RAMON_ARRUFAT_ALBERT_PEC2

Albert Ramon Arrufat

20/12/2024

Contenidos

Todos los archivos y código creados en este informe se encuentran en un repositorio Github. Para acceder a él, acceder al siguiente link:

https://github.com/aramonarr/RAMON-ARRUFAT-ALBERT-PEC2

1. Presentación

La PEC se basará en los datos de un estudio que, utilizando un modelo murino (de ratón) investigamos la utilidad de los antibióticos LINEZOLID y VANCOMICINA para inmunomodulación durante infecciones por Staphylococcus aureus resistente a meticilina (MRSA).

El dataset consta de 35 muestras, 15 tomadas antes de la infección y 20 después, 5, que eliminaremos, a las 2 horas de la misma y 15 a las 24 horas.

Mi solución de la PEC tiene dos objetivos principales:

1. Mostrar un modelo de resolución de las preguntas planteadas que sirva como ejemplo.
2. Incluir explicaciones sobre el desarrollo del trabajo, siguiendo las instrucciones de la PEC.

2. Objetivos

El objetivo principal de este trabajo es realizar un análisis exploratorio y de control de calidad sobre datos de microarrays de unos datos descargados de la base de datos [Gene Expression Omnibus(GEO)] (https://www.ncbi.nlm.nih.gov/geo/) utilizando el programa estadístico R y las librerías para análisis de datos ómicos integradas en Bioconductor.

En concreto se requiere que los datos descargados se almacenen en un objeto de clase `***ExpressionSet***` y que se acceda a ellos a través de este objeto para poder caracterizar, a través del cambio en la expresión génica, el efecto de la infección y del tratamiento con antibióticos, así como comparar los efectos de éstos. Es decir, haremos las comparaciones siguientes:

- Infectados vs no infectados sin tratamiento
- Infectados vs no infectados tratados con LINEZOLID
- Infectados vs no infectados tratados con VANCOMICINA

3. Materiales y Métodos

3.1.- Fuentes de Datos

Los datos se obtuvieron de la base [Gene Expression Omnibus (GEO)] (https://www.ncbi.nlm.nih.gov/geo/) utilizando el identificador **GSE38531**.

Este paquete permite descargar los datos indicados (por un identificador GSE o GDS) y crear con ellos una estructura de datos del tipo `***expressionSet***` que contiene una matriz de datos preprocesados (habitualmente normalizados, así como una tabla con información sobre covariables y tras aspectos del experimento.

3.2- Metodología

La exploración se llevará a cabo siguiendo la plantilla de un caso de estudio, [Analisis_de_datos_omicos-Ejemplo_0-Microarrays] (https://github.com/ASPteaching/Analisis_de_datos_omicos-Ejemplo_0-Microarrays) del profesor Alex Sánches Pla.

Básicamente dicha exploración consistirá en:

1. **Control de calidad:** Evaluación inicial con el paquete `*arrayQualityMetrics*` para identificar posibles problemas y evaluar la calidad de las muestras.

2. **Normalización:** Conversión de los datos crudos a una matriz de expresión utilizando el algoritmo RMA (Robust Multi-array Average).

3. **Filtrado:** Selección de las sondas con mayor variabilidad para centrarse en las más informativas.

4. **Análisis diferencial:** Creación de matrices de diseño y contrastes con `*limma*` para realizar las tres comparaciones propuestas en el estudio.

5. **Anotación y análisis funcional:** Uso de `*clusterProfiler*` para enriquecer los resultados biológicos y ver la asociación de identificadores como "Symbol", "EntrezID" o "EnsemblID" a los genes identificados.

6. **Análisis funcional:** Realización de análisis de sobre-representación y GSEA con el paquete `*clusterProfiler*` para interpretar la significación biológica de los resultados, complementados con visualizaciones para facilitar su interpretación.

Este enfoque garantizará la calidad y relevancia de los datos en cada etapa del análisis.

4. Resultados

```
>if (!requireNamespace("BiocManager", quietly = TRUE)) {
+   install.packages("BiocManager")
+ }
>BiocManager::install("GEOquery")
>library(GEOquery)
>library(limma)
>library(edgeR)
>library(pheatmap)
>install.packages("ggplot2")
```

Descargo los datos crudos del sitio de GEO
https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE38531

Con el fin de simplificar el análisis y dificultar un mínimo el intercambio no permitido de información eliminaremos algunas muestras:

- Por un lado, prescindiremos de las cinco muestras tomadas a las dos horas

- Por otro lado, y tal como se solicita, sortearé las muestras restantes de forma que conservamos tan sólo cuatro muestras de cada grupo. Esto lo haré con la función ***selectSamples*** que encontramos en el archivo *selectSamples.R* (descargado del material de PEC) y que nos permitirá extraer 24 muestras distintas a cada uno con tan solo llamarla usando como semilla (argumento "seed") mi número de expediente de la UOC(1418474).

```
>gse <- getGEO("GSE38531", GSEMatrix = TRUE, AnnotGPL = TRUE)
```

Extraemos los metadatos:

```
>pdata <- pData(gse[[1]])
```

Filtraremos las muestras tomadas a las 2 horas

```
>pdata$time <- as.numeric(gsub("time \\(0h/2h/24h post infection\\): ", "",
                              pdata$`time (0h/2h/24h post infection):ch1`))
```

Ahora podemos eliminar las muestras tomadas a las 2 horas

```
>pdata_filtrado <- subset(pdata, time != 2)
```

Verificamos que las muestras se eliminaron correctamente

```
>table(pdata_filtrado$time)

 0 24

15 15
```

Cargamos la función desde el archivo ***selectSamples.R*** y preparamos las columnas necesarias para el filtrado

```
>source("~/UOC/PEC2/selectSamples.R")

>pdata_filtrado$sample <- rownames(pdata_filtrado)

>pdata_filtrado$infection <- ifelse(is.na(pdata_filtrado$`pathogen strain:ch1`),
                                   "uninfected", "S. aureus USA300")

>pdata_filtrado$time <- paste0("hour ", pdata_filtrado$`time (0h/2h/24h post infection):ch1`)

>pdata_filtrado$agent <- gsub("drug treatment  \\(linezolid/vancomycin/no\\): ", "",
                              pdata_filtrado$`drug treatment  (linezolid/vancomycin/no):ch1`)
```

Aplicaremos la función para seleccionar 4 muestras por grupo

```
>set.seed(1418474)

>result <- filter_microarray(pdata_filtrado, seed = 1418474)
```

Pasamos a crear un nuevo ***ExpressionSet***, donde filtraremos la matriz de expresión original para incluir solo las muestras seleccionadas y asi poder crear un **ExpressionSet**.

