PEC1 Análisis de Datos Ómicos

Carmen Almudena Navarro Andrés

2024-11-02

Contents

RESUMEN

Este trabajo emplea el análisis de datos metabolómicos para estudiar los cambios en la expresión de metabolitos en la orina y sangre de mujeres sanas de entre 21 y 29 años tras beber zumo de manzana y zumo de arándano rojo, ambos ricos en procianidinas.

Los datos metabolómicos fueron organizados en un objeto de tipo SummarizedExperiment, y posteriormente se llevó a cabo un análisis exploratorio de datos (EDA). Este incluyó métodos de visualización como gráficos de densidad, PCA, volcano plots y heatmaps, así como pruebas como el t-test para la identificación de metabolitos diferencialmente expresados entre los tratamientos.

En cuanto a la expresión metabolómica global, apenas se observaron cambios entre los grupos. Sin embargo, sí que hubo diferencias significativas entre los grupos en varios metabolitos. Estos metabolitos diferencialmente expresados podrían ser indicativos de la respuesta metabólica específica a los compuestos presentes en la manzana y arándano rojo. Estos hallazgos proporcionan una base para futuras investigaciones sobre los efectos de las procianidinas en el metabolismo humano y posibles beneficios para la salud.

OBJETIVOS DEL ESTUDIO

El objetivo principal de este estudio fue analizar los cambios metabolómicos en mujeres jóvenes sanas inducidos por el consumo de zumo de arándanos y manzana. Para alcanzar este objetivo, el estudio se centró en:

1. Evaluar las variaciones en los perfiles metabólicos de la orina y plasma después del consumo de zumo de arándano rojo y zumo de manzana en comparación con los valores basales.

2. Identificar metabolitos específicos que respondan diferencialmente al consumo de arándano rojo o manzana.

MATERIALES Y MÉTODOS

Diseño del estudio

Se seleccionaron 18 mujeres sanas en etapa universitaria, de entre 21 y 29 años, con un índice de masa corporal de 18.5 a 25. A cada paciente se le proporcionó una lista de alimentos ricos en procianidinas que debían evitar durante el estudio (arándanos, manzanas, uvas, chocolate, y ciruelas). Finalmente, se excluyeros a 3 participantes del experimento por errores al seguir el protocolo.

El día 7 tras comenzar la dieta pobre en procianidinas, después de un ayuno nocturno, se recolectaron muestras de orina y sangre en ayunas como valores basales. Posteriormente, las participantes fueron asignadas aleatoriamente a dos grupos: el grupo de zumo de arándano y el grupo de zumode manzana (9 participantes en cada grupo). A cada participante se le proporcionaron 6 botellas de zumo (de 250 mL cada una) que debían consumir entre el día 7 y 9. Luego, en el día 10 se extrajeron las muestras de sangre y orina. Tras un período de lavado de dos semanas, los grupos intercambiaron los tratamientos y el protocolo se repitió. Todas las muestras de plasma y orina se almacenaron a -80°C hasta el momento del análisis.

Análisis metabolómico

Para evaluar los perfiles metabolómicos en plasma y orina, se empleó la cromatografía líquida con espectrometría de masas (LCMS).

Estos datos se descargaron del repositorio de GitHub metaboData (https://github.com/nutrimetabolomics/metaboData/), divididos en los archivos features.csv, que incluía columnas con el ID de las participantes y filas con el ID de PubChem del metabolito, con sus respectivos valores cuantitativos; metaboliteNames.csv, con el nombre del metabolito, además de su ID de PubChem y su ID de KEGG; y metadata.csv, que incluía el ID del paciente y su tratamiento. La información sobre el dataset y el estudio se encontró en la página web www.metabolomicsworkbench.org, a partir del ID del estudio indicado en el repositorio.

En primer lugar, se generó un objeto del tipo SummarizedExperiment a partir de estos datos, y luego se procedió a realizar el análisis exploratorio de datos. Este incluyó un resumen estadístico de los metabolitos por grupo de tratamiento, una representación de la distribución de los niveles de expresión por tratamiento con un gráfico de densidad, un análisis de componentes principales (PCA), un análisis de expresión diferencial de metabolitos con t-test y volcano plot; y un heatmap y agrupación jerárquica de los metabolitos con mayor variabilidad para agruparlos en función de su similitud.

RESULTADOS Y DISCUSIÓN

Primero vamos a cargar nuestros datos para proceder a construir el contenedor del tipo SummarizedExperiment

```
#Configuramos el directorio de trabajo y cargamos los 3 archivos
setwd("C:/Users/carme/OneDrive/Documents/Análisis de datos ómicos/PEC1")
features <- read.csv("features.csv", sep= ";")
metabolite_names <- read.csv("metaboliteNames.csv", sep=";")
metadata <- read.csv("metadata.csv", sep= ";")
# Visualizamos los datos
View(features)
```

```r
View(metabolite_names)
View(metadata)
write.table(metadata, "metadatos.md")
```

Ahora vamos a comprobar que los nombres de los IDs en metadata coincidan con los nombres de las columnas en featurespara poder crear el objeto SummarizedExperiment

```r
metadata_ids <- metadata$ID
feature_columns <- colnames(features)
stopifnot(metadata_ids == feature_columns)


# Transponemos  features para que las filas representen muestras y las columnas representen metabolitos

features <- t(features)

# Reordenamos metadata para que coincida con las filas de features, ya que las muestras deben estar en

metadata <- metadata[match(rownames(features), metadata$ID), ]

# Verificamos que los nombres de las filas de features coincidan con los IDs de metadata

stopifnot(rownames(features) == metadata$ID)

# Reordenamos metabolite_names para que coincida con las columnas de features

metabolite_names <- metabolite_names[match(colnames(features), metabolite_names$PubChem), ]
stopifnot(colnames(features) == metabolite_names$PubChem)
#https://uclouvain-cbio.github.io/bioinfo-training-02-rnaseq/summarizedexperiments.html
```

```r
# Cargamos el paquete Summarized Experiment
library(SummarizedExperiment)
```

```
## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## 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

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: '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, setdiff, table,
##     tapply, union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

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

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

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

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

## Loading required package: GenomeInfoDb
```

```
## 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
```

```
# Convertimos geatures en una matriz (lo volvemos a transponer)
features <- t(features)
assay_data <- as.matrix(features)

# Usamos el ID de PubChem como nombre de fila en metabolite_info, para que coincida con el nombre de fi
metabolite_info <- DataFrame(metabolite_names)
rownames(metabolite_info) <- metabolite_info$PubChem

# Usamos el ID como nombre de fila en metadata para que coincida con el nombre de columna en featuress
colData <- DataFrame(metadata)
rownames(colData) <- colData$ID

# Ahora construimos el objeto de SummarizedExperiment
se <- SummarizedExperiment(
  assays = list(counts = assay_data),
  rowData = metabolite_info,
  colData = colData
)

se
```

```
## class: SummarizedExperiment
## dim: 1541 45
## metadata(0):
## assays(1): counts
## rownames(1541): 443489 107754 ... 53297445 11954209
## rowData names(3): names PubChem KEGG
## colnames(45): b1 b10 ... c8 c9
## colData names(2): ID Treatment
```

assays contiene los datos metabólicos numéricos, rowData la información sobre los metabolitos y colData la información sobre la muestra y los tratamientos.

```r
#Guardamos el objeto SummarizedExperiment en formato binario
save(se, file = "summarized_experiment.Rda")
```

