PEC 1. Asignatura: Análisis de Datos Ómicos

José Alberto Camacho López

2025-04-02

1. Selección de dataset

En este primer apartado visitamos la página de Github del Borenstein Lab y encontramos datos curados de metabolómica (https://github.com/borenstein-lab/microbiome-metabolome-curated-data) y elegimos el dataset de FRANZOSA_IBD_2019 (https://github.com/borenstein-lab/microbiome-metabolome-curated-data/tree/main/data/processed_data/FRANZOSA_IBD_2019)

2. Carga de archivos y creación de un objeto de clase SummarizedExperiment

Cargamos las librerías necesarias

```
# Cargamos la librería readr para leeer archivos .tsv o .csv
library(readr)

# Cargamos la librería SummarizedExperiment
library(SummarizedExperiment)
```

```
## Cargando paquete requerido: MatrixGenerics
```

```
## Cargando paquete requerido: matrixStats
```

```
##
## Adjuntando el paquete: 'MatrixGenerics'
```

```
## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
```

```
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Cargando paquete requerido: GenomicRanges

## Cargando paquete requerido: stats4

## Cargando paquete requerido: BiocGenerics

##
## Adjuntando el paquete: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, saveRDS, setdiff,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Cargando paquete requerido: S4Vectors

##
## Adjuntando el paquete: 'S4Vectors'

## The following object is masked from 'package:utils':
##
##     findMatches

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Cargando paquete requerido: IRanges

##
## Adjuntando el paquete: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## Cargando paquete requerido: GenomeInfoDb

## Cargando paquete requerido: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
```

```
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Adjuntando el paquete: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
```

Cargamos la matriz de expresión génica (genera.counts.csv), con las muestras en las filas y las bacterias del microbioma en las columnas; el archivo de metadatos de las muestras (metadata.csv), que será utilizados como metadata de las filas (samples); y el archivo que contiene información adicional sobre los metabolitos (mtb.csv)

```
counts <- read.csv("genera.counts.tsv", sep = "\t", row.names = 1)
metadata <- read.csv("metadata.tsv" , sep = "\t")
metabolites <- read.csv("mtb.tsv", sep = "\t", row.names = 1)

# En metadata, quizás no son útiles algunas columnas. Nos quedamos con
las relevantes

metadata_clean <- metadata[, c("Sample", "Study.Group", "Age",
"Age.Units",
                               "Fecal.Calprotectin", "antibiotic",
"immunosuppressant",
                               "mesalamine", "steroids")]


# Verificar la estructura de la matriz
dim(counts)

## [1]   220 11720
```

Creamos el objeto Summarized Experiment.Aunque previamente fusionamos los datos de counts y los de los metabolitos, eliminamos la primera columna Sample y verificamos que metadata_clean$Sample coincide con rownames(data_matrix):

```
data_matrix <- cbind(counts[,-1], metabolites[,-1])
```

Verificamos que metadata_clean$Sample coincide con rownames(data_matrix):

```
all(metadata_clean$Sample %in% rownames(data_matrix))

## [1] TRUE
```

Ahora sí, creamos el objeto SummarizedExperiment:

```r
# Crea el SummarizedExperiment
se <- SummarizedExperiment(
  assays = list(counts = as.matrix(data_matrix)),
  rowData = metadata_clean
)

# Mostrar se
se
```

```
## class: SummarizedExperiment
## dim: 220 20566
## metadata(0):
## assays(1): counts
## rownames(220): PRISM.7122 PRISM.7147 ... Validation.UMCG9189151
##   Validation.UMCG9830707
## rowData names(9): Sample Study.Group ... mesalamine steroids
## colnames(20566):
##
d__Bacteria.p__Firmicutes_A.c__Clostridia.o__Oscillospirales.f__CAG.272.g
__Flemingibacterium
##
d__Bacteria.p__Firmicutes_A.c__Clostridia.o__Lachnospirales.f__Lachnospir
aceae.g__F0428
##   ... HILIC.pos_Cluster_2375..NA HILIC.pos_Cluster_2376..NA
## colData names(0):
```

