras muestras

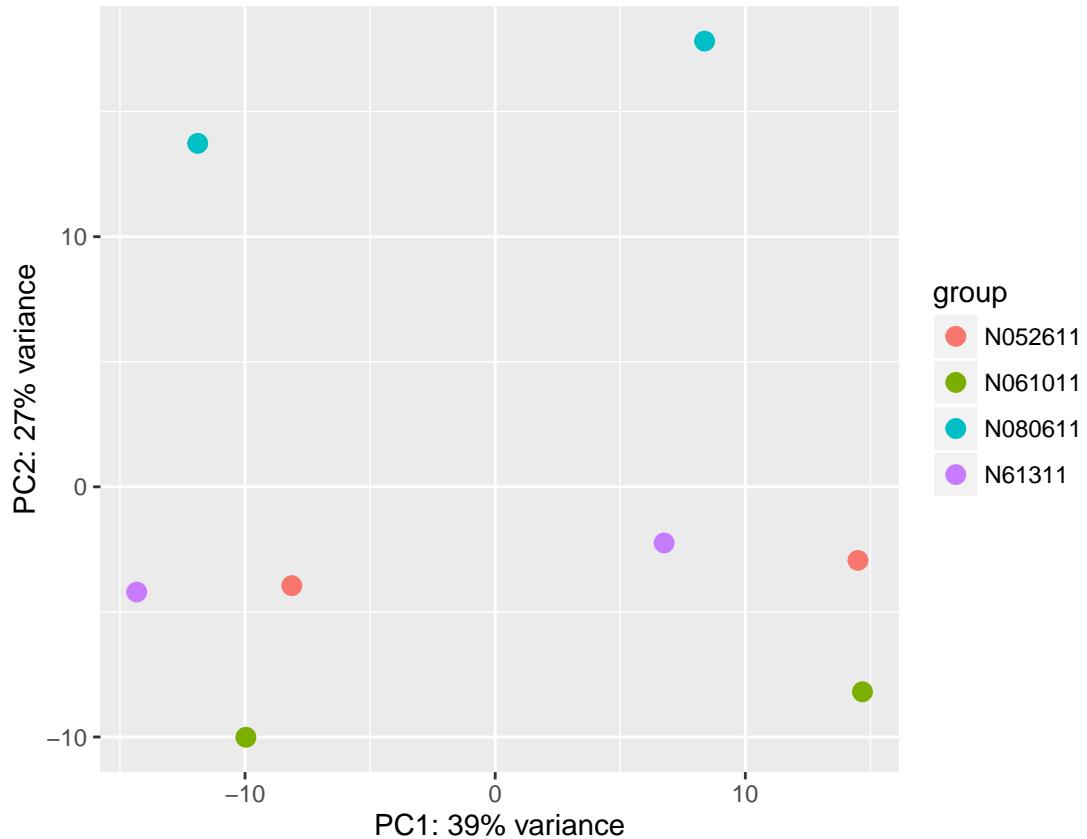
```
plotPCA(rld, intgroup = 'avgLength')
```



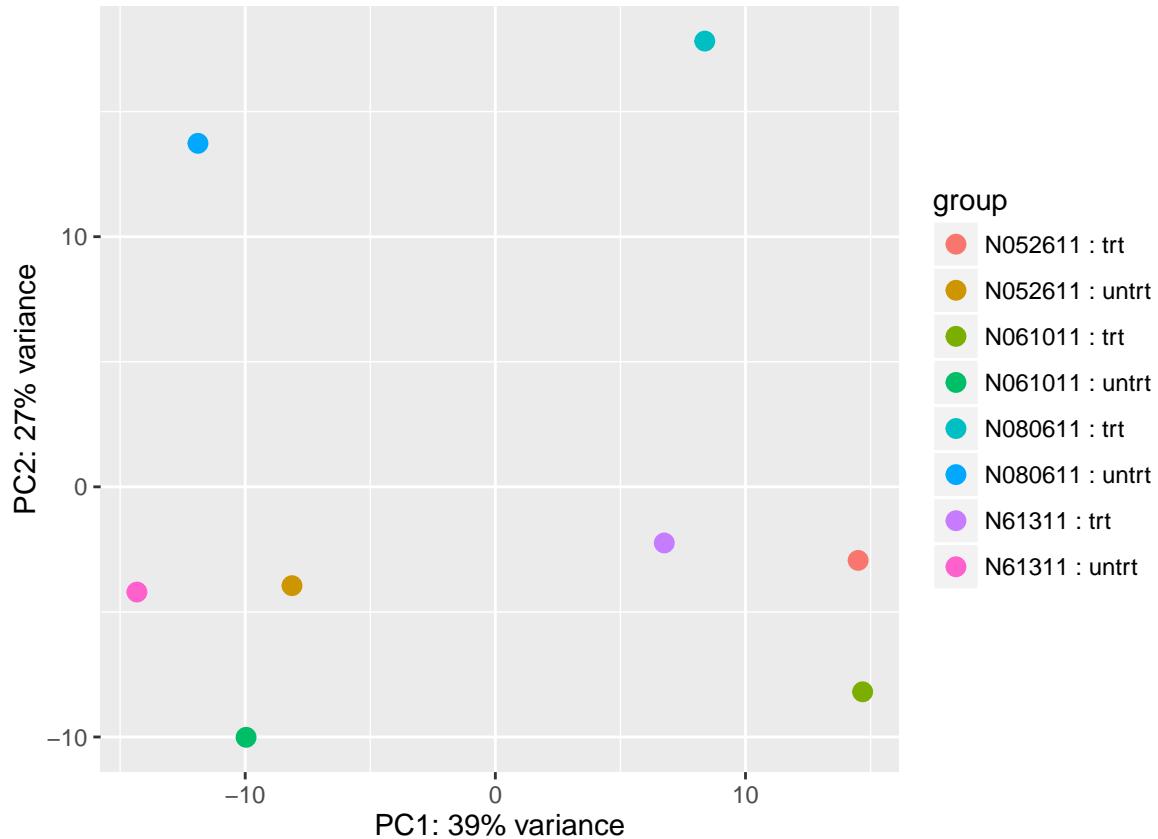
```
plotPCA(rld, intgroup = 'dex')
```



```
plotPCA(rld, intgroup = 'cell')
```



```
plotPCA(rld, intgroup = c('cell', 'dex'))
```

