Prueva de Evaluación Continua

Análisis de Datos Ómicos

Carmen Laura Frias Pérez

Contents

Introducción

Esta PEC consiste en implementar los conocimientos vistos hasta ahora en la asignatura. Para ello, he seleccionado un dataset de *metabolómica* del repositorio de *GitHub* proporcionado (https://github.com/nutrimetabolomics/metaboData/tree/main/Datasets), he creado un contenedor del tipo *SummarizedExperiment* con los datos y metadatos asociados, he hecho un análisis exploratorio del mismo y he creado un repositorio en *GitHub* para publicar los documentos generados a partir de esta entrega.

Abstract

Para llevar a cabo esta entrega, he seleccionado el dataset *human_cachexia.csv* y, a partir de este, he creado el contenedor *SummarizedExperiment* con toda la información necesaria (matriz numérica, colData, rowData y metadata).

Una vez obtenida, he hecho un estudio exploratorio descriptivo, mirando la distribución de los datos en general y en función del grupo, además de representaciones gráficas mediante *boxplots*. También he hecho una representación con Análisis de Componentes Principales (CPA), una representación de la matriz de correlación y dos clustering jerárquicos, uno en función del identificador del paciente y otro en función del metabolito.

Los principales resultados obtenidos han sido que el contenedor *SummarizedExperiment* está compuesto por 63 filas (metabolitos) y 77 columnas (identificador del paciente), la matriz numérica y la metadata asociada. Además, se ha visto que en el contenedor hay diferentes concentraciones de metabolitos y que, al mirar las representaciones de sus valores dependiendo del grupo (control o caquexia), en general, el grupo de pacientes con caquexia tiene valores más elevados de las concentraciones de estos metabolitos. Al hacer el CPA, se ha visto que hay situaciones en las que se solapan ambos grupos, pero que el grupo de pacientes con caquexia tiene una dispersión mayor de sus datos. En la matriz de correlación se ve también que hay correlaciones mayores o menores en función de los metabolitos que se correlacionan, pudiendo indicar que están más o menos relacionados en cuanto a sus vías metabólicas. Además, en ambos clustering jerárquicos, por grupos y por metabolitos, se observan dos agrupaciones diferenciadas. En el primero, puede deberse a que las muestras compartan perfiles similares de metabolitos, y en el segundo a que los metabolitos tengan vías metabólicas relacionadas.

Objetivos del estudio

Los objetivos del estudio son, a partir del dataset seleccionado, crear un contenedor del tipo *SummarizedExperiment* y, posteriormente, hacer un análisis exploratorio de los datos para ver como se distribuyen estos y colgar los documentos generados y empleados en un repositorio de *GitHub*.

Materiales y Métodos

El dataset que he seleccionado para desarrollar esta entrega es el que se encuentra en la carpeta **2024-Cachexia**. En esta carpeta, a parte del dataset, *human_cachexia.csv*, se encuentra otro documento llamado *description.md* que contiene información general sobre el dataset. En este, dice que el dataset ha sido utilizado en varios tutoriales de MetaboAnalyst, que las muestras del mismo no están emparejadas, que el conjunto de datos contiene dos grupos de muestras (caquexia y control), que todos todos los valores de los datos son numéricos y que no se han detectado valores faltantes (missing values: NaN) en los datos recogidos.

El dataset *human_cachexia.csv* contiene información sobre mediciones de metabolitos en diferentes pacientes identificados, además de información acerca de si estos pacientes presentan caquexia o no, formando parte del grupo control.

Una vez seleccionado el dataset deseado, planteo crear el contenedor *SummarizedExperiment* teniendo en cuenta las diferentes partes que debe tener (matriz numérica, colData, rowData y metadata). Antes de comenzar a hacerlo, observo la estructura general del dataset.

Cuando ya he creado el contenedor, hago un análisis exploratorio de los datos que se basa en:

- Visualización de los Metabolitos: representación de los datos en una tabla sencilla, estudio de sus dimensiones y extracción de los nombres de las columnas y las filas del mismo, además de exploración de su estructura.
- Estadísticas Descriptivas de los Metabolitos: cálculo de las estadísticas descriptivas para cada metabolito y tabla representativa.
- Boxplot de Metabolitos: representación de los valores de los metabolitos modificados mediante la función logarítmica.
- Conteo de Pacientes por Grupo: determinar la cantidad de pacientes que hay dependiendo del grupo.
- Boxplot Comparativo por Grupos: representación de los diferentes valores de las concentraciones de metabolitos que hay por grupos. Impresión de algunos de los *boxplots*.
- Análisis de Componentes Principales (CPA): estudiar la diferencia en la distribución de los grupos usando esta técnica.
- Matriz de correlación: cálculo de la matriz de covarianza y correlación para hacer un *heatmap* de los mismos.
- Clustering Jerárquico: en función del identificador del paciente y de los metabolitos usando el método de clustering completo y cálculando la distancia de la matriz con el método euclideano.

Una vez hecho el estudio exploratorio de los datos del contenedor y de haber comprobado todo lo que se ha realizado, genero el repositorio de *GitHub* y cuelgo el informe, los datos del dataset y del contenedor *SummarizedExperiment*, el código y los metadatos del contenedor.

Resultados

Preprocesamiento del Dataset

Una vez descargado el dataset seleccionado, he introducido todas las librerías necesarias para el desarrollo de esta entrega. Luego, he cargado el dataset *human_cachexia.csv* y he observado las primeras filas contenidas en el documento para observar su distribución y comprobar que se ha cargado correctamente.

Para introducir el dataset he usado la función *read.csv()*, que permite leer el archivo y convertirlo en un dataframe.

```
# Introducción del dataset human_cachexia.csv
data_cachexia <- read.csv("human_cachexia.csv")
```

Para visuallizar las primeras filas contenidas en el dataset, he usado la función *head()*, que permite ver los datos contenidos en las primeras 5 filas del documento para comprender mejor la distribución del mismo.

```
# Visualización de las primeras filas del dataset
head(data_cachexia[1:5, 1:5])
```

```
##    Patient.ID Muscle.loss X1.6.Anhydro.beta.D.glucose X1.Methylnicotinamide
## 1     PIF_178    cachexic                       40.85                 65.37
## 2     PIF_087    cachexic                       62.18                340.36
## 3     PIF_090    cachexic                      270.43                 64.72
## 4 NETL_005_V1    cachexic                      154.47                 52.98
```

```
## 5     PIF_115    cachexic                      22.20                 73.70
##   X2.Aminobutyrate
## 1            18.73
## 2            24.29
## 3            12.18
## 4           172.43
## 5            15.64
```

Para facilitar la identificación de las columnas durante el análisis, las he renombrado. Así, he conseguido eliminar los espacios en blanco que había, consiguiendo hacer un desarrollo más sencillo del código a utilizar. Para asignar el nombre a las columnas, he usado la función *colnames()*.

```
# Definición de los nuevos nombres para las columnas del dataset
columnas_reescritas <- c("Patient_ID", "Muscle_loss", "Anhydro_beta_D_glucose",
                         "Methylnicotinamide", "Aminobutyrate", "Hydroxyisobutyrate",
                         "Oxoglutarate", "Aminoisobutyrate", "Hydroxybutyrate",
                         "Hydroxyisovalerate", "Indoxylsulfate", "Hydroxyphenylacetate",
                         "Acetate", "Acetone", "Adipate", "Alanine", "Asparagine",
                         "Betaine", "Carnitine", "Citrate", "Creatine", "Creatinine",
                         "Dimethylamine", "Ethanolamine","Formate", "Fucose", "Fumarate",
                         "Glucose", "Glutamine", "Glycine","Glycolate", "Guanidoacetate",
                         "Hippurate", "Histidine", "Hypoxanthine", "Isoleucine",
                         "Lactate", "Leucine", "Lysine", "Methylamine", "Methylguanidine",
                         "N_N_Dimethylglycine", "O_Acetylcarnitine", "Pantothenate",
                         "Pyroglutamate", "Pyruvate", "Quinolinate", "Serine", "Succinate",
                         "Sucrose", "Tartrate", "Taurine", "Threonine", "Trigonelline",
                         "Trimethylamine_N_oxide", "Tryptophan", "Tyrosine",
                         "Uracil", "Valine", "Xylose", "cis_Aconitate", "myo_Inositol",
                         "trans_Aconitate", "pi_Methylhistidine", "tau_Methylhistidine")

# Asignación de los nuevos nombres a las columnas del dataset
colnames(data_cachexia) <- columnas_reescritas
```

A partir de aquí, ya puedo proceder a realizar la extracción de datos del dataset para crear el contenedor *SummarizedExperiment*.

