Informe PEC1

Adriana González

2024-11-06

Para esta actividad, he escogido el dataset "2018-MetabotypingPaper". En esta investigación, se estudia el impacto de la cirugía bariátrica en los perfiles metabólicos de los pacientes. El objetivo de la investigación es identificar distintos "metabotipos" basados en la evolución metabólica tras la cirugía.

En este informe, he descrito los distintos pasos de descarga de los datos, creación de un contenedor *SummarizedExperiment*, pre-procesamiento y exploración de los datos, y finalmente la subida de los archivos a un repositorio de GitHub.

Descarga de los datos

Para descargar los datos, podemos hacerlo a través del repositorio de GitHub. Para ello, escribimos en la terminal *git clone* seguido de la dirección del repositorio:

git clone https://github.com/nutrimetabolomics/Metabotyping2018

Una vez descargado en nuestro proyecto, podemos cargar los datos.

```
# Datos de indicadores antropométricos, clínicos y metabólicos
dataValues <- read.csv("Metabotyping2018/datasets/DataValues_S013.csv")

# Guardamos dataValues en formato texto
write.table(dataValues, file = "dataValues.txt", sep = ",", row.names = TRUE, col.names = TRUE)

# Información sobre las variables
featuresInfo <- read.csv("Metabotyping2018/datasets/DataInfo_S013.csv")

# Descripción metabolitos
AAInfo <- read.csv("Metabotyping2018/datasets/AAInformation_S006.csv")
```

Creación del contenedor

Creamos un contenedor del tipo *SummarizedExperiment*, incluyendo los valores de *dataValues* como datos. En cuanto a los metadatos, existe una tabla que proporciona información sobre el tipo de datos de cada variable medida en los datos, que utilizaremos como metadatos de las columnas. Además, podemos extraer algunas columnas de *dataValues* que poseen información relevante sobre los pacientes, para utilizarlas como metadatos de las filas.

En primer lugar, extraeremos dichas columnas desde *dataValues* y las almacenaremos en *samplesInfo* para su uso posterior como metadatos de las muestras.

```
samplesInfo <- dataValues[,3:6] # Información sobre los individuos
```

Preparamos nuestro dataset a partir de *dataValues*. Realizamos una transposición de los datos, para que las muestras se dispongan en las columnas y las variables, en las filas.

```
# Eliminamos columna del dataset con todo NAs
dataValues <- dataValues[,-258]

# Transponemos dataValues
dataValues <- t(dataValues)

# Nombramos las columnas con el número de muestra
colnames(dataValues) <- seq(1, 39)

# Convertimos conjunto de datos a matriz
dataValues <- as.matrix(dataValues[7:695,])

# Rellenamos los metadatos del experimento
metadata <- list(Experimenter_name="Magali Palau-Rodriguez, Sara Tulipani, Anna Marco-Ramell,
                 Antonio Miñarro, Olga Jáuregui, Alex Sanchez-Pla,
                 Bruno Ramos-Molina, Francisco J Tinahones, Cristina Andres-Lacueva",
            Contact_info="aandres@ub.edu",
            Title="Metabotypes of response to bariatric surgery independent of the
            magnitude of weight loss")
```

Para preparar los metadatos de las variables, eliminamos las filas innecesarias de *featuresInfo*.

```
# Eliminamos fila correspondiente a la columna con todo NAs del dataset
featuresInfo <- featuresInfo[-257, ]

# Seleccionamos variables excluyendo aquellas eliminadas de dataValues
featuresInfo <- featuresInfo[6:694,]

# Resumen datos
head(dataValues)
```

```
##           1       2       3       4       5       6       7       8
## MEDDM_T0  " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDCOL_T0 " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDINF_T0 " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 1"
## MEDHTA_T0 " 1"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## GLU_T0    " 85"   " 78"   " 75"   " 71"   " 82"   " 71"   " 80"   " 90"
## INS_T0    "11.40" "12.10" " 8.41" "12.80" " 6.01" " 9.88" " 9.20" " 3.40"
##           9       10      11      12      13      14      15      16
## MEDDM_T0  " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDCOL_T0 " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDINF_T0 " 0"    " 0"    " 1"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDHTA_T0 " 0"    " 0"    " 0"    " 0"    " 1"    " 1"    " 1"    " 1"
## GLU_T0    " 92"   " 84"   " 75"   "108"   "101"   "105"   "139"   "106"
## INS_T0    " 5.43" " 6.98" "13.30" "16.80" "17.10" "21.30" "36.60" "20.00"
##           17      18      19      20      21      22      23      24
## MEDDM_T0  " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
```

```
## MEDCOL_T0 " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDINF_T0 " 0"    " 0"    " 0"    " 1"    " 0"    " 0"    " 0"    " 1"
## MEDHTA_T0 " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## GLU_T0    "159"   "103"   "106"   "107"   "127"   "111"   "141"   "100"
## INS_T0    "17.60" "29.50" "13.30" "15.00" "15.00" "12.20" "32.30" "16.00"
##           25      26      27      28      29      30      31      32
## MEDDM_T0  " 0"    NA      " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDCOL_T0 " 0"    NA      " 0"    " 1"    " 0"    " 0"    " 0"    " 0"
## MEDINF_T0 " 0"    NA      " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDHTA_T0 " 0"    NA      " 0"    " 1"    " 0"    " 0"    " 0"    " 0"
## GLU_T0    "100"   "100"   "100"   "117"   "100"   "263"   "115"   "108"
## INS_T0    "12.80" "11.10" "19.60" "11.60" "13.70" "21.00" "19.00" "23.10"
##           33      34      35      36      37      38      39
## MEDDM_T0  " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDCOL_T0 " 0"    " 0"    " 0"    " 0"    " 0"    " 0"    " 0"
## MEDINF_T0 " 0"    " 0"    " 0"    " 0"    " 1"    " 0"    " 0"
## MEDHTA_T0 " 1"    " 1"    " 1"    " 0"    " 0"    " 0"    " 0"
## GLU_T0    "114"   "101"   "108"   "106"   "115"   "102"   "108"
## INS_T0    "27.80" "23.70" "17.70" "16.10" "43.00" "21.90" "42.70"
```

```
# Resumen metadatos variables/ filas
head(featuresInfo)
```

```
##            X   VarName  varTpe Description
## 6   MEDDM_T0  MEDDM_T0 integer    dataDesc
## 7  MEDCOL_T0 MEDCOL_T0 integer    dataDesc
## 8  MEDINF_T0 MEDINF_T0 integer    dataDesc
## 9  MEDHTA_T0 MEDHTA_T0 integer    dataDesc
## 10    GLU_T0    GLU_T0 integer    dataDesc
## 11    INS_T0    INS_T0 numeric    dataDesc
```

```
# Resumen metadatos individuos/ columnas
head(samplesInfo)
```

```
##   SURGERY AGE GENDER Group
## 1 by pass  27      F     1
## 2 by pass  19      F     2
## 3 by pass  42      F     1
## 4 by pass  37      F     2
## 5 tubular  42      F     1
## 6 by pass  24      F     2
```

Creamos el contenedor de tipo *SummarizedExperiment* que contiene los datos, la información de las columnas, la información de las filas y los metadatos de la investigación:

```
(se <- SummarizedExperiment(assays = list(count = dataValues),
                            rowData = featuresInfo,
                            colData = samplesInfo,
                            metadata = metadata))
```

```
## class: SummarizedExperiment
## dim: 689 39
```

```
## metadata(3): Experimenter_name Contact_info Title
## assays(1): count
## rownames(689): MEDDM_T0 MEDCOL_T0 ... SM.C24.0_T5 SM.C24.1_T5
## rowData names(4): X VarName varTpe Description
## colnames(39): 1 2 ... 38 39
## colData names(4): SURGERY AGE GENDER Group
```

```
# Guardamos el contenedor en un archivo RDA
save(se, file = "se_metabo.rda")
```

Pre-procesamiento de los datos

Eliminación de NAs y normalización

En primer lugar, analizaremos nuestro dataset para identificar la presencia de valores faltantes (o NAs). Se puede observar que muchas de las muestras contienen una cantidad considerable de NAs.

Podemos utilizar la función *PomaImpute* para imputar los valores faltantes (o NAs). Utilizaremos el método de imputación 'knn', que estima los NAs en función de las muestras más similares, lo que minimiza la pérdida de información.

```
# Contamos NAs
(na_count <- colSums(is.na(assay(se))))
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##  15 157   0   0  21   1 173  14 175 173 180  16  14  12   9   7   6   4   9   7
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39
##   3 182  21   7   3  30  23 333 155   9 180  26  21 148  21 327 356 186 327
```

```
# Eliminamos NAs
dataValues_clean <- PomaImpute(se, method = "knn")
```

Tras la eliminación de los NAs, algunos de los valores resultan negativos. Al normalizar, esto puede causar problemas si realizamos una transformación logarítmica. Por este motivo, al aplicar la función *PomaNorm* para la normalización de los datos, escogemos el método **auto_scaling**, que no conlleva ninguna transformación logarítmica.