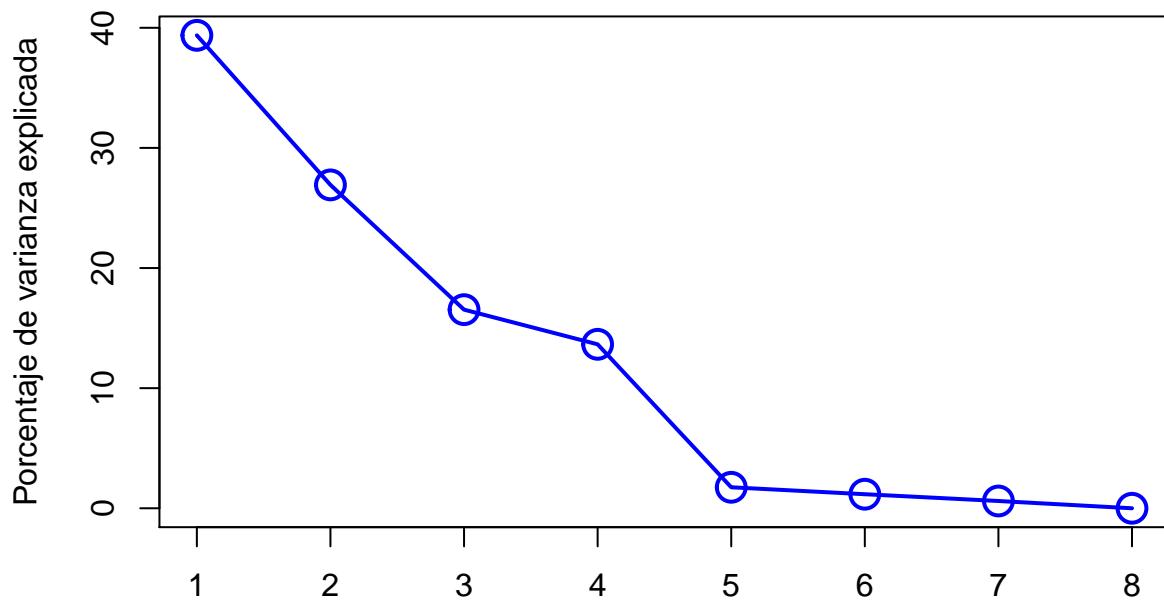


Varianza explicada

- <https://github.com/Bioconductor-mirror/DESeq2/blob/master/R/plots.R>

```
rv <- matrixStats:::rowVars(assay(rld))
select <- order(rv, decreasing=TRUE)[seq_len(500)]
pca <- prcomp(t(assay(rld)[select, ]))
percentVar <- pca$sdev^2 / sum( pca$sdev^2 ) * 100
names(percentVar) <- paste0('PC', seq_len(length(percentVar)))
```

```
plot(percentVar, type = 'o', pch = 21, cex = 2,
     xlab = '', ylab = 'Porcentaje de varianza explicada',
     col = 'blue', lwd = 2)
```



Manhattan-plot

- Necesitamos las coordenadas de los genes

```
library('EnsDb.Hsapiens.v75')

## Loading required package: ensemblDb
## Loading required package: GenomicFeatures
## Loading required package: AnnotationDbi
genes <- genes(EnsDb.Hsapiens.v75)
```

```
genes
```

```
## GRanges object with 64102 ranges and 6 metadata columns:
##           seqnames      ranges
##           <Rle>      <IRanges>
##   ENSG00000223972     1 [11869, 14412]
##   ENSG00000227232     1 [14363, 29806]
##   ENSG00000243485     1 [29554, 31109]
##   ENSG00000237613     1 [34554, 36081]
##   ENSG00000268020     1 [52473, 54936]
##   ...
##   ENSG00000224240     Y [28695572, 28695890]
##   ENSG00000227629     Y [28732789, 28737748]
##   ENSG00000237917     Y [28740998, 28780799]
##   ENSG00000231514     Y [28772667, 28773306]
##   ENSG00000235857     Y [59001391, 59001635]
##           strand |      gene_id
##           <Rle> |      <character>
##   ENSG00000223972     + | ENSG00000223972
```

```

##   ENSG00000227232      - | ENSG00000227232
##   ENSG00000243485      + | ENSG00000243485
##   ENSG00000237613      - | ENSG00000237613
##   ENSG00000268020      + | ENSG00000268020
##   ...
##   ENSG00000224240      + | ENSG00000224240
##   ENSG00000227629      - | ENSG00000227629
##   ENSG00000237917      - | ENSG00000237917
##   ENSG00000231514      - | ENSG00000231514
##   ENSG00000235857      + | ENSG00000235857
##           gene_name
##           <character>
##   ENSG00000223972      DDX11L1
##   ENSG00000227232      WASH7P
##   ENSG00000243485      MIR1302-10
##   ENSG00000237613      FAM138A
##   ENSG00000268020      OR4G4P
##   ...
##   ENSG00000224240      CYCSP49
##   ENSG00000227629      SLC25A15P1
##   ENSG00000237917      PARP4P1
##   ENSG00000231514      FAM58CP
##   ENSG00000235857      CTBP2P1
##           entrezid
##           <character>
##   ENSG00000223972      100287596;100287102
##   ENSG00000227232      100287171;653635
##   ENSG00000243485      100422834;100422831;100422919;100302278
##   ENSG00000237613      654835;645520;641702
##   ENSG00000268020
##   ...
##   ENSG00000224240
##   ENSG00000227629
##   ENSG00000237917
##   ENSG00000231514
##   ENSG00000235857
##           gene_biotype seq_coord_system
##           <character>    <character>
##   ENSG00000223972      pseudogene    chromosome
##   ENSG00000227232      pseudogene    chromosome
##   ENSG00000243485      lincRNA      chromosome
##   ENSG00000237613      lincRNA      chromosome
##   ENSG00000268020      pseudogene    chromosome
##   ...
##   ENSG00000224240      pseudogene    chromosome
##   ENSG00000227629      pseudogene    chromosome
##   ENSG00000237917      pseudogene    chromosome
##   ENSG00000231514      pseudogene    chromosome
##   ENSG00000235857      pseudogene    chromosome
##           symbol
##           <character>
##   ENSG00000223972      DDX11L1
##   ENSG00000227232      WASH7P
##   ENSG00000243485      MIR1302-10

