PEC 1

Joan Canseco i Ribas

2024-10-31

Contents

Abstract

En este proyecto, se ha llevado a cabo un análisis estructurado de datos metabolómicos para simular un proceso de análisis ómico. El objetivo ha sido familiarizarse con la organización y manejo de datos ómicos, aplicando diversas herramientas y métodos que se han ido mostrando durante la asignatura de "Análisis de datos ómicos". Para ello, se ha seleccionado un conjunto de datos de un estudio en el que se investigaron los metabolitos en respuesta a la cirugía bariátrica en 4 distintos timepoints en 39 pacientes con más de 10 años de obesidad. Estos 39 pacientes fueron clasificados en dos grupos, un grupo de pacientes que se les aplicaba cirugía by pass (n=26) y otro que se les aplicaba cirugía tubular (n=13).

Estos datos han sido organizados en un contenedor del tipo SummarizedExperiment para facilitar el trabajo con este dataset. Además, se han realizado análisis exploratorios en el que se han reflejado las características más relevantes del proyecto. También se han estudiado las características de los principales grupos de estudio, y se han comparado las diferencias de metabolitos entre el timepoint 0 y el timepoint 5 entre los dos tipos principales de cirugías para verificar cuáles son los metabolitos que más cambiaron al aplicar una u otra cirugía. En esta última prueba se han detectado 9 metabolitos con diferencias significativas entre cirugías. Finalmente, se ha creado un repositorio del proyecto en Github donde se han subido todos los documentos del proyecto.

Objetivos del estudio

Los objetivos generales de esta actividad son planificar y ejecutar una versión simplificada del proceso de análisis de datos ómicos, utilizando algunas de las herramientas y métodos que se han trabajado en la unidad. En concreto, se deberá:

- Seleccionar un dataset de un repositorio de github de metabolómica https://github.com/nutrimetabolomics/metaboData/.
- Crear un contenedor del tipo *SummarizedExperiment* que contenga los datos y metadatos del dataset.
- Llevar a cabo una exploración del dataset que dé una visión general del mismo.
- Elaborar un informe que describa el proceso realizado (este documento).
- Crear un repositorio en Github y subir el informe, el objeto contenedor con los datos y metadatos, el código R para la exploración de los datos, los datos y los metadatos.

Materiales y Métodos

Esta actividad se ha desarrollado desde un ordenador Windows.

Origen y naturaleza de los datos

Los datos analizados han sido extraídos del siguiente repositorio online: https://github.com/nutrimetabolomics/metaboData/

Una vez accedemos a dicho repositorio, clicamos el botón verde "<> Code", y clicamos la opción "Download ZIP". Una vez se ha descargado el repositorio en nuestro ordenador, clicamos botón derecho y seleccionamos "Extraer todo". Exploramos los distintos datasets y seleccionamos el dataset "2018-MetabotypingPaper" porque tiene los documentos bien estructurados y parece que se puede trabajar bien con él. Cogemos los documentos de dicho dataset y los llevamos a nuestro directorio de trabajo de la PEC1.

Se trata de un dataset de un estudio de 39 pacientes con más de 10 años de obesidad a los que se les ha realizado una cirugía bariátrica. Se clasifica el dataset en 2 grupos según la cirugía bariátrica recibida: grupo "by pass" compuesto por 26 pacientes y grupo "tubular" compuesto por 13 pacientes. También se recoge su estado metabólico, en cuanto a si los pacientes estaban sanos o enfermos metabólicamente en la variable Group. En estos pacientes se pretende estudiar como evolucionan ciertos metabolitos en 4 distintos timepoints tras la cirugía según qué cirugía hayan tomado.

Herramientas informáticas

Se han usado principalmente 3 herramientas informáticas y bioinformáticas para la resolución de la PEC1:

- R y Rstudio: Se ha usado para realizar la exploración del dataset para obtener una visión general del mismo. Además, se ha usado para la redacción del informe, usando Rmarkdown.
- Bioconductor: Aunque es un paquete de R, lo consideraremos como una herramienta independiente para hacer esta explicación. Bioconductor se ha usado para crear un contenedor del tipo *SummarizedExperiment*.
- Github: Se ha usado para la descarga de los datos (ya se ha indicado en la sección previa el procedimiento realizado), y se ha usado para presentar los resultados complementarios de la PEC1.

Creación de un contenedor del tipo SummarizedExperiment

Para la creación del contenedor se ha usado Bioconductor dentro de R (como se ha especificado anteriormente). Los pasos se han especificado en el apartado de Resultados.

