PEC 1.Análisis de datos ómicos - Caquexia dataset

Daniel Zugasti Fernandez

2025-03-30

Contents

1 Abstract

Este análisis exploratorio analiza patrones metabólicos en caquexia mediante PCA, destacando metabolitos como aminoácidos y creatinina, que sugieren una mayor degradación muscular y alteraciones en el metabolismo energético. Aunque no se observó una separación clara entre caquexia y control, se identificó heterogeneidad metabólica, lo que podría reflejar variaciones en la progresión de la enfermedad. El análisis de los primeros 11 componentes principales, que explican el 80% de la variabilidad, ofrece una base para identificar biomarcadores relevantes y mejorar la precisión de futuros análisis.

2 Objetivos

El objetivo de esta primera PEC es obtener un conjunto de datos de metabolómica a través de MetabolomicsWorkbench o del repositorio de GitHub y crear un objeto de clase SummarizedExperiment que contenga los datos y los metadatos. Posteriormente, realizar un análisis exploratorio de los datos que proporcione una visión general del conjunto de datos, redactar un informe que describa el proceso y, finalmente, crear un repositorio en GitHub que contenga toda la información.

3 Métodos

3.1 Justificación de la selección del dataset

Para este estudio, se seleccionó el conjunto de datos de metabolómica de caquexia humana disponible en MetaboAnalyst (https://github.com/nutrimetabolomics/metaboData/blob/main/Datasets/2024-Cachexia/description.md). Este dataset ha sido ampliamente utilizado en tutoriales de análisis metabolómico debido a su calidad y completitud: contiene únicamente valores numéricos sin datos faltantes y está compuesto por dos grupos de muestras bien definidos. La ausencia de valores perdidos facilita el procesamiento y análisis, mientras que la clasificación en grupos permite evaluar diferencias metabólicas relevantes.

Además de su idoneidad técnica, la elección de este dataset responde a un interés personal en la bioquímica, disciplina en la que me formé. La caquexia es un síndrome metabólico complejo que acompaña a enfermedades graves como el cáncer y la insuficiencia cardíaca, afectando profundamente el metabolismo y el estado nutricional de los pacientes.

3.2 Incorporación al objeto SummarizedExperiment

Para estructurar y organizar los datos de metabolómica de manera compatible con herramientas de análisis bioinformático en R, se creó un objeto de clase SummarizedExperiment. Este formato permite almacenar conjuntamente la matriz de expresión de metabolitos y los metadatos asociados a las muestras y variables, facilitando su manipulación y análisis.

Los datos fueron obtenidos del repositorio público de MetaboAnalyst y cargados en R. Se extrajeron las columnas correspondientes a los identificadores de los pacientes y su clasificación en función de la pérdida muscular. Posteriormente, se generó una matriz con las intensidades de los metabolitos, estructurando las filas como metabolitos y las columnas como muestras.

A continuación, se generaron dos conjuntos de metadatos: uno correspondiente a las muestras, que incluye los identificadores de los pacientes y su clasificación en uno de los dos grupos según la pérdida muscular; y otro correspondiente a las variables, que contiene la identificación de los metabolitos, permitiendo su referencia en el análisis.

Finalmente, la matriz de expresión y los metadatos se integraron en un objeto SummarizedExperiment, proporcionando una estructura eficiente para la exploración de los datos y la aplicación de métodos estadísticos en el análisis metabolómico.

3.3 Descripción del análisis exploratorio de los datos

Para obtener una visión general de los datos y evaluar diferencias entre grupos, se realizaron los siguientes análisis exploratorios:

3.3.1 Comprobación de valores faltantes

Se verificó la presencia de valores faltantes (NA) en los datos de expresión. Para ello, se extrajo la matriz de valores del objeto SummarizedExperiment utilizando la función assay() con la capa “counts”. Posteriormente, se aplicaron las funciones is.na() y any() para detectar la existencia de valores ausentes en el conjunto de datos.

3.3.2 Cálculo de la media de los metabolitos según el grupo de estudio

Se calcularon las medias de los valores de los metabolitos en función del grupo de estudio (control y caquexia). Para ello, se extrajeron los datos de expresión y se realizó un promedio por grupo utilizando la función aggregate(), con la variable muscle_loss de colData(se) como factor de agrupación.

Posteriormente, los datos fueron transpuestos para presentar los metabolitos como filas y los grupos como columnas. Los valores fueron redondeados a tres decimales para mejorar la legibilidad. Finalmente, se generó una tabla en formato LaTeX mediante kable(), permitiendo su visualización en múltiples páginas con la opción longtable.

3.Análisis de Componentes Principales (PCA) Se realizó un PCA para visualizar la variabilidad global de los datos y verificar si las muestras se agrupan según su clasificación mediante prcomp() indicando scale. true para escalar los datos ya que cada metabolito tiene sus valores...

3.3.3 Análisis de Componentes Principales (PCA)

Se realizó un Análisis de Componentes Principales (PCA) para evaluar la variabilidad global de los datos y explorar si las muestras se agrupan según su clasificación. El PCA se llevó a cabo con la función prcomp(), aplicando la opción scale. = TRUE para estandarizar los datos, dado que los metabolitos presentan escalas de magnitud diferentes.

