# DISEÑO Y METODOLOGÍA DEL ESTUDIO

El estudio que se realizó en el artículo (del que se sacan los datos para trabajar) consistió en un diseño experimental basado en modelos in vitro e in vivo para investigar la respuesta inmunológica inducida por el entrenamiento de monocitos y macrófagos. Esto se llevó a cabo mediante estímulos específicos, con énfasis en la activación por hemo y otros porfirinoides. Se realizaron experimentos con células humanas, murinas y modelos de infección sistémica en ratones:

### 1. Preparación y Cultivo Celular

#### 1.1. Obtención y Tratamiento de Suero Humano

- Se recolectó sangre de voluntarios sanos siguiendo aprobaciones éticas.
- Se procesó el suero mediante centrifugación, filtración estéril y almacenamiento a -80°C.

#### 1.2. Aislamiento y Cultivo de Células

- **Células humanas:**
    - Se obtuvieron monocitos de sangre periférica a partir de buffy coats donados voluntariamente.
    - Se siguió un protocolo de aislamiento previamente establecido.
- **Células murinas:**
    - Se extrajo médula ósea de fémures y tibias de ratones C57BL/6J de 8 a 12 semanas.
    - Se lisaron eritrocitos con tampón específico y las células se almacenaron a -150°C o se usaron directamente.
    - Se diferenciaron macrófagos derivados de médula ósea en presencia de GM-CSF (20 ng/mL).

### 2. Modelos de Entrenamiento Inmunológico y Estimulación Celular

#### 2.1. Protocolo de Entrenamiento de Monocitos

- Adaptado del protocolo de Bekkering.
- Se incubaron monocitos con medio de cultivo (control) o estímulos específicos (hemo, β-glucano, porfirinas, etc.) durante 24 h.
- Para estudios de inhibición, los inhibidores específicos se añadieron 1 h antes de la estimulación.
- Tras 24 h, las células se lavaron y se dejó en cultivo hasta el día 6.

- En el día 6, se reestimularon con LPS (10 ng/mL), y 24 h después se recolectaron sobrenadantes para análisis de citoquinas.

### 2.2. Estímulos y Tratamientos Farmacológicos

- **Entrenamiento inmunológico con hemo y β-glucano:**
  - Se usaron concentraciones de 50 µM para hemo y porfirinas, y 1 µg/mL para β-glucano.
- **Uso de inhibidores de vías de señalización:**
  - EGCG (inhibidor de acetiltransferasas).
  - R406 (inhibidor de la tirosina quinasa Syk).
  - SP600125 (bloqueo de la vía JNK).
  - Rapamicina (bloqueo de mTOR).
  - N-acetilcisteína (scavenger de ROS).
- **Mediciones de ROS:**
  - Se incubaron células con H2DCFDA y se analizaron por citometría de flujo.

## 3. Experimentos In Vivo en Ratones

### 3.1. Modelos Experimentales y Procedimientos Éticos

- Se utilizaron ratones C57BL/6 y Rag2−/− bajo condiciones específicas.
- Todos los procedimientos fueron aprobados por los comités éticos correspondientes en Alemania y Portugal.

### 3.2. Protocolo de Entrenamiento con Hemo y LPS

- Se administró hemo (2 mg/kg) o vehículo por inyección intraperitoneal.
- Una semana después, se desafió con LPS (15 mg/kg) para evaluar la respuesta inflamatoria y la supervivencia.

### 3.3. Modelo de Sepsis Polimicrobiana

- Se usó una suspensión estandarizada de heces humanas para inducir peritonitis polimicrobiana.
- Se monitorearon parámetros clínicos y se aplicaron escalas de gravedad para evaluar el estado de los animales.

## 4. Análisis Bioinformáticos y Estadísticos

### 4.1. Cuantificación de Expresión Génica (RNA-seq)

- Se alinearon las lecturas con el transcriptoma de Ensembl v68 usando **Bowtie 1**.
- Se cuantificó la expresión génica con **MMSEQ**.
- Se identificaron genes diferencialmente expresados usando **DESeq** con criterios estrictos (FC > 2.5, $p < 0.05$, RPKM ≥ 5).

### 4.2. Análisis Epigenómico (ChIP-seq)

- Se alinearon secuencias al genoma hg19 con **bwa**.
- Se eliminaron lecturas duplicadas y de baja calidad.
- Se identificaron picos de H3K27ac con **MACS2** y se normalizaron los datos con **DESeq1**.

### 4.3. Análisis de Accesibilidad Cromatínica (snATAC-seq)

- Se procesaron datos con **Cell Ranger ATAC** y se alinearon al genoma mm10.
- Se filtraron células con enriquecimiento de TSS ≥ 8 y >1000 fragmentos únicos.
- Se agruparon accesibilidades cromatínicas mediante **Seurat** y se usó **UMAP** para visualizar clústeres celulares.
- Se infirieron conexiones promotor-enhancer con **Cicero** y se realizaron análisis de footprinting con **ArchR**.

### Conclusión

El diseño experimental combinó enfoques **in vitro** (cultivo de monocitos y macrófagos) e **in vivo** (modelos murinos de inflamación y sepsis) junto con técnicas avanzadas de análisis genómico y epigenético. Se evaluó el efecto del hemo y otros estímulos sobre la reprogramación de monocitos/macrófagos mediante estudios de expresión génica, accesibilidad cromatínica y respuesta inflamatoria.

# ANÁLISIS DE UN DATASET DE BULK RNA-SEQ PÚBLICO

### 1. Carga de datos y organización

En el primer bloque de código, se cargan varias bibliotecas esenciales para análisis de RNA-seq, visualización de datos y anotación genética. A continuación, se crea un filtro para seleccionar solo los archivos necesarios de un directorio, usando la función grep. Los archivos seleccionados corresponden a distintas condiciones experimentales (T0, 4h, 24h, etc.) y se almacenan en una lista llamada archivos.

### 2. Lectura y procesamiento de datos

A continuación, se leen los archivos de expresión de cada muestra usando read.table(). Los datos se almacenan en la lista expresion_data. Para cada archivo, se extraen el nombre de la muestra y la condición experimental correspondiente. Esto es crucial para estructurar la matriz de expresión correctamente.

### 3. Creación de la matriz de expresión final

En este paso, se crea una matriz de expresión final (expresion_final) que contiene los valores de expresión para cada gen y muestra, uniendo los datos por feature_id. Aquí se puede observar un head(expresión_final):

| | feature_id <chr> | T0_HD34 <int> | RPMI_4h_HD34 <int> | BG_4h_HD34 <int> | RPMI_24h_HD34 <int> | BG_24h_HD34 <int> | RPMI_4h_HD34 <int> | BG_d6_HD34 <int> |
|---|---|---|---|---|---|---|---|---|
| 1 | ENSG00000000003 | 3 | 0 | 1 | 2 | 0 | 0 | 3 |
| 2 | ENSG00000000005 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | ENSG00000000419 | 334 | 436 | 476 | 381 | 133 | 28 | 147 |
| 4 | ENSG00000000457 | 135 | 169 | 147 | 205 | 70 | 22 | 78 |
| 5 | ENSG00000000460 | 62 | 80 | 85 | 161 | 61 | 14 | 30 |
| 6 | ENSG00000000938 | 4995 | 4307 | 9446 | 2760 | 713 | 569 | 2649 |

### 4. Filtrado de genes sin expresión

El código filtra los genes que no tienen expresión en ninguna muestra, es decir, aquellos cuya suma de cuentas es cero en todas las muestras. Esta filtración asegura que solo se mantengan los genes relevantes para el análisis posterior.

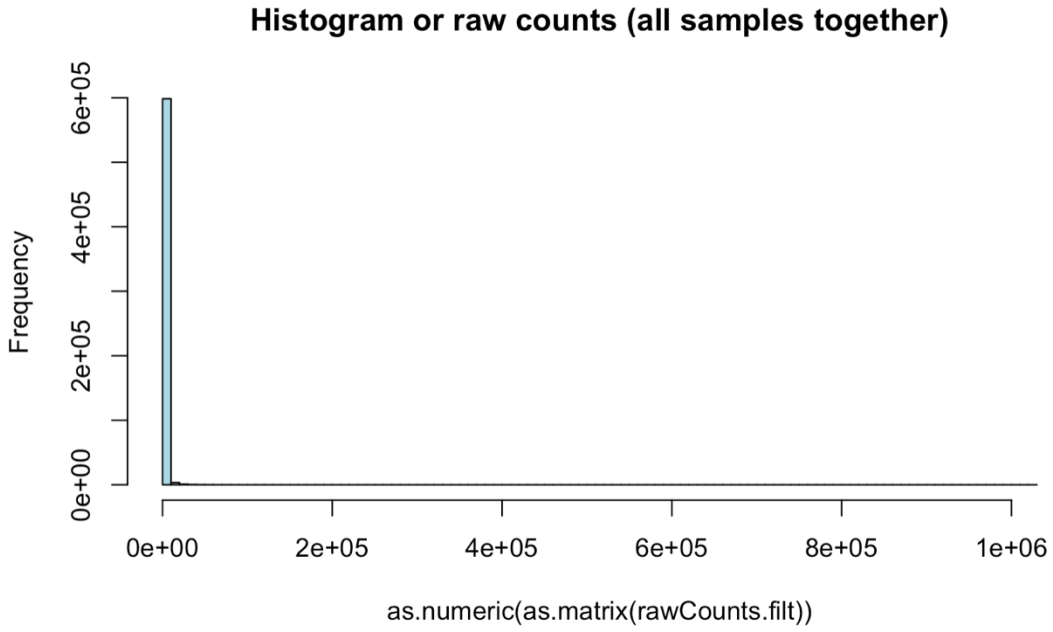
### 5. Filtrado adicional por baja expresión

Se realiza otro filtrado, eliminando genes cuya expresión es baja en la mayoría de las muestras (es decir, genes cuya expresión no supera el valor de 1 en más de una muestra). Este paso es crucial para eliminar el "ruido" en los datos y centrarse en los genes de interés.