```

```

##  ENSG00000237613      FAM138A
##  ENSG00000268020      OR4G4P
##
##      ...
##  ENSG00000224240      CYCSP49
##  ENSG00000227629      SLC25A15P1
##  ENSG00000237917      PARP4P1
##  ENSG00000231514      FAM58CP
##  ENSG00000235857      CTBP2P1
##
##  -----
##  seqinfo: 273 sequences from GRCh37 genome

```

Añadimos los qvalues

```

## Re-ordenar
genes <- genes[match(rownames(dsd), names(genes))]
genes$pvalue <- -log10(res$padj)
length(genes)

```

```

## [1] 64102
genes <- keepSeqlevels(genes, c(1:22, 'X', 'Y'))
length(genes)

```

```

## [1] 58153

```

```

library('ggbio')

## Loading required package: ggplot2
## Need specific help about ggbio? try mailing
##   the maintainer or visit http://tengfei.github.com/ggbio/
##
## Attaching package: 'ggbio'
## The following objects are masked from 'package:ggplot2':
## 
##     geom_bar, geom_rect, geom_segment,
##     ggsave, stat_bin, stat_identity, xlim

```

```

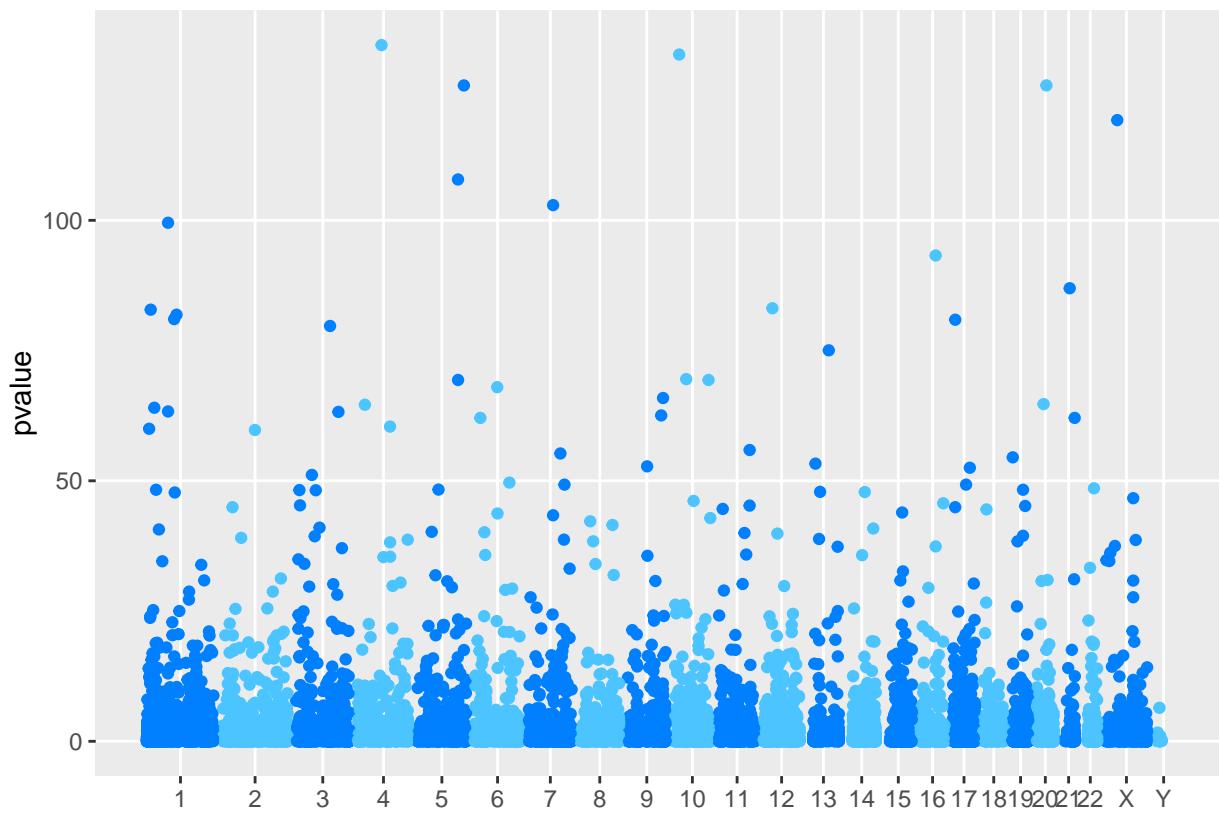
plotGrandLinear(genes, aes(y = pvalue))

```

```

## using coord	genome to parse x scale

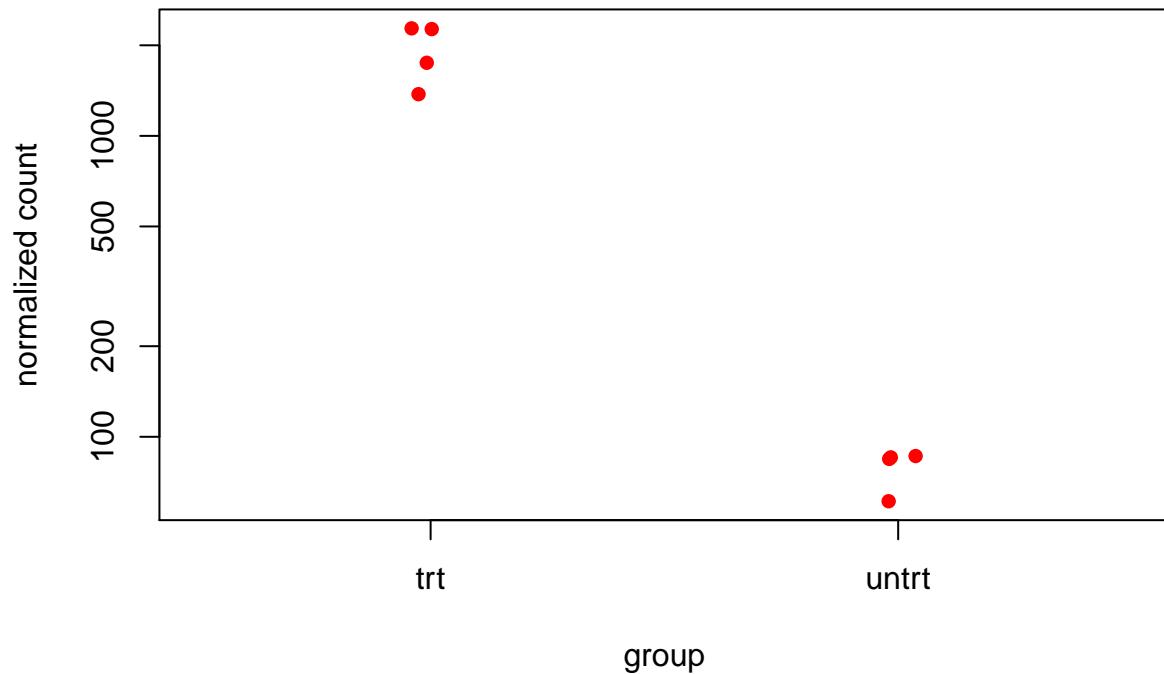
```



