PEC 1 ADO

Daniele Sebastiani

2024-11-01

Contents

Resumen ejecutivo

En esta primera PEC se nos pide realizar una exploración de los datos de un dataset de metabólomica seleccionado en un repositorio GitHub. Por eso, he seleccionado uno entre los disponibles y he descargado e comprobado los archivos.

El estudio del cual se ha realizado el dataset tiene como objetivo lo de medir la concentración de varios metabolitos en muestras de orina de 15 partecipantes después de consumir jugos de Arandano o de Manzana.

Una vez preparados los datos, he realizado un contenedor de clase `SummarizedExperiment` que almacena datos y metadatos asociados de forma adecuada.

Posteriormente, he llevado a cabo un análisis exploratorio de los datos, revisando el número y los nombres de variables y muestras así como la presencia de valores faltantes. Además he obtenido estadísticos y gráficos de resumen para evaluar la presencia de patrones y de componentes importantes.

Con los análisis realizados, pude enterarme de que exsiten diferencias entre los grupos de tratamientos, por lo que es la concentración de metabolitos que se han encontrado en las muestras de orina de las partecipantes al estudio.

Finalmente, parte del trabajo ha sido cargar los archivos generados en un repositorio GitHub personal.

Objetivos del estudio

El objetivo del trabajo es demonstrar la capacidad de descargar datos, manejarlos, almacenarlos en un contenedor de clase `SummarizedExperiment` y llevar a cabo un simple análisis exploratorio utilizando R y librerías para analisis de datos ómicos, como las de `Bioconductor`, que nos permita entender la naturaleza de los mismos.

Además, es importante tener la capacidad de extraer conclusiones después del análisis realizado.

Finalmente, también el manejo del repositorio GitHub desempeña un papel importante en este trabajo.

Materiales y Métodos

Para realizar este trabajo, voy a utilizar un dataset del repositorio github proporcionado. En concreto, utilizaré el dataset "**2024-fobitools-UseCase_1**" que se encuentra en este enlace.

Se requiere crear, con los datos proporcionados, un contenedor de clase `SummarizedExperiment`. Para ello, voy a utilizar la librería `SummarizedExperiment` previamente instalada, que pertenece al paquete `Bioconductor` y que permite gestionar datos y metadatos ómicos.

Para llevar a cabo el análisis exploratorio de los datos, me enfocaré primero en un análisis general de los datos mediante gáficos como el *boxplot* y la realización de estadísticos descriptivos para obtener una visión general. Posteriormente, llevaré a cabo un análisis multivariante realizando un *Análisis de las Componentes Principales* (**PCA**) y un *Agrupamiento Jerárquico*, con el fin de identificar patrones y grupos relacionados entre los datos.

Utilizaré las siguientes librerías:

- **SummarizedExperiment**: del paquete `Bioconductor` para la creación del objeto `SummarizedExperiment` y su manejo;
- **ggplot2**: del paquete `tidyverse`, para la realización de gráficos como el *Boxplot*;
- **tidyr**: del paquete `tidyverse`, para la transformación de los datos, como con la función `pivot_wider()`;
- **dplyr**: del paquete `tidyverse` para la manipulación de los datos, como la selección de un subconjunto de columnas con la función `select()`
- **RColorBrewer**: un paquete en R que permite obtener una paleta de color a utilizar en los gráficos.
- **usethis**: para la configuración de Git
- **gitcreds**: para manejar y asociar el entorno a mi cuenta GitHub.

Resultados

Seleccionar un dataset de metabolómica

Para la realización de esta PEC, se nos pide realizar una exploración de datos de un dataset de metabolómica. Lo primero que hay que hacer, será seleccionar y descargar un *dataset* entre los que tenemos a disposición.

He entrado en el repositorio de GitHub:

https://github.com/nutrimetabolomics/metaboData/

y he seleccionado uno de los dataset disponible en la carpeta "*Datasets*". En concreto, he seleccionado el dataset en la carpeta "**2024-fobitools-UseCase_1**".

Este dataset almacena datos de un estudio realizado sobre dieciocho estudiantes universitarias sanas, de entre 21 y 29 años, con un IMC normal de 18.5 a 25 que fueron divididas en grupos y que se le pidio de consumir jugo de arándano o jugo de manzana durante un periodo establecido. En un momento dado, se le solicitó regresar a la clínica para proporcionar una muestra de orina en ayunas y analizar la concentración de los metabolitos contenidos. Después de un período de lavado de dos semanas, las participantes cambiaron al régimen alternativo y repitieron el protocolo. Tres partecipantes fueron excluidas y se quedarno 15 estudiantes. El estudio se porpone investigar las diferencias en los metabolitos encontrados en las muestras de orina proporcionadas.

En la carpeta del *dataset* encuentro tres archivos principales formato CSV que son:

- **features.csv** = el archivo con los datos numéricos del estudio;
- **metadata.csv** = el archivo con los datos sobre las muestras;
- **metaboliteNames.csv** = archivo con datos sobre los metabolitos.

Descargar los datos y cargarlos en RStudio

Para descargar los archivos, puedo simplemente hacer clic en cada uno de ellos y clicar en el botón para la descarga.

Otra manera es utilizar un código para la descarga directa de los archivos. En este caso tengo que acceder al contenido "raw" de los datos.

```
path <- "https://raw.githubusercontent.com/nutrimetabolomics/metaboData/main/Datasets/2024-fobitools-Us
feat <- paste0(path, "features.csv")
data_dir <- "C:/Users/Usuario/Desktop/UOC/tercer semestre/Análisis de datos ómicos/unidad 1/PEC1/data/"
dir.create(data_dir, showWarnings=FALSE)
feat_dest <- file.path(data_dir, "feat.csv")
download.file(feat, destfile = feat_dest, mode="wb")

metaName <- paste0(path, "metaboliteNames.csv")
metaName_dest <- file.path(data_dir, "metaName.csv")
download.file(metaName, destfile = metaName_dest, mode="wb")

meta <- paste0(path, "metadata.csv")
meta_dest <- file.path(data_dir, "meta.csv")
download.file(meta, destfile = meta_dest, mode="wb")
```

Una vez descargatos, puedo leerlos en RStudio para los análisis posteriores.

```
features <- read.csv(paste0(data_dir,"feat.csv"), sep = ';')
metadata <- read.csv(paste0(data_dir, "meta.csv"), sep =';')
metaboliteNames <- read.csv(paste0(data_dir,"metaName.csv"),sep = ';')
```

Crear un objeto SummarizedExperiment

Quiero crear un contenedor de tipo `SummarizedExperiment` que almacene los datos y los metadatos del dataset.

Para asociar los datos de las muestras con los metadatos correspondientes, es necesario que ambos compartan el mismo nombre. Puedo extraer el nombre del archivo de "**metadata**", ya que está almacenado en la columna "*ID*"

```
library(SummarizedExperiment)
rownames(features) <- metaboliteNames$names
colnames(features) <- metadata$ID
rownames(metadata) <- metadata$ID
```

La clase `SummarizedExperiment` es un tipo de objeto realizado para almacenar y organizar datos y metadatos ómicos. Puedo crear un objeto `SummarizedExperiment` con la función `SummarizedExperiment` del paquete `Bioconductor`.

Este tipo de objeto está compuesto por tres componentes principales:

- **ASSAY**

El **Assay** contiene los datos cuantitativos del experimento, guardado como una o más matrices en una lista. En este tipo de datos, cada fila corresponde a datos de las características o *features* y cada columna a datos de las muestras.

- **colData**

El **colData** corresponde a los metadatos de las muestras y guarda por esto, datos sobre las columnas de la matriz del assay.

- **rowData**

El **rowData** corresponde a los metadatos de las características. Está organizado como un dataframe donde cada fila representa una característica y cada columna una información asociada a la misma.

