Análisis de datos ómicos

Primera prueba de evaluación continua

Vera Garmendia

2025-04-02

URL repositorio GitHub: https://github.com/VeraGarmendia/Garmendia-Dulfova-Vera-PEC1

1. SummarizedExperiment

SummarizedExperiment es una estructura similar a una matriz en la que las filas representan características de interés, como genes, transcritos, exones o metabolitos, y las columnas corresponden a las muestras experimentales. Este objeto puede contener uno o más ensayos, cada uno almacenado en una matriz u otra estructura numérica.

En muchos aspectos, **SummarizedExperiment** es similar a la estructura histórica **ExpressionSet**, pero con mayor flexibilidad en la representación de las filas. Mientras que **ExpressionSet** se diseñó principalmente para experimentos basados en microarrays, donde las filas representan características predefinidasEn particular, **SummarizedExperiment** permite utilizar tanto coordenadas genómicas basadas en **GRanges** como descripciones personalizadas mediante **DataFrames**, lo que lo hace especialmente útil para experimentos de secuenciación como RNA-Seq y ChIP-Seq.

1.1 Componentes clave

- `assays()`: conjunto de matrices (o estructuras similares a matrices) con dimensiones idénticas. Las filas representan **características biológicas** (genes, transcritos, proteínas, metabolitos, etc.), mientras que las columnas corresponden a **muestras**.
- `colData()`: contiene las anotaciones de las muestras en formato **DataFrame**. Cada fila representa una muestra y coincide exactamente con las columnas de los datos de expresión. Este slot almacena **covariables de las muestras**, como el grupo experimental o condiciones de tratamiento.
- `rowData()` / `rowRanges()`: almacena las anotaciones de las características biológicas (filas), como coordenadas genómicas de genes, exones u identificadores de diferentes bases de datos.
- `metadata()`: lista de metadatos no estructurados que describen el contenido general del objeto.

Este diseño modular permite **gestionar y analizar datos ómicos de manera eficiente**, garantizando la coherencia entre sus componentes. La estructura coordinada de **SummarizedExperiment** asegura que cualquier modificación en los datos, por ejemplo, la eliminación de una muestra, se refleje automáticamente en todas las partes del objeto. Asimismo, los slots de metadatos pueden expandirse con nuevas covariables sin afectar la integridad de los datos.

2. Selección del dataset

2.1. Materiales y métodos

Esta PEC tiene como objetivo planificar y ejecutar una versión simplificada del proceso de análisis de datos ómicos en metabolómica, utilizando R y sus librerias, como **Bioconductor**. Para ello, se creará un objeto **SummarizedExperiment**, que luego será empleado en un análisis exploratorio.

En primer lugar, seleccionaremos un conjunto de datos de metabolómica. En este caso, se ha seleccionado el dataset con ID **ST000291** (*"LC-MS Based Approaches to Investigate Metabolomic Differences in the Urine of Young Women after Drinking Cranberry Juice or Apple Juice"*) del repositorio **Metabolomics Workbench**, obtenido a través de GitHub.

He elegido este dataset porque está dividido en tres archivos, lo que me permite crear fácilmente un objeto **SummarizedExperiment**, y es un dataset bastante completo que facilitará la realización de un análisis exploratorio posterior.

El análisis exploratorio incluirá:

- **Análisis univariante**: mediante estadísticas básicas y generación de boxplots.
- **Análisis multivariante**: utilizando PCA (Análisis de Componentes Principales) y métodos de agrupación como Hierarchichal Clustering.

2.2. Resumen del estudio

En el estudio participaron 18 mujeres saludables (21-29 años) con un IMC normal. Se les pidió evitar alimentos ricos en procianidinas (arándanos, manzanas, uvas, chocolate, ciruelas) entre los días 1 y 6. En el día 7, se recogieron muestras de orina y sangre en ayunas. Las participantes se dividieron en dos grupos (n=9) para consumir jugo de arándano o de manzana durante tres días. El día 10, se repitieron las muestras. Después de un período de lavado de dos semanas, se cambió la dieta entre los grupos y se repitió el protocolo.

El objetivo del estudio fue investigar los cambios metabólicos inducidos por las procianidinas de arándanos y manzanas mediante metabolómica LCMS global.