Un gen a la vez

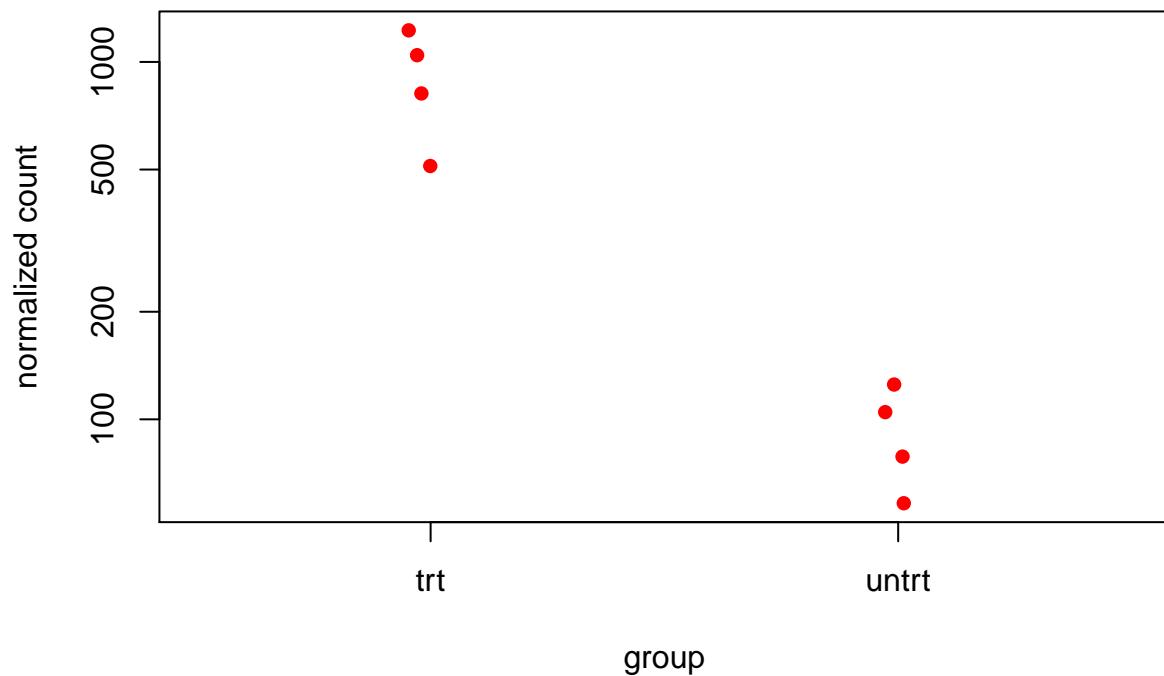
```
plotCounts(dsd, gene = top[1], intgroup = 'dex',
           col = 'red', pch = 16)
```

ENSG00000152583



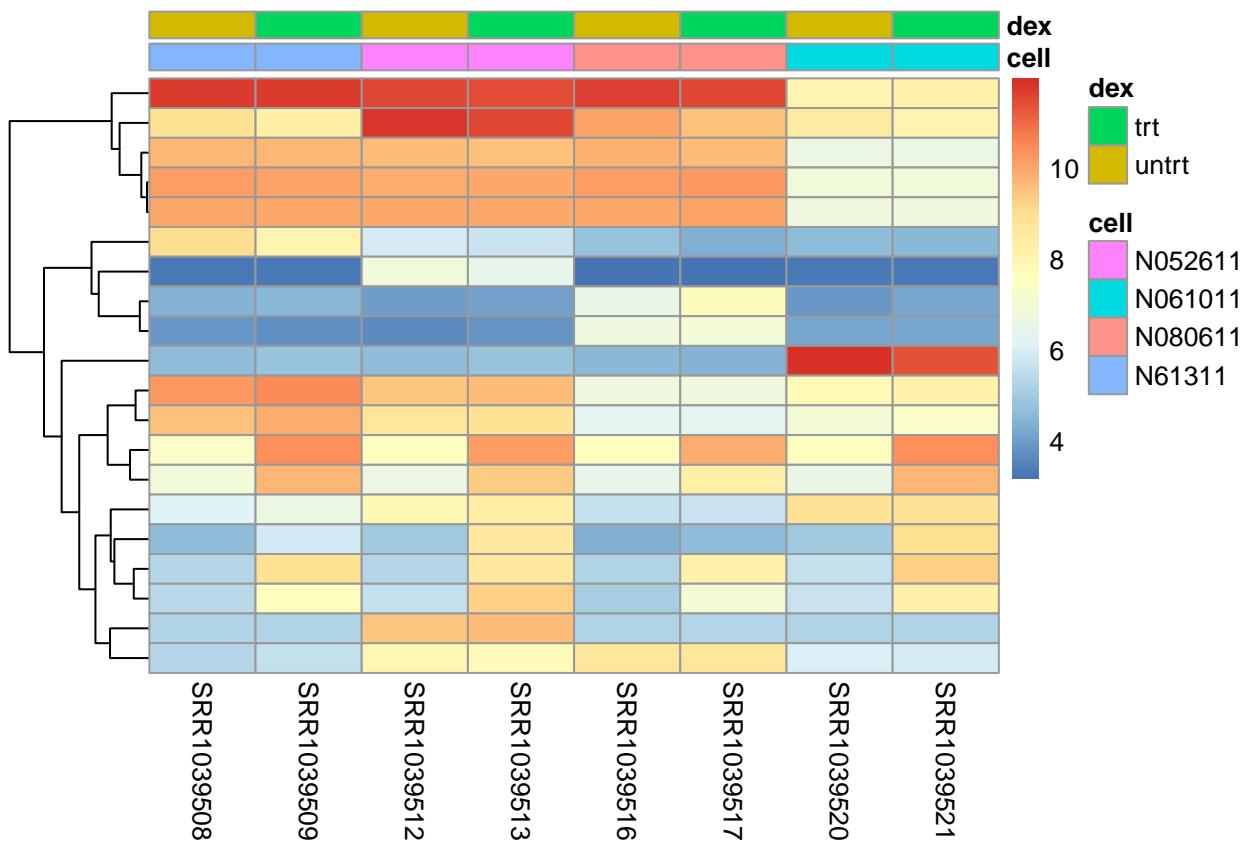
```
plotCounts(dsd, gene = top[2], intgroup = 'dex',
           col = 'red', pch = 16)
```