Creación del Contenedor: SummarizedExperiment

El contenedor *SummarizedExperiment* permite almacenar los datos en forma de matrices con la información numérica del dataset, además de información sobre las columnas, las filas y la metadata asociada. El uso de este tipo de contenedores facilita el manejo de datos biológicos y, en este caso, de los metabolitos.

Antes de generar el contenedor *SummarizedExperiment*, he extraído las columnas del dataset que contienen la información numérica de los metabolitos y les he hecho una trasposición. Así, al trasponer, quedan las filas y las columnas invertidas, con los nombres de los metabolitos en las filas y los identificadores de los pacientes en las columnas.

```
# Selección de las columnas numéricas para extraer las concentraciones de los metabolitos
columnas_matriz_numerica <- 3:ncol(data_cachexia)
data_matriz_numerica <- as.matrix(data_cachexia[,columnas_matriz_numerica])
```

La función empleada *as.matrix()* permite convertir el dataframe a una matriz. Esto es fundamental, ya que *SummarizedExperiment* trabaja con esta matriz generada.

```
# Eliminación del nombre de las columnas para que la matriz sea, únicamente, numérica
colnames(data_matriz_numerica) <- NULL

# trasposición de los datos para que las columnas contengan el identificador de los
# pacientes, mientras que las filas el nombre de los metabolitos
data_matriz_numerica_traspuesta <- t(data_matriz_numerica)
```

La función *t()* es útil para llevar a cabo la trasposición de los datos de la matriz.

Para extraer el nombre de las filas y las columnas he usado la matriz traspuesta generada previamente que, como vimos, los identificadores de los pacientes se encuentran en las columnas y el nombre de los metabolitos en las filas.

```
# Extracción del nombre de las columnas
colnames(data_matriz_numerica_traspuesta) <- data_cachexia$Patient_ID

# Extracción del nombre de las filas
rownames(data_matriz_numerica_traspuesta) <- colnames(data_cachexia)[columnas_matriz_numerica]
```

La extracción de esta información, facilitará posteriormente, en el *SummarizedExperiment*, nombrar las columnas y filas, que formarán parte del mismo. Así mismo, se pueden tener en cuenta al hacer el análisis exploratorio de los datos.

Para llevar a cabo la extracción de colData he creado un DataFrame que almacena la información de los pacientes que no es numérica. Así, he extraído la información referente al identificador de los pacientes (*Patient_ID*) y del nivel de pérdida muscular que tienen los pacientes (*Muscle_loss*, grupos: control y caquexia).

```
# Extracción de la información de los pacientes que no es numérica
colData_cachexia <- data_cachexia[ , c("Patient_ID", "Muscle_loss")]

# Conversión de colData a DataFrame
colData_cachexia <- DataFrame(colData_cachexia)
```

El uso de la función *DataFrame()* permite crear un contenedor que almacena la información que se ha introducido. Esta información es información no numérica, como el identificador del paciente y la asignación del grupo correspondiente.

La información acerca de los metabolitos la he almacenado en el rowData, incluyendo los nombres de los mismos, que fueron extraídos previamente. Así, he creado un DataFrame teniendo en cuenta los nombres de las filas, es decir, de los metabolitos, teniendo en cuenta la matriz traspuesta. El hecho de almacenar la información de los nombres de los metabolitos facilitará la identificación de los mismos en los análisis exploratorios que realizaré más adelante.

```
# Creación de DataFrame para filas
rowData_cachexia <- DataFrame(
  Metabolitos =rownames(data_matriz_numerica_traspuesta),
  seq_len(nrow(data_matriz_numerica_traspuesta)))
```

Los metadatos son la información adicional que va asociada a la estructura que presenta el *SummarizedExperiment*. Estos incluyen, en este caso, una descripción general de la distribución de los datos e información sobre lo que contienen las filas y columnas del mismo.

```r
# Creación de una lista con los metadatos del estudio
metadata_cachexia <- list(
  Datos_Dataset = list(
    Nombre_Dataset = "human_cachexia.csv",
    Fuente = "https://github.com/nutrimetabolomics/metaboData/tree/main/Datasets",
    Descripcion = "El conjunto de datos contiene información
    de la concentración de 63 metabolitos en 77 pacientes y
    se diferencia a los pacientes que tienen caquexia de los
    controles.",
    Fecha_Modificacion = Sys.Date()
  ),

  Datos_Filas = list(
    Descripcion = "En las filas de este SummarizedExperiment se recoge la información
    sobre los nombres de los metabolitos que se han medido.",
    Cantidad_Filas = "63",
    Nombre_Filas = c("Anhidro_beta_D_glucosa", "Metilnicotinamida", "Aminobutirato",
                     "Hidroxibutirato", "Oxoglutarato", "Aminoisobutirato",
                     "Hidroxibutirato","Hidroxisisobutirato", "Indoxilsulfato",
                     "Hidroxifenilacetato", "Acetato", "Acetona", "Adipato",
                     "Alanina", "Asparagina", "Betaína", "Carnitina",
                     "Citrato", "Creatina", "Creatinina", "Dimetilamina",
                     "Ethanolamina", "Formiato", "Fucosa", "Fumarato",
                     "Glucosa", "Glutamina", "Glicina", "Glicolato",
                     "Guanidoacetato", "Hippurato", "Histidina", "Hipoxantina",
                     "Isoleucina", "Lactato", "Leucina", "Lisina",
                     "Metilamina", "Metilguanidina", "N,N-Dimetilglicina",
                     "O-Acetilcarnitina", "Pantotenato", "Piruvato",
                     "Quinolinato", "Serina", "Succcinato", "Sacarosa",
                     "Tartrato", "Taurina", "Treonina", "Trigonelina",
                     "Trimetilamina_N_óxido", "Triptofano", "Tirosina", "Uracilo",
                     "Valina", "Xilosa","cis-Aconitato", "myo-Inositol",
                     "trans-Aconitato", "pi_Metilhistidina", "tau_Metilhistidina")
  ),

  Datos_Columnas = list(
    Descripcion = "En las columnas de este SummarizedExperiment se puede visualizar
    información acerca del identificador de los pacientes y de si
    los pacientes presentan pérdida muscular o caquexia o son parte del grupo normal.",
    Cantidad_Columnas = "77",
    Nombre_Columnas = c("Pacient_ID", "Muscle_loss"),
    ID_Pacientes = c("PIF_178", "PIF_087", "NETL_003_V1", "NETL_005_V1", "PIF_115",
                     "PIF_110", "NETL_019_V1", "NETCR_014_V1","NETCR_014_V2", "PIF_154",
                     "NETL_022_V1", "NETL_022_V2", "NETL_008_V1", "PIF_146", "PIF_119",
                     "PIF_099", "PIF_162", "PIF_160", "PIF_113", "PIF_143", "NETCR_007_V1",
                     "NETCR_007_V2", "PIF_137", "PIF_100", "NETL_004_V1", "PIF_094",
                     "PIF_132", "PIF_163", "NETCR_003_V1", "NETL_028_V1",
                     "NETL_028_V2", "NETCR_013_V1", "NETL_020_V1", "NETL_020_V2",
                     "PIF_192", "NETCR_012_V1", "NETCR_012_V2", "PIF_089", "NETCR_002_V1",
                     "PIF_179", "PIF_114", "NETCR_006_V1", "PIF_141", "NETCR_025_V1",
                     "NETCR_025_V2", "NETCR_016_V1", "PIF_116", "PIF_191", "PIF_164",
                     "NETL_013_V1", "PIF_188", "PIF_195", "NETCR_015_V1", "PIF_102",
                     "NETL_010_V1", "NETL_010_V2", "NETL_001_V1","NETCR_015_V2",
```

```
                    "NETCR_005_V1", "PIF_111", "PIF_171", "NETCR_008_V1", "NETCR_008_V2",
                    "NETL_017_V1", "NETL_017_V2","NETL_002_V1", "NETL_002_V2",
                    "PIF_190", "NETCR_009_V1","NETCR_009_V2", "NETL_007_V1", "PIF_112",
                    "NETCR_019_V2", "NETL_012_V1", "NETL_012_V2", "NETL_003_V1",
                    "NETL_003_V2")
  )
)
```

Por lo tanto, los metadatos permiten almacenar información fundamental para comprender el origen y la distribución de los datos, así como sus características.

Después de haber extraído y generado toda la información anterior, puedo proceder a almacenar esta información en un contenedor *SummarizedExperiment*.