Primero extraemos las filas correspondientes a las muestras seleccionadas

```
>selected_samples <- result$sample

>expression_data <- exprs(gse[[1]])[, selected_samples]
```

Creamos el nuevo ExpressionSet

```
>library(Biobase)

>nuevo_eset <- ExpressionSet(assayData = expression_data,
             phenoData = AnnotatedDataFrame(pdata_filtrado[selected_samples, ]))
```

Para poder trabajar directamente con los archivos **.CEL** en lugar del objeto descargado desde GEO, utilizaremos el paquete ***affy***. Para ello necesitamos hacer coincidir los nombres de ***selected_samples*** con los nombres de los archivos **.CEL** usando coincidencias parciales. Esto realizar mediante ***grep*** o ***grepl***.

- *sapply*: Itera sobre cada identificador en selected_samples y busca si está contenido en los nombres base de cel_files usando grepl.
- *grepl*: Realiza una coincidencia parcial, es decir, verifica si el ID está en alguna parte del nombre del archivo **.CEL**.

Extraemos los identificadores simples de los nombres seleccionados, asumiendo claro que

```
>selected_samples contiene GSM IDs como "GSM944857"

> selected_ids <- selected_samples

> cel_files <- list.files(path = "~/UOC/PEC2/GSE38531_RAW",

+                         pattern = "\\.CEL$", full.names = TRUE)
```

Filtramos archivos .CEL que contengan los IDs seleccionados

```
> cel_files_selected <- cel_files[sapply(selected_ids, function(id) {

+   any(grepl(id, basename(cel_files)))

+ })]
```

Instalamos y cargamos ***affy***

```
> if (!requireNamespace("affy", quietly = TRUE)) {

+   BiocManager::install("affy")

+ }

> library(affy)

>
```

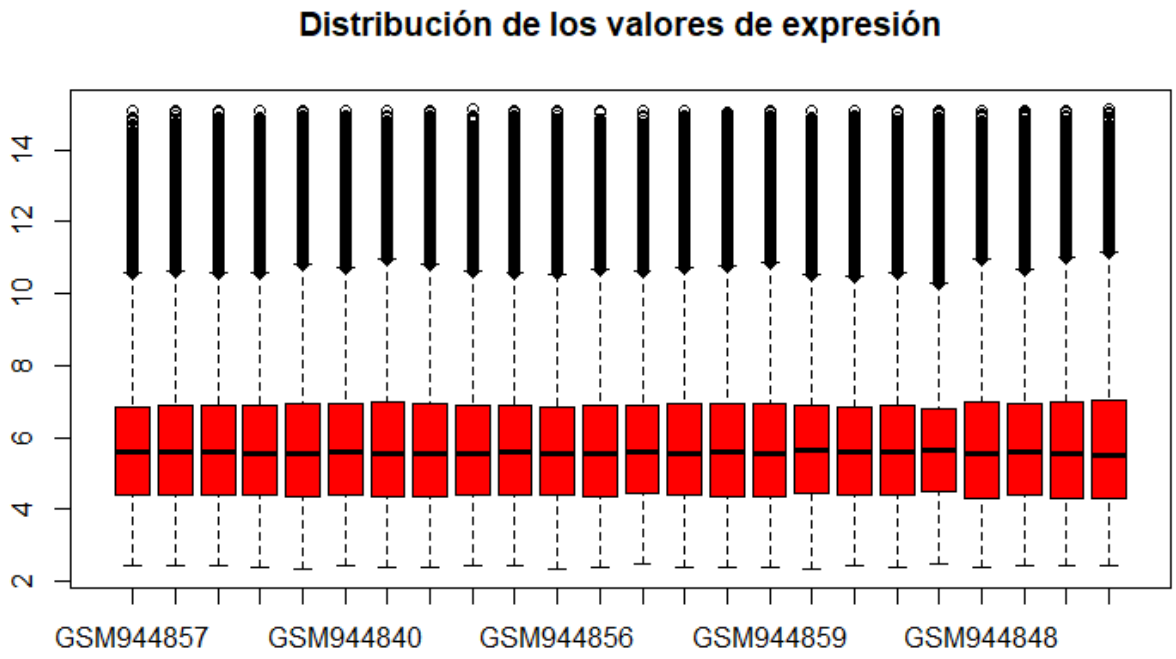
Ahora ya podemos leer los archivos .CEL de las muestras seleccionadas

```
> eset_cel <- ReadAffy(filenames = cel_files_selected)
```

4.1.- Análisis exploratorio de los datos y control de calidad

A partir de este ***ExpressionSet*** personalizado, con 24 muestras muestras podemos proceder a realizar una exploración básica. Pero antes de normalizar, exploramos los datos crudos con boxplots y densidades.

```
>boxplot(exprs(nuevo_eset), main = "Distribución de los valores de expresión", col = "red")
```

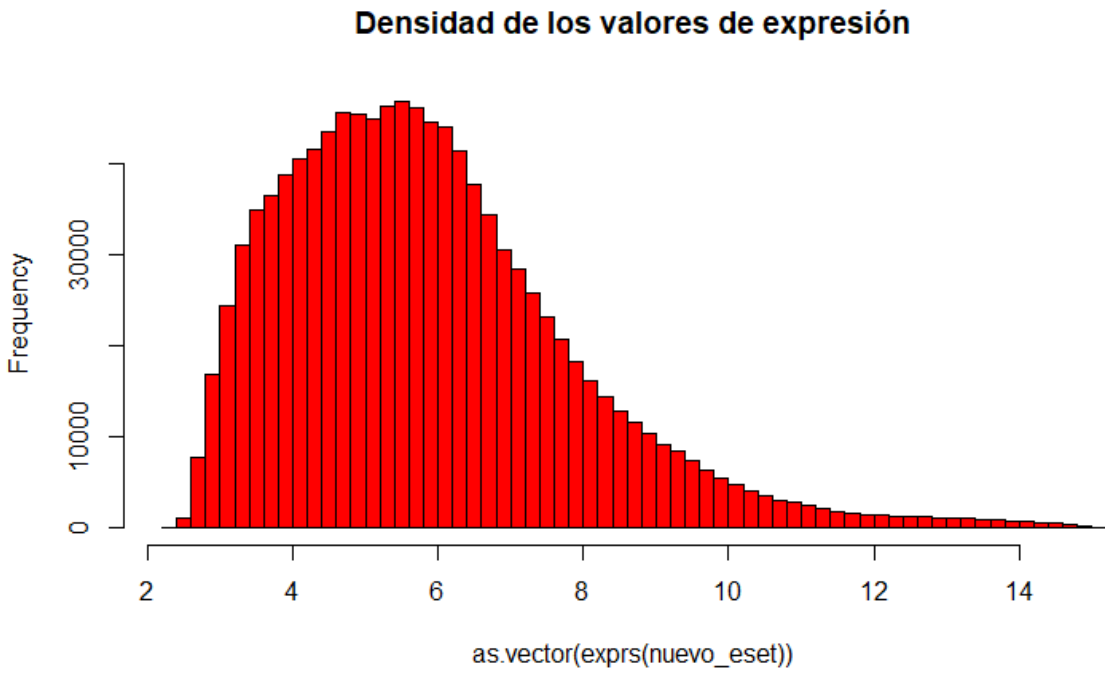


Este gráfico muestra la distribución de las intensidades crudas de los datos del microarray (sin normalizar) para diferentes muestras (etiquetadas como GSM944857, GSM944840, etc.).

