PEC1-INFORME

2025-04-02

Contenido

ABSTRACT

En esta PEC1 se ha explorado un dataset de metabolómica extraido del paper “Metabotypes of response to bariatric surgery independent of the magnitude of weight loss”. Mediante un análisis inicial utilizand diferentes técnicas de análisis tanto uni como multivariante principalmente mediante el paquete de R POMA de BioConductor se ha realizado un análisis exploratorio del dataset, eliminando información no relevante, limpiandolo y normalizandolo, así como detectando algunos patrones, destacando el papel de los glicerofosfolípidos en diferenciar entre el grupo metabolicamente sano y no. El código está disponible en el repositorio de github: https://github.com/isi1000/Sobrino-Sanchez-Isidro

OBJETIVOS

1. Limpiar y preparar un dataset real para ser transformado en un objeto de la clase SummarizedExperiment
2. Utilizar paquetes de Bioconductor específicos para análisis Ómico

3. Realizar un análisis exploratorio de los datos, de manera que nos ayude a tomar decisiones sobre los datos
4. Extraer algunas primeras conclusiones visibles en el análisis multivariante

MÉTODOS

Elección del dataset

El dataset se ha extraido del repositorio de github metaboData (https://github.com/nutrimetabolomics/metaboData) que a su vez es un recopilatorio de datasets de otras bases de datos. Los criterios para la elección del dataset han sido el formato (CSV preferiblemente) y la variedad de metadatos.

Preparación del dataset

El dataset original se ha separado en dataframes para separar la información medible de los metadatos de muestras y metabolitos, por último, se ha utilizado la clase SummarizedExperiment para estructurar la información en un formato manejable por paquetes bioinformáticos

Análisis general

El análisis general se ha realizado mediante el paquete POMA, paquete específico muy utilizado en metabolómica. En primer lugar se han imputado los valores faltantes y se ha eliminado las muestras problemática, por último se han observado las distribuciones y normalizado los datos.

Análisis multivariante

El análisis multivariante se ha realizado mediante el paquete POMA, consistiendo en un análisis de componentes principales, así como un clustering mediante heatmap y mediante K-means.

RESULTADOS

Elección del dataset

En primer lugar se descargó del repositorio de github metaboData (https://github.com/nutrimetabolomics/metaboData) el dataset perteneciente al paper “Metabotypes of response to bariatric surgery independent of the magnitude of weight loss”, en el cual se estudia la respuesta metabólica a diferentes tipos de cirugía de pérdida de peso. He elegido este dataset porque incluye varios puntos temporales, lo cual me da mucha flexibilidad en el análisis y además añade un poco de complicación al proceso de preparación de los datos. El archivo se encuentra en formato .csv y la información de los pacientes y los resultados numéricos estan mezclados en un mismo archivo. De la misma manera se encuentran mezclados todos los puntos temporales.

Preparación previa del dataset

Para construir un objeto SummarizedExperiment necesitamos 3 dataframes: 1 para las mediciones de los metabolitos, otro para la información de las muestras y el último para la información de los metabolitos.

Por ello en primer lugar vamos a separar los datos del paciente del dataframe general.

```
dataset=                                read.csv("DataValues_S013.csv",                                sep=',')
#head(dataset)
dim(dataset)

## [1]  39 696
```

Tras una primera inspección del dataframe comprobamos que tiene 696 columnas , que deberían representar los metabolitos y 39 filas que representan las muestras del estudio. Una inspección a fondo de las columnas, contrastadas con la información recogida en el paper original nos hace darnos cuenta de que no todas las columnas se corresponden con los metabolitos, concretamente la primera columna que recoge información de metabolitos es Ile_T0, siendo las anteriores parámetros clínicos de los pacientes. Además, el sufijo Tx indica

mediciones a diferentes puntos temporales, por lo que habría que separar el dataset en los diferentes puntos temporales. Puesto que hay demasiados puntos (5) y esta práctica es limitada vamos a quedarnos solo con el punto basal (T0) y el final (T5):

```
library(dplyr)

dataset_T0 <- dataset %>% select(contains("T0"))
dataset_T5<-dataset %>% select(contains("T5"))
dim(dataset_T0)

## [1]  39 172

dim(dataset_T5)

## [1]  39 172
```