Enfocandome sobre los archivos del estudio que he seleccionado, puedo ver que el archivo "*features*" almacena datos del **Assay**, el archivo "*metadata*", datos de **colData** y el archivo "*metaboliteNames*", datos de **rowData**.

```
metabolite_info <- data.frame(
  original_name = metaboliteNames$names,
  pubchem_id = metaboliteNames$PubChem,
  kegg_id = metaboliteNames$KEGG,
  stringsAsFactors = FALSE
)
```

Ahora puedo crear el objeto `SummarizedExperiment` con los datos preparados.

```
se <- SummarizedExperiment(
    assays = list(counts = as.matrix(features)),
    colData = metadata
)
rowData(se) <- DataFrame(metabolite_info)
```

A través del enlace relativo al estudio, puedo encontrar metadatos del mismo que peudo añadir y almacenar en el objeto "*metadata*" del `SummarizedExperiment`.

```
metadata(se) <- list(database_name = "2024-fobitools-UseCase_1",
                     description = "The present study aimed to investigate overall metabolic changes ca
                     publication_title = "LC-MS Based Approaches to Investigate Metabolomic Differences
                     authors = "Liu Haiyan",
                     publication_DOI = "doi: 10.21228/M8J590",
                     institute = "University of Florida",
                     department = "Food Science and Nutrition",
                     Laboratory = "Gu",
                     email = "haiyan66@ufl.edu")
```

Análisis exploratorio de los datos

Puedo visualizar el objeto `SummarizedExperiment` y ver algunas características.

```
se
```

```
## class: SummarizedExperiment
## dim: 1541 45
## metadata(9): database_name description ... Laboratory email
## assays(1): counts
## rownames(1541): 10-Desacetyltaxuyunnanin C 10-Hydroxydecanoic acid ...
##   Zizybeoside I Zoxazolamine
## rowData names(3): original_name pubchem_id kegg_id
## colnames(45): b1 b10 ... c8 c9
## colData names(2): ID Treatment
```

Puedo enterarme de que he creado un objeto:

- de clase `SummarizedExperiment`
- de dimensión 1541 y 45, es decir, con 1541 filas y 45 columnas;
- que contiene 9 metadatos asociados al estudio;
- con una matriz numérica en el *assay*;
- con 1541 columnas que represnetan los metabolitos;
- con un elemento *rowData* que almacena 3 tipos de datos sobre los metabolitos;
- con 45 columnas que representan las muestras;
- con un elemento *colData* que almacena 2 tipos de datos relacionados con las muestras.

En el proceso de exploración descriptiva de los datos, queremos aclarar información básica como el número y los nombres de las variables y muestras, así como la presencia de valores faltantes. Además, puedo obtener estadísticas y gráficos de resumen.

Voy a mirar primero la composición del contenidos del `SummarizedExperiment`.

```
dim(se)
```

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

Como esperado, tenemos datos sobre 45 muestras de los cuales tenemos 1541 *features*.

La función `colData()`, nos proporciona información sobre los metadatos de las muestas.

```
colData(se)
```

```
## DataFrame with 45 rows and 2 columns
##              ID   Treatment
##     <character> <character>
## b1           b1    Baseline
## b10         b10    Baseline
## b11         b11    Baseline
## b12         b12    Baseline
## b13         b13    Baseline
## ...         ...         ...
## c4           c4   Cranberry
## c6           c6   Cranberry
## c7           c7   Cranberry
## c8           c8   Cranberry
## c9           c9   Cranberry
```

Podemos enterarnos de que el archivo metadata guarda datos sobre el identificador de las muestras (*ID*) y sobre el tipo de tratamiento (*Treatment*) que puede ser jugo de arándano (*Cranberry*), jugo de manzana (*Apple*), o control (*Baseline*).

Con la función `colnames()`, puedo observar los nombres de las columnas del contenedor `SummarizedExperiment`, o sea, el nombre asociado a cada muestra.

```
colnames(se)
```

```
##  [1] "b1"  "b10" "b11" "b12" "b13" "b14" "b15" "b16" "b17" "b2"  "b4"  "b6"
## [13] "b7"  "b8"  "b9"  "a1"  "a10" "a11" "a12" "a13" "a14" "a15" "a16" "a17"
## [25] "a2"  "a4"  "a6"  "a7"  "a8"  "a9"  "c1"  "c10" "c11" "c12" "c13" "c14"
## [37] "c15" "c16" "c17" "c2"  "c4"  "c6"  "c7"  "c8"  "c9"
```

Con la función `rownames()`, puedo observar los nombres de las filas de objeto `SummarizedExperiment`, o sea, el nombre asociado a cada metabolito.

```
rownames(se) |> head()
```

```
## [1] "10-Desacetyltaxuyunnanin C"
## [2] "10-Hydroxydecanoic acid"
## [3] "10-Oxodecanoate_1"
## [4] "11beta,21-Dihydroxy-5beta-pregnane-3,20-dione"
## [5] "1,1-Dichloroethylene epoxide"
## [6] "11-Hydroxycanthin-6-one"
```

con la función `rowData()` puedo visualizar los `rowData`, es decir, los metadatos asociado a las *features*.

```
rowData(se)
```

```
## DataFrame with 1541 rows and 3 columns
##                                                      original_name
##                                                        <character>
## 10-Desacetyltaxuyunnanin C                     10-Desacetyltaxuyunn..
## 10-Hydroxydecanoic acid                        10-Hydroxydecanoic a..
## 10-Oxodecanoate_1                                   10-Oxodecanoate_1
## 11beta,21-Dihydroxy-5beta-pregnane-3,20-dione 11beta,21-Dihydroxy-..
## 1,1-Dichloroethylene epoxide                   1,1-Dichloroethylene..
## ...                                                               ...
## Ungeremine                                                 Ungeremine
## Valacyclovir                                             Valacyclovir
## Versiconal                                                 Versiconal
## Zizybeoside I                                            Zizybeoside I
## Zoxazolamine                                              Zoxazolamine
##                                                  pubchem_id     kegg_id
##                                                 <character> <character>
## 10-Desacetyltaxuyunnanin C                          5460449      C15538
## 10-Hydroxydecanoic acid                               74300      C02774
## 10-Oxodecanoate_1                                   19734156      C02217
## 11beta,21-Dihydroxy-5beta-pregnane-3,20-dione       21145110      C05475
## 1,1-Dichloroethylene epoxide                          119521      C14857
## ...                                                      ...         ...
## Ungeremine                                            159646      C12189
## Valacyclovir                                           60773      C07184
## Versiconal                                          25203618      C20507
## Zizybeoside I                                       11972301      C17564
## Zoxazolamine                                           6103      C13841
```

En `rowData`, tenemos datos sobre las *features*. En este caso, tenemos la columna "*original_name*" con los nombres de los metabolitos, la columna "*pubchem_id*" con el número ID de Pubchem asociado al metabolito y la columna "*Keigg_id*" el identificador KEGG asociado.

Puedo buscar la presencia o usencia de valores faltantes o infinitos.

```
sum(is.na(assay(se)))
```

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

```
sum(is.infinite(assay(se)))
```

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

Podemos observar que tenemos **8190** datos faltantes en la matriz de los datos peroningún valor infinito.

Puedo crear un nuevo dataframe para analizar la media, la desviación estándar, el valor mínimo y el valor máximo de cada metabolito del estudio.

