INFORME PEC 1 - Análisis de datos ómicos

Lucía López Ayllón

Tabla de contenidos

LINK AL REPOSITORIO: https://github.com/lucialopez7/Lopez-Ayllon-Lucia-PEC1

1. Objetivos.

El objetivo de este ejercicio es familiarizarse con el uso de Bioconductor y GitHub, con el fin de poder trabajar mediante las clases destinadas al almacenacenamiento datos ómicos para la exploración multivariante de datos. En este caso llevaremos a cabo un análisis de datos metobolómicos y, para ello, descargaremos un archivo .txt de la web de *metabolomicsWorkbench* y lo convertiremos en un objeto de tipo *SummarizedExperiment* para su posterior análisis. La razón por la que emplearemos objetos de tipo *SummarizedExperiment* y no de tipo *ExpressionSet* se explica a continuación.

SummarizedExperiment es una estructura de datos desarrollada en el ecosistema de Bioconductor (R) para almacenar y manejar datos de experimentos de alto rendimiento, como los obtenidos mediante tecnologías de expresión génica (por ejemplo, RNA-seq). Esta clase permite organizar los datos de expresión en una o más matrices (denominadas assays), donde las filas suelen representar genes o regiones genómicas, y las columnas, muestras biológicas. Además, incorpora de forma estructurada los metadatos asociados tanto a las filas (rowData) como a las columnas (colData), así como información general del experimento (metadata). Esta integración garantiza una coherencia interna entre los datos y su contexto biológico, facilitando análisis reproducibles y bien documentados.

Por otro lado, la clase *ExpressionSet*, perteneciente al paquete Biobase, fue una de las primeras estructuras diseñadas para este tipo de análisis, principalmente orientada a datos de microarrays. Aunque sigue siendo funcional, su diseño es menos flexible: solo admite una única matriz de datos, y su manejo de metadatos es más limitado al estar basado en la clase *AnnotatedDataFrame*. Además, no ofrece soporte directo para integrar rangos genómicos u otras representaciones modernas. En consecuencia, *SummarizedExperiment* ha sido adoptado como el nuevo estándar en Bioconductor para representar datos experimentales complejos.

2. Métodos

En primer lugar, seleccionaremos un dataset metabolómico de la web de *metabolomicsWorkbench* (*https://www.metabolomicsworkbench.org/*) y lo descargaremos.

En este caso, hemos seleccionado el estudio "Differential Metabolites and Disturbed Metabolic Pathways Associated with chlorpromazine Poisoning" de la Universidad de Medicina de Hebei, con el identificador

ST001739. Este estudio se centra en el análisis de las vías alteradas y los metabolitos diferenciales en la intoxicación letal por clorpromzaína en relación con el grupo de control y el de intoxicación no letal por clorpromazina.

El archivo descargado, "**ST001739_AN002832.txt**", se encuentra en formato de texto. Dentro de él, encontramos de forma compacta toda la información que debemos dividir mediante el uso de RStudio en los 3 conjuntos de datos que formarán nuestro objeto de la clase *SummarizedExperiment*.Los objetos de la clase *SummarizedExperiment* presentan una estructura organizada que facilita el manejo de experimentos biológicos. Está diseñado para mantener datos y metadatos bien organizados y relacionados entre sí.

El objeto, al que nombraremos como "se", estará formado por 3 elementos: assays, rowData y colData.

- Los datos principales o **assays**: será el elemento principal formado por una matriz de datos. Las columnas se corresponderán con las distintas muestras y las filas tendrán nombres de distintos metabolitos. La tabla estará rellena con las mediciones del nivel de expresión de cada metabolito para cada muestra.
- Los metadatos de las filas o **rowData**: cada fila contendrá el nombre de cada metabolito de "assays". En las columnas encontraremos información adicional sobre cada metabolito.
- Los metadatos de las columnas o **colData** contiene en sus filas cada uno de las muestras. En las columnas se proporciona información adicional sobre cada muestra.

Una vez el objeto SummarizedExperiment haya sido creado, llevaremos a cabo un análisis exploratorio para obtener una visión general sobre el dataset descargado.