Estos datasets contienen todas las muestras y el mismo numero de columnas (metabolitos+información clínica). El siguiente paso es extraer la información de las muestras para dejar los datasets de metabolitos solo con las mediciones metabólicas, finalmente fusionaremos las columnas con información clínica con las columnas generales que se refieren a las muestras en el dataset:

```
#localizar donde está la primera columna de metabolitos
index_ile=which(names(dataset_T0) == "Ile_T0")
index_ile

## [1] 34

#filtramos y creamos el dataset de muestras y el de metabolitos
##T0
T0_sample=dataset_T0[,1:(index_ile-1)]
dataset_T0=dataset_T0[,index_ile:172]
##T5
T5_sample=dataset_T5[,1:(index_ile-1)]
dataset_T5=dataset_T5[,index_ile:172]

dim(T0_sample)

## [1] 39 33

dim(T5_sample)

## [1] 39 33

dim(dataset_T0)

## [1]  39 139

dim(dataset_T5)

## [1]  39 139
```

Para construir el dataset con la información de los pacientes necesitamos unir la información temporal (T0 y T5) con la que es constante (ID, grupo experimental, etc). Se encuentra en las primeras 6 columnas aunque las dos primeras están repetidas por lo que vamos a limpiarla:

```
#extraer datos pacientes
sample_dataset=dataset[,3:6]


#unir con los datos temporales
sample_dataset=cbind(sample_dataset,T0_sample,T5_sample)
sample_dataset[,1:4] <- lapply(sample_dataset, as.factor)

## Warning in `[<-.data.frame`(`*tmp*`, , 1:4, value = list(SURGERY =
## structure(c(1L, : provided 70 variables to replace 4 variables
```

```
dim(sample_dataset)

## [1] 39 70

#head(sample_dataset)
```

Una vez preparados ambos datasets vamos a modificarlos para que coincidan con el formato que nos pide Bioconductor para construir un objeto de la clase SummarizedExperiment. Según su web, para construir la clase necesitamos:

La matriz assay, la cual debe ser una matriz numerica en cuyas columnas estén las muestras y en las filas los genes (metabolitos en este caso). Además los nombres de columnas y filas deben coincidir con los de las otras matrices que importaremos. Actualmente tenemos un dataset con las muestras en las filas, por tanto debemos hacer una trasposición y convertirlo en matriz:

```
matriz_T0 <- t(as.matrix(dataset_T0))
matriz_T5 <- t(as.matrix(dataset_T5))
#cambiamos los nombres de las columnas porque se han perdido
colnames(matriz_T0)=rownames(sample_dataset)
colnames(matriz_T5)=rownames(sample_dataset)
#head(matriz_T0)
```

Creación del objeto SummarizedExperiment

Como son 2 matrices diferentes la clase SummarizedExperiments nos da la opción de importarlo como lista, pero antes debemos eliminar los sufijos _Tx para que sean iguales:

```
#eliminar sufijo
rownames(matriz_T0)=gsub("_T0$", "", rownames(matriz_T0))
rownames(matriz_T5)=gsub("_T5$", "", rownames(matriz_T5))
#juntarlo en una lista
assays_list <- list(T0 = matriz_T0,T5 = matriz_T5)
```

A continuación vamos a crear los otros dos argumentos obligatorios de la clase: RowData y ColData. RowData es un dataframe que contiene los metadatos de las filas (en este caso los metabolitos) mientras que ColData es lo mismo pero para las muestras. Los nombres de las filas de ambos dataframes deben coincidir con los nombres de las filas y las columnas respectivamente en la matriz.