Dado que los valores son muy grandes, realizaré los análisis sobre los valores transformados a escala logarítmica para evitar posibles problemas.

```
assay_data <- as.matrix(assay(se))
assay_data <- apply(assay_data, 2, as.numeric)
assay_data_log <- log(assay_data + 1)
```

```r
assay_data_log <- apply(assay_data_log, 2, as.numeric)

stats <- apply(assay_data_log, 1, function(x) {
  finite_values <- x[!is.na(x) & is.finite(x)]
  if (length(finite_values) == 0) {
    c(mean = NA, sd = NA, min = NA, max = NA)
  } else {
    c(mean = mean(finite_values),
      sd = sd(finite_values),
      min = min(finite_values),
      max = max(finite_values))
  }
})

stats_df <- as.data.frame(t(stats))
stats_df['Metabolite'] <- metaboliteNames$names
head(stats_df)
```

```
##        mean       sd       min      max
## 1 13.020977 1.115179  9.622516 15.79513
## 2 14.519953 1.320546 10.969938 16.85046
## 3  6.565107 2.736774  0.000000 16.60052
## 4 12.016786 1.740626  7.669028 15.48698
## 5 12.440342 1.558449  9.711176 17.10017
## 6  7.902677 3.280651  0.000000 17.71140
##                                     Metabolite
## 1                    10-Desacetyltaxuyunnanin C
## 2                       10-Hydroxydecanoic acid
## 3                             10-Oxodecanoate_1
## 4 11beta,21-Dihydroxy-5beta-pregnane-3,20-dione
## 5                  1,1-Dichloroethylene epoxide
## 6                       11-Hydroxycanthin-6-one
```

Como sabemos que en el estudio las diferentes participantes han sido tratadas de forma diferente, puedo investigar los tipos de tratamientos.

```r
tipos_tratamientos <- unique(metadata$Treatment)

numero_tratamientos <- length(tipos_tratamientos)

cat("Numero de tratamientos en el estudio:", numero_tratamientos, "\n")
```

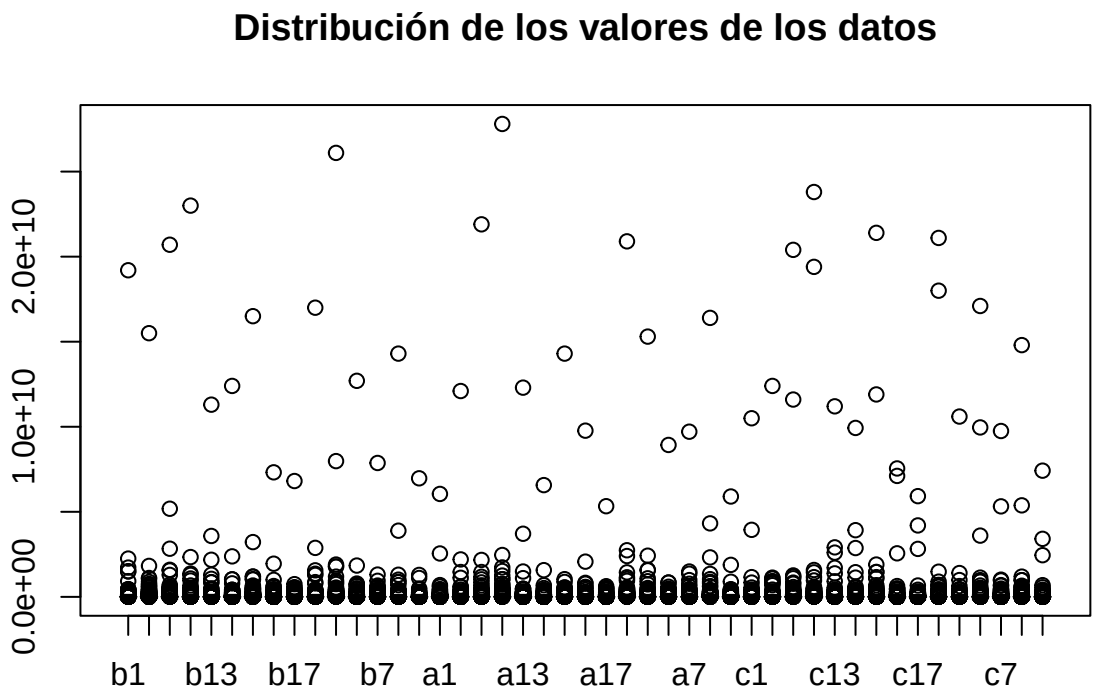
```
## Numero de tratamientos en el estudio: 3
```

```r
cat("Tipos de tratamientos:", tipos_tratamientos, "\n")
```

```
## Tipos de tratamientos: Baseline Apple Cranberry
```

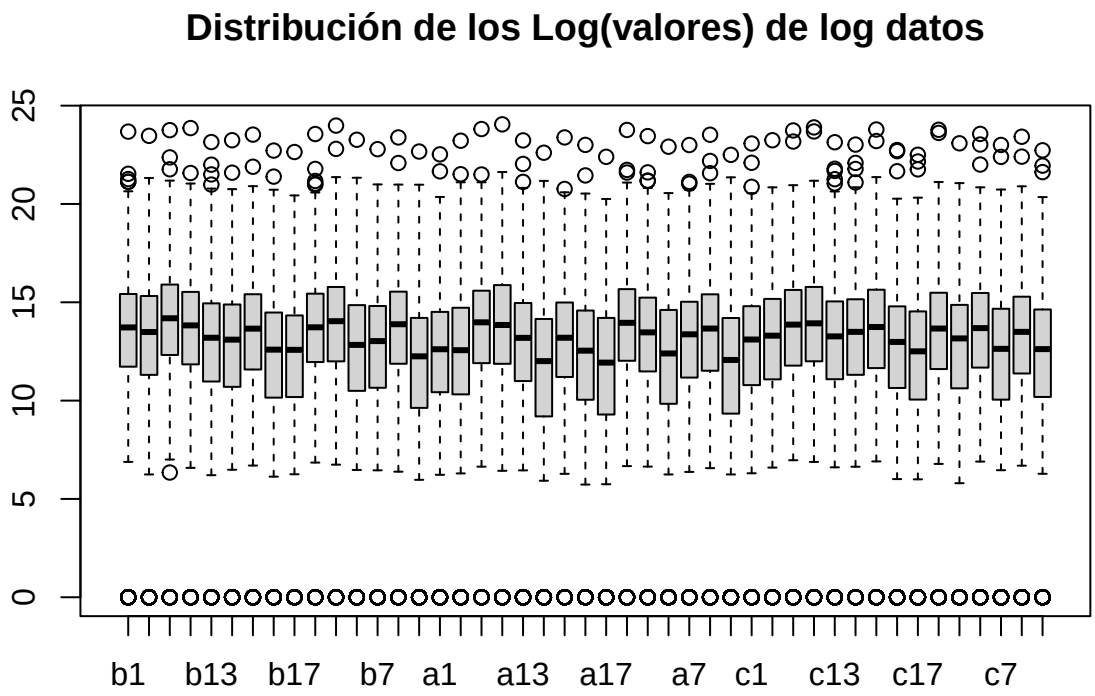
Puedo realizar un boxplot para visualizar la distribución de los datos en general.

```
boxplot(assay(se), main = "Distribución de los valores de los datos")
```



Aún en este caso, es mejor convertir los datos en escala logaritmica para la visualización.

```
par(mfrow=c(1,1))
boxplot(assay_data_log, main = "Distribución de los Log(valores) de log datos")
```



Dado que en el estudio tenemos tres tipos diferentes de tratamientos, puedo crear tres objetos `SummarizedExperiment` que almacenen los datos y metadatos de los distintos tratamientos, realizando un *subset*.

```
treatment_data <- colData(se)$Treatment

se_b <- se[, treatment_data == "Baseline"]
se_a <- se[, treatment_data == "Apple"]
se_c <- se[, treatment_data == "Cranberry"]
```

```
cat("Dimensión del se_a:", dim(se_a)[2], "mustras y", dim(se_a)[1], "metabolitas", "\nDimensión del se_
```

```
## Dimensión del se_a: 15 mustras y 1541 metabolitas
## Dimensión del se_b: 15 mustras y 1541 metabolitas
## Dimensión del se_c: 15 mustras y 1541 metabolitas
```

Cada subset está formado por 15 muestras y almacena datos sobre las 1541 *features*.