Ahora procedemos a realizar el análisis exploratorio de datos. Como hay varias filas exclusivamente con valores NA, procedemos a eliminarlas para que no afecten nuestra exploración de los datos.

```r
# Primero extraemos los datos metabolómicos y el tipo de tratamiento de cada muestra
assay_data <- assay(se)
assay_data <- assay_data[complete.cases(assay_data), ]
treatment <- as.character(colData(se)$Treatment)

# Partimos nuestros datos según el tratamiento
baseline_data <- assay_data[, treatment == "Baseline"]
apple_data <- assay_data[, treatment == "Apple"]
cranberry_data <- assay_data[, treatment == "Cranberry"]

# Creamos un dataframe con la media, mediana y varianza de cada metabolito, y ordenamos de mayor a meno

# Comenzamos con los datos basales
baseline_stats<-data.frame(
  Metabolite = rownames(baseline_data),
  Mean = rowMeans(baseline_data),
  Median = apply(baseline_data, 1, median),
  Variance = apply(baseline_data, 1, var)
)
baseline_stats$Metabolite_Name <- rowData(se)$names[match(baseline_stats$Metabolite, rownames(rowData(s
baseline_stats <- baseline_stats[order(-baseline_stats$Mean), ]
head(baseline_stats)
```

```
##         Metabolite        Mean   Median     Variance
## 588            588 14511333333 1.43e+10 3.627603e+19
## 5462194    5462194  1915200000 1.56e+09 1.921681e+18
## 464            464  1568180000 1.30e+09 3.550610e+18
## 586            586  1149466667 1.03e+09 1.150163e+18
## 1175          1175   972933333 9.58e+08 9.171407e+16
## 4784          4784   805666667 8.67e+08 2.533297e+17
##                              Metabolite_Name
## 588                             CREATININE_1
## 5462194 (E)-4-(Trimethylammonio)but-2-enoate
## 464                                Hippurate
## 586                                 CREATINE
## 1175                                   URATE
## 4784        Phenylmethanesulfonyl fluoride_1
```

```r
# Realizamos el mismo procedimiento con el grupo de zumo de manzana.
apple_stats <- data.frame(
  Metabolite = rownames(apple_data),
  Mean = rowMeans(apple_data),
  Median = apply(apple_data, 1, median),
  Variance = apply(apple_data, 1, var)
)
apple_stats$Metabolite_Name <- rowData(se)$names[match(apple_stats$Metabolite, rownames(rowData(se)))]
```

```r
apple_stats <- apple_stats[order(-apple_stats$Mean), ]
head(apple_stats)
```

```
##         Metabolite        Mean   Median     Variance
## 588            588 12884000000 1.21e+10 4.407188e+19
## 464            464  1675000000 1.88e+09 1.308273e+18
## 5462194    5462194  1244133333 8.40e+08 1.027995e+18
## 1175          1175   881266667 8.85e+08 6.995864e+16
## 586            586   727533333 4.99e+08 5.209690e+17
## 227            227   588886667 4.07e+08 1.958853e+17
##                             Metabolite_Name
## 588                            CREATININE_1
## 464                               Hippurate
## 5462194 (E)-4-(Trimethylammonio)but-2-enoate
## 1175                                 URATE
## 586                                CREATINE
## 227                              Anthranilate
```

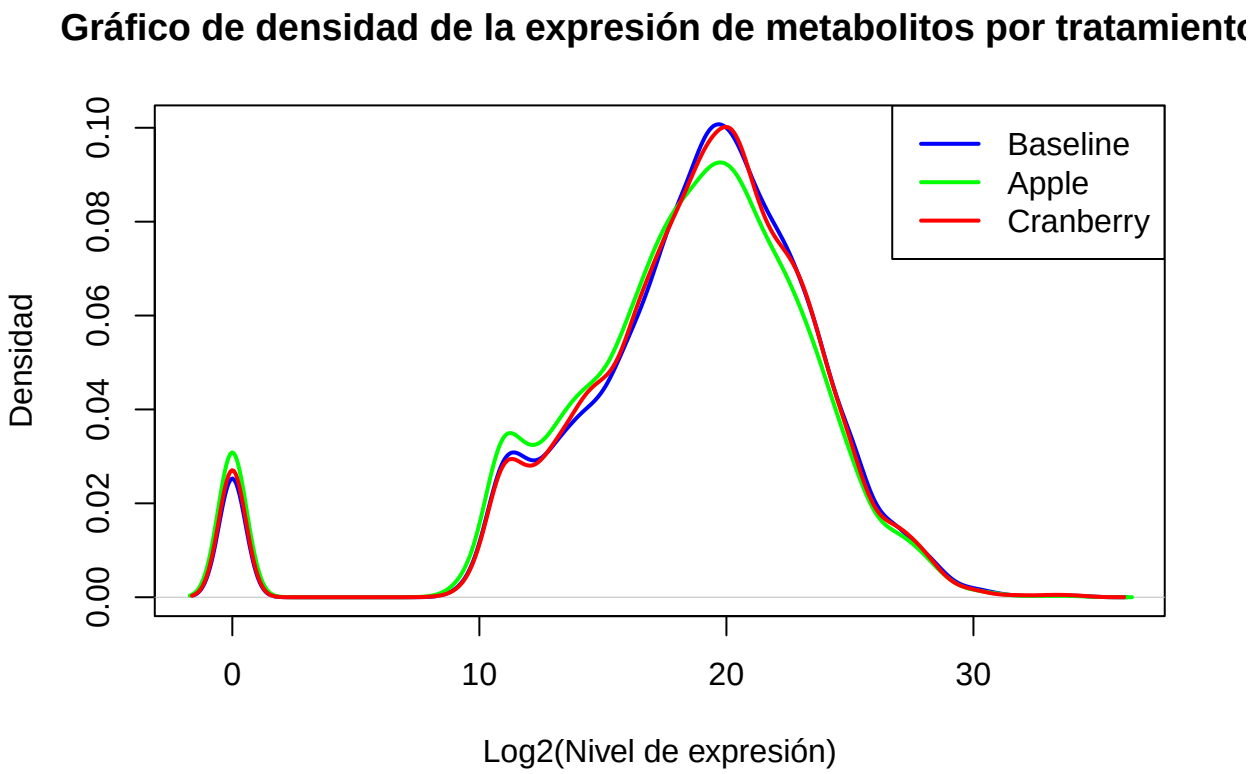
```r
# Realizamos el mismo procedimiento con el grupo de zumo de arándano rojo.
cranberry_stats <- data.frame(
  Metabolite = rownames(cranberry_data),
  Mean = rowMeans(cranberry_data),
  Median = apply(cranberry_data, 1, median),
  Variance = apply(cranberry_data, 1, var)
)
cranberry_stats$Metabolite_Name <- rowData(se)$names[match(cranberry_stats$Metabolite, rownames(rowData
cranberry_stats <- cranberry_stats[order(-cranberry_stats$Mean), ]
head(cranberry_stats)
```

```
##         Metabolite        Mean   Median     Variance
## 588            588 13591333333 1.12e+10 3.349554e+19
## 464            464  7191600000 5.32e+09 3.357908e+19
## 5462194    5462194  1793733333 1.26e+09 1.310448e+18
## 1175          1175  1016066667 1.13e+09 8.044492e+16
## 4784          4784   735866667 5.40e+08 1.616436e+17
## 586            586   635500000 5.86e+08 3.050248e+17
##                             Metabolite_Name
## 588                            CREATININE_1
## 464                               Hippurate
## 5462194 (E)-4-(Trimethylammonio)but-2-enoate
## 1175                                 URATE
## 4784         Phenylmethanesulfonyl fluoride_1
## 586                                CREATINE
```