Este nos indica que:

1. **Consistencia entre muestras**: Todas las cajas tienen tamaños similares y están alineadas a un nivel general, lo que sugiere que las intensidades de los datos crudos son bastante consistentes entre las muestras. Esto es bueno porque indica que no hay muestras con valores extremos o grandes sesgos antes de normalizar.
2. **Presencia de outliers**: Los círculos en la parte superior e inferior de las cajas representan outliers, que son esperables en datos de microarrays debido a las diferencias naturales entre genes.

```
>hist(as.vector(exprs(nuevo_eset)), breaks = 50, main = "Densidad de los valores de expresión ", col =
"red")
```



Este histograma nos muestra la densidad de las intensidades crudas de las sondas del microarray. Las intensidades están en escala logarítmica (base 2), bastante habitual en estos análisis para normalizar la variabilidad.

Este nos indica que:

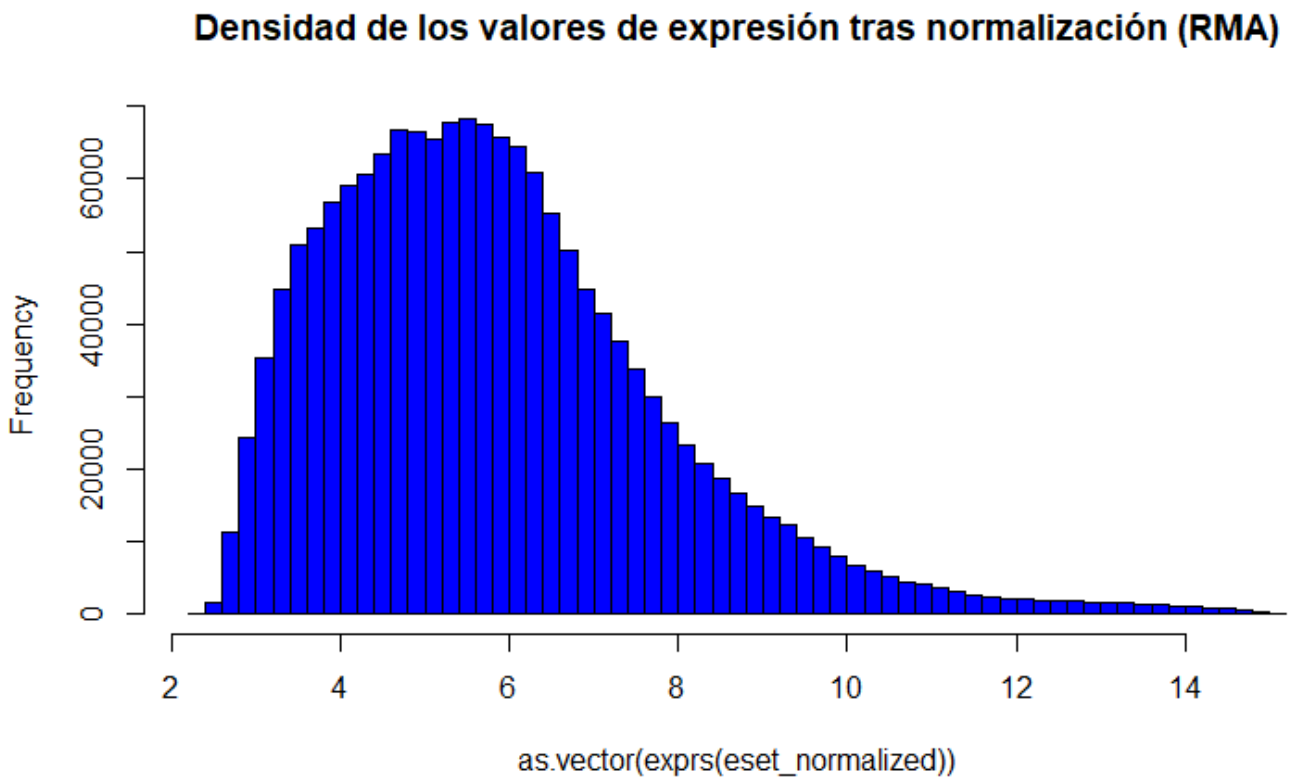
1. **Forma de la distribución**: La curva tiene un pico claro alrededor de ~6, lo que representa la mayor densidad de intensidades crudas. Bastante típico de datos de microarrays antes de la normalización.
2. **Asimetría a la derecha**: La cola hacia la derecha (valores >10) sugiere que hay algunos genes con intensidades más altas, lo que es esperable en datos crudos.

Aplicamos la normalización RMA para ajustar las intensidades.

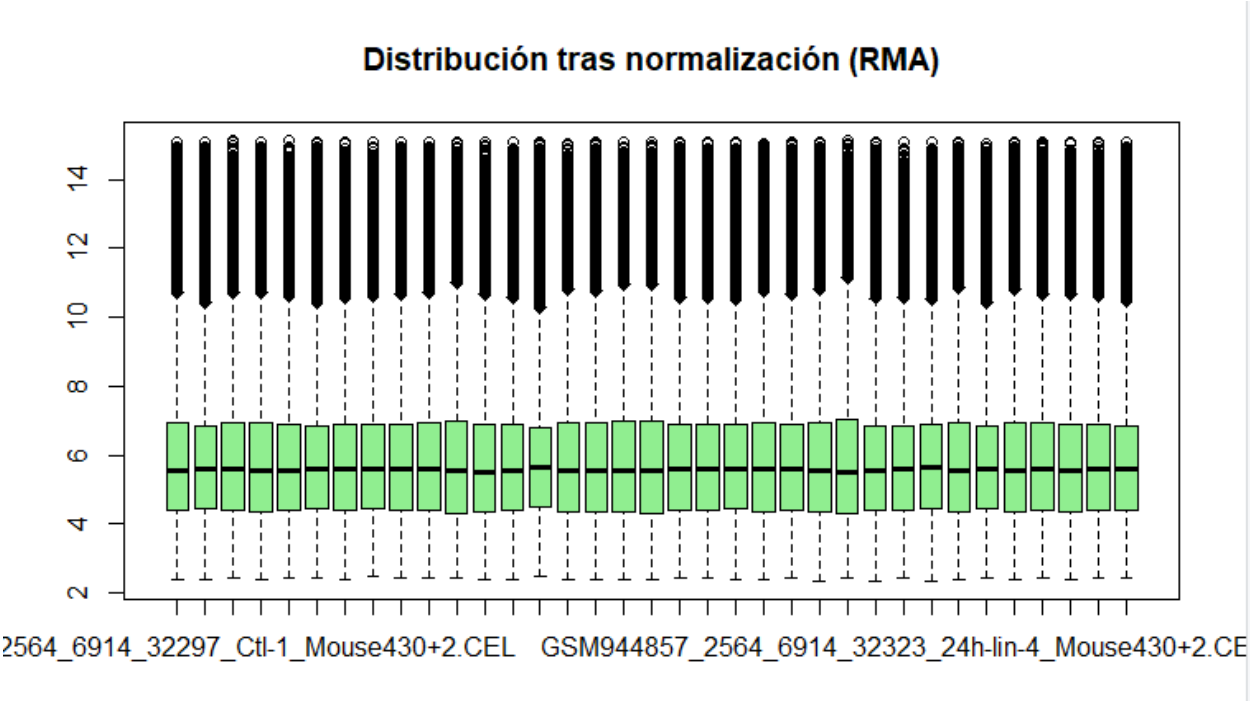
```
>eset_normalized <- rma(eset_cel)
```