### 6. Histograma de los datos crudos

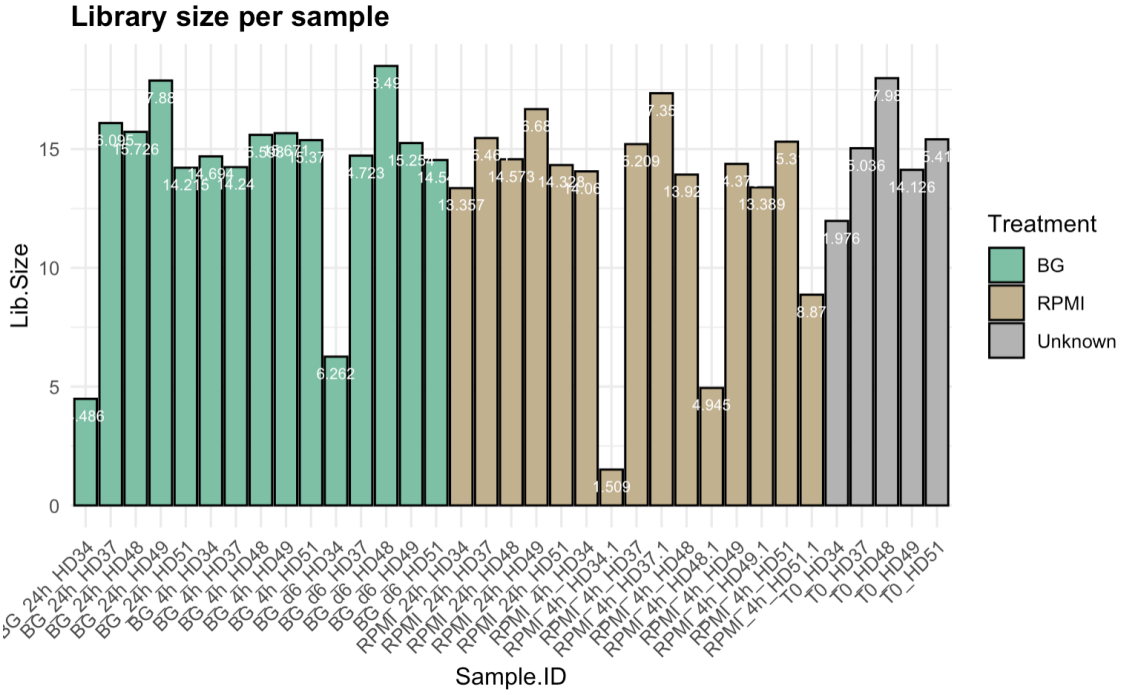
Se plotea un histograma de los valores de expresión cruda para ver la distribución de las cuentas por muestra.

Como se observa, la mayoría de las cuentas están cerca de cero, lo que indica que los datos no siguen una distribución normal, lo cual es esperado en datos de RNA-seq.



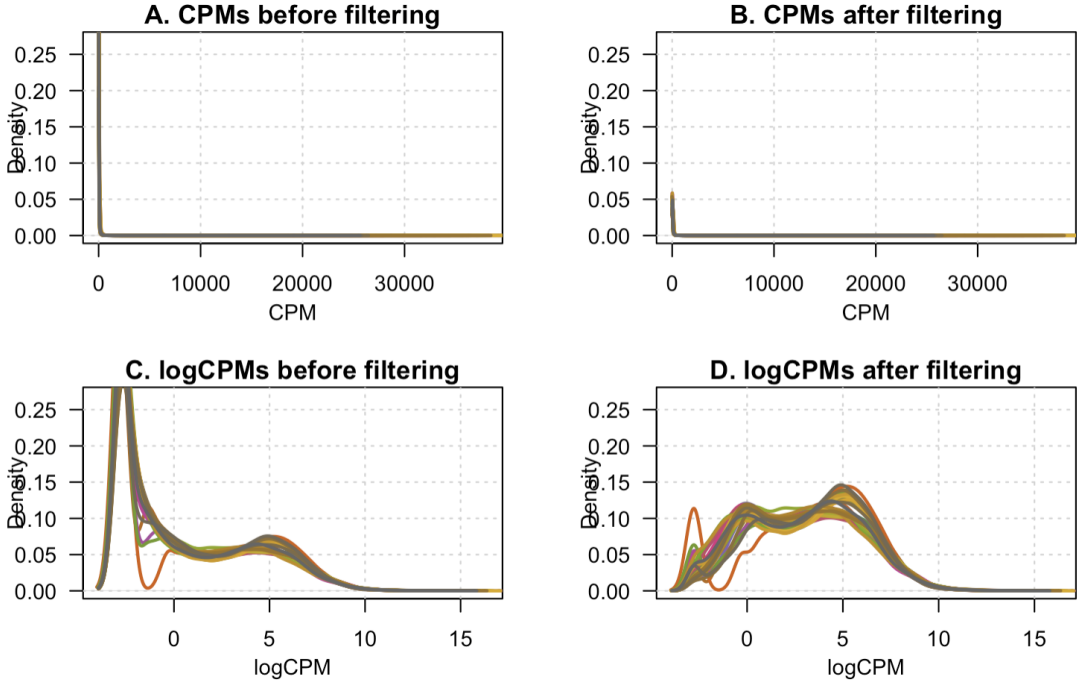

## 7. Análisis de tamaño de biblioteca

Se calcula el tamaño de las bibliotecas para cada muestra y se visualiza en un gráfico de barras. Este paso es útil para ver si las bibliotecas tienen tamaños muy dispares, lo que podría afectar el análisis de normalización.



## 8. Normalización intra-muestral (CPM)

Se calcula la expresión normalizada por millón de cuentas (CPM) antes y después del filtrado de genes. Esto permite comparar las expresiones entre diferentes muestras ajustando las diferencias en los tamaños de las bibliotecas.



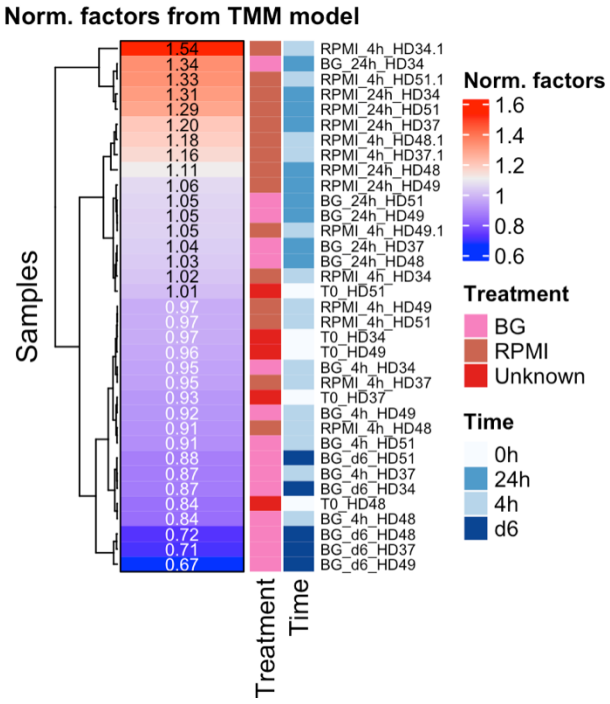

Se observa como al hacer un logaritmo tiende más a una distribución más normal(no del todo).