Vamos a realizar ahora un gráfico de densidad para estudiar la distribución de la expresión de metabolitos por grupo de tratamientos, para analizar las diferencias entre grupos. Realizamos la transformación logarítmica de los datos para reducir la asimetría en los valores.

```r
# Transformación logarítmica de los datos
log_assay_data <- log2(assay_data + 1)

baseline_values <- as.vector(log_assay_data[, treatment == "Baseline"])
```

```r
apple_values <- as.vector(log_assay_data[, treatment == "Apple"])
cranberry_values <- as.vector(log_assay_data[, treatment == "Cranberry"])


baseline_density <- density(baseline_values, na.rm = TRUE)
apple_density <- density(apple_values, na.rm = TRUE)
cranberry_density <- density(cranberry_values, na.rm = TRUE)

# Realizamos el gráfico de densidad
plot(baseline_density, main = "Gráfico de densidad de la expresión de metabolitos por tratamiento",
     xlab = "Log2(Nivel de expresión)", ylab = "Densidad", col = "blue", lwd = 2,
     ylim = range(0, max(baseline_density$y, apple_density$y, cranberry_density$y)))
lines(apple_density, col = "green", lwd = 2)
lines(cranberry_density, col = "red", lwd = 2)
legend("topright", legend = c("Baseline", "Apple", "Cranberry"),
       col = c("blue", "green", "red"), lwd = 2)
```

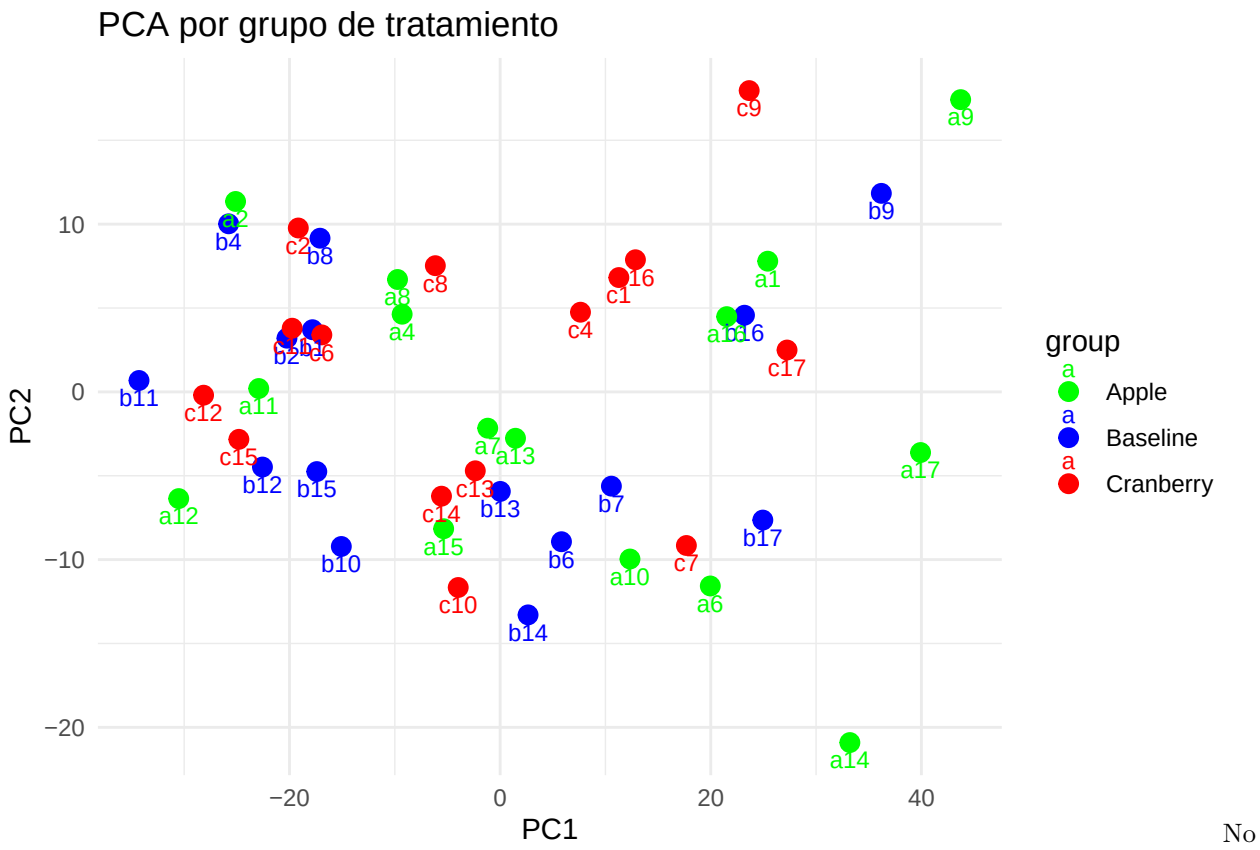


Podemos ver que las curvas de densidad son muy parecidas, lo que indica que la distribución de los niveles de expresión global de metabolitos es comparable en los tres grupos, por lo que los tratamientos no provocaron un gran cambio a nivel global. Los picos en las curvas podrían representar grupos de metabolitos con niveles de expresión similares, por lo que podrían pertenecer a la misma clase o estar relacionados de alguna forma. Los cambios en la expresión parecen estar reservados a ciertos metabolitos.