ENSG00000165995



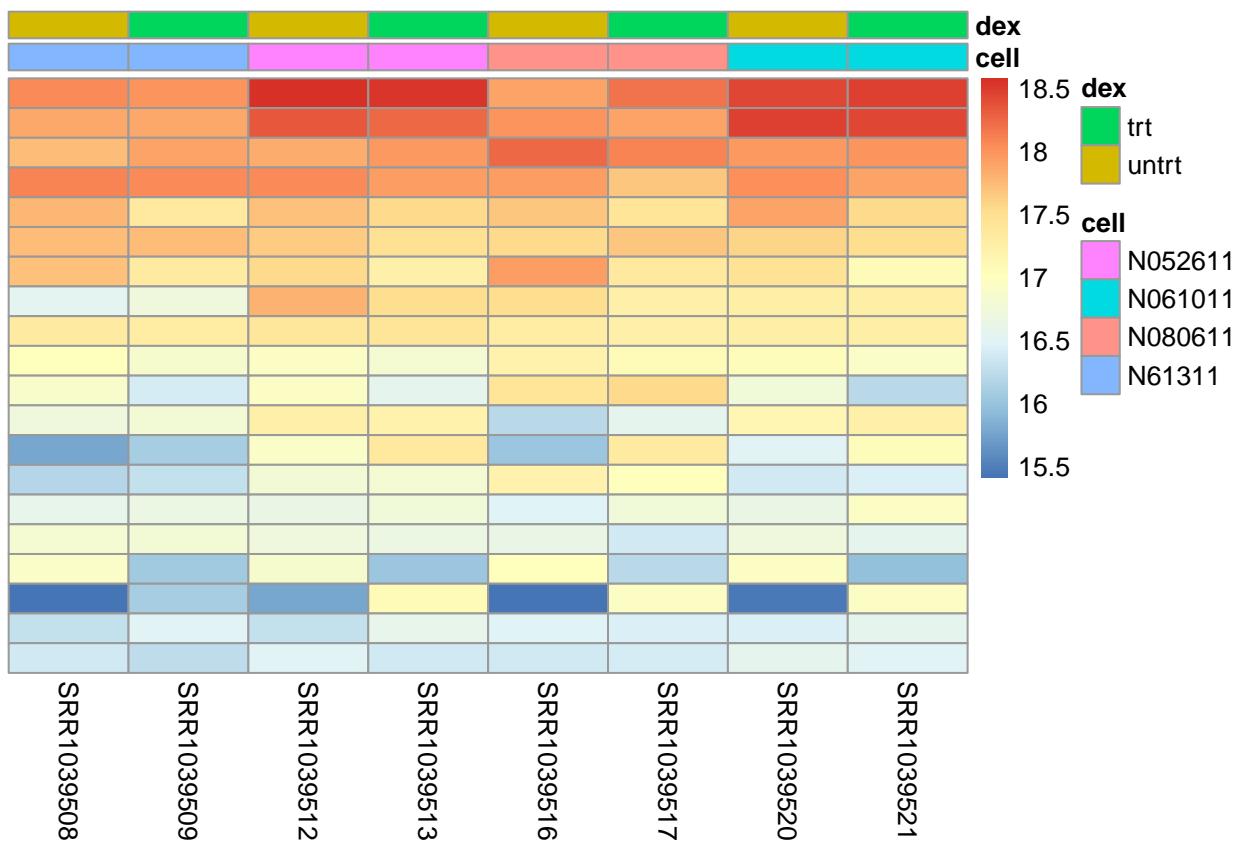
Varios genes a la vez

```
library('pheatmap')
df <- as.data.frame(colData(dsd)[, c('cell', 'dex')])
pheatmap(assay(rld)[select[1:20],], cluster_rows=TRUE,
         show_rownames=FALSE, cluster_cols=FALSE, annotation_col=df)
```