3. Resultados.

3.1. Preparación del archivo para su análisis.

El código empleado para obtener los tres archivos correspondientes a assays, rowData, y colData para después crear un objeto SummarizeExperiment a partir de ellos es el siguiente:

```
# 1) Definimos un vector con los paquetes requeridos. Se comprueba si cada paquete está ya
 ↪  instalado y, si no, se instala.

required_pkgs <- c("metabolomicsWorkbenchR", "fobitools", "SummarizedExperiment",
                   "dplyr", "rvest", "xlsx", "tibble")

for (pkg in required_pkgs) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    if (pkg %in% c("metabolomicsWorkbenchR", "fobitools", "SummarizedExperiment")) {
      BiocManager::install(pkg)
    } else {
      install.packages(pkg)
    }
  }
  library(pkg, character.only = TRUE)
}

# ------------------------------------------------------------------------------

# 2) Se lee el contenido del archivo ST001739_AN002832.txt descargado de *metabolomicsWorkbench*:

file_lines <- readLines("C:/Users/lucia/Desktop/análisis de datos ómicos/pec
 ↪  1/ST001739_AN002832.txt")


# 3) EXTRACCIÓN DE LOS DATOS NUMÉRICOS DE LOS METABOLITOS

# Buscamos la línea donde se inician los datos de metabolitos
start_line <- grep("MS_METABOLITE_DATA_START", file_lines)
```

```r
if(length(start_line) == 0){
  stop("No se encontró la línea 'MS_METABOLITE_DATA_START'")
}

# Como los datos se extienden hasta el siguiente marcador "#", se calcula
# el índice de esa línea y se usa para delimitar la selección

end_line <- start_line + which(grepl("^#", file_lines[(start_line+1):length(file_lines)]))[1]

# Extraemos las líneas entre el marcador de inicio y el marcador de final
met_data_lines <- file_lines[(start_line + 1):(end_line - 1)]

# Convertimos el bloque de texto extraido a una tabla
met_data <- read.table(text = met_data_lines, header = TRUE, sep="\t", stringsAsFactors = FALSE,
 ↪  check.names = FALSE, fill = TRUE)


# 4) PROCESAMOS LA INFORMACIÓN DE LAS MUESTRAS (colData)

# Buscamos las líneas que empiecen por SUBJECT_SAMPLE_FACTORS
# Extraemos el ID de la muestra, el fenotipo y el género

sample_lines <- grep("^SUBJECT_SAMPLE_FACTORS", file_lines, value = TRUE)
sample_info <- lapply(sample_lines, function(line) {
  parts <- unlist(strsplit(line, "\t"))
  sample_id <- parts[3]
  phenotype <- sub("Phenotype:", "", parts[4])
  gender <- sub("Other=", "", parts[5])
  return(c(SampleID = sample_id, Phenotype = phenotype, Gender = gender))
})

# Se crea un data frame llamado sample_metadata con la información de las muestras

sample_metadata <- as.data.frame(do.call(rbind, sample_info), stringsAsFactors = FALSE)
rownames(sample_metadata) <- sample_metadata$SampleID

# 4) PROCESAMOS LA SECCIÓN DE DATOS NUMÉRICOS (assays)

# Tomamos todas las filas de met_data excepto las dos primeras, pues contienen encabezados

raw_metabolite_names <- met_data[-c(1,2), 1]
metabolite_names <- trimws(raw_metabolite_names)
exprs_matrix <- as.matrix(met_data[-c(1,2), -1])
exprs_matrix <- apply(exprs_matrix, 2, as.numeric)
rownames(exprs_matrix) <- metabolite_names
colnames(exprs_matrix) <- colnames(met_data)[-1]

# Reordenar sample_metadata para que sus rownames (IDs de muestra) coincidan exacramente con los
 ↪  colnames de exprs_matrix
sample_metadata <- sample_metadata[colnames(exprs_matrix), , drop = FALSE]


# 4) PROCESAMOS LOS METADATOS DE LOS METABOLITOS (rowData)

# Extraemos la correspondiente sección del archivo

met_start <- grep("METABOLITES_START", file_lines)
met_end <- grep("METABOLITES_END", file_lines)
metabolites_block <- file_lines[(met_start + 1):(met_end - 1)]
```

```r
# La primera línea es la cabecera con nombres de columnas
headers <- strsplit(metabolites_block[1], "\t")[[1]]
data_lines <- metabolites_block[-1]
if (grepl("^Phenotype", data_lines[1])) {
  data_lines <- data_lines[-1]
}
met_info <- read.table(text = data_lines, header = FALSE, sep = "\t",