El *SummarizedExperiment* sirve para agrupar la matriz de datos numérica teniendo en cuenta la información no numérica (*Patient_ID, Muscle_loss*), los datos acerca de los nombres de las filas (contenidos en rowData y rownames) y columnas (contenidos en colData y colnames).

```
# Creación del objeto SummarizedExperiment
SummarizedExp_cachexia <- SummarizedExperiment(
  assays = list(counts = data_matriz_numerica_traspuesta),
  colData = colData_cachexia,
  rowData = rowData_cachexia,
  metadata = metadata_cachexia)

# Visualización del contenedor creado para confirmar que se ha generado correctamente
print(SummarizedExp_cachexia)
```

```
## class: SummarizedExperiment
## dim: 63 77
## metadata(3): Datos_Dataset Datos_Filas Datos_Columnas
## assays(1): counts
## rownames(63): Anhydro_beta_D_glucose Methylnicotinamide ...
##   pi_Methylhistidine tau_Methylhistidine
## rowData names(2): Metabolitos
##   seq_len.nrow.data_matriz_numerica_traspuesta..
## colnames(77): PIF_178 PIF_087 ... NETL_003_V1 NETL_003_V2
## colData names(2): Patient_ID Muscle_loss
```

El uso de la función *print()* permite ver los diferentes componentes que forman parte del *SummarizedExperiment*.

Una vez tengo el *SummarizedExperiment* creado y con los datos correctamente estructurados, procedo a realizar el análisis exploratorio de los datos.

Exploración de los Datos

En esta parte de la entrega, estudio el contenedor *SummarizedExperiment* mediante diferentes técnicas para observar cómo se distribuyen los datos en general y en función de los grupos definidos, tanto en función de los metabolitos empleados como de manera individual, según el identificador de los pacientes.

Visualización de los Metabolitos

Para comenzar con el análisis exploratorio, lo primero que hago es visualizar la estructura de los datos. Usando la función *kable()* se puede generar una tabla que permite inspeccionar las primeras filas contenidas en el assay de *SummarizedExperiment*. En este caso, he seleccionado que se imprima una tabla con los valores que forman parte de las primeras 10 filas y las primeras 5 columnas.

```
# Visualización de la estructuración de los datos del SummarizedExperiment
kableExtra::kable(assay(SummarizedExp_cachexia)[1:10,1:5])
```

	PIF_178	PIF_087	PIF_090	NETL_005_V1	PIF_115
Anhydro_beta_D_glucose	40.85	62.18	270.43	154.47	22.20
Methylnicotinamide	65.37	340.36	64.72	52.98	73.70
Aminobutyrate	18.73	24.29	12.18	172.43	15.64
Hydroxyisobutyrate	26.05	41.68	65.37	74.44	83.93
Oxoglutarate	71.52	67.36	23.81	1199.91	33.12
Aminoisobutyrate	1480.30	116.75	14.30	555.57	29.67
Hydroxybutyrate	56.83	43.82	5.64	175.91	76.71
Hydroxyisovalerate	10.07	79.84	23.34	25.03	69.41
Indoxylsulfate	566.80	368.71	665.14	411.58	165.67
Hydroxyphenylacetate	120.30	432.68	292.95	214.86	97.51

En esta parte, he mirado las dimensiones que forman parte del *SummarizedExperiment* así como impreso el número de filas y columnas que forman parte de él para verificar que se ha formado correctamente el contenedor.

```
# Representación de las dimensiones del SummarizedExperiment
dim(SummarizedExp_cachexia)
```

```
## [1] 63 77
```

```
# Extracción del nombre de las columnas del SummarizedExperiment
colnames(SummarizedExp_cachexia)
```

```
##  [1] "PIF_178"      "PIF_087"      "PIF_090"      "NETL_005_V1"  "PIF_115"
##  [6] "PIF_110"      "NETL_019_V1"  "NETCR_014_V1" "NETCR_014_V2" "PIF_154"
## [11] "NETL_022_V1"  "NETL_022_V2"  "NETL_008_V1"  "PIF_146"      "PIF_119"
## [16] "PIF_099"      "PIF_162"      "PIF_160"      "PIF_113"      "PIF_143"
## [21] "NETCR_007_V1" "NETCR_007_V2" "PIF_137"      "PIF_100"      "NETL_004_V1"
## [26] "PIF_094"      "PIF_132"      "PIF_163"      "NETCR_003_V1" "NETL_028_V1"
## [31] "NETL_028_V2"  "NETCR_013_V1" "NETL_020_V1"  "NETL_020_V2"  "PIF_192"
## [36] "NETCR_012_V1" "NETCR_012_V2" "PIF_089"      "NETCR_002_V1" "PIF_179"
## [41] "PIF_114"      "NETCR_006_V1" "PIF_141"      "NETCR_025_V1" "NETCR_025_V2"
## [46] "NETCR_016_V1" "PIF_116"      "PIF_191"      "PIF_164"      "NETL_013_V1"
## [51] "PIF_188"      "PIF_195"      "NETCR_015_V1" "PIF_102"      "NETL_010_V1"
## [56] "NETL_010_V2"  "NETL_001_V1"  "NETCR_015_V2" "NETCR_005_V1" "PIF_111"
## [61] "PIF_171"      "NETCR_008_V1" "NETCR_008_V2" "NETL_017_V1"  "NETL_017_V2"
## [66] "NETL_002_V1"  "NETL_002_V2"  "PIF_190"      "NETCR_009_V1" "NETCR_009_V2"
## [71] "NETL_007_V1"  "PIF_112"      "NETCR_019_V2" "NETL_012_V1"  "NETL_012_V2"
## [76] "NETL_003_V1"  "NETL_003_V2"
```

```
# Extracción del nombre de las filas del SummarizedExperiment
rownames(SummarizedExp_cachexia)
```

```
##  [1] "Anhydro_beta_D_glucose" "Methylnicotinamide"     "Aminobutyrate"
##  [4] "Hydroxyisobutyrate"     "Oxoglutarate"           "Aminoisobutyrate"
##  [7] "Hydroxybutyrate"        "Hydroxyisovalerate"     "Indoxylsulfate"
## [10] "Hydroxyphenylacetate"   "Acetate"                "Acetone"
## [13] "Adipate"                "Alanine"                "Asparagine"
## [16] "Betaine"                "Carnitine"              "Citrate"
## [19] "Creatine"               "Creatinine"             "Dimethylamine"
## [22] "Ethanolamine"           "Formate"                "Fucose"
## [25] "Fumarate"               "Glucose"                "Glutamine"
## [28] "Glycine"                "Glycolate"              "Guanidoacetate"
## [31] "Hippurate"              "Histidine"              "Hypoxanthine"
## [34] "Isoleucine"             "Lactate"                "Leucine"
## [37] "Lysine"                 "Methylamine"            "Methylguanidine"
## [40] "N_N_Dimethylglycine"    "O_Acetylcarnitine"      "Pantothenate"
## [43] "Pyroglutamate"          "Pyruvate"               "Quinolinate"
## [46] "Serine"                 "Succinate"              "Sucrose"
## [49] "Tartrate"               "Taurine"                "Threonine"
## [52] "Trigonelline"           "Trimethylamine_N_oxide" "Tryptophan"
## [55] "Tyrosine"               "Uracil"                 "Valine"
## [58] "Xylose"                 "cis_Aconitate"          "myo_Inositol"
## [61] "trans_Aconitate"        "pi_Methylhistidine"     "tau_Methylhistidine"
```

Así, he observado que las dimensiones de la matriz son 63 filas y 77 columnas y que los nombres que aparecen en cada una de estas son las que se pretendía, de manera que se ha creado correctamente el contenedor *SummarizedExperiment*.

Luego he empleado la función *str()* para explorar los datos del *SummarizedExperiment* de nuevo y compararlo con lo extraído previamente.

```
# Exploración de la estructura de los datos de SummarizedExperiment
str(assay(SummarizedExp_cachexia))
```

```
##  num [1:63, 1:77] 40.9 65.4 18.7 26.1 71.5 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:63] "Anhydro_beta_D_glucose" "Methylnicotinamide" "Aminobutyrate" "Hydroxyisobutyrate
##   ..$ : chr [1:77] "PIF_178" "PIF_087" "PIF_090" "NETL_005_V1" ...
```

De esta manera, se logra visualizar también lo mismo que al extraer las dimensiones y los nombres de las filas y columnas, afirmando que el contenedor *SummarizedExperiment* se ha podido crear correctamente.