Y visualizamos los datos después de normalización

```
>hist(as.vector(exprs(eset_normalized)), breaks = 50, main = "Densidad de los valores de expresión tras
normalización (RMA)", col = "blue")
```



```
>boxplot(exprs(eset_normalized), main = "Distribución tras normalización (RMA)", col = "lightgreen")
```



Este boxplot nos muestra las distribuciones de intensidades para las muestras después de la normalización utilizando el método RMA. En este podemos ver:

1. **Uniformidad entre muestras:**
    - Después de la normalización, las distribuciones de las intensidades para todas las muestras son prácticamente idénticas.
    - Esto indica que la normalización fue efectiva en corregir cualquier sesgo técnico, como diferencias en el rango de intensidades entre muestras.
2. **Rango ajustado:**
    - Los valores de intensidad ahora se encuentran en un rango más estrecho, centrado alrededor de 6-8 (en escala logarítmica base 2).
    - Este ajuste es típico del método RMA y asegura que los datos sean comparables entre las muestras.
3. **Reducción de outliers:**
    - Aunque aún pueden existir algunos outliers, estos están mucho más contenidos en comparación con los datos crudos.
    - Esto refleja la robustez del método RMA para manejar valores atípicos.

Complementamos el análisis exploratorio con un control de calidad exhaustivo con arrayQualityMetrics

```
>BiocManager::install("arrayQualityMetrics")
>library(arrayQualityMetrics)
>arrayQualityMetrics(expressionset = eset_normalized, outdir = "QC_report", force = TRUE)
The report will be written into directory 'QC_report'.
```

Adjunto el informe de calidad en mi directorio Github.

4.2.- Filtrado de datos

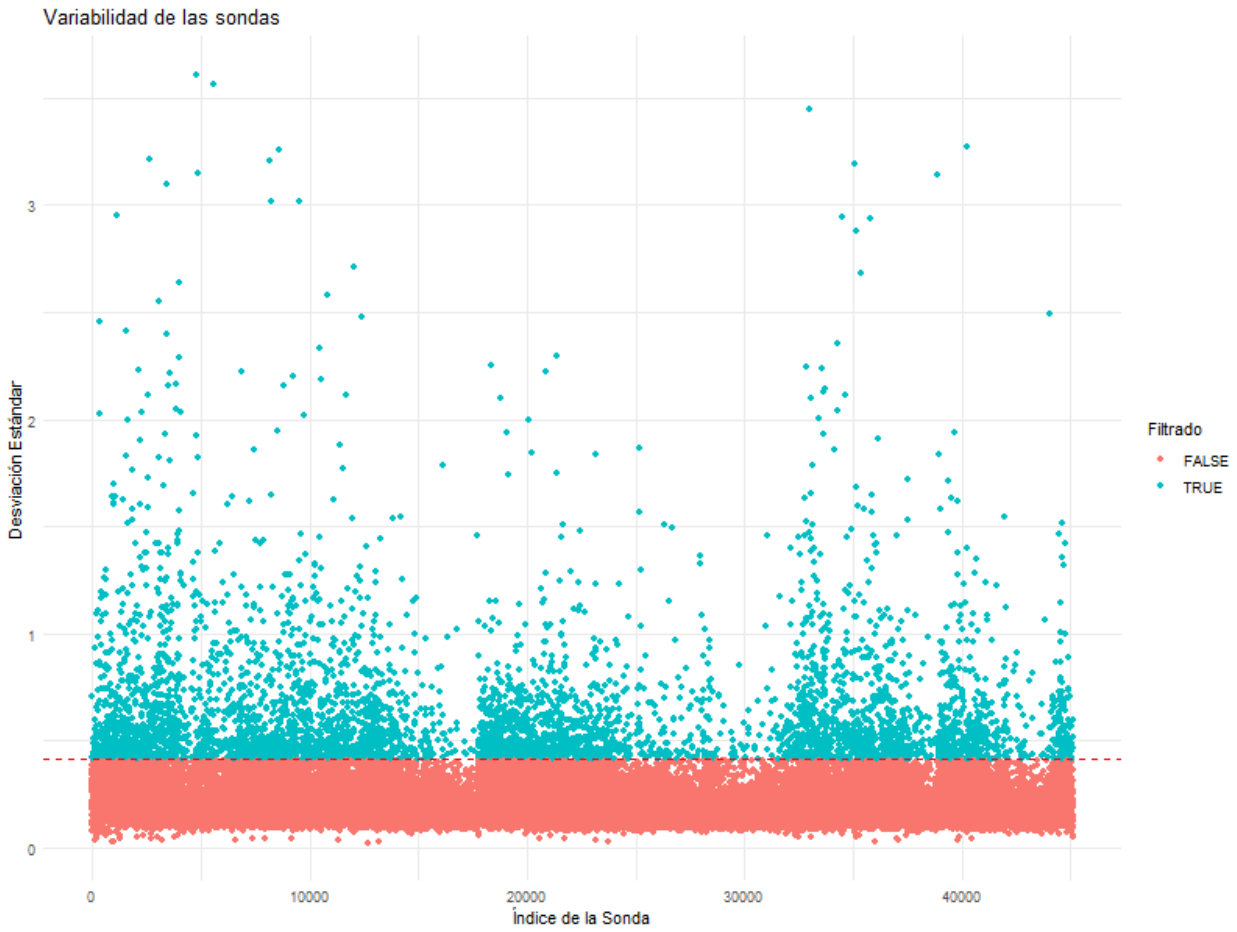
Para el filtrado de datos, seleccionamos las sondas más variables reteniendo el 10% con mayor variabilidad. Calculamos la desviación estándar por sonda:

```
>variabilidad <- apply(exprs(eset_normalized), 1, sd)
```