Exploración del dataset

Para la exploración de los datos se ha usado principalmente R y Rstudio. Los documentos donde hay los datos han sido:

- "DataValues_S013.csv": Se trata del documento donde hay almacenados los datos experimentales de los pacientes. Es un data frame con 39 filas (de 39 pacientes) y 696 variables. De las 696 variables, las 6 primeras son para identificar a los pacientes (X.1, SUBJECTS, SURGERY, AGE, GENDER y Group, en la que X.1 es un duplicado de SUBJECTS, por lo que realmente se podría considerar solo las 5 primeras columnas), y el resto de variables son los metabolitos tomados en 4 timepoints distintos (T0, T2, T4 y T5). Se denota a qué timepoint hace referencia cada variable porque al final del nombre de cada variable hay especificado el timepoint al que pertenecen con "_TX" (donde X es 0, 2, 4 o 5).
- "DataInfo_S013.csv": Se trata de un documento donde hay la metadata del documento anterior y está organizado de manera distinta. Tiene solo 4 columnas (X, es un duplicado de VarName, VarName, contiene el nombre de cada una de las variables del documento anterior, varTpe, especifica el tipo de variable que es, y Description, especifica qué información se recoge en cada variable) y 695 filas, correspondientes a las 695 variables del documento anterior (recordemos que eran 696 pero la primera columna era un duplicado de la segunda columna, por lo que son 695 efectivas).

Una exploración más exhaustiva se ha llevado a cabo en el apartado de Resultados. En esta exploración se ha realizado un análisis inicial de los datos. Además, se han estudiado las variables identificativas de los pacientes, centrando el foco sobretodo en el tipo de cirugía realizada en el paciente. Finalmente, también se ha estudiado las diferencias entre el T0 y el T5 entre tipo de cirugía, ya que nos interesa ver como han variado los metabolitos tras la cirugía en un grupo y en otro para saber qué efectos diferentes hay en una cirugía y en otra.

Creación del repositorio Github

Primeramente, se ha creado un nuevo repositorio online llamado "Canseco-i-Ribas-Joan-PEC1" desde el usuario "CansecoRibas".

Dentro del directorio local del proyecto (donde hay todos los documentos que se han ido mencionando en la parte de "Materiales y Métodos"), se ha creado primeramente un proyecto de R, para facilitar el trabajo con estos documentos. Posteriormente, y desde la Terminal de Rstudio, se ha creado un repositorio local Git usando el comando "git init". Después se ha establecido conexión con el repositorio online con "git remote add origin https://github.com/CansecoRibas/Canseco-i-Ribas-Joan-PEC1.git". Se han añadido todos los documentos al área de ensayo con "git add ." y se ha hecho el primer commit con "git commit -m 'Primer commit del proyecto' ". Seguidamente, se han enviado todos los documentos del repositorio local al repositorio online con "git push origin master".

Una vez se han vinculados los repositorios, se han ido haciendo commits a medida que se ha ido avanzando en el proyecto.

Se puede acceder al repositorio online desde: https://github.com/CansecoRibas/Canseco-i-Ribas-Joan-PEC1

Resultados

En este apartado se volcarán los resultados del proyecto.

Creación del contenedor del tipo SummarizedExperiment

El primer paso ha sido asegurarnos de que "BiocManager" está instalado:

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

Posteriormente se ha instalado desde Bioconductor el paquete "SummarizedExperiment" que es el paquete específico para crear el contendor.

```
if (!requireNamespace("SummarizedExperiment", quietly = TRUE)) BiocManager::install("SummarizedExperimer
```

Y lo llamamos con library.

```
library("SummarizedExperiment")
```

Ahora cargamos los datos ("DataValues_S013.csv") y los metadatos ("DataInfo_S013.csv") del dataset que están dentro de la carpeta "2018-MetabotypingPaper".

```
data_values <- read.csv("2018-MetabotypingPaper/DataValues_S013.csv")
metadata <- read.csv("2018-MetabotypingPaper/DataInfo_S013.csv")
```

Observamos estos dos archivos. En próximos apartados se ha hecho una mejor exploración de los datos. Dejamos este código en forma de comentarios porque sinó el informe se llena de datos densos.

```
#head(data_values)
#head(metadata)
```

En estos dos documentos hay dos variables iniciales que sobran, ya que son duplicaciones de las columnas adyacentes. Estas columnas son X.1 en "DataValues_S013.csv" y X en "DataInfo_S013.csv", por lo que las vamos a eliminar para que no nos molesten.

```
data_values_mod <- data_values[, 2:696]
metadata_mod <- metadata[, 2:4]
```

Vemos que hay algunas variables identificativas (las 5 primeras) en el documento "DataValues_S013.csv" y el resto son variables relacionadas con las muestras. En el documento "DataInfo_S013.csv" se recoge información de las variables del documento "DataValues_S013.csv". Algunas de las variables son las mismas pero en distintos timepoints por lo que vamos a crear una nueva variable que especifique el timepoint. De esta manera, conseguiremos obtener muestras de manera individual para las variables. Primero nos aseguramos que los paquetes dplyr y tidyr estén instalados.

```
if (!requireNamespace("tidyr", quietly = TRUE)) install.packages("tidyr")
if (!requireNamespace("dplyr", quietly = TRUE)) install.packages("dplyr")
```

