

PEC2 Análisis de Datos Ómicos

Rita Ortega Vallbona

8 de junio, 2020

Contents

| | |
|---|-----------|
| 1 Abstract | 1 |
| 2 Objetivos | 1 |
| 3 Materiales y métodos | 1 |
| 3.1 Los Datos | 1 |
| 3.2 Preprocesado de los datos | 2 |
| 3.2.1 Normalización | 12 |
| 3.3 Identificación de genes diferencialmente expresados | 12 |
| 3.4 Anotación de los resultados | 12 |
| 3.5 Busca de patrones de expresión y agrupación de las muestras (comparación entre las distintas comparaciones) | 12 |
| 3.6 Análisis de significación biológica (“ <i>Gene Enrichment Analysis</i> ”) | 12 |
| 4 Resultados | 12 |
| 5 Discusión | 12 |
| 6 Apéndice | 12 |
| Bibliografía | 12 |

1 Abstract

2 Objetivos

3 Materiales y métodos

3.1 Los Datos

Los datos proporcionados en el enunciado provienen del repositorio GTEx (*Genotype-Tissue Expression*), que recoge información de expresión específica de 54 tipos de tejido sano, proveniente de 1000 individuos

(“GTEx Portal,” n.d.). Este **portal** permite el acceso a los datos de expresión, imágenes de histología, etc. Obtenemos los datos de targets y counts de los archivos csv proporcionados en el enunciado: targets.csv y counts.csv.

Estos son datos de expresión (RNA-seq) pertenecientes a un análisis del tiroides, donde se comparan tres tipos de infiltración en 292 muestras:

- *Not infiltrated tissues (NIT)*: 236 muestras
- *Small focal infiltrates (SFI)*: 42 muestras
- *Extensive lymphoid infiltrates (ELI)*: 14 muestras

Con este script extraemos 10 muestras de cada grupo del archivo targets.csv y subsetteamos las columnas escogidas en el archivo counts.csv:

```
> # Separamos el dataframe que recoge los targets por grupos
> NIT <- subset(all_targets, Group == "NIT")
> SFI <- subset(all_targets, Group == "SFI")
> ELI <- subset(all_targets, Group == "ELI")
>
> # Seleccionamos 10 muestras de cada grupo y las unimos en un
> # único dataframe que recoge los targets con los que
> # trabajaremos
> set.seed(params$seed.extract)
> NIT10 <- NIT[sample(nrow(NIT), size = 10, replace = FALSE), ]
> SFI10 <- SFI[sample(nrow(SFI), size = 10, replace = FALSE), ]
> ELI10 <- ELI[sample(nrow(ELI), size = 10, replace = FALSE), ]
>
> mytargets <- rbind(NIT10, SFI10, ELI10, deparse.level = 0)
>
> # Extraemos los nombres de las muestras y cambiamos los
> # guiones por puntos para que coincidan con los nombres de
> # las muestras en el dataframe de counts
> sample_names <- mytargets[, 3]
> s_names <- gsub("-", ".", sample_names)
>
> # Subsetteamos las columnas escogidas del dataframe de counts
> mycounts <- dplyr::select(all_counts, s_names)
> row.names(mycounts) <- all_counts$X
```

De este modo hemos obtenido dos datasets: **mytargets** que recoge los detalles de cada una de las 30 muestras con las que vamos a trabajar, y **mycounts**, que representa la tabla de contajes de estas 30 muestras.

3.2 Preprocesado de los datos

Para poder identificar los tipos de muestra con mayor facilidad, procedemos a renombrar las columnas de **mycounts** con nombres cortos indicativos de a qué grupo pertenecen, y también lo asignamos a **mytargets**.

```
> newShortNames <- c(paste0("NIT", 1:10), paste0("SFI", 1:10),
+   paste0("ELI", 1:10))
> mytargets <- cbind(mytargets, newShortNames)
>
> # Asignamos los nuevos nombres cortos a las columnas de
```

```
> # mycounts
> colnames(mycounts) <- mytargets$newShortNames
```