Posteriormente, se calcularon las varianzas explicadas por los primeros componentes principales para interpretar su contribución a la variabilidad total del conjunto de datos. Los resultados fueron representados en un gráfico de dispersión de PC1 y PC2 utilizando ggplot2, donde las muestras fueron coloreadas según su grupo (muscle_loss). Además, se incluyeron etiquetas con los identificadores de los pacientes para facilitar la identificación de posibles patrones o valores atípicos.

3.3.4 Identificación de los metabolitos con mayor contribución a PC1 y PC2

Para identificar los metabolitos que más contribuyen a la variabilidad explicada por los primeros componentes principales, se extrajo la matriz de rotación del PCA (rotation), que contiene las cargas (loadings) de cada metabolito en las distintas componentes.

Se construyó un conjunto de datos con los valores de carga para PC1 y PC2 y se ordenaron en función de su contribución absoluta a PC1. Finalmente, se seleccionaron los 10 metabolitos con las mayores cargas en PC1 para su análisis e interpretación.

3.3.5 Identificación de valores atípicos en el PCA

Para detectar posibles valores atípicos en los datos, se analizaron las puntuaciones de las muestras en los dos primeros componentes principales (PC1 y PC2). Se estableció como criterio de outlier cualquier muestra cuya puntuación estuviera a más de dos desviaciones estándar de la media en PC1 o PC2.

Se generaron diagramas de caja (boxplots) para visualizar la distribución de las puntuaciones en cada componente y se resaltaron las muestras consideradas outliers mediante etiquetas con sus identificadores. Esta identificación de valores atípicos permite evaluar si existen muestras con patrones metabólicos diferenciados que podrían influir en los resultados del análisis.

3.3.6 Determinación de la varianza explicada por los componentes principales

Para evaluar cuántos componentes principales (PCs) son necesarios para explicar un porcentaje significativo de la variabilidad en los datos, se calculó la varianza explicada por cada componente y su acumulado. Se generó un gráfico de sedimentación (scree plot) en el que se representan las varianzas individuales de cada componente junto con la varianza acumulada.

Se identificó el número mínimo de componentes necesarios para explicar el 80%, 90% y 95% de la variabilidad en los datos, permitiendo seleccionar un número óptimo de componentes para su posterior interpretación y análisis, evitando incluir PCs que no aportan información relevante.

4 Resultados

4.1 Principales diferencias entre los objetos de la clase SummarizedExperiment y los objetos de la clase ExpressionSet.

1. Flexibilidad en la Información de las Filas:

SummarizedExperiment permite almacenar información de las filas tanto en GRanges (útil para datos de secuenciación como RNA-Seq o ChIP-Seq) como en DataFrames, lo que le da una gran flexibilidad para adaptarse a diferentes tipos de datos experimentales. ExpressionSet, por otro lado, utiliza exclusivamente DataFrames para almacenar la información de las filas, lo que lo hace adecuado para datos de expresión de microarrays.

2. Manejo de Múltiples Ensayos:

SummarizedExperiment es capaz de manejar múltiples ensayos (como diferentes tipos de secuenciación o mediciones) siempre y cuando estos ensayos compartan las mismas dimensiones (por ejemplo, las mismas muestras o características). Esta capacidad lo hace especialmente útil en experimentos complejos que generan varios tipos de datos provenientes de las mismas muestras. Por otro lado, ExpressionSet ha sido tradicionalmente limitado a un solo ensayo por objeto. Sin embargo, su funcionalidad se puede ampliar utilizando paquetes adicionales como MultiAssayExperiment, que permite manejar múltiples ensayos dentro de un único objeto.

3. Sincronización de Meta-datos:

SummarizedExperiment asegura que los meta-datos y los datos de los ensayos se mantengan sincronizados cuando se realizan subconjuntos o modificaciones, reduciendo el riesgo de desajustes entre los datos observacionales y los meta-datos, lo cual es fundamental para mantener la integridad de los resultados. En cambio, ExpressionSet no tiene un mecanismo tan riguroso para mantener esta sincronización.

4. Uso General:

SummarizedExperiment está diseñado principalmente para experimentos de secuenciación modernos (como RNA-Seq o ChIP-Seq), donde se generan datos complejos con múltiples tipos de ensayos. Por otro lado, ExpressionSet fue originalmente creado para datos de microarrays, aunque se ha extendido para trabajar con otros tipos de datos ómicos, especialmente cuando se combinan con paquetes adicionales como MultiAssayExperiment.

[1]

[2]

4.2 Importación y creación del objeto de clase SummarizedExperiment

El objeto SummarizedExperiment generado organiza los datos metabolómicos en una estructura que facilita su análisis. Este objeto tiene 63 filas y 77 columnas, donde cada fila representa un metabolito y cada columna una muestra correspondiente a un paciente. La matriz de datos (assays), almacenada en la capa "counts", contiene los valores de abundancia de los metabolitos.