Estadísticas Descriptivas de los Metabolitos

Para profundizar más en el estudio estadístico, he calculado las estadísticas descriptivas básicas de los metabolitos con la función *summary()*, redondeando los valores a 2 decimales. También he usado la función *apply()* para aplicar la función *summary()* a cada uno de los metabolitos. Así, posteriormente he empleado la función *kable()* para representar en una tabla los resultados obtenidos del estudio descriptivo.

```r
# Cálculo de las estadísticas descriptivas de los metabolitos
summary_general <- round(apply((assay(SummarizedExp_cachexia)), 1, summary), 2)
summary_df <- as.data.frame(summary_general)

# Impresión de las estadísticas descriptivas calculadas
kable(t(summary_df), caption="Estadísticos por cada Metabolito") %>%
  kable_styling("striped", full_width=F)
```

Table 2: Estadísticos por cada Metabolito

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Anhydro_beta_D_glucose	4.71	28.79	45.60	105.63	141.17	685.40
Methylnicotinamide	6.42	15.80	36.60	71.57	73.70	1032.77
Aminobutyrate	1.28	5.26	10.49	18.16	19.49	172.43
Hydroxyisobutyrate	4.85	15.80	32.46	37.25	54.60	93.69
Oxoglutarate	5.53	22.42	55.15	145.09	92.76	2465.13
Aminoisobutyrate	2.61	11.70	22.65	76.76	56.26	1480.30
Hydroxybutyrate	1.70	5.99	11.70	21.72	29.96	175.91
Hydroxyisovalerate	0.92	5.26	12.55	21.65	30.27	164.02
Indoxylsulfate	27.66	82.27	144.03	218.88	333.62	1043.15
Hydroxyphenylacetate	15.49	41.68	70.11	112.02	145.47	796.32
Acetate	3.49	16.28	39.65	66.14	86.49	411.58
Acetone	2.29	4.95	7.10	11.43	10.49	206.44
Adipate	1.55	6.11	10.18	24.76	19.11	327.01
Alanine	16.78	78.26	194.42	273.56	399.41	1312.91
Asparagine	6.69	20.49	42.10	62.28	89.12	273.14
Betaine	2.29	28.79	64.72	90.32	127.74	391.51
Carnitine	2.18	14.44	23.81	52.09	60.95	487.85
Citrate	59.74	788.40	1790.05	2235.35	3071.74	13629.61
Creatine	2.75	17.64	44.26	126.83	117.92	1863.11
Creatinine	1002.25	3498.19	7631.20	8733.97	12332.58	33860.35
Dimethylamine	41.26	142.59	304.90	358.17	454.86	1556.20
Ethanolamine	16.12	86.49	204.38	276.26	407.48	1436.55
Formate	6.42	53.52	95.58	147.40	167.34	1480.30
Fucose	5.70	29.37	61.56	88.67	123.97	407.48
Fumarate	0.79	2.23	4.10	8.44	7.85	96.54
Glucose	26.84	80.64	210.61	559.84	407.48	8690.62
Glutamine	23.34	113.30	225.88	306.87	445.86	1685.81
Glycine	38.09	262.43	528.48	880.72	1096.63	5064.45
Glycolate	5.42	50.91	130.32	187.99	267.74	720.54
Guanidoacetate	7.03	33.78	64.72	86.37	108.85	561.16
Hippurate	92.76	492.75	1224.15	2286.84	2921.93	19341.34
Histidine	14.15	66.69	174.16	292.64	419.89	1863.11
Hypoxanthine	3.78	20.70	40.04	61.10	83.93	265.07
Isoleucine	1.79	3.90	7.17	8.71	11.25	40.04
Lactate	7.32	35.52	81.45	158.46	139.77	3640.95
Leucine	2.51	9.12	19.11	24.36	31.19	103.54
Lysine	10.49	30.27	69.41	108.79	121.51	788.40
Methylamine	1.51	5.26	14.73	17.38	24.05	52.46

Methylguanidine	1.70	4.26	7.85	15.32	19.30	141.17
N_N_Dimethylglycine	0.79	7.03	21.98	26.35	40.04	120.30
O_Acetylcarnitine	1.23	3.94	11.47	19.73	20.91	254.68
Pantothenate	2.59	11.13	22.65	44.88	41.26	692.29
Pyroglutamate	21.33	68.72	157.59	211.45	301.87	1064.22
Pyruvate	0.90	4.85	13.46	21.29	29.08	184.93
Quinolinate	5.21	26.58	51.42	66.44	87.36	259.82
Serine	16.12	83.10	142.59	197.69	270.43	1248.88
Succinate	1.72	8.58	30.88	60.23	74.44	589.93
Sucrose	6.49	19.30	40.85	113.23	94.63	2079.74
Tartrate	2.20	6.89	12.94	40.00	25.79	837.15
Taurine	17.81	99.48	249.64	525.12	665.14	4272.69
Threonine	8.25	31.82	64.07	95.36	137.00	450.34
Trigonelline	10.07	53.52	114.43	270.44	340.36	2252.96
Trimethylamine_N_oxide	55.70	175.91	383.75	652.16	735.10	5486.25
Tryptophan	8.67	21.33	46.99	66.24	96.54	259.82
Tyrosine	4.22	23.57	60.34	81.76	113.30	539.15
Uracil	3.10	11.94	27.39	35.56	44.26	179.47
Valine	4.10	12.18	33.12	35.67	50.40	160.77
Xylose	10.07	29.96	50.40	100.93	89.12	2164.62
cis_Aconitate	12.94	36.23	129.02	204.22	254.68	1863.11
myo_Inositol	11.59	30.27	78.26	135.40	167.34	854.06
trans_Aconitate	4.90	12.43	26.84	40.63	57.40	217.02
pi_Methylhistidine	11.36	67.36	162.39	370.29	387.61	2697.28
tau_Methylhistidine	8.00	27.39	68.72	89.69	130.32	317.35

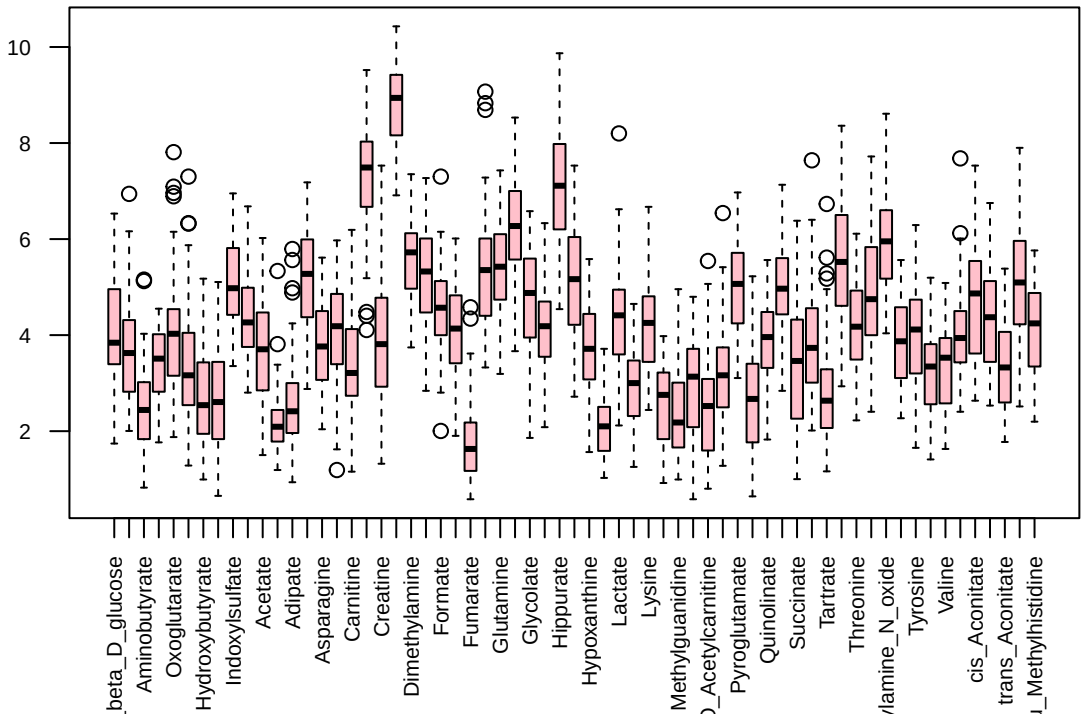
Boxplot de Metabolitos

Para ver visualmente la distribución de los datos, debido a la gran diferencia que hay entre el valor más alto de metabolito respecto a los valores más bajos, he aplicado una transformación logarítmica con la función *log1p()*. Así, he reducido la varianza y he visualizdo una representación de los datos de manera más normal.

```
# Aplicación de la transformación logarítmica
logaritmo_matriz_traspuesta <- log1p(t(assay(SummarizedExp_cachexia)))
```

Posteriormente, he creado un boxplot para observar la distribución de los valores de los metabolitos en el dataset entero. Para ello, he usado la función *boxplot()*.

```
boxplot(logaritmo_matriz_traspuesta, las=2, cex.axis=0.7, col = "pink")
```



En este, se observa que hay metabolitos que tienen valores más altos de sus concentraciones, mientras que hay otros que presentan valores más bajos.

Matriz de Correlación

Una forma interesante de estudiar la distribución de los datos es hacer una matriz de correlación de los diferentes metabolitos del dataset. Así, se puede ver las relaciones que existen entre estos.