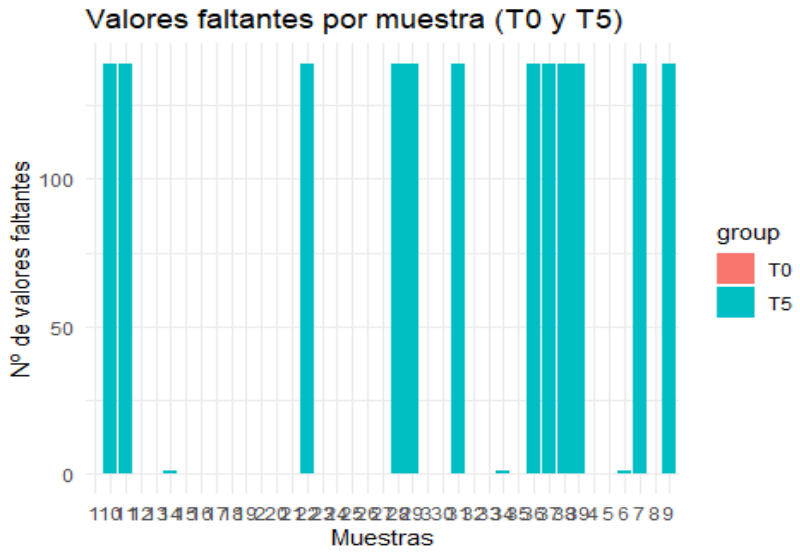
```
#construimos la matriz y nos aseguramos de que los nombres coincidan
RowDataframe=data.frame(metabolite=rownames(matriz_T0))
rownames(RowDataframe)=rownames(matriz_T0)
ColDataframe=sample_dataset
rownames(ColDataframe)=colnames(matriz_T0)

library(SummarizedExperiment)

#construir objeto SE
met_exp <- SummarizedExperiment(
  assays = assays_list,
  rowData = RowDataframe,
  colData = ColDataframe)
```

El objeto creado contiene una lista con los datos numericos de las mediciones de metabolitos en T0 y T1, así como los metadatos con la información relevante de las muestras y los metabolitos.

Análisis general

Para iniciar el análisis vamos a responder preguntas simples como el número de valores faltantes (y donde se encuentran), distribuciones por muestra (así podemos ver la escala, valorar si es necesaria una normalización, etc) y algunas cuestiones más:

Iniciamos con el número de valores faltantes en ambos puntos temporales:

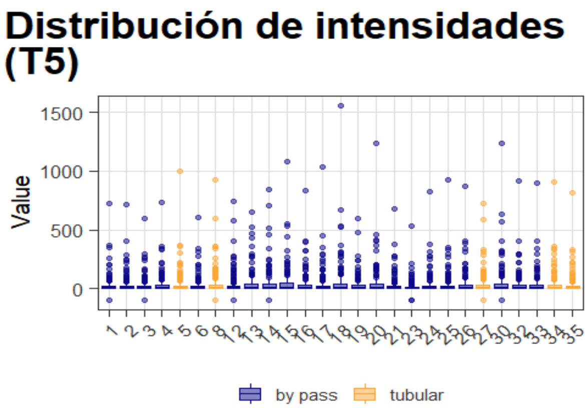


Podemos ver que no existen valores faltantes en T0, mientras que hay valores faltantes en las muestras 6,7,9,10,11,14,22,28,29,31,34,36,37,38 y 39 de T5, de estas las muestras 6,14 y 34 tienen un solo valor faltante, mientras que el resto tienen valores faltantes en todos los metabolitos, lo que podría indicar pérdida de esas muestras durante el experimento. De ahora en adelante vamos a eliminar esas muestras de nuestro análisis. A las muestras a las que le falta un solo valor faltante le vamos a imputar valores mediante el paquete de análisis metabolómico POMA. Este paquete incluye una función que imputa valores faltantes (PomaImpute), en este caso vamos a imputar con el algoritmo K-NN que asume el valor de la muestra que más se parezca (o muestras) y además vamos a eliminar aquellos metabolitos que tengan más de un 20% de valores faltantes. Para trabajar con POMA deberemos separar el objeto SummarizedExperiments en 2, ya que no admite objetos con 2 arrays diferentes.

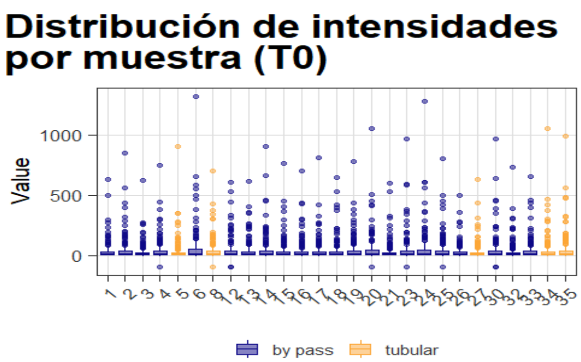
```
library(POMA)

library(ggtext)

library(magrittr)

#descartar muestras sin datos
met_exp_filtrado <- met_exp[, -c(7,9,10,11,22,28,29,31,36,37,38,39)]
#met_exp_filtrado

met_T0 <- SummarizedExperiment(
  assays = list(T0 = assay(met_exp_filtrado, "T0")),
  rowData = rowData(met_exp_filtrado),
  colData = colData(met_exp_filtrado)
)
met_T5 <- SummarizedExperiment(
  assays = list(T5 = assay(met_exp_filtrado, "T5")),
  rowData = rowData(met_exp_filtrado),
  colData = colData(met_exp_filtrado)
)
#imputar valores faltantes con POMA, se usa el método k-vecinos mas cercanos, los 0 se consideran NA y se eliminan aquellos metabolitos con más de un 20% de valores faltantes.
imputed_T5 = met_T5 %>%
  PomaImpute(method = "knn", zeros_as_na = TRUE, remove_na = TRUE, cutoff = 20)

## 0 features removed.

imputed_T0 = met_T0 %>%
  PomaImpute(method = "knn", zeros_as_na = TRUE, remove_na = TRUE, cutoff = 20)

## 0 features removed.
```