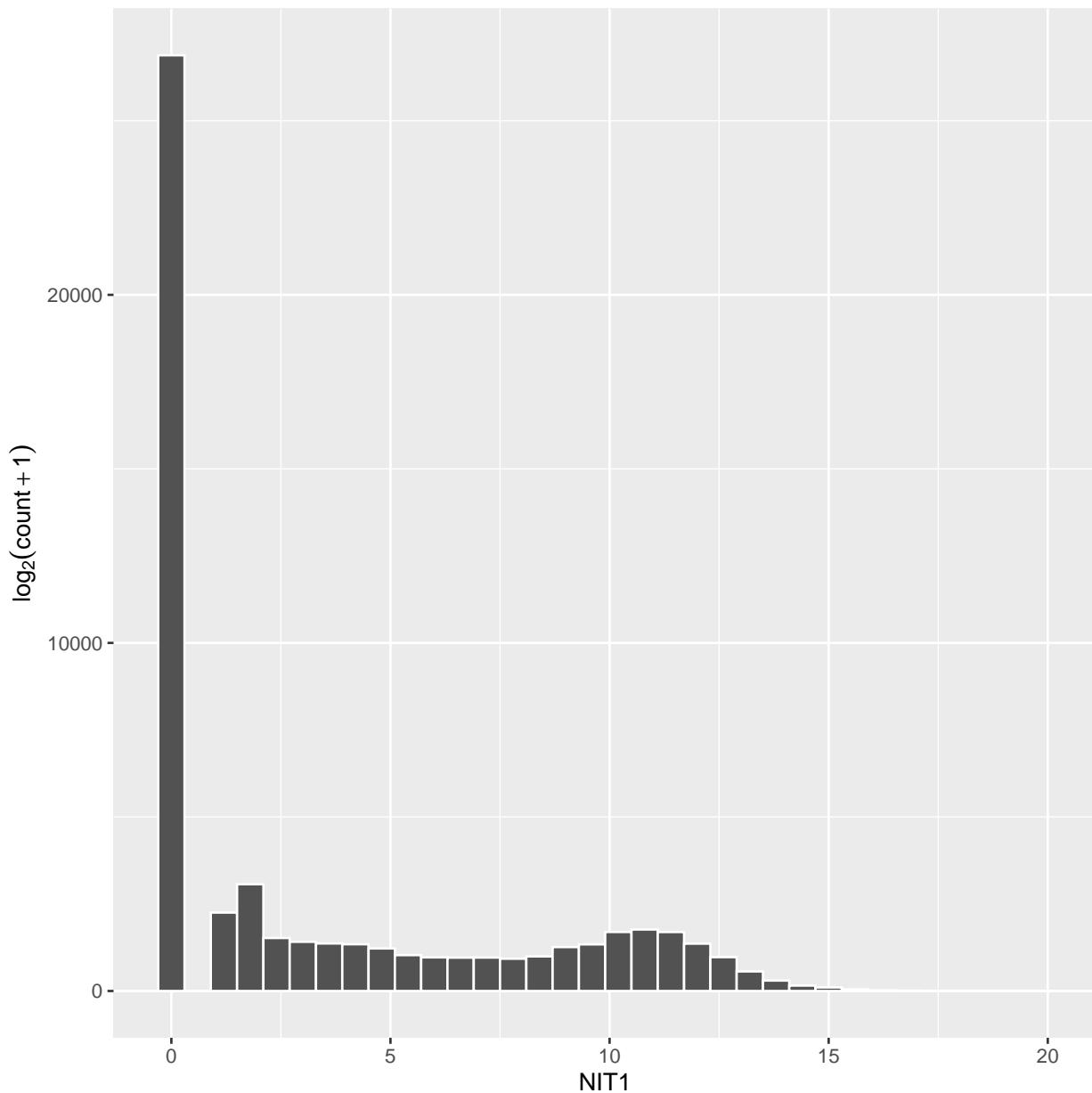
De este modo, nuestros datos de conteo quedan así:

```
> head(mycounts)
```

| | NIT1 | NIT2 | NIT3 | NIT4 | NIT5 | NIT6 | NIT7 | NIT8 | NIT9 | NIT10 | SFI1 | SFI2 | S |
|-------------------|------|------|------|------|------|------|------|------|------|-------|------|------|---|
| ENSG00000223972.4 | 1 | 0 | 2 | 1 | 2 | 0 | 4 | 4 | 5 | 3 | 5 | 3 | |
| ENSG00000227232.4 | 584 | 1487 | 412 | 800 | 542 | 1325 | 513 | 852 | 1034 | 487 | 656 | 533 | |
| ENSG00000243485.2 | 0 | 1 | 0 | 2 | 1 | 2 | 9 | 4 | 5 | 1 | 1 | 1 | |
| ENSG00000237613.2 | 0 | 0 | 1 | 2 | 0 | 0 | 10 | 4 | 5 | 2 | 1 | 0 | |
| ENSG00000268020.2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | |
| ENSG00000240361.1 | 1 | 2 | 0 | 1 | 0 | 0 | 4 | 3 | 3 | 0 | 1 | 0 | |