Los identificadores de pacientes y su estado de pérdida muscular se almacenan en colData, con las variables "patient_id" y "muscle_loss". Por otro lado, la información sobre los metabolitos se encuentra en rowData, donde cada fila corresponde a un metabolito identificado por su nombre. La matriz de datos se ha construido a partir del conjunto original, eliminando las columnas no relacionadas con los metabolitos, transponiendo la matriz para organizar los datos con metabolitos en filas y pacientes en columnas, y asignando los identificadores de paciente como nombres de columna, garantizando que la estructura del objeto sea compatible con los análisis posteriores.

```
## class: SummarizedExperiment
## dim: 63 77
## metadata(0):
```

```
## assays(1): counts
## rownames(63): 1,6-Anhydro-beta-D-glucose 1-Methylnicotinamide ...
##   pi-Methylhistidine tau-Methylhistidine
## rowData names(1): metabolite
## colnames(77): PIF_178 PIF_087 ... NETL_003_V1 NETL_003_V2
## colData names(2): patient_id muscle_loss
```

4.3 Análisis exploratorio

Una vez tenemos el objeto SummarizedExperiment, y aunque en la información del dataset se indica que no hay valores faltantes, procedemos a comprobarlo.

```
## [1] "El dataset no contiene missing values: TRUE"
```

Como vemos, el código nos devuelve TRUE, por lo que podemos continuar con el análisis exploratorio.

Como podemos ver en la siguiente tabla, prácticamente todos los metabolitos presentan una media superior en el grupo de caquexia respecto al grupo control.

Table 1: Mean Metabolite Values by Muscle Loss Status

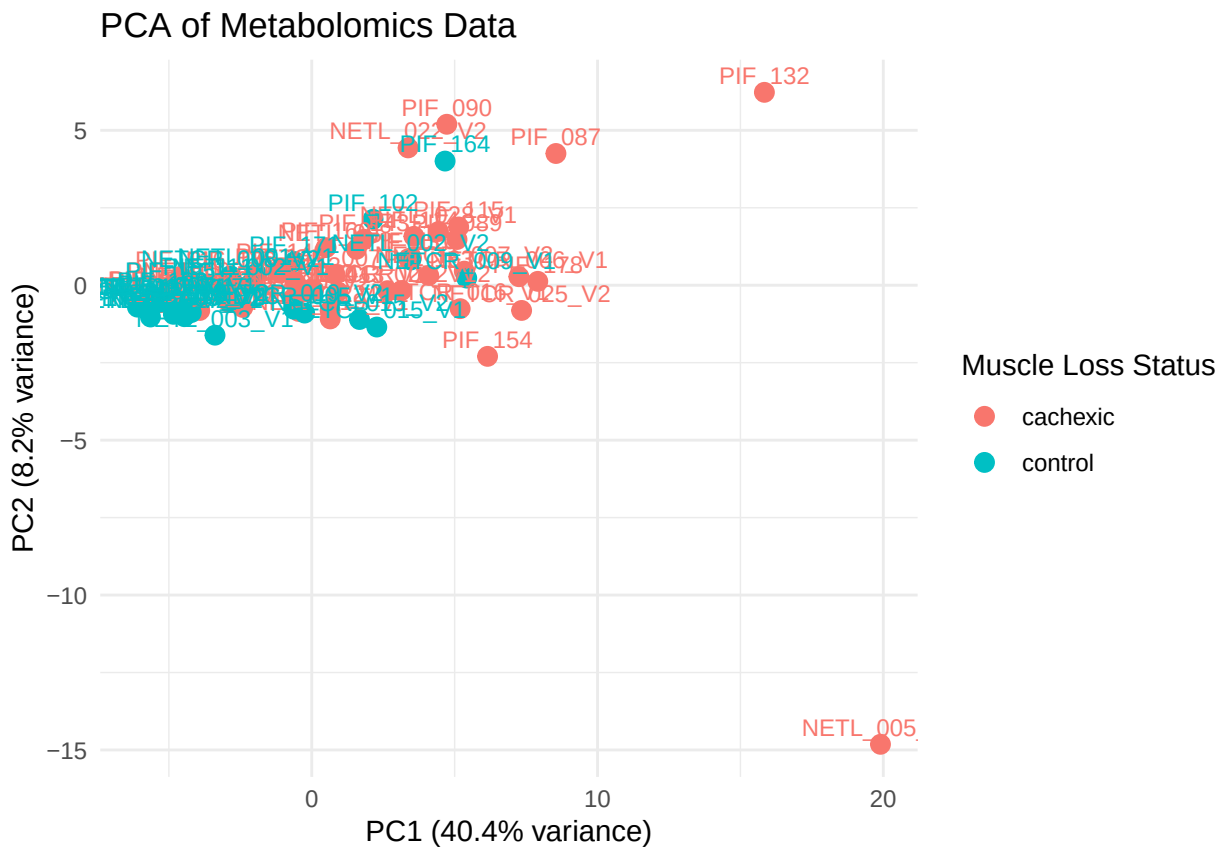
	cachexic	control
1,6-Anhydro-beta-D-glucose	128.689	69.505
1-Methylnicotinamide	70.564	73.155
2-Aminobutyrate	23.669	9.528
2-Hydroxyisobutyrate	43.238	27.871
2-Oxoglutarate	183.110	85.517
3-Aminoisobutyrate	100.275	39.911
3-Hydroxybutyrate	29.261	9.899
3-Hydroxyisovalerate	27.606	12.313
3-Indoxylsulfate	265.158	146.376
4-Hydroxyphenylacetate	119.823	99.799
Acetate	85.633	35.605
Acetone	13.346	8.420
Adipate	34.818	8.993
Alanine	347.591	157.584
Asparagine	75.391	41.749
Betaine	112.253	55.970
Carnitine	64.622	32.444
Citrate	2720.853	1474.719
Creatine	174.913	51.504
Creatinine	10722.140	5619.175
Dimethylamine	453.581	208.683
Ethanolamine	326.772	197.125
Formate	187.564	84.483
Fucose	108.599	57.445
Fumarate	10.922	4.552
Glucose	827.219	140.958
Glutamine	391.410	174.427
Glycine	1069.378	585.149
Glycolate	219.270	138.984
Guanidoacetate	97.624	68.740
Hippurate	2875.730	1364.240
Histidine	364.232	180.472
Hypoxanthine	67.087	51.714
Isoleucine	9.661	7.218