Una vez imputados los valores faltantes y limpiado el dataset vamos a explorar la distribución de las muestras: su media, desviación típica y posibles outliers. De esta forma podremos valorar la necesidad de utilizar métodos de normalización:

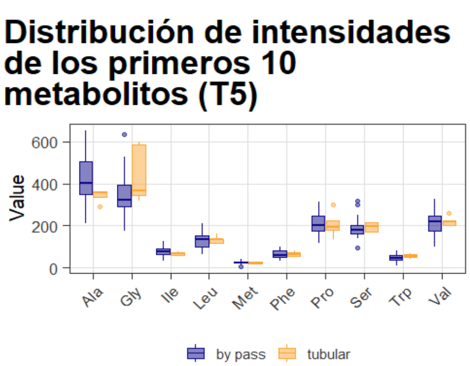
```
#boxplot T0
PomaBoxplots(imputed_T5, x = "samples") +
 labs(title = "Distribución de intensidades (T5)")
```



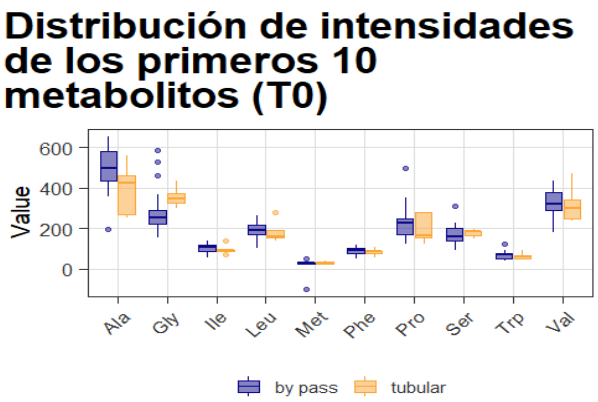

```
#boxplot T5
PomaBoxplots(imputed_T0, x = "samples") +
 labs(title = "Distribución de intensidades por muestra (T0)")
```




```
#boxplot T0
PomaBoxplots(imputed_T5[1:10,], x = "features") +
 labs(title = "Distribución de intensidades de los primeros 10 metabolitos (T5)")
```



```
#boxplot                                                                    T5
PomaBoxplots(imputed_T0[1:10,],          x          =          "features")          +
 labs(title = "Distribución de intensidades de los primeros 10 metabolitos (T0)")
```

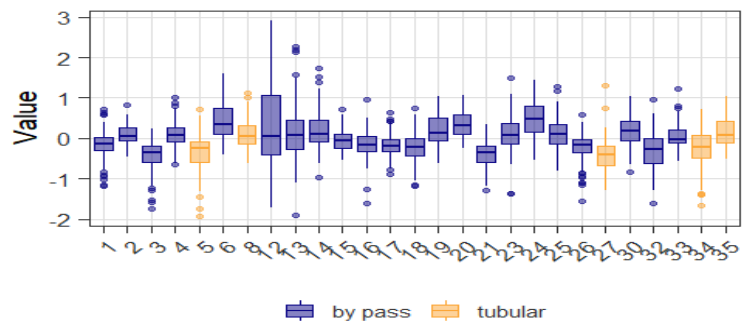


De las gráficas de muestras observamos varios puntos que salen de los cuantiles del boxplot y de los bigotes, podríamos interpretarlos como outliers, sin embargo observamos en los gráficos de metabolitos como hay algunos con valores bastante altos, lo que podría indicar que los outliers en las muestras son en realidad metabolitos con unos valores altos de forma natural. En las gráficas por metabolitos se pueden observar algunas diferencias entre T0 y T5, como por ejemplo en la glicina o alanina, así como diferencias entre grupos. Por último es importante destacar que se observan algunos valores negativos, lo cual es biológicamente imposible, ya se encontraban en el dataset inicial, por lo que probablemente son artefactos técnicos, deben ser eliminados e imputados de nuevo:

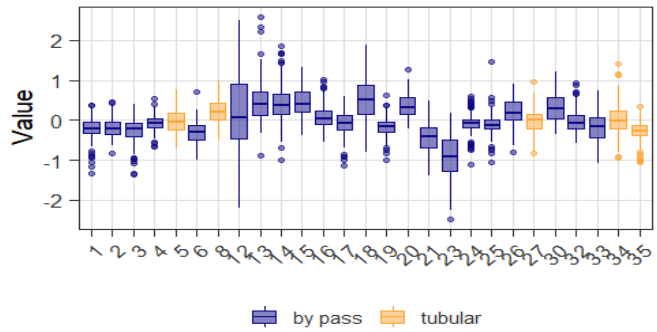
```
assay(imputed_T5)[assay(imputed_T5)<0]=NA
assay(imputed_T0)[assay(imputed_T0)<0]=NA
no0_T5                                   =                          imputed_T5                          %>%
 PomaImpute(method = "knn", zeros_as_na = TRUE, remove_na = TRUE, cutoff = 50)

## 0 features removed.

no0_T0                                   =                          imputed_T0                          %>%
 PomaImpute(method = "knn", zeros_as_na = TRUE, remove_na = TRUE, cutoff = 50)

## 0 features removed.
```

Dada la diferencia entre metabolitos es necesario normalizar, para ello utilizaremos otra función de POMA:

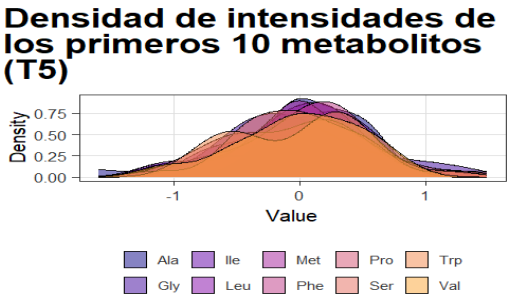
```
normalized_T0                          <-                          no0_T5                          %>%
 PomaNorm(method                                =                                "log_pareto")

normalized_T5                          <-                          no0_T0                          %>%
 PomaNorm(method = "log_pareto")

#boxplot                                                                    T0
PomaBoxplots(normalized_T5,          x          =          "samples")          +
 labs(title = "Distribución de intensidades (T5)")
```



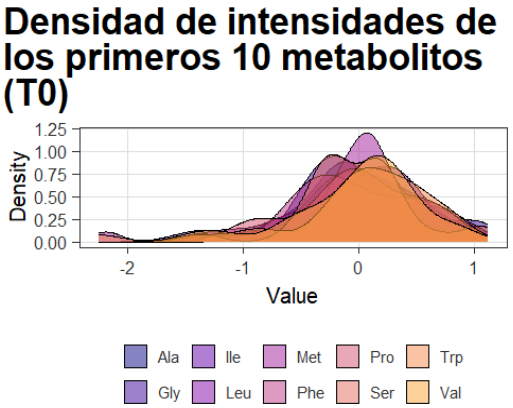

```
#boxplot                                                                    T5
PomaBoxplots(normalized_T0,                x          =          "samples")      +
 labs(title = "Distribución de intensidades por muestra (T0)")
```




```
#density                                                                    T0
PomaDensity(normalized_T5[1:10,],          x          =          "features")     +
 labs(title = "Densidad de intensidades de los primeros 10 metabolitos (T5)")
```



