PEC2

Francesc Torrents Torre

2024-12-20

Introducción y Objetivos

En esta PEC, una vez familiarizados con los datos de expresión, y los métodos y herramientas para la selección de genes y el análisis de la significación biológica, procedemos a la realización de un análisis de datos, que nos permitirá mejorar nuestra comprensión de un problema biológico mediante métodos y herramientas estadísticas y bioinformáticas.

Se basa en los datos de un estudio que, utilizando un modelo murino se investiga la utilidad de los antibióticos LINEOLID y VANCOMICINA para inmunomodulación durante infecciones por Staphylococcus aureus resistente a meticilina (MRSA).

Los resultados se subiran al repositorio github siguiente: https://github.com/Frantt96/Torrents-Torre-Francesc-PEC2

Métodos

El análisis se basa en un dataset con un total de 35 muestras, 15 de ellas tomadas antes de la infeccion y 20 despues. Estas se pueden extraer de los archivos facilitados por el profesor y en la pagina web: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE38531

Se generaran 3 listas de genes que deberan ser caracterizadas mediante análisis de significación biológica y finalmente compararse entre ellas. Antes de empezar con el analisis, hay que realizar una preparación de los datos suministrados.

Resultados

Primero de todo, prescindimos de las cinco muestras tomadas a las 2 horas y sorteamos las muestras restantes de forma que se conserven tan sólo cuatro muestras de cada grupo. Esto es realizado a traves de la función selectSamples que está disponible en el archivo facilitado en el campus "selectSamples.R". Nos permite extraer 24 muestras distintas indicando el DNI (77129830) como semilla.

```
filter_microarray <- function(allTargets, seed = 77129830) {
  # Configurar la semilla aleatoria
  set.seed(77129830)

  # Filtrar las filas donde 'time' no sea 'hour 2'
  filtered <- subset(allTargets, time != "hour 2")

  # Dividir el dataset por grupos únicos de 'infection' + 'agent'
```

```r
  filtered$group <- interaction(filtered$infection, filtered$agent)

  # Seleccionar 4 muestras al azar de cada grupo
  selected <- do.call(rbind, lapply(split(filtered, filtered$group), function(group_data) {
    if (nrow(group_data) > 4) {
      group_data[sample(1:nrow(group_data), 4), ]
    } else {
      group_data
    }
  }))

  # Obtener los índices originales como nombres de las filas seleccionadas
  original_indices <- match(selected$sample, allTargets$sample)

  # Modificar los rownames usando 'sample' y los índices originales
  rownames(selected) <- paste0(selected$sample, ".", original_indices)

  # Eliminar la columna 'group' y devolver el resultado
  selected$group <- NULL
  return(selected)
}



# Simular el dataset basado en la descripción proporcionada
allTargets <- data.frame(
  sample = c("GSM944831", "GSM944838", "GSM944845", "GSM944852", "GSM944859",
             "GSM944833", "GSM944840", "GSM944847", "GSM944854", "GSM944861",
             "GSM944834", "GSM944841", "GSM944848", "GSM944855", "GSM944862",
             "GSM944832", "GSM944839", "GSM944846", "GSM944853", "GSM944860",
             "GSM944835", "GSM944842", "GSM944849", "GSM944856", "GSM944863",
             "GSM944836", "GSM944843", "GSM944850", "GSM944857", "GSM944864",
             "GSM944837", "GSM944844", "GSM944851", "GSM944858", "GSM944865"),
  infection = c(rep("uninfected", 15), rep("S. aureus USA300", 20)),
  time = c(rep("hour 0", 15), rep("hour 2", 5), rep("hour 24", 15)),
  agent = c(rep("untreated", 5), rep("linezolid", 5), rep("vancomycin", 5),
            rep("untreated", 5), rep("untreated", 5), rep("linezolid", 5), rep("vancomycin", 5))
)

# Aplicar la función (cambiar 123 por vuestro ID de la UOC u otro número que podáis escribir en el docu
result <- filter_microarray(allTargets, seed=77129830)


print(result)
```

```
##                 sample        infection    time      agent
## GSM944850.28 GSM944850 S. aureus USA300 hour 24  linezolid
## GSM944836.26 GSM944836 S. aureus USA300 hour 24  linezolid
## GSM944864.30 GSM944864 S. aureus USA300 hour 24  linezolid
## GSM944857.29 GSM944857 S. aureus USA300 hour 24  linezolid
## GSM944833.6  GSM944833       uninfected  hour 0  linezolid
## GSM944861.10 GSM944861       uninfected  hour 0  linezolid
## GSM944854.9  GSM944854       uninfected  hour 0  linezolid
```

```
## GSM944847.8  GSM944847       uninfected  hour 0  linezolid
## GSM944863.25 GSM944863 S. aureus USA300 hour 24  untreated
## GSM944835.21 GSM944835 S. aureus USA300 hour 24  untreated
## GSM944849.23 GSM944849 S. aureus USA300 hour 24  untreated
## GSM944856.24 GSM944856 S. aureus USA300 hour 24  untreated
## GSM944831.1  GSM944831       uninfected  hour 0  untreated
## GSM944838.2  GSM944838       uninfected  hour 0  untreated
## GSM944845.3  GSM944845       uninfected  hour 0  untreated
## GSM944859.5  GSM944859       uninfected  hour 0  untreated
## GSM944851.33 GSM944851 S. aureus USA300 hour 24 vancomycin
## GSM944837.31 GSM944837 S. aureus USA300 hour 24 vancomycin
## GSM944865.35 GSM944865 S. aureus USA300 hour 24 vancomycin
## GSM944844.32 GSM944844 S. aureus USA300 hour 24 vancomycin
## GSM944855.14 GSM944855       uninfected  hour 0 vancomycin
## GSM944848.13 GSM944848       uninfected  hour 0 vancomycin
## GSM944834.11 GSM944834       uninfected  hour 0 vancomycin
## GSM944862.15 GSM944862       uninfected  hour 0 vancomycin
```

Se comprueba que todas las instrucciones del enunciado se cumplen.

Una vez aplicada la función selectSamples, obtendremos un nuevo objeto (result) que nos permitirá la creación de un nuevo ExpressionSet leyendo unicamente aquellos archivos .CEL que se hayan seleccionado.

En la carpeta donde residen todos los archivos .CEL se pueden eliminar manualmente los que no corresponden a las muestras que usaremos.

```
# Obtenemos los nombres de los archivos en la carpeta GSE38531_RAW
cel_files_in_directory <- list.files(path = "GSE38531_RAW", pattern = "\\.CEL$", full.names = TRUE)

# Verificamos los nombres en 'result' y mapearlos con los archivos reales
sample_names <- gsub("\\..*", "", rownames(result))
matched_files <- sapply(sample_names, function(sample) {
  # Buscamos en los archivos el que contiene el nombre de la muestra
  file <- grep(sample, cel_files_in_directory, value = TRUE)
  if (length(file) == 1) {
    return(file)
  } else {
    return(NA)  # Si no se encuentra = NA
  }
})

# Comprobamos si faltan archivos
missing <- which(is.na(matched_files))
if (length(missing) > 0) {
  cat("Faltan archivos para las siguientes muestras:\n")
  print(sample_names[missing])
} else {
  cat("Todos los archivos han sido mapeados correctamente.\n")
}
```

```
## Todos los archivos han sido mapeados correctamente.
```

```
library(affy)
```

```r
# Leemos los datos crudos desde los archivos mapeados
raw_data <- ReadAffy(filenames = matched_files)

# Revisamos los datos
summary(raw_data)
```

```
##    Length     Class      Mode
##        24 AffyBatch        S4
```

```r
# Normalizamos los datos
normalized_data <- rma(raw_data)
```

```
## Warning: replacing previous import 'AnnotationDbi::head' by 'utils::head' when
## loading 'mouse4302cdf'

## Warning: replacing previous import 'AnnotationDbi::tail' by 'utils::tail' when
## loading 'mouse4302cdf'

##

## Background correcting
## Normalizing
## Calculating Expression
```