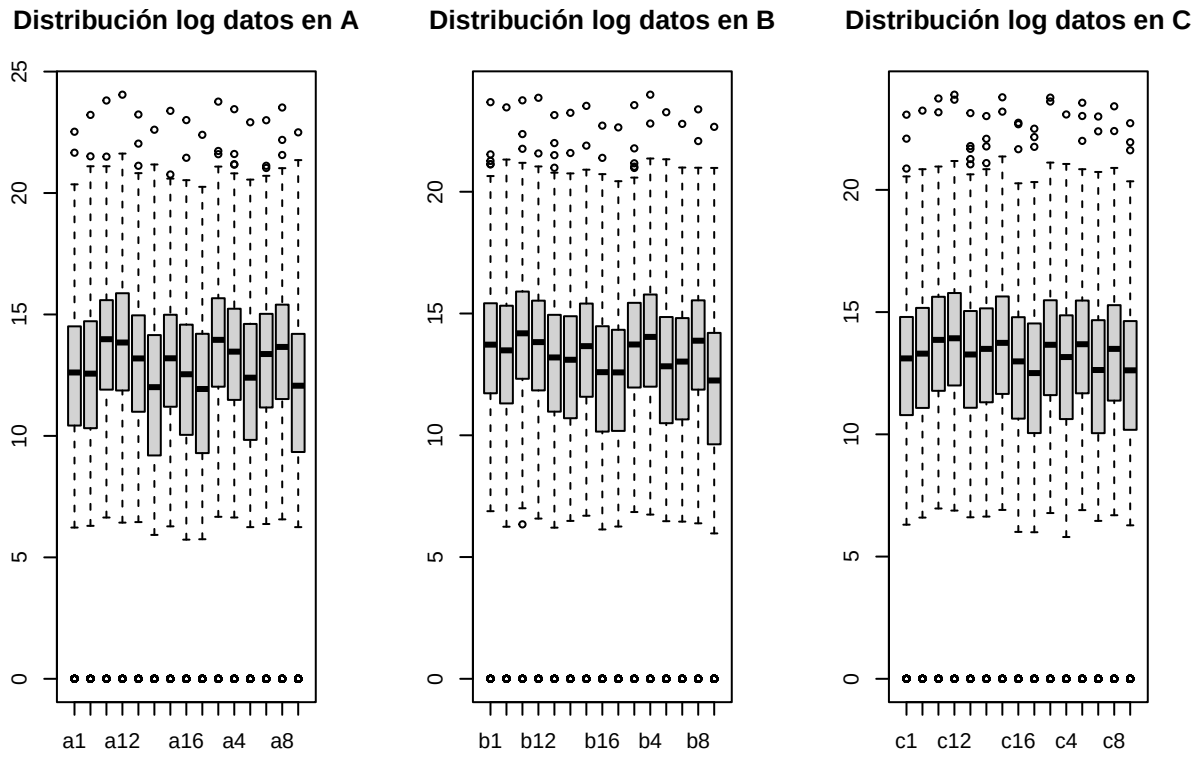
Puedo convertir en escala logaritmica también los datos contenidos en estos subsets para realizar gráficos y análisis.

```
data_matrix_a <- assay(se_a)
assay_data_a <- apply(data_matrix_a, 2, as.numeric)
assay_data_log_a <- log(assay_data_a + 1)
```

```
data_matrix_b <- assay(se_b)
assay_data_b <- apply(data_matrix_b, 2, as.numeric)
assay_data_log_b <- log(assay_data_b + 1)

data_matrix_c <- assay(se_c)
assay_data_c <- apply(data_matrix_c, 2, as.numeric)
assay_data_log_c <- log(assay_data_c + 1)
```

Puedo visualizar con un boxplot la distribución general de los datos en los diferente grupos de tratamientos.

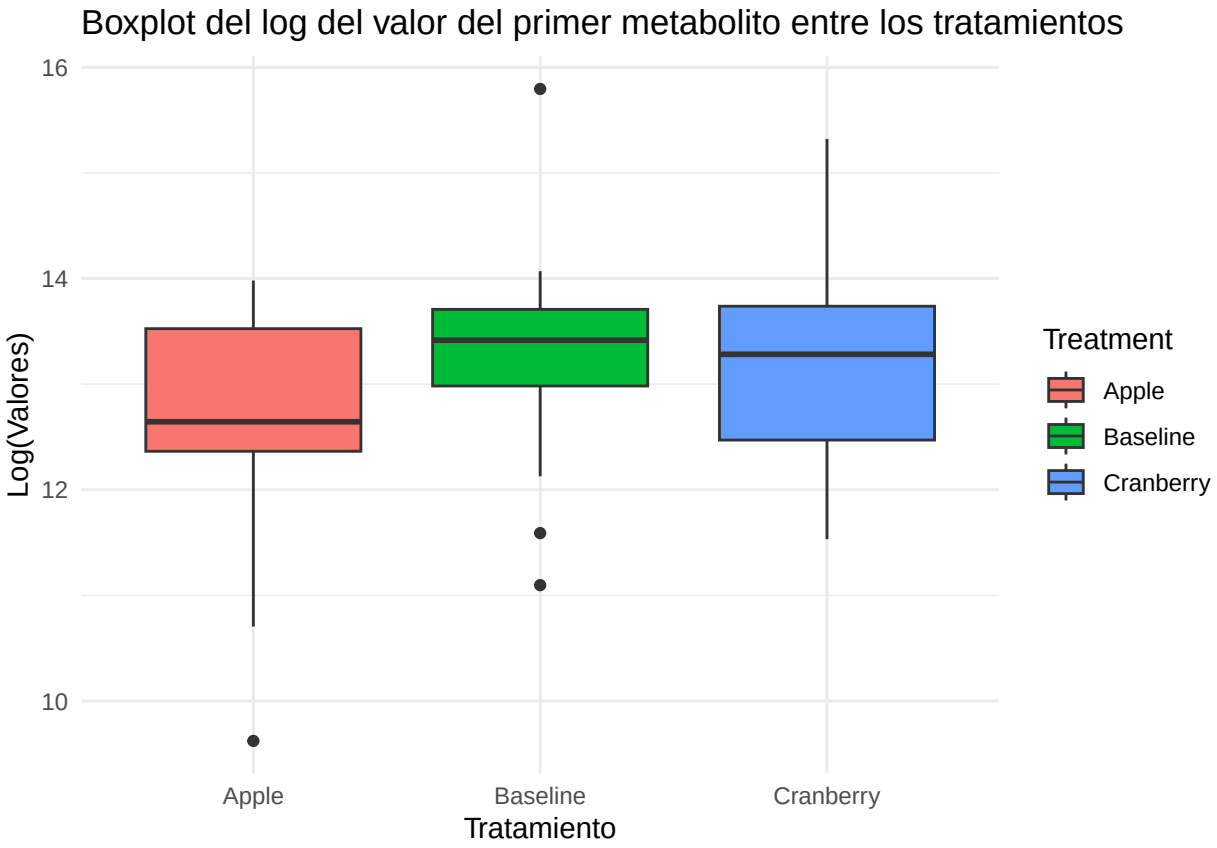
```
par(mfrow=c(1,3))

boxplot(assay_data_log_a, main = "Distribución log datos en A")
boxplot(assay_data_log_b, main = "Distribución log datos en B")
boxplot(assay_data_log_c, main = "Distribución log datos en C")
```



Puedo realizar un gráfico *boxplot* para ver la diferencia en un metabolito, por ejemplo, el primero del dataframe.

```
library(ggplot2)
feature_data <- assay(se)[1, ]
feature_data <- log(feature_data)
treatment_data <- colData(se)$Treatment

df <- data.frame(
```

```
  Treatment = treatment_data,
  Value = feature_data
)


ggplot(df, aes(x = Treatment, y = Value, fill = Treatment)) +
  geom_boxplot() +
  labs(title = "Boxplot del log del valor del primer metabolito entre los tratamientos",
       x = "Tratamiento",
       y = "Log(Valores)") +
  theme_minimal()
```



