Análisis exploratorio de datos ómicos

Cristina Juárez Alía

2024-11-06

Contents

1 Resumen

2 Objetivos del estudio

El objetivo principal de este trabajo consiste en la realización de un análisis exploratorio de datos metabolómicos desde un objeto SummarizedExperiment, poniendo así en práctica los conocimientos adquiridos durante el R1 de la asignatura de Análisis de Datos Ómicos.

Para el dataset escogido, procedente del artículo 'H-NMR urinary metabolomic profiling for diagnosis of gastric cancer', nos proponemos los siguientes objetivos:

- **1. Evaluar la calidad de los datos:**
  - **Comprobar si las mediciones son estables a través de los datos QC.**
  - **Determinar si existe 'Efecto batch'.**
- **2. Búsqueda de agrupaciones de las muestras según condición:**
  - **A través de PCA, incluyendo la búsqueda de metabolitos que expliquen la variabilidad observada entre grupos.**
  - **Utilizando diferentes métodos de clustering.**

3 Instalación de paquetes

A lo largo del desarrollo del trabajo se requieren los siguientes paquetes:

- **Biobase**
- **ggplot2**
- **scatterplot3d**
- **factoextra**

La ejecución de este bloque de código instala estos paquetes si aún no están instalados:

```
if (!requireNamespace("Biobase", quietly = TRUE)) {
  BiocManager::install("Biobase")
}

if (!requireNamespace("ggplot2", quietly = TRUE)) {
  install.packages("ggplot2")
}

if (!requireNamespace("scatterplot3d", quietly = TRUE)) {
  install.packages("scatterplot3d")
```

```
}

if (!requireNamespace("factoextra", quietly = TRUE)) {
  install.packages("factoextra")
}
```

```
library(SummarizedExperiment)
library(dplyr)
library(ggplot2)
library(knitr)
library(scatterplot3d)
library(factoextra)
```

4 Carga de datos

Los datos empleados se almacenan en un objeto de tipo SummarizedExperiment, que se encuentra descargado como archivo .Rda en el repositorio de este trabajo (consultar sección 'Repositorio en Github').

Dicho archivo se ha creado manualmente a partir de la tabla de datos original, en formato .xlsx, que se encuentra en el repositorio proporcionado en el enunciado de la actividad. Todos los pasos para el diseño del SummarizedExperiment y su descarga en .Rda se encuentran documentados en el apéndice de este informe.

Para cargar los datos en RStudio ejecutamos el siguiente comando, que supondrá la aparición de un nuevo objeto SummarizedExperiment en nuestro entorno de trabajo denominado se.

```
load("SummarizedExperiment_GCdata.Rda")
```

4.1 Estructura del SummarizedExperiment

5 Preprocesamiento de datos

5.1 Eliminación de metabolitos con medidas de baja calidad

El dataset cuenta originalmente con 149 metabolitos y 140 muestras.

Eliminamos del dataset aquellos metabolitos con un valor QC-RSD < 25%, tal y como se indica en el resumen del estudio original (esta información puede consultarse en el archivo Metadata.Rmd del trabajo); y que, además, el porcentaje de valores faltantes sea inferior al 20%, de acuerdo con la *regla del 80%* (1), y basándonos en el tutorial publicado sobre este dataset en Python por el *Centre for Metabolomics and Computational Biology* (2).

Esto se consigue filtrando sobre rowData(se):

```
# El se es sutituido por el subset del se filtrado.
se <-se[(rowData(se)$QC_RSD < 25) & (rowData(se)$Perc_missing < 20), ]
```

Como podemos comprobar, la matriz de datos queda con 72 metabolitos:

```
dim(se)
```

```
## [1]  72 140
```

5.2 Tratamiento de NA: imputación por el valor medio

La presencia de NA en el dataset puede interferir o incluso impedir la aplicación de ciertas funciones en R (como en el Análisis de Componentes Principales).

Dado que el estudio no asegura la posible causa de estos valores NA (no detectado, detección por debajo del umbral, error de medición, etc.), decidimos **'neutralizar' dichos valores por el promedio del metabolito correspondiente, y el grupo específico al que pertenece la muestra**. De esta forma evitamos distorsionar los promedios originales de cada grupo, controlando así la integridad de los datos originales.

Para ello, diseño una función imputar_medias que utiliza como argumento el SummarizedExperiment se. Esta función busca los NA en la matriz de datos, la muestra a la que pertenecen, y el metabolito. En base a esto, calcula el promedio de ese metabolito en el grupo correspondiente, y sustituye el NA por dicho valor. La función devuelve el objeto con dichas modificaciones.

El código de diseño de la función imputar_medias no se presenta en esta sección para facilitar la lectura, pero se puede consultar en el Apéndice.

Sustituimos se por el resultado de la función:

```
se <- imputar_medias(se)
```

Podemos comprobar que la matriz de datos ya no contiene valores NA:

```
anyNA(assay(se)) # Comprobamos que ya no existen NA
```

```
## [1] FALSE
```

