PEC 1. Análisis de datos ómicos

Joaquín Vila García

2024-11-04

PEC 1

El enlace al repositorio de GitHub en el que se almacenan los archivos de esta PEC es:

https://github.com/normoblasto/Vila-Garcia-Joaquin-PEC1

El dataset de metabolómica elegido para esta PEC es 2024-fobitools-UseCase_1, del repositorio "https://github.com/nutrimetabolomics/metaboData/" de GitHUb. Como se puede leer en la descripción, este paquete está sacado de Metabolomics Workbench (ID ST000291). Está basado en un estudio realizado para analizar las diferencias metabolómicas en la orina de mujeres jóvenes después de beber jugo de arándano o jugo de manzana.

Para desarrollar el ejercicio vamos a usar las librerías de R "readr", "metabolomicsWorkbenchR", "SummarizedExperiment".

Lo primero que haremos será cargar dichas librerías o paquetes:

```
# Carga del paquete readr
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.2.3
```

```
# Carga del paquete dplyr
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Carga del paquete SummarizedExperiment de BiocManager
library(SummarizedExperiment)
```

```
## Loading required package: MatrixGenerics
```

```
## Loading required package: matrixStats
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following object is masked from 'package:dplyr':
##
```

```
##     count
##
## Attaching package: '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
## Loading required package: GenomicRanges
## Warning: package 'GenomicRanges' was built under R version 4.2.2
## Loading required package: stats4
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
## 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, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min
## Loading required package: S4Vectors
## Warning: package 'S4Vectors' was built under R version 4.2.2
##
## Attaching package: 'S4Vectors'
## The following objects are masked from 'package:dplyr':
##
##     first, rename
```

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

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

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

## Loading required package: GenomeInfoDb

## Warning: package 'GenomeInfoDb' was built under R version 4.2.2

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

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

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

```r
# Carga del paquete metabolomicsWorkbenchR de BiocManager
library(metabolomicsWorkbenchR)
```