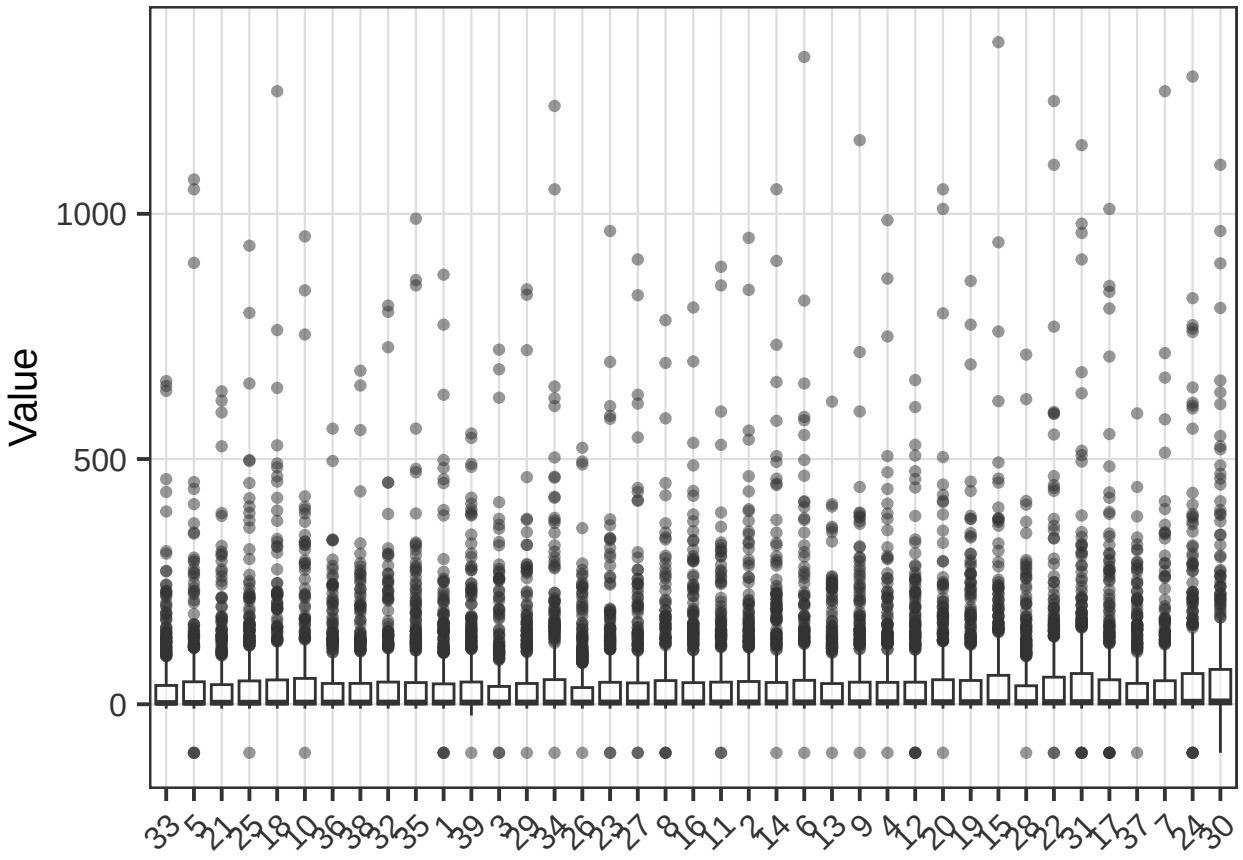
```
# Normalizamos datos
(dataValues_norm <- dataValues_clean %>% PomaNorm(method = "auto_scaling"))
```

```
## class: SummarizedExperiment
## dim: 513 39
## metadata(0):
## assays(1): ''
## rownames(513): MEDDM_T0 MEDCOL_T0 ... TRANSF_T5 FERR_T5
## rowData names(0):
## colnames(39): 1 2 ... 38 39
## colData names(4): SURGERY AGE GENDER Group
```

Tras la normalización observamos que el rango de los valores es mucho menor, lo cual facilitará su comparación. Los datos normalizados presentan una distribución similar y centrada en el cero, lo que nos indica que la normalización se ha realizado correctamente.

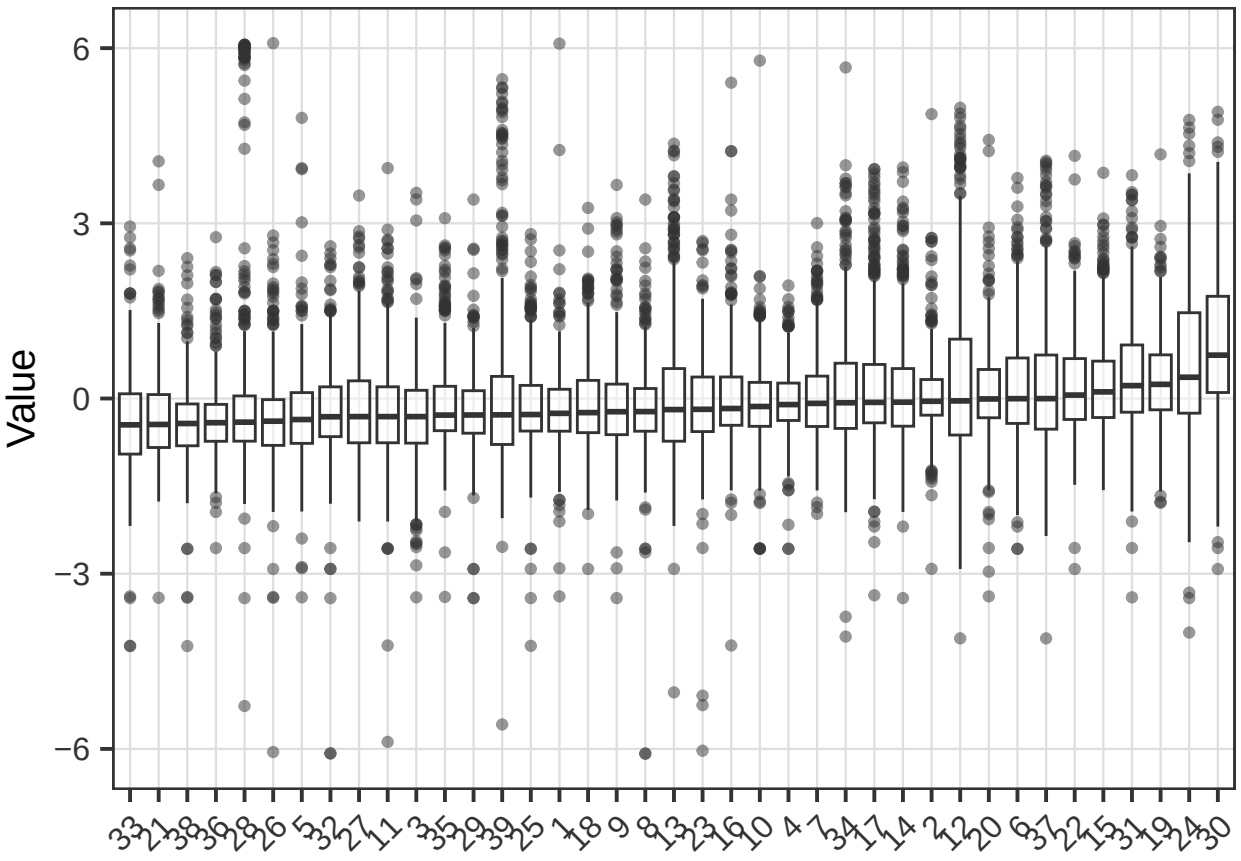
Datos antes de normalizar

```
(boxplot_notnorm <- PomaBoxplots(dataValues_clean, x = "samples"))
```



Datos después de normalizar

```
# Datos después de normalizar
(boxplot_norm <- PomaBoxplots(dataValues_norm, x = "samples"))
```



Exploración de los datos

Visualización de los datos

En primer lugar, utilizaremos gráficos para visualizar los datos. Definimos una función que crea histogramas o gráficos de barras, en función del tipo de variable.

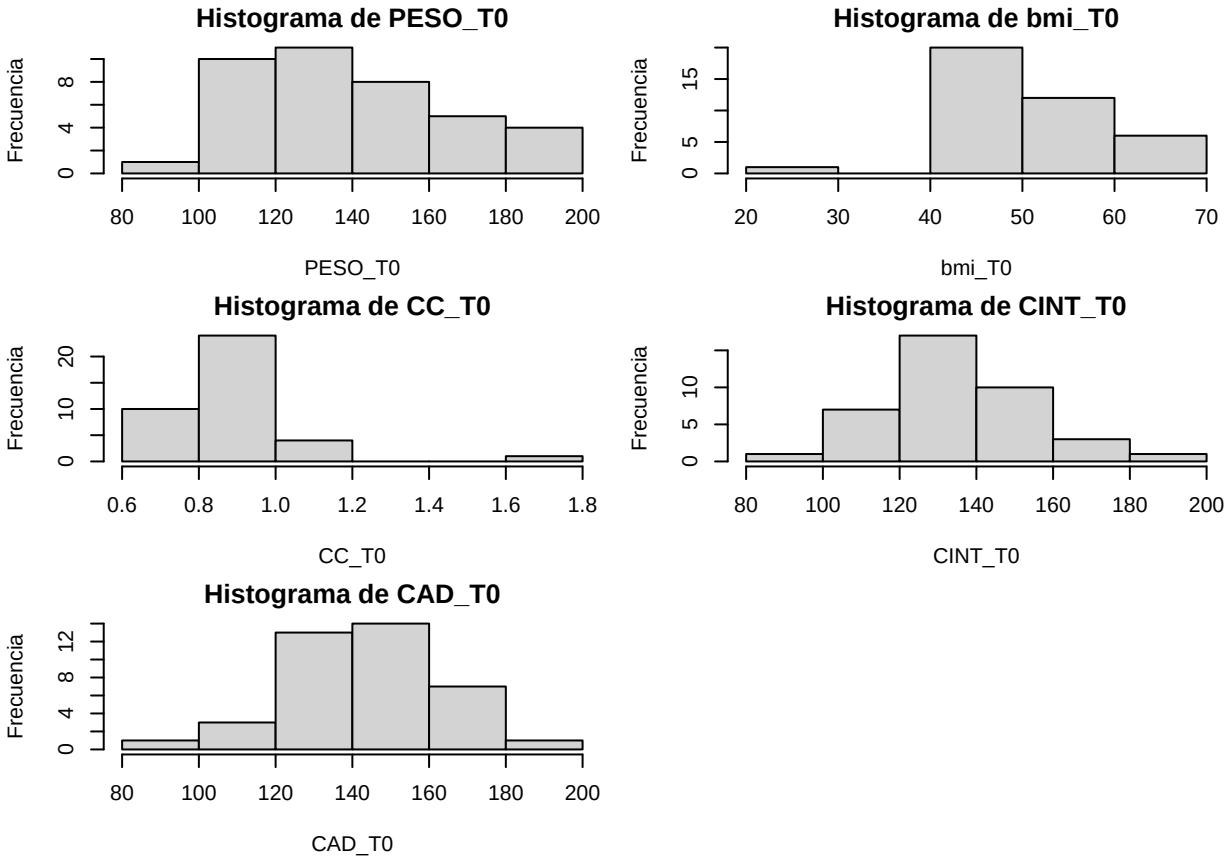
```
f <- function(x, name) {
  if (is.numeric(x)) {
    hist(x, breaks = 5, main = paste("Histograma de", name), xlab = name, ylab = "Frecuencia")
  } else {
    barplot(table(x), main = paste("Barplot de", name), xlab = name, ylab = "Frecuencia")
  }
}
```