2.3. Descarga de los datos

Una vez descargados, los datos consisten en tres archivos separados:

1. **Feature data**: 1541 variables o metabolitos, 45 muestras.
2. **Metadata**: 45 filas, dos columnas (nombre de muestra, nombre de tratamiento).
3. **Metabolites name**: 1541 filas, 3 columnas (nombre original del metabolito, ID de PubChem y ID de KEGG).

Así pues, importamos los datos con la función `read.csv()` y confirmamos que su estructura coindice con la mencionada.

3. Creación del objeto SummarizedExperiment

Para crear un objeto de clase SummarizedExperiment primero debemos instalar el paquete SummarizedExperiment de Bioconductor.

Un aspecto a tener en cuenta es que **SummarizedExperiment** trabaja con matrices, por lo que debemos transformar la tabla `features` en una matriz numérica con la función `as.matrix()`.

Transformamos `features` (data.frame) en una matriz numérica, donde:

- **Las filas** representan los 1541 metabolitos (nombrados con su identificador de PubChem)
- **Las columnas** corresponden a las 45 muestras (nombradas con el ID de las muestras).

Como mencionamos anteriormente, el número de filas de la matriz `features_matrix` debe coincidir con el número de filas de **feature metadata**, `metabolites`. Del mismo modo, el número de columnas de la matriz debe coincidir con el número de filas de **sample metadata**, `metadata`.

```
dim(features_matrix) # matriz
```

```
## [1] 1541   45
```

```
dim(metadata) # sample metadata
```

```
## [1] 45  2
```

```
dim(metabolites) # feature metadata
```

```
## [1] 1541    3
```

Debiado a que estas condiciones se cumplen, procederemos a crear el objeto **SummarizedExperiment** a partir de estas tablas:

- **Matriz de expresión** (`features_matrix`): se utilizará como el *assay*.
- **Metadatos de las muestras** (`metadata`): se utilizará como el *sample metadata*.
- **Metadatos de los metabolitos** (`metabolites`): se utilizará como el *feature metadata*.

3.1. Reorganización de los datos

Es fundamental garantizar que los nombres de las filas de `metadata`y `metabolites`coincidan correctamente, tanto en orden como en nombre, con la matriz de expresión (`features_matrix`). Para ello:

- **Alineación de nombres**:
  - **Muestras**: las columnas de `features_matrix` deben coincidir con las filas de `metadata`, por lo que asignamos los nombres de las filas en `metadata` usando `metadata$ID` y luego eliminamos esta columna para evitar redundancias.
  - **Metabolitos**: las filas de `features_matrix` deben coincidir con las filas de `metabolites`, usando los identificadores PubChem. Renombramos las filas de `metabolites` con `metabolites$PubChem` y luego eliminamos esta columna para evitar redundancias.
- **Alineación del orden**:
  - Para evitar errores en la comparación, debemos de ordenar ambas estructuras, `features_matrix` y `metabolites` según sus identificadores.

3.2. SummarizedExperiment()

Antes de crear el objeto SummarizedExperiment, con la función `SummarizedExperiment()`, siempre debemos realizar una comprobación con `stopifnot()`, que hará una doble verificación, que los nombres de las filas de `features_matrix` coincidan con los identificadores de PubChem en `metabolites` y que que los nombres de las columnas de `features_matrix` coincidan con los identificadores de muestra en `metadata`. Si hay alguna discrepancia, el código se detendrá, evitando la creación de nuestro objeto.

```
stopifnot(rownames(features_matrix_ordered) == metabolites_ordered$PubChem)
stopifnot(colnames(features_matrix_ordered) == metadata$ID)

# Creamos el objeto SummarizedExperiment
se <- SummarizedExperiment(assays = list(counts = features_matrix_ordered),
                           colData = metadata,
                           rowData = metabolites_ordered)
se
```

```
## class: SummarizedExperiment
## dim: 1541 45
## metadata(0):
## assays(1): counts
## rownames(1541): 10007 10023290 ... 998 UNKNOWN
## rowData names(2): names KEGG
## colnames(45): b1 b10 ... c8 c9
## colData names(1): Treatment
```

4. Análisis exploratorio e interpretación de los resultados desde el punto de vista biológico

Una vez creado nuestro objeto SummarizedExperiment `se`, procederemos a su análisis exploratorio.