Posteriormente cargamos el dataset elegido (2024-fobitools-UseCase_1). En mi caso, he decidido usar los datos raw directamente desde mi repositoroi GitHub, para que el archivo de R pueda ejecutarse fácilmente desde cualquier ordenador.

```r
# Cargar los datos de características (1541 variables x 45 muestras)
features <- read_csv2("https://raw.githubusercontent.com/normoblasto/Vila-Garcia-Joaquin-PEC1/main/2024-
```

```
## i Using "',' " as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.

## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)

## Rows: 1541 Columns: 45

## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (27): b1, b10, b11, b15, b16, b4, b7, b8, a1, a10, a11, a13, a14, a15, a...
```

```
## num (18): b12, b13, b14, b17, b2, b6, b9, a12, a16, a17, a4, a8, c1, c12, c1...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(features)
```

```
## # A tibble: 6 x 45
##   b1     b10   b11      b12    b13    b14 b15   b16      b17     b2 b4        b6
##   <chr>  <chr> <chr>  <dbl>  <dbl>  <dbl> <chr> <chr>  <dbl>  <dbl> <chr>  <dbl>
## 1 443489 9410~ 7570~ 6.12e5 858000 1.85e5 6710~ 1140~ 1.08e5 3.83e5 5930~ 7.24e6
## 2 107754 8300~ 6790~ 2.08e7 320000 1.29e6 1580~ 2340~ 1.18e6 1.26e6 1500~ 4.95e5
## 3 95430~ 1500  890   1.62e7   1250 9.68e2 657   809   7.67e2 8.26e2 2810  1.14e3
## 4 11011~ 2760~ 35700 6.31e5 369000 2.42e5 4720~ 5320~ 1.80e4 2.43e5 1310~ 1.58e5
## 5 52811~ 7060~ 1210~ 1.16e7 164000 4.24e5 7490~ 2670~ 3.05e6 9.91e4 1360~ 4.52e5
## 6 440341 6340  34100 3.19e4   9440 9.26e4 6740  14400 8.18e3 8.98e3 4610  1.01e4
## # i 33 more variables: b7 <chr>, b8 <chr>, b9 <dbl>, a1 <chr>, a10 <chr>,
## #   a11 <chr>, a12 <dbl>, a13 <chr>, a14 <chr>, a15 <chr>, a16 <dbl>,
## #   a17 <dbl>, a2 <chr>, a4 <dbl>, a6 <chr>, a7 <chr>, a8 <dbl>, a9 <chr>,
## #   c1 <dbl>, c10 <chr>, c11 <chr>, c12 <dbl>, c13 <dbl>, c14 <chr>, c15 <dbl>,
## #   c16 <dbl>, c17 <chr>, c2 <chr>, c4 <chr>, c6 <dbl>, c7 <chr>, c8 <chr>,
## #   c9 <chr>
```

```
# Cargar los metadatos (45 muestras con 2 columnas)
metadata <- read.csv2("https://raw.githubusercontent.com/normoblasto/Vila-Garcia-Joaquin-PEC1/main/2024-
head(metadata)
```

```
##    ID Treatment
## 1  b1  Baseline
## 2 b10  Baseline
## 3 b11  Baseline
## 4 b12  Baseline
## 5 b13  Baseline
## 6 b14  Baseline
```

```
# Cargar los nombres de metabolitos (1541 variables con 3 columnas)
metabolite_names <- read.csv2("https://raw.githubusercontent.com/normoblasto/Vila-Garcia-Joaquin-PEC1/ma
head(metabolite_names)
```

```
##                                          names  PubChem   KEGG
## 1                    10-Desacetyltaxuyunnanin C  5460449 C15538
## 2                       10-Hydroxydecanoic acid    74300 C02774
## 3                            10-Oxodecanoate_1 19734156 C02217
## 4 11beta,21-Dihydroxy-5beta-pregnane-3,20-dione 21145110 C05475
## 5                  1,1-Dichloroethylene epoxide   119521 C14857
## 6                       11-Hydroxycanthin-6-one   337601 C09212
```

Un objeto de tipo *SummarizedExperiment* es una estructura de datos diseñada para almacenar y organizar datos de experimentos biológicos y genómicos. Permite mantener una relación entre datos de expresión (matrices numéricas) y metadatos (información adicional sobre las muestras y características). Sus principales componentes son:

- **Assay(s)**: Este componente contiene una o más matrices de datos numéricos.
- **rowData:** Son los metadatos de las filas, proporcionando información adicional sobre cada característica. En nuestro experimento, la información contenida en la segunda fila de metaboliteNames.csv, que corresponde a los nombres de los metabolitos.

- **colData:** Son los metadatos de las columnas. En nuestro caso, la columna ID del archivo metadata.csv.
- **metadata**: contiene metadatos adicionales.

Una vez tenemos claro qué queremos incluir en el objeto SummarizedExperiment, podemos crear las matrices que lo formarán.

```
#Creamos la matriz principal de datos (features)
assay <- as.matrix(features)

# Matriz de filas (metabolitos)
row_data <- data.frame(metabolite_names[,1])
head(row_data)
```

```
##                           metabolite_names...1.
## 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
```

```
# Matriz de columnas
col_data <- data.frame(metadata[,1])
head(col_data)
```

```
##   metadata...1.
## 1            b1
## 2           b10
## 3           b11
## 4           b12
## 5           b13
## 6           b14
```

Para poder crear el objeto SE, tienen que coincidir los nombres de las columnas (rownames) de metadata y de features. Por ello, renombraremos las filas y columnas para asegurar que los nombres coincidan entre features y metadata.

```
colnames(features) <- metadata$ID
rownames(metadata) <- metadata$ID
```