                       stringsAsFactors = FALSE, fill = TRUE)
colnames(met_info) <- headers
met_info[,1] <- trimws(met_info[,1])
rownames(met_info) <- met_info[, 1]

# Reemplazar celdas vacías por NA
met_info[met_info == ""] <- NA

# Alinear metabolitos: quedarse solo con los nombres comunes
common_metabolites <- intersect(rownames(exprs_matrix), rownames(met_info))
if(length(common_metabolites) < length(rownames(exprs_matrix)) ||
   length(common_metabolites) < length(rownames(met_info))) {
  warning("No se encontraron coincidencias exactas para todos los metabolitos. Se usarán solo los
   ↪  comunes.")
}
exprs_matrix <- exprs_matrix[common_metabolites, , drop = FALSE]
met_info <- met_info[common_metabolites, , drop = FALSE]


# 5) CREAMOS EL OBJETO SummarizedExperiment

library(SummarizedExperiment)
se <- SummarizedExperiment(
  assays = list(counts = exprs_matrix),
  colData = sample_metadata,
  rowData = met_info)

# 6) VISUALIZAMOS EL NUEVO OBJETO
se


# 7) GENERAMOS ARCHIVOS CSV Y XLSX

# Archivos CSV para cada objeto
write.table(assay(se), sep = ";", file = "features.csv", quote = FALSE, row.names = TRUE)
write.table(colData(se), sep = ";", file = "metadata.csv", quote = FALSE, row.names = TRUE)
write.table(rowData(se), sep = ";", file = "metaboliteNames.csv", quote = FALSE, row.names = TRUE)

# Único archivo Excel con un objeto en cada hoja
library(xlsx)
write.xlsx(assay(se), file = "ST001739_RESULTADO.xlsx", sheetName = "features",
           col.names = TRUE, row.names = TRUE, append = FALSE)
write.xlsx(colData(se), file = "ST001739_RESULTADO.xlsx", sheetName = "metadata",
           col.names = TRUE, row.names = FALSE, append = TRUE)
write.xlsx(rowData(se), file = "ST001739_RESULTADO.xlsx", sheetName = "metaboliteNames",
           col.names = TRUE, row.names = FALSE, append = TRUE)


# ------------------------------------------------------------------------------
# 8) RESUMEN COMPLETO DEL OBJETO SummarizedExperiment
library(knitr)
```

```r
# Dimensiones generales
cat("## Resumen de dimensiones del objeto `SummarizedExperiment`\n\n")
cat(paste0("- Número total de metabolitos (features): ", nrow(se), "\n"))
cat(paste0("- Número total de muestras (samples): ", ncol(se), "\n\n"))

# Dimensiones específicas de cada componente
cat("### Dimensiones de cada componente:\n")
cat(paste0("- assays (datos de expresión): ", paste(dim(assay(se)), collapse = " x "), "\n"))
cat(paste0("- colData (metadata de muestras): ", paste(dim(colData(se)), collapse = " x "), "\n"))
cat(paste0("- rowData (metadata de metabolitos): ", paste(dim(rowData(se)), collapse = " x "),
↪  "\n\n"))

# --------------------------
# Vista previa de assays (expresión de metabolitos)
knitr::kable(
  assay(se)[1:5, 1:5],
  caption = "Vista previa de los datos de expresión (assays)",
  booktabs = TRUE
)
# --------------------------
# Vista previa de colData (información de muestras)
knitr::kable(
  head(as.data.frame(colData(se))),
  caption = "Información de las muestras (colData)",
  booktabs = TRUE
)
# -------------------------

# Vista previa de rowData (información de metabolitos)
knitr::kable(
  head(as.data.frame(rowData(se))[, 2:3]),
  caption = "Vista previa de rowData (primeras filas y columnas)",
  booktabs = TRUE
)
```

```
## class: SummarizedExperiment
## dim: 319 30
## metadata(0):
## assays(1): counts
## rownames(319): (+)-Calycanthine (±)-2,6-Dimethyl-5-heptenal ... Xanthine Zeranol
## rowData names(11): metabolite_name Human Metabolome Database ... mzCloud Library
##   Match: Reference MS2
## colnames(30): FCS1 FCS2 ... MT4 MT5
## colData names(3): SampleID Phenotype Gender
## ## Resumen de dimensiones del objeto `SummarizedExperiment`
##
## - Número total de metabolitos (features): 319
## - Número total de muestras (samples): 30
##
## ### Dimensiones de cada componente:
## - assays (datos de expresión): 319 x 30
## - colData (metadata de muestras): 30 x 3
## - rowData (metadata de metabolitos): 319 x 11
```