Extraemos los metadatos de las muestras:

```r
metadata_se <- rowData(se)
metadata_se
```

```
## DataFrame with 220 rows and 9 columns
##                                         Sample Study.Group       Age
Age.Units
##                                    <character> <character> <integer>
<character>
## PRISM.7122                          PRISM.7122          CD        38
Years
## PRISM.7147                          PRISM.7147          CD        50
Years
## PRISM.7150                          PRISM.7150          CD        41
Years
## PRISM.7153                          PRISM.7153          CD        51
Years
## PRISM.7184                          PRISM.7184          CD        68
Years
## ...                                        ...         ...       ...
...
## Validation.UMCG8255148 Validation.UMCG8255148          UC        21
Years
## Validation.UMCG8438880 Validation.UMCG8438880          CD        32
```

```
Years
## Validation.UMCG9119812 Validation.UMCG9119812          CD        38
Years
## Validation.UMCG9189151 Validation.UMCG9189151          CD        51
Years
## Validation.UMCG9830707 Validation.UMCG9830707          CD        43
Years
##                        Fecal.Calprotectin  antibiotic
immunosuppressant
##                                 <integer> <character>
<character>
## PRISM.7122                            207          No
Yes
## PRISM.7147                             NA          No
No
## PRISM.7150                            218          No
Yes
## PRISM.7153                             NA          No
No
## PRISM.7184                             20          No
No
## ...                                   ...         ...
...
## Validation.UMCG8255148                 40          No
No
## Validation.UMCG8438880                 45          No
Yes
## Validation.UMCG9119812                305          No
Yes
## Validation.UMCG9189151                 44          No
Yes
## Validation.UMCG9830707                 NA          No
No
##                         mesalamine    steroids
##                        <character> <character>
## PRISM.7122                    No          No
## PRISM.7147                   Yes          No
## PRISM.7150                    No          No
## PRISM.7153                   Yes          No
## PRISM.7184                    No          No
## ...                          ...         ...
## Validation.UMCG8255148       Yes          No
## Validation.UMCG8438880        No          No
## Validation.UMCG9119812        No          No
## Validation.UMCG9189151        No          No
## Validation.UMCG9830707        No         Yes
```

Las principales diferencias entre SummarizedExperiment, perteneciente a la librería de su mismo nombre,y ExpressionSet, que pertenece a la librería Biobase (ambos de Bioconductor), son

- El primero usa assays, esto es, múltiples matrices de expresión como counts, FPKM, TPM, etc., mientras ExpressionSet sólo permite una matriz de expresión (assayData), con las genes en las filas y las muestras en columnas.

- Para los datos fenotípicos es más flexible SummarizedExperiment pues guarda los datos en colData, un dataframe, al contrario que ExpressionSet, que lo hace en phenoData (objeto de tipo AnnotatedDataFrame).

- Para anotación de genes pasa igual, SummarizedExperiment usa rowData, de nuevo un dataframe, más sencillo de entender que el formato AnnotatedDataFrame de ExpressionSet.

- Los metadatos son guardados en ExperimentData, de formato MIAME (Minimum Information About a Microarray Experiment), mientras el formato metadata de SE es más flexible.

- Dado que ExpressionSet está orientado a análisis de microarrays es más antiguo y es compatible con algunos paquetes antiguos de Bioconductor; mientras SummarizedExperiment, más moderno, fue concebido para trabajar con RNA-Seq y, por lo tanto, se vincula con paquetes como DESeq2, edgeR o limma.