## 9. Normalización inter-muestral (TMM)

Se realiza la normalización entre muestras utilizando el método TMM (Trimmed Mean of M-values). Este método ajusta los tamaños de las bibliotecas y las diferencias sistemáticas entre las muestras, lo que permite una comparación más justa de la expresión génica entre diferentes condiciones.

| | T0_HD34 | RPMI_4h_HD34 | BG_4h_HD34 | RPMI_24h_HD34 | BG_24h_HD34 | RPMI_4h_HD34.1 | BG_d6_HD34 | T0_HD37 | RPMI_4h_HD37 | BG_4h_HD37 |
|---|---|---|---|---|---|---|---|---|---|---|
| ENSG00000000419 | 4.8577257 | 4.9314947 | 5.1010510 | 4.4540829 | 4.4777772 | 3.605263 | 4.763287 | 4.7679204 | 5.1711221 | 5.0791044 |
| ENSG00000000457 | 3.5616078 | 3.5751575 | 3.4197063 | 3.5682213 | 3.5603418 | 3.262097 | 3.855927 | 3.7019235 | 3.5300596 | 3.5436639 |
| ENSG00000000460 | 2.4600398 | 2.5159200 | 2.6438473 | 3.2245493 | 3.3644533 | 2.622631 | 2.500725 | 2.4505634 | 2.5989069 | 2.4704202 |
| ENSG00000000938 | 8.7534303 | 8.2295012 | 9.4058254 | 7.3025012 | 6.8924684 | 7.933483 | 8.927432 | 8.7817082 | 8.6610616 | 8.5205215 |
| ENSG00000000971 | -0.1129007 | 0.6871579 | 0.4519757 | 1.0636655 | 0.8572813 | 2.719863 | 2.403871 | -0.3322576 | -0.2467431 | -0.6654248 |
| ENSG00000001036 | 3.6634587 | 4.3167823 | 4.3347228 | 5.2706828 | 5.6218153 | 5.963648 | 6.747568 | 3.8602470 | 4.2683459 | 4.3128185 |
| ENSG00000001084 | 4.5578744 | 5.0814824 | 6.8627397 | 5.7404174 | 6.1674249 | 7.509416 | 7.063166 | 4.9433113 | 5.0968460 | 5.3983973 |
| ENSG00000001167 | 5.8965075 | 6.1048618 | 6.2266209 | 5.5029320 | 5.6266829 | 5.553049 | 5.993732 | 6.0581778 | 6.3599201 | 6.4460326 |
| ENSG00000001460 | 0.7672767 | 0.6871579 | 0.5257753 | 0.7076111 | 1.2084470 | 2.150950 | 1.788593 | 0.8484538 | 0.4109649 | 0.2559519 |
| ENSG00000001461 | 4.7460777 | 3.8850360 | 4.3035902 | 4.6734743 | 4.5916905 | 4.719161 | 4.423687 | 4.4686974 | 3.6706235 | 3.7215022 |

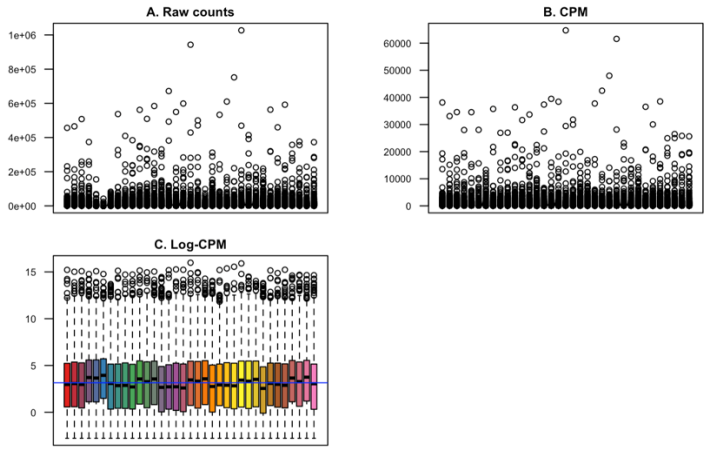
Aquí tenemos una vista parcial de los datos normalizados.




En este heatmap podemos observar que la mayoría de las muestras con tratamiento BG tienen más expresión total porque se han normalizado hacia abajo, porque un **menor factor de normalización** implica que una muestra tiene más "expresión total" por unidad de secuencia (es decir, tiene una biblioteca más grande en comparación con las otras), por lo que se ha normalizado hacia abajo. Esto es común en muestras con una alta cantidad de secuenciación inicial

## 10. Visualización de la normalización con boxplots

Finalmente, se generan varios gráficos de boxplot para visualizar las distribuciones de las expresiones de los datos crudos, los CPM, los log-CPM y los TMM log-CPM. Estos gráficos permiten comparar cómo cambia la distribución de los datos tras cada paso de normalización.




### 11. Función plotPCA para Graficar los Resultados del PCA

El código define una función llamada plotPCA que tiene como objetivo crear un gráfico de dispersión (scatter plot) de los resultados de un análisis de componentes principales (PCA). La función toma varios parámetros, incluyendo:

- **pcaObject**: El objeto resultante del análisis PCA realizado previamente.
- **col.points**: La variable por la cual se categorizan los puntos (muestras) en el gráfico, como el tratamiento o el tiempo.
- **shape.points**: Un parámetro opcional que permite diferenciar las muestras por su forma, aunque si no se define, se usa el valor predeterminado.
- **palette**: Una paleta de colores para las categorías de las muestras.
- **legend.col**: El título de la leyenda que describe la variable de categorización (por ejemplo, "Treatment").
- **point.size**: El tamaño de los puntos en el gráfico.
- **title**: El título del gráfico.
- **pcs**: Los números de los componentes principales que se desean mostrar en el gráfico (por defecto, los primeros dos).

La función extrae la varianza explicada por los componentes principales seleccionados (PC1 y PC2), calcula las coordenadas de las muestras en los componentes principales y luego genera el gráfico con **ggplot2**. El gráfico incluye puntos con colores y formas correspondientes a las categorías especificadas y una leyenda para cada variable.

### 12. Realización de la PCA

En esta sección, se realiza el **Análisis de Componentes Principales (PCA)** utilizando la función prcomp. El código realiza lo siguiente:

- Primero, se **escala** los datos utilizando la función scale() aplicada a los valores transpuestos de lcpmTMM, lo cual asegura que las variables (genes) tengan la misma importancia en el análisis.
- Luego, se aplica la función prcomp sobre los datos escalados, lo que devuelve un objeto con la descomposición de los componentes principales.
- Finalmente, se imprime un resumen de la PCA con la función summary, que muestra la varianza explicada por cada componente principal.

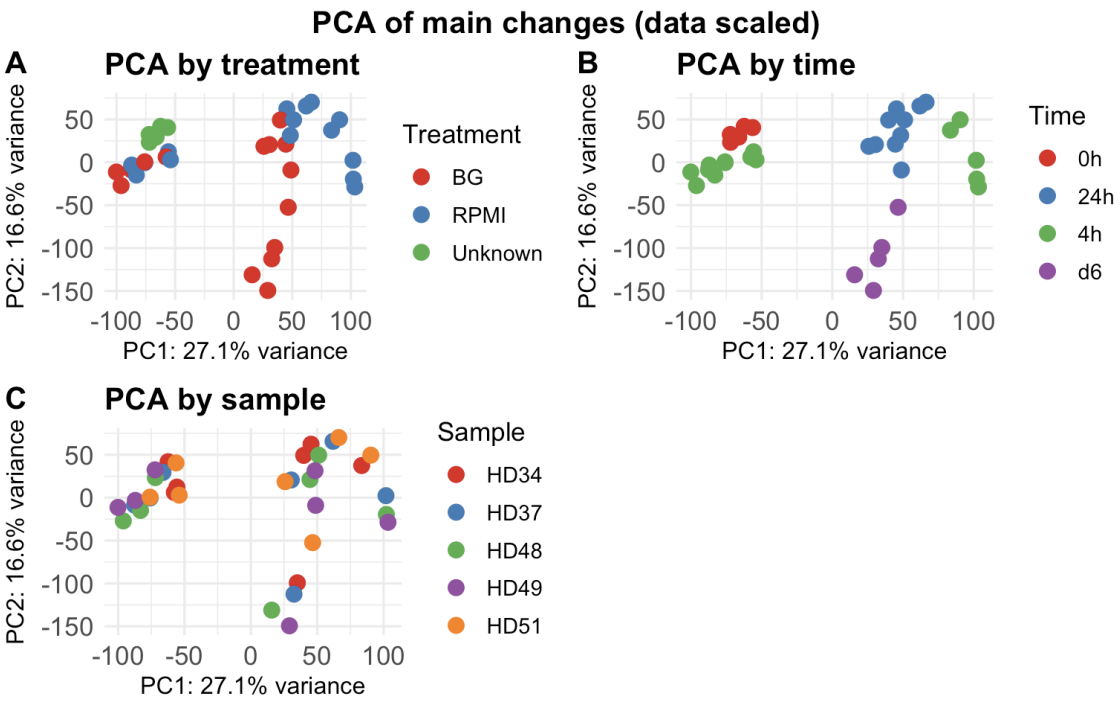
Este paso nos ayuda a entender la distribución de los datos en el espacio de los componentes principales, y qué proporción de la variabilidad es capturada por cada componente.