Table 1: Mean Metabolite Values by Muscle Loss Status *(continued)*

	cachexic	control
Lactate	217.632	65.748
Leucine	31.262	13.557
Lysine	121.282	89.229
Methylamine	21.216	11.360
Methylguanidine	17.365	12.128
N,N-Dimethylglycine	34.490	13.597
O-Acetylcarnitine	25.564	10.598
Pantothenate	39.944	52.623
Pyroglutamate	270.292	119.258
Pyruvate	26.866	12.566
Quinolinate	83.747	39.324
Serine	245.830	122.263
Succinate	79.629	29.836
Sucrose	150.024	55.580
Tartrate	47.235	28.676
Taurine	655.720	320.522
Threonine	118.233	59.519
Trigonelline	359.638	130.687
Trimethylamine N-oxide	820.341	388.669
Tryptophan	81.824	41.833
Tyrosine	100.742	52.014
Uracil	37.514	32.493
Valine	45.583	20.133
Xylose	129.289	56.509
cis-Aconitate	276.026	91.724
myo-Inositol	181.838	62.641
trans-Aconitate	48.814	27.809
pi-Methylhistidine	441.553	258.640
tau-Methylhistidine	105.668	64.650

Dado que observamos diferencias en la media de los metabolitos entre ambos grupos, realizamos un Análisis de Componentes Principales (PCA) para explorar la estructura global de los datos, permitiendonos reducir la dimensionalidad del conjunto de datos y visualizar posibles patrones de agrupación entre las muestras.

Además, nos ayudará a identificar si la variabilidad en los metabolitos es suficiente para separar los grupos de caquexia y control, así como detectar posibles outliers que puedan influir en el análisis posterior.



En la representación de la PCA, observamos que la primera componente principal (PC1) explica el 40.4% de la variabilidad en los datos, lo que indica que esta dimensión captura una parte significativa de la estructura global del conjunto de datos. La segunda componente principal (PC2) explica el 8.2% de la variabilidad, lo que sugiere que, aunque contribuye a la representación de los datos, su impacto es menor. En conjunto, estas dos componentes explican el 48.6% de la variabilidad total, lo que significa que aún hay información relevante distribuida en componentes adicionales.

Aunque se observa cierta dispersión en ambas direcciones, no se aprecia una clara separación entre los grupos de caquexia y control, lo que indica que las diferencias metabólicas entre ambos no son lo suficientemente marcadas en estas dos dimensiones principales. Sin embargo, identificamos algunos posibles outliers en el grupo de caquexia, que podrían estar influyendo en la estructura global de los datos y merecen una inspección más detallada.

Para comprender mejor qué metabolitos están impulsando la variabilidad capturada en las primeras componentes principales, identificamos aquellos que más contribuyen a PC1 y PC2.