Como realizado anteriormente, puedo calcular los estadísticos para cada grupo de tratamiento.

```
stats <- apply(assay_data_log_a, 1, function(x) {
  finite_values <- x[!is.na(x) & is.finite(x)]
  if (length(finite_values) == 0) {
    c(mean = NA, sd = NA, min = NA, max = NA)
  } else {
    c(mean = mean(finite_values, na.rm = TRUE),
      sd = sd(finite_values, na.rm = TRUE),
      min = min(finite_values, na.rm = TRUE),
      max = max(finite_values, na.rm = TRUE))
  }
})
```

```r
stats_a_df <- as.data.frame(t(stats))
stats_a_df['Metabolite'] <- metaboliteNames$names
head(stats_a_df)
```

```
##        mean       sd       min       max
## 1 12.635946 1.189225  9.622516 13.981026
## 2 14.506984 1.119313 13.079458 16.744034
## 3  5.765034 2.353037  0.000000  7.313887
## 4 11.916042 2.354805  7.669028 15.257713
## 5 12.510985 2.058954  9.711176 17.100174
## 6  7.253958 3.137971  0.000000 10.936690
##                                       Metabolite
## 1                     10-Desacetyltaxuyunnanin C
## 2                        10-Hydroxydecanoic acid
## 3                             10-Oxodecanoate_1
## 4 11beta,21-Dihydroxy-5beta-pregnane-3,20-dione
## 5                   1,1-Dichloroethylene epoxide
## 6                        11-Hydroxycanthin-6-one
```

```r
stats <- apply(assay_data_log_b, 1, function(x) {
  finite_values <- x[!is.na(x) & is.finite(x)]
  if (length(finite_values) == 0) {
    c(mean = NA, sd = NA, min = NA, max = NA)
  } else {
    c(mean = mean(finite_values, na.rm = TRUE),
      sd = sd(finite_values, na.rm = TRUE),
      min = min(finite_values, na.rm = TRUE),
      max = max(finite_values, na.rm = TRUE))
  }
})

stats_b_df <- as.data.frame(t(stats))
stats_b_df['Metabolite'] <- metaboliteNames$names
head(stats_b_df)
```

```
##        mean       sd       min      max
## 1 13.279250 1.105690 11.097425 15.79513
## 2 14.322611 1.521439 10.969938 16.85046
## 3  7.551506 2.532736  6.455199 16.60052
## 4 12.221530 1.314580  9.798183 15.48698
## 5 12.743386 1.384462 11.233225 16.26652
## 6  9.138983 1.027621  7.501634 11.43606
##                                       Metabolite
## 1                     10-Desacetyltaxuyunnanin C
## 2                        10-Hydroxydecanoic acid
## 3                             10-Oxodecanoate_1
## 4 11beta,21-Dihydroxy-5beta-pregnane-3,20-dione
## 5                   1,1-Dichloroethylene epoxide
## 6                        11-Hydroxycanthin-6-one
```

```r
stats <- apply(assay_data_log_c, 1, function(x) {
  finite_values <- x[!is.na(x) & is.finite(x)]
```

```
  if (length(finite_values) == 0) {
    c(mean = NA, sd = NA, min = NA, max = NA)
  } else {
    c(mean = mean(finite_values, na.rm = TRUE),
      sd = sd(finite_values, na.rm = TRUE),
      min = min(finite_values, na.rm = TRUE),
      max = max(finite_values, na.rm = TRUE))
  }
})

stats_c_df <- as.data.frame(t(stats))
stats_c_df['Metabolite'] <- metaboliteNames$names
head(stats_c_df)
```

```
##        mean       sd       min      max
## 1 13.147735 1.012028 11.532738 15.32181
## 2 14.730264 1.350447 12.138869 16.53021
## 3  6.378782 3.134124  0.000000 13.59237
## 4 11.912787 1.477011  9.561068 14.95973
## 5 12.066656 1.106286 10.292179 14.22098
## 6  7.315089 4.523741  0.000000 17.71140
##                                     Metabolite
## 1                    10-Desacetyltaxuyunnanin C
## 2                       10-Hydroxydecanoic acid
## 3                             10-Oxodecanoate_1
## 4 11beta,21-Dihydroxy-5beta-pregnane-3,20-dione
## 5                  1,1-Dichloroethylene epoxide
## 6                       11-Hydroxycanthin-6-one
```

Puede resultar interesante investigar cuales metabolitos tienen, entre los grupos de tratamientos, la mayor diferencia entre la media. Para ello, voy a unir los dataframes con los estadísticos entre un solo dataframe, etiquetando las muestras como grupo “A”, “B” o “C”.

Luego puedo crear un dataframe con los metabolitos, la media de los valores y columnas para los diferentes grupos. Finalmente, puedo calcular las diferencias en las medias, ordenar del mayor al menor según la diferencia y visualizar los primeros valores.

```
library(tidyr)
library(dplyr)

stats_a_df$group <- "A"
stats_b_df$group <- "B"
stats_c_df$group <- "C"

all_stats_df <- bind_rows(stats_a_df, stats_b_df, stats_c_df)
selected_df <- select(all_stats_df, Metabolite, group, mean)
wide_stats_df <- pivot_wider(selected_df, names_from = group, values_from = mean)

wide_stats_df$max_diff <- pmax(wide_stats_df$A, wide_stats_df$B, wide_stats_df$C) -
                          pmin(wide_stats_df$A, wide_stats_df$B, wide_stats_df$C)
wide_stats_df <- wide_stats_df[order(-wide_stats_df$max_diff), ]
head(wide_stats_df)
```

```
## # A tibble: 6 x 5
```

```
##   Metabolite                                          A     B     C max_diff
##   <chr>                                           <dbl> <dbl> <dbl>    <dbl>
## 1 2,3,6-Trihydroxypyridine                         7.44  6.57 15.5      8.96
## 2 Robustaol A                                      5.63  4.39 11.9      7.46
## 3 Kikkanol A                                       2.88  2.69  9.66     6.97
## 4 5-Nitro-ortho-anisidine_1                        7.18  6.07 13.0      6.90
## 5 2-Amino-4-oxo-6-(1',2',3'-trihydroxypropyl)-diquin~ 2.57  1.54  8.10     6.55
## 6 L-Formylkynurenine_1                             7.97  8.29 13.8      5.84
```

Puedo realizar otro gráfico boxplot para visualizar la diferencia entre los valores del metabolito entre los diferentes grupos.

```
metabolite_max_diff <- wide_stats_df$Metabolite[1]
metabolite_index <- which(rownames(se) == metabolite_max_diff)
cat('El metabolito con máxima diferencia entre los tratamientos:', metabolite_max_diff)
```