3. Análisis Exploratorio

A continuación, procedemos a realizar un análisis exploratorio de nuestros datos:

```
# Vemos primeras filas de la matriz de counts y sus dimensiones
head(assay(se), 5)[, 1:5] # solo las primeras 5 filas y primeras 5
columnas
```

```
##
d__Bacteria.p__Firmicutes_A.c__Clostridia.o__Oscillospirales.f__CAG.272.g
__Flemingibacterium
## PRISM.7122
53
## PRISM.7147
785
## PRISM.7150
21881
## PRISM.7153
30
## PRISM.7184
139
##
d__Bacteria.p__Firmicutes_A.c__Clostridia.o__Lachnospirales.f__Lachnospir
aceae.g__F0428
## PRISM.7122
37
## PRISM.7147
491
```

```
## PRISM.7150
359
## PRISM.7153
110
## PRISM.7184
462
##
d__Bacteria.p__Firmicutes_A.c__Clostridia.o__Lachnospirales.f__Lachnospir
aceae.g__Lachnoanaerobaculum
## PRISM.7122
472
## PRISM.7147
2548
## PRISM.7150
790
## PRISM.7153
401
## PRISM.7184
1167
##
d__Bacteria.p__Firmicutes.c__Bacilli.o__Lactobacillales.f__Carnobacteriac
eae.g__Carnobacterium_A
## PRISM.7122
0
## PRISM.7147
103
## PRISM.7150
64
## PRISM.7153
30
## PRISM.7184
16
##
d__Bacteria.p__Firmicutes_A.c__Clostridia.o__Lachnospirales.f__Lachnospir
aceae.g__GCA.900066755
## PRISM.7122
1268
## PRISM.7147
8012
## PRISM.7150
63641
## PRISM.7153
1222
## PRISM.7184
1308
```

```
dim(assay(se))
```

```
## [1]   220 20566
```

```r
# Ver dimensiones de los metadatos
dim(rowData(se))
```

```
## [1] 220   9
```

Realizamos un resumen inicial de los datos de conteo:

```r
# Convertimos los valores de conteo a un vector
counts_vector <- as.vector(t(assay(se)))
summary(counts_vector)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##        0        0        0     1752       46 59525299
```

Esto muestra que los datos de conteo tienen una gran dispersión, pues tanto el mínimo como la mediana y primer cuartil es cero, además de un tercer cuartil realmente bajo. Sabemos entonces que más del 50% de los datos son cero. Es un distribución muy sesgada. Comprobamos la proporción de ceros:

```r
mean(as.vector(t(assay(se)) == 0))
```

```
## [1] 0.552032
```

En este caso, deberemos trabajar con una transformación logarítmica de la data, antes de realizar los boxplots de distribución. Lo hacemos en tres slicing de nuestras miles de muestras:

```r
# Aplicar la transformación logarítmica, log1p (log(1 + x)) a los datos
de conteo
data_log <- log1p(assay(se))

# Luego, aplicar estandarización (z-score)
data_std <- scale(data_log)

# Crear el objeto SummarizedExperiment con los datos estandarizados
se_std <- SummarizedExperiment(
  assays = list(counts = data_std),
  rowData = rowData(se),
  colData = colData(se)
)

# Ver la estructura del objeto resultante
se_std
```

```
## class: SummarizedExperiment
## dim: 220 20566
## metadata(0):
## assays(1): counts
## rownames(220): PRISM.7122 PRISM.7147 ... Validation.UMCG9189151
##   Validation.UMCG9830707
## rowData names(9): Sample Study.Group ... mesalamine steroids
```