Para comenzar, lo primero que he hecho ha sido calcular las matrices de covarianza y correlación de la información numérica del *SummarizedExperiment* usando las funciones *cov()* y *cor()*. Al calcular la matriz de covarianza además he usado la función *dim()* para extraer las dimensiones del *SummarizedExperiment* y tenerla en cuenta en el cálculo de las covarianzas. Luego de usar las funciones para calcular la covarianza y la correlación, he empleado la función *kable()* para representar en una tabla los primeros valores obtenidos de dichas correlaciones. En ambos casos he especificado que se vea la información de las 10 primeras filas y las 4 primeras columnas.

```
# Cálculo de la matriz de covarianza por cada metabolito
dimensions <- dim(t(assay(SummarizedExp_cachexia)))[1]
matriz_covarianza <- cov(t(assay(SummarizedExp_cachexia))*(dimensions-1)/dimensions)

# Representación de la matriz de covarianza por cada metabolito
kable(matriz_covarianza[1:10,1:4])
```

	Anhydro_beta_D_glucose	Methylnicotinamide	Aminobutyrate	Hydroxyisobutyrate
Anhydro_beta_D_glucose	16470.3730	990.99290	913.42854	1523.3772
Methylnicotinamide	990.9929	17282.52926	5.27808	992.2423
Aminobutyrate	913.4285	5.27808	742.88386	248.9041
Hydroxyisobutyrate	1523.3772	992.24225	248.90407	559.1182
Oxoglutarate	-504.9448	3126.39285	2468.72394	3124.8339
Aminoisobutyrate	1607.6577	497.67456	1607.73015	613.4824
Hydroxybutyrate	707.3334	489.13578	424.80248	320.1585
Hydroxyisovalerate	996.0183	1143.96481	74.64606	246.7438
Indoxylsulfate	7084.1179	8965.02535	1685.42372	1781.8400
Hydroxyphenylacetate	5543.0573	3037.22327	938.34658	1248.9974

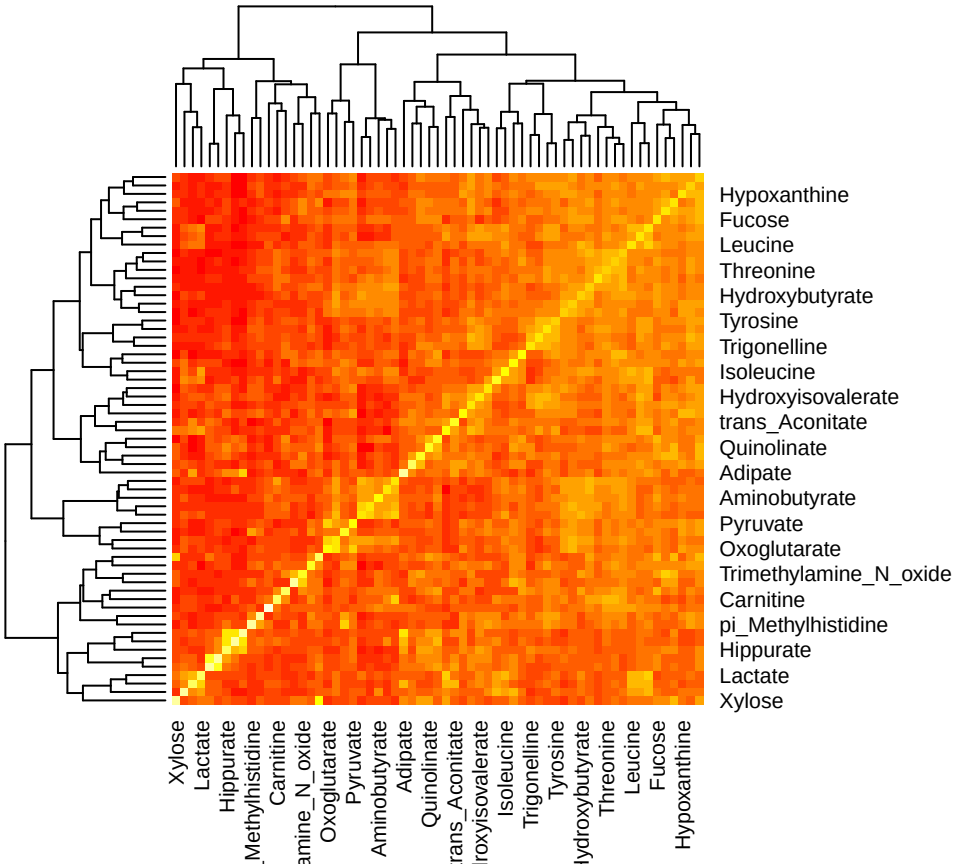
```
# Cálculo de la matriz de correlación
matriz_correlacion <- cor(t(assay(SummarizedExp_cachexia)))

# Representación de la matriz de correlación
kable(matriz_correlacion[1:10,1:4])
```

	Anhydro_beta_D_glucose	Methylnicotinamide	Aminobutyrate	Hydroxyisobutyrate
Anhydro_beta_D_glucose	1.0000000	0.0587375	0.2611334	0.5020003
Methylnicotinamide	0.0587375	1.0000000	0.0014730	0.3191995
Aminobutyrate	0.2611334	0.0014730	1.0000000	0.3862066
Hydroxyisobutyrate	0.5020003	0.3191995	0.3862066	1.0000000
Oxoglutarate	-0.0116381	0.0703442	0.2679171	0.3908981
Aminoisobutyrate	0.0664436	0.0200795	0.3128701	0.1376137
Hydroxybutyrate	0.2131407	0.1438864	0.6027269	0.5236092
Hydroxyisovalerate	0.3152029	0.3534139	0.1112298	0.4238079
Indoxylsulfate	0.2840759	0.3509524	0.3182358	0.3878085
Hydroxyphenylacetate	0.3622119	0.1937484	0.2887140	0.4429704

En esta tablas resultantes, se puede observar la covarianza calculada para cada metabolito representada en una matriz y, también la correlación.

A la hora de representar la matriz de correlación mediante la función *heatmap()*, es importante tener en cuenta la matriz de correlación calculada previamente, ya que los valores que se van a ver visualmente representados son los de esa matriz. Esta función *heatmap()* permite visualizar con un gradiente de color las correlaciones que se dan de manera más fuerte o débil entre los metabolitos. En este caso, de color amarillo se ven las relaciones con mayor correlación, mientras que en naranja/rojo se ven las relaciones con menor correlación.

```
# Representación de un heatmap de la matriz de correlación
heatmap(as.matrix(matriz_correlacion), col = heat.colors(16))
```



Como se puede observar, la representación se ha podido realizar satisfactoriamente. Esto lo sé porque en la diagonal se ve en amarillo, con la máxima correlación posible generada por las relaciones entre las mismas variables. A lo largo del *heatmap* se pueden ver diferentes tipos de relaciones, habiendo metabolitos con mayor o menor correlación, indicando, a su vez, que pueden estar más o menos relacionados.

Conteo de los Pacientes por Grupo

He considerado interesante saber la cantidad de pacientes que hay en cada grupo de *Muscle_loss* (control y caquexia) para poder realizar otro tipo de representaciones y poder comparar ambos grupos. Más adelante, podría ser interesante visualizar si hay diferencias significativas, por ejemplo, en los valores que hay por cada metabolito dependiendo del grupo. Esta vez, para representar los valores obtenidos, he usado la función *table()*

```
# Contaje de la cantidad de pacientes por grupo de Muscle_loss
table(data_cachexia$Muscle_loss)
```

