

# Bioinfo\_Módulo1

▼ 31 de enero, 2023

- Beneficios de R: Gratuito, modularidad, comunidad extensa...
- Es importante aprender a pedir ayuda: descripción de los errores, uso de “**reprex**” en R.
- Beneficios de GitHub: Controles de versión , colaborativo, ayuda al monitoreo de trabajo, creación de páginas web con el archivo HTML y un archivo .nojekyll

## Introducción a Bioconductor

Tipos de paquetes de Bioconductor:

- Software (tipo principal, generado tanto por Bioconductor como público general)
- Annotation (interacción con bases de datos de anotación)
- Experiment Data (datos recabados experimentalmente)
- Workflows (muestra uso de paquetes para análisis)

\*Cada paquete puede pertenecer sólo a una rama (un tipo de paquete)

\*Podemos encontrar paquetes a través de biocViews

- **Estructura de un paquete:**
  - Etiquetas (badges)
  - Descripción
  - Citar
  - Instalación
  - Documentación (“vignette” → explica cómo usar las funciones del paquete y el orden)
  - Detalles (URL, bug reports...) y estadísticas de descarga

En todo momento se encuentran disponibles 2 versiones de Bioconductor (ramas):

- release (para descargas)
- devel (para desarrollo y pruebas)

## Actividad Grupal:

### RESOLVE: An R package for the efficient analysis of mutational signatures from cancer genomes

Me llamó la atención la forma en que el paquete realiza la búsqueda de señales de mutación, pues como la descripción menciona es complicado distinguir entre señales biológicamente relevantes y artefactos de la información o de cómputo. La novedad que presenta para resolver esto es la extracción eficiente, "exposure estimation" y el intervalo de confianza.

Para realizar la instalación en R (versión "4.2"):

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

BiocManager::install("RESOLVE")
```

**Author:** Daniele Ramazzotti [aut] , Luca De Sano [cre, aut]

**Maintainer:** Luca De Sano <luca.desano at gmail.com>

A la fecha (31/01/23) pasa las pruebas de los 3 sistemas operativos y se encuentra en el lugar 2163/2183 de descargas. No cuenta con preguntas o respuestas y usa 99 dependencias.

**Cita:** Ramazzotti D, De Sano L (2022). *RESOLVE: RESOLVE: An R package for the efficient analysis of mutational signatures from cancer genomes*

. R package version 1.0.0, <https://github.com/danro9685/RESOLVE>

### RgnTX: Colocalization analysis of transcriptome elements in the presence of isoform heterogeneity and ambiguity

Este paquete me llamó la atención porque no me queda muy claro cómo es que permite la integración de las anotaciones del transcriptoma para modelar patrones de splicing alternativo complejos, aunque creo que es importante pues experimentalmente puede ser complicado encontrar todas las diferentes isoformas.

Para realizar la instalación en R (versión "4.2"):

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

BiocManager::install("RgnTX")
```

**Author:** Yue Wang [aut, cre], Jia Meng [aut]

**Maintainer:** Yue Wang <yue.wang19 at student.xjtlu.edu.cn>

A la fecha (31/01/23) pasa las pruebas de los 3 sistemas operativos y se encuentra en el lugar 2141/2183 de descargas. No cuenta con preguntas o respuestas y usa 115 dependencias.

**Cita:** Wang Y, Meng J (2022). *RgnTX: Colocalization analysis of transcriptome elements in the presence of isoform heterogeneity and ambiguity*. R package version 1.0.0.

## crisprDesign: Comprehensive design of CRISPR gRNAs for nucleases and base editors

Este paquete me parece sencillo pero útil pues permite el diseño de RNAs guía para el sistema de nucleasa de CRISPR, siendo esta una tecnología creciente en los últimos años. Funciona únicamente para el caso de humanos y realiza búsquedas de “off-targets”, los cuales son un gran reto para el uso de esta técnica.