Dado el gran número de variables que posee nuestro dataset, seleccionaremos un subconjunto para la visualización, a fin de mostrar la distribución de las variables a modo de ejemplo. Utilizaremos los datos no normalizados para preservar la escala original de los datos.

Visualización de Indicadores Antropométricos

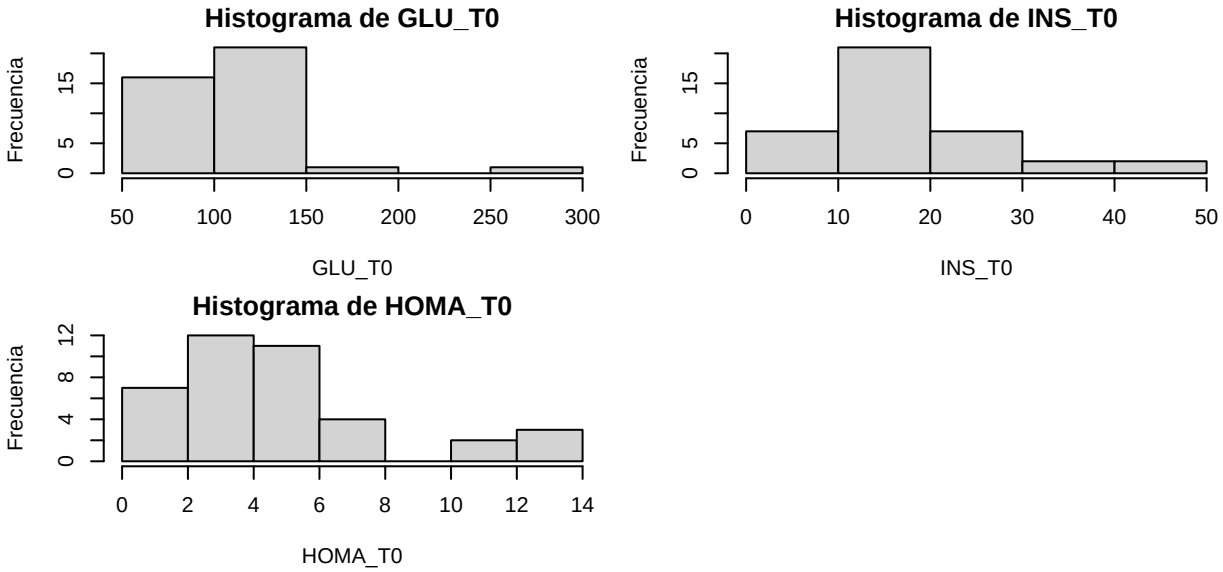
```
# Configuramos la disposición de los gráficos
par(mfrow = c(3, 2), mar = c(4, 4, 2, 1))

# Seleccionamos columnas con indicadores antropométricos T0 (medidos antes de la cirugía)
```

```r
ind_antropo <- (t(assay(dataValues_clean)))[,c(8:12)]

# Aplicamos la función a cada fila, usando invisible() para evitar que los resultados
# de lapply se impriman
invisible(lapply(1:ncol(ind_antropo), function(i) f(ind_antropo[,i],
                                                    colnames(ind_antropo)[i])))
```



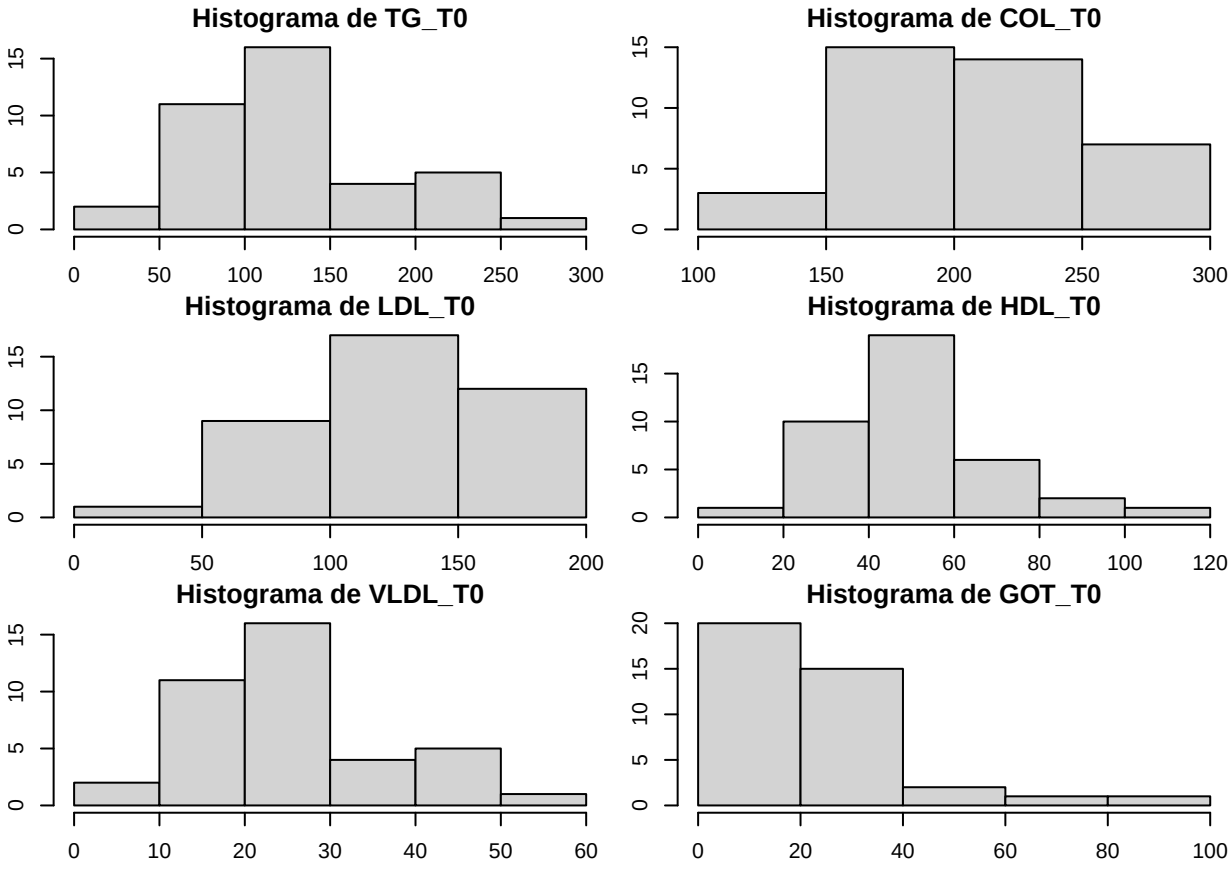
Visualización de Indicadores de regulación de glucosa

```r
par(mfrow = c(3, 2), mar = c(4, 4, 2, 1))

# Seleccionamos columnas con indicadores de regulación de glucosa T0 (medidos antes de la cirugía)
ind_gluc <- (t(assay(dataValues_clean)))[,c(5:7)]

invisible(lapply(1:ncol(ind_gluc), function(i) f(ind_gluc[, i],
                                                 colnames(ind_gluc)[i])))
```