Ahora realizamos el Análisis de Componentes Principales (PCA) para reducir las dimensiones de los datos

```r
# Escalamos los datos
assay_data_scaled <- t(scale(t(log_assay_data)))

#Realizamos el PCA
pca_results <- prcomp(t(assay_data_scaled), scale. = TRUE)
```

```
groups <- colData(se)$Treatment
pca_data <- as.data.frame(pca_results$x)
pca_data$sample <- colnames(assay_data)
pca_data$group  <- groups

# Seleccionamos los 2 componentes principales para hacer el gráfico
library(ggplot2)

ggplot(pca_data, aes(x = PC1, y = PC2, color = group, label = sample)) +
  geom_point(size = 3) +
  geom_text(vjust = 1.5, size = 3) +
  labs(title = "PCA por grupo de tratamiento", x = "PC1", y = "PC2") +
  theme_minimal() +
  scale_color_manual(values = c("green", "blue", "red"))
```

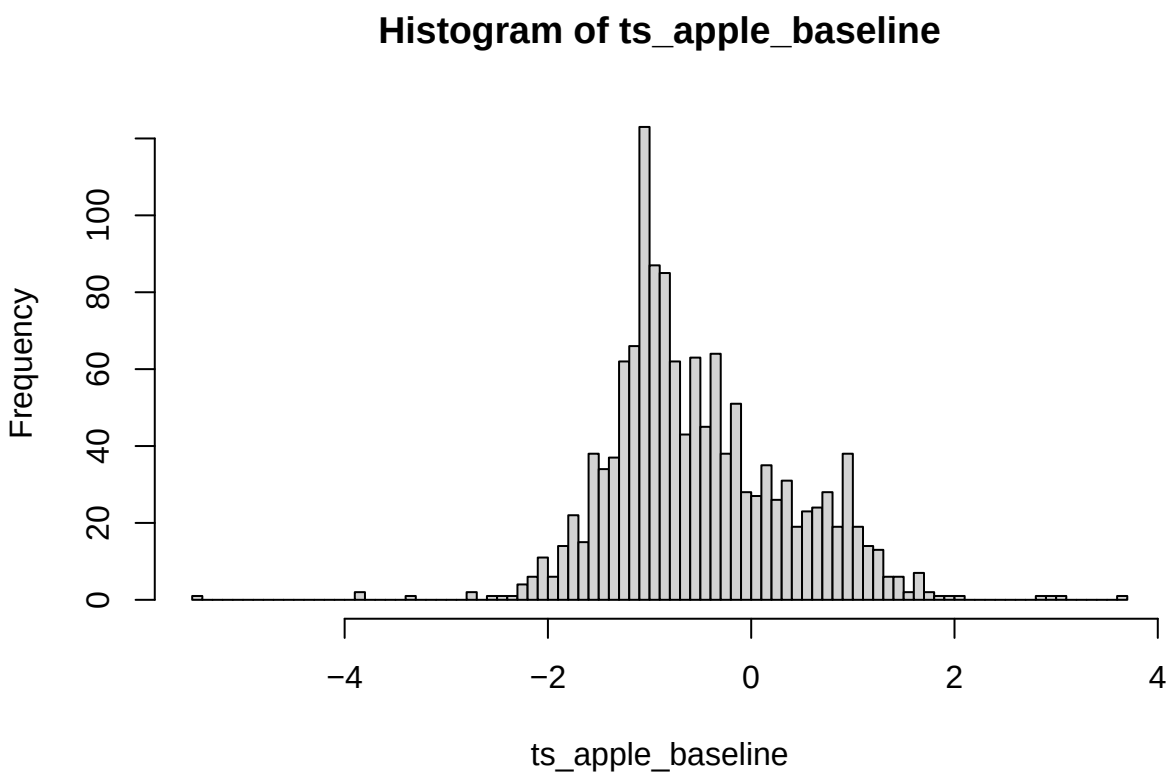


No se ve una agrupación o clustering clara por grupos, más bien se solapan. Esto indica que los perfiles globales de metabolitos de estos grupos son bastante similares, al menos en las dimensiones capturadas por PC1 y PC2.