Para realizar la instalación en R (versión “4.2”):

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

BiocManager::install("crisprDesign")
```

**Author:** Jean-Philippe Fortin [aut, cre], Luke Hoberecht [aut]

**Maintainer:** Jean-Philippe Fortin <fortin946 at gmail.com>

A la fecha (31/01/23) pasa las pruebas de los 3 sistemas operativos y se encuentra en el lugar 2096/2183 de descargas. No cuenta con preguntas o respuestas y usa 146 dependencias.

**Cita:** Hoberecht L, Perampalam P, Lun A, Fortin J (2022). “A comprehensive Bioconductor ecosystem for the design of CRISPR guide RNAs across nucleases and technologies.” *bioRxiv*.

### ▼ 01 de febrero, 2023

#### Modelos estadísticos en R

Función `model.matrix`: ( $Y \sim X1 + X2$ )

```
## model.matrix
mat <- with(trees, model.matrix(log(Volume) ~ log(Height) + log(Girth)))
mat
```

#### SummarizedExperiment

```
## Cargamos la librería
library("SummarizedExperiment")
# Creamos los datos para nuestro objeto de tipo SummarizedExperiment
# para 200 genes a lo largo de 6 muestras
nrows <- 200
```

```
ncols <- 6
# Números al azar de cuentas
set.seed(20210223)

counts <- matrix(runif(nrows * ncols, 1, 1e4), nrows)

## Información de nuestros genes
rowRanges <- GRanges(
  rep(c("chr1", "chr2"), c(50, 150)),
  IRanges(floor(runif(200, 1e5, 1e6)), width = 100),
  strand = sample(c("+", "-"), 200, TRUE),
  feature_id = sprintf("ID%03d", 1:200)
)
names(rowRanges) <- paste0("gene_", seq_len(length(rowRanges)))

## Información de nuestras muestras
colData <- DataFrame(
  Treatment = rep(c("ChIP", "Input"), 3),
  row.names = LETTERS[1:6]
)

## Juntamos ahora toda la información en un solo objeto de R
rse <- SummarizedExperiment(
  assays = SimpleList(counts = counts),
  rowRanges = rowRanges,
  colData = colData
)
## Exploremos el objeto resultanterse
rse
```

## Ejercicio

Explica que sucede en las siguientes líneas de código de R.