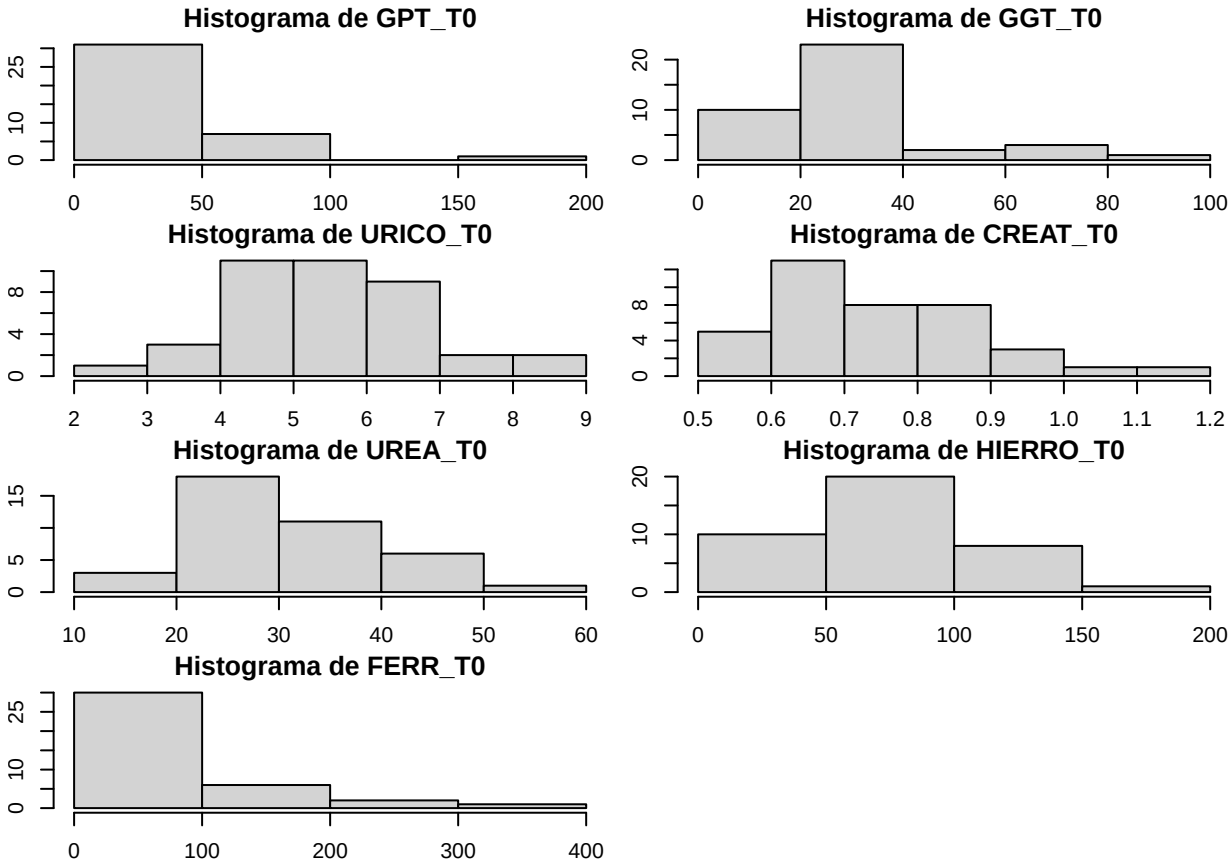


Visualización de Indicadores clínicos

```
par(mfrow = c(3, 2), mar = c(1.8, 1.8, 1.8, 1))

# Seleccionamos columnas con indicadores clínicos T0 (medidos antes de la cirugía)
ind_clin <- (t(assay(dataValues_clean)))[, c(13:18)]

invisible(lapply(1:ncol(ind_clin), function(i) f(ind_clin[, i],
                                                 colnames(ind_clin)[i])))
```




```
par(mfrow = c(4, 2), mar = c(1.8, 1.8, 1.8, 1))

ind_clin2 <- (t(assay(dataValues_clean)))[, c(19:25)]

invisible(lapply(1:ncol(ind_clin2), function(i) f(ind_clin2[, i],
                                                  colnames(ind_clin2)[i])))
```



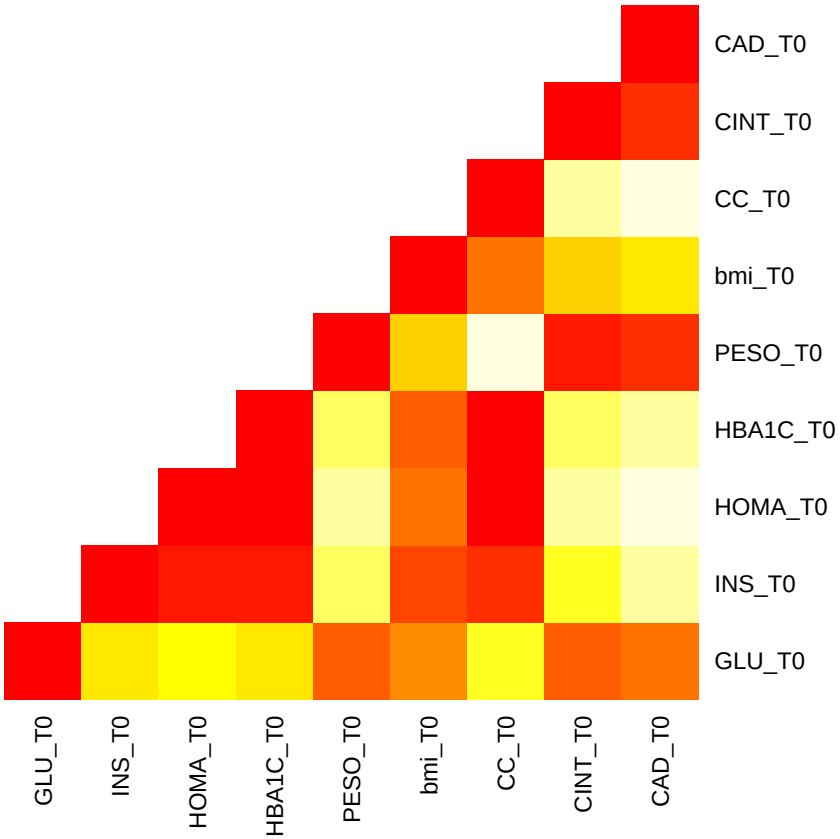
Matriz de distancias

Para analizar las relaciones entre las variables previamente visualizadas (antropométricas y regulación de glucosa), utilizaremos una matriz de distancias. Primero, crearemos un subconjunto del objeto *Summarized-Experiment* que incluye sólo las variables de interés.

Datos no normalizados

```
# Creamos un subconjunto del contenedor
col_selec <- se[rownames(se) %in% c("PESO_T0", "bmi_T0", "CINT_T0",
                                    "CAD_T0","CC_T0", "GLU_T0", "INS_T0",
                                  "HOMA_T0", "HBA1C_T0"), ]

# Calculamos matriz de distancias
manDist <- dist(assay(col_selec))

# Convertimos en matriz y convertimos valores de la mitad inferior en NA
lower_dist <- as.matrix(manDist)
lower_dist[lower.tri((lower_dist))] <- NA

# Generamos un heatmap para visualizar las distancias entre los individuos
heatmap(lower_dist,
        col = heat.colors(16),
```

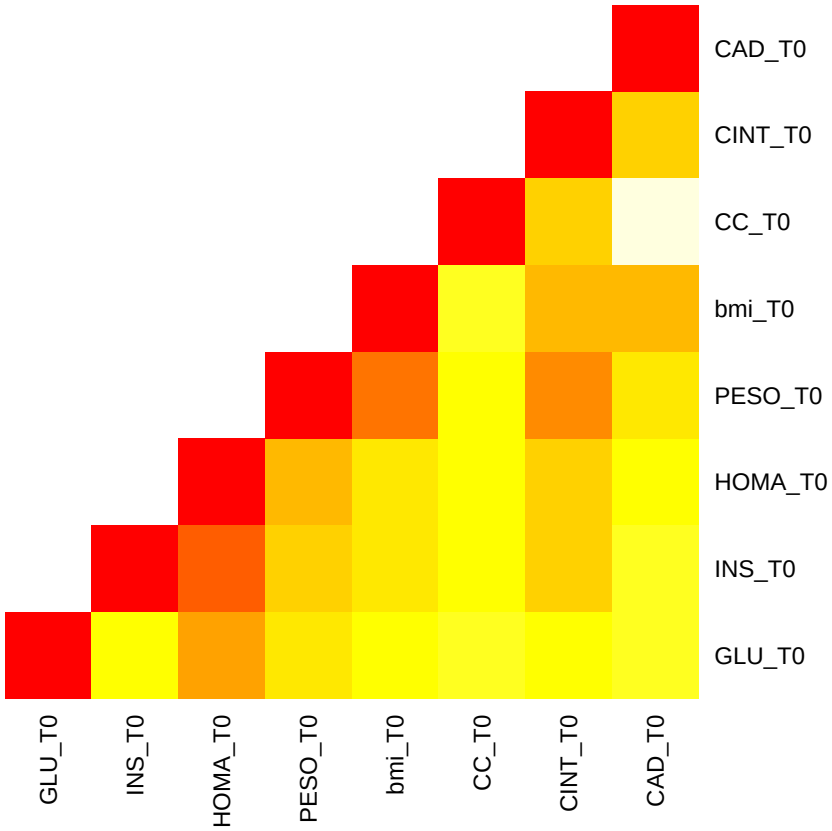
```
        Rowv = NA,        # No reorganizar filas
        Colv = NA,        # No reorganizar columnas
        scale = "none",   # No escalar
        margins = c(5, 5), # Márgenes para las etiquetas
        cexRow = 0.9,     # Tamaño fuente filas
        cexCol = 0.9)     # Tamaño fuente columnas
```



Datos normalizados

```
# Creamos un subconjunto del contenedor
col_selec <- dataValues_norm[rownames(dataValues_norm) %in% c("PESO_T0", "bmi_T0", "CINT_T0",
                                        "CAD_T0","CC_T0", "GLU_T0", "INS_T0",
                                    "HOMA_T0", "HBA1C_T0"), ]

# Calculamos matriz de distancias
manDist <- dist(assay(col_selec))

# Convertimos en matriz y convertimos valores de la mitad inferior en NA
lower_dist <- as.matrix(manDist)
lower_dist[lower.tri((lower_dist))] <- NA

# Generamos un heatmap para visualizar las distancias entre los individuos
heatmap(lower_dist,
        col = heat.colors(16),
```

```
        Rowv = NA,        # No reorganizar filas
        Colv = NA,        # No reorganizar columnas
        scale = "none",   # No escalar
        margins = c(5, 5), # Márgenes para las etiquetas
        cexRow = 0.9,     # Tamaño fuente filas
        cexCol = 0.9)     # Tamaño fuente columnas
```



Los valores altos en la matriz de distancias, representados por colores blancos y amarillos, indican que las observaciones son muy diferentes entre sí. Sin embargo, los valores bajos, representados por colores más rojos, indican que las observaciones son similares entre sí.