Table 2: Vista previa de los datos de expresión (assays)

	FCS1	FCS2	FCS3	FCS4	FCS5
(+)-Calycanthine	66359494.9	55879052.2	66716186	64727640.5	65195655
(±)-2,6-Dimethyl-5-heptenal	43248910.2	27642884.1	36918162	28322137.4	28715867
(±)-2-Hydroxy-4-(methylthio)butanoic acid	1875692.5	1408317.9	2640152	1368760.5	1467932
(±)-Camphoric acid	821651.3	1542508.7	6116601	4134125.5	3857492
(±)-Furaneol	632463.4	641911.9	1049808	271303.4	1195391

Table 3: Información de las muestras (colData)

	SampleID	Phenotype	Gender
FCS1	FCS1	Lethal chlorpromazine poisoning	Female
FCS2	FCS2	Lethal chlorpromazine poisoning	Female
FCS3	FCS3	Lethal chlorpromazine poisoning	Female
FCS4	FCS4	Lethal chlorpromazine poisoning	Female
FCS5	FCS5	Lethal chlorpromazine poisoning	Female
MCS1	MCS1	Lethal chlorpromazine poisoning	Male

Table 4: Vista previa de rowData (primeras filas y columnas)

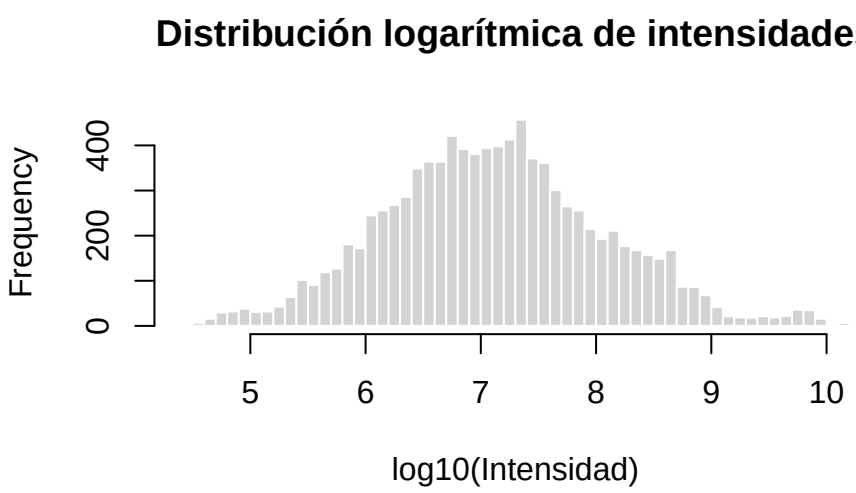
	Human.Metabolome.Database	KEGG
(+)-Calycanthine	HMDB0029561	C10573
(±)-2,6-Dimethyl-5-heptenal	HMDB0031834	NA
(±)-2-Hydroxy-4-(methylthio)butanoic acid	HMDB0037115	NA
(±)-Camphoric acid	HMDB0034491	NA
(±)-Furaneol	HMDB0040594	NA
(2S)-2-[(2-Methyl-2-sulfanylpropanoyl)amino]-3-sulfanylpropanoic acid	NA	NA

3.2. Análisis exploratorio de los datos.

Histograma logarítmico de los valores de intensidad de los metabolitos

Nos permite visualizar la distribución general de los valores de intensidad de los metabolitos. Puesto que los datos de intensidades en metabolómica suelen tener una distribución altamente sesgada, se aplica una transformación logarítmica.