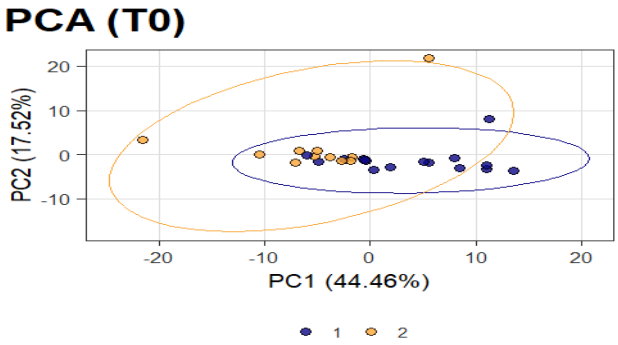

```
#density                                                                    T5
PomaDensity(normalized_T0[1:10,],          x          =          "features")     +
 labs(title = "Densidad de intensidades de los primeros 10 metabolitos (T0)")
```



Análisis multivariante

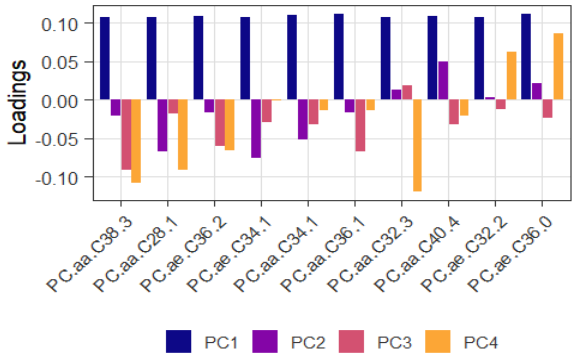
En primer lugar vamos a hacer una análisis de componentes principales para ver si las muestras se separan bien por grupos o podemos detectar algún otro patrón:

```
PCA_T0=PomaPCA(
normalized_T0,
outcome = "Group",
center = TRUE,
scale = TRUE,
ncomp = 4,
labels = FALSE,
ellipse = TRUE,
load_length = 1
)

PCA_T5=PomaPCA(
normalized_T5,
outcome = "Group",
center = TRUE,
scale = TRUE,
ncomp = 4,
labels = FALSE,
ellipse = TRUE,
load_length = 1
)
PCA_T0$factors_plot+
  labs(title = "PCA (T0)")
```



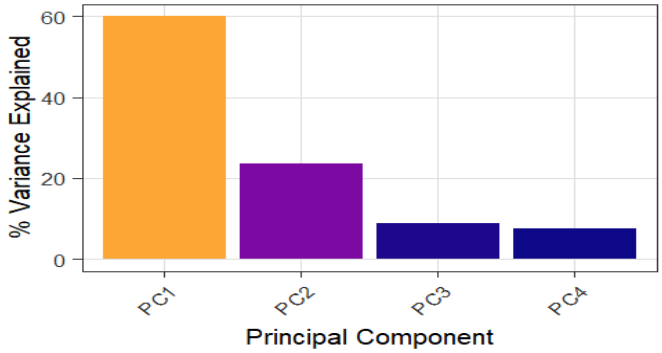

```
PCA_T0$loadings_plot+
  labs(title = "Loading plot (T0)")
```

Loading plot (T0)



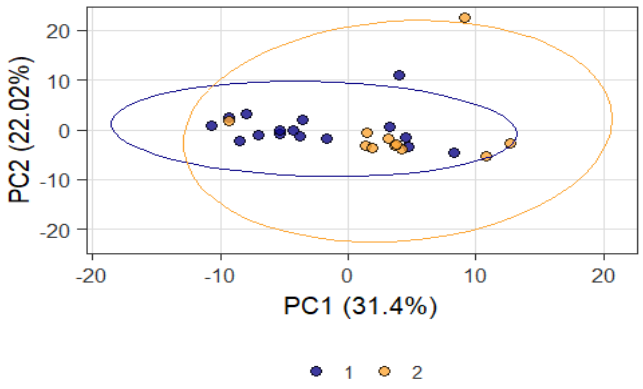

```
PCA_T0$eigenvalues_plot+
  labs(title = "Scree Plot (T0)")
```

Scree Plot (T0)

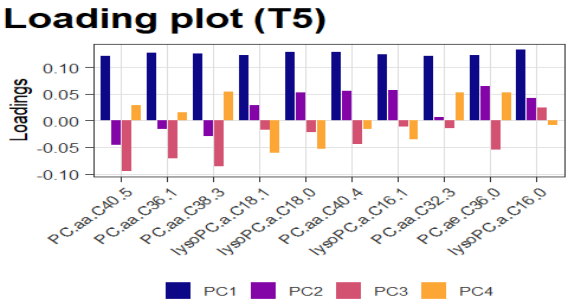



```
PCA_T5$factors_plot+
  labs(title = "PCA (T5)")
```