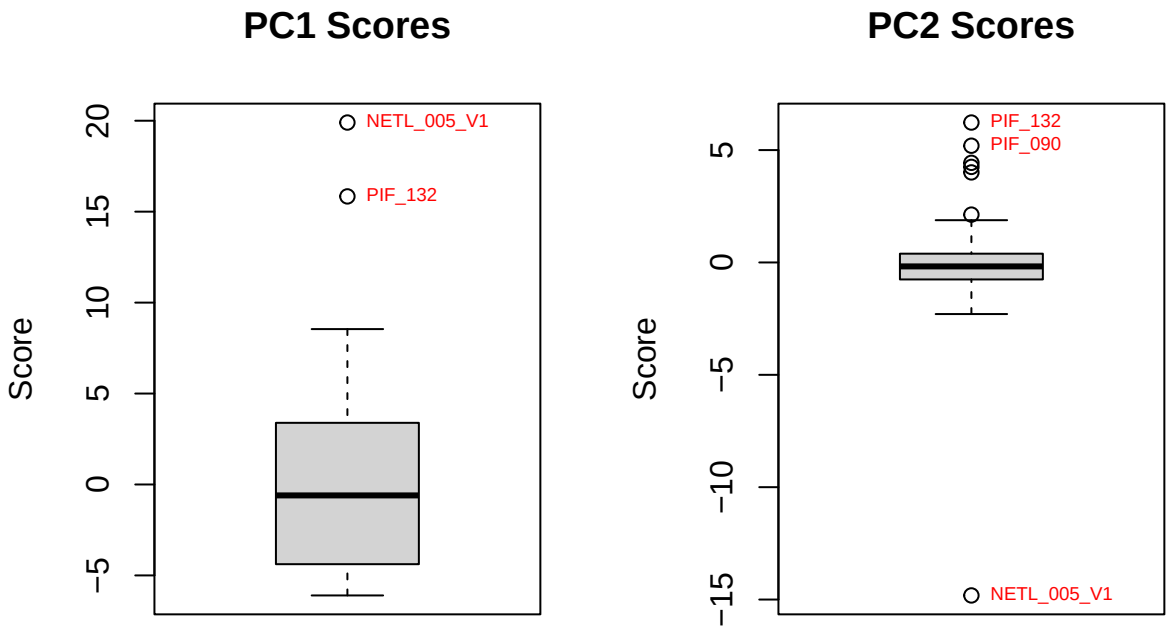
```
##                    metabolite PC1_loading PC2_loading
## Creatinine         Creatinine   0.1754974  0.09968692
## Glutamine           Glutamine   0.1708957 -0.13077088
## Ethanolamine     Ethanolamine   0.1704181  0.03505266
## Asparagine         Asparagine   0.1691602 -0.06978002
## Threonine           Threonine   0.1684597 -0.04124448
## Valine                 Valine   0.1679131  0.07611425
## Alanine               Alanine   0.1673433  0.10663523
## cis-Aconitate cis-Aconitate   0.1661170 -0.12691812
## Serine                 Serine   0.1649668 -0.15537386
## Fucose                 Fucose   0.1629741 -0.04466456
```

Dado que PC1 explica el mayor porcentaje de la variabilidad en los datos (40.4%), los metabolitos con cargas más altas en esta componente son los que más influyen en la diferenciación de las muestras en esta dimensión, lo que los hace especialmente importantes para entender la variabilidad principal. Sin embargo, PC2 también captura una parte significativa de la variabilidad adicional, lo que significa que los metabolitos con mayores contribuciones a esta componente pueden revelar patrones metabólicos complementarios, no completamente explicados por PC1.

Los metabolitos con las mayores cargas en PC1 incluyen la creatinina (0.1755), la glutamina (0.1709) y el etanolamina (0.1704), mientras que en PC2, los metabolitos como la alanina (0.1066), la valina (0.0761) y el etanolamina (0.0351) son los que más contribuyen a esta dimensión.

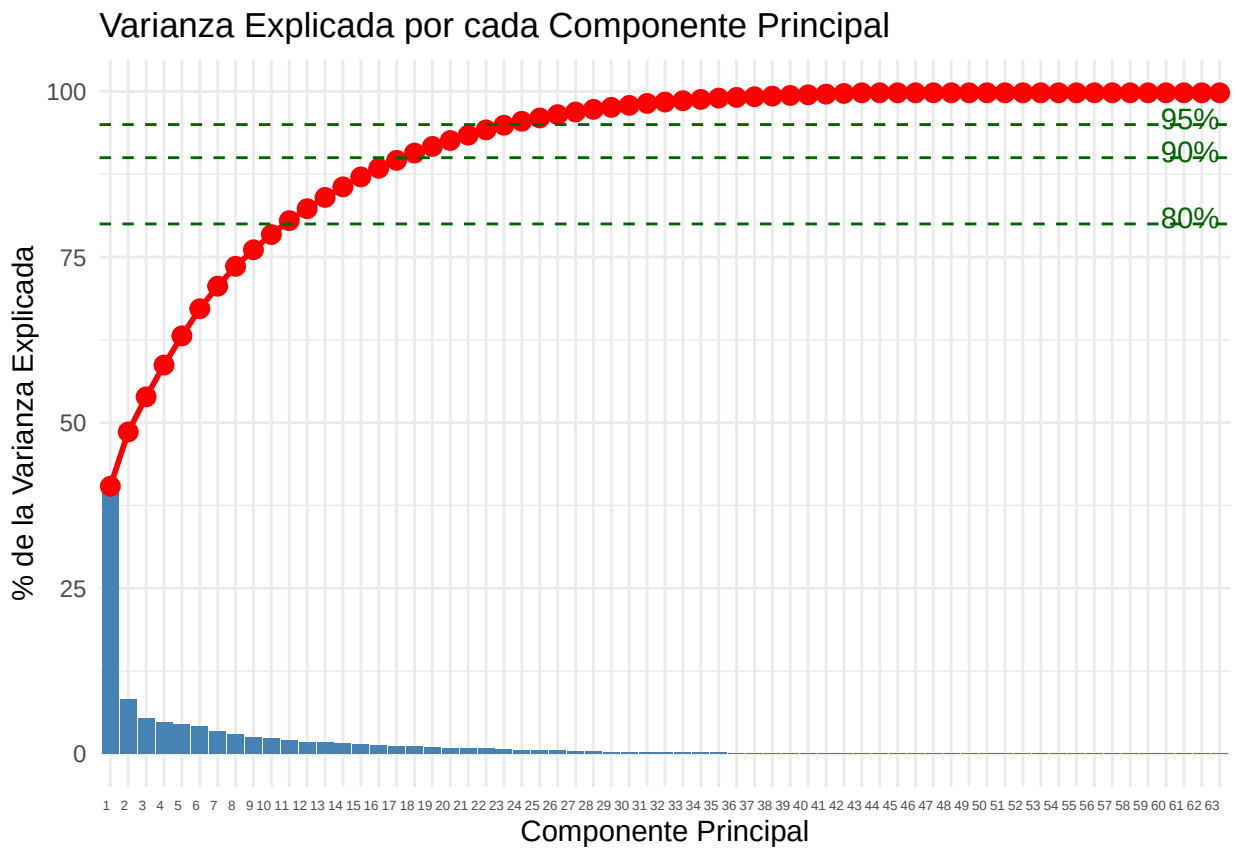
Otros metabolitos relevantes incluyen la asparagina, la treonina, el ácido cis-aconítico, la serina y la fucosa, todos con contribuciones destacadas tanto en PC1 como en PC2, lo que sugiere que podrían ser clave para entender los mecanismos metabólicos subyacentes en la caquexia.