4.1. Valores missing (NA)

Como en cualquier análisis exploratorio de datos, la presencia de valor NA puede afectar a la interpretación de los resultados. En caso de que haya, las imputaremos por la media de cada columna.

```
# Accedemos a la matriz del objseto se y miramos si hay valores NA
sum(is.na(assay(se)))
```

```
## [1] 8190
```

```
# Como hay muchos valores NA, los imputamos con la media de cada columna
assay(se) <- apply(assay(se), 2, function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x))
# Verificamos que no queden valores NA
sum(is.na(assay(se)))
```

```
## [1] 0
```

4.2. Análisis univariante

Como nuestras muestras están categorizadas según el tratamiento recibido, "baseline", "apple" y "cranberry", podemos asignicar colores a cada uno de esos tratamientos para facilitar la visualización.

```
# Asignamos colores a cada tratamiento
groupColors <- c("Baseline" = "blue", "Apple" = "green", "Cranberry" = "red")

# Creamos un vector de colores para cada muestra según su tratamiento
col <- groupColors[colData(se)$Treatment]
```

A continuación, se muestran las estadísticas básicas de los primeros cinco metabolitos en las muestras, desglosadas según el tratamiento recibido.

```
# Filtramos las muestras por tratamiento
baseline <- se[, se$Treatment == "Baseline"]
apple <- se[, se$Treatment == "Apple"]
cranberry <- se[, se$Treatment == "Cranberry"]

cat("Baseline\n")
```

```
## Baseline
```

```
round(apply(assay(baseline[1:5,]),1, summary))
```

```
##           10007 10023290 100275 10037499   1005
## Min.      28600    37700   2800     2330  35400
## 1st Qu.   69500    98150  11715    81100 212000
## Median    88700   287000 110000   134000 392000
## Mean     391320   517880 159868   308015 425060
## 3rd Qu.  188000   729000 194000   281000 604500
## Max.    3820000  1760000 854000  1530000 995000
```

```
cat("\nApple\n")
```

```
##
## Apple
```

```
round(apply(assay(apple[1:5,]),1, summary))
```

```
##            10007 10023290 100275 10037499   1005
## Min.       10500     6660   1780        0  23000
## 1st Qu.    35700    85750  13800    12200  96150
## Median     88800   204000  31600   101000 445000
## Mean     1198853   320764  76145   175404 382620
## 3rd Qu.   733000   381000  70750   167000 604500
## Max.    11000000  1560000 432000  1050000 670000
```

```
cat("\nCranberry\n")
```

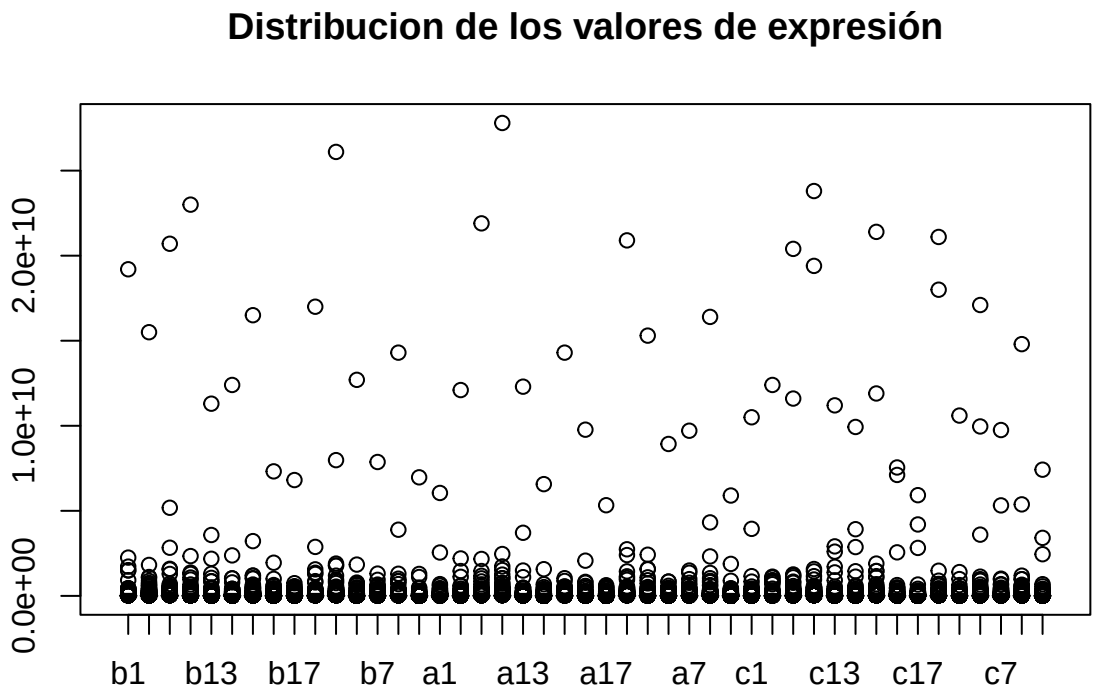
```
##
## Cranberry
```

```
round(apply(assay(cranberry[1:5,]),1, summary))
```

```
##            10007 10023290 100275 10037499    1005
## Min.        6760     7540   4540     2280   66500
## 1st Qu.    41400    96650  12350    58100  228500
## Median     71700   155000  82100    95600  698000
## Mean      992257  2239463 152697   270482  551500
## 3rd Qu.   275500   833500 226000   293500  807000
## Max.    11800000 27400000 785000  1300000 1060000
```