```
## Comando 1
rse[1:2, ]
## Comando 2
rse[, c("A", "D", "F")]
```

Con el comando 1 se muestran el primer y segundo renglón del objeto y todas las columnas

Con el comando 2 se imprimen todos los renglones del objeto y las columnas A,D y F

## iSEE

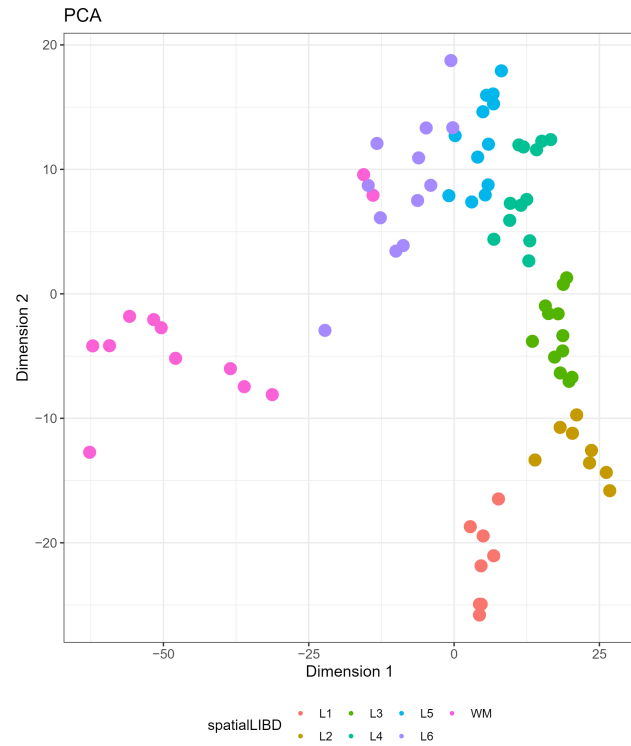
```
# Explora el objeto rse de forma interactiva
library("iSEE")
iSEE::iSEE(rse)
```

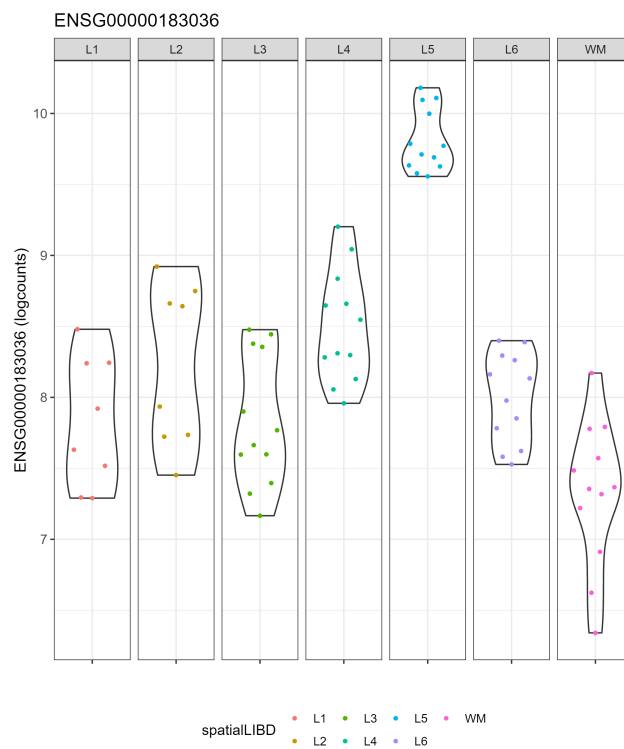
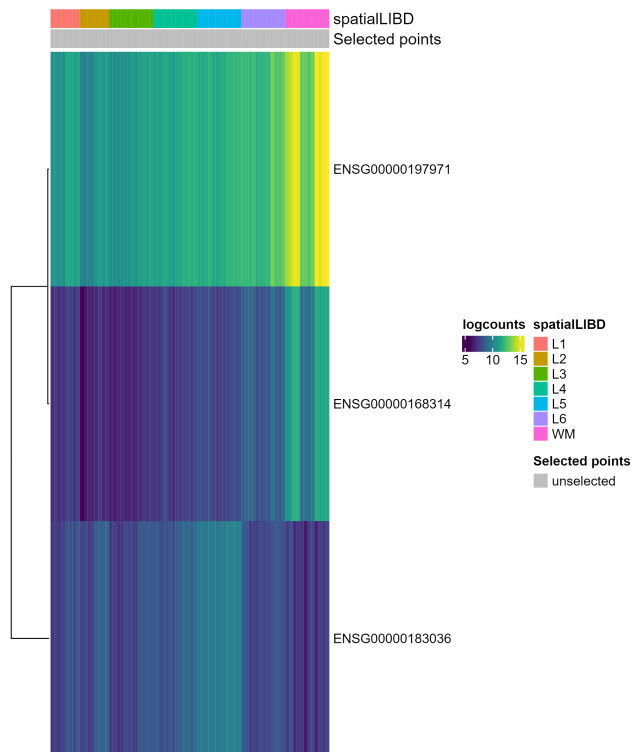
## Descargar datos con spatialLIBD

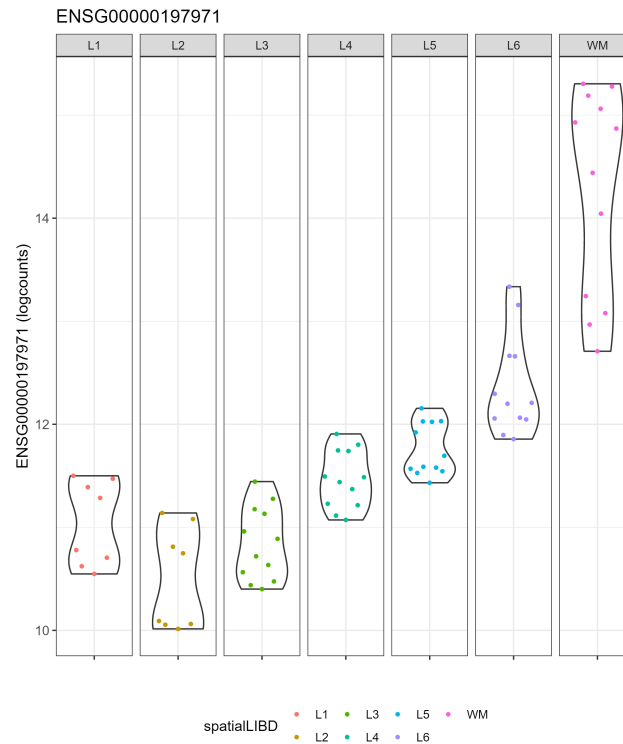
```
# Descarguemos unos datos de spatialLIBD
sce_layer <- spatialLIBD::fetch_data("sce_layer")
```