Los llamamos con library.

```r
library("tidyr")
library("dplyr")
```

Creamos una nueva variable que se llame Timepoint haciendo pivot_longer para así obtener las muestras de manera individual que es lo que nos interesa.

```r
data_values_samples <- data_values_mod %>%
  # Hacemos pivot_longer para transformar el df.
  pivot_longer(
    cols = 6:695, # Seleccionamos las columnas de metabolitos.
    names_to = c(".value", "Timepoint"), # Extraemos los metabolitos como
    #columnas y establecemos timepoints como nueva columna.
    names_sep = "_T" # Establecemos el separador entre el nombre del metabolito
    # y el timepoint.
  ) %>%
  # Especificamos como queremos que se muestren los timepoints.
  mutate(Timepoint = paste0("T", Timepoint)) %>%
  filter(!Timepoint == "TNA") # Eliminamos las filas que se nos han creado por
# error que están completamente vacías.
```

Ahora modificamos el metadata para adecuarlo a las nuevas variables. Primero añadimos la nueva fila de timepoints creando un nuevo df.

```r
metadata_timepoints <- data.frame(
  VarName = "Timepoint",
  varTpe = "character",
  Description= "Punto temporal de recogida de muestra del paciente (0, 2, 4 o 5)"
)
```

Modificamos las variables del metadata eliminando "_TX" del final y eliminamos variables repetidas.

```r
metadata_metabolites <- metadata_mod %>%
  mutate(VarName = gsub("_T[0-5]$", "", VarName)) %>% # Buscamos y eliminamos el "_TX" del final.
  distinct(VarName, .keep_all = TRUE)  # Quitamos las variables repetidas en Varname.
```

Finalmente unimos los dos dfs con rbind.

```r
# Finalmente unimos los dos dfs con rbind.
metadata_samples <- rbind(metadata_timepoints, metadata_metabolites)
```

Vemos que metadata_samples tiene las mismas filas que columnas tiene data_values_samples que es lo que estábamos buscando.

```r
dim(metadata_samples)
```

```
## [1] 180   3
```

```r
dim(data_values_samples)
```

```
## [1] 156 180
```

Una vez modificados ambos documentos, nos quedan los datos separados por muestras individuales cogidas en diferentes timepoints. Ahora ya podemos crear el objeto "SummarizedExperiment". Vemos también que las 4 columnas posteriores a las 6 columnas identificativas de muestra (5 columnas iniciales + columna Timepoint) también recogen un factor asociado a la muestra, por lo que también sería interesante sacarlas del metabolite_data.

Para ello, seleccionamos las variables experimentales (a partir de la variable 11 hasta el final) y creamos una matriz transpuesta llamada "metabolite_data", ya que las filas deben ser las variables y las columnas deben ser las distintas muestras.

```
metabolite_data <- t(as.matrix(data_values_samples[, 11:ncol(data_values_samples)]))
```

Creamos rowData a partir de la información de la metadata para especificar la información de las variables experimentales.

```
rowData <- metadata_samples[11:nrow(metadata_samples), ]
```

Creamos ColData seleccionando las variables de identificación de los pacientes.

```
colData <- data_values_samples[, 1:10]
```

Creamos el objeto *SummarizedExperiment* con los 3 componentes que hemos mencionado.

```
sum_exp_object <- SummarizedExperiment(
  assays = list(counts = metabolite_data),
  rowData = rowData,
  colData = colData
)
```

Vemos qué pinta tiene.

```
sum_exp_object
```

```
## class: SummarizedExperiment
## dim: 170 156
## metadata(0):
## assays(1): counts
## rownames(170): GLU INS ... X lysoPC.a.C14.0
## rowData names(3): VarName varTpe Description
## colnames: NULL
## colData names(10): SUBJECTS SURGERY ... MEDINF MEDHTA
```

Vemos que es un objeto con 174 filas (variables experimentales de metabolitos) y 156 columnas, que son las muestras (4 para cada paciente, ya que son 4 timepoints). Se puede estudiar un poco más el objeto con las siguientes funciones, pero las dejaremos como comentarios para que no se muestren unos tochos terribles en el informe.

```
#assay(sum_exp_object) # Devuelve la matriz principal de datos.
#rowData(sum_exp_object) # Devuelve un df con información sobre los metabolitos
#(filas de la matriz).
#colData(sum_exp_object) # Devuelve un df con información sobre las muestras
#(columnas de la matriz).
```

Solo queda guardar el objeto tal y como nos pide el enunciado.

```
save(sum_exp_object, file = "sum_exp_object.Rda")
```

Exploración del dataset

Empezamos con el análisis de ambos documentos, "DataValues_S013.csv" y "DataInfo_S013.csv". Como se ha indicado anteriormente, el primero contiene los datos en relación a los metabolitos recogidos en muestras de 4 distintos timepoints de 39 pacientes, y el segundo contiene la explicación de las características de las variables y su información. Por lo que nos fijaremos para hacer los análisis en el primer documento ("DataValues_S013.csv").