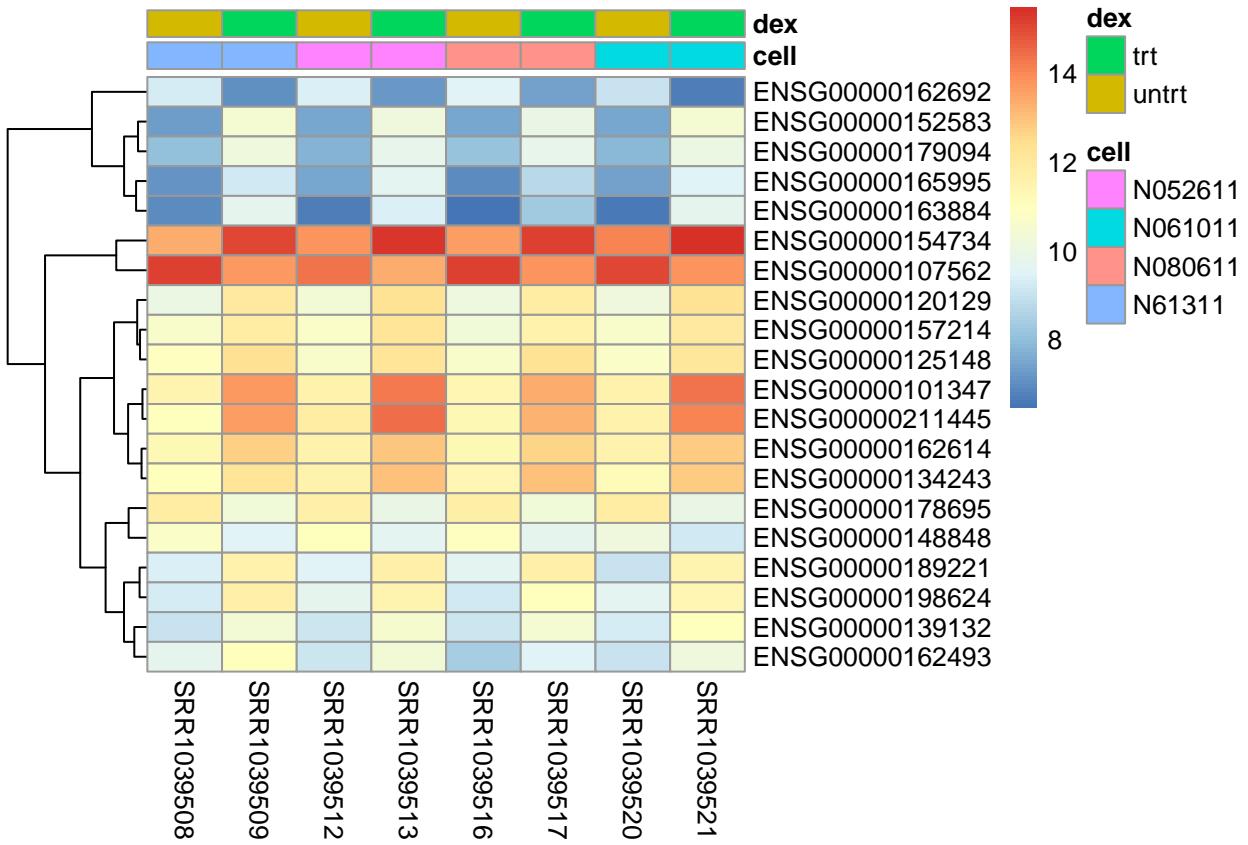
Los más expresados

```
select <- order(rowMeans(counts(dsd, normalized=TRUE)), decreasing=TRUE)
pheatmap(assay(rld)[select[1:20],], cluster_rows=FALSE,
         show_rownames=FALSE, cluster_cols=FALSE, annotation_col=df)
```



Los más diferentes

```
select <- order(res$padj, decreasing = FALSE)
diferentes <- assay(rld)[select[1:20], ]
pheatmap(diferentes, cluster_rows=TRUE,
          show_rownames=TRUE, cluster_cols=FALSE, annotation_col=df)
```



Con símbolos

```
library('org.Hs.eg.db')

##
coll <- function(x) { paste(x, collapse = ' - ')}

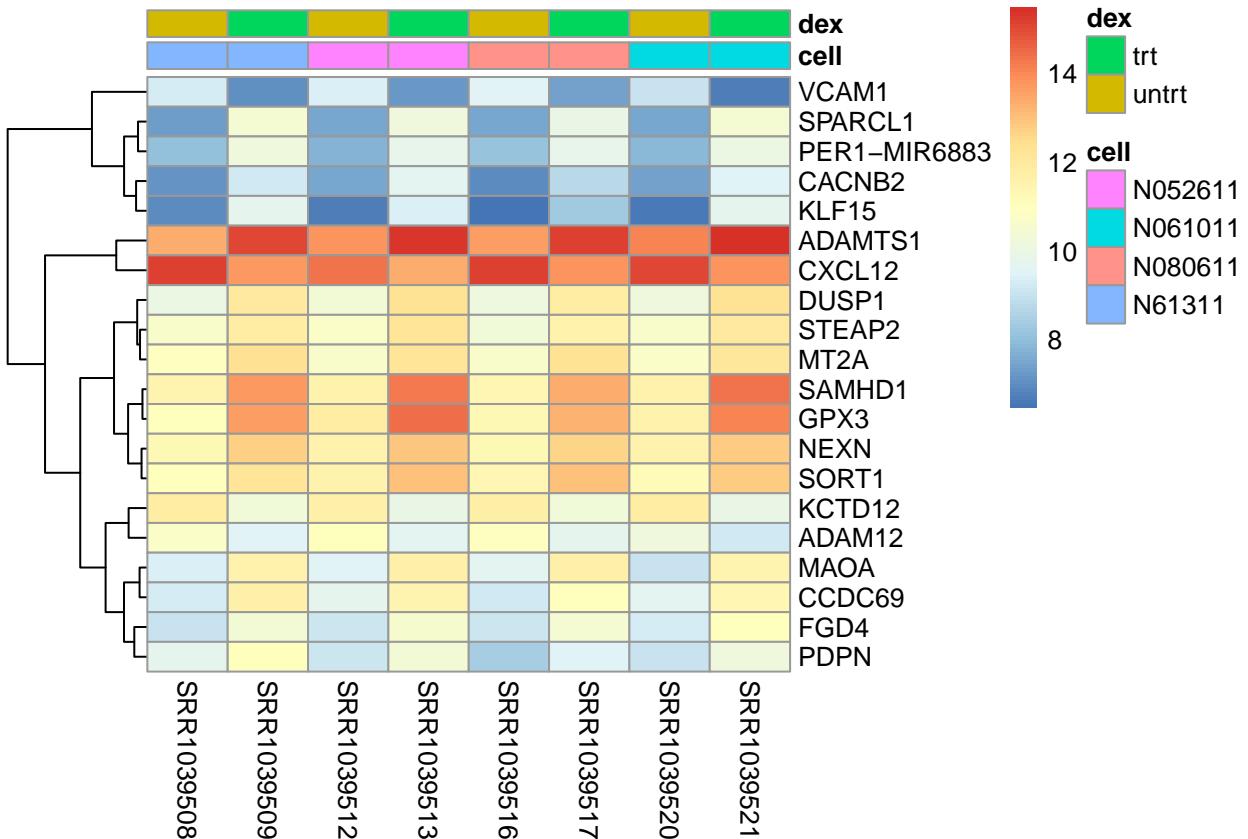
simbolos <- mapIds(org.Hs.eg.db,
  keys = rownames(diferentes), column = 'SYMBOL',
  keytype = 'ENSEMBL', multiVals = coll)

## 'select()' returned 1:many mapping between
## keys and columns
simbolos[1:2]

## ENSG00000152583 ENSG00000165995
##      "SPARCL1"      "CACNB2"

## Cambiamos los nombres de los genes
rownames(diferentes) <- simbolos
```

```
pheatmap(diferentes, cluster_rows=TRUE,
  show_rownames=TRUE, cluster_cols=FALSE, annotation_col=df)
```