6 Análisis de calidad

6.1 Evaluación de la estabilidad de las medidas

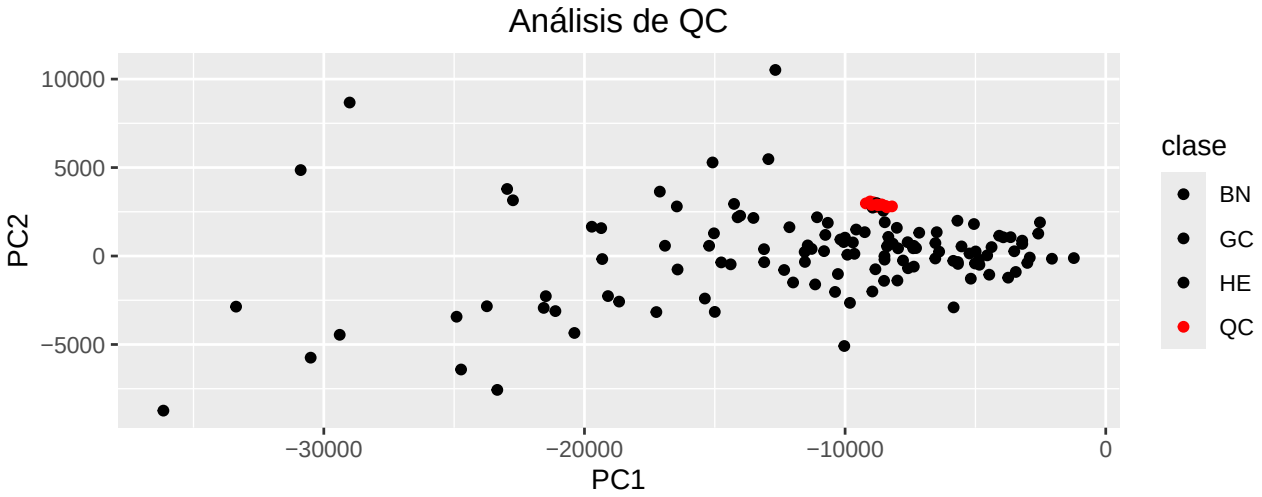
Las muestras QC (Quality Control) en los ensayos metabolómicos permiten evaluar la calidad del mismo, en términos de estabilidad. En un Análisis de Componentes Principales, **lo ideal es que estas muestras estén bien agrupadas (2).**

Para diseñar el gráfico utilizamos ggplot.

El código empleado para visualizar el gráfico se muestra a continuación:

```
# Representación de los datos con ggplot

# PCA con prcomp
# Utilizamos t() para organizar las muestras en filas y metabolitos en columnas
pca1 <- prcomp(t(assay(se)), center = FALSE, scale. = FALSE)

# Organizamos el dataframe con los datos de interés para representar
# Seleccionamos los dos primeros componentes principales desde la matriz de rotación $x
datos_pca1 <- as.data.frame(pca1$x[, 1:2]) %>%
  cbind(clase = colData(se)$clase) # Distinguiremos las muestras por clases

# Elaboramos el gráfico mediante ggplot
pca_2d <- ggplot(datos_pca1) +
```

```
  aes(PC1, PC2, color = clase) + # Los puntos se distinguirán por clase (grupo)
  geom_point(size = 1.5) +
  scale_color_manual(values = c('black', 'black', 'black', "red")) + # Los QC color rojo
  ggtitle("Análisis de QC") +
  theme(plot.title = element_text(hjust = 0.5))

pca_2d
```



Podemos concluir que existe estabilidad en las mediciones.

Dado que el propósito de las muestras QC es realizar esta comprobación. Una vez concluida, **creamos un nuevo SummarizedExperiment eliminando las muestras QC de la matriz.**

El código empleado para creación del nuevo se se muestra en el Apéndice.

Como resultado de la eliminación de los QC, se adquiere nuevas dimensiones. Trabajaremos con 123 muestras:

```
dim(se)
```

```
## [1]  72 123
```

7 Escalado de los datos.

Escalamos los datos de los metabolitos restando por la media y dividiendo por la desviación estándar específica del grupo al que pertenezca el dato.

Para ello diseñamos una función denominada scale_by_group que utiliza como argumentos el SummarizedExperiment y el nombre de grupo o clase del cual se desea extraer la matriz de datos escalada.

El diseño de la función scale_by_group no se presenta en esta sección, pero se puede encontrar en el Apéndice.

Ejecutamos dicha función para cada uno de los grupos:

```
# Utilizamos lapply para utilizar como argumento 'clase' cada una de las denominaciones
# de cada grupo (BN, HE, GC, QC)
```

```r
# El resultado ('results') es una lista de las matrices resultantes de la función
results <- lapply(c("BN","HE","GC"), function(clase) {
  scale_by_group(se, clase)
})

# Cada uno de los elementos de 'results' se pasa como argumento a cbind().
# Así conseguimos unir todas las matrices que contienen los datos normalizados
assay_scaled <- do.call(cbind, results)


# Sustituimos el assay(se) original por 'assay_scale'
# Primero reorganizamos el assay escalado
assay_scaled <- assay_scaled[, match(colnames(assay(se)), colnames(assay_scaled)), drop = FALSE]
assay(se) = assay_scaled
```

8 Análisis exploratorio multivariante.

8.1 Análisis de componentes principales

El PCA a continuación tiene varios propósitos:

- Analizar el **efecto batch**
- Analizar la **separación entre grupos** de muestra según variabilidad
- Revelar cuáles son los **metabolitos más influyentes** en la variabilidad observada

Preparamos los datos para el PCA con los comandos explicados a continuación:

```r
# PCA con prcomp
# Utilizamos t() para organizar las muestras en filas
# En este caso especificamos center = FALSE y scale.=FALSE porque ya hemos escalado los datos
# a conveniencia anteriormente
pca2 <- prcomp(t(assay(se)), center = FALSE, scale. = FALSE)


# Preparamos el dataframe con los datos a representar
# Seleccionamos los 3 primeros PC
datos_pca2 <- as.data.frame(pca2$x[, 1:3]) %>%
  cbind(clase = colData(se)$clase, batch = colData(se)$batch) # Añadimos los datos clase y batch
```

Las varianzas explicadas por cada PC se muestran en la siguiente tabla (**el código utilizado para realizar este cálculo se encuentra en el Apéndice**).