```
# Al igual que nuestro objeto rse podemos usar iSEE::iSEE() para explorar los datos.  
iSEE::iSEE(sce_layer)
```

## Ejercicio: imágenes con interfaz gráfica de iSEE







## Recount3 (Rna-seq)

Primero cargamos el paquete de R que automáticamente carga todas las dependencias incluyendo a SummarizedExperiment.

```
## Load recount3 R package
library("recount3")
```

Después tenemos que identificar un estudio de interés y determinar si queremos acceder la información a nivel de genes, exones, etc. Sabiendo el estudio de interés, podemos descargar los datos usando la función `create_rse()` como mostramos a continuación. `create_rse()` tiene argumentos con los cuales podemos especificar la anotación que queremos usar (las opciones dependen del organismo).

### Revisamos todos los proyectos con datos de humano en recount3

```
human_projects <- available_projects()

# Encuentra tu proyecto de interés. Aquí usaremos
## SRP009615 de ejemplo
proj_info <- subset(
```

```

human_projects,
project == "SRP009615" & project_type == "data_sources"
)

## Crea un objeto de tipo RangedSummarizedExperiment (RSE)
## con la información a nivel de genes
rse_gene_SRP009615 <- create_rse(proj_info)

## Explora el objeto RSE
rse_gene_SRP009615


```

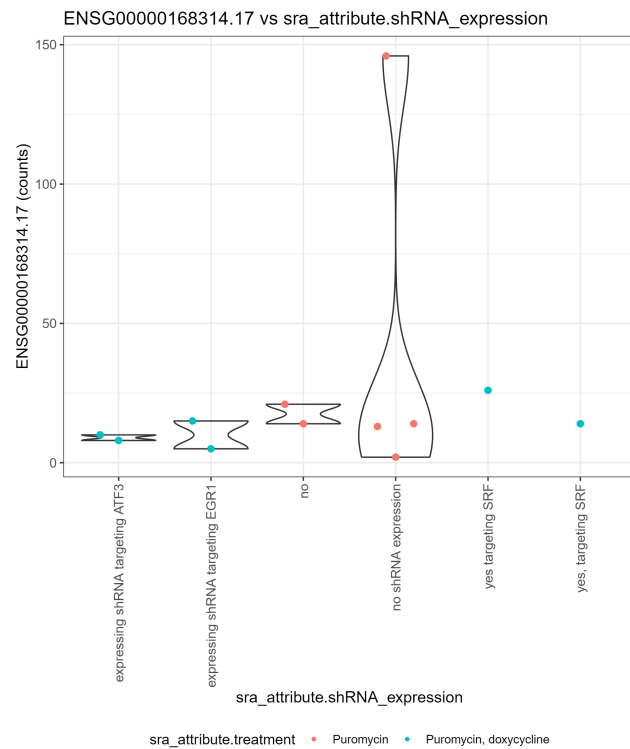
## Ejercicio

Utiliza iSEE para reproducir la imagen encontrada en

### 4 Datos de RNA-seq a través de recount3 | Intro RNA-seq LCG-UNAM 2023

Check the original documentation in English [here](#) and [here](#). Primero cargamos el paquete de R que automáticamente carga todas las dependencias incluyendo a SummarizedExperiment. Después tenemos que identificar un estudio de interés y determinar si queremos acceder a la información a nivel de genes, exones, etc.

 [https://colladotor.github.io/maseq\\_LCG-UNAM\\_2023/datos-de-ma-seq-a-trav%C3%A9s-de-recount3.html#usar-recount3](https://colladotor.github.io/maseq_LCG-UNAM_2023/datos-de-ma-seq-a-trav%C3%A9s-de-recount3.html#usar-recount3)



\*Uso de with() para definir un space

## Regresiones lineales



Intercept: valor de 'y' cuando 'x' vale 0