Hagamos un reporte

En un archivo de texto, *conclusiones.md* copien y peguen:

```
# Nuestro heatmap

```{r 'mi_heatmap'}
library('pheatmap')
ann_df <- as.data.frame(colData(dds)[, c('cell', 'dex')])
select <- order(res$padj, decreasing = FALSE)
diferentes <- assay(rld)[select[1:20],]
pheatmap(diferentes, cluster_rows=TRUE,
 show_rownames=TRUE, cluster_cols=FALSE,
 annotation_col=ann_df)
```

```

En la imagen anterior vemos...

```
# Conclusiones
```

Lo que gusten

```

library('regionReport')
DESeq2Report(dds = dsd, intgroup = c('cell', 'dex'),
             nBest = 20, nBestFeatures = 2,
             customCode = file.path(getwd(), 'conclusiones.md'))

```

El reporte resultante es DESeq2Exploration

Ejercicio final

- En el reporte muestren los símbolos de los genes en vez de los identificadores de *ENSEMBL*.
- Agreguen sus conclusiones.
- Suban su archivo html a un gist <https://gist.github.com/> público
- Copien la liga
- Peguen la liga en <http://rawgit.com/>
- Compartan la liga a sus jefes y/o amigos =)
- Ejemplo final: liga pública

Más información

- <http://bioconductor.org/>
- <http://biorxiv.org/>
- <http://biorxiv.org/content/early/2016/08/30/021592>

Información técnica

```

##  setting  value
##  version  R version 3.3.1 (2016-06-21)
##  system   x86_64, darwin13.4.0
##  ui        X11
##  language (EN)
##  collate  en_US.UTF-8
##  tz        America/New_York
##  date     2016-10-18

##  package      * version date      source
##  acepack       1.3-3.3 2014-11-24 CRAN (R 3.3.0)
##  affy          1.51.1  2016-09-01 Bioconductor
##  affyio        1.43.0  2016-05-27 Bioconductor
##  annotate      1.51.1  2016-09-18 Bioconductor
##  AnnotationDbi * 1.35.5 2016-10-15 Bioconductor
##  AnnotationHub 2.5.14  2016-10-15 Bioconductor
##  assertthat     0.1    2013-12-06 CRAN (R 3.3.0)
##  backports      1.0.3   2016-06-28 CRAN (R 3.3.0)

```

```

##  bibtex          0.4.0  2014-12-31 CRAN (R 3.3.0)
##  Biobase         * 2.33.4 2016-10-15 Bioconductor
##  BiocGenerics   * 0.19.2 2016-07-08 Bioconductor
##  BiocInstaller    1.23.9 2016-09-04 Bioconductor
##  BiocParallel     1.7.9  2016-10-15 Bioconductor
##  biomaRt         2.29.3 2016-10-17 Bioconductor
##  Biostrings       2.41.4 2016-06-17 Bioconductor
##  biovizBase      1.21.0 2016-05-20 Bioconductor
##  bitops           1.0-6   2013-08-17 CRAN (R 3.3.0)
##  BSgenome        1.41.2  2016-06-17 Bioconductor
##  bumphunter      1.13.1  2016-07-11 Bioconductor
##  checkmate       1.8.1   2016-06-28 CRAN (R 3.3.0)
##  chron            2.3-47  2015-06-24 CRAN (R 3.3.0)
##  cluster          2.0.5   2016-10-08 CRAN (R 3.3.0)
##  codetools        0.2-15  2016-10-05 CRAN (R 3.3.1)
##  colorout         * 1.1-2   2016-05-05 Github (jalvesaq/colorout@6538970)
##  colorspace       1.2-7   2016-10-11 CRAN (R 3.3.0)
##  data.table       1.9.6   2015-09-19 CRAN (R 3.3.0)
##  DBI              0.5-1   2016-09-10 CRAN (R 3.3.0)
##  DEFormats        1.1.2   2016-05-27 Bioconductor

```

```

##  package          * version  date      source
##  knitrBootstrap   1.0.0    2016-05-05 Github (jimhester/knitrBootstrap@cdaa4a9)
##  labeling          0.3      2014-08-23 CRAN (R 3.3.0)
##  lattice           0.20-34 2016-09-06 CRAN (R 3.3.0)
##  latticeExtra      0.6-28  2016-02-09 CRAN (R 3.3.0)
##  limma             3.29.24 2016-10-17 Bioconductor
##  locfit            1.5-9.1 2013-04-20 CRAN (R 3.3.0)
##  lubridate         1.6.0    2016-09-13 CRAN (R 3.3.0)
##  magrittr          1.5      2014-11-22 CRAN (R 3.3.0)
##  markdown          0.7.7   2015-04-22 CRAN (R 3.3.0)
##  Matrix            1.2-7.1 2016-09-01 CRAN (R 3.3.0)
##  matrixStats       0.51.0   2016-10-09 CRAN (R 3.3.0)
##  memoise           1.0.0    2016-01-29 CRAN (R 3.3.0)
##  mime              0.5      2016-07-07 CRAN (R 3.3.0)
##  munsell           0.4.3    2016-02-13 CRAN (R 3.3.0)
##  nnet              7.3-12  2016-02-02 CRAN (R 3.3.1)
##  org.Hs.eg.db     * 3.4.0   2016-10-06 Bioconductor
##  OrganismDbi      1.15.2   2016-10-15 Bioconductor
##  pheatmap          * 1.0.8   2015-12-11 CRAN (R 3.3.0)
##  pkgmaker          0.22     2014-05-14 CRAN (R 3.3.0)
##  plyr              1.8.4    2016-06-08 CRAN (R 3.3.0)
##  preprocessCore    1.35.0   2016-05-27 Bioconductor
##  qvalue            2.5.2    2016-05-27 Bioconductor
##  R6                 2.2.0    2016-10-05 CRAN (R 3.3.1)
##  RBGL              1.49.3   2016-08-14 Bioconductor
##  RColorBrewer      1.1-2    2014-12-07 CRAN (R 3.3.0)
##  Rcpp              0.12.7   2016-09-05 CRAN (R 3.3.0)
##  RCurl              1.95-4.8 2016-03-01 CRAN (R 3.3.0)
##  RefManageR        0.11.0   2016-09-11 CRAN (R 3.3.1)