Si queremos que los nombres de las filas sean los de los metabolitos, podemos definirlo de este modo:

```
rownames(metabolite_names) <- metabolite_names$names
```

```
# Creamos el objeto SummarizedExperiment usando los valores arriba seleccionados
se <- SummarizedExperiment(assays = list(counts = assay),

                           rowData = metabolite_names,
                           colData = metadata)
```

Guardamos el archivo "SummarizedExperimentData.Rda":

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

Ahora haremos una breve exploración del objeto SummarizedExperiment que hemos creado.

```
# Cargar el archivo
load("SummarizedExperimentData.Rda")
```

```
# Revisar la estructura general del objeto
se
```

```
## class: SummarizedExperiment
## dim: 1541 45
## metadata(0):
## assays(1): counts
## rownames(1541): 10-Desacetyltaxuyunnanin C 10-Hydroxydecanoic acid ...
##   Zizybeoside I Zoxazolamine
## rowData names(3): names PubChem KEGG
## colnames(45): b1 b10 ... c8 c9
## colData names(2): ID Treatment
```

Vemos que la clase del archivo es SummarizedExperiment y que tiene una dimensión de 1541x45. Es decir, se han medido 1541 metabolitos en un total de 45 muestras.

Vemos que "metadata(0)", con lo que no hemos almacenado información adicional como metadatos.

Las columnas están etiquetadas con nombres que representan las muestras individuales (b1, b10, etc.). Esto indica que hay 45 muestras únicas en este experimento.

Más en detalle:

```
# Ver los metadatos de las muestras
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
```

```
# Resumen de las columnas en `colData`
summary(colData(se))
```

```
## [1] "DataFrame object of length 2 with 0 metadata columns"
```

```
# Ver los metadatos de las variables
rowData(se)
```

```
## DataFrame with 1541 rows and 3 columns
##                                                          names
##                                                    <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        KEGG
##                                                <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
```

```
# Primeras filas de `rowData`
head(rowData(se))
```

```
## DataFrame with 6 rows and 3 columns
##                                                                 names
##                                                           <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..
## 11-Hydroxycanthin-6-one                         11-Hydroxycanthin-6-..
##                                                  PubChem        KEGG
##                                                <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
## 11-Hydroxycanthin-6-one                             337601      C09212
```

RowData contiene los nombres de los metabolitos analizados.

Comprobaremos si hay valores NA:

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

```
## [1] 7826
```

Vemos que hay 7826 valores NA.

Comprobaremos si hay valores infinitos:

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

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

No hay valores infinitos.

Vamos a calcular media, mediana y DE de los valores de las muestras de los metabolitos. Para ello, hay que eliminar los valores NA.

```r
assay(se) <- apply(assay(se), c(1, 2), function(x) if(is.na(as.numeric(x))) NA else as.numeric(x))

metabolite_stats <- data.frame(
  mean = apply(assay(se), 1, mean, na.rm = TRUE),
  median = apply(assay(se), 1, median, na.rm = TRUE),
  sd = apply(assay(se), 1, sd, na.rm = TRUE)
)

head(metabolite_stats)
```

```
##                                                  mean  median      sd
## 10-Desacetyltaxuyunnanin C                     819072.5  590000 1212009
## 10-Hydroxydecanoic acid                       4173133.0 1390000 5285152
## 10-Oxodecanoate_1                              604067.3     822 2804045
## 11beta,21-Dihydroxy-5beta-pregnane-3,20-dione  862292.8  229000 1968845
## 1,1-Dichloroethylene epoxide                  1499983.2  188000 4428957
## 11-Hydroxycanthin-6-one                       1137742.5    4405 7414464
```