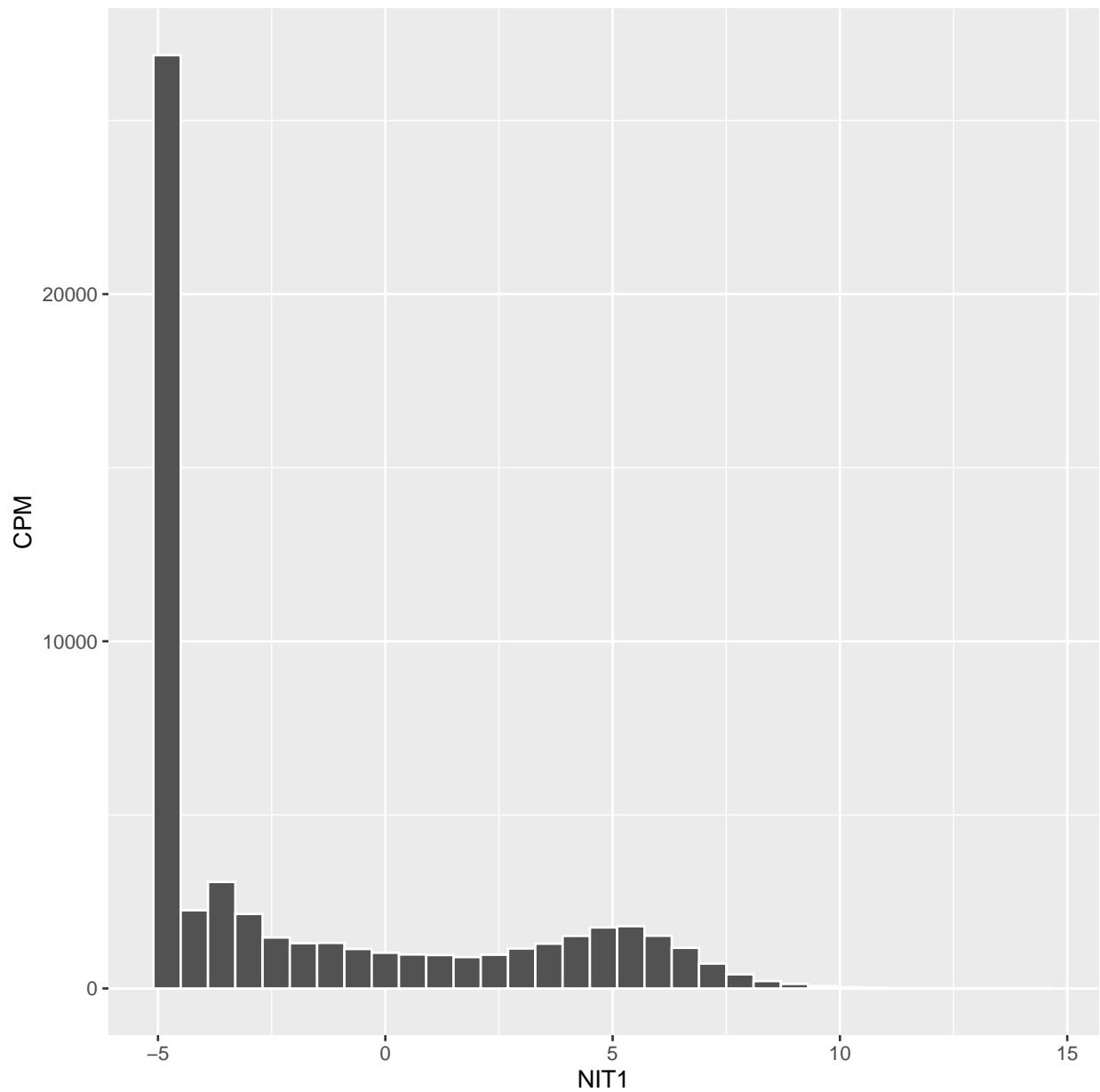
Antes de proceder con el análisis de los datos, debemos evaluar la calidad de los datos crudos con los que vamos a trabajar, para detectar cualquier fallo técnico que pueda afectar a la interpretación de los datos. Analizaremos la calidad de los datos con representaciones gráficas, y dado que los datos resultantes de conteos suelen estar altamente sesgados, aplicaremos previamente una transformación logarítmica en base 2 para “normalizar” los datos. Además hemos de tener en cuenta que algunos de los conteos resultan en 0, para que ello no suponga un problema tomaremos *pseudocounts*, sumando una constante en todos los conteos, para que no den problema en la transformación logarítmica.

```
> pseudoCounts <- log2(mycounts + 1)
>
> library(ggplot2)
>
> ggplot(pseudoCounts, aes(x = pseudoCounts[, 1])) + ylab(expression(log[2](count +
+      1))) + xlab(names(pseudoCounts)[1]) + geom_histogram(colour = "white",
+      fill = "#525252", binwidth = 0.6)
```