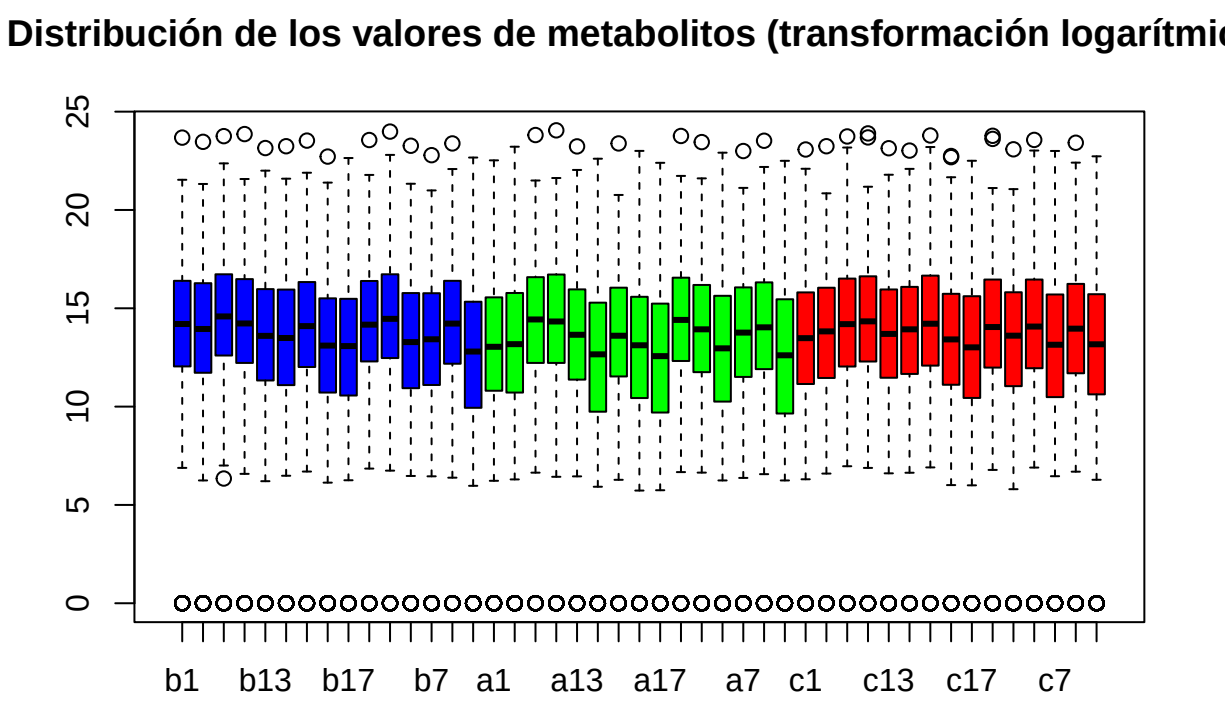
Ahora analizamos la distribución de los valores de expresión de los metabolitos en las diferentes muestras.

```
boxplot(assay(se), col = col, main="Distribucion de los valores de expresión")
```