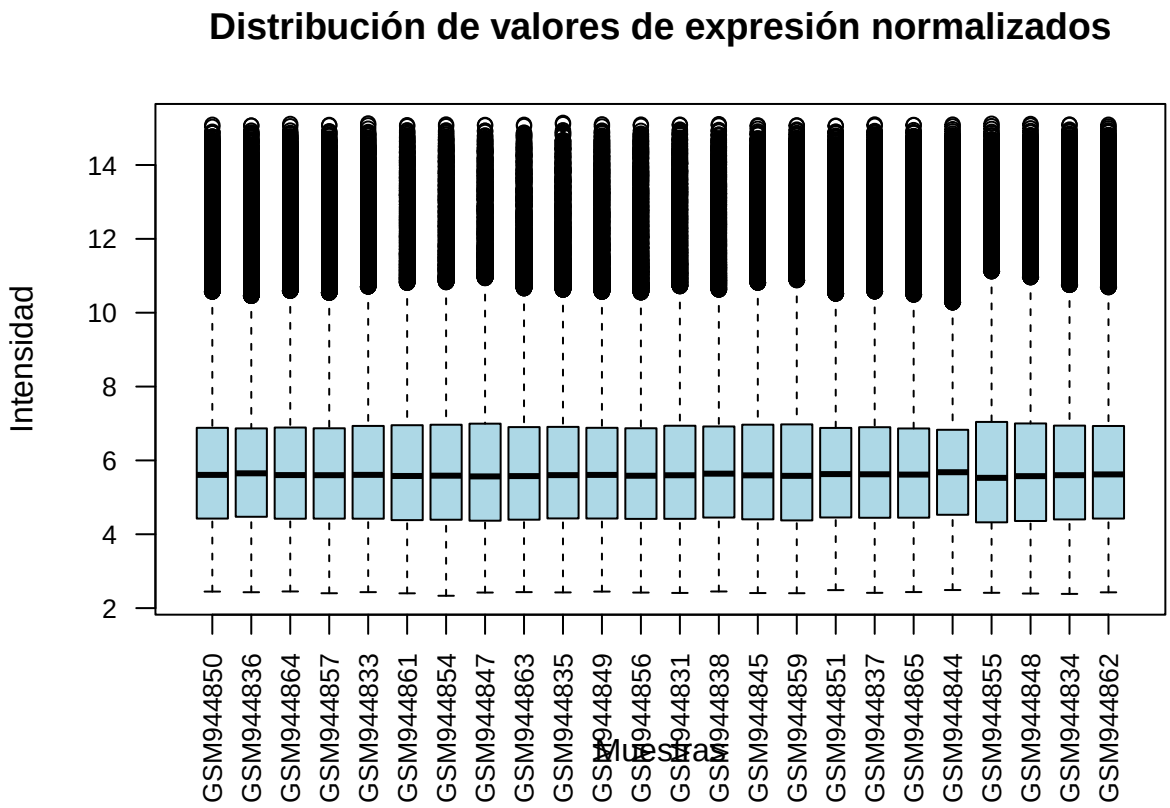
```r
# Agregamos información de las muestras desde 'result'
pData(normalized_data) <- result

# Verificamos ExpressionSet
print(normalized_data)
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 45101 features, 24 samples
##   element names: exprs
## protocolData
##   sampleNames: GSM944850_2564_6914_32316_24h-lin-3_Mouse430+2.CEL
##     GSM944836_2564_6914_32302_24h-lin-1_Mouse430+2.CEL ...
##     GSM944862_2564_6914_32328_Ctl-Van-5_Mouse430+2.CEL (24 total)
##   varLabels: ScanDate
##   varMetadata: labelDescription
## phenoData
##   sampleNames: GSM944850.28 GSM944836.26 ... GSM944862.15 (24 total)
##   varLabels: sample infection time agent
##   varMetadata: labelDescription
## featureData: none
## experimentData: use 'experimentData(object)'
## Annotation: mouse4302
```

Los datos, una vez normalizados utilizando el algoritmo RMA, seran utilizados para realizar el Análisis Exploratorio y el Control de Calidad.

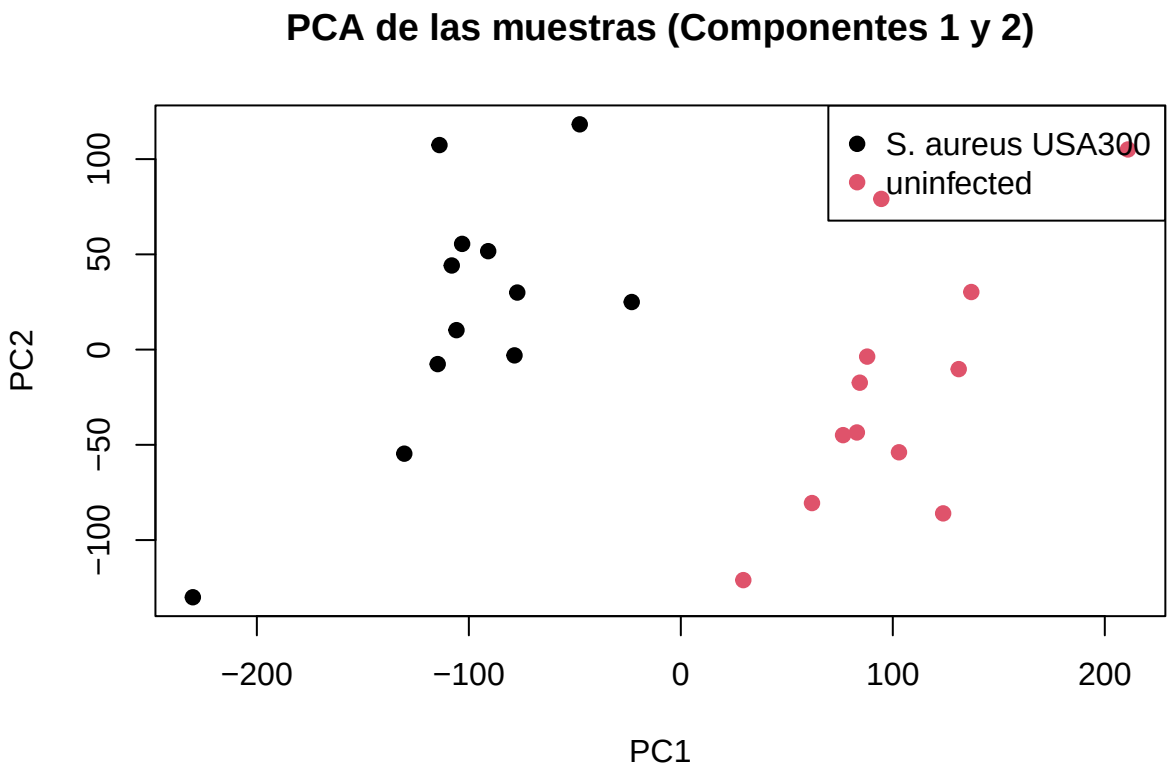
Inicialmente realizamos gráficos para evaluar la distribución de las expresiones y detectar posibles problemas:

```r
# Boxplot para verificar distribuciones

short_labels <- gsub("_.*", "", colnames(exprs(normalized_data)))
boxplot(exprs(normalized_data),
        main = "Distribución de valores de expresión normalizados",
        xlab = "Muestras", ylab = "Intensidad",
        names = short_labels, las = 2, col = "lightblue", cex.axis = 0.8)
```



Este gráfico nos muestra las distribuciones de valores de expresión normalizados para cada muestra. Los resultados nos muestran que tienen distribuciones similares en cuanto a rango/intensidad. No se observan muestras con distribuciones con diferencias considerables.

```r
# PCA (Análisis de Componentes Principales) para observar agrupamiento
library(stats)
pca <- prcomp(t(exprs(normalized_data)), scale. = TRUE)
plot(pca$x[, 1:2], col = as.factor(result$infection),
     pch = 19, main = "PCA de las muestras (Componentes 1 y 2)",
     xlab = "PC1", ylab = "PC2")
legend("topright", legend = unique(result$infection),
       col = 1:length(unique(result$infection)), pch = 19)
```



En este gráfico se busca visualizar la separación entre grupos de muestras (infectados vs no infectados) Los puntos negros representan muestras infectadas por S.aureus mientras que los puntos rojos representan muestras que no han sido infectadas por esta bacteria. Se puede observar una clara separación entre los dos grupos en las primeras dos componentes, sugiriendo diferencias biológicas entre infectados y no infectados. Dentro de cada grupo, las muestras parecen estar agrupadas con coherencia.

```
library(arrayQualityMetrics)

# Generar reporte de calidad
arrayQualityMetrics(expressionset = normalized_data,
                    outdir = "QualityControlReport",
                    force = TRUE)
```

```
## The report will be written into directory 'QualityControlReport'.

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
```

```
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style attribute
## name(s): subscripts, group.number, group.value
```

```
## (loaded the KernSmooth namespace)
```

Generamos un reporte de calidad. Está disponible en el siguiente enlace:

Reporte de calidad - arrayQualityMetrics

Filtrado de los datos

Para quedarnos con el 10% de las sondas con mayor variabilidad, necesitamos calcular la desviación estándar de cada sonda a lo largo de las muestras que tenemos. Generaremos un vector donde cada valor corresponde a la desviación estándar de una sonda.

```
variability <- apply(exprs(normalized_data), 1, sd)
summary(variability)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01995 0.14394 0.18116 0.23713 0.25927 3.48846
```

Ahora, calcularemos el umbral del percentil 90 para quedarnos con el 10% de las sondas más variables:

```
threshold <- quantile(variability, 0.9)
threshold
```

```
##       90%
## 0.4086203
```

El resultado es: el umbral para el 10% superior es 0.4086203, lo que indica que nos quedaremos con las sondas cuya desviación estándar sea mayor a este valor (generando un nuevo ExpressionSet con las sondas filtradas)

```
# Filtramoslas sondas que superen el umbral de variabilidad
filtered_indices <- which(variability > threshold)
filtered_data <- normalized_data[filtered_indices, ]

# Verificamos las dimensiones
dim(filtered_data)
```

```
## Features  Samples
##     4510       24
```

Ahora nos hemos quedado con 4510 sondas (en comparación con las 45101 originales.)