Y filtramos el 10% de sondas con mayor variabilidad

```
>threshold <- quantile(variabilidad, 0.9)
>filtered_eset <- eset_normalized[variabilidad >= threshold, ]
```

```
> ggplot(data, aes(x = Probe, y = Variability, color = Filtered)) +
+     geom_point() +
+     geom_hline(yintercept = threshold, linetype = "dashed", color = "red") +
+     labs(
+         title = "Variabilidad de las sondas",
+         x = "Índice de la Sonda",
+         y = "Desviación Estándar",
+         color = "Filtrado"
+     ) +
+     theme_minimal()
> dev.off()
```



Como podemos ver del gráfico anterior, los colores nos indican:

- **Azul ("TRUE")**: Representa las sondas que tienen una desviación estándar mayor o igual al percentil 90. Estas son las sondas con mayor variabilidad, seleccionadas por el filtro.
- **Rojo ("FALSE")**: Indica las sondas cuya desviación estándar está por debajo del umbral del percentil 90 y no fueron seleccionadas.

La línea horizontal roja discontinua nos marca el umbral del percentil 90 de la desviación estándar. Donde las sondas por encima de esta línea son las más variables.

Algunas observaciones de este gráfico serán:

- Solo una minoría de las sondas tiene una desviación estándar lo suficientemente alta como para superar el umbral del percentil 90.
- Hay varias sondas con valores excepcionalmente altos de desviación estándar (outliers), que podrían ser de interés particular para un análisis más detallado.
- La mayoría de las sondas tienen una baja variabilidad, como lo indica la densidad de puntos rojos por debajo del umbral.

4.3.- Construcción de las matrices de diseño y de contrastes

Construimos las matrices de diseño y los contrastes para realizar las siguientes comparaciones:

1. Infectados vs no infectados tratados con Linezolid.
2. Infectados vs no infectados tratados con Vancomicina.
3. Linezolid vs Vancomicina.

1. Filtramos las muestras comunes entre la matriz de expresión y pData

```
>common_samples <- intersect(colnames(expression_data), rownames(pdata_filtrado))
>expression_data_filtered <- expression_data[, common_samples]
>pdata_filtered <- pdata_filtrado[common_samples, ]
```

2. Creamos un objeto ExpressionSet

```
> library(Biobase)
> filtered_eset <- ExpressionSet(
+     assayData = expression_data_filtered,
+     phenoData = AnnotatedDataFrame(pdata_filtered)
+ )
```

3. Creamos la matriz de diseño usando la variable '***agent***'

```
> library(limma)
> design <- model.matrix(~ 0 + pData(filtered_eset)$agent)
> colnames(design) <- levels(as.factor(pData(filtered_eset)$agent))
```

4. Definimos la matriz de contrastes usando los nombres correctos

```
> matriz_contrastes <- makeContrasts(
+   Linezolid_vs_Untreated = "linezolid - no",
+   Vancomycin_vs_Untreated = "vancomycin - no",
+   Linezolid_vs_Vancomycin = "linezolid - vancomycin",
+   levels = design
+ )
```

Ahora que hemos definido la matriz de diseño y la matriz de contrastes correctamente, procedemos con el análisis de expresión diferencial en ***limma***. Para el análisis de expresión diferencial, ajustaremos el modelo lineal y obtendremos las listas de genes diferencialmente expresados para cada comparación

Ajustamos el modelo lineal:

```
> fit <- lmFit(exprs(filtered_eset), design)
> fit2 <- contrasts.fit(fit, matriz_contrastes)
> fit2 <- eBayes(fit2)
```

Generamos los resultados para cada comparación:

```
> results_linezolid <- topTable(fit2, coef = "Linezolid_vs_Untreated", adjust = "fdr", number = Inf)
> results_vancomycin <- topTable(fit2, coef = "Vancomycin_vs_Untreated", adjust = "fdr", number = Inf)
> results_comparison <- topTable(fit2, coef = "Linezolid_vs_Vancomycin", adjust = "fdr", number = Inf)
```

Y guardaremos los resultados:

```
> write.csv(results_linezolid, "DEG_Linezolid_vs_Untreated.csv")
> write.csv(results_vancomycin, "DEG_Vancomycin_vs_Untreated.csv")
> write.csv(results_comparison, "DEG_Linezolid_vs_Vancomycin.csv")
```

Para visualizar los resultados del análisis de expresión diferencial realizaremos un volcano plot. Este gráfico nos mostrará el log2 fold change frente al -log10 del valor p ajustado, destacando los genes significativamente regulados diferencialmente.

```
> library(ggplot2)
> library(ggrepel)
```

Generamos gráficos para las tres comparaciones

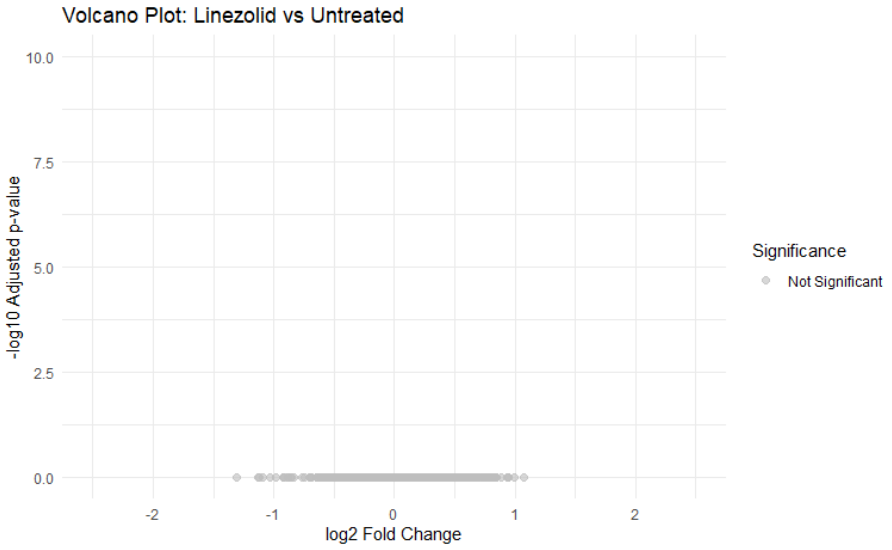
```
> generate_volcano_plot <- function(results, comparison_label) {
```