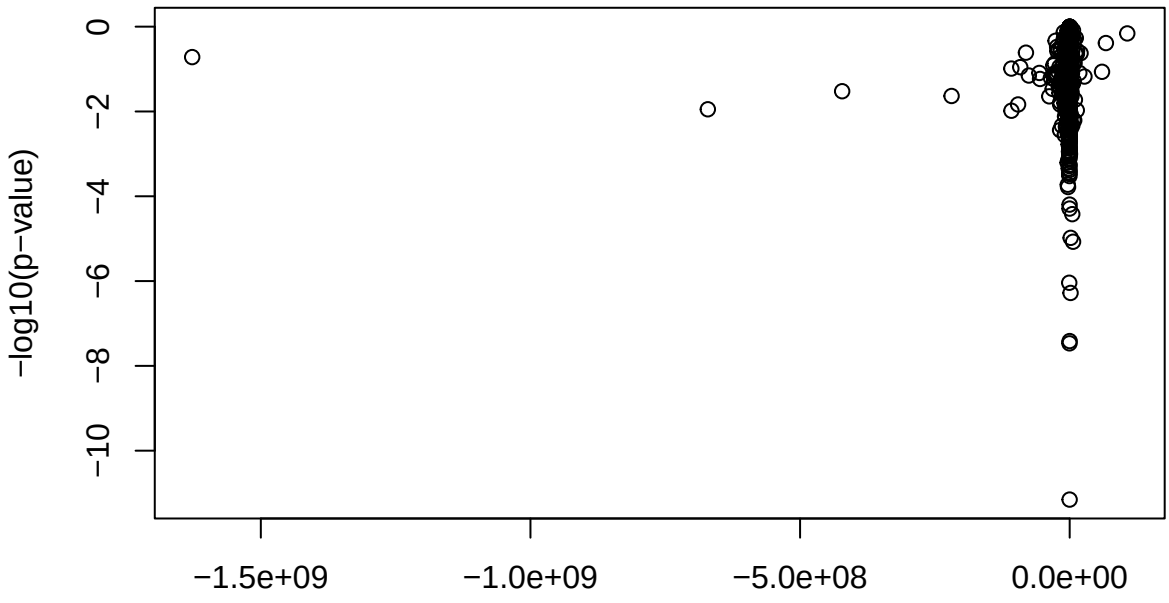
Ahora realizamos el t-test, lo realizaremos por grupos de dos a dos.

```
#https://aspteaching.github.io/Analisis_de_datos_omicos-Ejemplo_0-Microarrays/ExploreArrays.html
group_baseline <- which(treatment == "Baseline")
group_apple <- which(treatment == "Apple")
group_cranberry <- which(treatment == "Cranberry")

# Creamos nuestra función para realizar el t-test entre dos grupos
ttest <- function(x, group1, group2) {
```

```r
  tt <- t.test(x[group1], x[group2])
  return(c(tt$statistic, tt$p.value, tt$estimate[1] - tt$estimate[2]))
}
# Probamos nuestra función ttest con el grupo baseline y apple.
ans_apple_baseline <- apply(assay_data, 1, function(x) ttest(x, group_apple, group_baseline))
ts_apple_baseline <- ans_apple_baseline[1, ]
pvals_apple_baseline <- ans_apple_baseline[2, ]
fc_apple_baseline <- ans_apple_baseline[3, ]

# Creamos un histograma de los estadísticos t
hist(ts_apple_baseline, breaks=100)
```



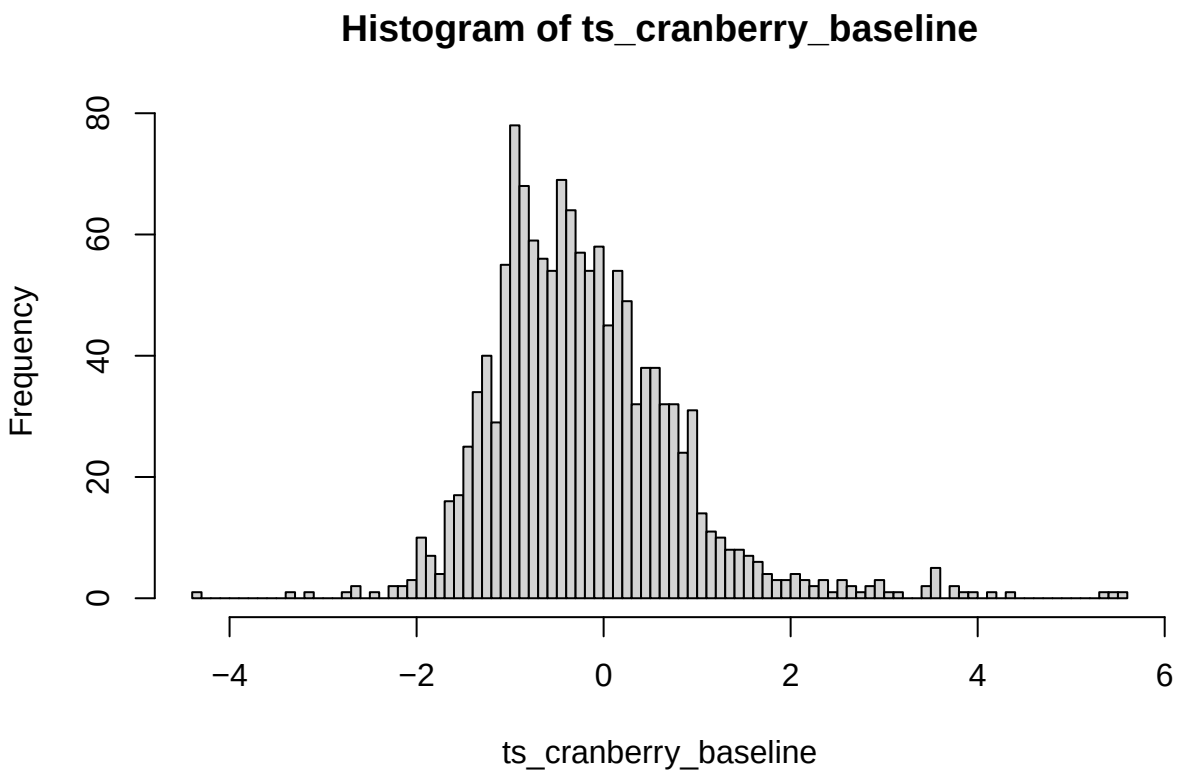

```r
#https://www.htgmolecular.com/blog/2022-08-25/understanding-volcano-plots
# Realizamos el volcano plot
plot(fc_apple_baseline, log(pvals_apple_baseline),
     xlab = "Fold Change (Apple vs Baseline)",
     ylab = "-log10(p-value)")
```