Construcción de las matrices de diseño y de contrastes

La matriz de diseño define las condiciones experimetnales de las muestras. Utilizaremos la información que hemos obtenido anteriormente para crearla.

```
library(limma)

# Creamos una nueva variable que combine infección y agente
pData(filtered_data)$group <- interaction(pData(filtered_data)$infection,
                                          pData(filtered_data)$agent, sep = "_")

# Verificamos los grupos únicos
unique(pData(filtered_data)$group)
```

```
## [1] S. aureus USA300_linezolid  uninfected_linezolid
## [3] S. aureus USA300_untreated  uninfected_untreated
## [5] S. aureus USA300_vancomycin uninfected_vancomycin
## 6 Levels: S. aureus USA300_linezolid ... uninfected_vancomycin
```

```
# Creamos la matriz de diseño
design <- model.matrix(~ 0 + group, data = pData(filtered_data))

# Renombramos las columnas para que sean más legibles
```

```r
colnames(design) <- levels(pData(filtered_data)$group)

# Verificamos la matriz
design
```

```
##              S. aureus USA300_linezolid uninfected_linezolid
## GSM944850.28                          1                     0
## GSM944836.26                          1                     0
## GSM944864.30                          1                     0
## GSM944857.29                          1                     0
## GSM944833.6                           0                     1
## GSM944861.10                          0                     1
## GSM944854.9                           0                     1
## GSM944847.8                           0                     1
## GSM944863.25                          0                     0
## GSM944835.21                          0                     0
## GSM944849.23                          0                     0
## GSM944856.24                          0                     0
## GSM944831.1                           0                     0
## GSM944838.2                           0                     0
## GSM944845.3                           0                     0
## GSM944859.5                           0                     0
## GSM944851.33                          0                     0
## GSM944837.31                          0                     0
## GSM944865.35                          0                     0
## GSM944844.32                          0                     0
## GSM944855.14                          0                     0
## GSM944848.13                          0                     0
## GSM944834.11                          0                     0
## GSM944862.15                          0                     0
##              S. aureus USA300_untreated uninfected_untreated
## GSM944850.28                           0                     0
## GSM944836.26                           0                     0
## GSM944864.30                           0                     0
## GSM944857.29                           0                     0
## GSM944833.6                            0                     0
## GSM944861.10                           0                     0
## GSM944854.9                            0                     0
## GSM944847.8                            0                     0
## GSM944863.25                           1                     0
## GSM944835.21                           1                     0
## GSM944849.23                           1                     0
## GSM944856.24                           1                     0
## GSM944831.1                            0                     1
## GSM944838.2                            0                     1
## GSM944845.3                            0                     1
## GSM944859.5                            0                     1
## GSM944851.33                           0                     0
## GSM944837.31                           0                     0
## GSM944865.35                           0                     0
## GSM944844.32                           0                     0
## GSM944855.14                           0                     0
## GSM944848.13                           0                     0
```

```
## GSM944834.11                          0                    0
## GSM944862.15                          0                    0
##              S. aureus USA300_vancomycin uninfected_vancomycin
## GSM944850.28                           0                     0
## GSM944836.26                           0                     0
## GSM944864.30                           0                     0
## GSM944857.29                           0                     0
## GSM944833.6                            0                     0
## GSM944861.10                           0                     0
## GSM944854.9                            0                     0
## GSM944847.8                            0                     0
## GSM944863.25                           0                     0
## GSM944835.21                           0                     0
## GSM944849.23                           0                     0
## GSM944856.24                           0                     0
## GSM944831.1                            0                     0
## GSM944838.2                            0                     0
## GSM944845.3                            0                     0
## GSM944859.5                            0                     0
## GSM944851.33                           1                     0
## GSM944837.31                           1                     0
## GSM944865.35                           1                     0
## GSM944844.32                           1                     0
## GSM944855.14                           0                     1
## GSM944848.13                           0                     1
## GSM944834.11                           0                     1
## GSM944862.15                           0                     1
## attr(,"assign")
## [1] 1 1 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
```

Cada columna representa una combinación especifica de las condiciones experimentales, y las filas corresponden a las muestras. Las columnas nos indican si la muestra pertenece al grupo infectado o no y si reciben tratamiento o no (1 para sí, 0 para no).

```r
# Corregimos los nombres de las columnas de la matriz de diseño
colnames(design) <- make.names(colnames(design))
colnames(design)
```

```
## [1] "S..aureus.USA300_linezolid"  "uninfected_linezolid"
## [3] "S..aureus.USA300_untreated"  "uninfected_untreated"
## [5] "S..aureus.USA300_vancomycin" "uninfected_vancomycin"
```

```r
# Creamos la matriz de contrastes con los nombres exactos de las columnas de diseño
contrast_matrix <- makeContrasts(
  Infectados_vs_NoInfectados = S..aureus.USA300_untreated - uninfected_untreated,
  Linezolid = S..aureus.USA300_linezolid - uninfected_linezolid,
  Vancomicina = S..aureus.USA300_vancomycin - uninfected_vancomycin,
  levels = design
)
```

```
# Verificamos
contrast_matrix
```

```
##                              Contrasts
## Levels                        Infectados_vs_NoInfectados Linezolid Vancomicina
##   S..aureus.USA300_linezolid                           0         1           0
##   uninfected_linezolid                                 0        -1           0
##   S..aureus.USA300_untreated                           1         0           0
##   uninfected_untreated                                -1         0           0
##   S..aureus.USA300_vancomycin                          0         0           1
##   uninfected_vancomycin                                0         0          -1
```

La matriz de contrastes especifica las comparaciones propuestas: - Compara muestras infectadas con no infectadas - Compara muestras infectadas tratadas con linezolid con no infectadas tratadas con linezolid - Compara muestras infectadas tratadas con vancomicina con no infectadas tratadas con vancomicina.

Con la matriz de diseño y de contrastes, podemos aplicar un modelo ajustado:

```
fit <- lmFit(exprs(filtered_data), design)

# Aplicamos los contrastes
fit_contrasts <- contrasts.fit(fit, contrast_matrix)

# Estimación Bayesiana
fit_contrasts <- eBayes(fit_contrasts)
summary(decideTests(fit_contrasts))
```

```
##        Infectados_vs_NoInfectados Linezolid Vancomicina
## Down                         2159      2183        2277
## NotSig                       1009      1036        1305
## Up                           1342      1291         928
```

El resumen de los resultados muestra el numero de genes diferencialmente expresados para cada comparación. Se puede observar que hay una gran cantidad de genes diferencialmente expresados en las tres comparaciones, indicando diferencias claras entre los grupos. La Vancomicina parece tener el mayor impacto en la reduccion de la expresión génica en comparación con Linezolid.

Obtención de las listas de genes diferencialmente expresados para cada comparación

Se utiliza la funcion topTable de limma para la obtención de los genes diferencialmente expresados en cada comparación.

```
# Obtener las listas de genes diferencialmente expresados
topTable_Inf_vs_NoInf <- topTable(fit_contrasts, coef = "Infectados_vs_NoInfectados", number = Inf, adju
topTable_Linezolid <- topTable(fit_contrasts, coef = "Linezolid", number = Inf, adjust = "BH")
topTable_Vancomicina <- topTable(fit_contrasts, coef = "Vancomicina", number = Inf, adjust = "BH")

# Visualizar los primeros genes de cada lista
head(topTable_Inf_vs_NoInf)
```

```
##                  logFC   AveExpr         t      P.Value    adj.P.Val        B
## 1427747_a_at  6.552210 10.610001  20.15628 1.657587e-15 7.475719e-12 25.45093
## 1421262_at    7.191766  7.187221  19.21400 4.430532e-15 9.990850e-12 24.52426
## 1422953_at    3.963496 11.191628  17.62098 2.586577e-14 3.888488e-11 22.84247
## 1450188_s_at  6.209387  6.334060  16.28504 1.269552e-13 1.431420e-10 21.30774
## 1433939_at   -2.017983  8.309129 -15.79346 2.345295e-13 1.991341e-10 20.71153
## 1418722_at    6.000006 10.946055  15.60142 2.993968e-13 1.991341e-10 20.47373
```

```
head(topTable_Linezolid)
```

```
##                  logFC   AveExpr         t      P.Value    adj.P.Val        B
## 1421262_at    6.159436  7.187221  16.45596 1.029384e-13 4.642522e-10 21.34168
## 1427747_a_at  4.978626 10.610001  15.31553 4.326895e-13 9.757148e-10 19.99193
## 1436419_a_at -2.273625  8.494368 -13.33449 6.573144e-12 9.881626e-09 17.39241
## 1433939_at   -1.622960  8.309129 -12.70187 1.678166e-11 1.342266e-08 16.48594
## 1448562_at    4.452143  7.706099  12.66486 1.774734e-11 1.342266e-08 16.43168
## 1422953_at    2.840672 11.191628  12.62911 1.873521e-11 1.342266e-08 16.37912
```

```
head(topTable_Vancomicina)
```

```
##                  logFC   AveExpr         t      P.Value    adj.P.Val        B
## 1421262_at    6.728738  7.187221  17.97695 1.723219e-14 7.771719e-11 23.06994
## 1427747_a_at  5.387002 10.610001  16.57180 8.939647e-14 2.015890e-10 21.52978
## 1450188_s_at  5.645549  6.334060  14.80629 8.459609e-13 1.271761e-09 19.39263
## 1419681_a_at  5.339981  7.417689  14.01642 2.487659e-12 2.804836e-09 18.35457
## 1436419_a_at -2.336923  8.494368 -13.70572 3.854435e-12 3.476700e-09 17.93113
## 1418722_at    5.137536 10.946055  13.35879 6.345132e-12 4.769424e-09 17.44775
```

Esto nos muestra el listado de todos los genes diferencialmente expresados. No obstante, podemos filtrar las listas para obtener solo los genes con un p-value ajustado por debajo del umbral (0.05)

```
# Filtro de genes diferencialmente expresados (p-valor ajustado < 0.05)
DEGs_Inf_vs_NoInf <- topTable_Inf_vs_NoInf[topTable_Inf_vs_NoInf$adj.P.Val < 0.05, ]
DEGs_Linezolid <- topTable_Linezolid[topTable_Linezolid$adj.P.Val < 0.05, ]
DEGs_Vancomicina <- topTable_Vancomicina[topTable_Vancomicina$adj.P.Val < 0.05, ]

# Verificar el número de genes significativos
nrow(DEGs_Inf_vs_NoInf)
```

```
## [1] 3501
```

```
nrow(DEGs_Linezolid)
```