```
### Summary ###
summary(lm(log(Volume) ~ log(Height) + log(Girth), data = trees))

## Call:
## lm(formula = log(Volume) ~ log(Height) + log(Girth), data = trees)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.168561 -0.048488  0.002431  0.063637  0.129223
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.63162     0.79979  -8.292 5.06e-09 ***
## log(Height)  1.11712     0.20444   5.464 7.81e-06 ***
## log(Girth)   1.98265     0.07501  26.432 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08139 on 28 degrees of freedom
## Multiple R-squared:  0.9777, Adjusted R-squared:  0.9761
## F-statistic: 613.2 on 2 and 28 DF,  p-value: < 2.2e-16
```

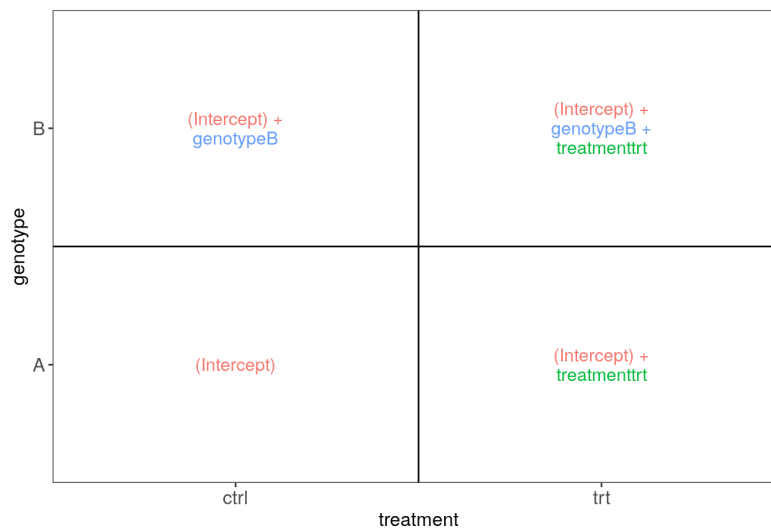
## ExploreModelMatrix

Es un paquete de Bioconductor que nos ayuda a entender los modelos estadísticos que estamos usando gracias a visualizaciones

## Creando imágenes con ExploreModelMatrix

```
# Creemos las imágenes usando ExploreModelMatrix
vd <- ExploreModelMatrix::VisualizeDesign(
  sampleData = sampleData,
  designFormula = ~ genotype + treatment,
  textSizeFitted = 4
)

# Veamos las imágenes
cowplot::plot_grid(plotlist = vd$plotlist)
```



### \*Control vs tratamiento → niveles de expresión:

ej. coef. 10 = tratamiento tiene 10 veces más expresión que control

ej. coef. -5 = control tiene 5 veces más expresión que tratamiento

### Interfaz gráfica para ExploreModelMatrix:

```
## Es posible usar shiny para usar ExploreModelMatrix
app <- ExploreModelMatrix(
  sampleData = sampleData,
  designFormula = ~ genotype + treatment
)
if (interactive()) shiny::runApp(app)
```

\*~ 0 + se usa para no tener intercepto (no quieres algo específico como referencia)!!!

## Ejercicio

- Interpreta `ResponseResistant.Treatmentpre` del ejercicio 2. Puede ser útil tomar un *screenshot* (captura de pantalla) y anotarla con líneas de colores. Si haces eso, puedes incluir la imagen en tus notas.

Response = Resistant		1	2
ind.n	6	(Intercept) + ResponseResistant:ind.n6	(Intercept) + ResponseResistant:ind.n6 + ResponseResistant:Treatmentpre
	5	(Intercept) + ResponseResistant:ind.n5	(Intercept) + ResponseResistant:ind.n5 + ResponseResistant:Treatmentpre
	4	(Intercept) + ResponseResistant:ind.n4	(Intercept) + ResponseResistant:ind.n4 + ResponseResistant:Treatmentpre
	3	(Intercept) + ResponseResistant:ind.n3	(Intercept) + ResponseResistant:ind.n3 + ResponseResistant:Treatmentpre
	2	(Intercept) + ResponseResistant:ind.n2	(Intercept) + ResponseResistant:ind.n2 + ResponseResistant:Treatmentpre
	1	(Intercept)	(Intercept) + ResponseResistant:Treatmentpre
		post	pre
		Treatment	

$C2 - C1$

Columna 2 (pre) - columna 1 (post)

Si tuvieramos un coef. 7:

la columna 1 (post) es 7 veces mayor a la columna 2 (pre)

\*cuando el grupo es ResponseResistant:Treatmentpre

- ¿Por qué es clave el 0 al inicio de la fórmula en el ejercicio 3?

Para no tener intercepto

		batch6	batch6 + conditionko_plus
5	batch5 batch5 + conditionctrl_plus		
4		batch4	batch4 + conditionko_plus
3	batch3 batch3 + conditionctrl_plus		
2		batch2	batch2 + conditionko_plus
1	batch1 batch1 + conditionctrl_plus		
	ctrl_minus      ctrl_plus      ko_minus      ko_plus	condition	

En caso de tener intercepto, 'batch 1' se tomaría como referencia y no lo veríamos en los cuadritos

### Para aprender más

*A guide to creating design matrices for gene expression experiments:*

- <http://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/designmatrices.html>
- <https://f1000research.com/articles/9-1444>

### Datos de SRP045638 con recount3

Vamos a usar datos de <https://www.ncbi.nlm.nih.gov/sra/?term=SRP045638>  
procesados con recount 3

### Limpieza de datos:

- `edgeR::filterByExpr()` <https://bioconductor.org/packages/edgeR/>  
<https://rdrr.io/bioc/edgeR/man/filterByExpr.html>
- `genefilter::genefilter()` <https://bioconductor.org/packages/genefilter/>  
<https://rdrr.io/bioc/genefilter/man/genefilter.html>
- `jaffelab::expression_cutoff()` [http://research.libd.org/jaffelab/reference/expression\\_cutoff.html#](http://research.libd.org/jaffelab/reference/expression_cutoff.html#)

### Normalización de datos (edgeR y calcNormFactors)

La normalización pretende eliminar los sesgos en los datos, generalmente se multiplica por un factor (de 2)

```
library("edgeR") # BiocManager::install("edgeR", update = FALSE)
dge <- DGEList(
  counts = assay(rse_gene_SRP045638, "counts"),
```