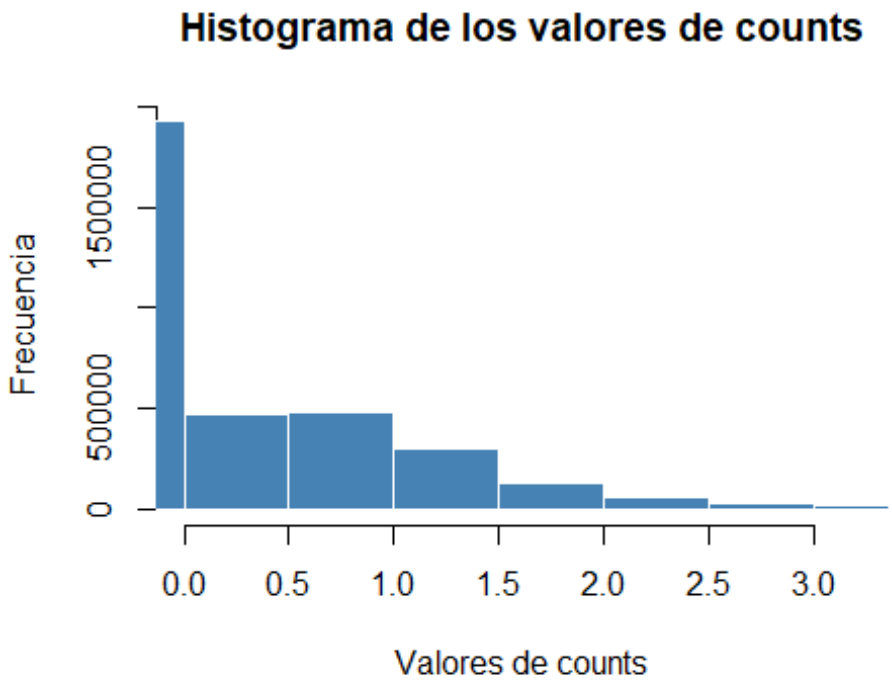
```
## colnames(20566):
##
d__Bacteria.p__Firmicutes_A.c__Clostridia.o__Oscillospirales.f__CAG.272.g
__Flemingibacterium
##
d__Bacteria.p__Firmicutes_A.c__Clostridia.o__Lachnospirales.f__Lachnospir
aceae.g__F0428
##   ... HILIC.pos_Cluster_2375..NA HILIC.pos_Cluster_2376..NA
## colData names(0):
```

```r
counts_vector <- as.vector(assay(se_std))
summary(counts_vector)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -8.8775 -0.4671 -0.1663  0.0000  0.4174 14.7650
```

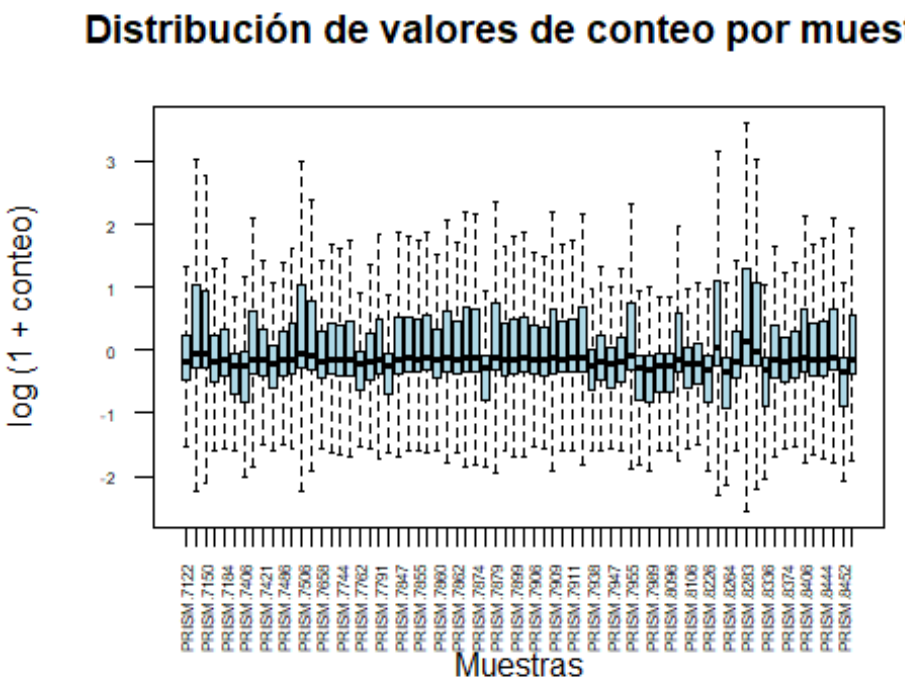
La distribución se distribuye más homogéneamente. Ahora, la miramos en un histograma:

```r
# Crear histograma
hist(
  counts_vector,
  breaks = 50,
  col = "steelblue",
  border = "white",
  main = "Histograma de los valores de counts",
  xlab = "Valores de counts",
  ylab = "Frecuencia",
  xlim = c(0, quantile(counts_vector, 0.99))
)
```