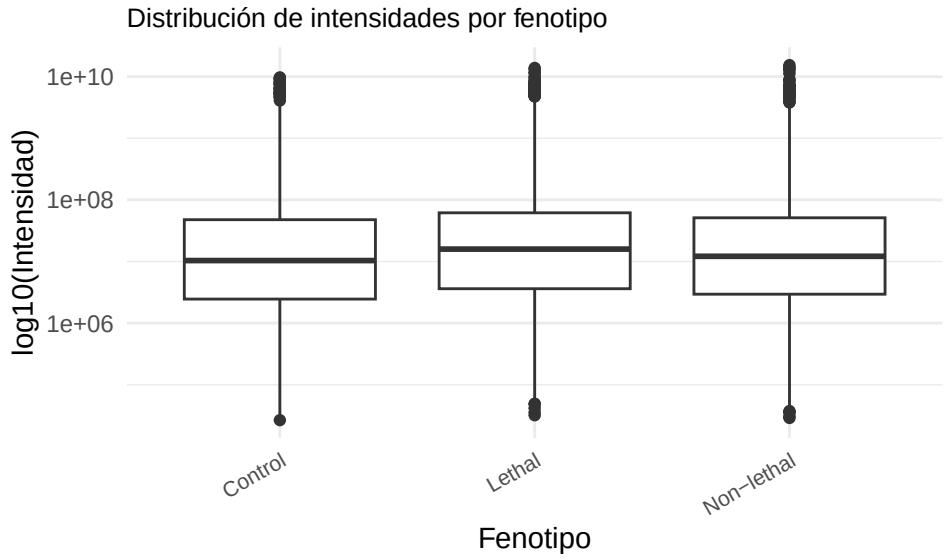
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 2.660e+04 2.962e+06 1.246e+07 1.760e+08 5.322e+07 1.540e+10
```



Observamos una distribución aproximadamente normal.

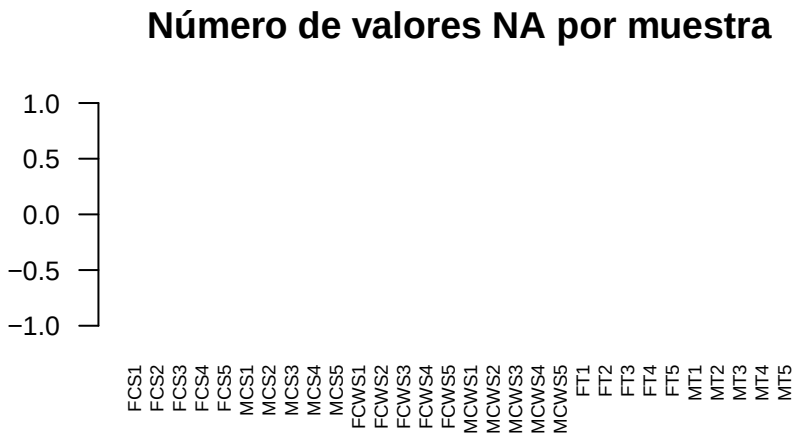
Boxplot de las intensidades de los metabolitos en función de los fenotipos de las muestras

Nos permite comprarar la distribución de las intensidades según el grupo de muestra y comprobar si los fenotipos tienen patrones distintos en la expresión metabólica.



Observamos distribuciones muy similares entre los grupos, por lo que podría no haber una alteración generalizada del metabolismo asociada al fenotipo. Sería necesario analizar metabolito por metabolito para ver si ciertos metabolitos específicos podrían estar diferencialmente expresados.

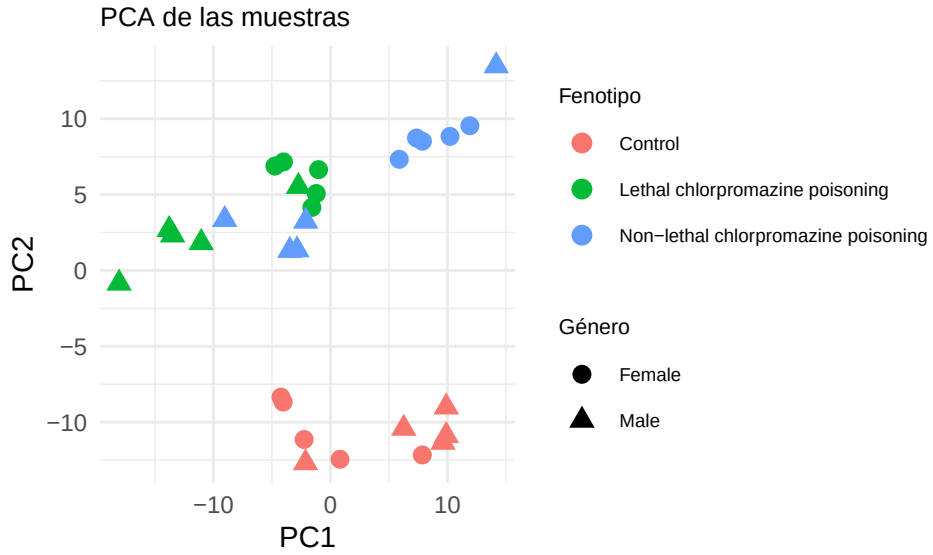
Análisis del número de valores faltantes por muestra



No existen valores faltantes en ninguna muestra.

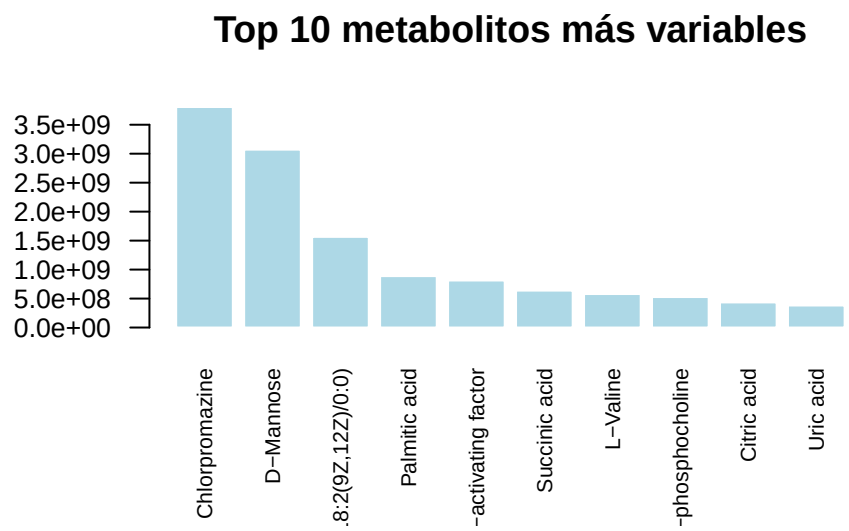
Análisis de Componentes Principales (PCA)

Es útil para explorar agrupamientos naturales de las muestras y patrones globales, como la separación de muestras por genotipo o la presencia de outliers.



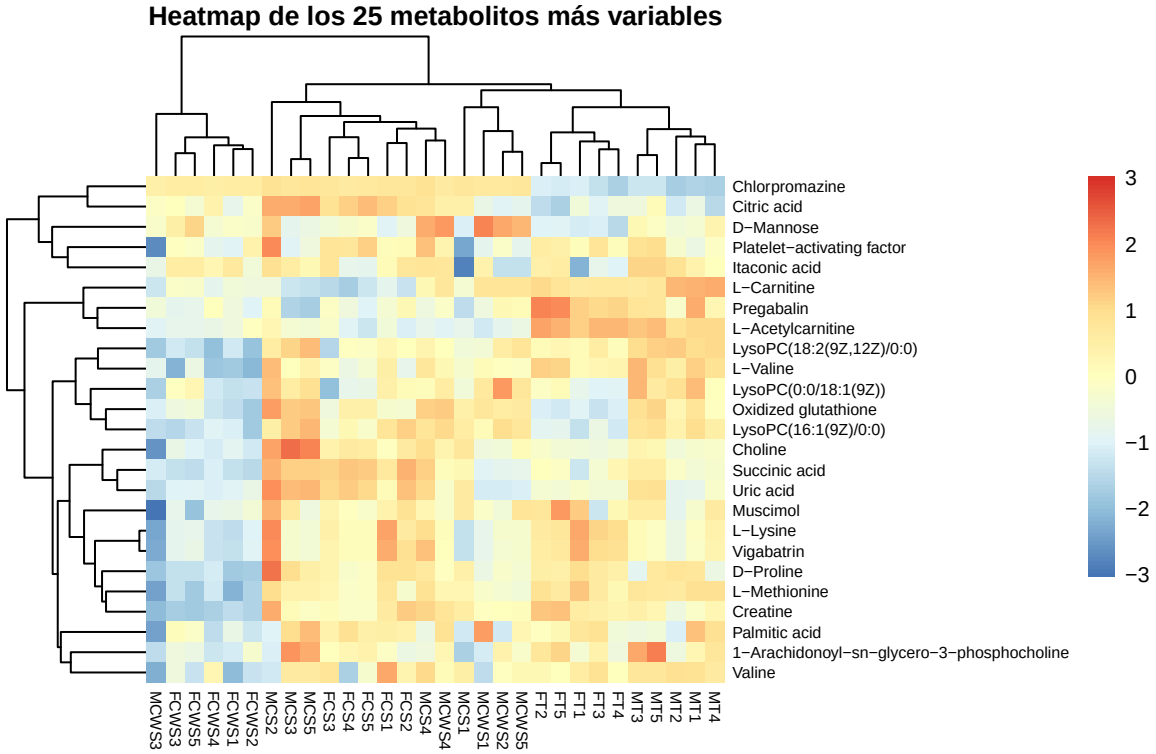