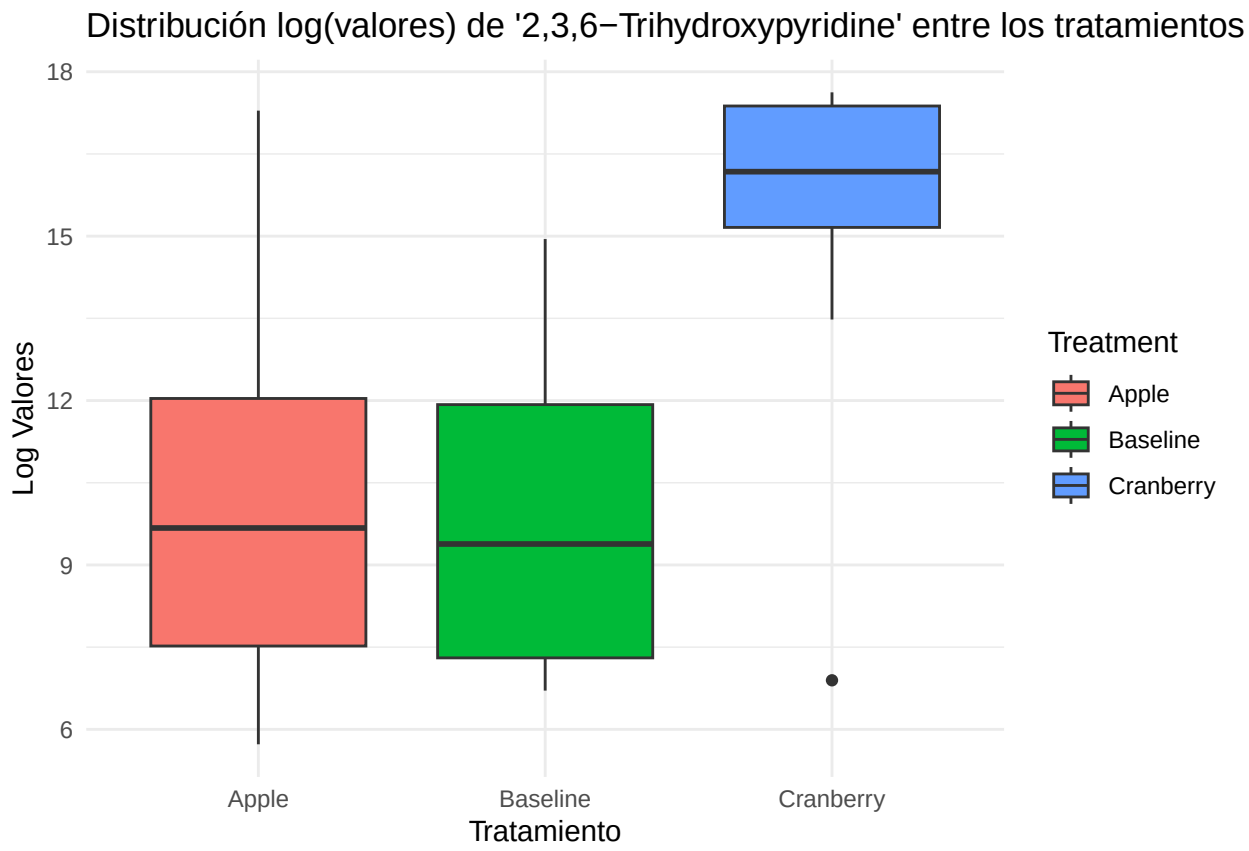
```
## El metabolito con máxima diferencia entre los tratamientos: 2,3,6-Trihydroxypyridine
```

```
print(metabolite_index)
```

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

```
feature_data <- assay(se)[45, ]
feature_data <- log(feature_data)
treatment_data <- colData(se)$Treatment


df <- data.frame(
  Treatment = treatment_data,
  Value = feature_data
)


ggplot(df, aes(x = Treatment, y = Value, fill = Treatment)) +
  geom_boxplot() +
  labs(title = "Distribución log(valores) de '2,3,6-Trihydroxypyridine' entre los tratamientos",
       x = "Tratamiento",
       y = "Log Valores") +
  theme_minimal()
```

```
## Warning: Removed 9 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```




```
metabolite_max_diff_2 <- wide_stats_df$Metabolite[2]
cat('El segundo metabolito con máxima diferencia entre los tratamientos:', metabolite_max_diff_2)
```

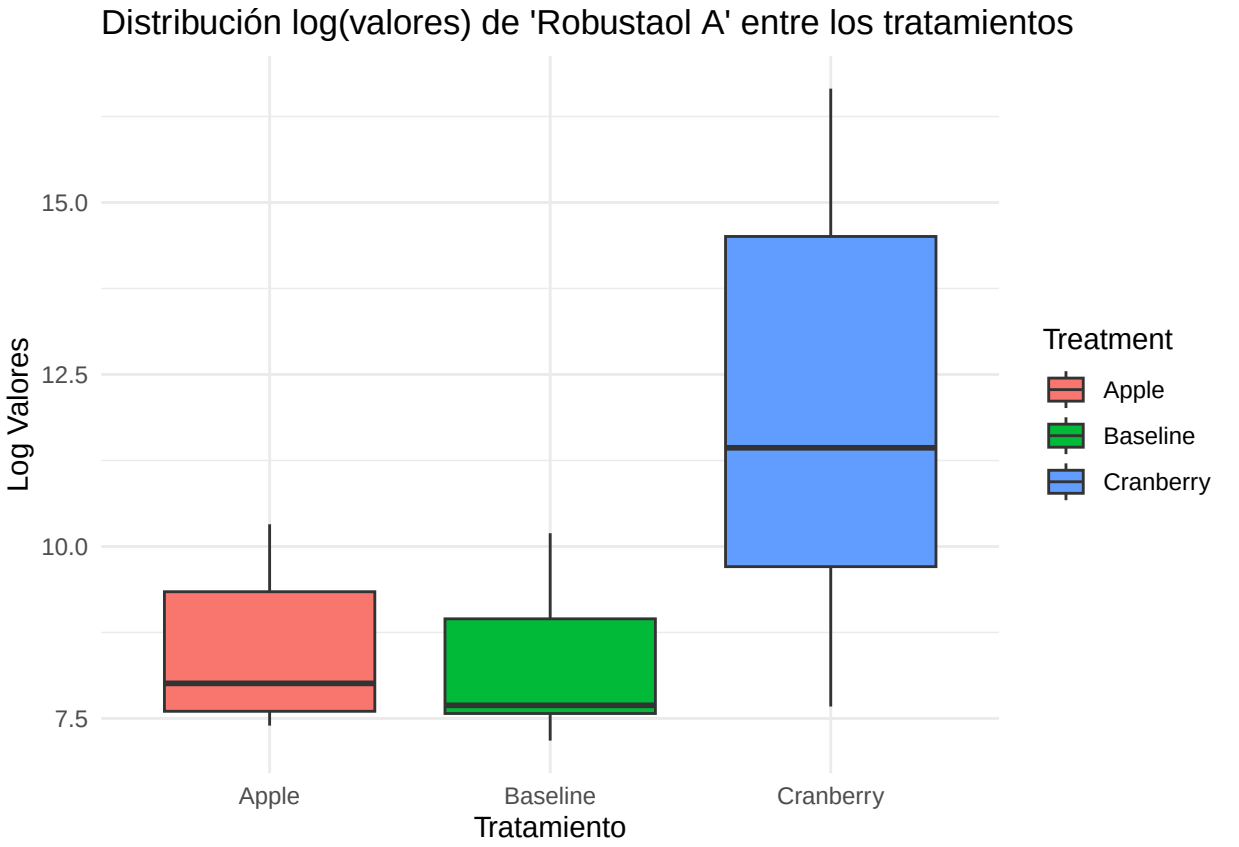
```
## El segundo metabolito con máxima diferencia entre los tratamientos: Robustaol A
```

```
metabolite_index_2 <- which(rownames(se) == metabolite_max_diff_2)
print(metabolite_index_2)
```

```
## [1] 1097
```

```
feature_data <- assay(se)[1097, ]
feature_data <- log(feature_data)
treatment_data <- colData(se)$Treatment


df <- data.frame(
  Treatment = treatment_data,
  Value = feature_data
)


ggplot(df, aes(x = Treatment, y = Value, fill = Treatment)) +
  geom_boxplot() +
  labs(title = "Distribución log(valores) de 'Robustaol A' entre los tratamientos",
```