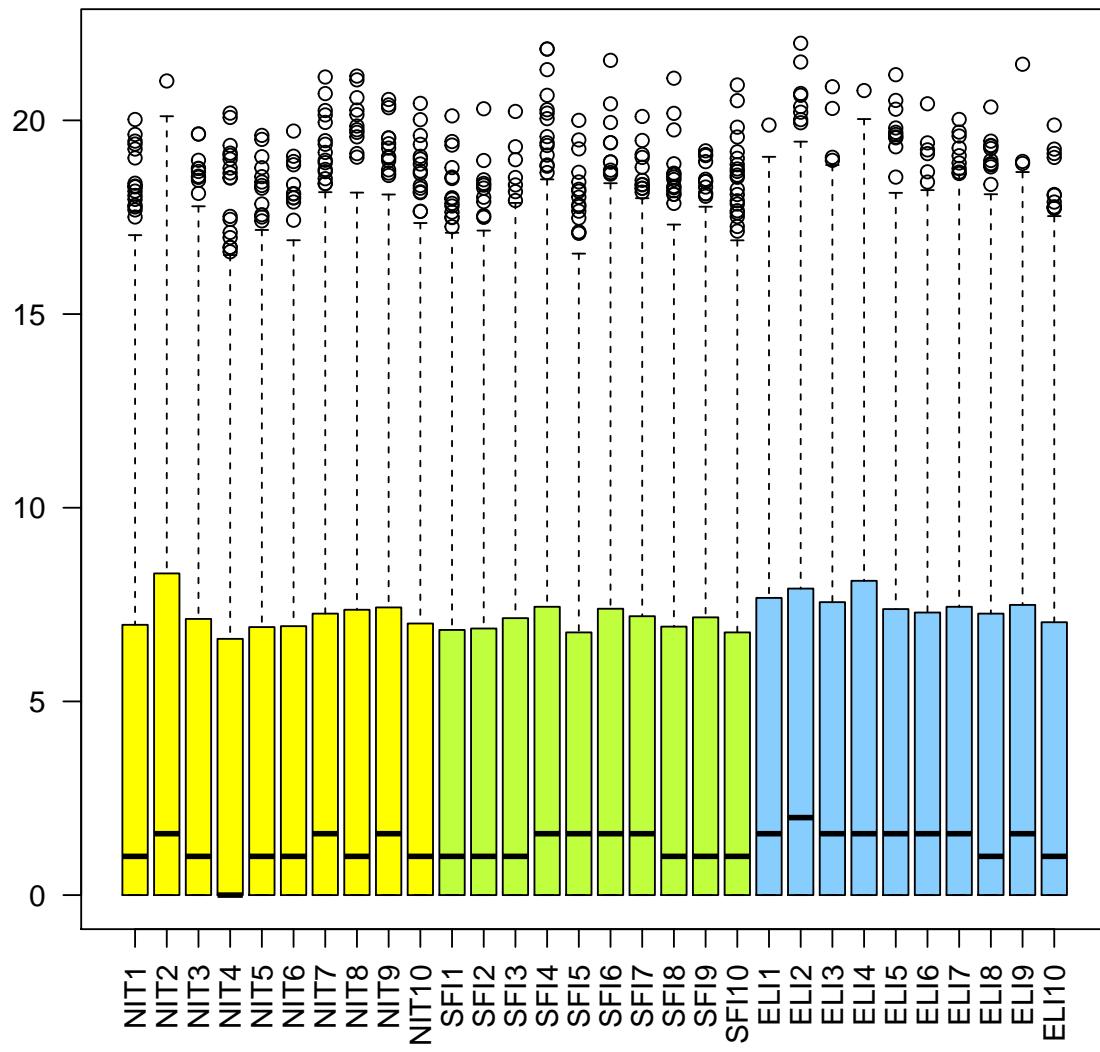
Esta transformación logarítmica podría no ser suficiente para evaluar realmente cómo se distribuyen los datos, ya que los datos procedentes de RNA-seq no son homocedásticos, sino que su varianza aumenta según aumenta la media. Intentamos utilizar una función especializada del paquete `edgeR` para hacer una transformación de los datos y comparamos los resultados:

```
> library(edgeR)
> pseudoCounts2 <- cpm(mycounts, prior.count = 2, log = TRUE)
>
> library(ggplot2)
> ggplot(pseudoCounts2, aes(x = pseudoCounts2[, 1])) + ylab("CPM") +
+   xlab(names(mycounts)[1]) + geom_histogram(colour = "white",
+   fill = "#525252", binwidth = 0.6)
```