A continuación, evaluamos la presencia de outliers en las primeras dos componentes principales (PC1 y PC2).



Utilizando un criterio basado en valores que superan las 2 desviaciones estándar, identificamos tres posibles outliers: PIF_132, NETL_005_V1 y PIF_090, como podemos comprobar en los boxplots.

Finalmente, evaluamos cuántos de los 63 componentes principales son necesarios para explicar el 80%, 90% y 95% de la varianza.




```
## PCs necesarios para el 80% de la varianza: 11
## PCs necesarios para el 90% de la varianza: 18
## PCs necesarios para el 95% de la varianza: 24
```

Observamos que para explicar el 80% de la varianza, se requieren 11 componentes principales, mientras que para alcanzar el 90% y el 95%, se necesitan 18 y 24 componentes, respectivamente. En el gráfico, se puede observar cómo, a partir del componente 30, la cantidad de varianza explicada por cada componente adicional disminuye considerablemente, indicando que los componentes posteriores tienen una contribución marginal a la explicación de la varianza.

5 Discusión

Este análisis revela un patrón metabólico característico de la caquexia, reflejado en la composición de los metabolitos con mayor contribución a la primera componente principal (PC1). Se observa un predominio de aminoácidos como glutamina, asparagina, treonina, valina, alanina y serina, lo que sugiere un proceso de catabolismo proteico asociado con la pérdida de masa muscular, un fenómeno ampliamente descrito en la fisiopatología de la caquexia. La degradación de proteínas musculares libera aminoácidos en la circulación, que pueden ser utilizados como sustratos energéticos o para la síntesis de otros compuestos esenciales durante condiciones de estrés metabólico [3].

Además, el aumento de creatinina, que muestra la mayor contribución a PC1 (loading = 0.175), refuerza la hipótesis de una mayor degradación del tejido muscular en pacientes con caquexia, ya que la creatinina es un marcador bien establecido del metabolismo muscular. La presencia de cis-aconitato, un intermediario del ciclo de Krebs, sugiere posibles alteraciones en el metabolismo energético mitocondrial, lo que es consistente con la disfunción metabólica descrita en la caquexia. Por otro lado, la contribución de etanolamina, un precursor

de fosfolípidos de membrana, podría indicar un aumento en la degradación de membranas celulares, otro proceso común en estados de desgaste sistémico. [4]

Aunque el análisis de componentes principales (PCA) no reveló una clara separación entre los grupos de caquexia y control, la identificación de ciertos outliers (PIF_132, NETL_005_V1 y PIF_090) sugiere la existencia de heterogeneidad metabólica dentro del grupo de caquexia. Esta heterogeneidad podría reflejar diferentes grados de progresión de la enfermedad o respuestas individuales al catabolismo, lo cual podría tener implicaciones para tratamientos personalizados. Sin embargo, es importante señalar que los outliers identificados podrían ser el resultado de errores en las muestras o en el proceso de análisis, por lo que se recomienda realizar una inspección más detallada de estas muestras para confirmar si realmente representan variaciones biológicas o si se deben a problemas técnicos.

Por otro lado, el uso de los primeros 11 componentes principales, que explican el 80% de la variabilidad en el análisis de PCA, puede ser particularmente útil para reducir la complejidad de los datos en análisis posteriores, como la clasificación de muestras o la identificación de patrones metabólicos específicos asociados con caquexia, no solo proporcionando una representación condensada de la variabilidad global del conjunto de datos, sino que también permitiendo centrar el análisis en las características metabólicas más relevantes, mejorando la precisión del modelo y minimizando el riesgo de sobreajuste (overfitting) o la pérdida de información significativa.

6 Conclusiones

Este análisis exploratorio ha identificado un patrón metabólico característico de la caquexia, marcado por un predominio de aminoácidos y creatinina, lo que sugiere una mayor degradación muscular y alteraciones en el metabolismo energético. Aunque no se observó una separación clara entre los grupos de caquexia y control, la heterogeneidad metabólica dentro del grupo de caquexia resalta la necesidad de considerar variaciones individuales en la progresión de la enfermedad. Los primeros componentes principales del PCA proporcionan una base sólida para futuros análisis y pueden ser útiles para la identificación de biomarcadores específicos, mejorando la precisión de modelos y evitando sobreajuste en análisis posteriores.