Análisis preliminar

Primero se ha hecho una exploración inicial. Se ha cogido el documento data_values_samples, en el que hemos transformado los datos para que las filas sean las muestras recogidas (y no los pacientes como antes). Verificaremos las dimensiones del documento para ver si lo que se cumple es cierto.

```
dim(data_values_samples)
```

```
## [1] 156 180
```

Vemos que es un documento con 156 filas (correspondiente a los 4 timepoints de los 39 pacientes) y 180 columnas (correspondientes a las variables de estudio). Hacemos ahora un resumen estadístico muy básico de las 180 variables para ver qué valores toman y qué tipos de variables tenemos. Es un poco complicado trabajar con este número tan elevado de variables ya que los outputs son enormes, por lo que en este caso tampoco vamos a mostrar el resultado.

```
#summary(data_values_samples)
```

También es importante verificar los missings de los datos. Vamos a mirar cuantos missings hay por cada variable.

```
colSums(is.na(data_values_samples))
```

```
##       SUBJECTS        SURGERY            AGE         GENDER          Group
##              0              0              0              0              0
##      Timepoint          MEDDM         MEDCOL         MEDINF         MEDHTA
##              0             13             13             15             13
##            GLU            INS           HOMA          HBA1C HBA1C.mmol.mol
##             11             14             14             91             91
##           PESO            bmi             CC           CINT            CAD
##             10             10             24             14             14
##            TAD            TAS             TG            COL            LDL
##             25             25             11             11             18
##            HDL           VLDL            PCR            LEP          ADIPO
##             13             49             68             68             71
##            GOT            GPT            GGT          URICO          CREAT
##             12             12             15             16             15
##           UREA         HIERRO         TRANSF           FERR            Ile
```

```
##             15             13             25             16             18
##            Leu            Val            Ala            Pro            Gly
##             18             18             18             18             18
##            Ser            Trp            Phe            Met            Orn
##             18             18             18             18             18
##            Arg            His            Asn            Asp            Glu
##             18             18             18             18             18
##            Gln            Cit            Tyr            Thr            Lys
##             18             18             18             18             18
##     Creatinine     Kynurenine     Putrescine      Sarcosine      Serotonin
##             18             18             21             18             18
##        Taurine           SDMA             C0             C2          C3.OH
##             18             18             18             18             18
##   C6..C4.1.DC.  C5.DC..C6.OH.          C7.DC             C8            C10
##             18             18             18             18             18
##          C10.1          C10.2          C14.1          C14.2          C16.1
##             18             18             18             18             18
##          C16.2       C16.2.OH          C18.1       C18.1.OH          C18.2
##             18             18             18             18             18
## lysoPC.a.C16.0 lysoPC.a.C16.1 lysoPC.a.C17.0 lysoPC.a.C18.0 lysoPC.a.C18.1
##             18             18             18             18             18
## lysoPC.a.C18.2 lysoPC.a.C20.3 lysoPC.a.C20.4 lysoPC.a.C24.0 lysoPC.a.C26.0
##             18             18             18             18             18
## lysoPC.a.C26.1 lysoPC.a.C28.0 lysoPC.a.C28.1    PC.aa.C24.0    PC.aa.C28.1
##             18             18             18             18             18
##    PC.aa.C30.0    PC.aa.C32.0    PC.aa.C32.1    PC.aa.C32.3    PC.aa.C34.1
##             18             18             18             18             18
##    PC.aa.C34.2    PC.aa.C34.3    PC.aa.C34.4    PC.aa.C36.0    PC.aa.C36.1
##             18             18             18             18             18
##    PC.aa.C36.2    PC.aa.C36.3    PC.aa.C36.4    PC.aa.C36.5    PC.aa.C38.0
##             18             18             18             18             18
##    PC.aa.C38.1    PC.aa.C38.3    PC.aa.C38.4    PC.aa.C38.5    PC.aa.C38.6
##             18             18             18             18             18
##    PC.aa.C40.1    PC.aa.C40.2    PC.aa.C40.3    PC.aa.C40.4    PC.aa.C40.5
##             18             18             18             18             18
##    PC.aa.C40.6    PC.aa.C42.0    PC.aa.C42.1    PC.aa.C42.2    PC.aa.C42.4
##             18             18             18             18             18
##    PC.aa.C42.5    PC.aa.C42.6    PC.ae.C30.0    PC.ae.C32.1    PC.ae.C32.2
##             18             18             18             18             18
##    PC.ae.C34.0    PC.ae.C34.1    PC.ae.C34.2    PC.ae.C34.3    PC.ae.C36.0
##             18             18             18             18             18
##    PC.ae.C36.1    PC.ae.C36.2    PC.ae.C36.3    PC.ae.C36.4    PC.ae.C36.5
##             18             18             18             18             18
##    PC.ae.C38.0    PC.ae.C38.2    PC.ae.C38.3    PC.ae.C38.4    PC.ae.C38.5
##             18             18             18             18             18
##    PC.ae.C38.6    PC.ae.C40.1    PC.ae.C40.2    PC.ae.C40.3    PC.ae.C40.4
##             18             18             18             18             18
##    PC.ae.C40.5    PC.ae.C40.6    PC.ae.C42.1    PC.ae.C42.2    PC.ae.C42.3
##             18             18             18             18             18
##    PC.ae.C42.4    PC.ae.C42.5    PC.ae.C44.3    PC.ae.C44.4    PC.ae.C44.5
##             18             18             18             18             18
##    PC.ae.C44.6  SM..OH..C14.1  SM..OH..C16.1  SM..OH..C22.1  SM..OH..C22.2
##             18             18             18             18             18
##  SM..OH..C24.1       SM.C16.0       SM.C16.1       SM.C18.0       SM.C18.1
```

```
##             18             18             18             18             18
##       SM.C20.2       SM.C24.0       SM.C24.1              X lysoPC.a.C14.0
##             18             18             18            156            118
```