### 13. PCA por Categorías: Tratamiento, Tiempo y Muestra

En esta parte del código, se generan tres gráficos de dispersión para visualizar los resultados del PCA, categorizando las muestras por tres variables diferentes:

- **PCA por Tratamiento**: Se usa la variable "Treatment" de samplesMetadata para asignar colores a las muestras. Este gráfico permite ver cómo se agrupan las muestras según el tratamiento recibido.
- **PCA por Tiempo**: Se usa la variable "Time" para categorizar las muestras por el tiempo experimental.
- **PCA por Muestra**: Se utiliza la variable "Sample" para visualizar cómo se distribuyen las muestras individuales en el espacio de los componentes principales.

Cada gráfico se crea utilizando la función plotPCA, que es la misma que se describió anteriormente, y se les asigna un título adecuado. Estos gráficos ayudan a visualizar las diferencias y similitudes entre las muestras en función de su tratamiento, tiempo o identidad.



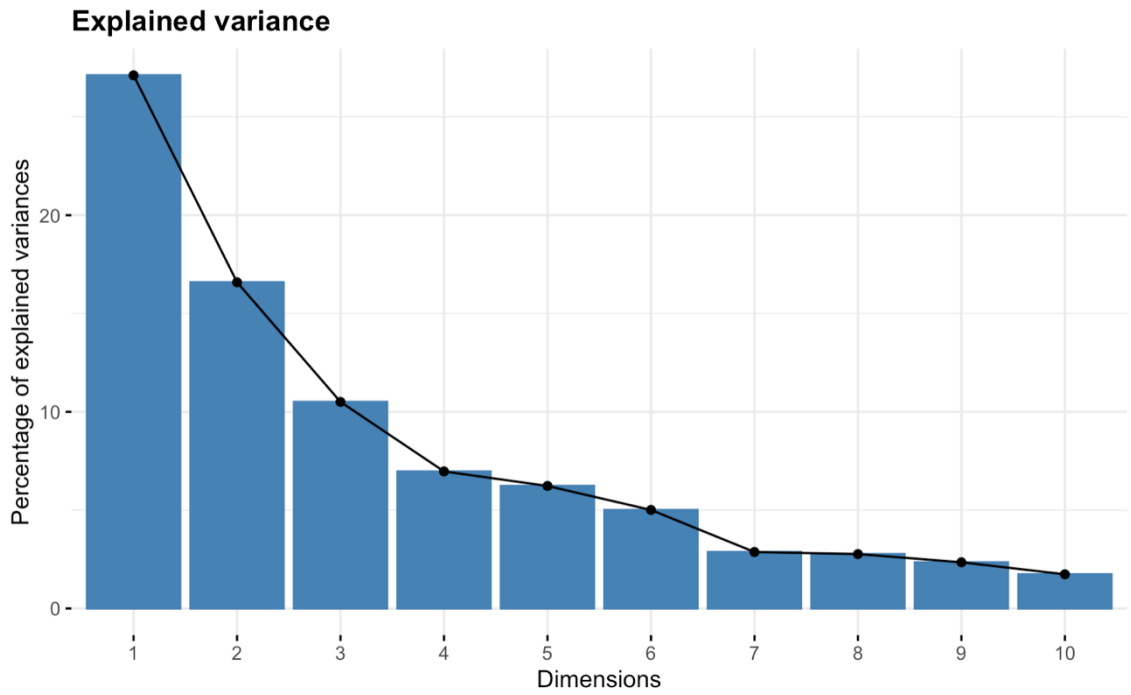

Podemos observar que en la PCA por tiempo hay una gran diferencia en el principal componente 1 entre los tiempos 24 horas y d6.

En la PCA por tratamiento observamos en el principal componente 1 un diferencia clara entre los tratados y los no tratados y en el componente principal 2 una diferencia de los tratados por RPMI y los no tratados respecto a los tratados por BG

### 14. Visualización de la Varianza Explicada por los Componentes Principales

El siguiente paso es visualizar la cantidad de **varianza explicada** por cada uno de los componentes principales. Para esto, se utiliza la función fviz_eig de la librería **factoextra**, que genera un gráfico de barras mostrando la proporción de la varianza explicada por cada componente. Este gráfico permite evaluar qué tan representativos son los componentes principales.

El código incluye un título personalizado para este gráfico ("Explained variance") y un formato adecuado para la visualización.




**15. Identificación de los Genes Más Influyentes en los Primeros Componentes Principales**

En esta parte, el código extrae los **genes más influyentes** en los primeros dos componentes principales (PC1 y PC2) utilizando la matriz de "rotación" del objeto PCA, que contiene los coeficientes de cada gen para cada componente.

Para cada componente (PC1 y PC2), se seleccionan los **10 genes con mayor contribución positiva** (los que tienen el mayor valor absoluto en los coeficientes) y los **10 genes con mayor contribución negativa** (aquellos con los coeficientes más bajos). Los genes seleccionados son aquellos que tienen una mayor influencia en la variabilidad representada por esos componentes.

El resultado es un conjunto de **genes relevantes** para la variabilidad observada en los primeros dos componentes principales:

```
      PC1                PC2
 [1,] "ENSG00000164081" "ENSG00000123329"
 [2,] "ENSG00000160703" "ENSG00000198740"
 [3,] "ENSG00000122971" "ENSG00000181800"
 [4,] "ENSG00000137491" "ENSG00000172932"
 [5,] "ENSG00000124574" "ENSG00000160219"
 [6,] "ENSG00000117305" "ENSG00000173020"
 [7,] "ENSG00000160446" "ENSG00000010295"
 [8,] "ENSG00000157353" "ENSG00000231259"
 [9,] "ENSG00000154079" "ENSG00000023892"
[10,] "ENSG00000140688" "ENSG00000100354"
[11,] "ENSG00000124789" "ENSG00000124145"
[12,] "ENSG00000213079" "ENSG00000115216"
[13,] "ENSG00000109670" "ENSG00000106701"
[14,] "ENSG00000244462" "ENSG00000126767"
[15,] "ENSG00000160201" "ENSG00000196371"
[16,] "ENSG00000161547" "ENSG00000188706"
[17,] "ENSG00000230551" "ENSG00000179294"
[18,] "ENSG00000015479" "ENSG00000232810"
[19,] "ENSG00000058673" "ENSG00000107175"
[20,] "ENSG00000162664" "ENSG00000196262"
```

### 16. Visualización de la Expresión de los Genes Más Influyentes

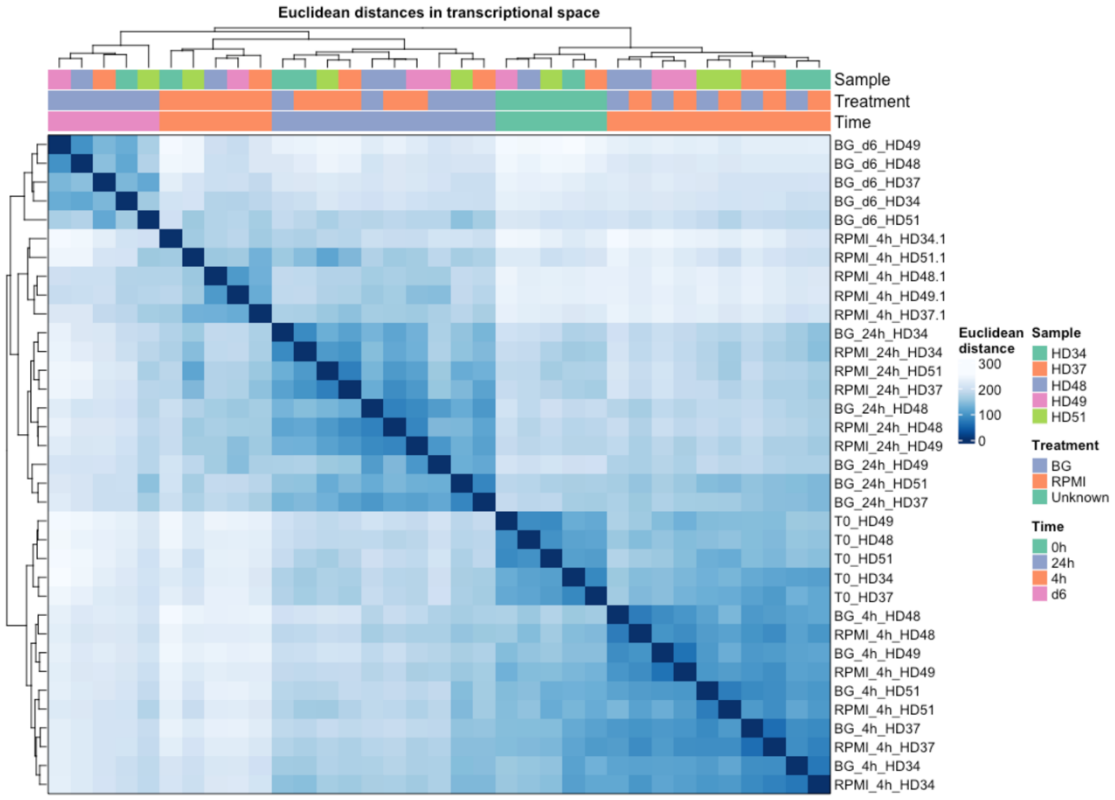
Una vez identificados los genes más influyentes, el código extrae los datos de expresión correspondientes a esos genes. Utiliza la matriz de expresión lcpmTMM para seleccionar las columnas correspondientes a los genes más influyentes, y luego realiza una transposición y una estadística descriptiva de los valores. Esto permite visualizar cómo varían estos genes a través de las muestras y obtener un resumen de sus valores de expresión.

```
 ENSG00000164081 ENSG00000160703  ENSG00000122971 ENSG00000137491   ENSG00000124574
 Min.   :2.475   Min.   :0.4799   Min.   :1.155   Min.   :-2.2046   Min.   :3.542
 1st Qu.:3.178   1st Qu.:1.8511   1st Qu.:1.996   1st Qu.:-0.7147   1st Qu.:4.035
 Median :4.373   Median :3.1343   Median :3.552   Median : 5.2331   Median :4.950
 Mean   :4.122   Mean   :2.7812   Mean   :3.181   Mean   : 3.5965   Mean   :4.764
 3rd Qu.:4.978   3rd Qu.:3.7458   3rd Qu.:4.069   3rd Qu.: 7.1502   3rd Qu.:5.421
 Max.   :5.697   Max.   :4.7822   Max.   :5.131   Max.   : 8.4236   Max.   :6.065
```