```
genes = rowData(rse_gene_SRP045638)
)
dge <- calcNormFactors(dge)
```

## Análisis de expresión diferencial (limma)

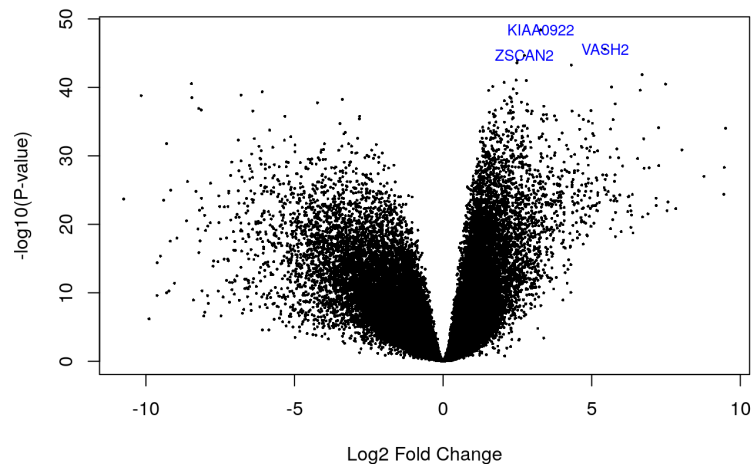
Con algunos paquetes como variancePartition y scater podemos explorar nuestros set de datos

Por ahora continuaremos con el siguiente modelo estadístico.

```
mod <- model.matrix(~ prenatal + sra_attribute.RIN + sra_attribute.sex + assigned_gene_prop,
  data = colData(rse_gene_SRP045638)
)
colnames(mod)
```

Para realizar el análisis de expresión diferencial como tal podemos usar limma:

```
library("limma")
vGene <- voom(dge, mod, plot = TRUE)
```



En la imagen tipo volcano, se convierten valores de p-value pequeños a un valor positivo y grande (-log10)

## Visualizando genes DE

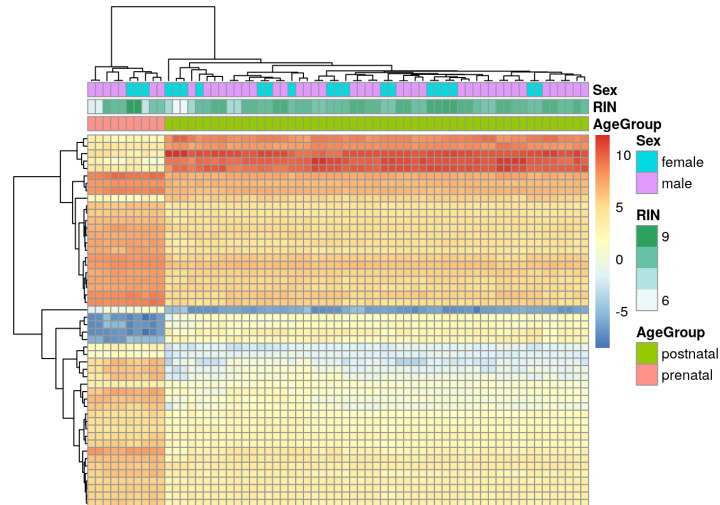
De vGene\$E podemos extraer los datos normalizados por limma-voom

```
## Extraer valores de los genes de interés
exprs_heatmap <- vGene$E[rank(de_results$adj.P.Val) <= 50, ]
```

```
## Creemos una tabla con información de las muestras
df <- as.data.frame(colData(rse_gene_SRP045638)[, c("prenatal", "sra_attribute.RIN", "sra_attribute.sex")])
```

```
colnames(df) <- c("AgeGroup", "RIN", "Sex")

## Hagamos un heatmaplibrary("pheatmap")
pheatmap(
  exprs_heatmap,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  show_rownames = FALSE,
  show_colnames = FALSE,
  annotation_col = df
)
```



Al tener una gran diferencia de los perfiles de expresión en el DLPFC entre muestra pre y post-natales podríamos verlo con PCM o **MDS (multidimensional scaling)** como a continuación.

Workflow:

```
## Para colores
library("RColorBrewer")

## Conviertiendo los grupos de edad a colores
col.group <- df$AgeGroup
levels(col.group) <- brewer.pal(nlevels(col.group), "Set1")

col.group <- as.character(col.group)

## MDS por grupos de edad
plotMDS(vGene$E, labels = df$AgeGroup, col = col.group)
```

## Ejercicio

Agreguen los nombres de los genes a nuestro pheatmap

Pistas:

- Revisen la información de `rowRanges(rse_gene_SRP045638)` o de `results`.
- Exploren que hace la función `match()`.