```
      x = "Tratamiento",
      y = "Log Valores") +
 theme_minimal()
```



Puedo observar que, efectivamente, existen diferencias entre la distribución de los valores (convertidos a escala logarítmica) en al menos algunos metabolitos, y que la diferencia reside en las muestras tratadas con el jugo de arándano (*Cranberry*).

Análisis multivariante de los datos ómicos

Para reducir la dimensión, puedo llevar a cado un *Análisis de Componentes Principales* (**PCA**). Con este tipo de análisis, dada una matriz de datos **k x N** con **k** variables (*features*) en **N** muestras, se realiza una transformación de las variables en **k** nuevas componentes que reflejan las diferentes fuentes de la variabilidad de los datos pero no están correlacionados. En este caso, cada componente representa una fuente de diferente variabilidad.

Estas nuevas componentes se constuyen de forma que cada una explica más de la siguiente. Por esto, es posible utilizar los valores de las primeras componentes principales para obtener una representación de los datos en dimensión reducida con respecto al original. Esta transformación sirve para revelar los patrones dominantes y las principales tendencias en los datos.

Puedo crear una función "*plotPCA*" que permita realizar el gráfico de las dos primeras componentes principales según la matriz de datos que proporcione.

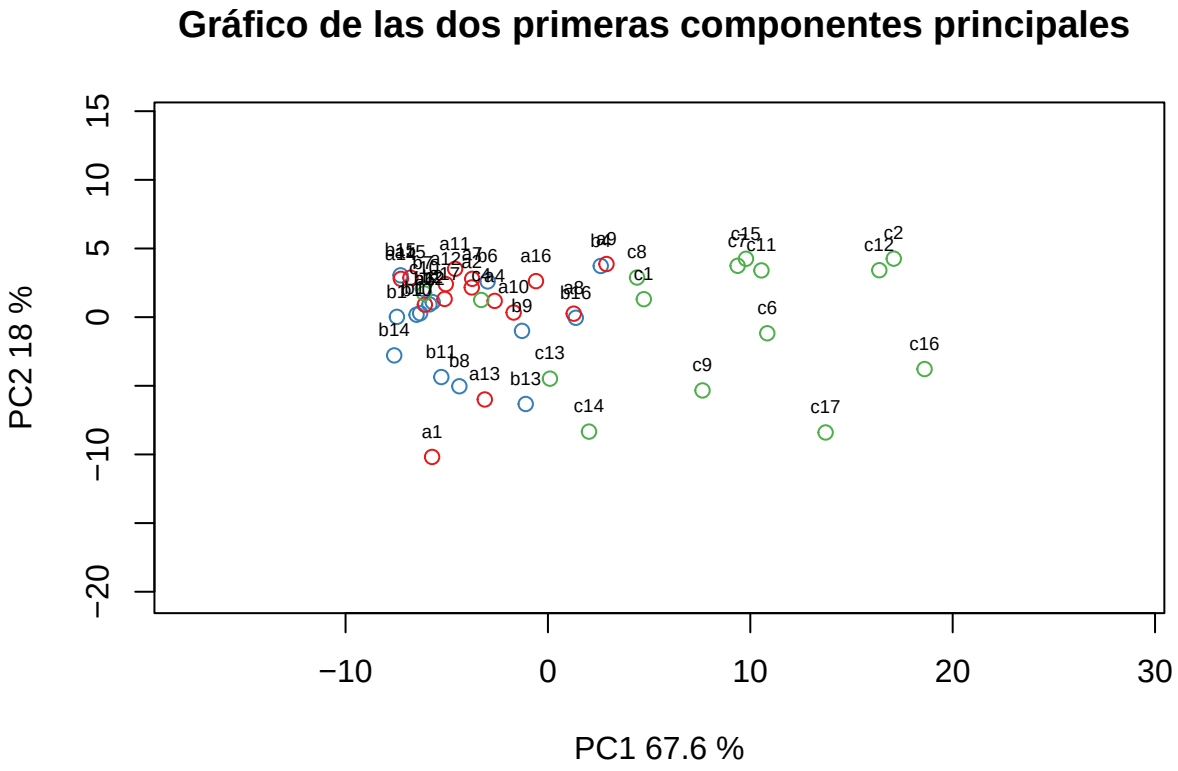
```r
plotPCA <- function ( X, labels=NULL, colors=NULL, dataDesc="", scale=FALSE, cex4text=0.8)
{
  pcX<-prcomp(X)
  loads<- round(pcX$sdev^2/sum(pcX$sdev^2)*100,1)
  xlab<-c(paste("PC1",loads[1],"%"))
  ylab<-c(paste("PC2",loads[2],"%"))
  if (is.null(colors)) colors=1
  plot(pcX$x[,1:2],xlab=xlab,ylab=ylab, col=colors,
       xlim=c(min(pcX$x[,1])-10, max(pcX$x[,1])+10),
       ylim=c(min(pcX$x[,2])-10, max(pcX$x[,2])+10))
  text(pcX$x[,1],pcX$x[,2], labels, pos=3, cex=cex4text)
  title(paste("Gráfico de las dos primeras componentes principales", dataDesc, sep=" "), cex=0.8)
}
```

Ahora puedo aplicar la transformación a la matriz que contiene los datos. Para evitar problemas debidos a la magnitud de los datos, es recomendable realizar el análisis en los datos transformados. Voy a llevar a cabo una transformación a escala normal, donde cada variable se ajusta para que tenga una media de 0 y una desviación estándar de 1. Además, voy a generar un gráfico en el que los diferentes puntos tengan un color distinto según el grupo de tratamiento al que pertenecen. Primero, tengo que eliminar los valores faltantes para realizar el análisis y el gráfico.

```r
library(RColorBrewer)

assay_data_scale <- scale(assay_data)

na_count <- sum(is.na(assay_data_scale))
cat("Número de NA en los datos escalados:", na_count, "\n")
```

```
## Número de NA en los datos escalados: 8190
```

```r
assay_data_clean <- assay_data_scale[rowSums(is.na(assay_data_scale)) == 0, ]

sampleNames <- metadata$ID
treatments <- metadata$Treatment
treatment_colors <- brewer.pal(length(unique(treatments)), "Set1")
colors <- treatment_colors[as.numeric(factor(treatments))]

plotPCA(t(assay_data_clean), labels=sampleNames, colors=colors, cex4text=0.6)
```



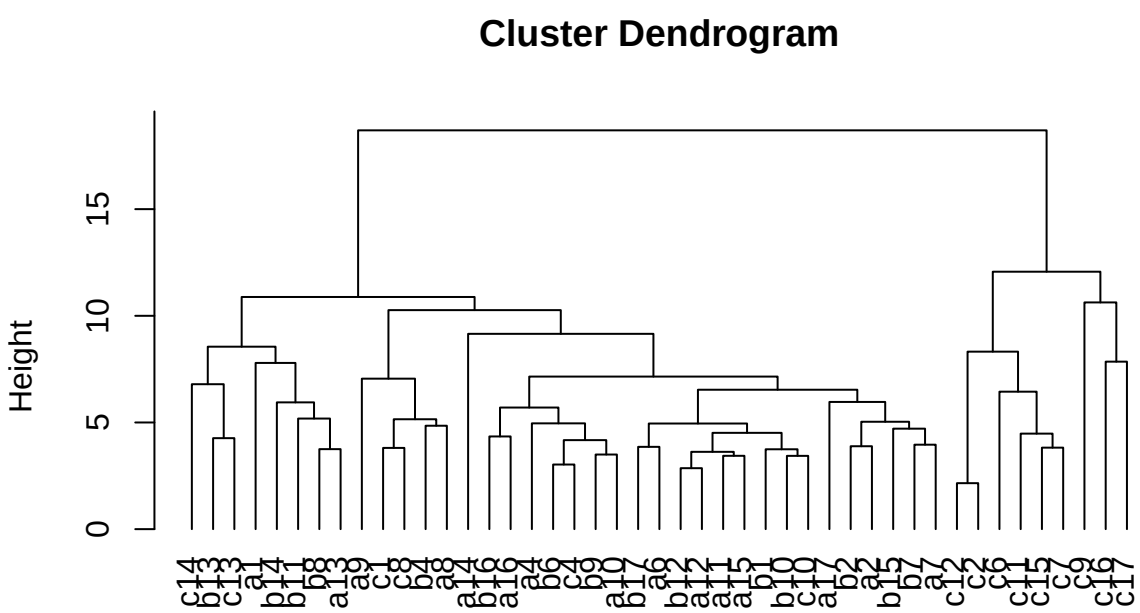
El análisis de las dos primeras componentes principales explica el 85.6% de la variabilidad de los datos (67.6% corresponde a la primera componente y 18% a la segunda componente). Puedo deducir que la primera componente parece estar relacionada con el tratamiento, ya que las muestras 'c', tratadas con jugo de arándano, se sitúan mayormente en la parte derecha del gráfico.

Otro análisis que puedo llevar a cabo es el descubrimiento de grupos en los datos para encontrar patrones, es decir:

- grupos de variables relacionadas que muestren cambios combinados entre condiciones;
- grupos de muestras que muestren patrones de expresión similares.

Una técnica útil para realizar el análisis es el utilizo del cluster jerarquico utilizando una distancia euclidiana y el método de agrupamiento promedio (*average linkage*) con la produción de un árbol o *dendograma*. Aún en este análisis se recomienda utilizar los datos transformados.