Parece que la distribución de valores de expresión es bastante grande, dificultando la interpretación. Así pues, podemos hacer una transformación logarítmica para reducir la dispersión y facilitar su visualización.

```
boxplot(log(assay(se)+1), col = col, main="Distribución de los valores de metabolitos (transformación l
```



Mucho mejor ahora, por lo que es mejor trabajar con los datos transformados logarítmicamente.

4.2. Análisis multivariante

El análisis en componentes principales (PCA) es una técnica estadística que facilita la visualización de datos en dimensiones reducidas y, sobre todo, ayuda a detectar patrones o agrupaciones que podrían no ser evidentes a simple vista. Esta técnica transforma las variables originales, en nuestro caso los metabolitos, en nuevas variables, llamadas componentes principales.

Para el análisis de PCA, utilizamos la función `prcomp()`, que necesita como entrada una matriz donde las muestras son filas y las variables, es decir, los metabolitos son las columnas. Por esta razón, debemos transponer nuestra matriz `assay(se)` con la función `t()`.

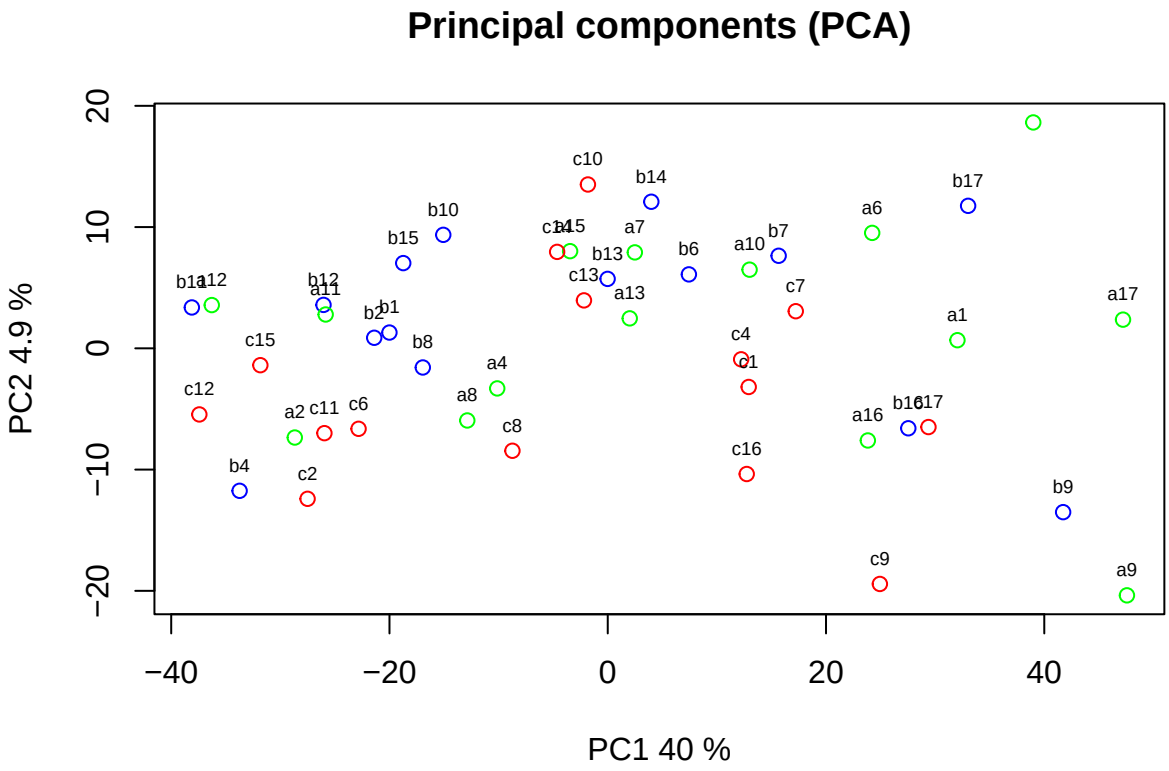
```
# Realizamos el Análisis de Componentes Principales
pca <- prcomp(t(log(assay(se)+1)), scale. = TRUE)
```