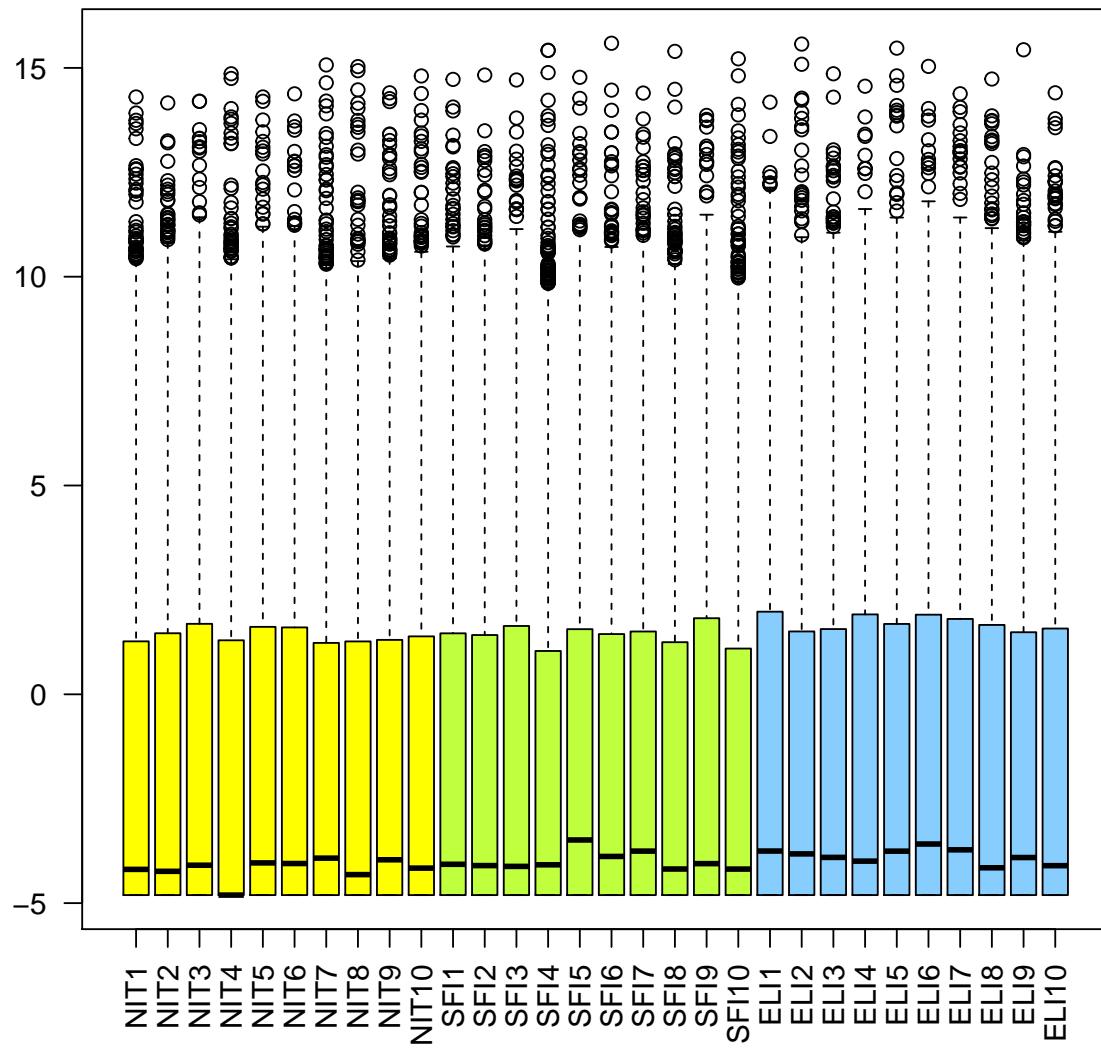


Para poder visualizar la distribución de los datos de las 30 muestras con las que trabajamos, las representamos en forma de boxplot:

Distribución de contajes de pseudoCounts



Distribución de contajes de pseudoCounts2 (con edgeR)



Representamos los datos en gráficos de densidades para detectar si pudiera haber modos secundarios de la distribución de los datos:

Para comprobar la reproducibilidad de las muestras, representamos los MA-plots para comparar muestras entre sí:

```
> myMAplot <- function(dataset, numberX, numberY) {
+   data <- dataset
+
+   nameX <- names(data)[numberX]
+   nameY <- names(data)[numberY]
+
+   x <- data[, numberX]
+   y <- data[, numberY]
```