### 17. Creación de un Heatmap con Distancias Euclidianas

El código genera un **heatmap** que visualiza las distancias euclidianas entre las muestras, lo que proporciona una visión sobre cuán similares o diferentes son las muestras en términos de sus perfiles de expresión génica.

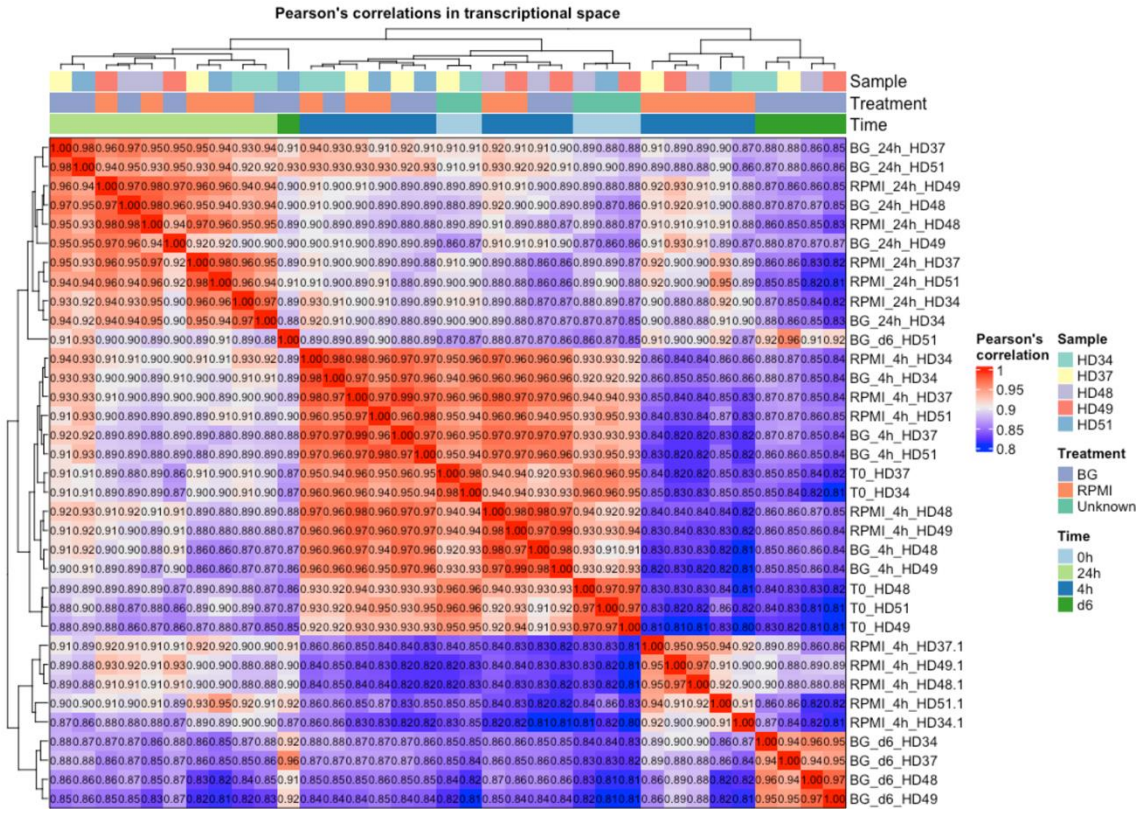
- Primero, se calcula la **distancia euclidiana** entre las muestras utilizando la función dist() en la transposición de los datos de expresión lcpmTMM.
- Luego, se crea un objeto de anotación (HeatmapAnnotation) que agrega información sobre el tratamiento y el tiempo de cada muestra, utilizando la librería **dplyr** para seleccionar las variables relevantes de samplesMetadata.
- Finalmente, se genera el **heatmap** utilizando la función Heatmap de la librería **ComplexHeatmap**. Este heatmap visualiza las distancias entre las muestras, con colores representando las distancias más cercanas (claro) o más distantes (oscuro).



## 18. Creación de un Heatmap con la Correlación de Pearson

De manera similar al heatmap anterior, pero en lugar de distancias, se calcula la **correlación de Pearson** entre las muestras, que mide la relación lineal entre sus perfiles de expresión.

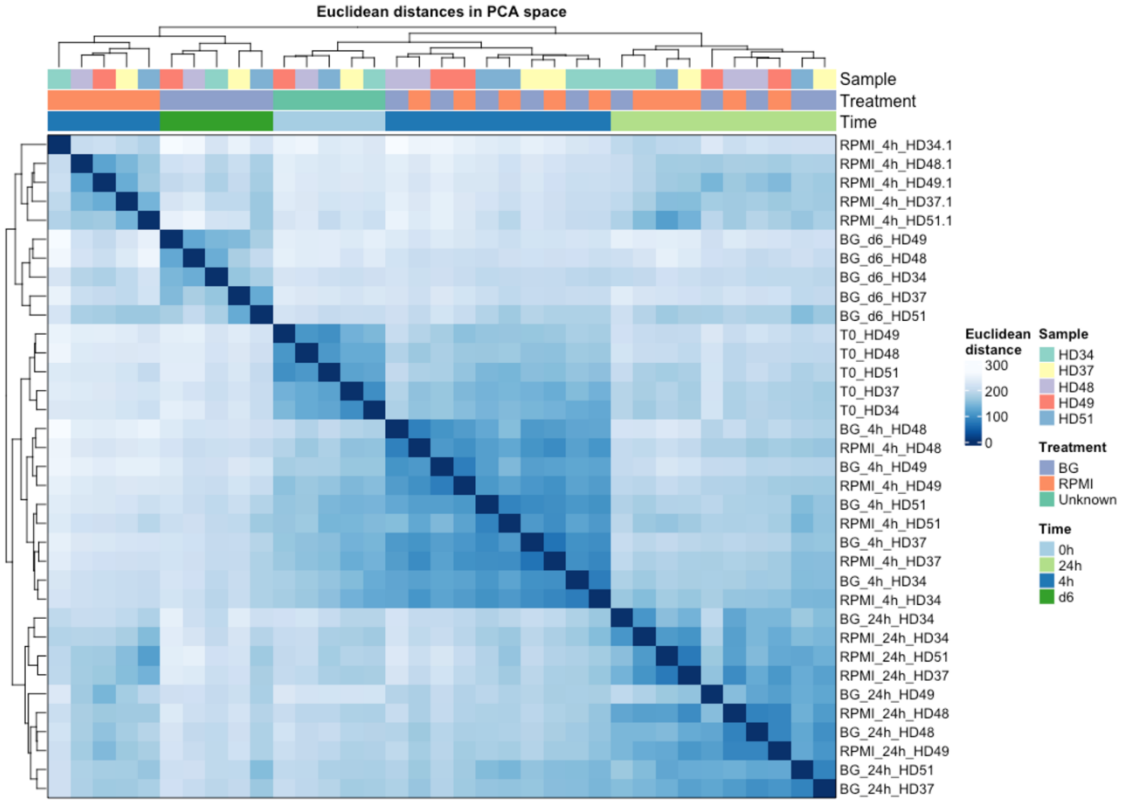
- Se calcula la matriz de **correlación de Pearson** usando la función cor() aplicada a los datos de expresión.
- Se genera un **heatmap** para visualizar estas correlaciones, donde los valores cercanos a 1 indican alta similitud entre las muestras, y los valores cercanos a -1 indican alta disimilitud.
- En el heatmap, se muestra la correlación numérica dentro de cada celda usando la opción cell_fun.