Una vez realizado el PCA, es importante calcular la cantidad de variabilidad que cada componente principal explica.

```
# Calculamos la varianza explicada por cada componente principal
loads <- round(pca$sdev^2 / sum(pca$sdev^2) * 100, 1)
```

El PCA se puede visualizar fácilmente en un gráfico de dispersión de los primeros dos componentes principales (PC1 y PC2), que generalmente capturan la mayor parte de la variabilidad.

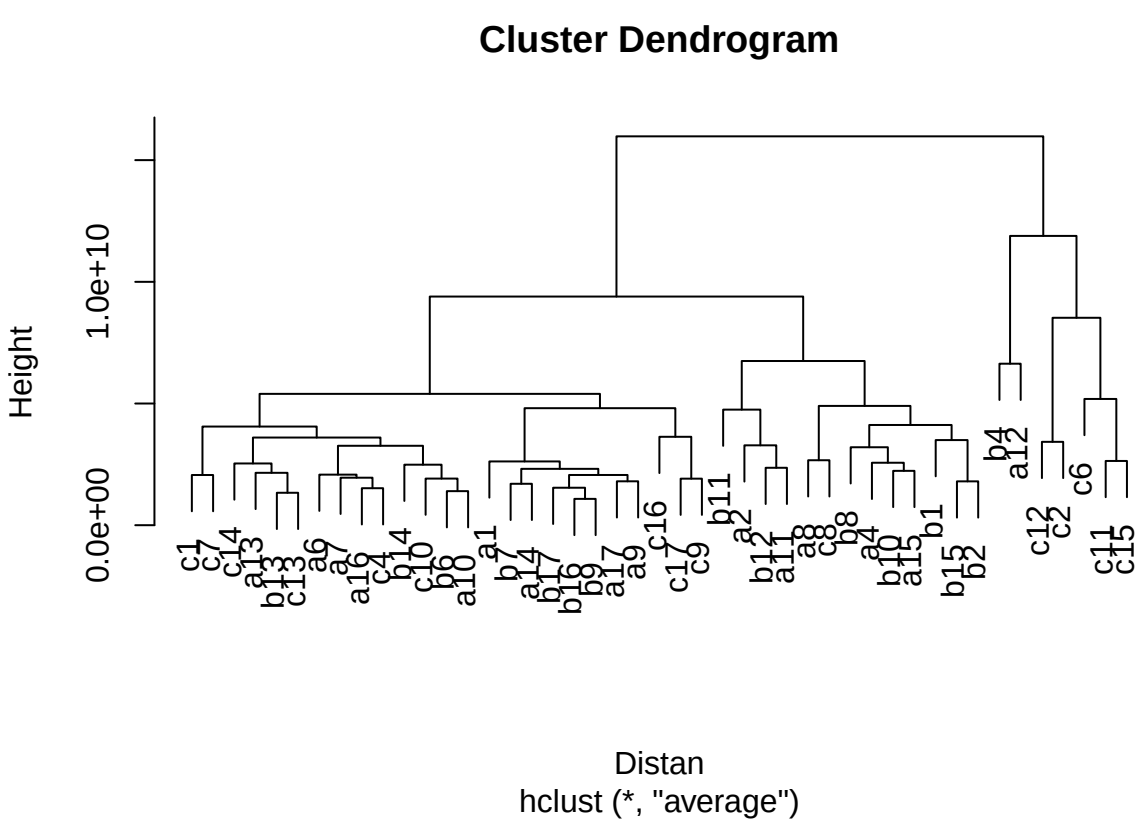
```r
# Creamos el gráfico de dispersión de los dos primeros componentes principales
xlab <- c(paste("PC1", loads[1], "%"))
ylab <- c(paste("PC2", loads[2], "%"))
plot(pca$x[, 1:2], xlab = xlab, ylab = ylab, col = col,
     main = "Principal components (PCA)")

names2plot <- rownames(colData(se))

text(pca$x[,1],pca$x[,2], names2plot, pos=3, cex=.6)
```



A partir del gráfico podemos concluir que PC1 explica el 40% de la variabilidad y PC2 el 4.9%, lo que indica que una gran parte de la variabilidad se captura en el primer componente, PC2 contribuyendo menos. Sin embargo, no observamos un patrón de agrupación entre grupos de tratamiento, lo que nos podría hacer pensar que los tratamientos no tienen efecto suficientes en los metabolitos para poder diferenciarlo entre las muestras.