En el análisis PCA, la primera componente principal (PC1) representa la dirección de mayor variación en el conjunto de datos y permite observar las diferencias más destacadas entre muestras. La segunda componente (PC2) es ortogonal a la primera y recoge una fuente secundaria de variabilidad. En este caso, PC1 separa principalmente las muestras letales, mientras que PC2 diferencia controles y no letales, lo que refleja cómo distintas condiciones afectan de forma diferencial el perfil metabólico.

Gráfico de barras que muestra los 10 metabolitos con una intensidad más variable



Mapa de calor basado en los metabolitos más variables

Nos permite conocer si ciertas muestras comparten perfiles similares.



Muchas muestras en una misma zona del del mapa de calor comparten perfiles comunes, lo que es consistente con lo observado en el PCA. Esto sugiere que ciertos grupos de muestras comparten perfiles metabólicos similares, posiblemente relacionados con su fenotipo o condición experimental

4. Discusión.

El análisis exploratorio de los datos metabolómicos muestra que hay ciertos patrones en cómo se expresan los metabolitos según el fenotipo de las muestras. Aplicar la transformación logarítmica fue un paso clave para normalizar los datos y poder visualizarlos y analizarlos mejor.

Aunque los boxplots no muestran grandes diferencias generales en los niveles de expresión entre los distintos grupos, el PCA sí revela una separación clara entre ellos, sobre todo entre los casos letales y no letales. Esto sugiere que los perfiles metabólicos sí cambian en función del fenotipo, aunque no de forma global en todos los metabolitos.

El heatmap con los metabolitos más variables también ayudó a ver algunos compuestos concretos que destacan (como chlorpromazine o citric acid), y que podrían estar relacionados con el tipo de intoxicación. Además, como no hay valores perdidos en el dataset, podemos confiar en la calidad de los datos para seguir haciendo análisis más avanzados.

Lo ideal sería continuar este análisis analizando la intensidad de cada metabolito en particular.

5. Conclusiones.

- RStudio es una potente herramienta a utilizar durante nuestros análisis estadísticos gracias al uso de Bioconductor.
- Los objetos de tipo SummarizedExpression facilitan el procesamiento de archivos de texto y su organización en una estructura más ordenada.

6. Referencias.

a) https://www.bioconductor.org/packages/release/bioc/vignettes/SummarizedExperiment/inst/doc/SummarizedExperiment.html

b) https://www.bioconductor.org/help/course-materials/2006/biocintro_oct/labs/ExpressionSet/ExpressionSetIntro-solved.pdf

c) https://www.metabolomicsworkbench.org/data/DRCCMetadata.php?Mode=Study&StudyID=ST001739&StudyType=MS&ResultType=1

ANEXO 1 - Código correspondiente a los gráficos

Histograma logarítmico de los valores de intensidad de los metabolitos