Las correlaciones observadas en los datos no normalizados no se observan en los normalizados, lo que indica que la escala de los datos influye en su correlación, y confirma la importancia de utilizar datos normalizados para obtener correlaciones representativas.

Las correlaciones más altas se observan entre el peso y BMI, o peso y tamaño de cintura. También entre la variable HOMA (*Homeostatic Model Assessment*) y la insulina o el nivel de glucosa.

Análisis de correlación

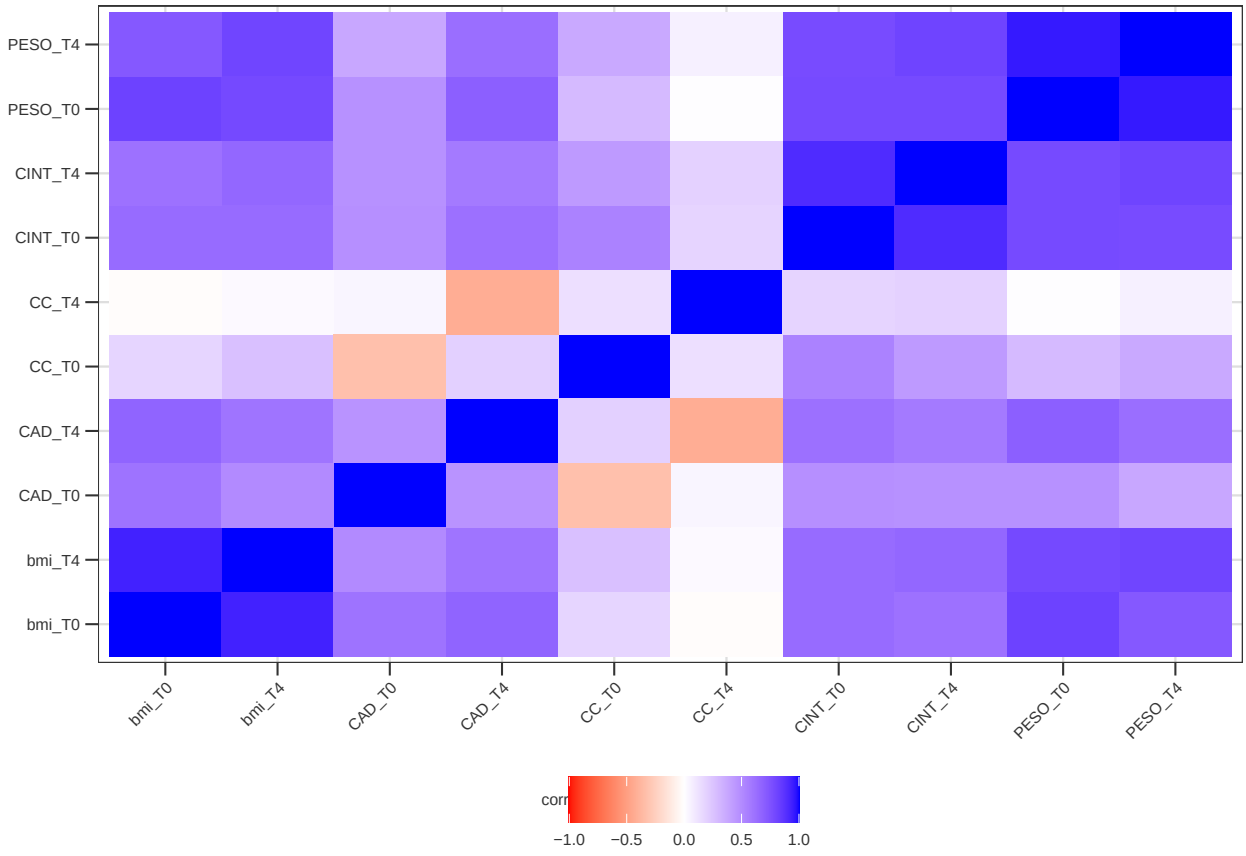
Para entender cómo se relacionan las variables antropométricas y de regulación de glucosa en distintos momentos de recopilación de datos, podemos llevar a cabo un análisis de correlación.

Correlación de Indicadores Antropométricos

Compararemos los valores antes de la cirugía bariátrica (T0) y cuatro meses después de la misma (T4). No se ha comparado la variable T5 ya que se han eliminado muchas columnas debido a la presencia de NAs.

```r
# Seleccionamos las filas con los indicadores de interés
col_selec_antropo <- dataValues_norm[rownames(dataValues_norm) %in% c("PESO_T0", "bmi_T0",
                                                                      "CINT_T0", "CAD_T0","CC_T0", "PESO_T4",
                                                                      "bmi_T4", "CINT_T4", "CAD_T4","CC_T4"), ]

# Realizamos el análisis de correlación
PomaCorr(col_selec_antropo)
```

```
## $correlations
## # A tibble: 45 x 5
##    feature1 feature2  corr   pvalue      FDR
##    <chr>    <chr>    <dbl>    <dbl>    <dbl>
##  1 PESO_T0  PESO_T4  0.954 5.56e-21 2.50e-19
##  2 bmi_T0   bmi_T4   0.931 9.85e-18 2.22e-16
##  3 CINT_T0  CINT_T4  0.901 5.88e-15 8.83e-14
##  4 PESO_T0  bmi_T0   0.815 2.69e-10 3.03e- 9
##  5 PESO_T4  CINT_T4  0.806 6.05e-10 5.45e- 9
##  6 PESO_T4  bmi_T4   0.803 7.87e-10 5.91e- 9
##  7 PESO_T0  bmi_T4   0.785 3.28e- 9 2.11e- 8
##  8 PESO_T0  CINT_T0  0.780 4.69e- 9 2.50e- 8
##  9 PESO_T0  CINT_T4  0.780 5.00e- 9 2.50e- 8
## 10 CINT_T0  PESO_T4  0.775 6.91e- 9 3.11e- 8
## # i 35 more rows
##
## $corrplot
```

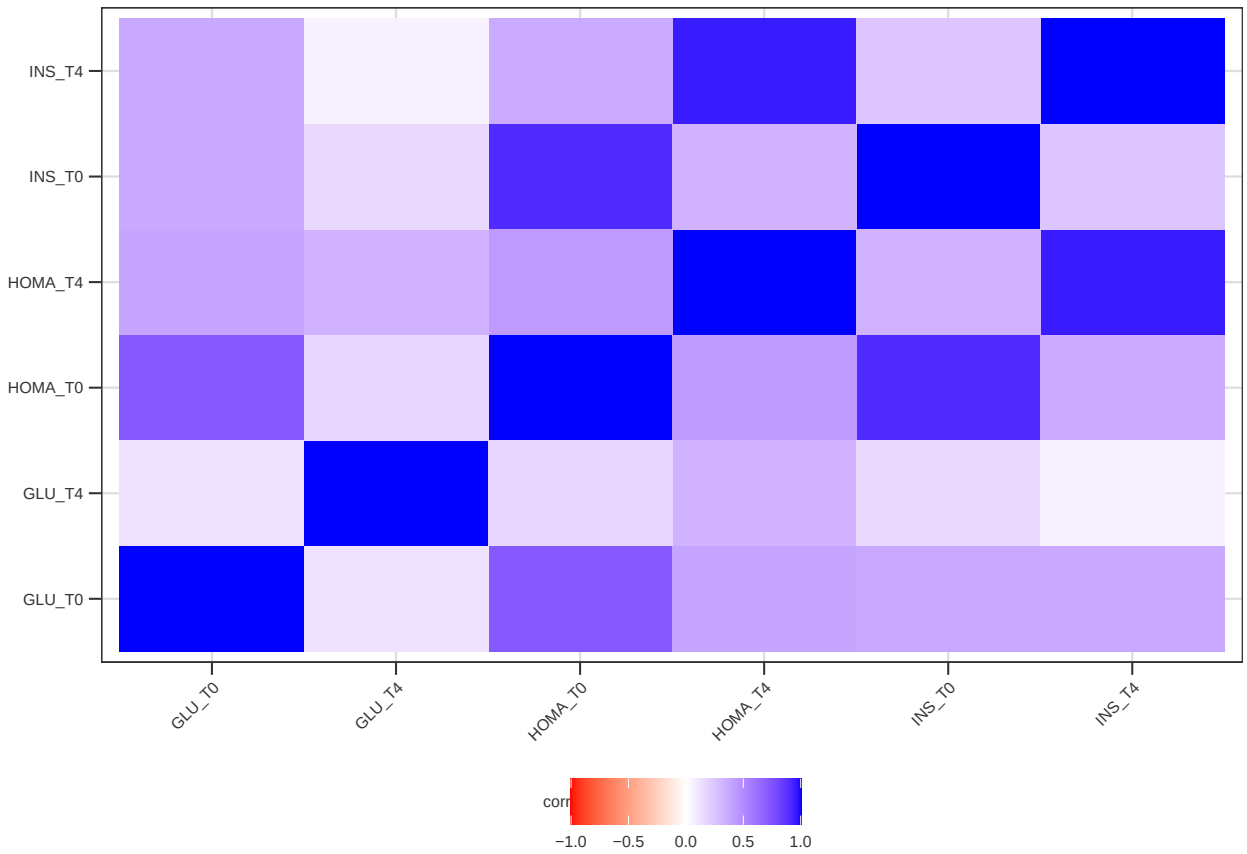


En el gráfico, podemos observar que existe una correlación fuerte para el BMI, tamaño de la cintura y el peso, entre los dos momentos de toma de datos. No es así para el tamaño de la cadera y la relación cintura/cadera.