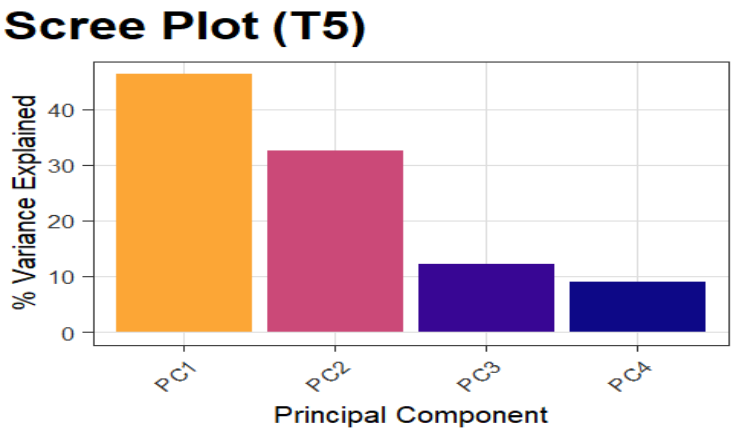
PCA (T5)




```
PCA_T5$loadings_plot+
  labs(title = "Loading plot (T5)")
```



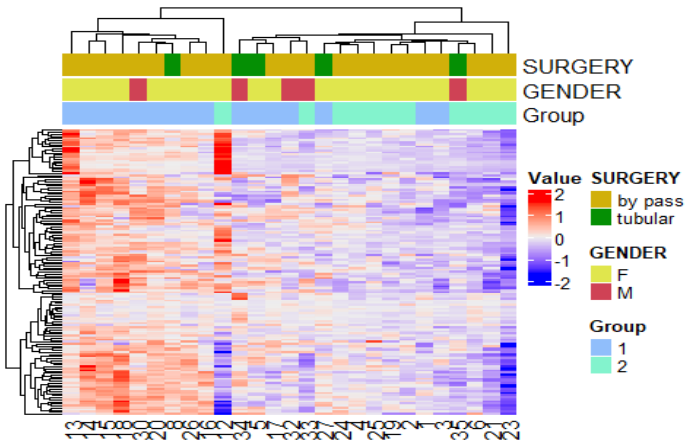

```
PCA_T5$eigenvalues_plot+
  labs(title = "Scree Plot (T5)")
```



Del análisis de componentes principales se puede intuir cierto solapamiento entre grupos, aunque esto podría estar influido por la alta dispersión que encontramos en el grupo 2, incluso parece observarse una posible muestra outlier. Además, vemos como los metabolitos que más contribuyen a la separación entre grupos son los glicerofosfolípidos, lo cual tiene sentido en el contexto de un paper sobre pérdida de peso. No vamos a estudiar otros componentes principales ya que los 2 primeros ya explican suficiente porcentaje de la varianza. Es reseñable como los PCAs no cambian demasiado entre ambos puntos temporales y como los glicerofosfolípidos siguen siendo determinantes.

Por último vamos a realizar un clustering jerárquico de las muestras, para ver si encontramos algún patrón relevante:

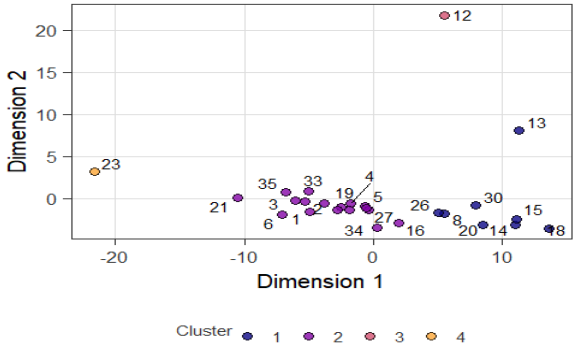
```
PomaHeatmap(normalized_T0,covs=c(1,3,4))+
  labs(title = "heatmap (T0)")
```



Vemos como las muestras del grupo 1 tienden a tener valores de expresión más altos que las del grupo 2. Respecto al tipo de cirugía o el género las muestras no parecen seguir un patrón concreto (motivo por el cual no se clusterizan). A cotinuación vamos a aprovechar el paquete POMA que incluye un clustering jerárquico usando K-means para poder obtener un número de clusters óptimo.