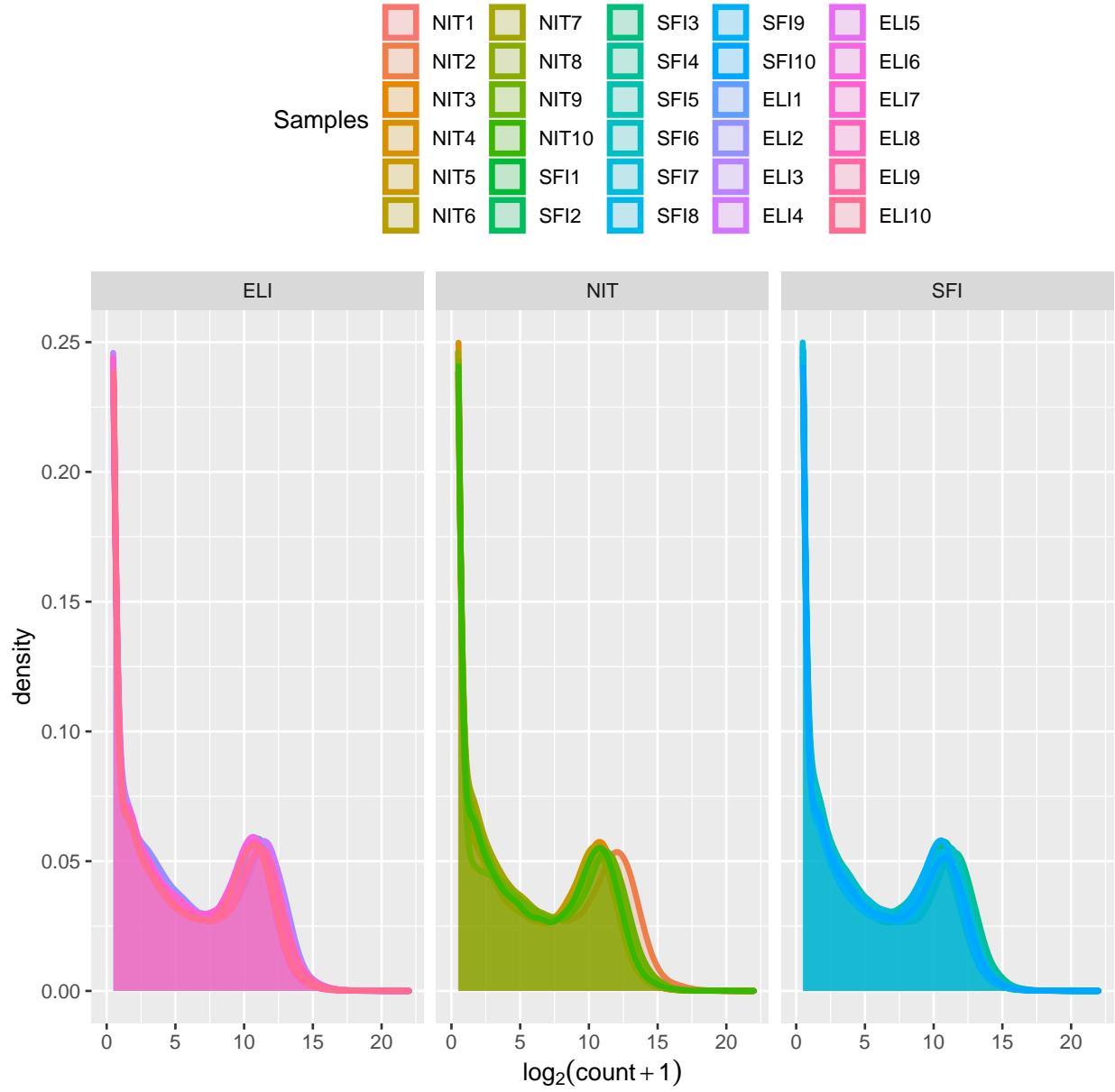
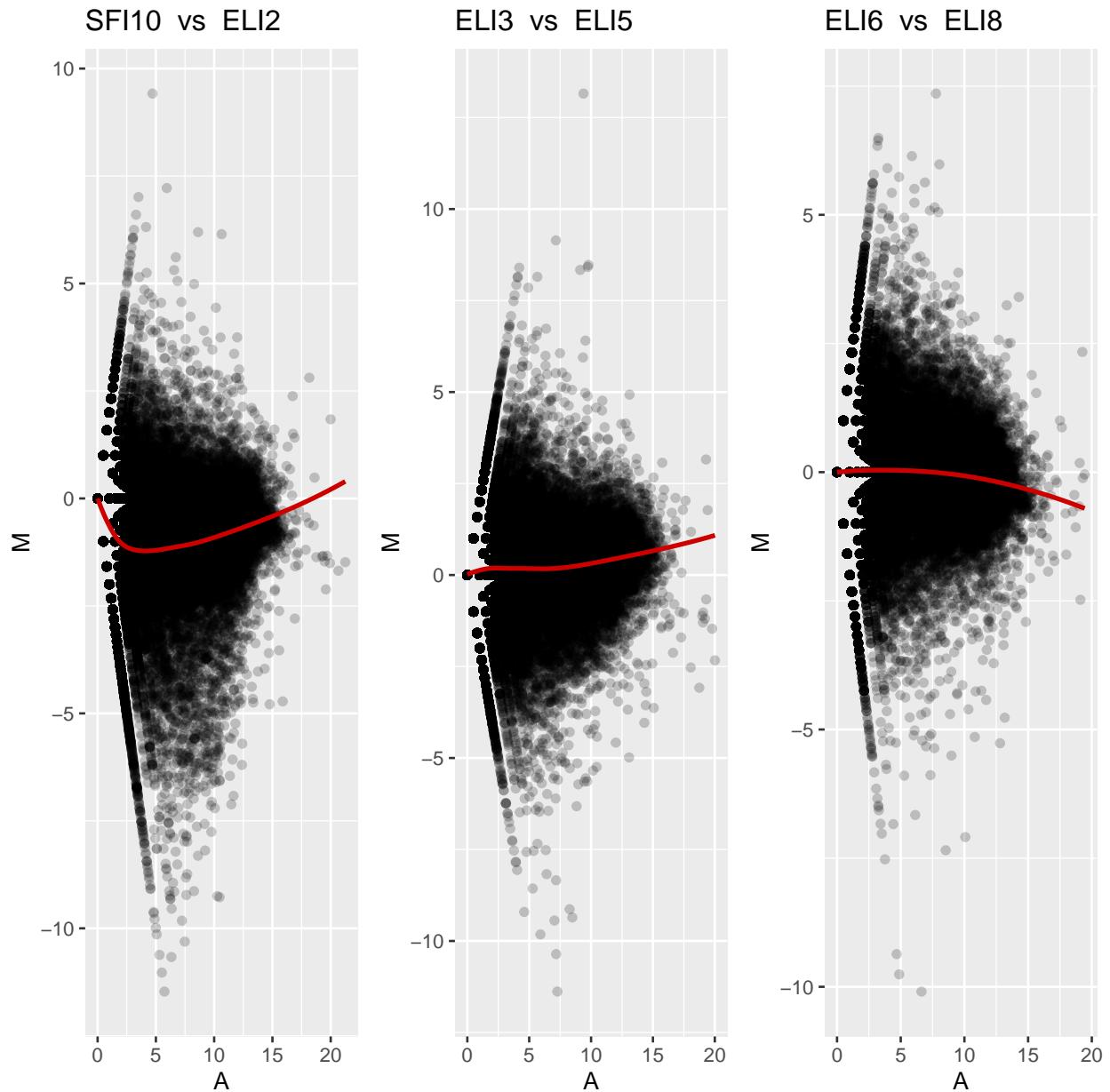