## 19. Creación de un Heatmap en el Espacio PCA

En este paso, el código calcula las **distancias euclidianas** entre las muestras, pero en el espacio reducido de la PCA (es decir, utilizando las coordenadas de las muestras en los componentes principales). Este enfoque permite visualizar las relaciones entre las muestras en un espacio de menor dimensión, lo que puede proporcionar una mejor comprensión de las diferencias y similitudes entre las muestras.

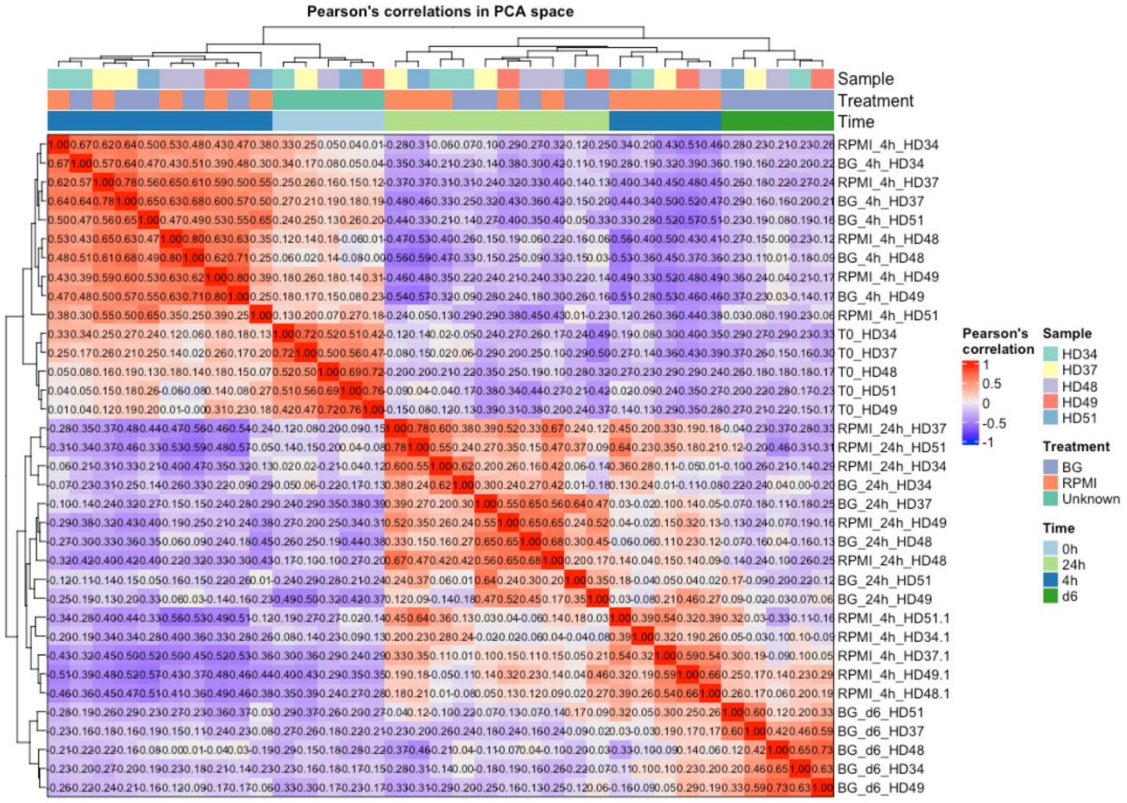
- Se calculan las distancias euclidianas entre las puntuaciones de las muestras en el espacio de la PCA utilizando la función dist() sobre PCA.scaled$x.
- Después, se genera el **heatmap** similar al anterior, visualizando las distancias entre las muestras en este nuevo espacio de componentes principales.




### 20. Creación de un Heatmap con Correlación de Pearson en el Espacio PCA

Finalmente, se calcula la **correlación de Pearson** entre las muestras, pero en el espacio de los componentes principales (es decir, sobre las puntuaciones de los componentes principales).

- Se calcula la matriz de correlación de Pearson utilizando las puntuaciones de las muestras en el espacio de la PCA (PCA.scaled$x).
- Luego, se genera un **heatmap** similar al anterior, donde se visualizan las correlaciones entre las muestras en el espacio reducido de la PCA.




### 21. Creación de la columna condition en samplesMetadata

En este paso, se agrega una nueva columna llamada condition al objeto samplesMetadata. Esta columna combina los valores de Treatment y Time, separados por un guion bajo (_). De esta manera, se facilita la identificación de cada condición experimental en los análisis posteriores.

### 22. Construcción del modelo de diseño y análisis de expresión diferencial

Se cargan las librerías limma y edgeR, necesarias para el análisis de expresión diferencial. Luego, se construye una matriz de diseño (design), la cual representa las distintas condiciones experimentales de la muestra. Esta matriz servirá como base para ajustar un modelo lineal mediante la función lmFit(), que modela la expresión génica en función de las condiciones experimentales.

### 23. Definición de contrastes y ajuste del modelo

Se establece una matriz de contrastes con la función makeContrasts(). En este caso, se comparan varias condiciones experimentales, como conditionUnknown_0h frente a conditionBG_4h, conditionUnknown_0h frente a conditionBG_24h, entre otras. Posteriormente, se aplican estos contrastes al modelo ajustado con contrasts.fit().

### 24. Aplicación de corrección Bayesiana y obtención de genes diferencialmente expresados

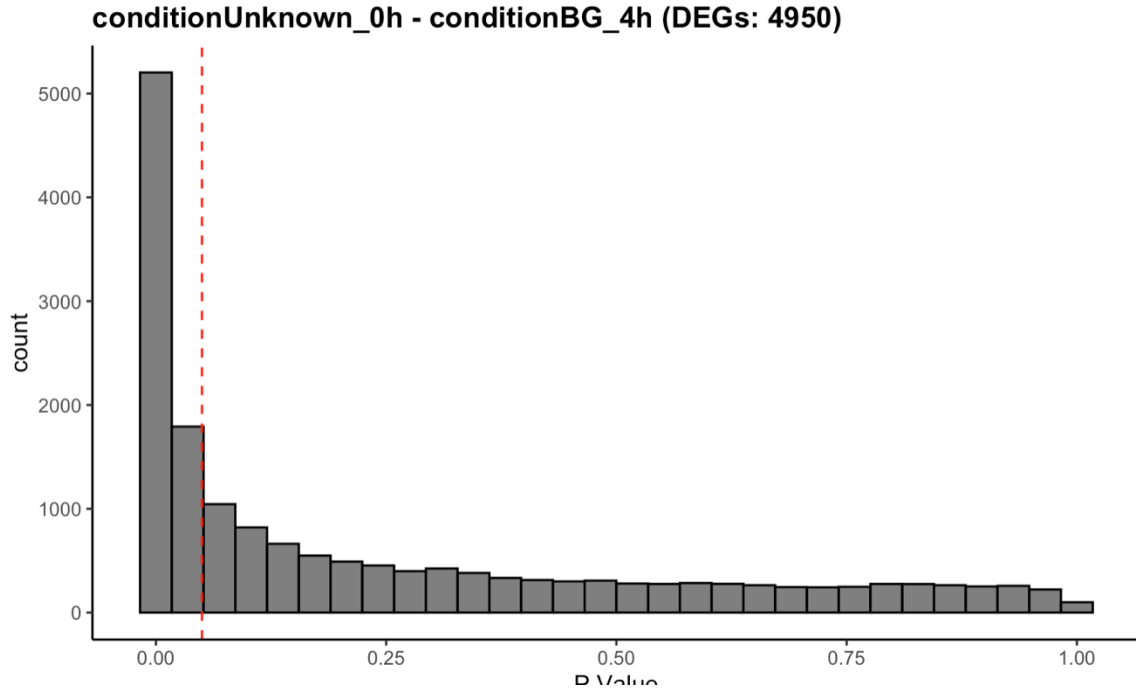
Se aplica la corrección Bayesiana con la función eBayes(), lo que permite mejorar la estimación de los estadísticos del modelo. Luego, se extraen los genes diferencialmente expresados (DEGs) con la función topTable(), utilizando un coeficiente específico (por ejemplo, Unknown_0hvs_RPMI_4h). Finalmente, los resultados se guardan en archivos CSV para futuras consultas.

### 25. Determinación del número de genes significativamente expresados

Se filtran los resultados de los genes diferencialmente expresados (DEGs), seleccionando aquellos con un valor de adj.P.Val menor o igual a 0.05. El número total de estos genes se almacena y se imprime en la consola.