Table 1: Varianzas Explicadas por los Componentes Principales

PC	Varianza explicada
PC1	36.05 %
PC2	6.01 %
PC3	4.82 %

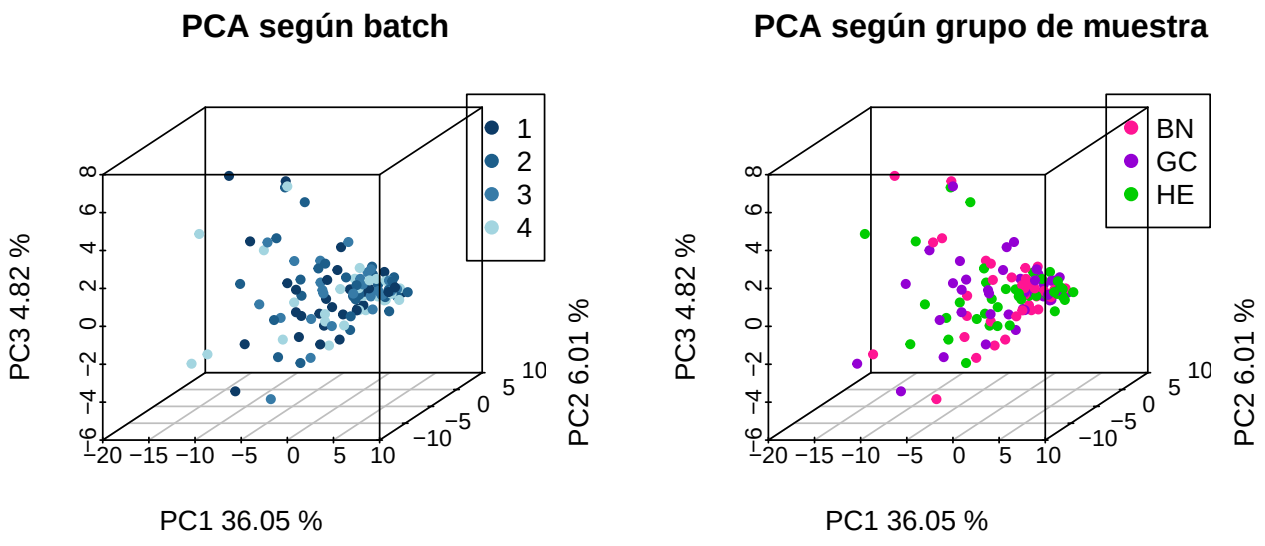
PC	Varianza explicada
Total	46.88 %

A continuación se representa el PCA en 3D, el cual recoge hasta un 46.88% de la variabilidad observada de los datos.

En el gráfico de la izquierda, las muestras se distinguen según batch (1, 2, 3 o 4); mientras que en el gráfico de la derecha, según el grupo al que pertenecen (BN, GC o HE).

La función plot_pca empleada para elaborar estas gráficas se ha diseñado específicamente para crear un gráfico 3D utilizando scatterplot3d (3) con las tres primeras PC, especificando un factor mediante el cual se desea distinguir las muestras (en este caso 'batch' y 'clase'). **El diseño de esta función se muestra en el Apéndice.**

```
par(mfrow = c(1,2))

# PCA según batch
colores2 <- c("#0D3B66", "#1D5E8A", "#377BA7", "#A3D5E0") # Colores para distinguir por batch

plot_pca(df=datos_pca2, colores=colores2, factor=datos_pca2$batch,
      titulo=paste("PCA según batch"))

# PCA según muestras
colores1 <- c('deeppink', 'darkviolet', 'green3') # Colores para distinguir por grupo

plot_pca(df=datos_pca2, colores=colores1, factor=datos_pca2$clase,
      titulo=paste("PCA según grupo de muestra"))
```



Podemos extraer las siguientes conclusiones:

- **No existe efecto batch:** las muestras de cada batch se distribuyen aleatoriamente. No se percibe ninguna agrupación que pueda explicarse por este factor. Este es otro aspecto que **influye positivamente en la calidad del experimento**.

- **No se perciben agrupaciones que distingan entre distintos tipos de muestras**. El análisis demuestra que las muestras no se pueden distinguir entre sí según la variabilidad observada en sus metabolitos.

8.1.1 Metabolitos más influyentes en la variabilidad observada

Es común examinar las variables (en este caso, metabolitos) que mayor peso tienen en la variabilidad observada entre muestras. Esto es especialmente interesante en caso de que se observen agrupaciones asociadas a cada grupo, pues estariamos extrayendo aquellos metabolitos que permiten hacer una distinción entre diferentes grupos, y que, por tanto, podrían ser buenos candidatos como biomarcadores de la condición.

A pesar de que en el gráfico anterior no podemos ver dicha distinción, realizaremos este paso para comprobar los 15 metabolitos más influyentes en el PC1.

Para ello ejecutamos el siguiente código:

```
# Ordenamos los valores de los coeficientes asociados a cada metabolito en el PC1,
# de mayor a menor (utilizamos abs() porque no distinguimos entre valores - o +)
metabo_PC1 <- sort(abs(pca2$rotation[,1]), decreasing = TRUE)

# Extraemos los 15 primeros
metabo_PC1_r <- metabo_PC1[1:15]
nombres <- rowData(se)[names(metabo_PC1_r), ]$Label
```

Resumimos los resultados en la siguiente tabla:

Table 2: Metabolitos más influyentes en PC1

	Metabolito	Coeficiente
M88	N-Acetylglutamine	0.1764818
M65	Glycylproline	0.1737720
M48	Creatinine	0.1729453
M104	Proline	0.1717931
M74	Isoleucine	0.1681942
M90	N-Acetylornithine	0.1676557
M107	Pyroglutamate	0.1630389
M149	τ-Methylhistidine	0.1584275
M25	6-Hydroxynicotinate	0.1569756
M5	2-Aminoadipate	0.1549299
M37	Azelate	0.1539151
M77	Lactulose	0.1526421
M122	Valine	0.1524309
M73	Indole-3-lactate	0.1511153
M62	Glutamine	0.1474721