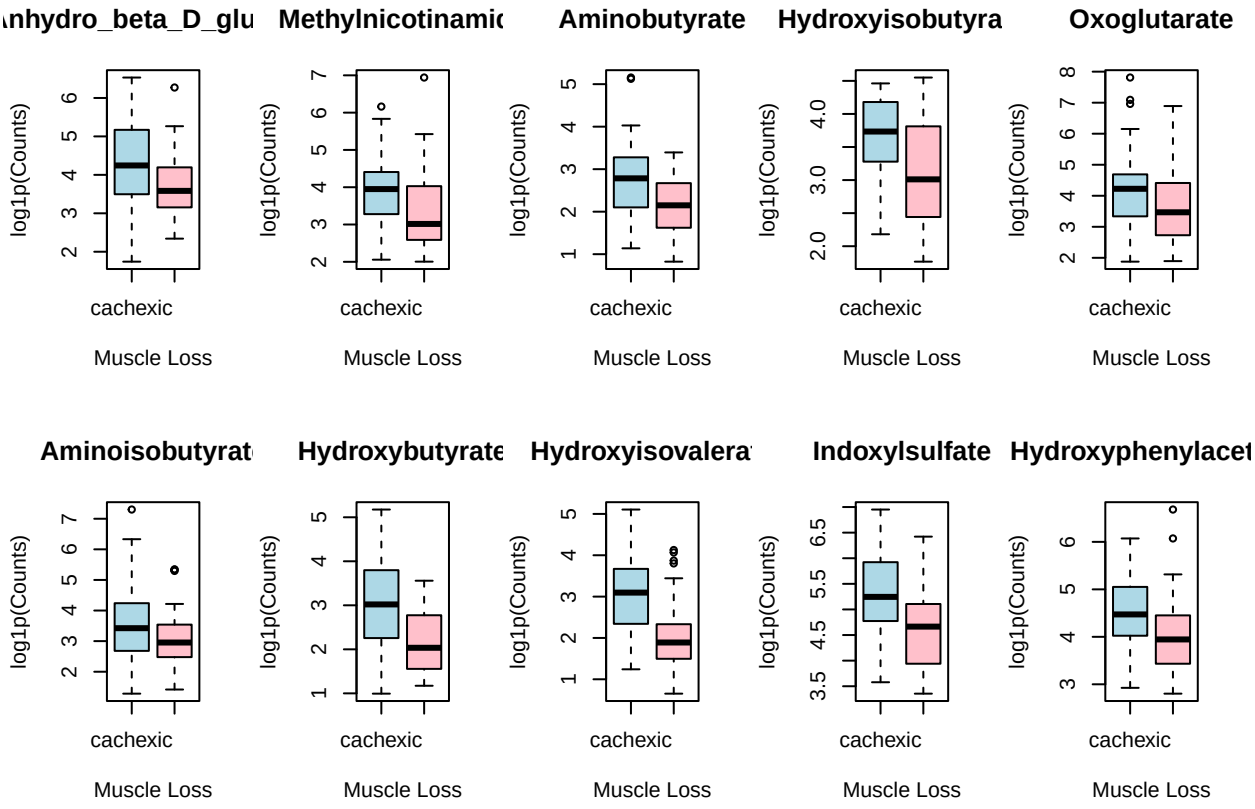
```
##
## cachexic  control
##       47       30
```

De esta manera, se puede determinar que en el grupo de pacientes con caquexia hay 47 pacientes, mientras que en el grupo control hay 30 pacientes.

Boxplot Comparativo por Grupos

Teniendo en cuenta la cantidad de pacientes que hay por cada grupo, he considerado representar mediante *boxplot()* los valores de las concentraciones de metabolitos que hay en cada grupo. Para no extender excesivamente este apartado, he hecho la representación del boxplot para los primeros 10 metabolitos del dataset. No obstante, previamente ya he observado esta representación para todos los metabolitos.

```
# Creación de un gráfico layout en el cual representar posteriormente los boxplots
par(mfrow = c(2,5))

# Extracción de la información de Muscle_loss
muscle_loss <- colData(SummarizedExp_cachexia)$Muscle_loss

# Carga de los datos numéricos en función de los grupos
boxplot_grupos <- data.frame(logaritmo_matriz_traspuesta)
boxplot_grupos$Muscle_loss <- muscle_loss

# Bucle para hacer el boxplot para cada metabolito en función del grupo
for(i in 1:10){
  boxplot(boxplot_grupos[[i]] ~ boxplot_grupos$Muscle_loss,
          main = colnames(boxplot_grupos)[i],
          xlab = "Muscle Loss",
          ylab = "log1p(Counts)",
          col = c("lightblue", "pink"))
}
```



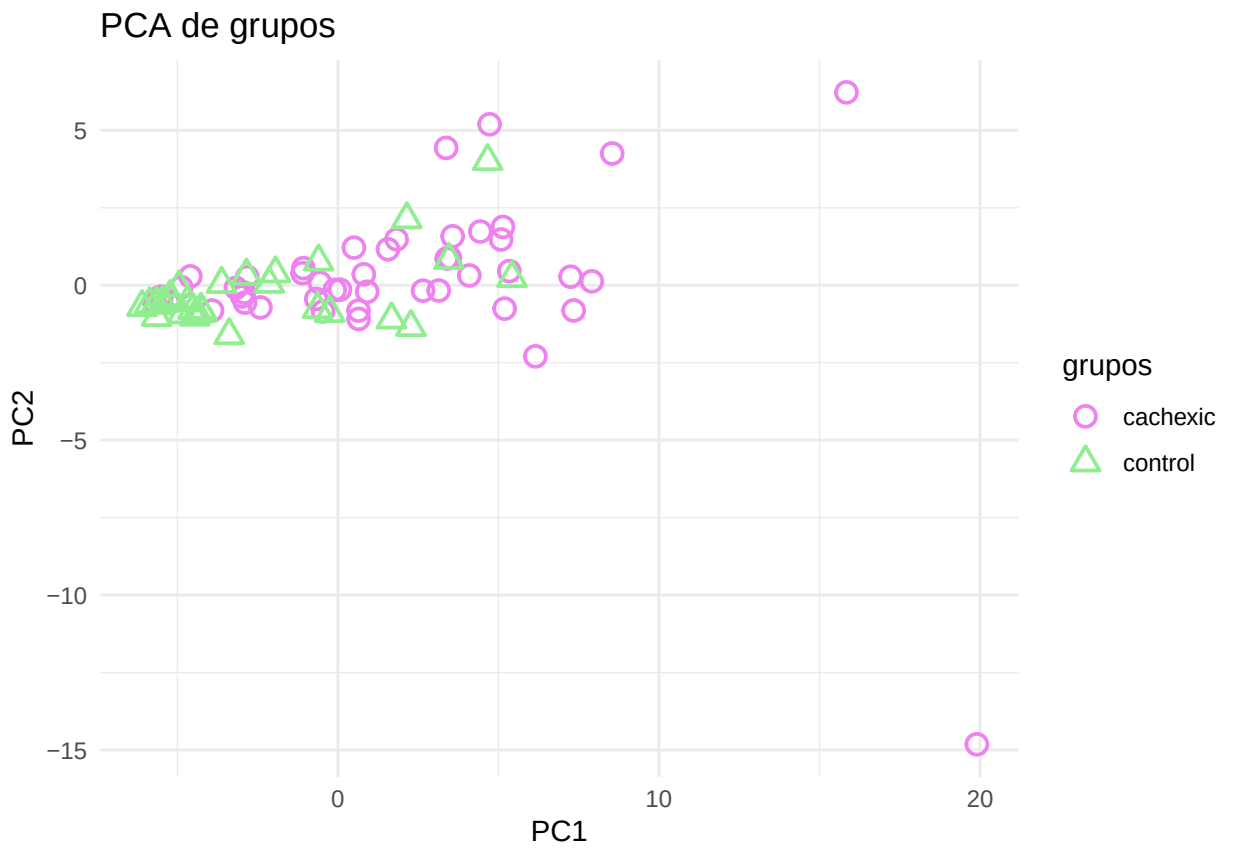
De acuerdo a estas representaciones, se puede considerar que los pacientes del grupo con caquexia presentan, en general, mayor nivel de metabolitos que el grupo control. Esto se podría estudiar a nivel de significancia en un estudio futuro. Además, se podría determinar en qué metabolitos hay diferencias significativas y valorar más profundamente la posibilidad de que las vías metabólicas relacionadas con estos metabolitos puedan ser dianas terapeúticas.

Análisis de Componentes Principales (CPA)

Para reforzar el estudio previo en función de los grupos, he realizado un Análisis de Componentes Principales (CPA).

Para llevar a cabo este análisis, he usado los datos traspuestos de la matriz con los valores numéricos de los metabolitos traspuestos empleando la función *t()*. También he especificado que la media de cada metabolito sea cero (*center=T*) y que su desviación estándar esté escalada, es decir, sea de uno (*scale.=T*) Luego, he transformado los datos en DataFrame con la función *as.data.frame()* y he añadido la columna con información del grupo en el que estaba cada paciente. Una vez obtenida esta información, he usado la función *ggplot()* para representar el CPA.

```
# trasposición de los datos y formación del DataFrame
analisis_comp_principales <- prcomp(t(assay(SummarizedExp_cachexia)),center=T,scale.=T)
analisis_comp_principales_df <- as.data.frame(analisis_comp_principales$x)

# Adición de la columna grupos al DataFrame para diferenciar los grupos de Muscle_loss
analisis_comp_principales_df$grupos <- colData(SummarizedExp_cachexia)$Muscle_loss

# Graficar PCA
ggplot(analisis_comp_principales_df, aes(x = PC1, y = PC2, color = grupos, shape= grupos)) +
  geom_point(size = 3, stroke=1, fill = NA) +
  labs(title = "PCA de grupos") +
  theme_minimal() +
  scale_shape_manual(values = c(21, 24)) +
  scale_color_manual(values = c("violet", "lightgreen"))
```



En esta representación del CPA, se pueden ver las muestras del grupo control representadas con triangulos verdes y las muestras del grupo de pacientes caquéxicos con circulos violetas. Se observa que hay zonas en el plot donde se solapan los PC para ambos grupos, mientras que los valores asociados al grupo de pacientes con caquexia presentan mayor dispersión.