### 26. Generación de histogramas de valores p

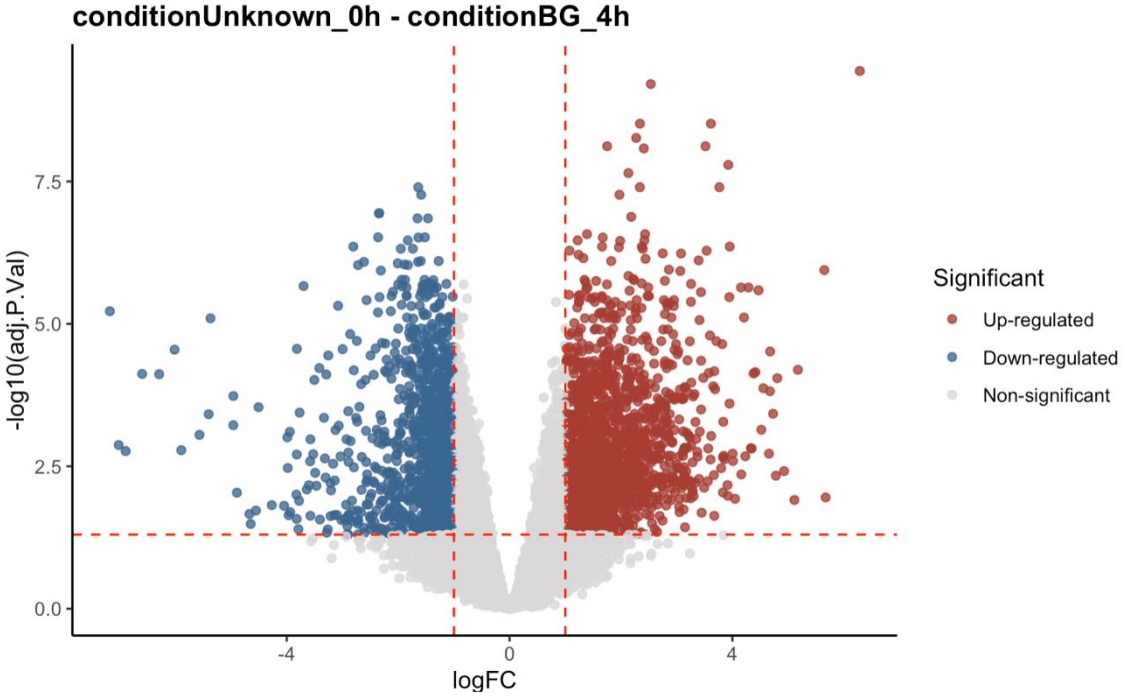
Se genera un histograma de valores p (P.Value) para visualizar la distribución de significancia estadística de los genes expresados diferencialmente. Además, se añade una línea roja discontinua en 0.05 para indicar el umbral de significancia.




### 27. Creación del gráfico de volcán (Volcano Plot)

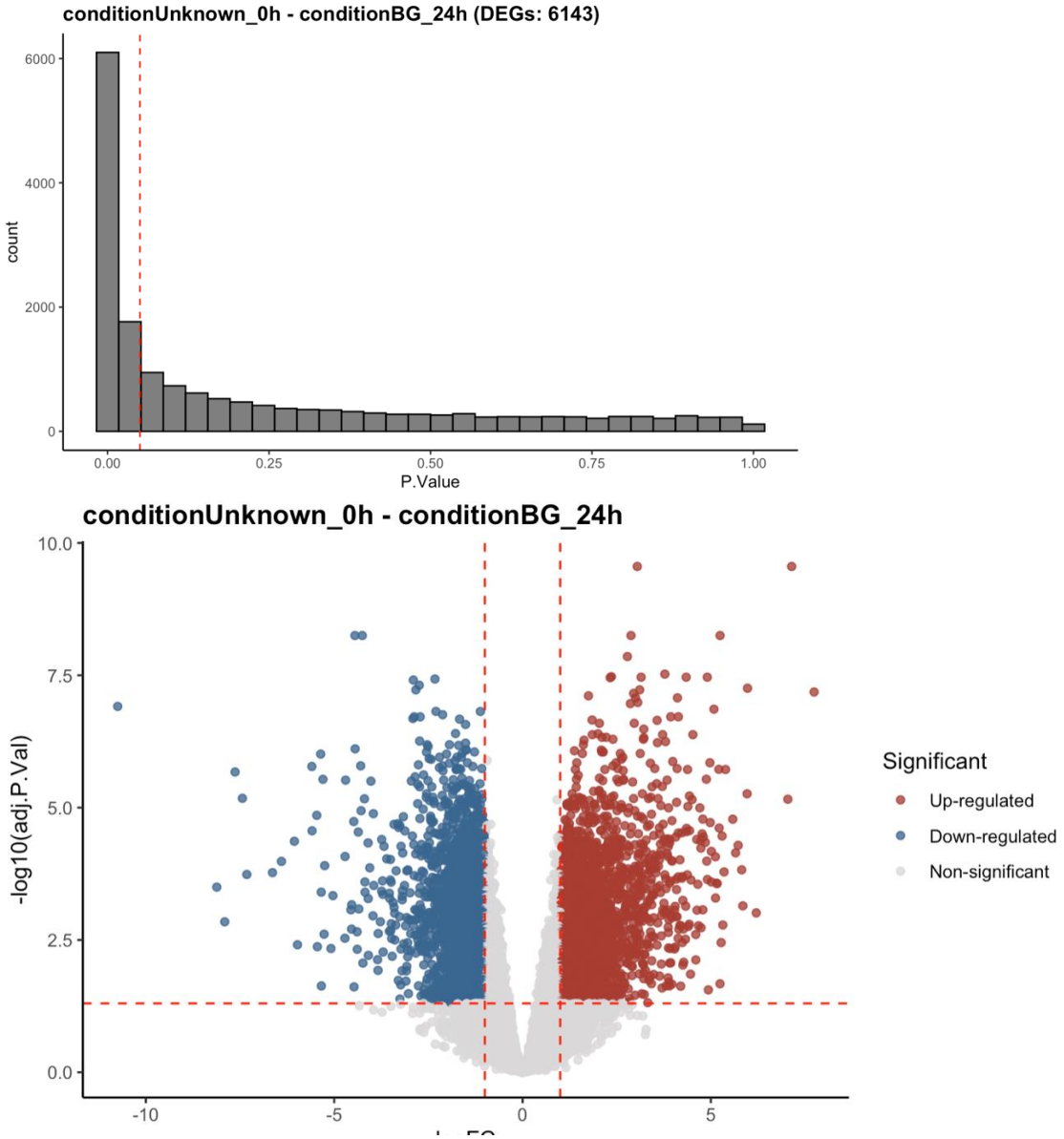
Se genera un gráfico de volcán donde se representa la relación entre el logFC (logaritmo de la razón de cambio) y la -log10 del valor ajustado de p (adj.P.Val). Los genes significativamente sobreexpresados (Up-regulated) y subexpresados (Down-regulated) se colorean de manera diferenciada, mientras

que los genes sin cambios significativos (Non-significant) se muestran en gris.



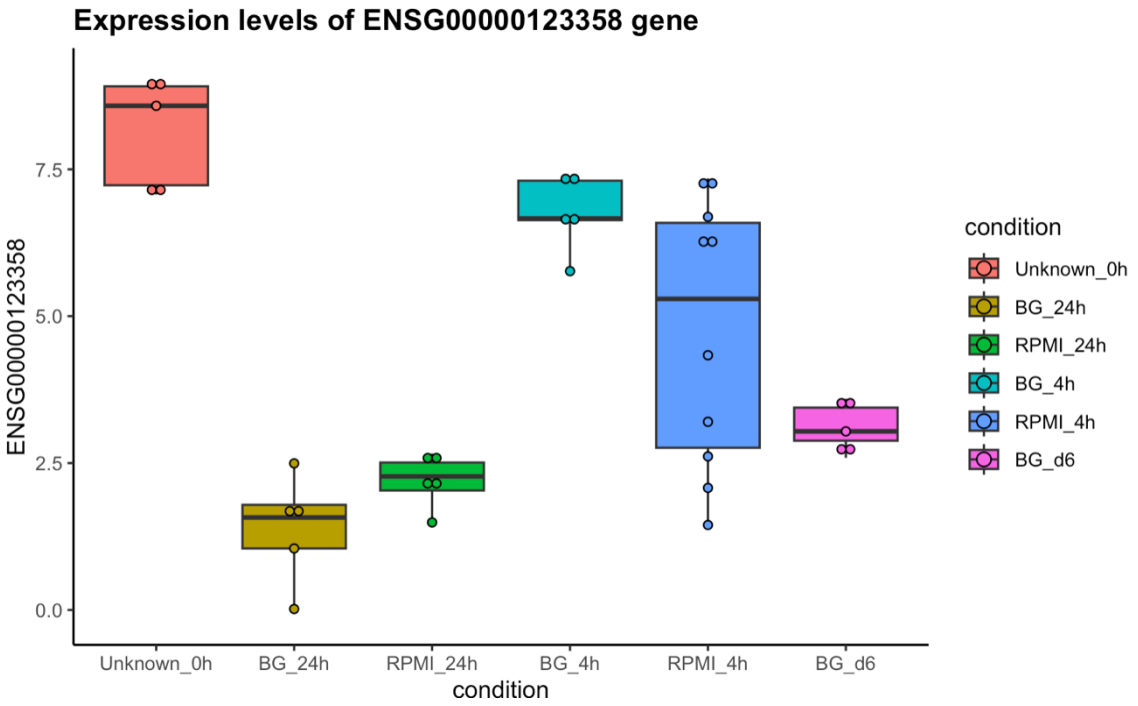

**28. Repetición del análisis para otro contraste (conditionUnknown_0h vs conditionBG_24h)**

Se repiten los mismos pasos previos, pero esta vez utilizando otro contraste (Unknown_0hvs_RPMI_24h). Se extraen los genes diferencialmente expresados, se filtran los significativos, y se generan nuevamente el histograma y el gráfico de volcán.