Correlación de Indicadores de regulación de glucosa

```
col_selec_gluc <- dataValues_norm[rownames(dataValues_norm) %in% c("GLU_T0", "INS_T0",
                                                                   "HOMA_T0", "GLU_T4", "INS_T4", "HOMA_T4"), ]

PomaCorr(col_selec_gluc)
```

```
## $correlations
## # A tibble: 15 x 5
##    feature1 feature2   corr   pvalue      FDR
##    <chr>    <chr>     <dbl>    <dbl>    <dbl>
##  1 INS_T4   HOMA_T4  0.949  3.41e-20 5.11e-19
##  2 INS_T0   HOMA_T0  0.901  5.36e-15 4.02e-14
##  3 GLU_T0   HOMA_T0  0.720  2.37e- 7 1.19e- 6
##  4 HOMA_T0  HOMA_T4  0.437  5.39e- 3 2.02e- 2
##  5 GLU_T0   HOMA_T4  0.388  1.46e- 2 4.38e- 2
##  6 GLU_T0   INS_T4   0.370  2.05e- 2 4.40e- 2
##  7 GLU_T0   INS_T0   0.369  2.09e- 2 4.40e- 2
##  8 HOMA_T0  INS_T4   0.362  2.35e- 2 4.40e- 2
##  9 GLU_T4   HOMA_T4  0.330  4.00e- 2 6.08e- 2
## 10 INS_T0   HOMA_T4  0.329  4.06e- 2 6.08e- 2
## 11 INS_T0   INS_T4   0.244  1.35e- 1 1.84e- 1
## 12 HOMA_T0  GLU_T4   0.174  2.90e- 1 3.62e- 1
## 13 INS_T0   GLU_T4   0.162  3.23e- 1 3.73e- 1
## 14 GLU_T0   GLU_T4   0.122  4.61e- 1 4.94e- 1
## 15 GLU_T4   INS_T4   0.0545 7.42e- 1 7.42e- 1
##
## $corrplot
```



En el gráfico anterior, se observan correlaciones altas entre la variable HOMA y la insulina para el mismo tiempo de recopilación, lo que coincide con los resultados de la matriz de distancias.

Análisis Factorial Múltiple (MFA)

Debido al gran número de variables de nuestro dataset, sería interesante aplicar algunua técnica de reducción de la dimensionalidad. Vamos a estudiar cómo se relaccionan diferentes grupos de variables que tienen en común el momento de recopilación.

Los datos se recopilaron en cuatro momentos distintos, siendo "T0" los datos recopilados antes de la operación bariátrica, y T2, T4 y T5, los datos recopilados 2, 4 y 5 meses después de la operación, respectivamente.

```
# Transponemos los datos para que las variables se dispongan en las columnas
datos_norm <- t(assay(dataValues_norm))

# Contamos número de variables en cada grupo (momento de recopilación de datos)
(count_T0 <- sum(grepl("T0$", colnames(datos_norm))))
```

```
## [1] 164
```

```
(count_T2 <- sum(grepl("T2$", colnames(datos_norm))))
```

```
## [1] 167
```

```r
(count_T4 <- sum(grepl("T4$", colnames(datos_norm))))
```

```
## [1] 166
```

```r
(count_T5 <- sum(grepl("T5$", colnames(datos_norm))))
```

```
## [1] 16
```

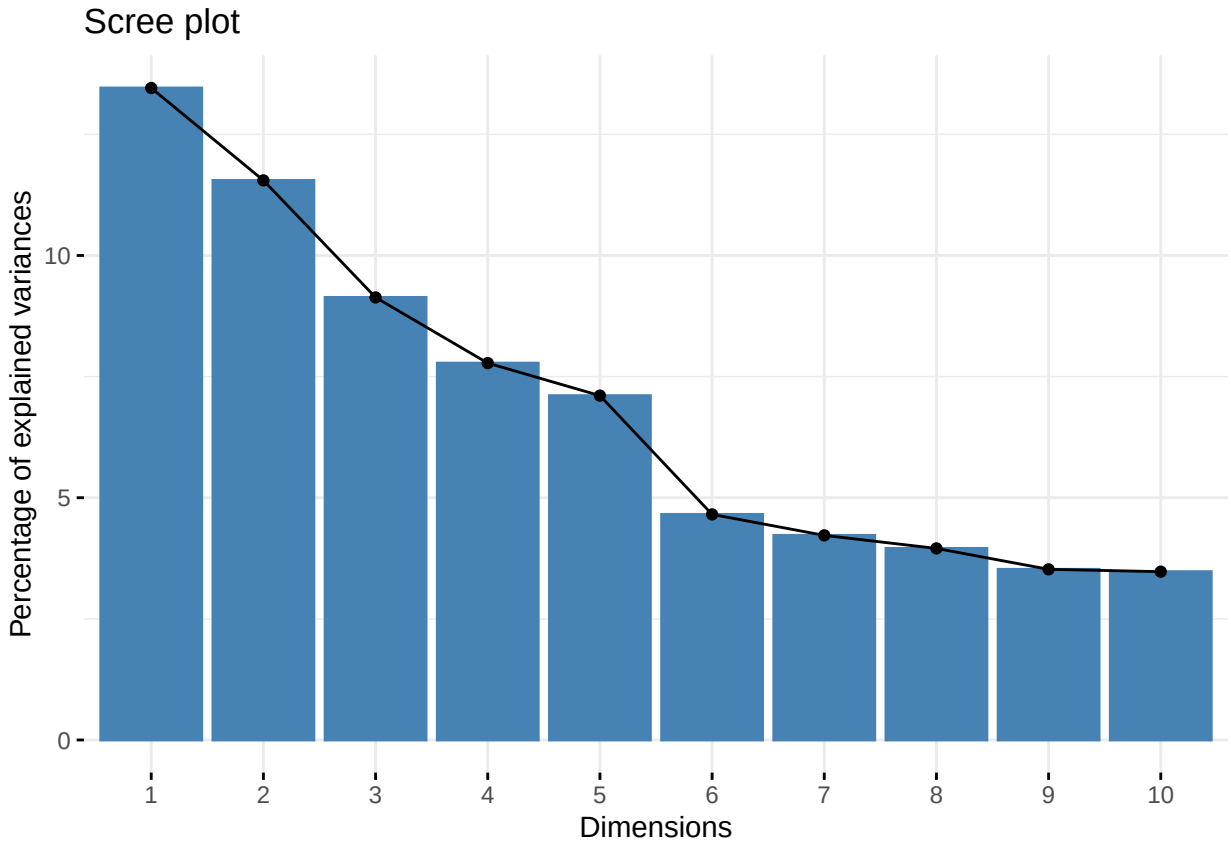
```r
# Formamos los grupos de acuerdo al numero de variables
groups <- c(164, 167, 166, 16)

# Realizamos el Análisis Factorial Múltiple
mfa_results <- MFA(datos_norm,
                   group = groups,
                   name.group = c("T0", "T2", "T4", "T5"),
                   graph = FALSE)
```

Calculamos la proporción de la varianza explicada por cada cada dimensión y generamos un *scree plot* para visualizarlo. Observamos que las primeras 5 dimensiones explican la mayor parte de la varianza, alcanzando un total del 53%. A partir de la quinta dimensión, la línea se vuelve más plana, lo que indica que cada dimensión aporta cada vez menos a la explicación de la varianza.

```r
# Varianzas
eig.val <- get_eigenvalue(mfa_results)
head(eig.val)
```

```
##       eigenvalue variance.percent cumulative.variance.percent
## Dim.1  1.9887914        13.456859                    13.45686
## Dim.2  1.7069846        11.550055                    25.00691
## Dim.3  1.3498810         9.133767                    34.14068
## Dim.4  1.1495921         7.778543                    41.91922
## Dim.5  1.0503442         7.106997                    49.02622
## Dim.6  0.6881068         4.655972                    53.68219
```

```r
# Scree plot
fviz_screeplot(mfa_results)
```



En la siguiente tabla y gráfica, podemos ver cómo los distintos grupos contribuyen a las dimensiones. Observamos que el grupo que contribuye en mayor medida a la primera dimensión es el T4, y a la segunda dimensión, el T5. Esto coincide con los resultados del artículo, que indica que el impacto de la cirugía bariátrica es tan intenso que las diferencias metabólicas iniciales se anulan.

```
# Grupos de variables
group <- get_mfa_var(mfa_results, "group")

# Contribuciones a las dimensiones
head(group$contrib)
```

```
##       Dim.1    Dim.2     Dim.3    Dim.4    Dim.5
## T0 25.08103 23.38120 33.989036 18.53526 17.79435
## T2 15.14404 22.56404 49.446961 21.67836 40.20792
## T4 31.61890 19.25892  8.210009 13.73504 17.56777
## T5 28.15603 34.79584  8.353995 46.05134 24.42995
```

```
# Visualización de contribuciones por grupo
fviz_mfa_var(mfa_results, "group")
```