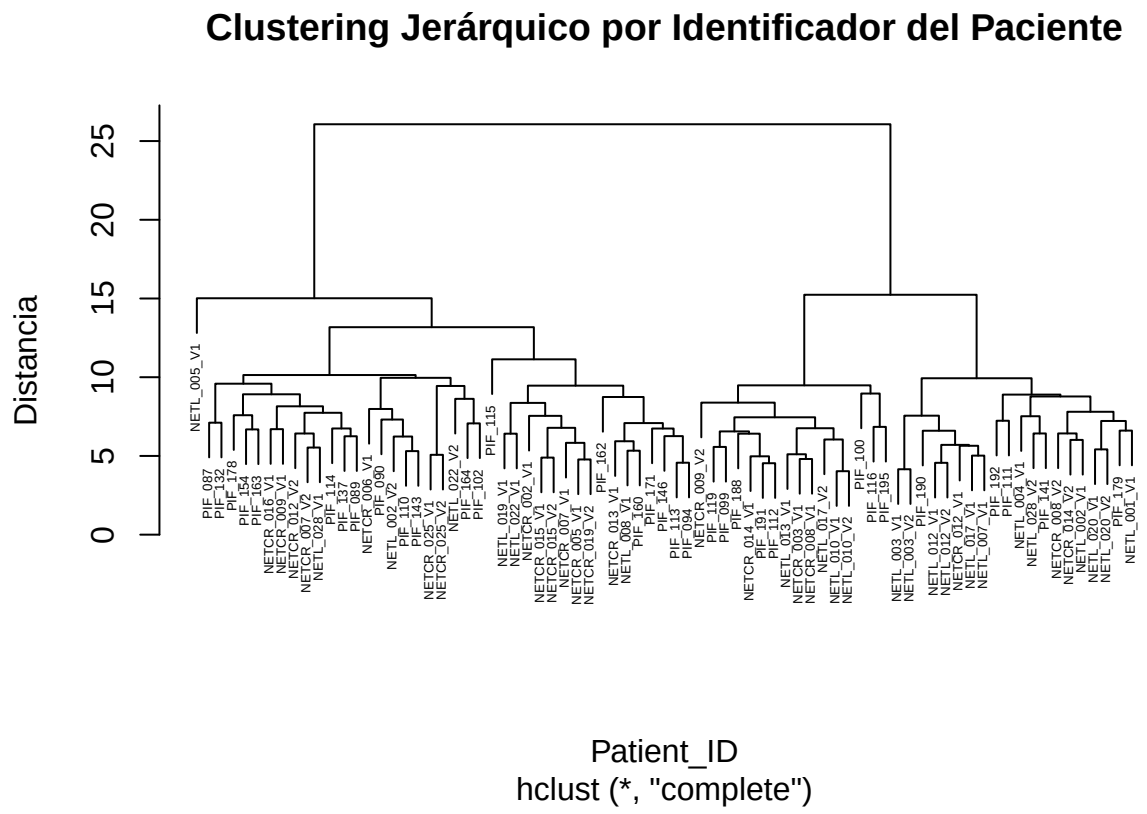
Clustering Jerárquico por Identificador del Paciente

El clustering jerárquico es una buena manera de agrupar los valores numéricos teniendo en cuenta las agrupaciones que se pueden realizar con los datos contenidos en el *SummarizedExperiment*.

En este caso, he representado un clustering por muestras para observar la distribución de los grupos. Para ello, he realizado un escalado de los datos y he implementado la función *log1p()* para poder ajustar con la función logarítmica los valores de la traspuesta de la matriz numérica del *SummarizedExperiment*. Posteriormente, he calculado la distancia de la matriz con el método euclideano con la función *dist()*. Una vez obtenido esto, he empleado la función *hclust()* para aplicar el clustering jerárquico teniendo en cuenta la distancia de la matriz calculada a partir de los datos escalados. Luego, he usado la función *plot()* para representar este clustering.

```
# Escalado de los datos una vez aplicada la función logarítmica a la traspuesta de la matriz numérica
matriz_numerica_traspuesta_escalada <- scale(log1p(t(assay(SummarizedExp_cachexia))))

# Cálculo de la distancia de la matriz con el método euclideano
distancia_matriz_traspuesta <- dist(matriz_numerica_traspuesta_escalada, method="euclidean")

# Aplicación del clustering jerárquico con el método completo
resultado_clustering_identificador <- hclust(distancia_matriz_traspuesta, method="complete")
```

```r
# Representación del clustering jerárquico
plot(resultado_clustering_identificador, labels = colData(SummarizedExp_cachexia)$Patient_ID,
     main ="Clustering Jerárquico por Identificador del Paciente",
     xlab ="Patient_ID",
     ylab ="Distancia",
     cex = 0.4)
```



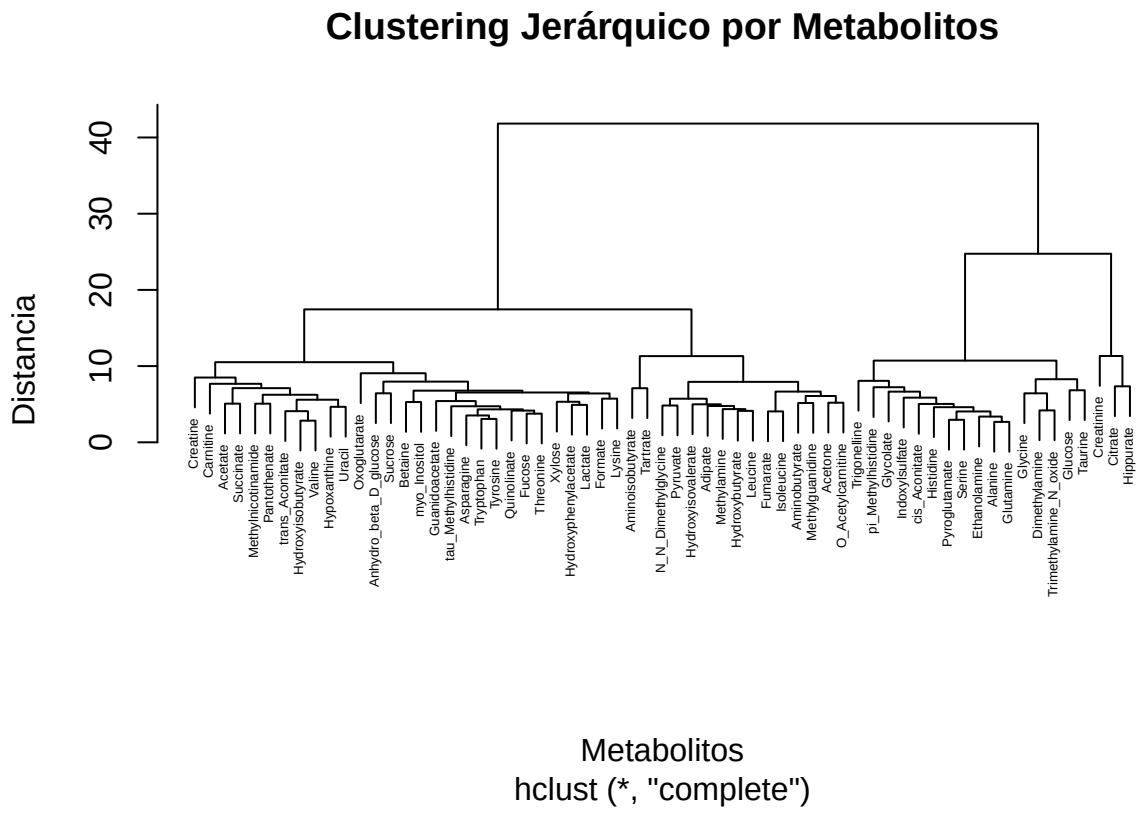
En este clustering jerárquico por identificador del paciente, se puede observar que hay dos grupos diferenciados que contienen los metabolitos para diversos pacientes (*Patient_ID*). Estos subgrupos se han generado en función de los perfiles similares que hay entre los pacientes teniendo en cuenta los valores de las concentraciones de los metabolitos.