8.1.1.1 Boxplot multivariante

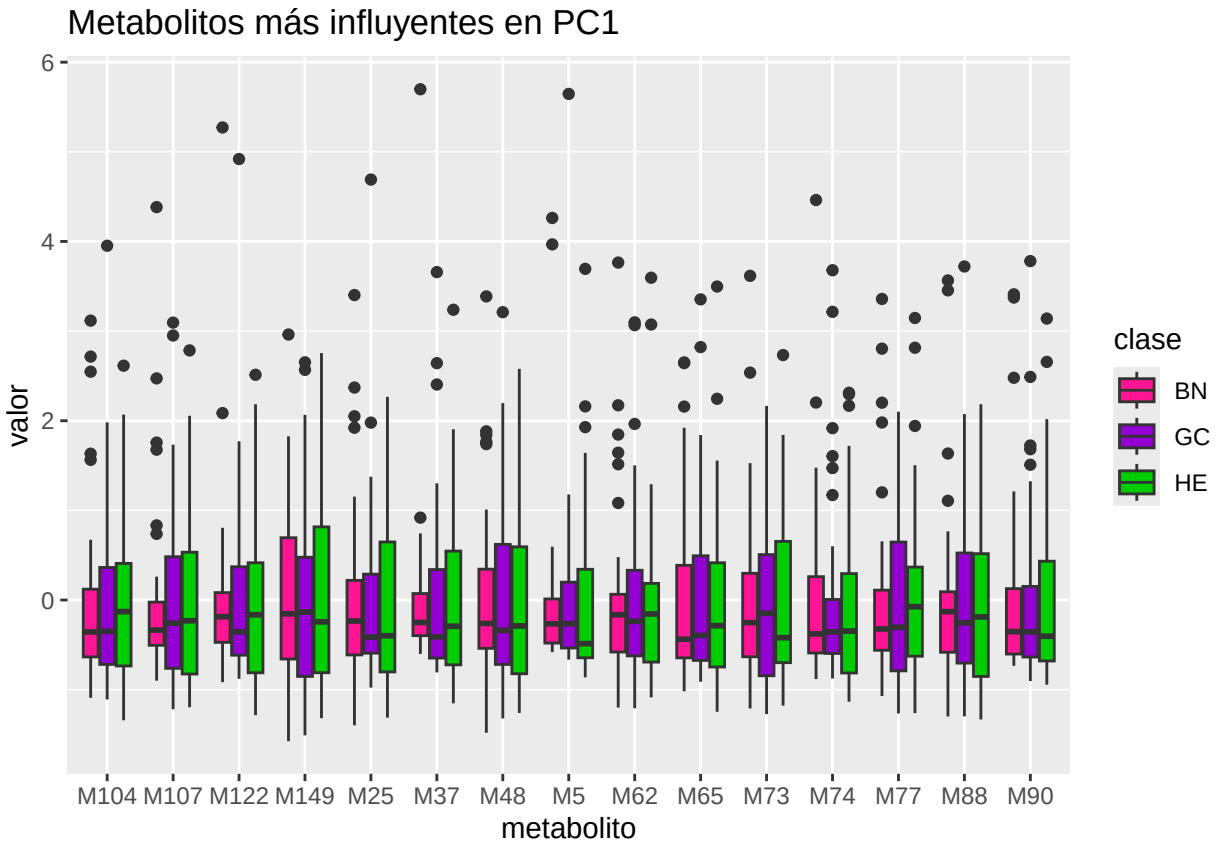
Elaboramos un boxplot multivariante (4) con los datos de los metabolitos seleccionados anteriormente, aquellos que más peso tienen en el PC1.

Aunque en el PCA o se observa distinción entre grupos, este gráfico nos puede ayudar a desvelar si existen diferencias en la distribución de estos metabolitos para los distintos grupos que podrían no capturarse en

el PCA.

El código para la preparación de los datos para la elaboración de este boxplot no se muestran en esta sección, pero se encuentra en el Apéndice.

```
# df_boxplot es un dataframe preparado con todos los datos necesarios para realizar el gráfico
ggplot(df_boxplot, aes(x=metabolito, y=valor, fill=clase)) +
 geom_boxplot() +
 scale_fill_manual(values = c("BN" = "deeppink", "GC" = "darkviolet", "HE" = "green3"))+
 ggtitle("Metabolitos más influyentes en PC1")
```



A través del gráfico no podemos observar diferencias relevantes entre los distintos grupos para ninguno de los metabolitos.

8.2 Análisis de Clustering

En esta sección tratamos de organizar los datos de las muestras de los grupos 'GC' (cáncer gástrico) y 'HE' (sanos) en clusteres, según su condición.

Para ello probaremos algunos de los métodos de clustering más populares.

Creamos una nueva matriz seleccionando los datos correspondientes.

```
# Seleccionamos solo dos grupos (por simplificar el análisis)

GC <- rownames(colData(se)[colData(se)$clase == "GC", ])
HE <- rownames(colData(se)[colData(se)$clase == "HE", ])
```

```
# Creamos la nueva matriz con los datos seleccionados

matrix = assay(se)[names(metabo_PC1),c(HE,GC)]
```