7 Referencias

[1] https://bioconductor.org/packages/release/bioc/vignettes/SummarizedExperiment/inst/doc/SummarizedExperiment.html

[2] https://www.sthda.com/english/wiki/expressionset-and-summarizedexperiment

[3] Ferrer M, Anthony TG, Ayres JS, Biffi G, Brown JC, Caan BJ, Cespedes Feliciano EM, Coll AP, Dunne RF, Goncalves MD, Grethlein J, Heymsfield SB, Hui S, Jamal-Hanjani M, Lam JM, Lewis DY, McCandlish D, Mustian KM, O'Rahilly S, Perrimon N, White EP, Janowitz T. Cachexia: A systemic consequence of progressive, unresolved disease. Cell. 2023 Apr 27;186(9):1824-1845. doi: 10.1016/j.cell.2023.03.028. PMID: 37116469; PMCID: PMC11059056.

[4] Pandhi P, Streng KW, Anker SD, et al. The value of spot urinary creatinine as a marker of muscle wasting in patients with new-onset or worsening heart failure. *J Cachexia Sarcopenia Muscle*. 2021;12(3):555-567. doi:10.1002/jcsm.12690

8 Apéndice: Código

A continuación se muestra todo el código R utilizado en este análisis:

```
library(readr)
library(SummarizedExperiment)

# Descargamos los datos de la URL
```

```r
url <- "https://raw.githubusercontent.com/nutrimetabolomics/metaboData/main/Datasets/2024-Cachexia/human
data <- read_csv(url)

# Extraemos la identificación del paciente y el estado de pérdida muscular
patient_ids <- data[[1]]
muscle_loss <- data[[2]]

# Creamos una matriz solo con los datos de metabolitos (columnas 3 a 65)
data_matrix <- as.matrix(data[, -(1:2)])

# Transponemos la matriz para tener los metabolitos como filas y los pacientes como columnas
data_matrix <- t(data_matrix)
colnames(data_matrix) <- patient_ids

# Creamos colData (información sobre los pacientes)
colData <- DataFrame(
    patient_id = patient_ids,
    muscle_loss = muscle_loss,
    row.names = patient_ids
)

  # Creamos rowData (información sobre metabolitos)
rowData <- DataFrame(
    metabolite = rownames(data_matrix),
    row.names = rownames(data_matrix)
)

# Creamos el objeto SummarizedExperiment
se <- SummarizedExperiment(
    assays = list(counts = data_matrix),
    rowData = rowData,
    colData = colData
)

se
# Extraemos la matriz de datos metabólicos del objeto SummarizedExperiment
# que contiene los valores de los 63 metabolitos para cada paciente
assay_data <- assay(se, "counts")

# Comprobamos si hay valores faltantes (NA) en el conjunto de datos
# - is.na() identifica dónde hay NA
# - any() detecta si existe al menos un NA
# - La negación (!) invierte el resultado: TRUE significa "datos completos"
has_complete_data <- !any(is.na(assay_data))

print(paste("El dataset no contiene missing values:", has_complete_data))
library(knitr)
library(kableExtra)
library(dplyr)

# Extraemos los datos y calculamos las medias por grupo
a <- as.data.frame(aggregate(t(assay_data), list(Group=colData(se)$muscle_loss), mean))
```

```r
# Extraemos los nombres de los grupos
groups <- a$Group
# Eliminamos la columna de grupos antes de transponer
a$Group <- NULL

# Transponemos los datos y conservamos los metabolitos como rownames
transposed_data <- as.data.frame(t(a))
colnames(transposed_data) <- as.character(groups)
transposed_data <- round(transposed_data, 3)

# Creamos una tabla formateada de múltiples páginas
kable(transposed_data,
      caption = "Mean Metabolite Values by Muscle Loss Status",
      format = "latex",
      longtable = TRUE,  # Habilitamos tablas de varias páginas
      booktabs = TRUE,
      row.names = TRUE) %>%  # Usamos los nombres de fila como identificadores de metabolitos
  kable_styling(font_size = 9,
                latex_options = c("repeat_header", "striped")) %>%
  column_spec(1, width = "8cm")
library(ggplot2)

# Perform PCA with proper scaling (corrected to avoid double scaling)
pca_result <- prcomp(t(assay_data), scale. = TRUE)

# Calculate variance explained by each principal component
variance_explained <- round(pca_result$sdev^2 / sum(pca_result$sdev^2) * 100, 1)

# Create a data frame for plotting
pca_df <- data.frame(
  PC1 = pca_result$x[, 1],
  PC2 = pca_result$x[, 2],
  muscle_loss = factor(muscle_loss),
  patient_id = patient_ids
)

# Create PCA plot
ggplot(pca_df, aes(x = PC1, y = PC2, color = muscle_loss)) +
  geom_point(size = 3) +
  geom_text(aes(label = patient_id), vjust = -0.5, size = 3, show.legend = FALSE) +
  labs(
    title = "PCA of Metabolomics Data",
    x = paste0("PC1 (", variance_explained[1], "% variance)"),
    y = paste0("PC2 (", variance_explained[2], "% variance)"),
    color = "Muscle Loss Status"
  ) +
  theme_minimal()