Vemos que las variables identificativas están completas, mientras que todas las variables de metabolitos tienen algún missing, aunque en general son pocos (la mayoría tienen menos de 19 missings de 156 valores totales). Además, hay una variable "X" que son todo missings, por lo que la podemos eliminar.

```
data_values_samples <- data_values_samples[, -which(names(data_values_samples) == "X")]
```

Análisis de los pacientes

Como hay muchas variables de metabolitos, haremos una exploración inicial de las variables identificativas de los pacientes, que, como hemos dicho son las 5 primeras variables (excluyendo la variable Timepoint que en este caso no tiene mucho sentido mantenerla porque hace referencia a las muestras).

```
data_values_patients <- data_values_samples %>%
  select("SUBJECTS", "SURGERY", "AGE", "GENDER", "Group") %>% # Seleccionamos columnas
  distinct() # Eliminamos duplicados de pacientes
```

Verificamos que tenemos 39 pacientes en este df.

```
nrow(data_values_patients)
```

```
## [1] 39
```

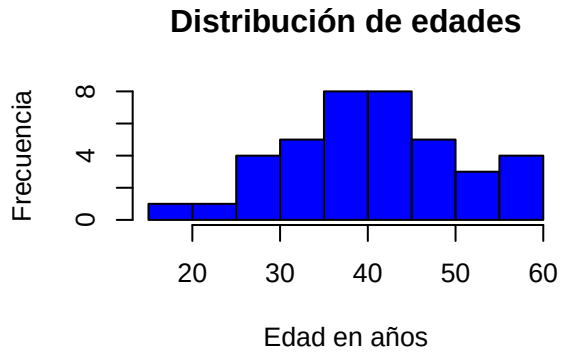
Verificamos las variables que deban ser factors realmente sean factors.

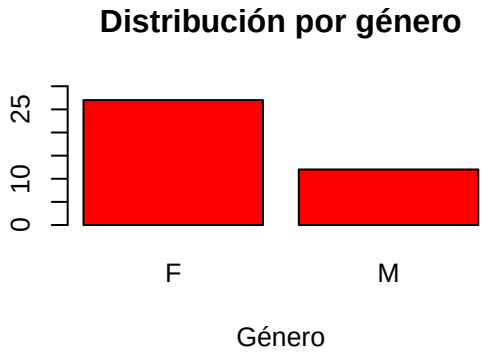
```
data_values_patients$SURGERY <- factor(data_values_patients$SURGERY)
data_values_patients$GENDER <- factor(data_values_patients$GENDER)
data_values_patients$Group <- factor(data_values_patients$Group)
```

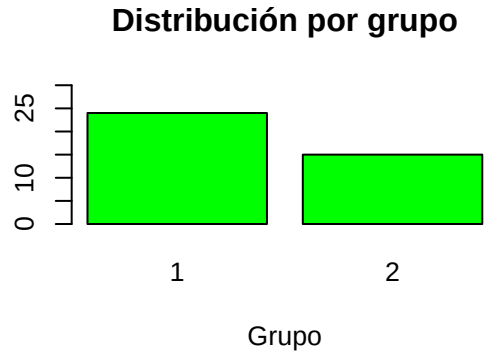
Ahora podemos hacer estadística básica con estos datos de los pacientes. Los datos realmente importantes de este df se encuentran en las variables "SURGERY", "AGE", "GENDER" y "Group", ya que "SUBJECTS" solo hace referencia al número de paciente. La variable "AGE" la veremos con un histograma mientras que los factores los veremos con diagramas de barras.