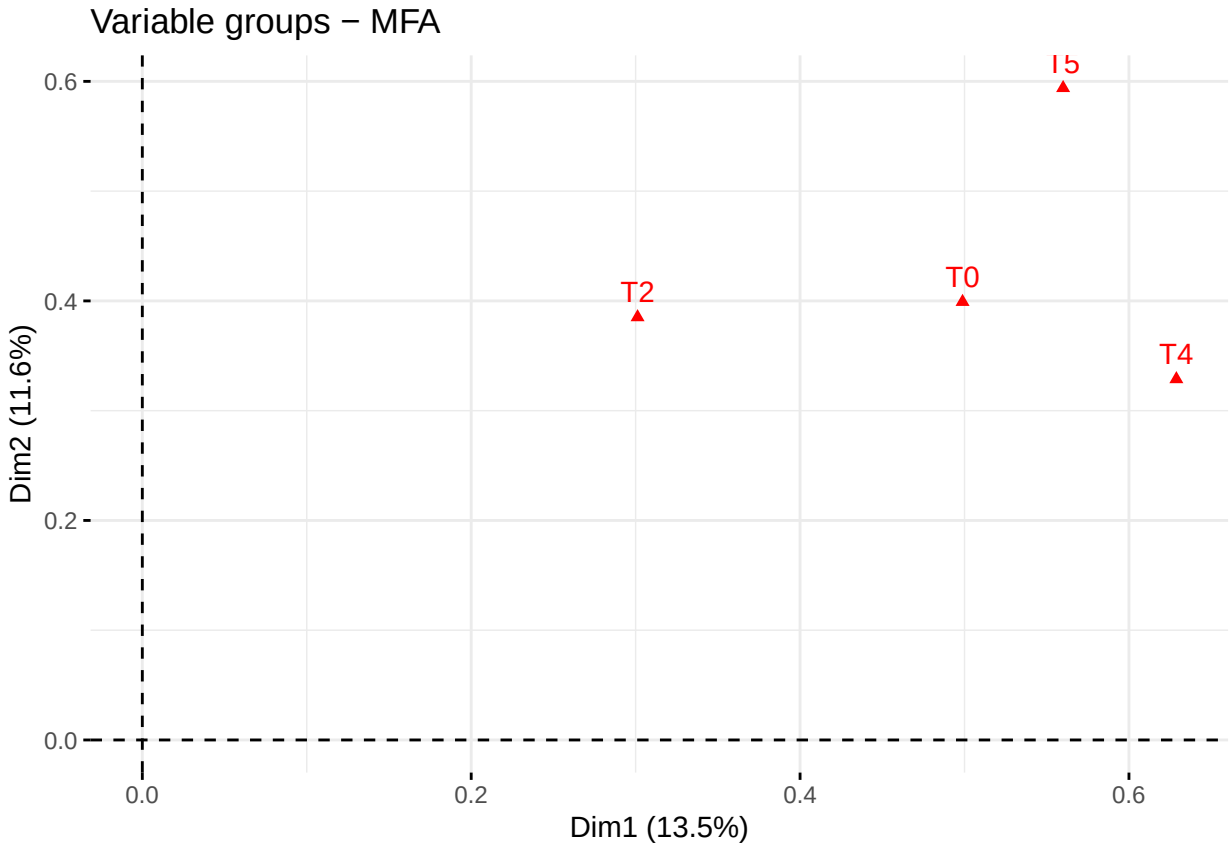
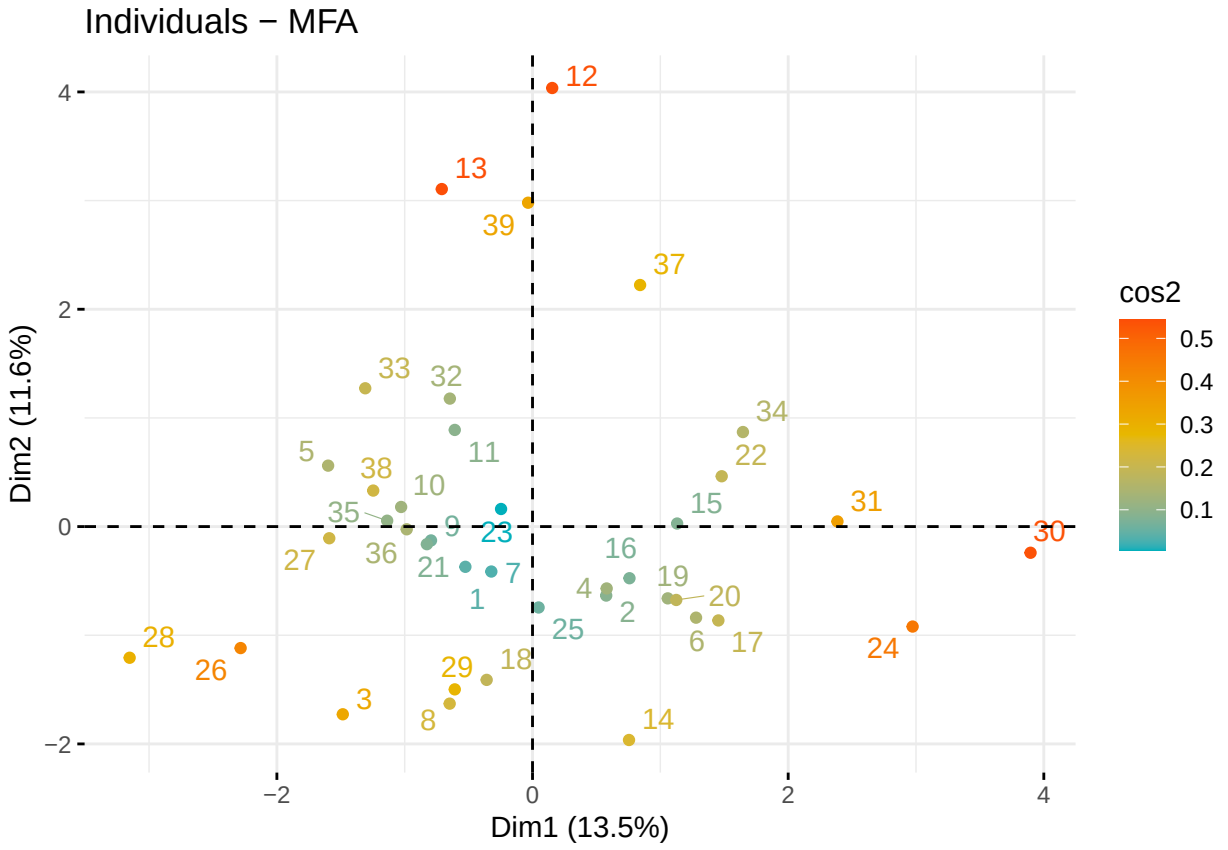


Gráfico de los perfiles de los individuos

A continuación, representaremos cómo se distribuyen los individuos en el espacio de las dimensiones principales, según las variables de los diferentes grupos (T0, T2, T4, T5), y sus contribuciones a las dimensiones principales.

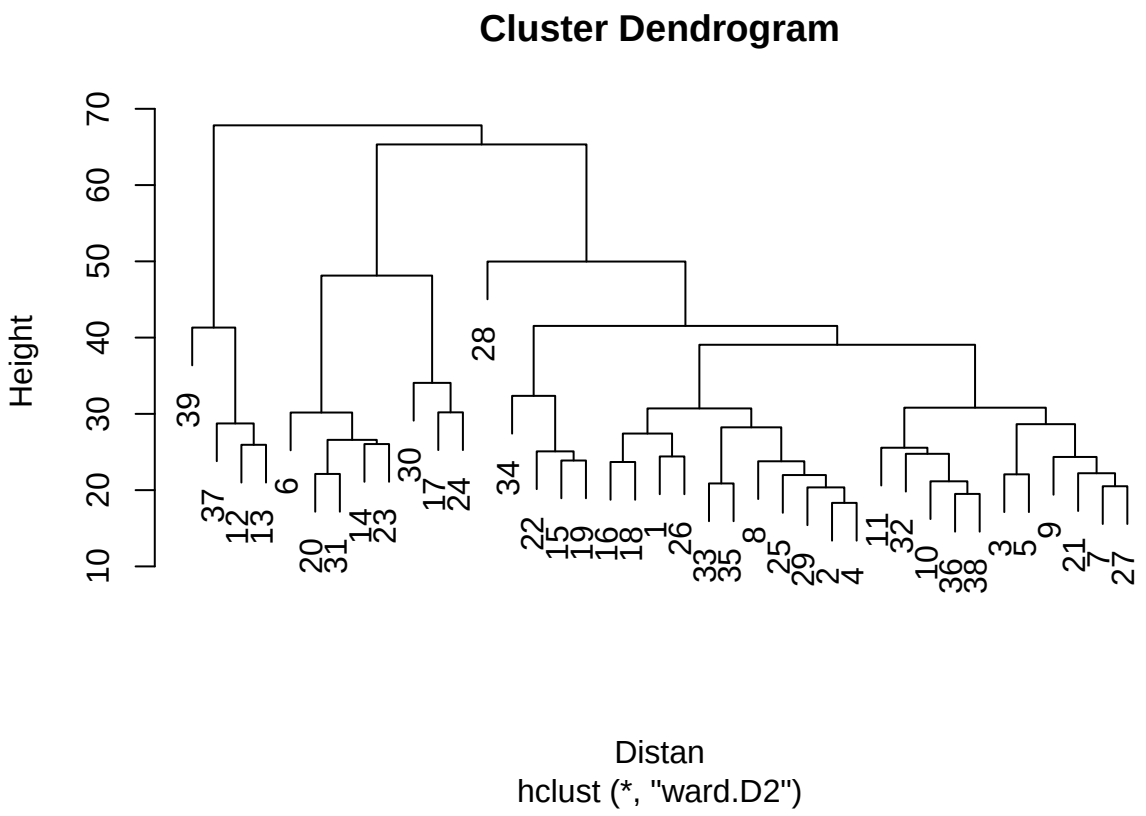
```
# Gráfico de los perfiles de los individuos
fviz_mfa_ind(mfa_results,
             col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE)
```



Agrupación jerárquica de las muestras

Por último, haremos una aproximación para establecer la relación entre las muestras a través de un método de agrupamiento jerárquico. Este método permite identificar patrones de similitud entre las muestras basándose en sus características. Representaremos la estructura jerárquica del agrupamiento con un dendograma. Podemos modificar el método de la función *hclust* para obtener un agrupamiento que se adapte a nuestros datos.

```
assay_norm <- assay(dataValues_norm)

# Calculamos matriz de distancias utilizando el método euclidean
Distan <- dist(t(assay_norm), method="euclidean")

# Realizamos el agrupamiento jerárquico utilizando el método ward.D2
hc <- hclust(Distan, method="ward.D2")

# Representamos la estructura jerárquica
plot(hc)
```