También podemos hacer agrupaciones en los datos con un análisis de clustering jerárquico, que puede ayudar tanto a visualizar agrupaciones esperadas como a descubrir agrupaciones inesperadas en las muestras.

```r
# Definimos el método de cálculo de distancia
distmeth <- c("euclidean")
Distan <- dist(t(assay(se)), method=distmeth) # Matriz de distancia entre muestras
# Definimos el método de agrupamiento
treemeth <- c("average")
hc <- hclust(Distan, method=treemeth)
plot(hc)
```



Al igual que en el PCA, la agrupación de muestras de diferentes tratamientos en el análisis de clustering podría sugerir que, en términos de las características medidas, los niveles de expresión de los metabolitos, estas muestras son similares entre sí, a pesar de pertenecer a tratamientos distintos. Pero a pesar de esto parace que algunas muestras de un mismo tratamiento se agrupan.

Como conclusión podriamos indicar que los tratamientos no están produciendo diferencias claras en el perfil de las muestras, o que las variaciones biológicas dentro de cada grupo de tratamiento son mayores que las diferencias entre tratamientos. En este caso, es posible que los efectos de los tratamientos sean sutiles.

También destacar la gran presencia de valores NA, las cuales las hemos imputado por media de columnas, que puede generar una gran variabilidad de los datos y, por ende, la capacidad de los análisis para detectar diferencias claras entre tratamientos. Una alternativa sería buscar otros métodos de imputación, como kNN.

5. Bibliografía

ASP Teaching. Análisis de conglomerados (Cluster Analysis). https://aspteaching.github.io/AMVCasos/#an%C3%A1lisis-de-conglomerados-cluster-analysis

ASP Teaching. Análisis de datos ómicos - Microarrays. https://aspteaching.github.io/Analisis_de_datos_omicos-Ejemplo_0-Microarrays/ExploreArrays.html

Bioconductor. fobitools: MW_ST000291 enrichment analysis. https://www.bioconductor.org/packages/devel/bioc//vignettes/fobitools/inst/doc/MW_ST000291_enrichment.html

Bioconductor. SummarizedExperiment package. https://www.bioconductor.org/packages/release/bioc/html/SummarizedExperiment.html

Bioconductor. SummarizedExperiment vignette. https://bioconductor.org/packages/release/bioc/vignettes/SummarizedExperiment/inst/doc/SummarizedExperiment.html

Ciencia de Datos. Análisis de Componentes Principales (PCA). https://cienciadedatos.net/documentos/35_principal_component_analysis#Ejemplo_PCA_aplicado_a_gen%C3%B3mica

CompGenomR Genomic Intervals with SummarizedExperiment Class. https://compgenomr.github.io/book/genomic-intervals-with-more-information-summarizedexperiment-class.html

Mano, L. PCA and Heatmap: Workshop on Data Visualization in R. https://nbisweden.github.io/workshop-data-visualization-r/2204/lab_pca_hmap.html

Metabolomics Workbench. Study ST000291 Metadata. https://www.metabolomicsworkbench.org/data/DRCCMetadata.php?Mode=Study&StudyID=ST000291

PubMed.(2020). Artículo en PubMed. https://pubmed.ncbi.nlm.nih.gov/32133462/

STHDA. ExpressionSet and SummarizedExperiment. https://www.sthda.com/english/wiki/expressionset-and-summarizedexperiment#:~:text=The%20ExpressionSet%20is%20generally%20used,is%20in%20the%20Biobase%20libra

UCLouvain-CBIO. RNA-seq Analysis. https://uclouvain-cbio.github.io/bioinfo-training-02-rnaseq/sec-se.html