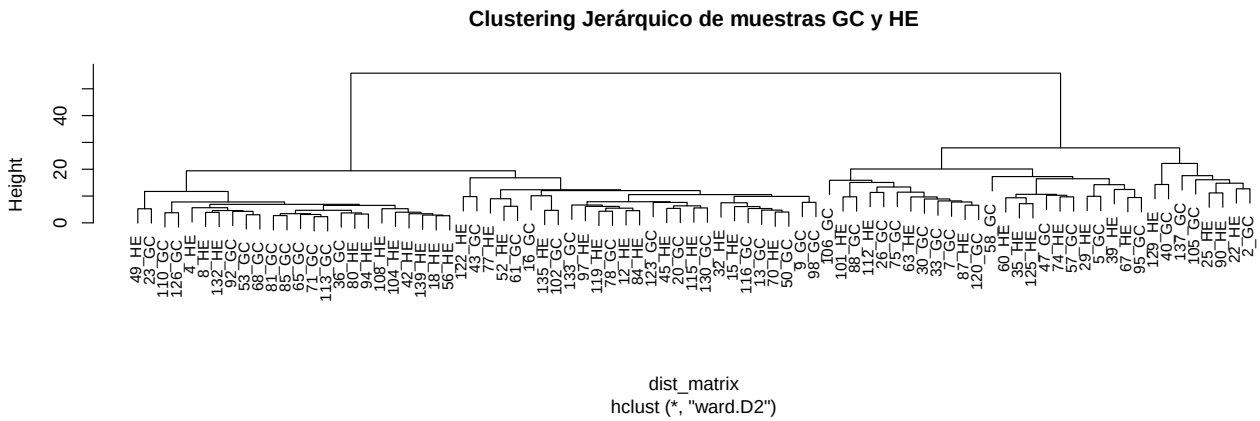
8.2.1 Análisis jerárquico: dendograma

Elaboramos un análisis jerárquico y lo representamos en forma de dendograma.

Utilizamos el método de cálculo de distancias euclídeas, y como método de clustering 'ward.D2', uno de los métodos más populares, que minimiza la varianza interna, ayudando a crear clústeres más compactos (5).

El nombre de las muestras se ha simplificado eliminando el término "sample_" para facilitar la lectura del gráfico. El código elaborado para realizar esta modificación se encuentra en el Apéndice

```
# Matriz de distancias entre muestras.
# Utilizamos método euclídeo
dist_matrix <- dist(t(matrix), method = "euclidean")

# Algoritmo de clustering. Método ward.D2
hc <- hclust(dist_matrix, method = "ward.D2")

plot(hc, main = "Clustering Jerárquico de muestras GC y HE", cex = 0.8)
```

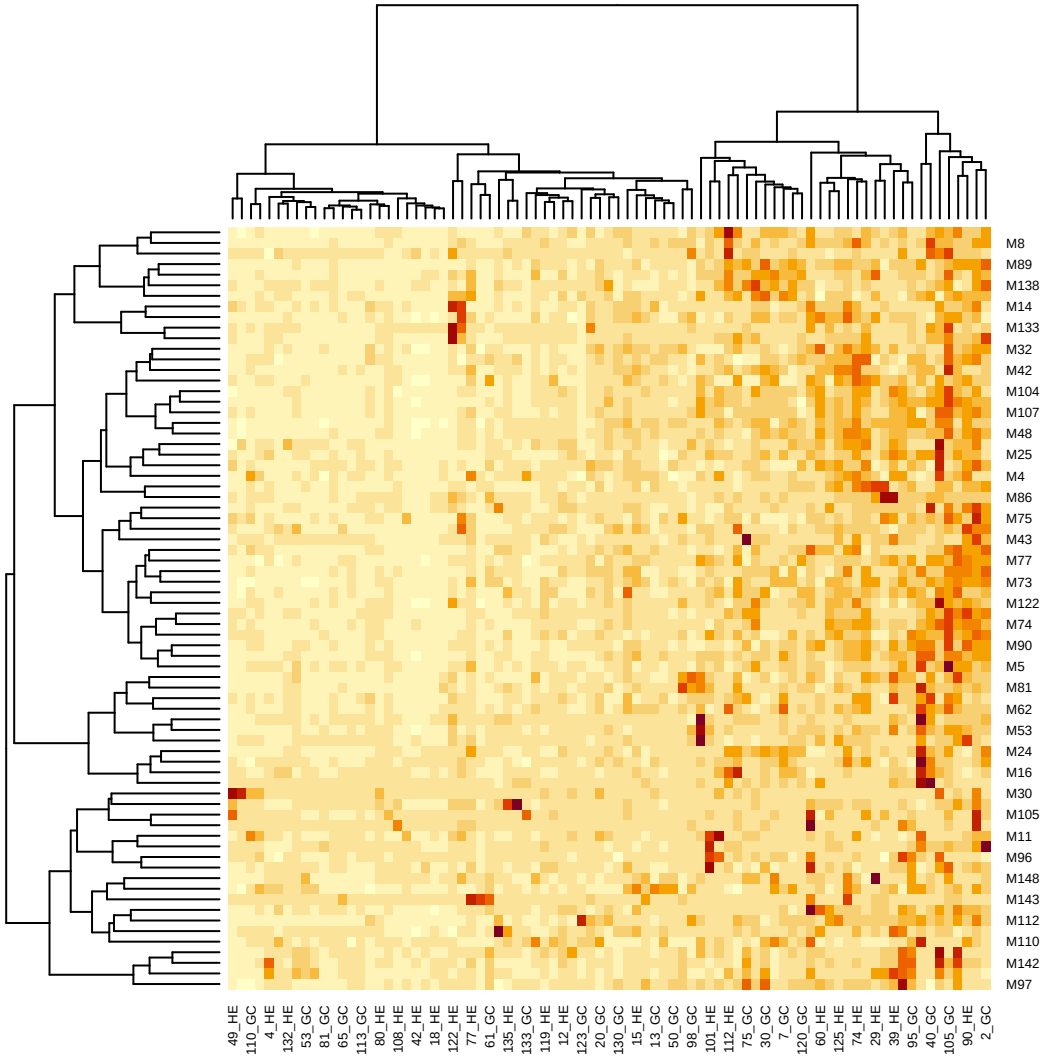


Como se puede comprobar, el clustering jerárquico por los métodos utilizados no separa con éxito ambos grupos de muestras.