Clustering Jerárquico por Metabolitos

Igual que en el caso anterior, puede ser interesante representar el clustering jerárquico en función de los metabolitos. Para ello, he realizado el mismo procedimiento que en el apartado anterior, pero esta vez sin usar la traspuesta de la matriz numérica, de manera que en las filas se encuentran los nombres de los metabolitos y estos se asocian con los valores de las concentraciones de los mismos.

```r
# Escalado de los datos una vez aplicada la función logarítmica a la matriz numérica
matriz_numerica_escalada <- scale(log1p(assay(SummarizedExp_cachexia)))

# Cálculo de la distancia de la matriz con el método euclideano
distancia_matrix <- dist(matriz_numerica_escalada, method="euclidean")
```

```r
# Aplicación del clustering jerárquico con el método completo
resultado_clustering_metabolitos <- hclust(distancia_matrix, method ="complete")

# Representación del clustering jerárquico
plot(resultado_clustering_metabolitos <- hclust(distancia_matrix, method ="complete"),
     labels =rownames(assay(SummarizedExp_cachexia)),
     main ="Clustering Jerárquico por Metabolitos",
     xlab ="Metabolitos",
     ylab ="Distancia", cex = 0.4)
```



En este caso, también se ve como hay agrupados dos grupos diferentes. La presencia de estos dos grupos puede deberse a las relaciones que hay entre los metabolitos, por ejemplo, a nivel de las vías metabólicas.

GitHub

He generado el repositorio de *GitHub*, nombrado *"Frias-Perez-CarmenLaura-PEC1"*, para almacenar los datos y compartirlos. La información que se puede encontrar recogida en este repositorio es la siguiente:

- Informe: El informe con la solución propuesta de la entrega en formato R Markdown (*.rmd*).
- Datos: Los datos del dataset en formato texto (*.csv* o *.txt*) y los del *SummarizedExperiment* en formato binario (*.Rda*).
- Código: El código R que he usado para explorar los datos (*.R*).
- Metadatos del dataset: En un archivo Markdown (*.md*).

Para descargar los documentos de los datos del SummarizedExp_cachexia en formato binario y de los metadatos del mismo en formato Markdown, he empleado la siguiente información:

```
# Guardado del SummarizedExp_cachexia en un archivo .Rda
save(SummarizedExp_cachexia, file = "SummarizedExp_cachexia.Rda")

# Guardado de los metadatos del dataset human_cachexia.csv en un archivo .md
writeLines(c(
  "## Descripción General\n",
  "Este dataset contiene información sobre las mediciones
  de diferentes metabolitos en diferentes pacientes identificados,
  además de información sobre si estos pacientes presentan caquexia
  o no, formando parte del grupo control.\n",
  "El dataset ha sido utilizado en varios tutoriales de
  MetaboAnalyst y contiene dos grupos de muestras: uno de
  caquexia y otro de control. Los valores de los metabolitos
  son numéricos y no se han detectado valores faltantes (NaN)
  en los datos recogidos.\n",

  "### Información del Dataset\n",
  "Nombre del Dataset: 'human_cachexia.csv'\n",
  "Fuente: 'https://github.com/nutrimetabolomics/metaboData/tree/main/Datasets'\n",

  "## Información sobre las Columnas\n",
  "Descripción: Las columnas de este dataset contienen
  los identificadores de los pacientes, los grupos a los
  que pertenecen y los diferentes metabolitos estudiados.\n",
  "Número de pacientes: 77\n",
  "Identificadores de los Pacientes: PIF_178, PIF_087, NETL_003_V1,
  NETL_005_V1, PIF_115, PIF_110, NETL_019_V1,
  NETCR_014_V1, NETCR_014_V2, PIF_154, NETL_022_V1,
  NETL_022_V2, NETL_008_V1, PIF_146, PIF_119,
  PIF_099, PIF_162, PIF_160, PIF_113,
  PIF_143, NETCR_007_V1, NETCR_007_V2, PIF_137,
  PIF_100, NETL_004_V1, PIF_094, PIF_132,
  PIF_163, NETCR_003_V1, NETL_028_V1, NETL_028_V2,
  NETCR_013_V1, NETL_020_V1, NETL_020_V2, PIF_192,
  NETCR_012_V1, NETCR_012_V2, PIF_089, NETCR_002_V1,
  PIF_179, PIF_114, NETCR_006_V1, PIF_141,
  NETCR_025_V1, NETCR_025_V2, NETCR_016_V1, PIF_116,
  PIF_191, PIF_164, NETL_013_V1, PIF_188,
  PIF_195, NETCR_015_V1, PIF_102, NETL_010_V1,
  NETL_010_V2, NETL_001_V1, NETCR_015_V2, NETCR_005_V1,
  PIF_111, PIF_171, NETCR_008_V1, NETCR_008_V2,
  NETL_017_V1, NETL_017_V2, NETL_002_V1, NETL_002_V2,
  PIF_190, NETCR_009_V1, NETCR_009_V2, NETL_007_V1,
  PIF_112, NETCR_019_V2, NETL_012_V1, NETL_012_V2,
  NETL_003_V1, NETL_003_V2.\n",
  "Cantidad de grupos: 2.\n",
  "Grupos: Caquexia y Control.\n",
  "Información sobre los grupos: En el grupo de caquexia hay 47 pacientes,
  mientras que en el grupo control hay 30.\n",
  "Número de metabolitos: 63\n",
  "Nombres de los metabolitos: Anhydro_beta_D_glucose, Methylnicotinamide, Aminobutyrate,
```

```
  Hydroxyisobutyrate, Oxoglutarate, Aminoisobutyrate,
  Hydroxybutyrate, Hydroxyisovalerate, Indoxylsulfate,
  Hydroxyphenylacetate, Acetate, Acetone,
  Adipate, Alanine, Asparagine,
  Betaine, Carnitine, Citrate,
  Creatine, Creatinine, Dimethylamine,
  Ethanolamine, Formate, Fucose,
  Fumarate, Glucose, Glutamine,
  Glycine, Glycolate, Guanidoacetate,
  Hippurate, Histidine, Hypoxanthine,
  Isoleucine, Lactate, Leucine,
  Lysine, Methylamine, Methylguanidine,
  N_N_Dimethylglycine, O_Acetylcarnitine, Pantothenate,
  Pyroglutamate, Pyruvate, Quinolinate,
  Serine, Succinate, Sucrose,
  Tartrate, Taurine, Threonine,
  Trigonelline, Trimethylamine_N_oxide, Tryptophan,
  Tyrosine, Uracil, Valine,
  Xylose, cis_Aconitate, myo_Inositol,
  trans_Aconitate, pi_Methylhistidine, tau_Methylhistidine.\n"
), "data_cachexia_metadata.md")
```

La dirección url del repositorio en la que se puede encontrar esta información es la siguiente:

https://github.com/CarmenLaura03/Frias-Perez-CarmenLaura-PEC1.git

Discusión, limitaciones y conclusiones del estudio

En esta entrega, se ha seleccionado el dataset *human_cachexia.csv* y se ha creado un contenedor *SummarizedExperiment* con la información contenida en el mismo (matriz numérica, colData, rowData, metadata).

Posteriormente, he hecho un análisis exploratorio del contenedor creado, pudiendo así estudiar de manera gráfica y numérica la distribución de los valores con todos los valores y en función de los grupos según *Muscle_loss* (control y caquexia).

En las estadísticas descriptivas y numéricas se ha observado la estructura del contenedor e información general de este, además de las medias de los metabolitos. También, a nivel gráfico, he representado diferentes *boxplots* para ver, en general, las concentraciones de estos teniendo en cuenta una aproximación logarítmica. También he hecho un *boxplot* para ver cómo varían estos valores entre grupos y, en general, se ha visto que los valores de las concentraciones de metabolitos eran mayores en el grupo de pacientes con caquexia. He realizado también un Análisis de Componentes Principales, que ha permitido reforzar las diferencias visuales y numéricas observadas previamente.

Para estudiar en mayor profundidad la correlación de los metabolitos, he calculado la matriz de correlación y la he representado mediante *heatmap()*, pudiendo observar las diferentes relaciones que hay entre los metabolitos. En esta representación, se ha visto que los metabolitos tienen correlaciones más o menos fuertes dependiendo, seguramente, de las vías metabólicas en las que están involucrados.

Por último, he representado un clustering jerárquico en función de los grupos y otro en función de los metabolitos para observar las agrupaciones que existen entre los datos. Así, se ha observado en ambos casos que hay dos agrupaciones. En el caso del clustering por muestras, estas agrupaciones podrían deberse a que hay muestras que comparten perfiles similares de metabolitos, mientras que, en el caso del clustering por metabolitos, podrían ser causadas en función de las vías metabólicas que los relacionan.

Una vez realizado este análisis, para finalizar la entrega, he creado un repositorio en *GitHub* con el informe, los datos y el código empleado.