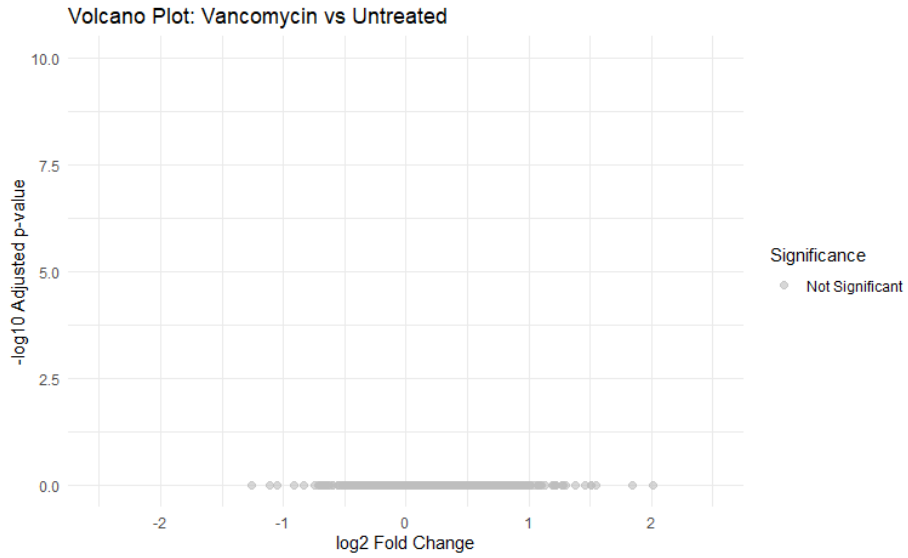
Creamos columna de significancia si no existe

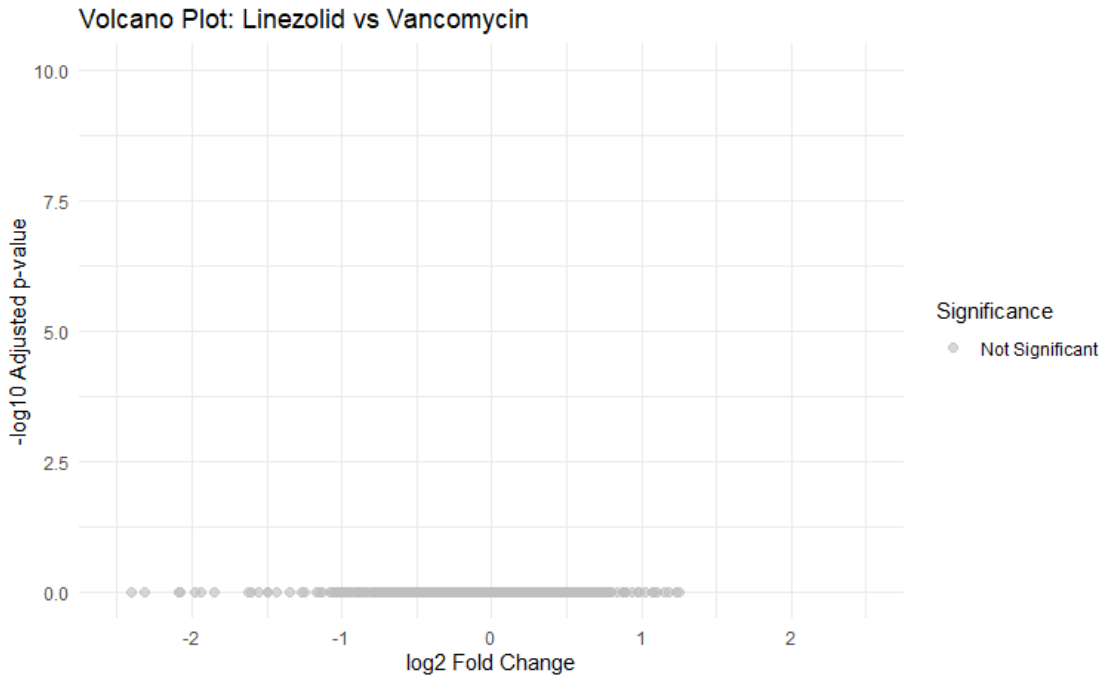
```
+   if (!"Significance" %in% colnames(results)) {
+     results$Significance <- with(
+       results,
+       ifelse(adj.P.Val < 0.1 & abs(logFC) > 0.5, "Significant", "Not Significant")
+     )
+   }
+
+   ggplot(results, aes(x = logFC, y = -log10(adj.P.Val), color = Significance)) +
+     geom_point(alpha = 0.6, size = 2) +
+     scale_x_continuous(limits = c(-2.5, 2.5)) +
+     scale_y_continuous(limits = c(0, 10)) +
+     scale_color_manual(values = c("Significant" = "red", "Not Significant" = "grey")) +
+     labs(
+       title = paste("Volcano Plot:", comparison_label),
+       x = "log2 Fold Change",
+       y = "-log10 Adjusted p-value",
+       color = "Significance"
+     ) +
+     theme_minimal()
+ }
```

Y finalmente visualizamos el gráfico para cada comparación:

```
> generate_volcano_plot(results_linezolid, "Linezolid vs Untreated")
> generate_volcano_plot(results_vancomycin, "Vancomycin vs Untreated")
> generate_volcano_plot(results_comparison, "Linezolid vs Vancomycin")
```







En las tres comparaciones:

- No hay evidencia de cambios significativos en la expresión génica entre los tratamientos y los controles.
- Esto puede deberse a:
    - **Ausencia de efecto biológico**: Los tratamientos no alteran sustancialmente la expresión génica bajo las condiciones experimentales.
    - **Falta de poder estadístico**: Pocas réplicas o alta variabilidad en los datos podrían dificultar detectar diferencias.