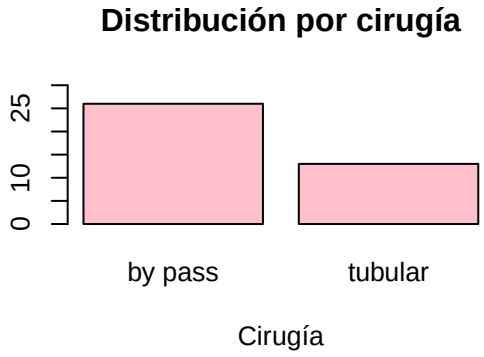
```
par(mfrow = c(2,2)) # Así podemos ver los 4 gráficos a la vez.
# Histograma de la edad.
hist(data_values_patients$AGE,
     main="Distribución de edades",
     col="blue",
     xlab="Edad en años",
     ylab="Frecuencia")

# Diagrama de barras para el género.
barplot(table(data_values_patients$GENDER),
        main="Distribución por género",
        col="red",
        xlab="Género",
```

```
        ylim=c(0,30))

# Diagrama de barras para el grupo.
barplot(table(data_values_patients$Group),
        main="Distribución por grupo",
        col="green",
        xlab="Grupo",
        ylim=c(0,30))

# Diagrama de barras para la cirugía.
barplot(table(data_values_patients$SURGERY),
        main="Distribución por cirugía",
        col="pink",
        xlab="Cirugía",
        ylim=c(0,30))
```