Podemos, por ejemplo, analizar qué metabolito es más frecuente en cada grupo de tratamiento:

```r
# Extraer los datos de expresión y los datos de tratamiento
data_matrix <- assay(se)
metadata <- colData(se)

# Convertir el data_matrix en un data.frame y añadir el tratamiento como columna
data_df <- as.data.frame(t(data_matrix))
data_df$Treatment <- metadata$Treatment

# Calcular el promedio de cada metabolito por grupo de tratamiento
mean_by_treatment <- data_df %>%
  group_by(Treatment) %>%
  summarise(across(everything(), mean, na.rm = TRUE))
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(everything(), mean, na.rm = TRUE)`.
## i In group 1: `Treatment = "Apple"`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```r
# Encontrar el metabolito más frecuente (promedio más alto) en cada grupo
# Primero, identificamos el valor promedio más alto (Max_Average) para cada grupo
mean_by_treatment <- mean_by_treatment %>%
  rowwise() %>%
  mutate(
    Max_Average = max(c_across(-Treatment), na.rm = TRUE)
  )

# Ahora buscamos el nombre del metabolito correspondiente al promedio máximo
```

```r
most_frequent_metabolite <- mean_by_treatment %>%
  rowwise() %>%
  mutate(
    Most_Frequent_Metabolite = names(select(., -Treatment))[which.max(c_across(-Treatment) == Max_Averag
  ) %>%
  ungroup() %>%
  select(Treatment, Most_Frequent_Metabolite, Max_Average)

# Mostrar los resultados
most_frequent_metabolite
```

```
## # A tibble: 3 x 3
##   Treatment Most_Frequent_Metabolite   Max_Average
##   <chr>     <chr>                            <dbl>
## 1 Apple     Heterodendrin_1          518383333333.
## 2 Baseline  Heterodendrin_1          736944666706.
## 3 Cranberry Heterodendrin_1          664180000000
```

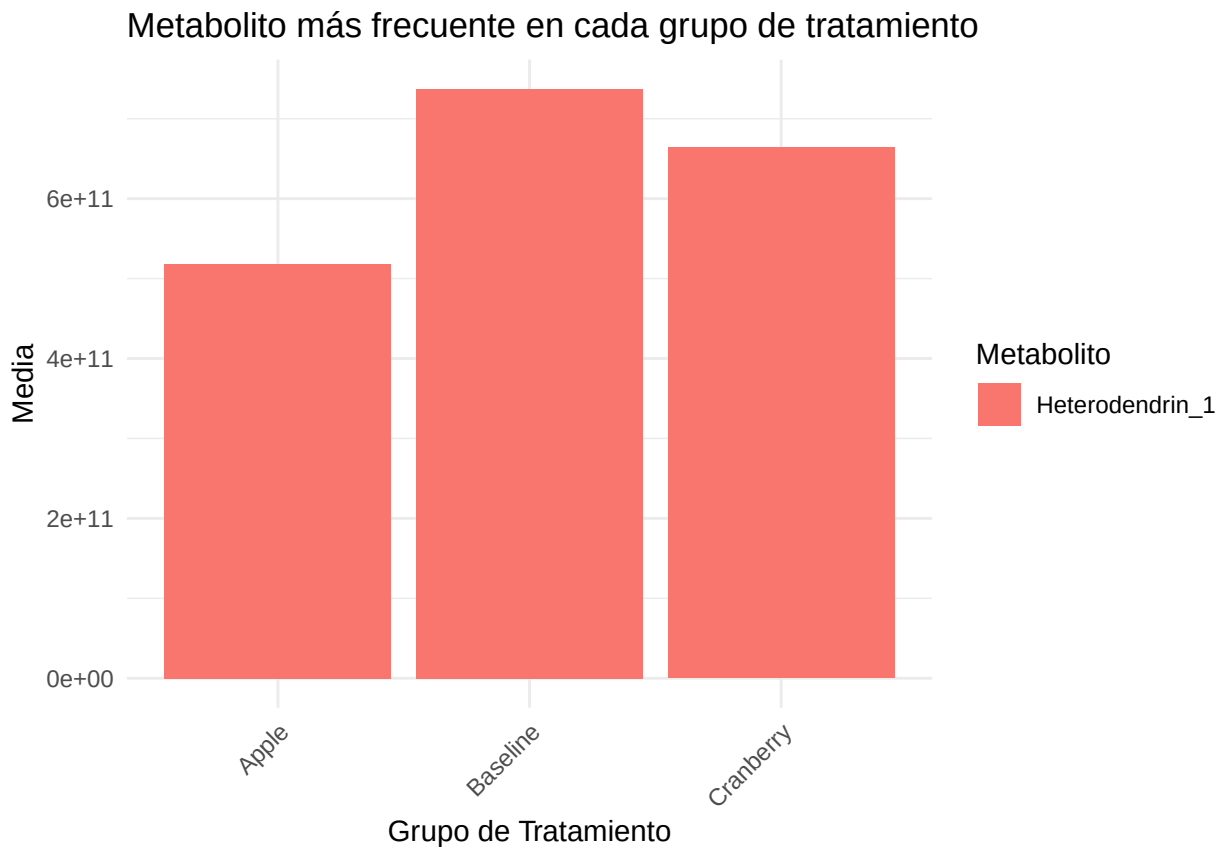
Vemos que el más frecuente es Heterodendrin_1 en los tres grupos.