4.4.- Anotación de genes y análisis funcional (GO, KEGG)

Anotaremos las listas de genes obtenidas utilizando **org.Mm.eg.db** (ratón).

```
> if (!requireNamespace("BiocManager", quietly = TRUE)) {
+     install.packages("BiocManager")
+ }
>BiocManager::install("clusterProfiler")
>library(clusterProfiler)
>BiocManager::install("org.Mm.eg.db")
>library(org.Mm.eg.db)
>BiocManager::install("mouse4302.db")
>library(mouse4302.db)
```

Ahora mapeamos los IDs de microarray a ENTREZID y SYMBOL

```
> mapped_entrez <- AnnotationDbi::select(
+     mouse4302.db,
+     keys = gene_ids,
+     columns = c("ENTREZID", "SYMBOL"),
+     keytype = "PROBEID"
+ )
```

Y eliminamos duplicados si es necesario

```
> mapped_entrez <- mapped_entrez[!duplicated(mapped_entrez$PROBEID), ]
> head(mapped_entrez)
       PROBEID ENTREZID        SYMBOL
1 1455422_x_at    18952       Septin4
2   1418827_at    67276          Eri1
3   1440404_at     <NA>          <NA>
4   1431759_at   619677 4930529C04Rik
6   1440368_at   193796         Kdm4b
7   1425644_at    16847          Lepr
```

Guardamos la anotación:

```
>write.csv(annotated_genes, "Annotated_genes_Linezolid.csv", row.names = FALSE)
```

Una vez anotados los genes podemos intentaremos interpretar los resultados determinando si las listas se encuentran enriquecidas en algunas categorías biológicas

```
> library(clusterProfiler)
> library(clusterProfiler)
> library(org.Mm.eg.db)
> library(enrichplot)
```

Realizamos el análisis de enriquecimiento GO(Gene Ontology)). Para ello extraemos los IDs de tipo **ENTREZID** del objeto ***annotated_genes*** y los asignamos a ***entrez_ids***

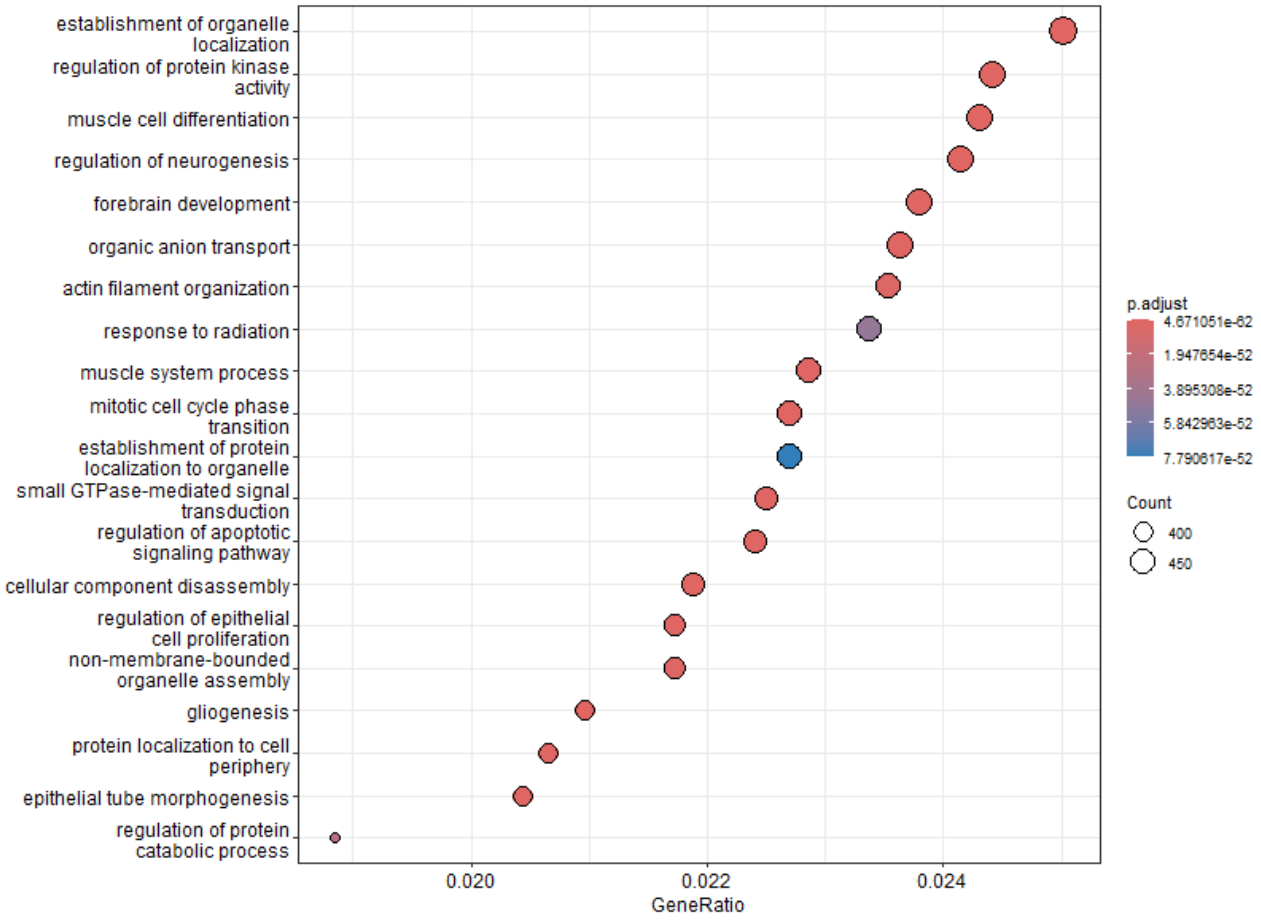
```
> entrez_ids <- annotated_genes$ENTREZID
```

Verificamos si hay valores **NA** en **ENTREZIDs,** filtrando para eliminar posibles NA antes de continuar:

```
> entrez_ids <- entrez_ids[!is.na(entrez_ids)]
> go_enrichment <- enrichGO(
+     gene          = entrez_ids,
+     OrgDb         = org.Mm.eg.db,
+     keyType       = "ENTREZID",
+     ont           = "BP",
+     pAdjustMethod = "BH",
+     pvalueCutoff  = 0.1,
+     qvalueCutoff  = 0.5
+ )
```

Visualizamos los resultados:

```
> if (!requireNamespace("ggplot2", quietly = TRUE)) {
+     install.packages("ggplot2")
+ }
> library(ggplot2)
> dotplot(go_enrichment, showCategory = 20)
```



Según podemos ver el eje X (GeneRatio) representa la proporción de genes en nuestra lista de interés que están asociados con cada término GO. Y el eje Y muestra los nombres de las categorías biológicas o procesos enriquecidos (por ejemplo, "establishment of organelle localization", "muscle cell differentiation", etc.).

Podemos establecer que:

- El tamaño del punto representa la cantidad de genes en la lista que están asociados con cada término GO.
- Categorías con puntos más grandes tienen más genes involucrados.
- El color del punto indica el valor ajustado de p (p.adjust) para cada término GO.
- Colores más rojos indican términos más significativos.
- Colores más azules indican términos menos significativos.

En el gráfico, los valores de p ajustados son extremadamente pequeños, lo que indica una alta significancia estadística para la mayoría de los términos.

Entre los términos más significativos (con mayor tamaño y color más intenso), se encuentran:

- **"establishment of organelle localization"**: Indica procesos relacionados con la organización de orgánulos celulares.
- **"muscle cell differentiation"**: Se refiere al proceso de especialización de células musculares.
- **"regulation of neurogenesis"**: Está asociado a la formación de nuevas células neuronales.
- **"forebrain development"**: Relacionado con el desarrollo del cerebro anterior.