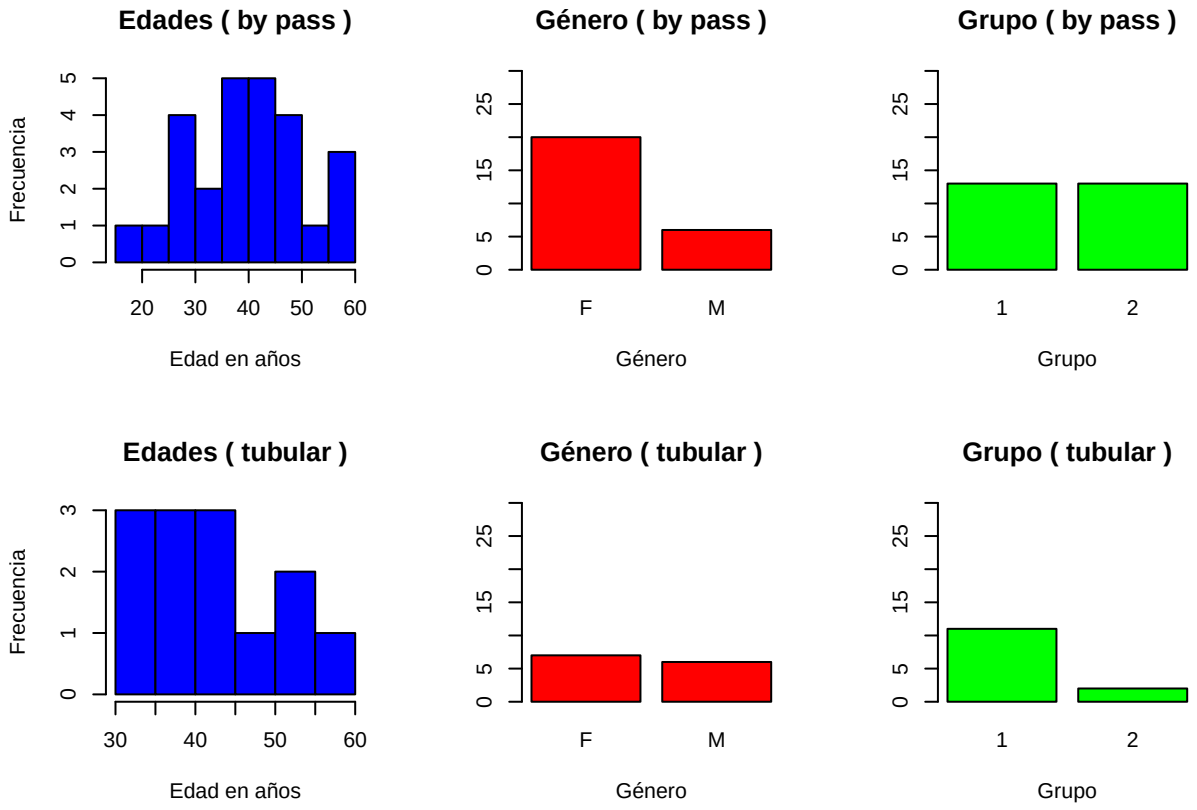




Estos gráficos muestran que la mayoría de edades de los participantes están entre los 30 y los 50 años, que hay el doble de mujeres que de hombres, que hay más componentes del grupo 1 que del grupo 2 y que hay más pacientes con cirugía by pass que tubular. Ahora miraremos la relación entre la variable SURGERY (la variable que indica el tipo de cirugía que ha tomado cada paciente) y las demás, ya que la variable SURGERY es la variable clave del estudio.

```
# Separamos los datos según la variable SURGERY.
split_data <- split(data_values_patients, data_values_patients$SURGERY)

# Redistribuimos el espacio gráfico.
```

```r
par(mfrow = c(2, 3))

# Hacemos loop.
for (surgery_status in names(split_data)) {
  data_subset <- split_data[[surgery_status]]

  # Histograma de la edad.
  hist(data_subset$AGE,
       main=paste("Edades (", surgery_status, ")"),
       col="blue",
       xlab="Edad en años",
       ylab="Frecuencia")

  # Diagrama de barras para el género.
  barplot(table(data_subset$GENDER),
          main=paste("Género (", surgery_status, ")"),
          col="red",
          xlab="Género",
          ylim=c(0,30))

  # Diagrama de barras para el grupo.
  barplot(table(data_subset$Group),
          main=paste("Grupo (", surgery_status, ")"),
          col="green",
          xlab="Grupo",
          ylim=c(0,30))
}
```



Vemos que los grupos son bastante diferentes entre ellos, también probablemente porque la n es relativamente pequeña, aun que no sabemos si puede afectar a nuestros resultados.

Análisis

Se propone dilucidar qué metabolitos varían más desde el T=0 hasta el T=5 (el último timepoint que hay) en cada una de las cirugías (bypass y tubular). Para ver eso, primero vamos a verificar que las columnas de metabolitos sean numéricas, y sinó, las eliminaremos para este análisis.

```
numeric_cols <- sapply(data_values_samples, is.numeric)
data_values_samples_num <- data_values_samples[, numeric_cols | colnames(data_values_samples) %in% c("SU
```

Parece que todas ellas son numéricas. Nos quedamos solo con las muestras de los timepoints 0 y 5. También eliminamos las columnas “MEDDM”, “MEDCOL”, “MEDINF” y “MEDHTA”, ya que son factores asociados a las muestras, que para este análisis no nos van a interesar.

```
data_values_timepoints <- data_values_samples_num[data_values_samples_num$Timepoint %in% c("T0", "T5"),
```

Creamos un df para almacenar las diferencias de metaboitos entre timepoints.

```
data_values_difference <- data.frame()
```

Calculamos la diferencia entre metabolitos para cada paciente (columnas 7 hasta el final).

```
for (patient in unique(data_values_timepoints$SUBJECTS)) {
  patient_data <- data_values_timepoints[data_values_timepoints$SUBJECTS == patient, ]

  # Verificamos que los pacientes tengan ambos timepoints.
  if (all(c("T0", "T5") %in% patient_data$Timepoint)) {
    tp0 <- patient_data[patient_data$Timepoint == "T0", ]
    tp5 <- patient_data[patient_data$Timepoint == "T5", ]

    # Copiamos tp5 a diff_row.
    diff_row <- tp5
    # Computamos diferencias en metabolitos entre t5 y t0.
    diff_row[, 7:ncol(tp5)] <- tp5[, 7:ncol(tp5)] - tp0[, 7:ncol(tp0)]
    # Establecemos valor de columna Timepoint.
    diff_row$Timepoint <- "Difference between T5 and T0"

    # Unimos dfs.
    data_values_difference <- rbind(data_values_difference, diff_row)
  }
}
```

Creamos vector con los nombres de las columnas de los metabolitos.

```
metabolite_columns <- colnames(data_values_samples_num)[11:ncol(data_values_samples_num)]
```

Cremos un df para almacenar los resultados significativos.

```
significant_results <- data.frame(Metabolite = character(),
                                  Test = character(),
                                  p_value = numeric(),
                                  stringsAsFactors = FALSE)
```