Agrupación por el método de k-means

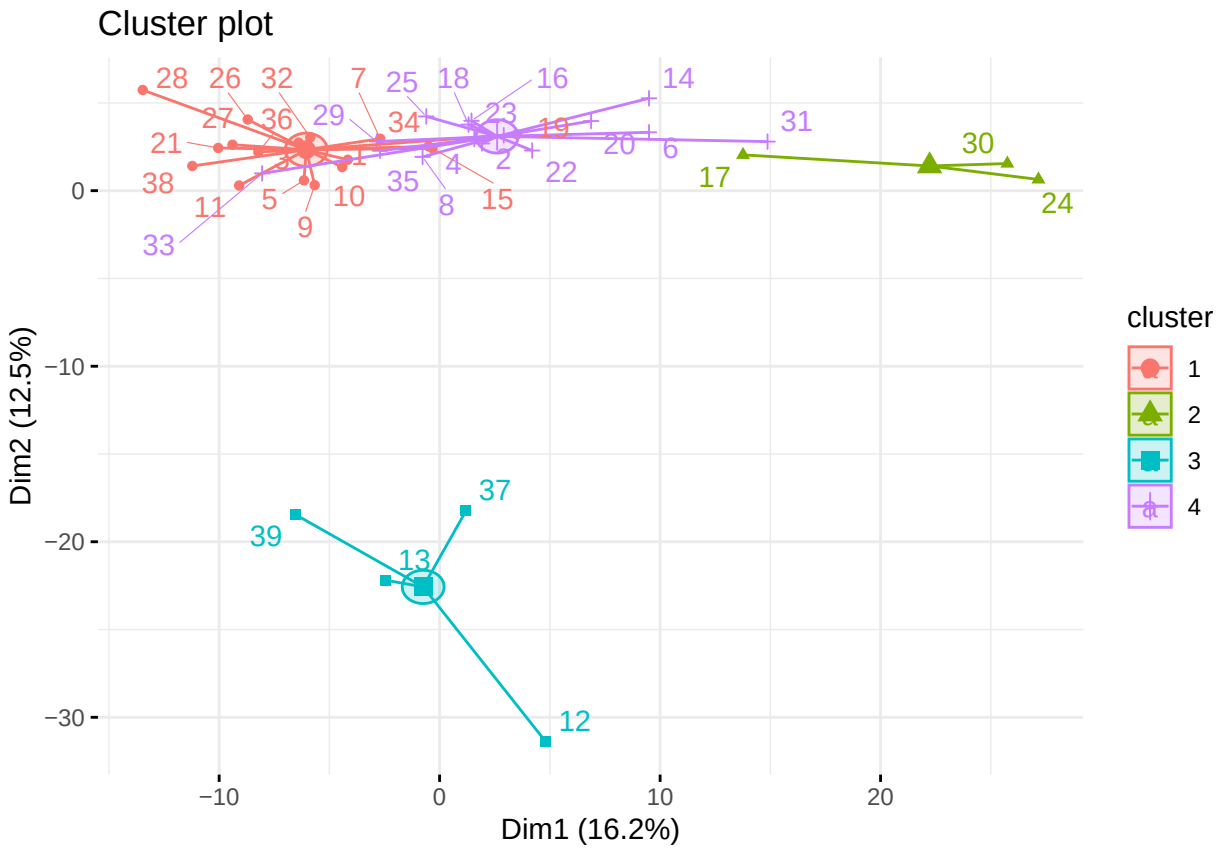
Utilizaremos también el método de k-means, una técnica de agrupamiento no jerárquico, para dividir las muestras en un número específico de grupos, basándose en sus características.

```
# Definimos el número de clusters
k <- c(4)

# Realizamos el agrupamiento k-means
km <- kmeans(datos_norm, k, iter.max=1000)

# Calculamos la media de la suma de cuadrados dentro de los grupos
mean(km$withinss)
```

```
## [1] 3423.937
```

```
# Visualizamos resultados del agrupamiento
fviz_cluster(km, data = datos_norm,
ellipse.type = "euclid",
star.plot = TRUE,
repel = TRUE,
ggtheme = theme_minimal()
)
```

```
## Too few points to calculate an ellipse
```



Creación del archivo con los metadatos

Guardamos los metadatos del estudio en un archivo markdown.

```
archivo <- "metadatos.md"

titulo <- c("# Metadatos del Estudio", "")

# Inicializamos el vector para almacenar los metadatos formateados
output <- c()

# Iteramos sobre los nombres de los campos en la lista de metadata
for(name in names(metadata)){
  output <- c(output, paste0("- **", name, "**: ", metadata[[name]]))
}

# Escribimos título y metadatos en el archivo de salida
writeLines(c(titulo, output), archivo)
```

Reposición de datos en github

En primer lugar, creamos un repositorio de GitHub con el nombre indicado ("Gonzalez-Palomo-Adriana-PEC1), y lo hacemos público. La dirección del repositorio es la siguiente:

https://github.com/adrianagpal/Gonzalez-Palomo-Adriana-PEC1.git

A continuación, inicializamos el repositorio localmente:

git init

Añadimos los archivos con *git add*, por ejemplo:

git add metadatos.md

Una vez añadidos, realizamos un *git commit* con un mensaje descriptivo de lo que estamos subiendo al repositorio. Finalmente, realizamos un *git push*. La primera vez que subimos archivos al repositorio, sin embargo, es necesario realizar:

git push -u origin main

Referencias

Morgan, Martin, Valerie Obenchain, Jim Hester, and Hervé Pagès. 2020. SummarizedExperiment: SummarizedExperiment Container. https://bioconductor.org/packages/SummarizedExperiment.

Castellano-Escuder, Pol. 2024. Get Started: POMA. https://www.bioconductor.org/packages/release/bioc/vignettes/POMA/inst/doc/POMA-workflow.html

Castellano-Escuder, Pol. 2022. POMA Workflow. http://bioconductor.jp/packages/3.16/bioc/vignettes/POMA/inst/doc/POMA-demo.html

Sánchez, A., Carmona, F. 2024. Casos y Ejemplos de Análisis Multivariante con R. https://aspteaching.github.io/AMVCasos/

Palau-Rodriguez M, Tulipani S, Marco-Ramell A, Miñarro A, Jáuregui O, Sanchez-Pla A, et al. (2018) Metabotypes of response to bariatric surgery independent of the magnitude of weight loss. PLoS ONE 13(6): e0198214. https://doi.org/10.1371/journal.pone.0198214


```
sessionInfo()
```

```
## R version 4.4.1 (2024-06-14 ucrt)
## Platform: x86_64-w64-mingw32/x64
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_Europe.utf8  LC_CTYPE=English_Europe.utf8
## [3] LC_MONETARY=English_Europe.utf8 LC_NUMERIC=C
## [5] LC_TIME=English_Europe.utf8
##
## time zone: Europe/Madrid
## tzcode source: internal
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] ggtext_0.1.2                tidyr_1.3.1
```

```
##  [3] dplyr_1.1.4                 factoextra_1.0.7
##  [5] ggplot2_3.5.1               FactoMineR_2.11
##  [7] SummarizedExperiment_1.34.0 Biobase_2.64.0
##  [9] GenomicRanges_1.56.2        GenomeInfoDb_1.40.1
## [11] IRanges_2.38.1              S4Vectors_0.42.1
## [13] BiocGenerics_0.50.0         MatrixGenerics_1.16.0
## [15] matrixStats_1.4.1           POMA_1.14.0
## [17] BiocManager_1.30.25
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.1        farver_2.1.2            fastmap_1.2.0
##  [4] digest_0.6.37           estimability_1.5.1      lifecycle_1.0.4
##  [7] cluster_2.1.6           multcompView_0.1-10     magrittr_2.0.3
## [10] compiler_4.4.1          rlang_1.1.4             tools_4.4.1
## [13] utf8_1.2.4              yaml_2.3.10             knitr_1.48
## [16] ggsignif_0.6.4          S4Arrays_1.4.1          labeling_0.4.3
## [19] htmlwidgets_1.6.4       scatterplot3d_0.3-44    DelayedArray_0.30.1
## [22] xml2_1.3.6              abind_1.4-8             withr_3.0.2
## [25] purrr_1.0.2             grid_4.4.1              fansi_1.0.6
## [28] ggpubr_0.6.0            xtable_1.8-4            colorspace_2.1-1
## [31] emmeans_1.10.5          scales_1.3.0            MASS_7.3-61
## [34] flashClust_1.01-2       tinytex_0.53            cli_3.6.3
## [37] mvtnorm_1.3-1           rmarkdown_2.28          crayon_1.5.3
## [40] generics_0.1.3          rstudioapi_0.17.1       httr_1.4.7
## [43] zlibbioc_1.50.0         impute_1.78.0           XVector_0.44.0
## [46] vctrs_0.6.5             Matrix_1.7-1            carData_3.0-5
## [49] jsonlite_1.8.9          car_3.1-3               rstatix_0.7.2
## [52] ggrepel_0.9.6           Formula_1.2-5           glue_1.8.0
## [55] DT_0.33                 gtable_0.3.6            UCSC.utils_1.0.0
## [58] munsell_0.5.1           tibble_3.2.1            pillar_1.9.0
## [61] htmltools_0.5.8.1       GenomeInfoDbData_1.2.12 R6_2.5.1
## [64] evaluate_1.0.1          lattice_0.22-6          highr_0.11
## [67] backports_1.5.0         leaps_3.2               gridtext_0.1.5
## [70] broom_1.0.7             Rcpp_1.0.13             SparseArray_1.4.8
## [73] xfun_0.48               pkgconfig_2.0.3
```