```
sampleNames <- metadata$ID
clust.euclid.average <- hclust(dist(t(assay_data_scale)),method="average")
plot(clust.euclid.average, hang=-1)
```

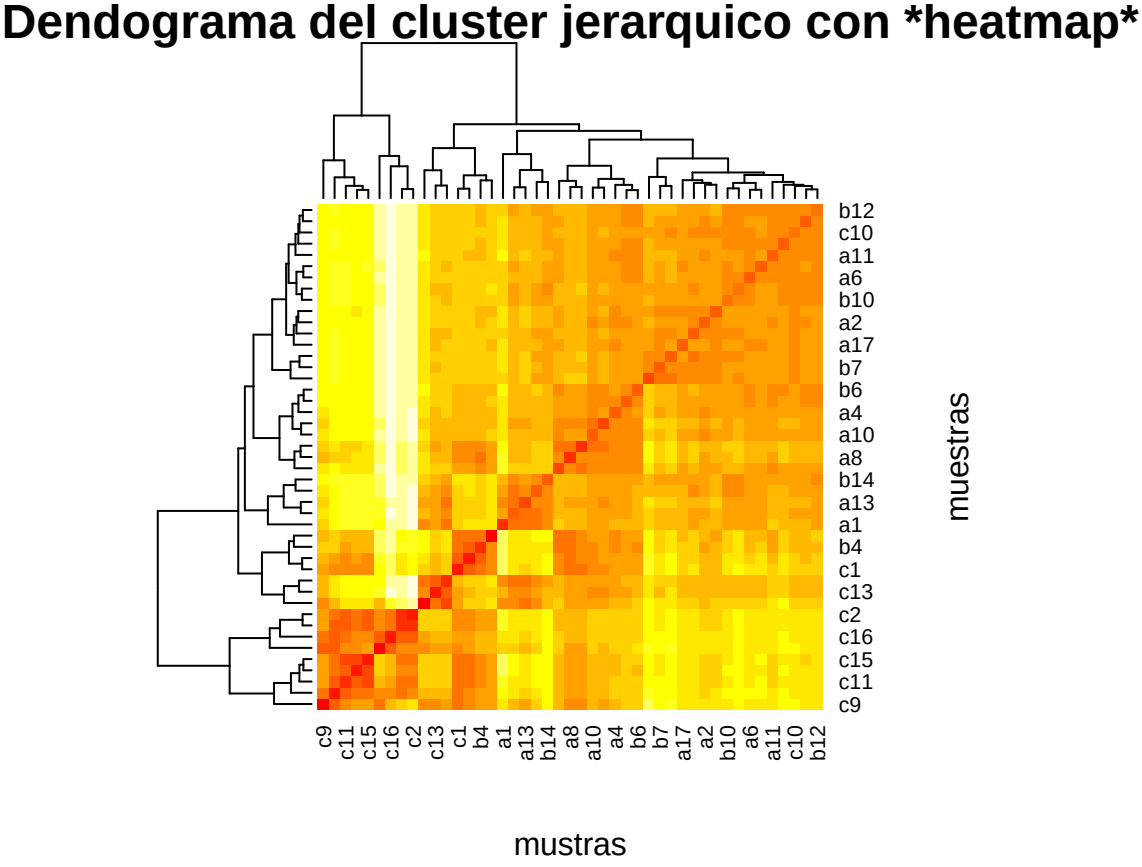


En el dendograma encontramos cada muestra en el eje horizontal y las distancias en el eje vertical. La altura indica la distancia a la que se fusionan los clústeres. En este caso, más alto es el punto de fusión, mayor diferencia entre los grupos hay. En nuestro gráfico, podemos observar la presencia de cuatro cluster principales.

Cabe destacar que dos de los clusters principales, representados en el lado derecho del gráfico, están representados únicamente por muestras del grupo "c" tratados con el jugo de arándano (*Cranberry*), subrayando la diferencia de este grupo con respecto a los demás.

Otra manera de visualizar el agrupamiento jerarquico es con el utilizo de *heatmap*, o mapa de calor, que ofrece una representación visual clara de las similitudes y diferencias en los perfiles de expresión de los metabolitos entre las diferentes muestras.

```
manDist <- dist(t(assay_data_scale))
heatmap (as.matrix(manDist),
         col=heat.colors(16),
         main = "Dendograma del cluster jerarquico con *heatmap*",
         xlab = "mustras",
         ylab = "muestras")
```



Discussión

A través del análisis exploratorio de los datos realizados sobre los valores transformados a escala logarítmica y normal, pude observar que realmente existen diferencias entre los distintos grupos considerados en el estudio. En particular, el grupo tratado con jugo de arándano presenta una concentración de metabolitos en las muestras de orina que es diferente de la de los demás grupos.

Esta conclusión se basa en el cálculo de estadísticas como la media, los valores mínimos y máximos, y las comparaciones sucesivas entre los grupos, así como en el análisis de las componentes principales y el agrupamiento jerárquico en clusters.

Para confirmar estos hallazgos, sería útil realizar análisis adicionales, como un estudio ANOVA, seguido de pruebas post hoc.

Repositorio GitHub

Finalmente, tengo que crear un repositorio de GitHub que contenga:

- el informe
- el objeto contenedor con los datos y los metadatos en formato binario (.Rda)
- el código R para la exploración de los datos
- los datos en formato texto

- los metadatos acerca del dataset en un archivo markdown.

Primero, debo configurar Git en mi entorno de trabajo. Para hacerlo, en la consola de RStudio, utilizo la función `edit_git_config()` de la librería `usethis`. Al ejecutar esta función, se abrirá el archivo **.gitconfig**, donde podré especificar información relacionada con mi cuenta de GitHub.

En este archivo, tengo que definir mi nombre de usuario y mi correo electrónico. Estos tienen que coincidir con los que tengo asociados a mi cuenta de GitHub.

[user]

name = 'Daniele Sebastiani'

email = 'd.seba.farma@gmail.com'

Ahora tengo que crear un nuevo proyecto. Como ya había creado los archivos con los que he trabajado, voy a asociar el nuevo proyecto con una carpeta ya existente.

A través de:

- File
- New project
- Existing directory
- especificar la ruta de la carpeta con los datos.
- Create Project

Ahora, regresando a la console de RStudio, utilizo la función `create_github_token()` de la librería `usethis` para generar el token para la creación del repositorio. Voy a copiar el token recién creado.

Sucesivamente, utilizo la función `gitcreds_set()` de la librería `gitcreds` y voy a pegar el token para asociar los archivos al repositorio.

Finalmente, utilizo la función `use_github()` de la librería `usethis` para asociar el entorno al repositorio GitHub y poder cargar los archivos.

Voy a generar el objeto contenedor con los datos y los metadatos en formato binario (.Rda).

```
save(se, file = "SummarizedExperiment_PEC1.Rda")
```

Puedo utilizar los botónes "Commit" y "Push" para actualizar los datos al modificarlos y subirlos al repositorio GitHub.

Finalmente, voy a pegar aquí el enlace de mi repositorio.

https://github.com/Danie7e/PEC_1-Analisis-de-Datos-Omicos-Sebastiani-Daniele

Enlace GitHub