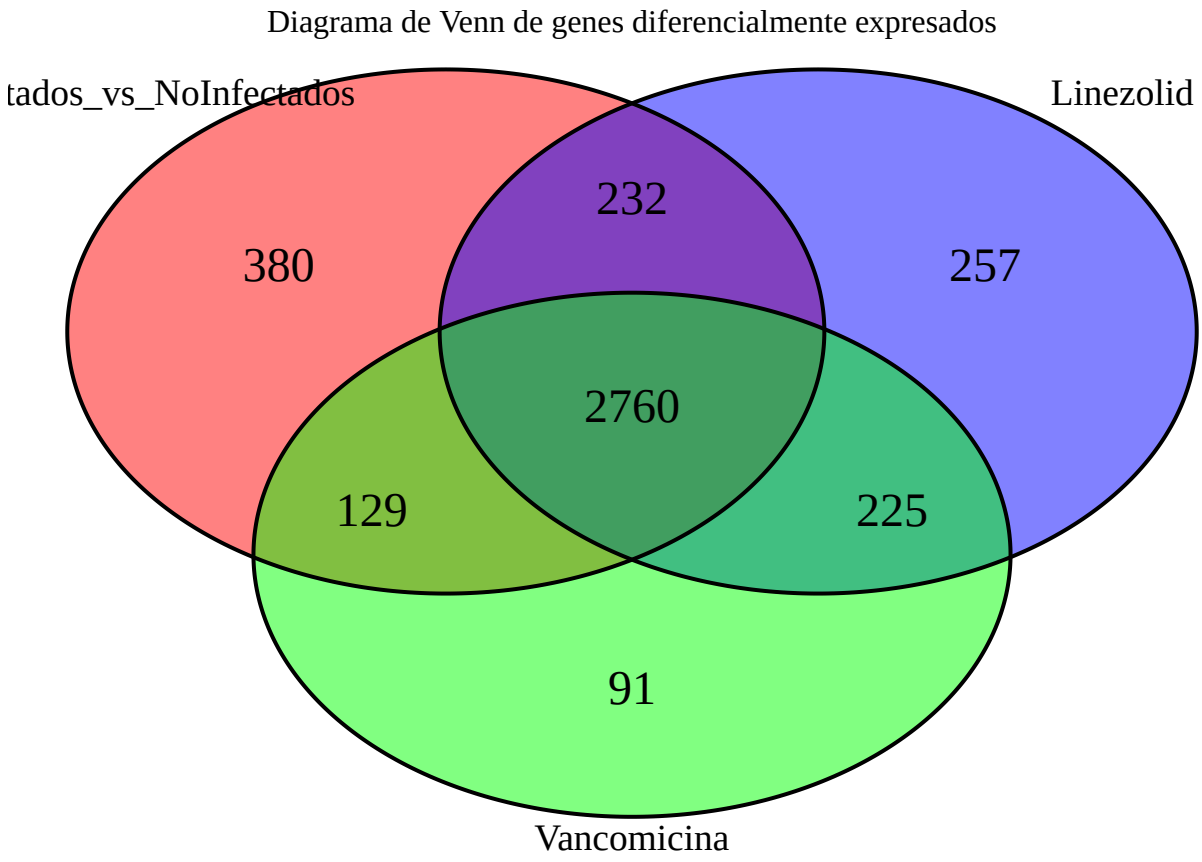
```
## [1] 3474
```

```
nrow(DEGs_Vancomicina)
```

```
## [1] 3205
```

Pasamos a comparar las listas de genes utilizando gráficos, en este caso el diagrama de Venn (visualización de la intersección de genes entre las listas)

```r
# Instalar paquete VennDiagram si no está instalado
if (!requireNamespace("VennDiagram", quietly = TRUE)) install.packages("VennDiagram")
library(VennDiagram)

# Extraer los IDs de genes de cada lista
genes_Inf_vs_NoInf <- rownames(DEGs_Inf_vs_NoInf)
genes_Linezolid <- rownames(DEGs_Linezolid)
genes_Vancomicina <- rownames(DEGs_Vancomicina)

# Crear el diagrama de Venn
venn.plot <- venn.diagram(
  x = list(Infectados_vs_NoInfectados = genes_Inf_vs_NoInf,
           Linezolid = genes_Linezolid,
           Vancomicina = genes_Vancomicina),
  filename = NULL,
  fill = c("red", "blue", "green"),
  alpha = 0.5,
  cex = 1.5,
  cat.cex = 1.2,
  main = "Diagrama de Venn de genes diferencialmente expresados"
)

# Mostrar el diagrama de Venn
grid.draw(venn.plot)
```



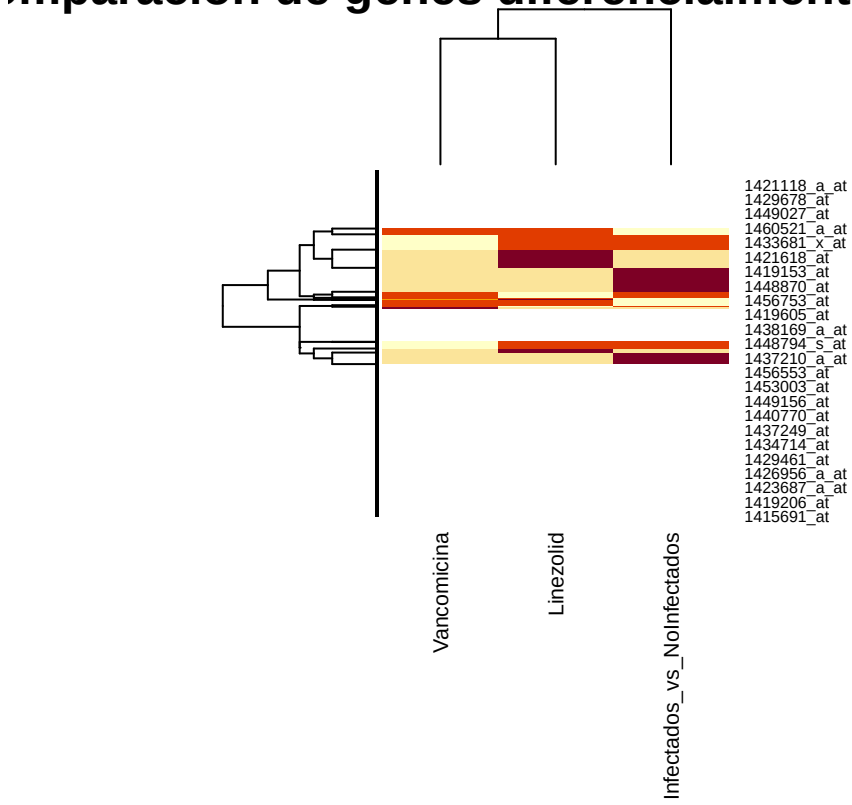
Otra alternativa es la utilización de la función DecideTests de Limma para clasificar los genes en “Up”,

"Down" o "No significativo" para cada contraste permitiendo la comparación de las listas de una manera sencilla.

```
comparison <- decideTests(fit_contrasts)
summary(comparison)
```

```
##        Infectados_vs_NoInfectados Linezolid Vancomicina
## Down                         2159      2183        2277
## NotSig                       1009      1036        1305
## Up                           1342      1291         928
```

```
# Visualizamos la comparación como un heatmap

heatmap(as.matrix(comparison),
        main = "Comparación de genes diferencialmente expresados",
        cexRow = 0.6,     # Ajusta el tamaño de las etiquetas en el eje Y
        cexCol = 0.8,     # Ajusta el tamaño de las etiquetas en el eje X
        margins = c(10, 10))  # Ajusta los márgenes (X, Y)
```



Anotación de los genes

Primero de todo, como estamos trabajando con modelos murinos, tenemos que utilizar el paquete específico para este organismo, en este caso org.Mm.eg.db

```r
# Instalar paquetes necesarios
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
BiocManager::install(c("annotate", "org.Mm.eg.db"))
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: https://cran.rstudio.com/

## Bioconductor version 3.18 (BiocManager 1.30.25), R 4.3.1 (2023-06-16 ucrt)

## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'annotate' 'org.Mm.eg.db'

## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.3.1/library
##   packages:
##     boot, cluster, codetools, foreign, KernSmooth, lattice, mgcv, nlme, rpart,
##     spatial, survival
```

```r
# Cargar los paquetes
library(annotate)
library(org.Mm.eg.db)
```

Se utiliza la columna rownames de las tablas de genes diferencialmente expresados como identificadores.

```r
probe_ids <- rownames(topTable_Inf_vs_NoInf)
topTable_Inf_vs_NoInf <- topTable(fit_contrasts, coef = "Infectados_vs_NoInfectados", number = Inf, adj
```

```r
if (!requireNamespace("mouse4302.db", quietly = TRUE)) {
  BiocManager::install("mouse4302.db")
}
# Instalar el paquete de anotación para la plataforma Mouse430
BiocManager::install("mouse4302.db")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: https://cran.rstudio.com/

## Bioconductor version 3.18 (BiocManager 1.30.25), R 4.3.1 (2023-06-16 ucrt)

## Warning: package(s) not installed when version(s) same as or greater than current; use
##   'force = TRUE' to re-install: 'mouse4302.db'

## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.3.1/library
##   packages:
##     boot, cluster, codetools, foreign, KernSmooth, lattice, mgcv, nlme, rpart,
##     spatial, survival
```

```
library(mouse4302.db)

# Mapeamos las IDs de las sondas a Symbol, EntrezID y EnsemblID
gene_symbols <- mapIds(mouse4302.db, keys = probe_ids, column = "SYMBOL", keytype = "PROBEID", multiVals
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
entrez_ids <- mapIds(mouse4302.db, keys = probe_ids, column = "ENTREZID", keytype = "PROBEID", multiVals
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
ensembl_ids <- mapIds(mouse4302.db, keys = probe_ids, column = "ENSEMBL", keytype = "PROBEID", multiVals
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
# Creamos una tabla con las anotaciones
annotated_genes <- data.frame(
  ProbeID = probe_ids,
  Symbol = gene_symbols,
  EntrezID = entrez_ids,
  EnsemblID = ensembl_ids,
  stringsAsFactors = FALSE
)

head(annotated_genes)
```

```
##                   ProbeID Symbol EntrezID          EnsemblID
## 1427747_a_at 1427747_a_at   Lcn2    16819 ENSMUSG00000026822
## 1421262_at     1421262_at   Lipg    16891 ENSMUSG00000053846
## 1422953_at     1422953_at   Fpr2    14289 ENSMUSG00000052270
## 1450188_s_at 1450188_s_at   Lipg    16891 ENSMUSG00000053846
## 1433939_at     1433939_at   Aff3    16764 ENSMUSG00000037138
## 1418722_at     1418722_at    Ngp    18054 ENSMUSG00000032484
```