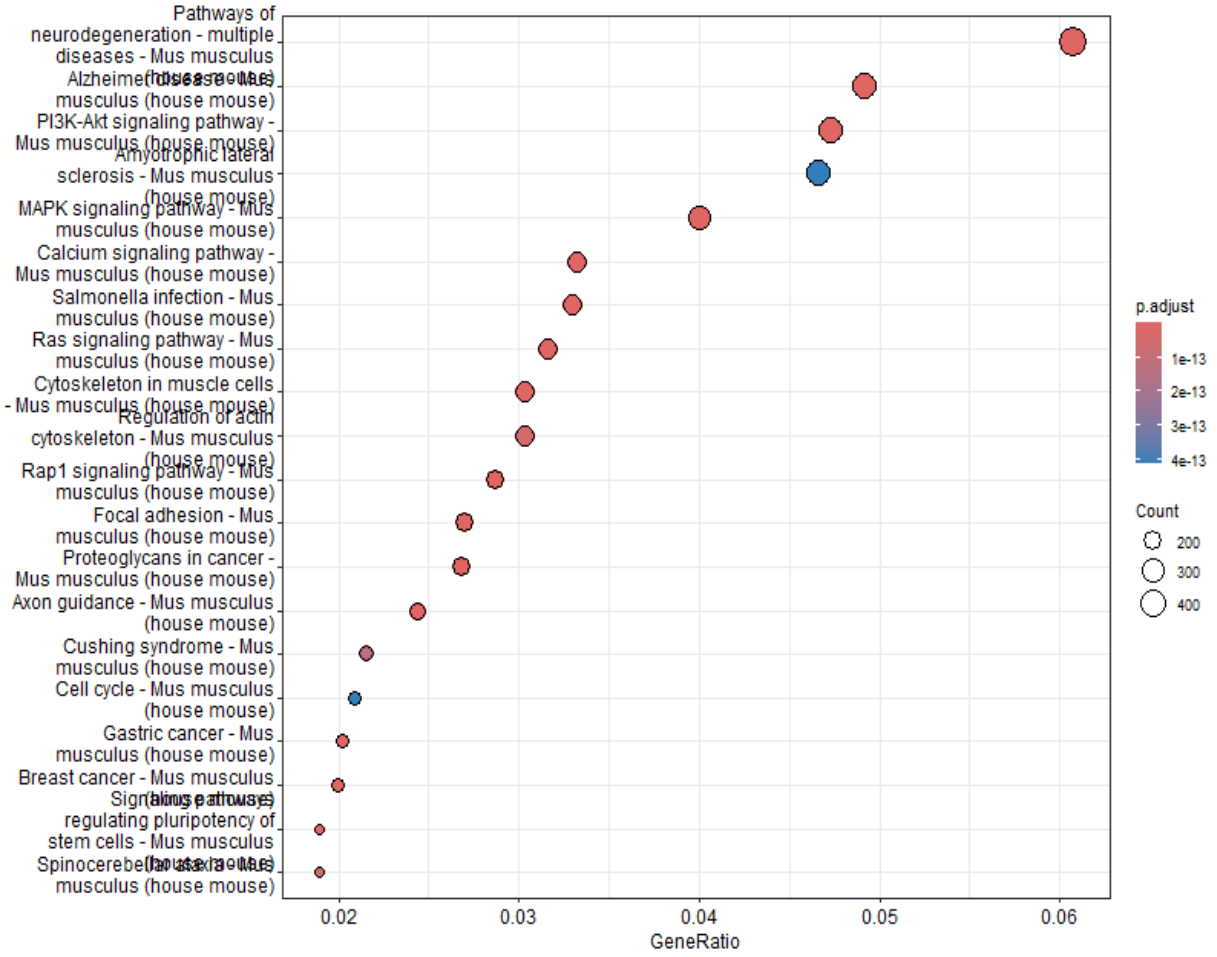
Estos términos sugieren que los genes diferencialmente expresados están enriquecidos en procesos relacionados con la organización celular, el desarrollo de tejidos (particularmente musculares y neuronales), y mecanismos de transporte y señalización.

Realizamos también el análisis para rutas KEGG:

```
> kegg_enrichment <- enrichKEGG(
+     gene          = entrez_ids,
+     organism      = "mmu", # Código KEGG para ratón
+     pAdjustMethod = "BH",
+     pvalueCutoff  = 0.05
```

Visualizamos los resultados

```
>dotplot(kegg_enrichment, showCategory = 20)
```



Como podemos ver a partir del gráfico anterior las rutas más significativas incluyen:

- **Pathways of neurodegeneration - multiple diseases**: Implica que muchos genes están involucrados en enfermedades neurodegenerativas.
- **Alzheimer disease**: Relacionado con procesos específicos en esta enfermedad.
- **PI3K-Akt signaling pathway**: Una ruta crítica en la regulación del crecimiento celular, supervivencia y metabolismo.

Esto puede sugerir que los genes en la lista están implicados en procesos relacionados con enfermedades neurodegenerativas o señalización celular.

1. **GeneRatio alto**:
    - Rutas con valores altos de GeneRatio, como **"Pathways of neurodegeneration"**, tienen una alta proporción de genes enriquecidos en la lista.
    - Esto indica que estas rutas son biológicamente relevantes en el contexto del análisis.

2. **Diversidad biológica**:
    - Las rutas enriquecidas abarcan temas como:
        - Enfermedades específicas (Alzheimer, cáncer de mama, cáncer gástrico).
        - Señalización celular (PI3K-Akt, MAPK, Ras, Rap1).
        - Procesos celulares como adhesión focal, ciclo celular y regulación del citoesqueleto.

Estos resultados nos sugieren que, aunque no se encontraron diferencias significativas entre las condiciones experimentales concretas (en función de infección o tratamiento), el subconjunto de genes más variables está relacionado con funciones celulares y vías señalizadoras clave en procesos de desarrollo, diferenciación, organización celular y potencial relevancia en patologías complejas (neurodegenerativas y cáncer).

5. Conclusiones

El análisis de datos de microarrays realizado **no** evidenció diferencias sustanciales en la expresión génica entre las condiciones comparadas. Sin embargo, la caracterización funcional del subconjunto más variable de sondas indicó un claro enriquecimiento en procesos biológicos y rutas de señalización relevantes a nivel celular y fisiológico. Esto sugiere que las condiciones estudiadas, bajo el diseño actual y el conjunto de datos disponible, no inducen cambios drásticos en la expresión génica, o que es necesaria una mayor potencia estadística para detectar efectos sutiles. La información funcional obtenida podría servir como punto de partida para estudios más enfocados o con diseños experimentales ajustados que permitan detectar cambios más delicados en la expresión génica.