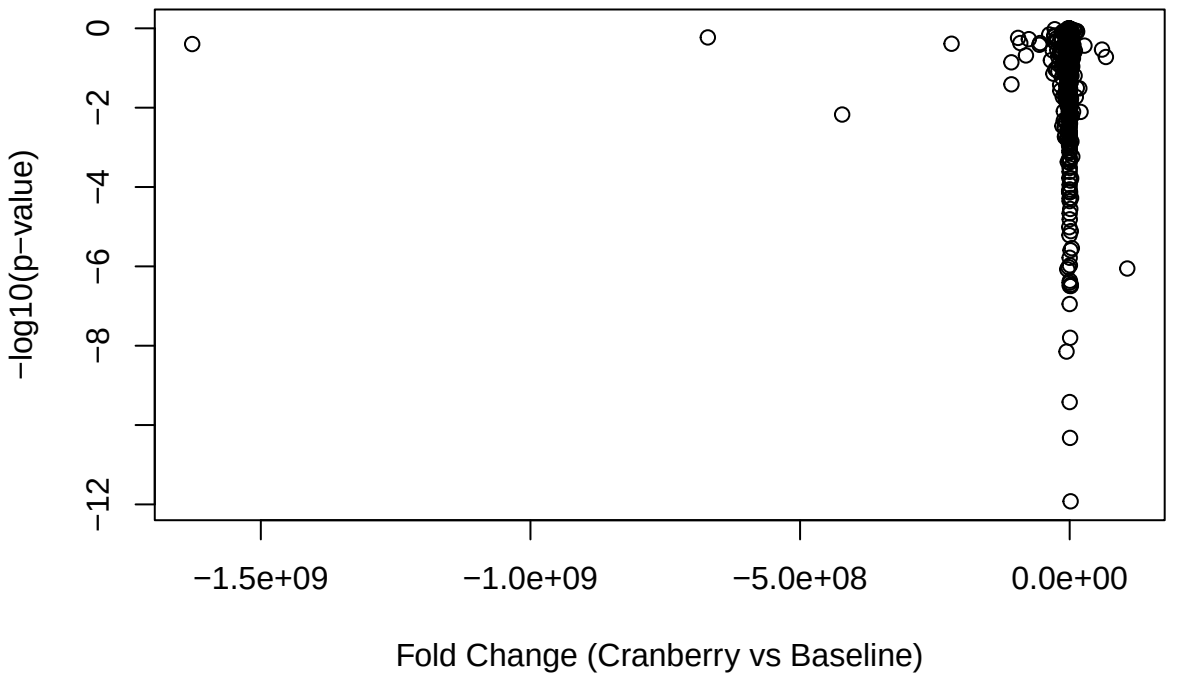

```
pval_thresholds <- c(0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.0000001)
# Filtramos los índices de los metabolitos con los pvalores que buscamos
significant_indices <- which(pvals_apple_baseline < max(pval_thresholds))

# Obtenemos los nombres de los metabolitos y sus p-valores
significant_metabolite_names <- rowData(se)$names[significant_indices]
significant_pvalues <- pvals_apple_baseline[significant_indices]


significant_metabolites_df <- data.frame(
  Metabolite = significant_metabolite_names,
  P_Value = significant_pvalues
)
significant_metabolites_df
```

```
##                          Metabolite      P_Value
## 79034    12-Hydroxydodecanoic acid_1 1.433276e-05
## 10413       4-Hydroxybutanoic acid_1 6.267410e-03
## 20975673             Butanoic acid_1 5.706803e-04
## 8117              Diethylene glycol_1 2.382545e-03
## 7768              epsilon-Caprolactam 6.868329e-03
## 5275508           Methyl farnesoate_1 6.020164e-04
## 439230               (R)-Mevalonate_1 1.873338e-03
```

```
#Probamos nuestra función ttest con el grupo baseline y cranberry

ans_cranberry_baseline <- apply(assay_data, 1, function(x) ttest(x, group_cranberry, group_baseline))
ts_cranberry_baseline <- ans_cranberry_baseline[1, ]
pvals_cranberry_baseline <- ans_cranberry_baseline[2, ]
fc_cranberry_baseline <- ans_cranberry_baseline[3, ]
hist(ts_cranberry_baseline, breaks=100)
```




```
plot(fc_apple_baseline, log(pvals_cranberry_baseline),
     xlab = "Fold Change (Cranberry vs Baseline)",
     ylab = "-log10(p-value)")
```

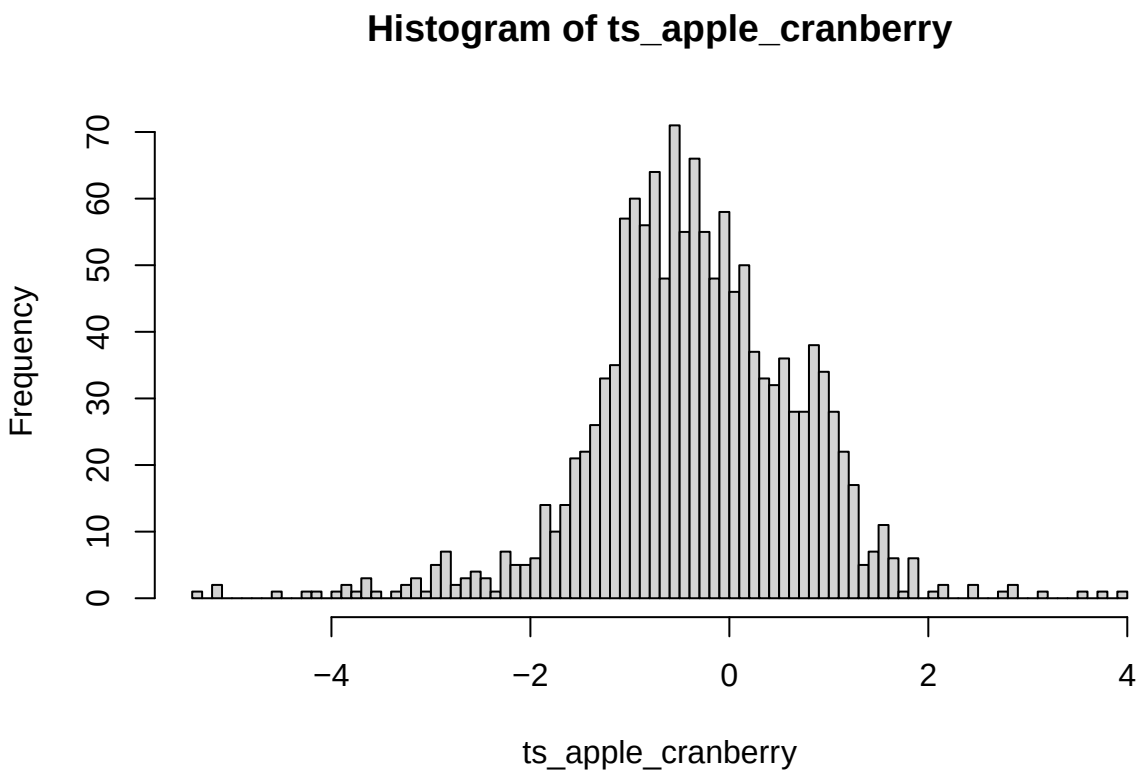



```
#Creamos el dataframe con los metabolitos con p-valores significativos en cranberry vs baseline
significant_indices <- which(pvals_cranberry_baseline < max(pval_thresholds))
```