Combinamos las anotaciones con los resultados de los genes

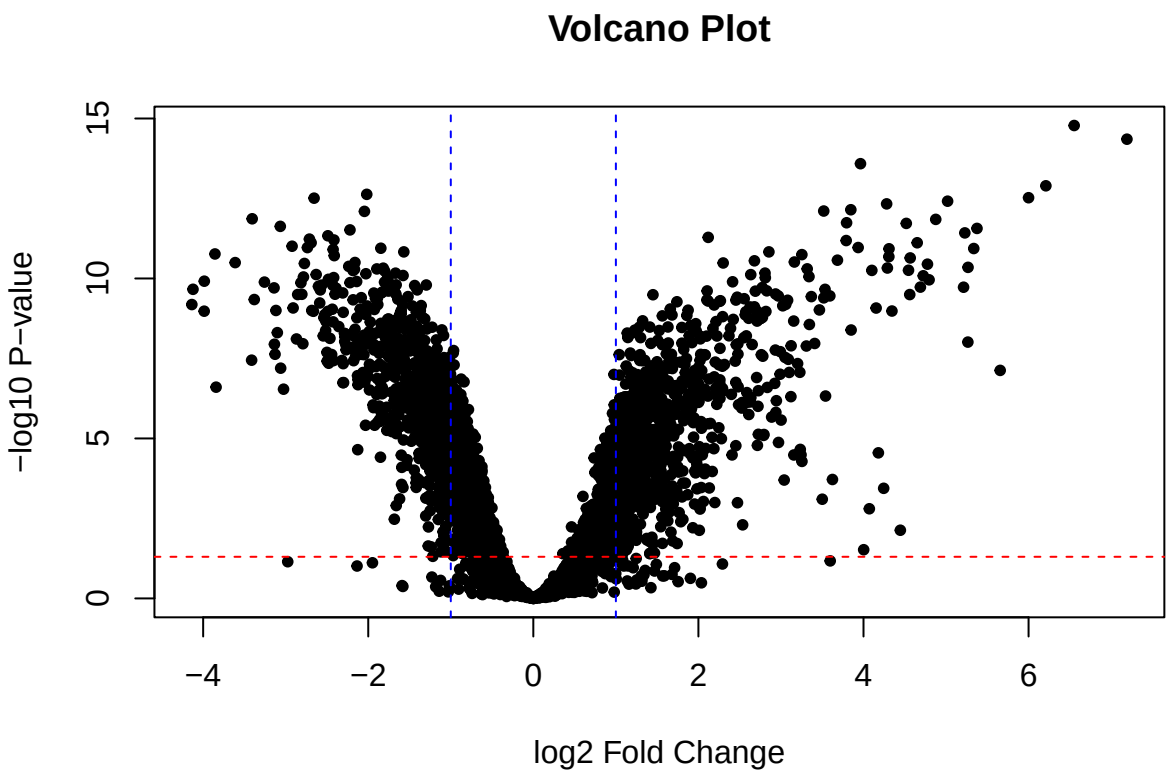
```
# Combinar las anotaciones con los resultados de Limma
annotated_results <- merge(annotated_genes, topTable_Inf_vs_NoInf, by.x = "ProbeID", by.y = "row.names"

# Mostrar las primeras filas del resultado combinado
head(annotated_results)
```

```
##        ProbeID Symbol EntrezID          EnsemblID      logFC   AveExpr
## 1   1415677_at  Dhrs1    52585 ENSMUSG00000002332  0.8243638  8.033064
## 2   1415682_at   Xpo7    65246 ENSMUSG00000022100 -1.1671935 10.225763
## 3   1415691_at   Dlg1    13383 ENSMUSG00000022770 -0.7151429  7.912106
## 4   1415703_at  Huwe1    59026 ENSMUSG00000025261 -0.6720011  8.200350
## 5   1415708_at   Tug1   544752 ENSMUSG00000056579 -0.6574545  9.285180
## 6 1415716_a_at  Rps27    57294 ENSMUSG00000090733 -0.6628767 12.798859
##           t      P.Value    adj.P.Val          B
```

```
## 1  4.813796 8.638285e-05 0.0002246751  0.9879197
## 2 -3.999312 6.205685e-04 0.0012240641 -0.9667840
## 3 -4.682862 1.184863e-04 0.0002904203  0.6730868
## 4 -4.898458 7.045067e-05 0.0001894648  1.1912936
## 5 -5.046147 4.941586e-05 0.0001401670  1.5455000
## 6 -3.581496 1.697596e-03 0.0029289056 -1.9525321
```

A partir de los resultados obtenidos, podemos generar gráficos para explorar los datos. El gráfico Volcano Plot nos facilita el resaltado de los genes más significativos y con mayores cambios de expresión.

```r
with(annotated_results, {
  plot(logFC, -log10(P.Value),
       pch = 20, main = "Volcano Plot",
       xlab = "log2 Fold Change", ylab = "-log10 P-value")
  abline(h = -log10(0.05), col = "red", lty = 2)  # Línea de p-value significativo
  abline(v = c(-1, 1), col = "blue", lty = 2)     # Líneas para logFC ±1
})
```



A partir del gráfico, los genes de interes más relevante se situan en las esquinas superiores izquierda y derecha. Estos tienen cambios grandes en la expresión y son altamente significativos. Los genes situados en la esquina superior derecha son genes sobreexpresados y los de la esquina superior izquierda son genes subexpresados.

Podemos extraer estos genes claves con un ajuste significativo y filtrar la tabla con los sobreexpresados y los subexpresados.

```r
# Filtrado de  genes sobreexpresados
upregulated_genes <- annotated_results[annotated_results$logFC > 1 & annotated_results$adj.P.Val < 0.05

# Filtrado de genes subexpresados
downregulated_genes <- annotated_results[annotated_results$logFC < -1 & annotated_results$adj.P.Val < 0

# Número de genes sobre y subexpresados
cat("Genes sobreexpresados:", nrow(upregulated_genes), "\n")
```

```
## Genes sobreexpresados: 852
```

```r
cat("Genes subexpresados:", nrow(downregulated_genes), "\n")
```

```
## Genes subexpresados: 845
```

```r
head(upregulated_genes)
```

```
##         ProbeID   Symbol EntrezID          EnsemblID    logFC   AveExpr
## 23   1415871_at    Tgfbi    21810 ENSMUSG00000035493 2.007627  9.201326
## 26 1415897_a_at    Mgst1    56615 ENSMUSG00000008540 1.965525 10.511419
## 31 1415918_a_at     Tpi1    21991 ENSMUSG00000023456 1.045723  9.183993
## 32 1415922_s_at Marcksl1    17357 ENSMUSG00000047945 2.247863  8.835951
## 39   1415971_at   Marcks    17118 ENSMUSG00000069662 1.059885  9.487311
## 48 1416010_a_at     Ehd1    13660 ENSMUSG00000024772 1.329430  7.686606
##           t      P.Value    adj.P.Val          B
## 23 7.929338 7.702841e-08 7.461430e-07  8.0520807
## 26 4.680093 1.192818e-04 2.916295e-04  0.6664259
## 31 5.687709 1.079923e-05 3.956502e-05  3.0696143
## 32 6.050744 4.649379e-06 1.963755e-05  3.9169380
## 39 4.319818 2.854141e-04 6.254702e-04 -0.1998770
## 48 7.615855 1.478335e-07 1.246222e-06  7.3941362
```

```r
head(downregulated_genes)
```

```
##         ProbeID  Symbol EntrezID          EnsemblID     logFC   AveExpr
## 2    1415682_at    Xpo7    65246 ENSMUSG00000022100 -1.167193 10.225763
## 22 1415869_a_at  Trim28    21849 ENSMUSG00000005566 -1.092753  8.857636
## 24   1415872_at Hnrnph1    59013 ENSMUSG00000007850 -1.026358  8.503145
## 33   1415936_at   Bcar3    29815 ENSMUSG00000028121 -1.820287  7.686580
## 37   1415964_at    Scd1    20249 ENSMUSG00000037071 -2.271695  9.668634
## 46   1416007_at   Satb1    20230 ENSMUSG00000023927 -1.442595  9.276673
##            t      P.Value    adj.P.Val          B
## 2  -3.999312 6.205685e-04 1.224064e-03 -0.9667840
## 22 -6.862500 7.472861e-07 4.417117e-06  5.7590226
## 24 -4.725308 1.069401e-04 2.671473e-04  0.7751787
## 33 -8.074765 5.718058e-08 5.928378e-07  8.3527618
## 37 -9.267626 5.513715e-09 1.023327e-07 10.7109191
## 46 -4.761384 9.802176e-05 2.496206e-04  0.8619298
```

Análisis de la significación biológica

Una vez anotados los genes podemos intentar interpretar los resultados intentando determinar si las listas se encuentran enriquecidas o no en algunas categorias biológicas. Para ello utilizareoms un analisis de sobre-representación o un Gene Set Enrichment Analysis

```
# Instalar paquetes necesarios
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
BiocManager::install(c("clusterProfiler", "org.Mm.eg.db", "enrichplot", "DOSE"))
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: https://cran.rstudio.com/
```

```
## Bioconductor version 3.18 (BiocManager 1.30.25), R 4.3.1 (2023-06-16 ucrt)
```