En formato gráfica de barras:

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```r
# Crear la gráfica de barras para los metabolitos más frecuentes en cada tratamiento
ggplot(most_frequent_metabolite, aes(x = Treatment, y = Max_Average, fill = Most_Frequent_Metabolite)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(
        title = "Metabolito más frecuente en cada grupo de tratamiento",
        x = "Grupo de Tratamiento",
        y = "Media",
        fill = "Metabolito"
    ) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Podemos también hacer un análisis de los componentes principales. Para ello, podemos usar PCAtools.

```
#cargamos las librerías

if (!requireNamespace("DESeq2", quietly = TRUE))
    BiocManager::install("DESeq2")

if (!requireNamespace("PCAtools", quietly = TRUE))
    BiocManager::install("PCAtools")
library(DESeq2)
```

```
## Warning: package 'DESeq2' was built under R version 4.2.2
```

```
library(PCAtools)
```

```
## Loading required package: ggrepel
```

```
##
## Attaching package: 'PCAtools'
```

```
## The following objects are masked from 'package:stats':
##
##     biplot, screeplot
```

```
# Extraer la matriz de conteos del objeto `SummarizedExperiment`
count_data <- assay(se)

# Identificar filas y columnas sin `NA`
rows_to_keep <- apply(count_data, 1, function(row) all(!is.na(row)))
```

```r
cols_to_keep <- apply(count_data, 2, function(col) all(!is.na(col)))
```

```r
# Filtrar el `assay`, `rowData` y `colData` en el objeto `SummarizedExperiment`
se_filtered <- se[rows_to_keep, cols_to_keep]
```

```r
# Comprobar si existen valores `NA` en el `assay` del objeto filtrado
sum(is.na(assay(se_filtered)))  # Esto debería ser 0
```

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

```r
# Extraer la matriz de conteos del objeto filtrado
count_data_filtered <- assay(se_filtered)

# Aplicar transformación logarítmica para estabilizar la varianza
log_counts <- log2(count_data_filtered + 1)

# Realizar el PCA con la matriz transpuesta para que las muestras sean filas
pca_result <- prcomp(t(log_counts), scale. = TRUE)
```

```
## Error in svd(x, nu = 0, nv = k): a dimension is zero
```

Nota: No he sido capaz de completar el análisis de componentes principales porque algún tipo de problema estoy teniendo al depurar los datos para eliminar los valores NA.

En cualquier caso, la conclusión de esta PEC es que los *SummarizedExperiment* son objetos que nos permiten integrar eficientemente datos de expresión con metadatos de muestras y anotaciones de características, facilitando análisis complejos en estudios de alto rendimiento.