```

```

## package           * version date     source
## evaluate          0.10    2016-10-11 CRAN (R 3.3.0)
## foreach           1.4.3   2015-10-13 CRAN (R 3.3.0)
## foreign            0.8-67  2016-09-13 CRAN (R 3.3.0)
## formatR            1.4     2016-05-09 CRAN (R 3.3.0)
## Formula           1.2-1   2015-04-07 CRAN (R 3.3.0)
## genefilter         1.55.2   2016-05-27 Bioconductor
## geneplotter        1.51.0   2016-05-05 Bioconductor
## GenomeInfoDb      * 1.9.15  2016-10-15 Bioconductor
## GenomicAlignments 1.9.6    2016-07-17 Bioconductor
## GenomicFeatures   * 1.25.20 2016-10-03 Bioconductor
## GenomicFiles       1.9.12   2016-07-31 Bioconductor
## GenomicRanges     * 1.25.95 2016-10-17 Bioconductor
## GGally              1.2.0    2016-07-01 CRAN (R 3.3.0)
## ggbio              * 1.21.7   2016-10-15 Bioconductor
## ggplot2             * 2.1.0    2016-03-01 CRAN (R 3.3.0)
## graph               1.51.0   2016-05-05 Bioconductor
## gridExtra           2.2.1    2016-02-29 CRAN (R 3.3.0)
## gtable              0.2.0    2016-02-26 CRAN (R 3.3.0)
## hexbin              * 1.27.1   2015-08-19 CRAN (R 3.3.0)
## Hmisc                3.17-4   2016-05-02 CRAN (R 3.3.0)
## htmltools            0.3.5    2016-03-21 CRAN (R 3.3.0)
## httpuv              1.3.3    2015-08-04 CRAN (R 3.3.0)
## httr                 1.2.1    2016-07-03 CRAN (R 3.3.0)
## interactiveDisplayBase 1.11.3   2016-05-20 Bioconductor
## IRanges              * 2.7.17   2016-10-15 Bioconductor
## iterators            1.0.8    2015-10-13 CRAN (R 3.3.0)
## knitrCitations      1.0.7    2015-10-28 CRAN (R 3.3.0)
## knitr                * 1.14     2016-08-13 CRAN (R 3.3.0)

```

```

## package           * version date     source
## BiocGenerics      * 0.19.2  2016-07-08 Bioconductor
## BiocInstaller      1.23.9   2016-09-04 Bioconductor
## BiocParallel        1.7.9    2016-10-15 Bioconductor
## biomaRt            2.29.3   2016-10-17 Bioconductor
## Biostrings          2.41.4   2016-06-17 Bioconductor
## biovizBase          1.21.0   2016-05-20 Bioconductor
## bitops              1.0-6    2013-08-17 CRAN (R 3.3.0)
## BSgenome             1.41.2   2016-06-17 Bioconductor
## bumphunter          1.13.1   2016-07-11 Bioconductor
## checkmate            1.8.1    2016-06-28 CRAN (R 3.3.0)
## chron                2.3-47   2015-06-24 CRAN (R 3.3.0)
## cluster              2.0.5    2016-10-08 CRAN (R 3.3.0)
## codetools            0.2-15   2016-10-05 CRAN (R 3.3.1)
## colorout             * 1.1-2    2016-05-05 Github (jalvesaq/colorout@6538970)
## colorspace            1.2-7    2016-10-11 CRAN (R 3.3.0)
## data.table            1.9.6    2015-09-19 CRAN (R 3.3.0)
## DBI                  0.5-1    2016-09-10 CRAN (R 3.3.0)
## DEFormats            1.1.2    2016-05-27 Bioconductor

```

```
##  derfinder           1.7.16  2016-10-15 Bioconductor
##  derfinderHelper     1.7.5   2016-10-05 Bioconductor
##  DESeq2              * 1.13.16 2016-10-15 Bioconductor
##  devtools             * 1.12.0   2016-06-24 CRAN (R 3.3.0)
##  dichromat            2.0-0    2013-01-24 CRAN (R 3.3.0)
##  digest                0.6.10   2016-08-02 CRAN (R 3.3.0)
##  doRNG                 1.6      2014-03-07 CRAN (R 3.3.0)
##  edgeR                  3.15.6   2016-10-15 Bioconductor
##  EnsDb.Hsapiens.v75  * 1.1.0    2016-10-18 Bioconductor
##  ensemblDb            * 1.5.14   2016-09-22 Bioconductor
```

```
##  package          * version  date       source
##  survival          2.39-5   2016-06-26 CRAN (R 3.3.0)
##  tibble             1.2      2016-08-26 CRAN (R 3.3.0)
##  VariantAnnotation 1.19.13  2016-10-17 Bioconductor
##  vsn                 * 3.41.8   2016-10-17 Bioconductor
##  withr               1.0.2    2016-06-20 CRAN (R 3.3.0)
##  XML                  3.98-1.4 2016-03-01 CRAN (R 3.3.0)
##  xtable               1.8-2    2016-02-05 CRAN (R 3.3.0)
##  XVector              0.13.7   2016-07-24 Bioconductor
##  yaml                  2.1.13   2014-06-12 CRAN (R 3.3.0)
##  zlibbioc            1.19.0   2016-05-05 Bioconductor
```