8.2.2 Heatmap

Podemos representar el mismo clustering jerárquico incluyendo información sobre los niveles de cada metabolitos mediante un Heatmap (6).

El código para elaborar el Heatmap se oculta en esta sección. Para consultarlo ir al Apéndice

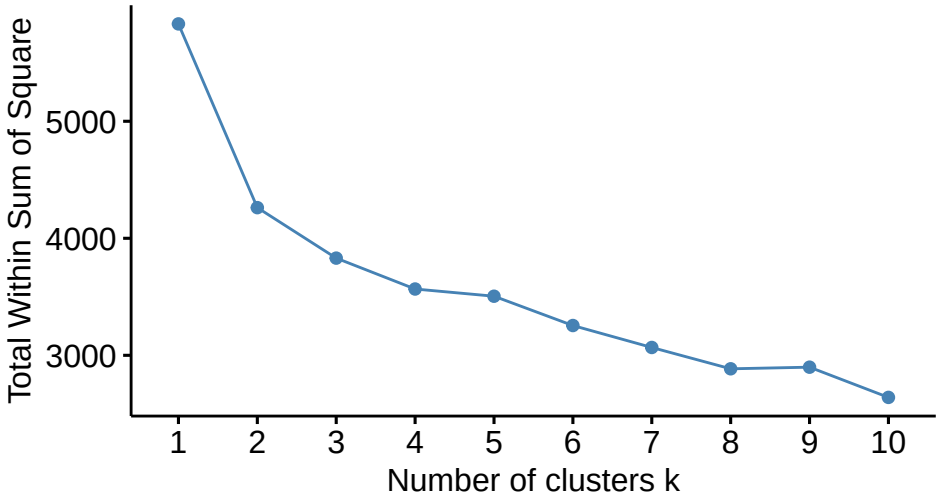


8.2.3 Método de k-means

En primer lugar conviene estimar el número óptimo de clusteres (k) a asignar al algoritmo.

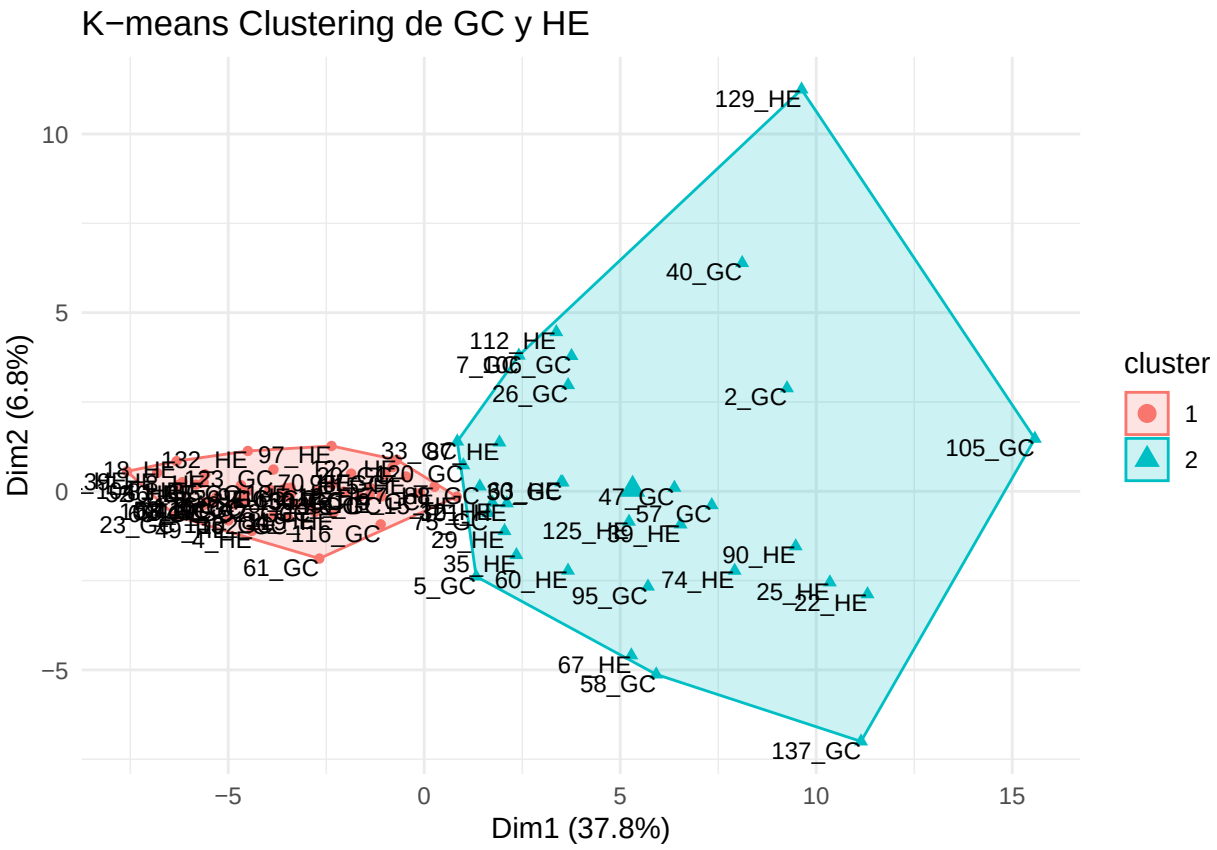
Para ello existen varios métodos. En este trabajo utilizaremos el Método del Codo:

```
fviz_nbclust(t(matrix), kmeans, method = "wss") +
  labs(title = "")
```



El gráfico del Método del Codo no muestra un punto de inflexión claro. No obstante, procedemos con el análisis ajustando k = 2, según lo esperado. El resultado se presenta a continuación:

El código empleado para diseñar el gráfico se encuentra en el Apéndice.