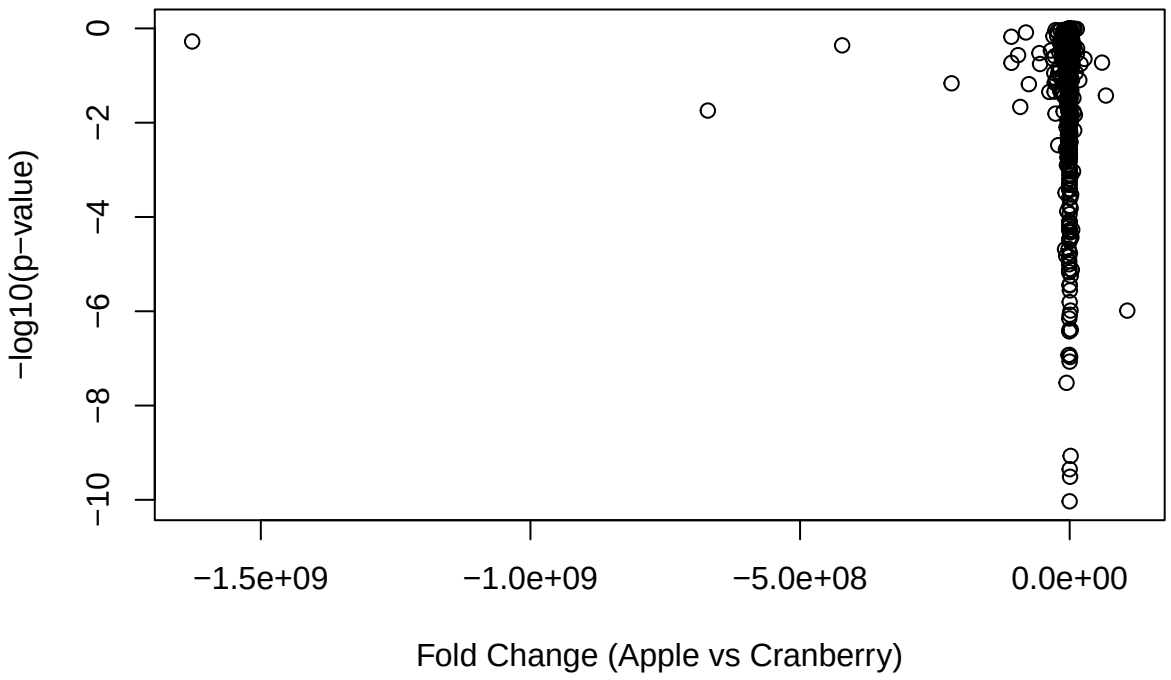
```r
significant_metabolite_names <- rowData(se)$names[significant_indices]
significant_pvalues <- pvals_cranberry_baseline[significant_indices]


significant_metabolites_df <- data.frame(
  Metabolite = significant_metabolite_names,
  P_Value = significant_pvalues
)
significant_metabolites_df
```

```
##                                Metabolite      P_Value
## 13891         2-Chloro-1,4-naphthoquinone 2.431351e-03
## 443071   5-Methyl-3-isoxazolyl sulfate_1 1.509003e-03
## 17531                     Azinphos-ethyl_1 2.545646e-03
## 3035199       Decarbamoylgonyautoxin 1_1 6.022137e-03
## 439361                 D-Prephenyllactate 1.642070e-03
## 5378303                       Furamizole 8.117020e-03
## 21718008                    Heterocladol 3.716477e-03
## 71485                         Sucralose 8.105095e-05
## 5353                         Sulmazole_1 3.078430e-03
## 33676             4-Ketocyclophosphamide 1.726471e-03
## 241                             Benzene_1 9.428600e-03
## 5486800                      Coenzyme B 2.894539e-04
## 54678503               Dihydroxyfumarate 9.589946e-04
## 464                            Hippurate 2.348997e-03
## 92433                      Imazosulfuron 3.283761e-05
## 11046097                       Indanofan 6.643423e-06
## 798                               Indole 1.562000e-03
## 94214          METHYL BETA-D-GALACTOSIDE 4.096395e-04
## 153367        N-(Acetyloxy)benzenamine_1 6.657315e-03
## 441306                         Netilmicin 5.429174e-03
## 5281740                      Otonecine_1 3.921592e-03
## 996                                Phenol 1.518156e-03
## 5462519                Yersiniabactin_1 2.305192e-03
```

```r
#Probamos nuestra función ttest con el grupo apple y cranberry.
ans_apple_cranberry <- apply(assay_data, 1, function(x) ttest(x, group_apple, group_cranberry))
ts_apple_cranberry <- ans_apple_cranberry[1, ]
pvals_apple_cranberry <- ans_apple_cranberry[2, ]
fc_apple_cranberry <- ans_apple_cranberry[3, ]
hist(ts_apple_cranberry, breaks=100)
```




```r
plot(fc_apple_baseline, log(pvals_apple_cranberry),
     xlab = "Fold Change (Apple vs Cranberry)",
     ylab = "-log10(p-value)")
```




```r
#Creamos el dataframe con los metabolitos con p-valores significativos en cranberry vs apple
significant_indices <- which(pvals_apple_cranberry < max(pval_thresholds))
```

```r
significant_metabolite_names <- rowData(se)$names[significant_indices]
significant_pvalues <- pvals_apple_cranberry[significant_indices]

significant_metabolites_df <- data.frame(
  Metabolite = significant_metabolite_names,
  P_Value = significant_pvalues
)
significant_metabolites_df
```

```
##                                                Metabolite      P_Value
## 13891                          2-Chloro-1,4-naphthoquinone 9.768249e-04
## 443071                   5-Methyl-3-isoxazolyl sulfate_1 5.284047e-03
## 17531                                  Azinphos-ethyl_1 3.858173e-03
## 439361                               D-Prephenyllactate 1.686511e-03
## 4488                                    Niflumic acid_1 5.679420e-03
## 71485                                         Sucralose 8.693355e-05
## 5353                                       Sulmazole_1 3.011004e-03
## 442988                                      Theogallin 9.223453e-03
## 1131                              Thiamin monophosphate 7.918614e-03
## 79034                          12-Hydroxydodecanoic acid_1 4.400110e-05
## 33676                               4-Ketocyclophosphamide 8.507170e-03
## 11915                       alpha-Oxo-benzeneacetic acid_1 8.030251e-03
## 20975673                                 Butanoic acid_1 6.079522e-03
## 5486800                                       Coenzyme B 5.430069e-04
## 8117                                Diethylene glycol_1 7.172531e-03
## 54678503                              Dihydroxyfumarate 8.510128e-04
## 464                                           Hippurate 2.512805e-03
## 780                                    Homogentisate_1 1.616567e-03
## 92433                                    Imazosulfuron 7.400934e-05
## 11046097                                     Indanofan 1.153005e-04
## 798                                             Indole 1.665301e-03
## 91619    L-2-Amino-4-(hydroxymethylphosphinyl)butanoate 6.708992e-03
## 94214                           METHYL BETA-D-GALACTOSIDE 9.391452e-04
## 5275508                               Methyl farnesoate_1 4.290305e-03
## 153367                         N-(Acetyloxy)benzenamine_1 2.124698e-03
## 441306                                       Netilmicin 9.823143e-04
## 5281740                                    Otonecine_1 5.974238e-03
## 996                                             Phenol 2.302839e-03
## 439230                                 (R)-Mevalonate_1 2.516357e-03
## 11236                        Semicarbazide hydrochloride 4.347890e-03
## 107849                                 Versicolorin B_1 9.331367e-03
```

Con los histogramas de los estadísticos t y los volcano plots obtenidos a partir de los t-tests, podemos concluir que apenas hay metabolitos con diferencias significativas entre grupos, sin embargo, hemos localizado aquellos con un p-valor significativo.