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   `force = TRUE` to re-install: 'clusterProfiler' 'org.Mm.eg.db' 'enrichplot'
##   'DOSE'
```

```
## Installation paths not writeable, unable to update packages
##   path: C:/Program Files/R/R-4.3.1/library
##   packages:
##     boot, cluster, codetools, foreign, KernSmooth, lattice, mgcv, nlme, rpart,
##     spatial, survival
```

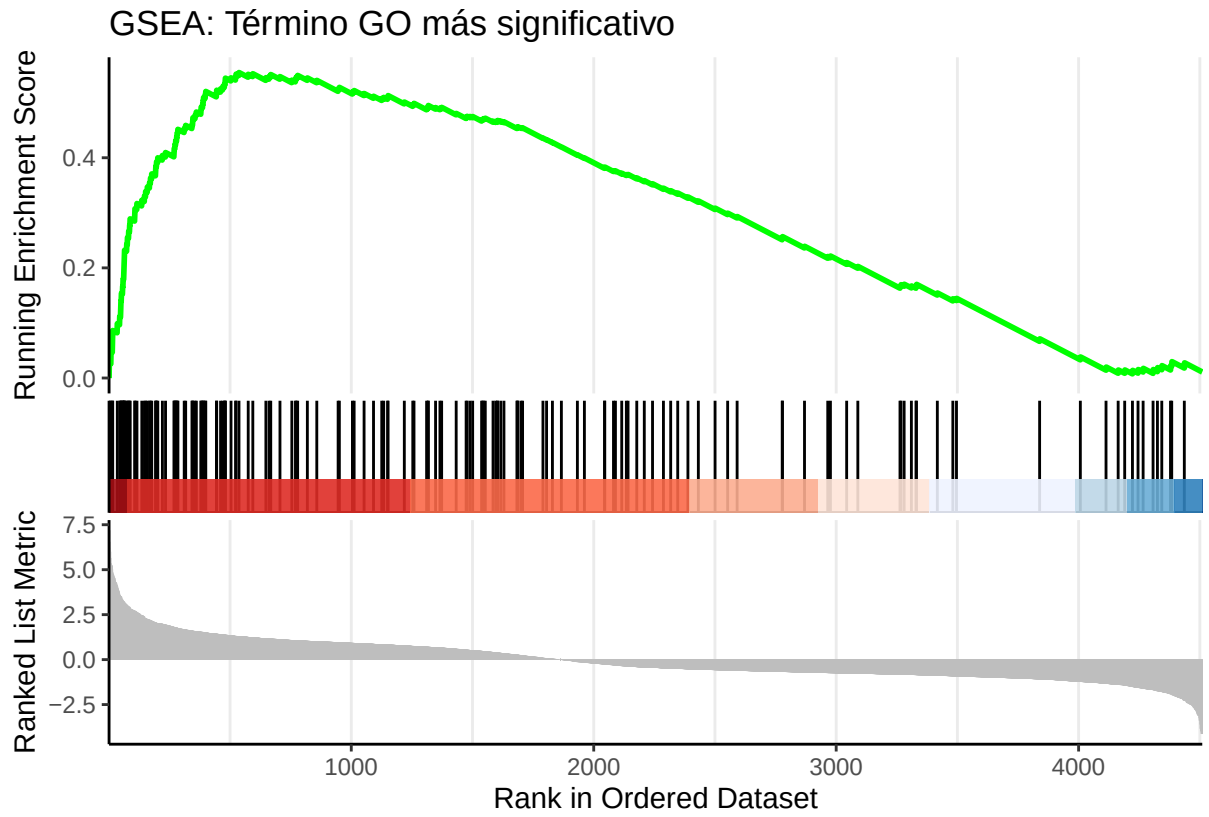
```
# Cargar paquetes
library(clusterProfiler)
library(org.Mm.eg.db)
library(enrichplot)
library(DOSE)


# Preparar el ranking de genes basado en logFC
gene_list <- annotated_results$logFC
names(gene_list) <- annotated_results$EntrezID
gene_list <- sort(gene_list, decreasing = TRUE)

# Realizar GSEA en GO
gsea_results <- gseGO(
  geneList = gene_list,
  OrgDb = org.Mm.eg.db,
  keyType = "ENTREZID",
  ont = "BP",  # Biología de procesos
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05
)
```

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

```
## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize,
## gseaParam, : There are duplicate gene names, fgsea may produce unexpected
## results.

## Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize =
## minSize, : For some pathways, in reality P-values are less than 1e-10. You can
## set the 'eps' argument to zero for better estimation.

## leading edge analysis...

## done...
```

```
# Visualizar resultados GSEA
gseaplot2(gsea_results, geneSetID = 1, title = "GSEA: Término GO más significativo")
```



Este gráfico evalua si un conjunto de genes esta sobreexpresado entre los genes altamente expresados, o subexpresados, en el rankin de genes. La línea verde representa la cumulación del enriquecimiento de los genes del conjunto a lo largo del ranking, un pico alto en la curva indica que los genes del conjunto están enriquecidos.

Cada marca vertical negra representa un gen en el conjunto. Los genes estan ordenados por su posicion en el ranking (a partir de logFC)

La barra de color roja representa genes sobreexpresados y la de color azul genes subexpresados.

En la gráfica obtenida, como el pico ocurre en la parte izquierda, nos indica que está enriquecido entre los genes sobreexpresados. Los genes del conjunto tienen un papel relevante en la condición de interes y estan sobreexpresados en el grupo experimental.

A partir de aqui, podemos expresar los resultados del GSEA. Identificamos el nombre del termino GO enriqucido y consultamos el listado de genes asociados al termino, para posteriormente realizar un analisis gráfico a partir de un dotplot

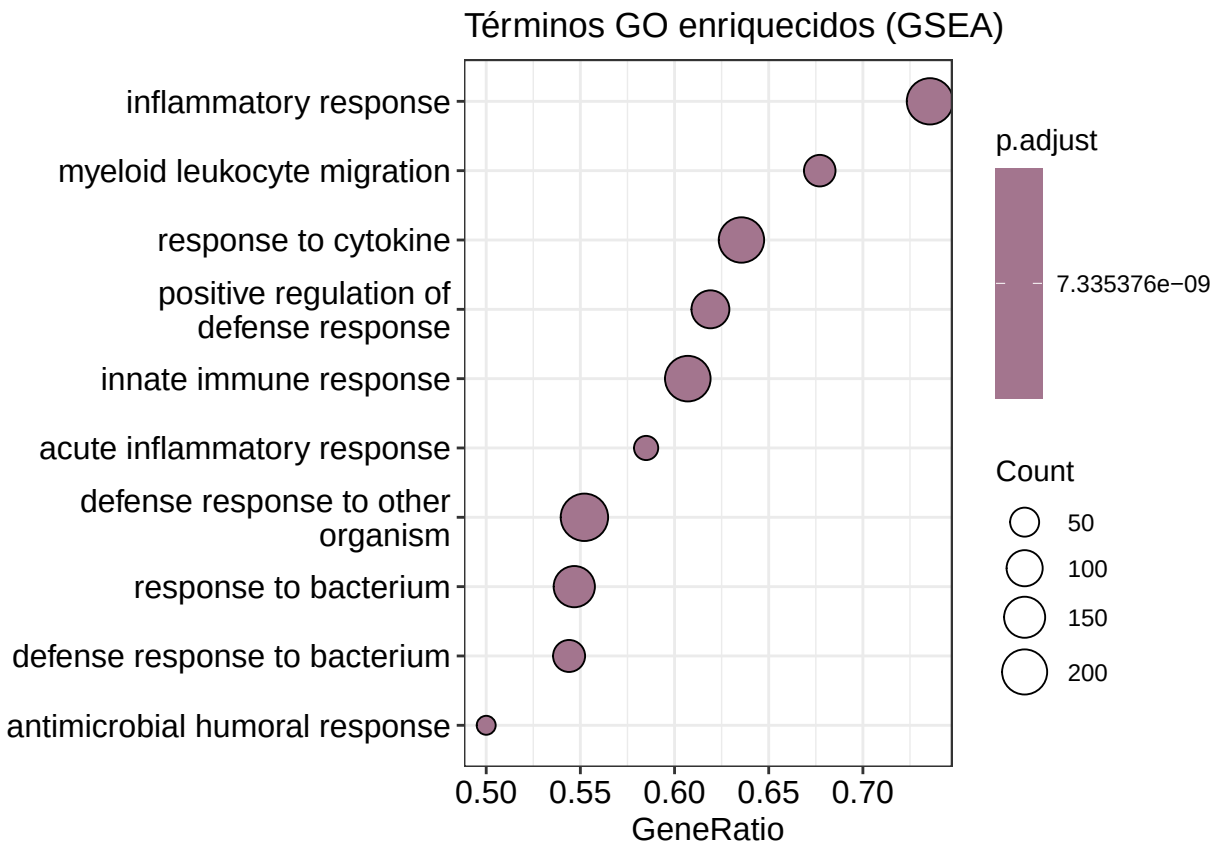
```
# Términos GO más significativos
head(as.data.frame(gsea_results))
```