# Obtener los metabolitos que más contribuyen a PC1 y PC2
loadings <- pca_result$rotation  # Extrae la matriz de rotación del PCA
top_loadings <- data.frame(
  metabolite = rownames(loadings),  # Nombres de los metabolitos
  PC1_loading = loadings[, 1],      # Contribución a PC1
  PC2_loading = loadings[, 2]       # Contribución a PC2
```

```
)

# Ordenar por contribución absoluta a PC1
top_loadings_PC1 <- top_loadings[order(abs(top_loadings$PC1_loading), decreasing = TRUE), ][1:10, ]
print(top_loadings_PC1)
par(mfrow = c(1,2))

# Extraemos los valores de PC1 y PC2
pc1_values <- pca_result$x[,1]
pc2_values <- pca_result$x[,2]

# Calculamos los umbrales de outliers (más de 2 desviaciones estándar)
pc1_outliers <- which(abs(pc1_values) > 2 * sd(pc1_values))
pc2_outliers <- which(abs(pc2_values) > 2 * sd(pc2_values))

# Identificamos muestras que son outliers en PC1 o PC2
all_outliers <- unique(c(pc1_outliers, pc2_outliers))

# Obtenemos los nombres
outlier_labels_pc1 <- patient_ids[pc1_outliers]
outlier_labels_pc2 <- patient_ids[pc2_outliers]

# Boxplot de PC1
boxplot(pc1_values, main="PC1 Scores", ylab="Score")
text(x = 1, y = pc1_values[pc1_outliers], labels = outlier_labels_pc1,
     col="red", cex=0.6, pos=4, offset=0.5)

# Boxplot de PC2
boxplot(pc2_values, main="PC2 Scores", ylab="Score")
text(x = 1, y = pc2_values[pc2_outliers], labels = outlier_labels_pc2,
     col="red", cex=0.6, pos=4, offset=0.5)

par(mfrow = c(1,1))
# Calculamos la varianza acumulada explicada
cumulative_variance <- cumsum(variance_explained)

# Creamos un dataframe para graficar la varianza
variance_df <- data.frame(
  PC = factor(1:length(variance_explained),
              levels = 1:length(variance_explained)),
  Individual = variance_explained,
  Cumulative = cumulative_variance
)

# Determinamos el numero de PC necesarios para estos porcentages
pc_80 <- min(which(cumulative_variance >= 80))
pc_90 <- min(which(cumulative_variance >= 90))
pc_95 <- min(which(cumulative_variance >= 95))

# Creamos un grafico de sedimentacion con linea acumulativa
ggplot(variance_df, aes(x = PC)) +
  geom_bar(aes(y = Individual), stat = "identity", fill = "steelblue") +
  geom_line(aes(y = Cumulative, group = 1), color = "red", size = 1) +
```

```
  geom_point(aes(y = Cumulative), color = "red", size = 3) +
  geom_hline(yintercept = c(80, 90, 95), linetype = "dashed", color = "darkgreen") +
  annotate("text", x = length(variance_explained), y = c(81, 91, 96),
           label = c("80%", "90%", "95%"), color = "darkgreen", hjust = 1) +
  labs(
    title = "Varianza Explicada por cada Componente Principal",
    x = "Componente Principal",
    y = "% de la Varianza Explicada"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(size = 5, angle = 0, hjust = 1))

cat(sprintf("PCs necesarios para el 80%% de la varianza: %d\n", pc_80))
cat(sprintf("PCs necesarios para el 90%% de la varianza: %d\n", pc_90))
cat(sprintf("PCs necesarios para el 95%% de la varianza: %d\n", pc_95))
# Guardamos el objeto SummarizedExperiment en formato binario (.Rda)
save(se, file = "cachexia_data.rda")

# ------------------------------------------------------------------------
# 1. Exportamos la matriz de datos principal (mediciones de metabolitos)
# ------------------------------------------------------------------------
# Extraemos la matriz de datos del SummarizedExperiment
data_matrix <- as.data.frame(assay(se))

# Añadimos los nombres de metabolitos como primera columna
data_matrix <- cbind(metabolite = rownames(data_matrix), data_matrix)

# Exportamos en formato CSV
write.csv(data_matrix, file = "cachexia_measurements.csv", row.names = FALSE)

# ------------------------------------------------------------------------
# 2. Exportamos metadatos de las muestras (información de pacientes)
# ------------------------------------------------------------------------
# Extraemos los metadatos de las muestras (colData)
sample_metadata <- as.data.frame(colData(se))

# Exportamos los metadatos de las muestras a CSV
write.csv(sample_metadata, file = "sample_metadata.csv", row.names = FALSE)

# ------------------------------------------------------------------------
# 3. Exportamos los metadatos de características (información de metabolitos)
# ------------------------------------------------------------------------
# Creamos un data frame simple con una sola columna para los nombres de metabolitos
metabolite_list <- data.frame(metabolite = rownames(se))

# Exportamos a CSV
write.csv(metabolite_list, file = "metabolite_features.csv", row.names = FALSE)
```