Figure 1: Gráficos de densidades que muestra las densidades empíricas de las muestras individuales dentro de cada una de las tres condiciones experimentales

```

+   M = x - y
+   A = (x + y)/2
+
+   dfMA <- data.frame(A, M)
+
+   library(ggplot2)
+   ggplot(dfMA, aes(x = A, y = M)) + geom_point(size = 1.5,
+       alpha = 1/5) + stat_smooth(se = F, method = "loess",
+       color = "red3") + ggttitle(paste(nameX, " vs ", nameY))
+
}

```



Filtraje

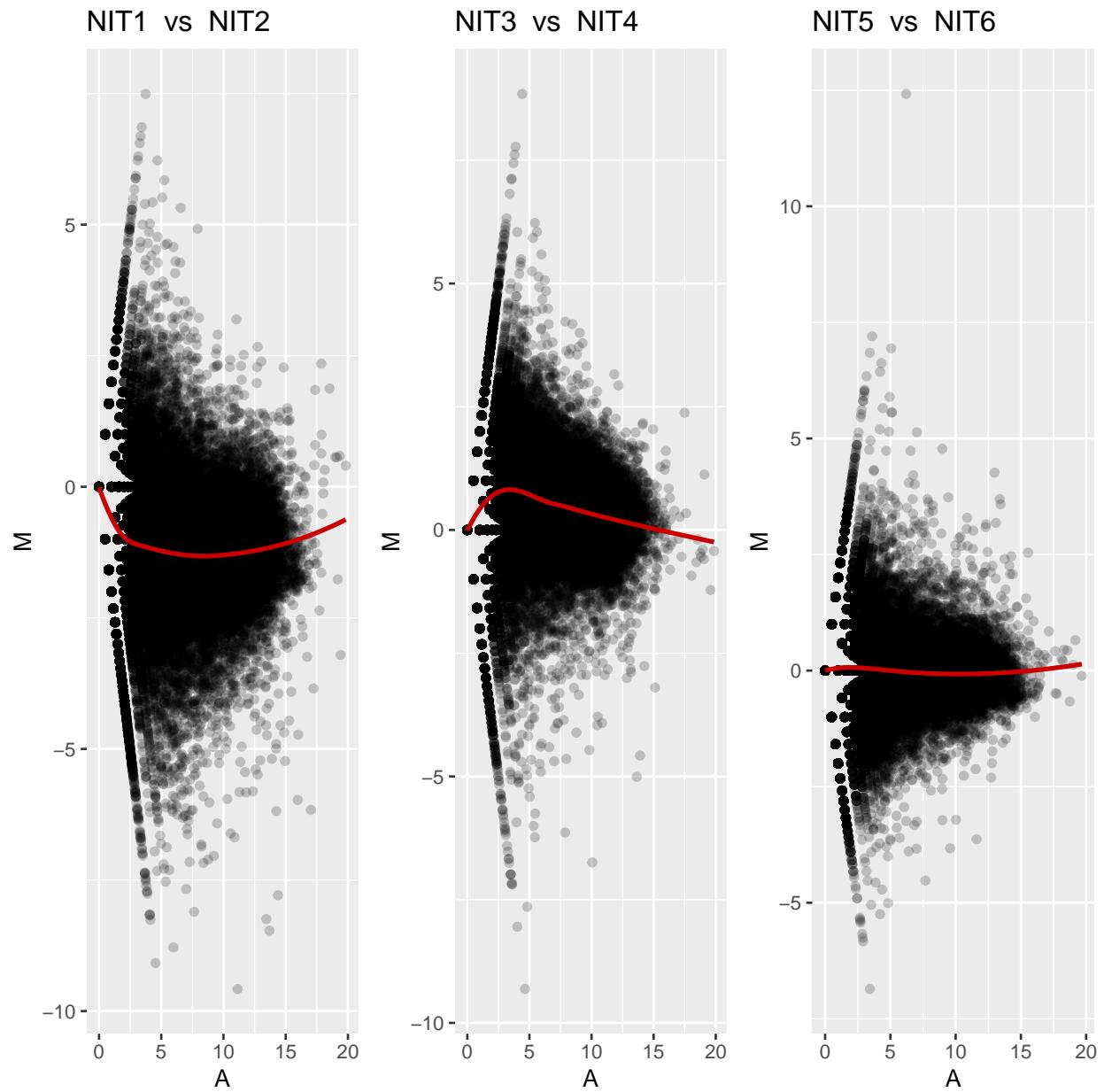


Figure 2: Representación MA de tres comparaciones entre muestras del grupo NIT