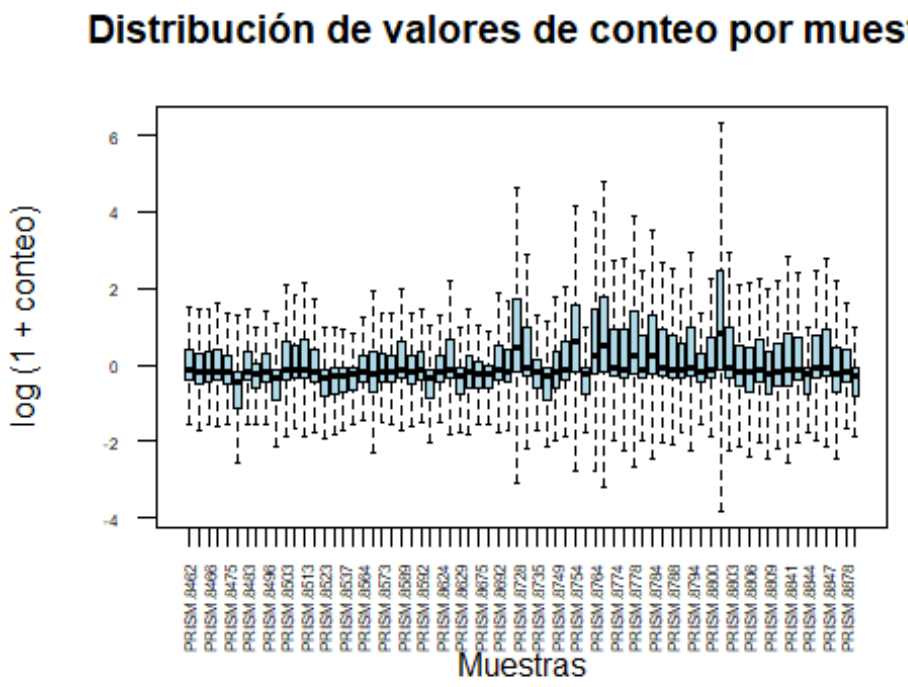
Y realizamos boxplots por conjunto de muestras, ya que son demasiadas para verlas en un único gráfico:

```
# Seleccionamos un subconjunto de muestras

subset_samples1 <- 0:70

# Creamos el boxplot
boxplot(t(assay(se_std))[, subset_samples1],
        outline = FALSE,  # Oculta outliers para mejor visualización
        col = "lightblue",
        main = "Distribución de valores de conteo por muestra",
        xlab = "Muestras",
        ylab = "log (1 + conteo)",
        cex.axis = 0.5,
        las = 2)
```



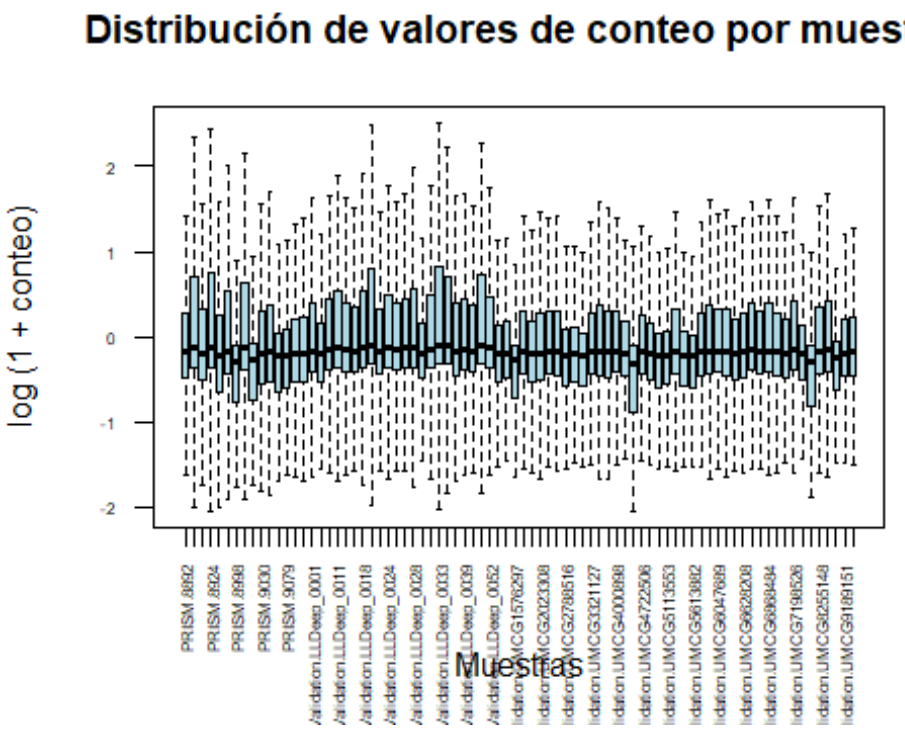
Dispersión bastante alta.

```
# Seleccionamos un subconjunto de muestras

subset_samples2 <- 71:140

# Creamos el boxplot
boxplot(t(assay(se_std))[, subset_samples2],
        outline = FALSE,  # Oculta outliers para mejor visualización
        col = "lightblue",
        main = "Distribución de valores de conteo por muestra",
        xlab = "Muestras",
        ylab = "log (1 + conteo)",
        cex.axis = 0.5,
        las = 2)
```