Como podemos comprobar, el método de K-Means tampoco proporciona resultados favorables para clusterizar según lo esperado.

9 Conclusiones

10 Enlace a Github

11 Apéndice

11.1 Creación del SummarizedExperiment

```
# Carga de datos desde el .xlsx

library(readxl)
# Datos de assay, en la pestaña "Data"
GCdata <- as.matrix(read_excel("GastricCancer_NMR.xlsx", sheet = "Data"))

# Info de metabolitos, en la pestaña "Peak"
metabolitos <- as.data.frame(read_excel("GastricCancer_NMR.xlsx", sheet = "Peak"))

library(readr)
# El archivo Muestras.txt es un extracto del archivo ST001047_AN001711.txt original
# del estudio, y está disponible en el repositorio
muestras <- as.data.frame(read_table("Muestras.txt", col_names = FALSE))

# Preparación de tablas:

### rowData (muestras)

nombre_muestras <- GCdata[,2] # Extraemos el nombre de las muestras
clase <- as.factor(GCdata[,"Class"]) # Extraemos a qué clase pertenecen
batch <- as.factor(as.numeric(substr(muestras[,6],7,7))) # Extraemos el batch al que pertenecen
# substr() se utiliza para simplificar el valor en el que originalmente se muestra el batch
# ("Batch_1" cambia a 1, etc.)

muestras <- data.frame(clase,batch) # dataframe con la clase y el batch
rownames(muestras) = nombre_muestras # Asignamos el nombre de las muestras a las filas

### colData (metabolitos)

nombre_metabolitos <- metabolitos[,"Name"] # Extraemos el nombre de todos los metabolitos
rownames(metabolitos) = nombre_metabolitos # Asignamos dichos nombres a las filas

metabolitos = metabolitos[,-c(1,2)] # Ajuste para eliminar columnas no necesarias


### assay (datos experimentales)

GCdata <- GCdata[,5:153] %>% t() # Omitimos las columnas que no necesitamos
# Y transponemos la matriz para que las filas sean metabolitos y las columnas muestras
```

```r
colnames(GCdata) = nombre_muestras # Asignamos nombre a columnas
rownames(GCdata) = nombre_metabolitos # Y a filas

GCdata <- apply(GCdata,2,as.numeric)

### metadata

metadatos <- list(study_ID = "ST001047",
                  institute = "University of Alberta",
                  name = "David Broadhurst",
                  email = "d.broadhurst@ecu.edu.au")


# Creación de SummarizedExperiment:

library("SummarizedExperiment")

se <- SummarizedExperiment(assays = list(data = GCdata),
                           colData = muestras,
                           rowData = metabolitos,
                           metadata = metadatos)

### Guardo como .Rda al objeto SummarizedExperiment
save(se, file = "SummarizedExperiment_GCdata.Rda")
```

11.2 Diseño de la función imputar_medias

```r
imputar_medias <- function(se){
 # Recogemos los nombres de las muestras correspondientes a cada grupo
 BN <- rownames(colData(se)[colData(se)$clase == "BN", ])
 HE <- rownames(colData(se)[colData(se)$clase == "HE", ])
 GC <- rownames(colData(se)[colData(se)$clase == "GC", ])
 QC <- rownames(colData(se)[colData(se)$clase == "QC", ])

 # Identificamos las posiciones en las que se encuentran los NA en la matriz
 # 'indices' guarda una matriz con el nombre del metabolito, número de fila,
 # y número de columna.
 indices <- which(is.na(assay(se)), arr.ind = TRUE)

 #'metabolitos' guarda el nombre de los metabolitos con NA
 metabolitos <- rownames(indices)

 # 'ind_col' guarda las columnas donde se encuentran los NA
 ind_col <- indices[,"col"]
 # Dichos números se traducen en el nombre de las muestras donde se encuentran
 # los NA
```

```
  muestras <- colnames(assay(se))[ind_col]

  # Se recorren los vectores de muestras y metabolitos a la vez
  for (i in seq_along(metabolitos)) {

    muestra <- muestras[i]
    metabolito <- metabolitos[i]

    # Se determina a qué grupo (clase) pertenece la muestra
    clase_muestra <- colData(se)[muestra,]$clase

    if (clase_muestra == "BN"){
      media = mean(assay(se)[metabolito,BN], na.rm = TRUE) # Media de los BN
    } else if (clase_muestra == "HE"){
      media = mean(assay(se)[metabolito,HE], na.rm = TRUE) # Media de los HE
    } else if (clase_muestra == "GC"){
      media = mean(assay(se)[metabolito,GC], na.rm = TRUE) # Media de los GC
    } else if (clase_muestra == "QC"){
      media = mean(assay(se)[metabolito,QC], na.rm = TRUE) # Media de los QC
    } else if (clase_muestra == "QC"){
    }

    # El valor NA original se sustituye por la media calculada según el grupo
    # al que pertenece la muestra, directamente en la matriz de datos
    assay(se)[metabolito, muestra] = media
  }
  return(se) # La función devuelve el SummarizedExperiment modificado
}
```

11.3 Eliminación de los QC

```
# QC guarda los nombres de las muestras QC
QC <- rownames(colData(se)[colData(se)$clase == "QC", ])

# Diseño de nuevo assay: se seleccionan todas las columnas cuyo nombre no esté incluido en QC
nueva_matrix <- assay(se)[, !(colnames(assay(se)) %in% QC), drop = FALSE]

# Diseño de nuevo colData(): hacemos misma operación pero transformamos a dataframe
nuevo_colData <- as.data.frame(colData(se)[!(colnames(assay(se)) %in% QC), , drop = FALSE])

# Eliminamos los levels de 'clase' sin valores (nivel QC, porque ya hemos eliminado sus valores)
nuevo_colData$clase <- droplevels(nuevo_colData$clase)

# Creación delnuevo SummarisedExperiment
se <- SummarizedExperiment(
  assays = SimpleList(counts = nueva_matrix),
  colData = nuevo_colData,
```

```
  rowData = rowData(se)
)

rm(QC, nueva_matrix, nuevo_colData)
```