```
expr <- assay(se)

summary(as.vector(expr))

hist(log10(expr), breaks = 50,
     main = "Distribución logarítmica de intensidades",
     xlab = "log10(Intensidad)",
     col = "lightgray", border = "white")
```

Boxplot de las intensidades de los metabolitos en función de los fenotipos de las muestras

```r
library(ggplot2)
library(reshape2)

expr_long <- melt(expr)
colnames(expr_long) <- c("Metabolito", "Muestra", "Valor")

expr_long$Fenotipo <- factor(
  colData(se)[expr_long$Muestra, "Phenotype"],
  levels = c("Control", "Lethal chlorpromazine poisoning", "Non-lethal chlorpromazine poisoning"),
  labels = c("Control", "Lethal", "Non-lethal")
)

# Boxplot por fenotipo con escala logarítmica
ggplot(expr_long, aes(x = Fenotipo, y = Valor)) +
  geom_boxplot() +
  scale_y_log10() +
  labs(title = "Distribución de intensidades por fenotipo",
       x = "Fenotipo", y = "log10(Intensidad)") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 30, hjust = 1, size = 8),
    plot.title = element_text(size = 10)
  )
```

Análisis del número de valores faltantes por muestra

```r
na_counts <- apply(expr, 2, function(x) sum(is.na(x)))

barplot(na_counts,
        las = 2,
        main = "Número de valores NA por muestra",
        cex.names = 0.6,    # Reduce tamaño del texto en eje X
        cex.axis = 0.8,     # Reduce tamaño del texto en eje Y
        col = "lightblue",
        border = "white")
```

Análisis de Componentes Principales (PCA)

```r
# Transformación logarítmica de la matriz de expresión
expr_log <- log10(assay(se) + 1)

# Quitar filas (metabolitos) con valores NA para que PCA funcione
expr_log <- expr_log[complete.cases(expr_log), ]

# Calcular PCA (muestras en columnas - por eso hacemos la transpuesta)
pca <- prcomp(t(expr_log), scale. = TRUE)

# Crear data frame con las 2 primeras componentes y añadir metadatos
pca_df <- as.data.frame(pca$x[, 1:2])
pca_df$Fenotipo <- colData(se)[rownames(pca_df), "Phenotype"]
pca_df$Género <- colData(se)[rownames(pca_df), "Gender"]

# Graficar
library(ggplot2)
ggplot(pca_df, aes(x = PC1, y = PC2, color = Fenotipo, shape = Género)) +
```

```
  geom_point(size = 3) +
  labs(title = "PCA de las muestras", x = "PC1", y = "PC2") +
  theme_minimal() +
  theme(
    legend.title = element_text(size = 8),
    legend.text = element_text(size = 7),
    plot.title = element_text(size = 10)
  )
```

Gráfico de barras que muestra los 10 metabolitos con una intensidad más variable

```
variabilidad <- apply(expr, 1, sd, na.rm = TRUE)
top_metabolitos <- head(sort(variabilidad, decreasing = TRUE), 10)

barplot(top_metabolitos,
        las = 2,
        main = "Top 10 metabolitos más variables",
        cex.names = 0.7,    # Reduce el tamaño de los nombres (eje X)
        cex.axis = 0.8,     # Reduce el tamaño del eje Y
        col = "lightblue",
        border = "white")
```

Mapa de calor basado en los metabolitos más variables

```
library(pheatmap)

# Seleccionar los metabolitos con mayor varianza
variabilidad <- apply(assay(se), 1, var)
top_metabs <- names(sort(variabilidad, decreasing = TRUE))[1:25]
top_data <- assay(se)[top_metabs, ]
top_data <- log10(top_data + 1)  # evitar log(0)

# Heatmap reducido
pheatmap(top_data,
         scale = "row",
         main = "Heatmap de los 25 metabolitos más variables",
         fontsize = 8,           # Tamaño general del texto
         fontsize_row = 6,       # Tamaño de nombres de metabolitos
         fontsize_col = 6,       # Tamaño de nombres de muestras
         border_color = NA)
```