En este caso la dispersión es muy alta.

```
# Seleccionamos un subconjunto de muestras

subset_samples3 <- 141:220

# Creamos el boxplot
boxplot(t(assay(se_std))[, subset_samples3],
        outline = FALSE,  # Oculta outliers para mejor visualización
        col = "lightblue",
        main = "Distribución de valores de conteo por muestra",
        xlab = "Muestras",
        ylab = "log (1 + conteo)",
        cex.axis = 0.5,
        las = 2)
```



Cierta dispersión aquí.

Realizamos asimismo un cálculo de las componentes principales y la representación gráfica de la PC1 VS. PC2

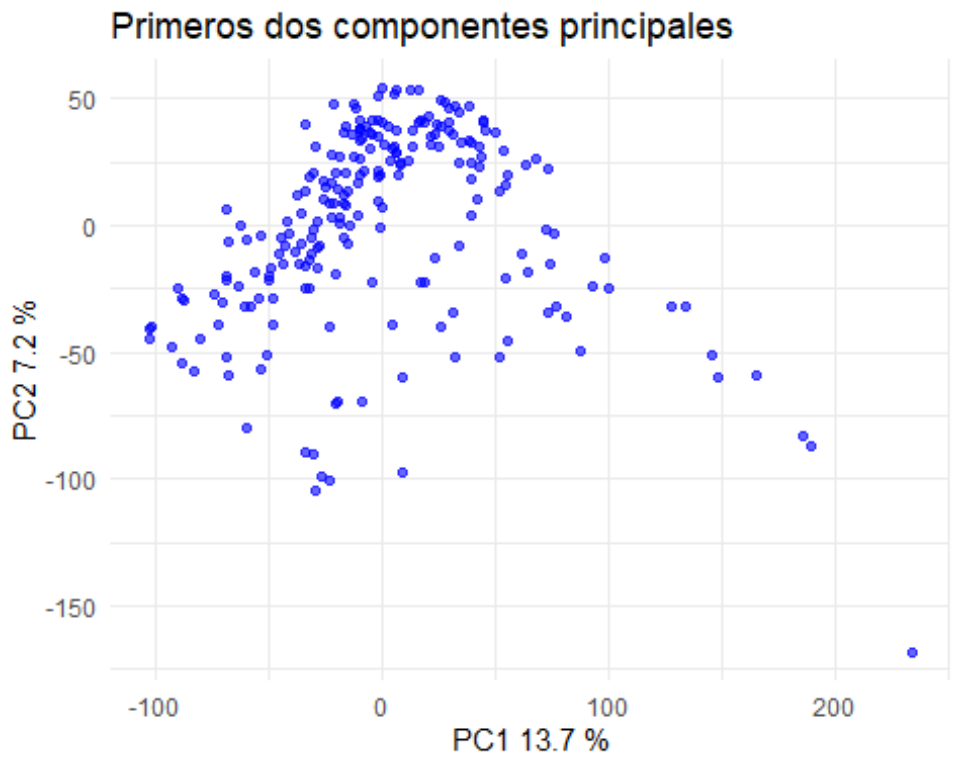
```
# Realizar PCA usando prcomp
PCAS <- prcomp(assay(se_std))

# Mostramos cúales son las PCA
names(PCAS)

## [1] "sdev"     "rotation" "center"   "scale"    "x"

# Mostramos las primeras filas de los primeros dos componentes
principales
head(PCAS$x[, 1:2])

##                    PC1        PC2
## PRISM.7122  -8.8543192  34.061730
## PRISM.7147  92.8440687 -24.149805
## PRISM.7150  71.8482616  -1.705047
## PRISM.7153 -16.6856543   7.747795
## PRISM.7184  -0.7243619  20.240239
## PRISM.7238 -50.0862131 -21.811204
```