```
##                    ID                    Description setSize enrichmentScore
## GO:0042742 GO:0042742  defense response to bacterium     125       0.6152095
## GO:0006954 GO:0006954          inflammatory response     295       0.5272803
## GO:0009617 GO:0009617          response to bacterium     278       0.5277508
## GO:0097529 GO:0097529    myeloid leukocyte migration      96       0.6089979
## GO:0045087 GO:0045087         innate immune response     341       0.4876024
## GO:0019730 GO:0019730 antimicrobial humoral response      48       0.6867552
##                 NES pvalue     p.adjust       qvalue rank
## GO:0042742 3.199731  1e-10 7.335376e-09 4.997403e-09  537
## GO:0006954 3.130205  1e-10 7.335376e-09 4.997403e-09 1164
## GO:0009617 3.114154  1e-10 7.335376e-09 4.997403e-09  793
## GO:0097529 3.036526  1e-10 7.335376e-09 4.997403e-09  785
## GO:0045087 2.936435  1e-10 7.335376e-09 4.997403e-09  951
## GO:0019730 2.933068  1e-10 7.335376e-09 4.997403e-09  654
##                              leading_edge
## GO:0042742 tags=39%, list=12%, signal=36%
## GO:0006954 tags=53%, list=26%, signal=43%
## GO:0009617 tags=40%, list=18%, signal=36%
## GO:0097529 tags=51%, list=17%, signal=43%
## GO:0045087 tags=43%, list=21%, signal=38%
## GO:0019730 tags=38%, list=15%, signal=33%
##
## GO:0042742
## GO:0006954 20210/104183/12655/16178/12475/17394/18405/16365/20310/20209/14289/22038/15937/22038/1265
## GO:0009617
## GO:0097529
## GO:0045087                                                  16819/17002/100034251/66107/68891/12475/16365/6
## GO:0019730
```

```
# Genes del término más significativo
gsea_genes <- gsea_results@result$geneID[1]  # Genes del primer término
cat("Genes del término enriquecido:", gsea_genes, "\n")
```

```
## Genes del término enriquecido:
```

```
# Dotplot para los principales términos GO enriquecidos
dotplot(gsea_results, showCategory = 10) + ggtitle("Términos GO enriquecidos (GSEA)")
```



El eje de las Y nos muestra los términos GO más significativos según el análisis, el eje de las X nos representa la proporcion de genes en el término GO enriquecido y el tamaño del punto nos indica el número de genes en el conjunto que estan asociados al término GO.

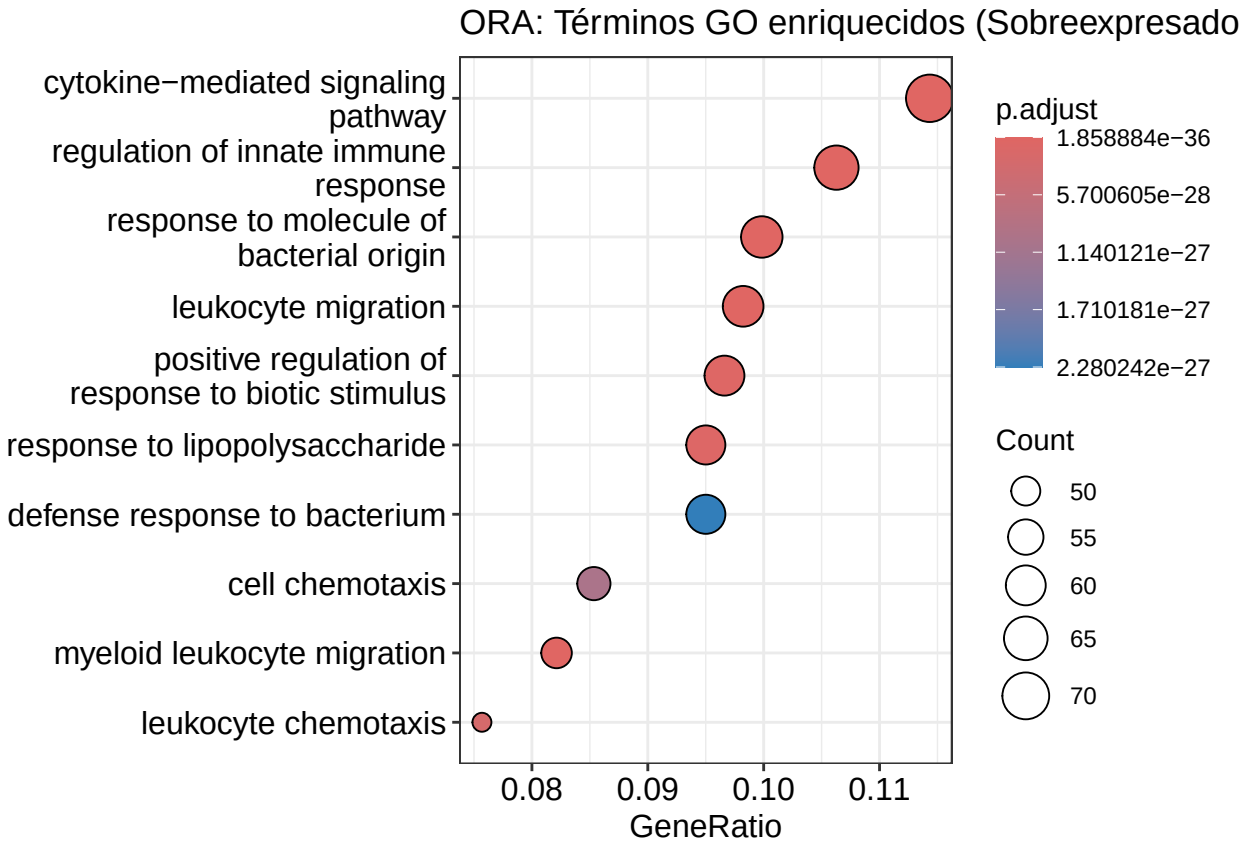
Podemos realizar, como complemento, un analisis de sobre-representación (ORA), enfocado únicamente en los genes filtrados como significativamente sobreexpresados o subexpresados.

```
# Extraemos los Entrez IDs de los genes significativos
upregulated_entrez <- upregulated_genes$EntrezID
downregulated_entrez <- downregulated_genes$EntrezID

# ORA para genes sobreexpresados en "Biological Process" (BP)
go_up_BP <- enrichGO(
  gene = upregulated_entrez,
  OrgDb = org.Mm.eg.db,
  keyType = "ENTREZID",
  ont = "BP",  # Ontología: Biological Process
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05,
  qvalueCutoff = 0.05
)

# ORA para genes subexpresados en "Biological Process" (BP)
go_down_BP <- enrichGO(
  gene = downregulated_entrez,
  OrgDb = org.Mm.eg.db,
  keyType = "ENTREZID",
```

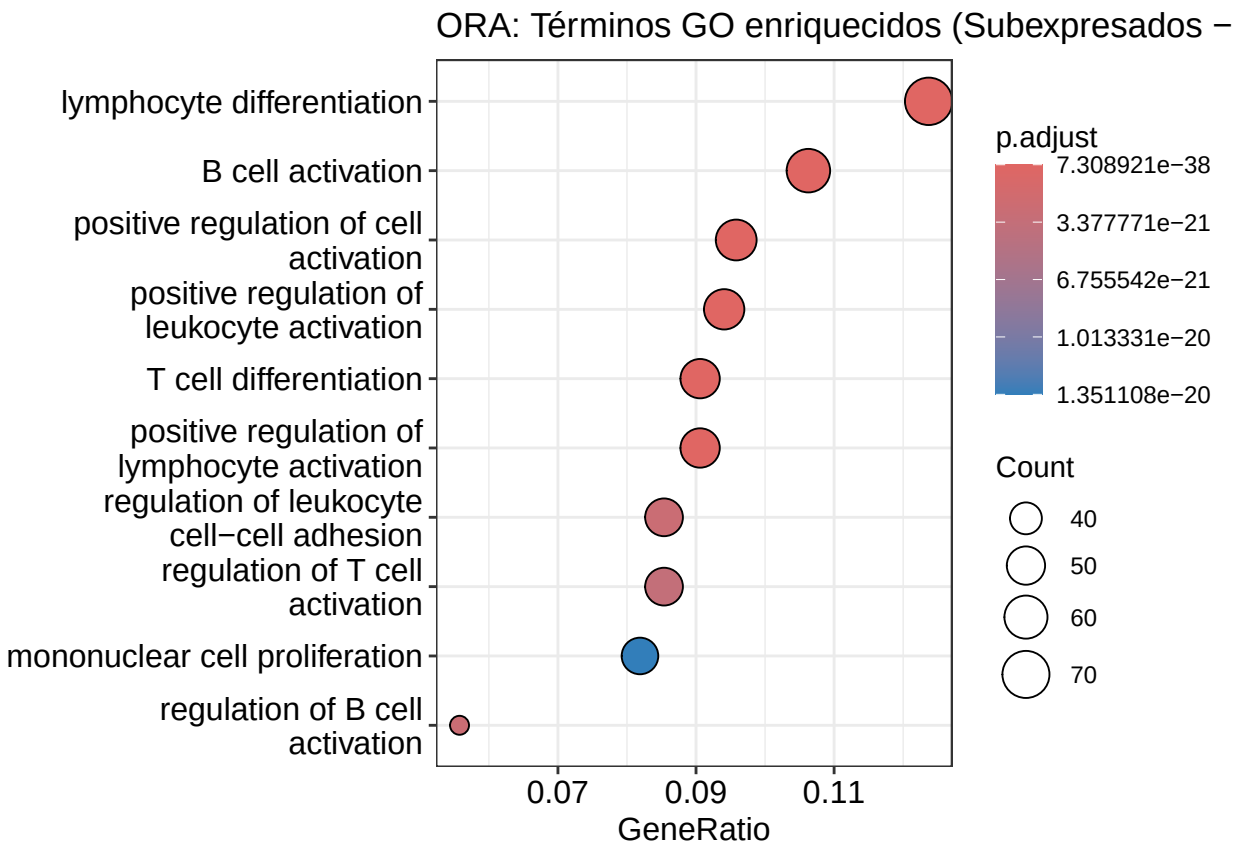
```
  ont = "BP",
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05,
  qvalueCutoff = 0.05
)
```

Ahora, vamos a visualizar los resultados:

```
# Términos GO enriquecidos para genes sobreexpresados
dotplot(go_up_BP, showCategory = 10) + ggtitle("ORA: Términos GO enriquecidos (Sobreexpresados - BP)")
```




```
# Términos GO enriquecidos para genes subexpresados
dotplot(go_down_BP, showCategory = 10) + ggtitle("ORA: Términos GO enriquecidos (Subexpresados - BP)")
```



También podemos realizar el análisis de enriquecimiento para rutas KEGG para estudiar las rutas metabólicas y moleculares en las que están implicados los genes del experimento.

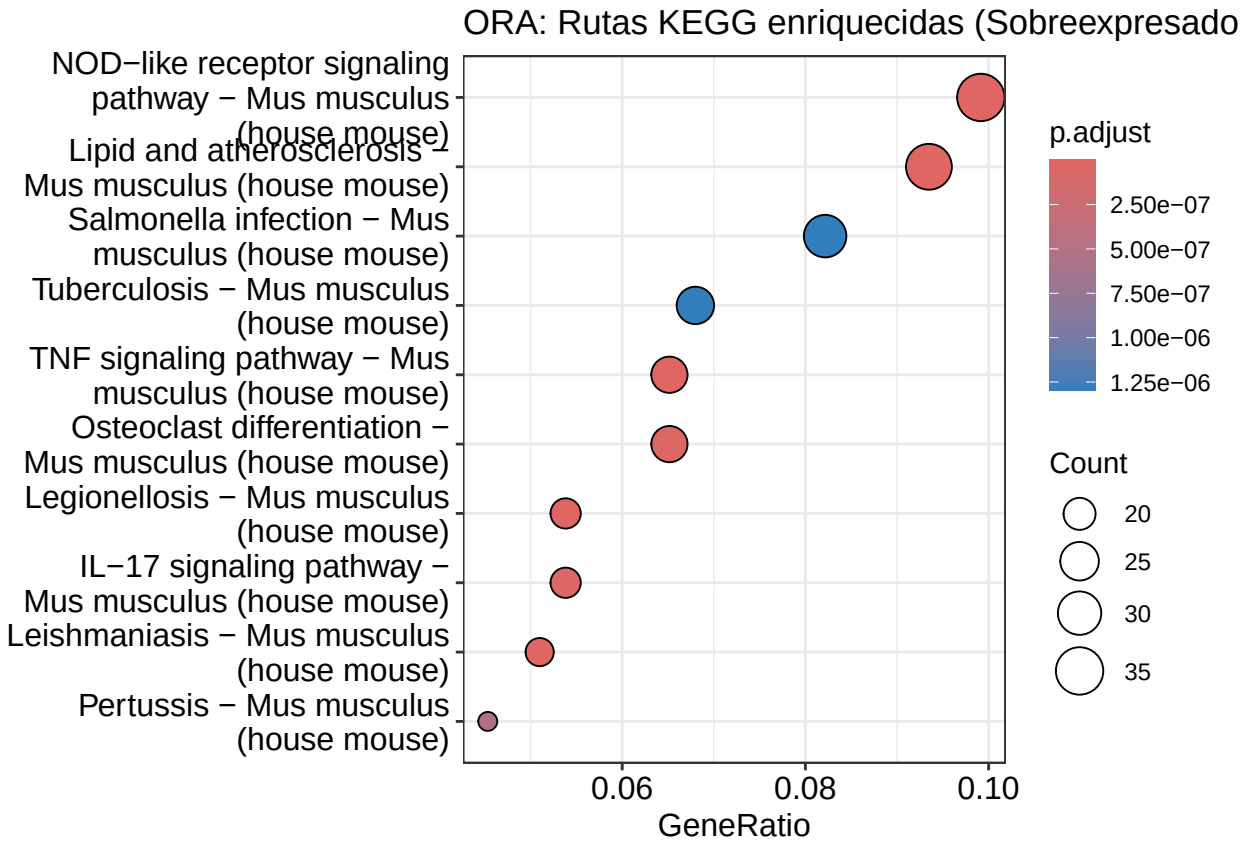
```
# ORA para genes sobreexpresados en rutas KEGG
kegg_up <- enrichKEGG(
  gene = upregulated_entrez,
  organism = "mmu",  # Organismo: mmu = Mus musculus
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05,
  qvalueCutoff = 0.05
)
```

```
## Reading KEGG annotation online: "https://rest.kegg.jp/link/mmu/pathway"...
```

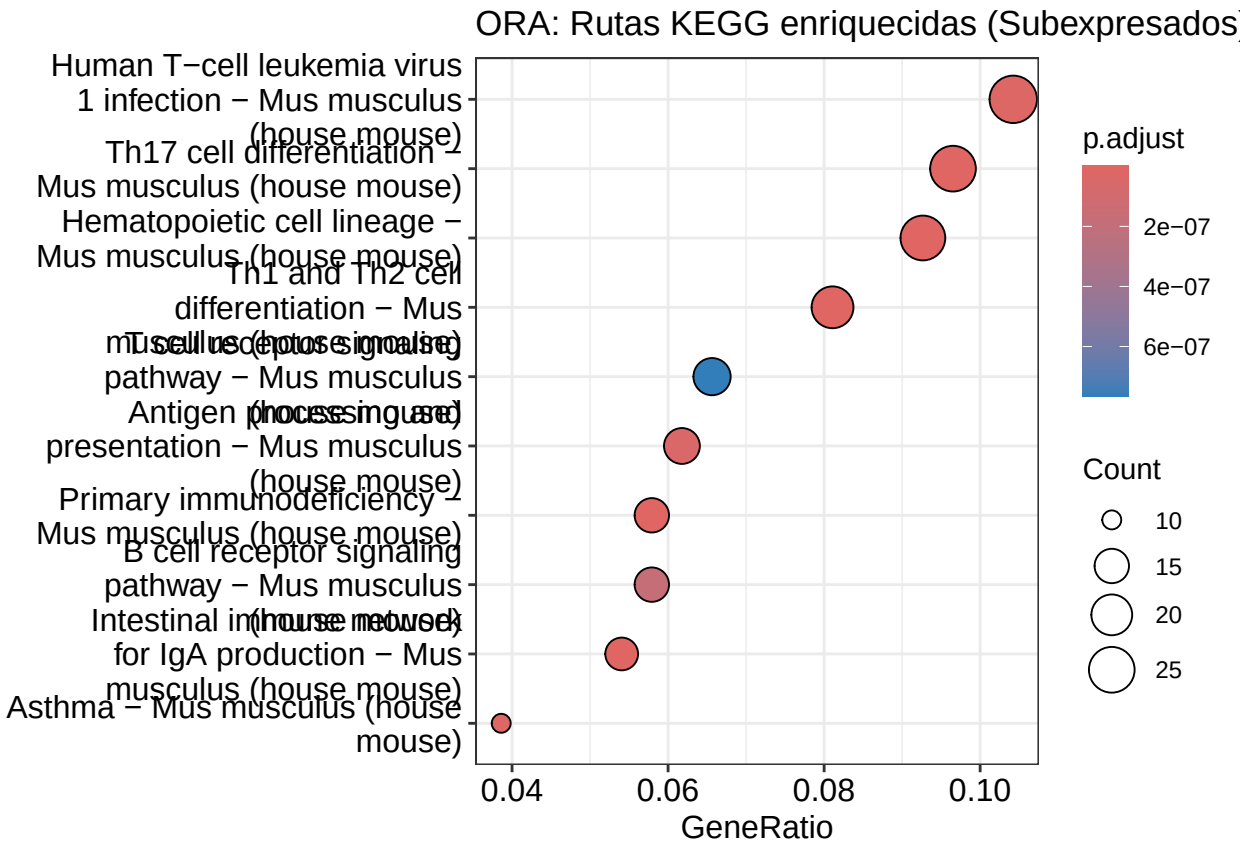
```
## Reading KEGG annotation online: "https://rest.kegg.jp/list/pathway/mmu"...
```

```
# ORA para genes subexpresados en rutas KEGG
kegg_down <- enrichKEGG(
  gene = downregulated_entrez,
  organism = "mmu",
  pAdjustMethod = "BH",
  pvalueCutoff = 0.05,
  qvalueCutoff = 0.05
)
```

```r
# Visualización de rutas KEGG
dotplot(kegg_up, showCategory = 10) + ggtitle("ORA: Rutas KEGG enriquecidas (Sobreexpresados)")
```




```r
dotplot(kegg_down, showCategory = 10) + ggtitle("ORA: Rutas KEGG enriquecidas (Subexpresados)")
```



Conclusiones finales

Este proyecto ilustra cómo los datos de expresión génica, combinados con herramientas bioinformáticas y estadísticas avanzadas, pueden proporcionar una comprensión detallada de las respuestas biológicas a infecciones bacterianas y tratamientos. Los resultados destacan genes y rutas biológicas importantes que podrían ser objetivos potenciales para futuras investigaciones. Además, se demuestra la utilidad de herramientas como limma y clusterProfiler para realizar análisis completos y visualmente informativos.

El trabajo no solo cumple con los objetivos planteados, sino que también refuerza la importancia de la bioinformática en la investigación biomédica y la capacidad de analizar grandes volúmenes de datos de manera eficiente y significativa.

Los resultados sugieren que los procesos inmunológicos están fuertemente activados en las muestras infectadas, con una respuesta adaptativa modulada por los tratamientos antibióticos. Linezolid y vancomicina afectan distintas vías inmunológicas, lo que destaca su impacto diferencial en la regulación de la respuesta del huésped frente a la infección.

Ambos antibióticos demostraron modular genes que son críticos en la inmunomodulación, aunque la magnitud y el número de genes afectados fueron diferentes.

Linezolid parece tener un impacto más marcado en la regulación de las rutas de señalización inmunológica, mientras que vancomicina mostró un efecto significativo en genes relacionados con respuestas antimicrobianas directas.