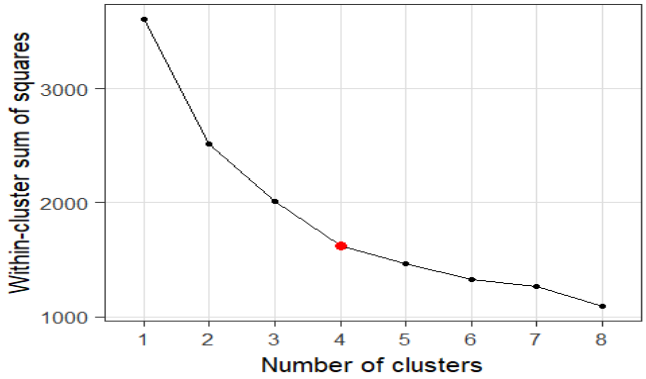
```
POMA_T0=PomaClust(
normalized_T0,
method                    =                    "euclidean",
k                         =                    NA,
k_max                     =                    8,
show_clusters             =                    TRUE,
labels                    =                    TRUE
)
POMA_T0$mds_plot
```

```
## Warning: ggrepel: 4 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```




```
POMA_T0$optimal_clusters_plot
```

```
## Warning in ggplot2::geom_point(ggplot2::aes(x = k[elbowi], y = wss[elbowi]), : All aesthetics have length
1, but the data has 8 rows.
## ℹ Please consider using `annotate()` or provide this layer with data containing
##   a single row.
```



Vemos como la gráfica hace un codo en el número 4, motivo por el cual se han agrupado las muestras en 4 clusters. Vemos 2 grupos numerosos el 1 y el 2. El grupo número 1 se corresponde con el verdadero grupo 1 del experimento mayormente, mientras que el grupo 2 contiene casi todos los miembros del grupo 2 experimental y algunos del uno. Por su parte los grupos 3 y 4 se corresponden con 2 muestras del grupo 2 experimental. El hecho de que estas muestras estén tan separadas de las demás podría ser indicativo de que

son outliers o se deben a errores experimentales, además, en el PCA también veíamos como 2 muestras aumentaban mucho la dispersión del grupo 2 solapándolo con el grupo 1.

DISCUSIÓN

Este dataset presentaba diversos problemas en cuanto a su estructura, motivo por el cual se ha tenido que separar de 1 solo archivo en 3 datasets diferentes. Además, se ha tenido que descartar gran parte de la información por provenir de un estudio muy complejo que involucraba diferentes puntos temporales.

Había varias muestras con valores faltantes, motivo por el cual se han descartado varias de ellas al no tener información para ninguno de los metabolitos en el T5. El resto de muestras con valores faltantes han sido imputadas mediante con el método del vecino más cercano, impiediendo la pérdida de información. Además se han normalizado los valores para facilitar el análisis posterior.

El análisis multivariante nos ha sugerido que el "grupo" separaba mejor las muestras que otras características como el tipo de cirujía. En el paper original se aclara que el grupo 1 son los pacientes metabólicamente sanos y el grupo 2 los que no lo son. En las proyecciones mediate PCA hemos podido observar una clara separación entre los grupos en T0, a pesar de estar influenciada por la presencia de 2 posibles outliers en el grupo 2. Esta separación se mantiene muy similar en T5 (quinto punto medido tras la cirujía), lo que podría indicar que no hay una mejora a nivel metabólico tras esta cirujía, aunque de nuevo, la presencia de 2 posibles outliers en el grupo 2 hacen necesario un análisis más exahustivo para confirmarlo. El análisis jerárquico vuelve a sugerir la presencia de estos dos outliers separando en grupos diferentes las muestras 23 y 12.

CONCLUSIÓN

1. Se ha conseguido construir el objeto SummarizedExperiments
2. El dataset mostraba varios problemas en cuanto a valores faltantes, dispersión de los datos y complejidad
3. Los grupos de estudio (metabolicamente sano o no) separan correctamente las muestras en T0.
4. No se observa un cambio demasiado relevante tras la cirujía (T5) en la separación entre estos grupos.
5. Se han detectado 2 posibles muestras outliers que dificultan el análisis.

REFERENCIAS

https://www.bioconductor.org/packages/release/bioc/html/POMA.html

https://github.com/nutrimetabolomics/metaboData/tree/main/Datasets/2018-MetabotypingPaper