Debemos calcular cuánto se dispersan estas dos componentes principales con la varianza explicada por cada una de estas componentes:

```r
# Calculamos la varianza explicada por cada componente
explained_variance <- PCAS$sdev^2 / sum(PCAS$sdev^2)

# Hacemos el porcentaje de varianza explicada
PC1_perc <- explained_variance[1] * 100
PC2_perc <- explained_variance[2] * 100
```

Mostramos las dos primeras componentes principales en un gráfico de dispersión:

```r
library(ggplot2)

# Crear un data.frame con las primeras dos componentes principales
df_PCA <- data.frame(PC1 = PCAS$x[, 1],
                     PC2 = PCAS$x[, 2],
                     Sample = rownames(PCAS$x))

# Crear el gráfico de dispersión con los porcentajes de varianza
explicada
ggplot(df_PCA, aes(x = PC1, y = PC2)) +
  geom_point(color = "blue", alpha = 0.6) +
  labs(
    title = "Primeros dos componentes principales",
    x = paste("PC1", round(PC1_perc, 1), "%"),
    y = paste("PC2", round(PC2_perc, 1), "%")
  ) +
  theme_minimal()
```



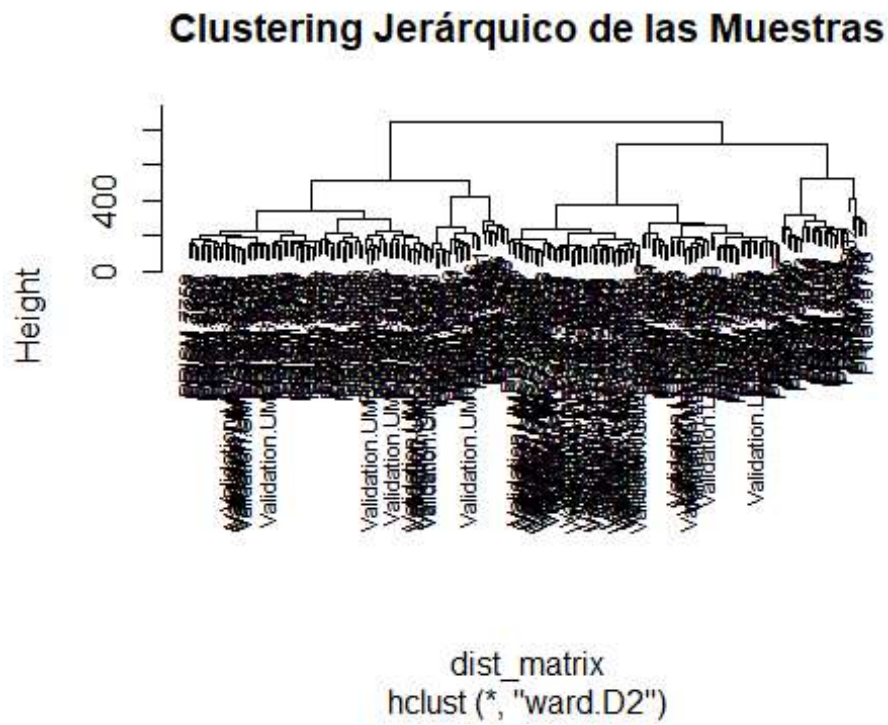
Por último, caracterizamos las muestras en un clustering jerárquico