11.4 Diseño de la función scale_by_group

```
scale_by_group <- function(se, clase) {

  # Guardamos los nombres de las muestras de cada grupo
  BN <- rownames(colData(se)[colData(se)$clase == "BN", ])
  HE <- rownames(colData(se)[colData(se)$clase == "HE", ])
  GC <- rownames(colData(se)[colData(se)$clase == "GC", ])

  # Dependiendo de la 'clase' especificada en el argumento, 'muestras' adquiere los
  # correspondientes nombres
  if (clase == "BN") {
    muestras = BN
  } else if (clase == "HE") {
    muestras = HE
  } else if (clase == "GC") {
    muestras = GC
  }

  # Calcular medias de cada uno de los metabolitos para las muestras del grupo ('clase') concreto
  medias = apply(assay(se)[, muestras], 1, mean, na.rm = TRUE)
  sds = apply(assay(se)[, muestras], 1, sd, na.rm = TRUE)

  # Matriz con datos escalados.
  # Operación vectorial: a los elementos de cada una de las filas se les resta la media
  # que corresponde (específica del grupo o 'clase')
  m <- ((assay(se)[,muestras] - medias)/sds)

  return(m) # La función devuelve la matriz de datos escalados, sólo para el grupo especificado
}
```

11.5 Cálculo de varianzas explicadas

```
# Calculamos las varianzas explicadas por cada componente principal
var_expl1 <- round(((pca2$sdev[1])^2/sum((pca2$sdev)^2)*100),2)
var_expl2 <- round(((pca2$sdev[2])^2/sum((pca2$sdev)^2)*100),2)
var_expl3 <- round(((pca2$sdev[3])^2/sum((pca2$sdev)^2)*100),2)
var_expl_total <- sum(var_expl1,var_expl2,var_expl3)
```

11.6 Diseño de la función plot_pca

```
plot_pca <- function(df, colores, factor, titulo) {
 # Crear el gráfico 3D
 scatterplot3d(df$PC1, df$PC2, df$PC3,
           color = colores[as.numeric(factor)],
           pch = 20,
           xlab = paste("PC1", var_expl1, "%"),
           ylab = paste("PC2", var_expl2, "%"),
           zlab = paste("PC3", var_expl3, "%"))

 # Añadir la leyenda
 legend("topright",
      legend = levels(factor),
      col = colores[1:length(levels(factor))],
      pch = 19,
      inset = c(0.001, 0.001))

 # Añadir el título
 title(main = titulo)
}
```

11.7 Preparación de datos para boxplot multivariante

```
valores_met <- function(se,metabolitos,clase){

 BN <- rownames(colData(se)[colData(se)$clase == "BN", ])
 HE <- rownames(colData(se)[colData(se)$clase == "HE", ])
 GC <- rownames(colData(se)[colData(se)$clase == "GC", ])

 if (clase == "BN"){
  muestras = BN
 } else if (clase == "HE"){
  muestras = HE
 } else if (clase == "GC"){
  muestras = GC
 }

 df = data.frame()
 for (metabolito in metabolitos){
  datos = data.frame(assay(se)[metabolito, muestras])
  datos = cbind(datos,rep(clase,length(muestras)), rep(metabolito,length(muestras)))
  df = rbind(df, datos)
 }
 colnames(df) = c("valor", "clase", "metabolito")
 return(df)
}
```

```r
# Aplica `valores_met()` para cada clase
seleccion <- lapply(c("BN", "HE", "GC"), function(clase)
 {valores_met(se = se, metabolitos = names(metabo_PC1_r), clase = clase)})

# Combina los resultados en un solo dataframe,lo utilizaremos para el boxplot
df_boxplot <- do.call(rbind, seleccion)
```

11.8 Simplificación nombre de muestras

```r
# Vamos a añadir al nombre de cada muestra la terminación GC o HE
# en función del grupo al que pertenezca.
# Esto facilita la interpretación de los próximos gráficos.

nombres_columnas <- colnames(matrix)
clases_muestras <- colData(se)[nombres_columnas, ]$clase
nuevos_nombres <- paste0(nombres_columnas, "_", clases_muestras)
colnames(matrix) = substr(nuevos_nombres, 8, nchar(nuevos_nombres))
```

11.9 Diseño del Heatmap

```r
heatmap(
 matrix,
 Rowv = TRUE, # Reorganiza filas según agrupamiento jerárquico
 Colv = TRUE, # Reorganiza columnas según agrupamiento jerárquico

 distfun = function(x) dist(x, method = "euclidean"), # Método de cálculo de distancias
 hclustfun = function(x) hclust(x, method = "ward.D2"), # Método de clustering
 cexRow = 0.5,  # Tamaño de letra
 cexCol = 0.5
)
```

11.10 Diseño del gráfico K-Means

```r
# Diseñamos el algoritmo k means.
# centers indica k =2
# el número de iteraciones máximas es 1000
km <- kmeans(t(matrix), centers = 2, iter.max = 1000)

fviz_cluster(km, data = t(matrix), # Graficamos los resultados del algoritmo
          geom = "point",
          stand = FALSE, # No estandarizamos los datos antes de graficar
          ellipse.type = "convex",
          main = "K-means Clustering de GC y HE") +
 geom_text(aes(label = rownames(t(matrix))),
         size = 3,
```

```
        vjust = 1,
        hjust = 1) +
theme_minimal()
```