Para cada metabolito comparamos los valores de las diferencias entre el T0 y el T5 entre ambos grupos (by pass y tubular) usando un t-test (en caso de normalidad de los datos) o un test de Wilcoxon (en caso de no normalidad de los datos). Posteriormente calculamos las medias de cada uno de los grupos, y en caso de que el test calculado muestre diferencias significativas del metabolito entre grupos, se reflejará en el df creado como "significant_results".

```
for (metabolite in metabolite_columns) {
  # Dividimos las diferencias por tipo de cirugía y eliminamos los NA.
  bypass_values <- na.omit(data_values_difference[data_values_difference$SURGERY == "by pass", metaboli
  tubular_values <- na.omit(data_values_difference[data_values_difference$SURGERY == "tubular", metaboli

  # Verificamos que tengan más de una muestra.
  if (length(bypass_values) > 1 & length(tubular_values) > 1) {
    # Prueba de normalidad.
    normality_test <- shapiro.test(c(bypass_values, tubular_values))$p.value > 0.05

    # Realizamos una prueba u otra según los resultados de normalidad.
    if (normality_test) {
      test_result <- t.test(bypass_values, tubular_values)
      test_type <- "t-test"
    } else {
```

```
      test_result <- wilcox.test(bypass_values, tubular_values)
      test_type <- "Wilcoxon"
    }

    # Calculamos las medias de cada grupo.
    mean_bypass <- mean(bypass_values, na.rm = TRUE)
    mean_tubular <- mean(tubular_values, na.rm = TRUE)

    # Guardamos solo los resultados si el test es significativo.
    if (test_result$p.value < 0.05) {
      significant_results <- rbind(significant_results,
                                   data.frame(Metabolite = metabolite,
                                              Test = test_type,
                                              p_value = test_result$p.value,
                                              Media_bypass = mean_bypass,
                                              Media_tubular = mean_tubular))
    }
  }
}
```

Finalmente, mostramos los resultados obtenidos.

```
significant_results
```

```
##       Metabolite     Test     p_value Media_bypass Media_tubular
## 1             TG   t-test 0.046423721 -45.08333333        -7.625
## 2            COL   t-test 0.010714643 -53.33333333        -5.000
## 3            C10 Wilcoxon 0.007963623  -0.03818182         0.294
## 4          C10.1 Wilcoxon 0.017609626  -0.86772727         0.124
## 5          C14.1   t-test 0.005662982  -0.04136364         0.084
## 6          C14.2   t-test 0.001965792  -0.02454545         0.030
## 7          C16.1   t-test 0.045763433  -0.01454545         0.012
## 8 lysoPC.a.C17.0   t-test 0.027428688   0.05636364         0.678
## 9 lysoPC.a.C18.0   t-test 0.031861434  -6.66818182        11.280
```

Solo se han encontrado diferencias entre grupos en las diferencias de lo metabolitos “TG”, “COL”, “C10”, “C10.1”, “C14.1”, “C14.2”, “C16.1”, “IysoPC.a.C17.0” y “IysoPC.a.C18.0”, y en todos ellos en el grupo bypass hay valores inferiores de diferencia respecto al grupo tubular.

Discusión y limitaciones y conclusiones del estudio

En esta actividad se ha analizado un dataset de un estudio con 39 pacientes que llevaban más de 10 años con obesidad y que fueron sometidos a cirugía bariátrica. Se clasificaron en dos grupos en función del tipo de cirugía recibida: el grupo “by pass” con 26 pacientes y el grupo “tubular” con 13 pacientes. El objetivo fue estudiar cómo evolucionan ciertos metabolitos en cuatro timepoints después de la intervención, considerando el tipo de cirugía.

Para facilitar los análisis, se ha organizado la información en un contenedor tipo SummarizedExperiment, permitiendo una gestión estructurada de datos y metadatos. El análisis inicial del dataset ha mostrado que contenía 156 muestras correspondientes a 39 pacientes y 180 variables, con escasa cantidad de valores faltantes, lo cual es positivo para asegurar la validez de futuros análisis. Además, se han revisado las

características basales entre los dos grupos, y se han identificado posibles diferencias visuales que podrían indicar perfiles iniciales distintos entre grupos, pero faltaría más análisis para confirmarlo.

En el análisis de metabolitos, se han observado diferencias significativas en la evolución entre el tiempo 0 y el tiempo 5 entre ambas cirugías. Los metabolitos “TG”, “COL”, “C10”, “C10.1”, “C14.1”, “C14.2”, “C16.1”, “LysoPC.a.C17.0” y “LysoPC.a.C18.0” han mostrado valores inferiores en el grupo by pass en comparación con el grupo tubular, lo que sugiere que podrían existir patrones de cambio diferentes según el tipo de cirugía.

Una limitación importante de este estudio es el reducido tamaño muestral en cada grupo de cirugía (by pass, n=26; tubular, n=13), y que en algunas muestras hay datos faltantes, lo que restringe la robustez de las comparaciones, dificultando análisis estadísticos más avanzados y la generalización de los resultados. Quizás se podrían reclutar más pacientes para futuros análisis.

Finalmente, se ha creado un repositorio en GitHub con todos los documentos relacionados con este análisis, disponible en el siguiente enlace:

https://github.com/CansecoRibas/Canseco-i-Ribas-Joan-PEC1.