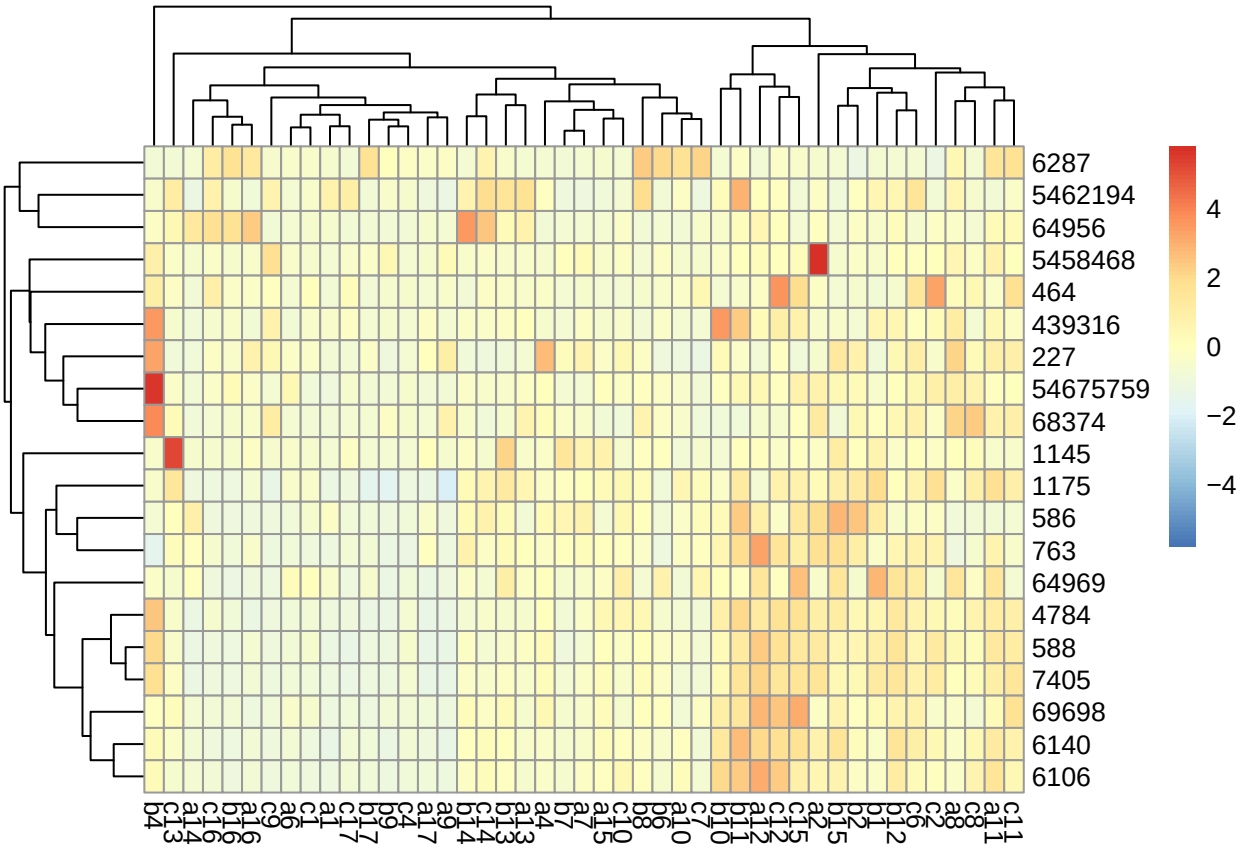
Ahora vamos a realizar una agrupación jerárquica de las muestras y metabolitos. Ya que hay más de 1500 metabolitos, voy a seleccionar una pequeña muestra de los 20 metabolitos con más variabilidad, para que sea más sencilla su visualización. Así nos enfocamos en los metabolitos más informativos.

```r
# Calculamos la varianza para cada metabolito

metabolite_variances <- apply(assay_data, 1, var)
```

```r
# Seleccionamos los 20 metabolitos con mayor varianza
top20_var <- names(sort(metabolite_variances, decreasing = TRUE))[1:20]

# Filtramos el objeto se original para obtener solo los metabolitos seleccionados
se20 <- se[top20_var, ]
rowData(se20)
```

```
## DataFrame with 20 rows and 3 columns
##                          names     PubChem        KEGG
##                    <character> <character> <character>
## 588               CREATININE_1         588      C00791
## 464                  Hippurate         464      C01586
## 5462194 (E)-4-(Trimethylammo..     5462194      C04114
## 586                   CREATINE         586      C00300
## 4784    Phenylmethanesulfony..        4784      C06747
## ...                         ...         ...         ...
## 7405           5-OXO-L-PROLINE        7405      C01879
## 5458468 (2S,3S)-2-Hydroxytri..     5458468      C04655
## 763           GUANIDINOACETATE         763      C00581
## 6140         L-Phenylalanine_1        6140      C00079
## 6106                    LEUCINE        6106      C00123
```

```r
# Creamos el heatmap con agrupación jerárquica

library(pheatmap)

assay_data_top20 <- assay(se20)

pheatmap(
  assay_data_top20,
  scale = "row",
  clustering_distance_rows = "euclidean",
  clustering_distance_cols = "euclidean",
  clustering_method = "complete",
  show_rownames = TRUE,
  show_colnames = TRUE
)
```



Los tonos rojos representan niveles más altos de expresión, mientras que tonos azules representan niveles más bajos. Los tonos amarillos indican niveles intermedios. Podemos observar los gráficos de agrupación jerárquica.

Link del repositorio de github: https://github.com/carmenanavarro/Navarro-Andres-Carmen-PEC1