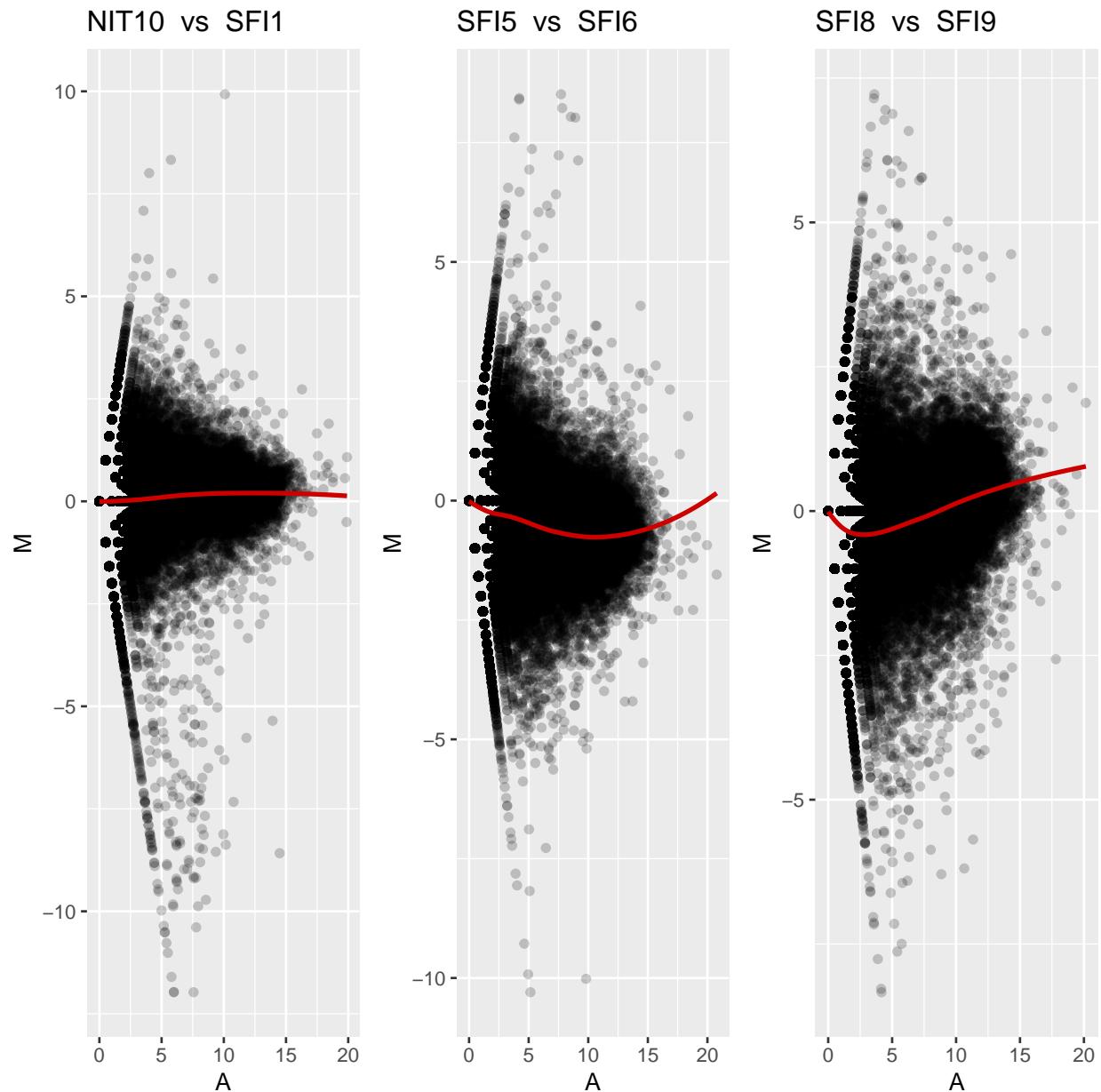


Figure 3: Representación MA de tres comparaciones entre muestras del grupo SFI

- 3.2.1 Normalización**
- 3.3 Identificación de genes diferencialmente expresados**
- 3.4 Anotación de los resultados**
- 3.5 Busca de patrones de expresión y agrupación de las muestras (comparación entre las distintas comparaciones)**
- 3.6 Análisis de significación biológica (“*Gene Enrichment Analysis*”)**

4 Resultados

5 Discusión

6 Apéndice

Bibliografía

“GTEx Portal.” n.d. <https://www.gtexportal.org/home/>.