Para ello y en primer lugar, calculamos la distancia euclídea entre muestras:

```
# Cálculo de la distancia euclidia entre muestras
dist_matrix <- dist(assay(se_std), method = "euclidean")

# Realiza clustering jerárquico
hc <- hclust(dist_matrix, method = "ward.D2")

hc

##
## Call:
## hclust(d = dist_matrix, method = "ward.D2")
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 220
```

Ploteamos el dendograma:

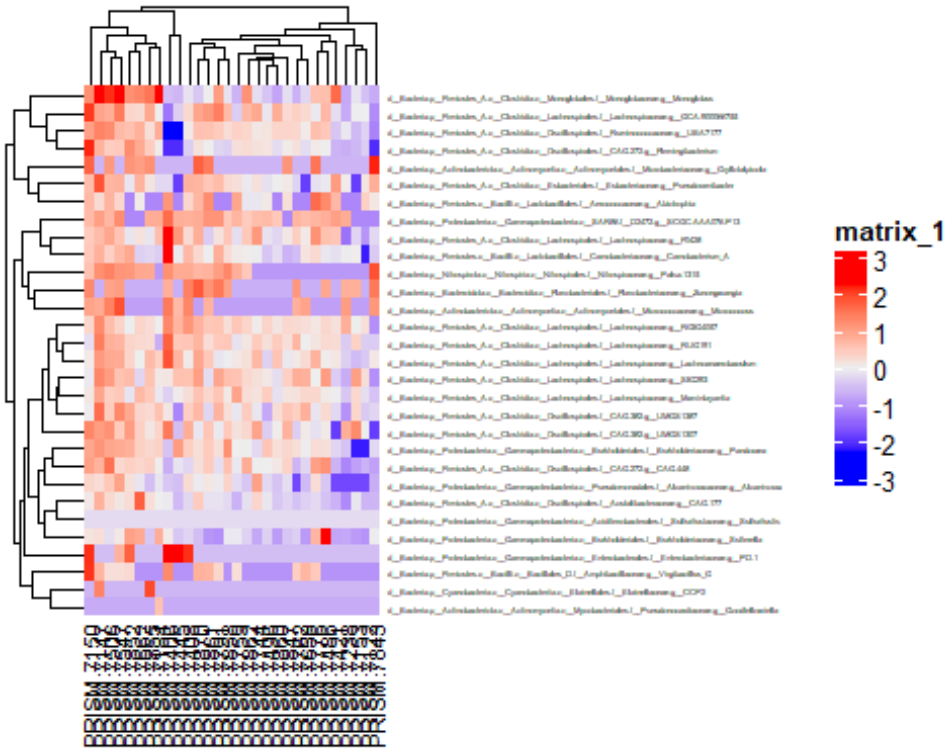
```
# Dendrograma de clustering jerárquico
plot(hc, labels = colnames(t(assay(se_std))), main = "Clustering
Jerárquico de las Muestras", cex = 0.7)
```



Parece haber tres grupos marcados y por eso tal vez veíamos tres grupos de distribuciones diversas.

Realizamos un heatmap de expresión:

```
library(ComplexHeatmap)
Heatmap(
  t(assay(se_std))[1:30, 1:30],
  column_names_gp = gpar(fontsize = 8),
  row_names_gp = gpar(fontsize = 3)
) # Recortamos, para que sea legible
```



Vemos que los metabolitos primero y 27º son los más sobreexpresados de la lista.

4. Creación de los archivos

Creamos los archivos pedidos en el enunciado:

```
# Guardamos el objeto SummarizedExperiment
#save(se, file = "data/se_object.Rda") # Para los datos originales
#save(se_std, file = "data/se_std_object.Rda") # Para los datos
estandarizados

# Guardamos la matriz y los metadatos en txt
#write.table(assay(se), file = "data/counts_matrix.txt", sep = "\t",
quote = FALSE)
#write.table(rowData(se), file = "data/metadata.txt", sep = "\t", quote =
FALSE)

#write.table(assay(se_std), file = "data/counts_matrix_std.txt", sep =
```

```
"\t", quote = FALSE)
#write.table(rowData(se_std), file = "data/metadata_std.txt", sep = "\t",
quote = FALSE)
```