## 29. Generación de diagramas de caja (Boxplots) de expresión génica

Se seleccionan los genes más significativamente expresados en el contraste conditionUnknown_0h - conditionBG_24h. Luego, se generan diagramas de caja (boxplots) para visualizar sus niveles de expresión en función de las condiciones experimentales. Para mejorar la interpretación, se añaden puntos superpuestos con un gráfico de puntos (dotplot).




### 30. Preparación para el análisis de enriquecimiento funcional

Se mapean los identificadores de genes (ENSEMBL) a sus símbolos correspondientes (SYMBOL) utilizando la base de datos de anotaciones de org.Hs.eg.db. Esto permitirá asociar los genes analizados con bases de datos de rutas metabólicas y procesos biológicos.

### 31. Descarga de bases de datos KEGG y MSigDB

Se obtienen los enlaces de los genes de KEGG y sus descripciones, filtrando aquellos conjuntos de genes que contienen entre 5 y 500 genes. Posteriormente, se extraen también los conjuntos de genes de MSigDB, enfocándose en la colección "H" (Hallmark genes), que agrupa firmas genéticas representativas de procesos biológicos bien definidos.

### 32. Selección de genes significativamente expresados para enriquecimiento funcional

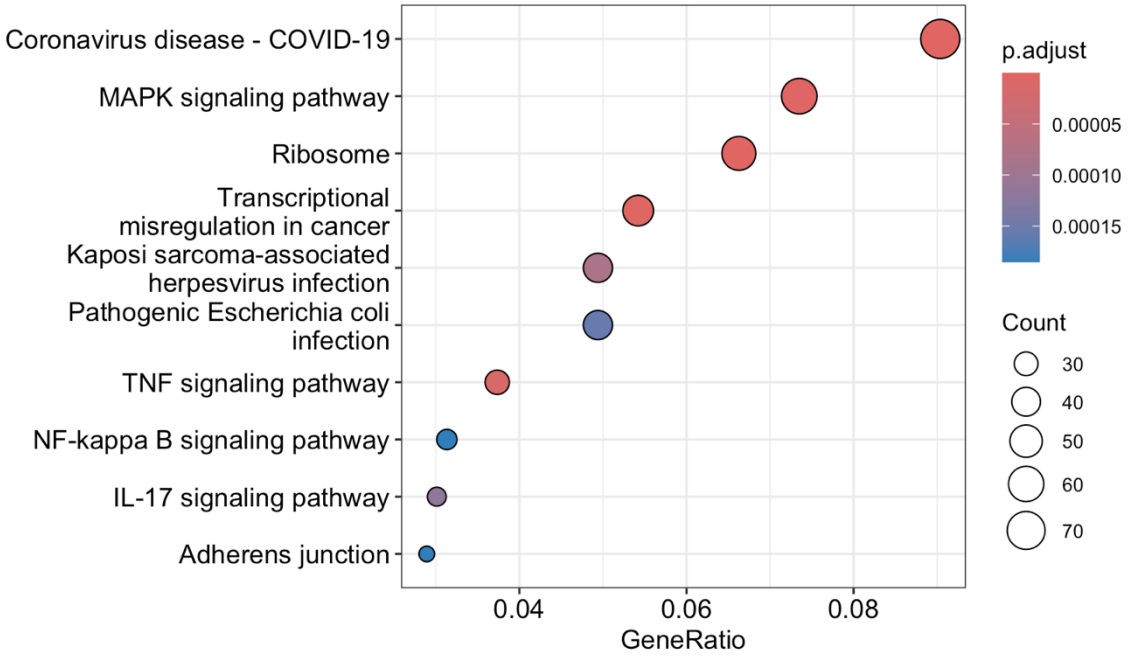
Se extraen los genes significativamente sobreexpresados (logFC >= 1) tanto en el contraste 0h vs 4h como en 0h vs 24h, para luego almacenarlos en listas separadas.

### 33. Conversión de identificadores de genes para el análisis de enriquecimiento

Se convierten los identificadores de genes de tipo ENSEMBL a ENTREZID, lo cual es necesario para realizar análisis de enriquecimiento con bases de datos como KEGG y MSigDB. Se eliminan los valores nulos para garantizar la correcta identificación de los genes.

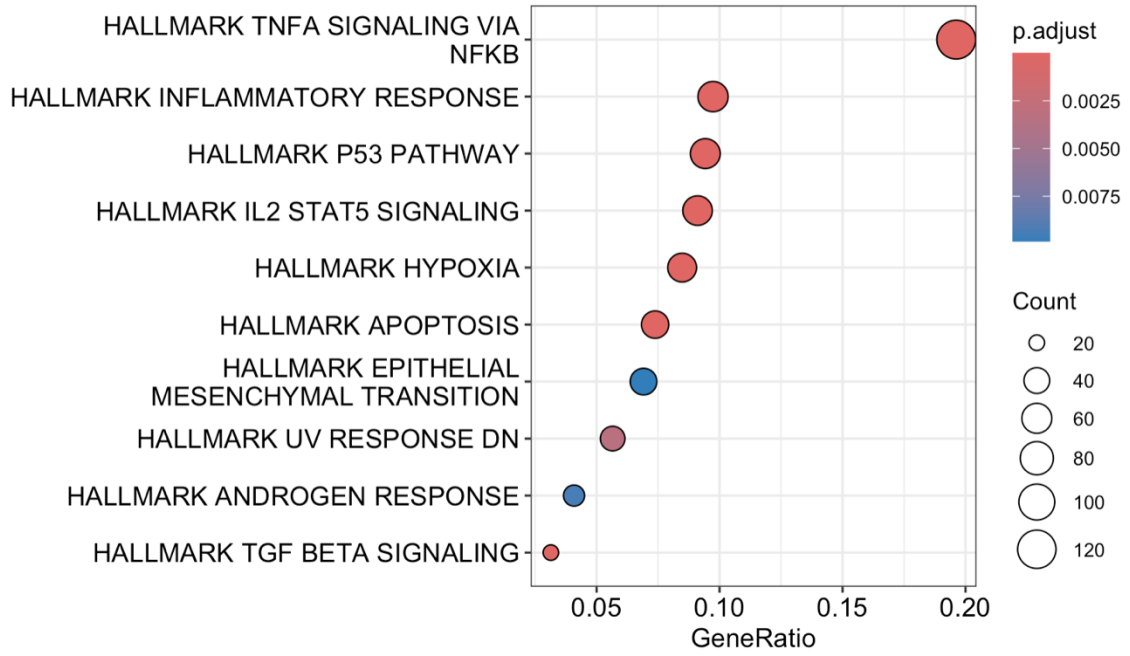
### 34. Análisis de enriquecimiento funcional con KEGG

Se realiza un análisis de enriquecimiento funcional (Over Representation Analysis - ORA) con la base de datos KEGG. Este análisis identifica qué rutas metabólicas están sobre-representadas en la lista de genes diferencialmente expresados. Los resultados se visualizan mediante gráficos de puntos (dotplots), resaltando las principales rutas enriquecidas.




**35. Análisis de enriquecimiento funcional con MSigDB**

Se lleva a cabo un análisis de enriquecimiento funcional con los conjuntos de genes de la base de datos MSigDB. Se filtra la colección "H" (Hallmark genes) y se realiza ORA para determinar qué procesos biológicos están enriquecidos en la lista de genes diferencialmente expresados. Al igual que en KEGG, los resultados se presentan en gráficos de puntos (dotplots)




---

# CONCLUSIONES

De acuerdo con el diseño experimental del estudio, la estimulación con **hemo** parece ser la condición biológica que induce más cambios en comparación con otros estímulos como el β-glucano o las porfirinas.

La estimulación con hemo y otros agentes induce cambios en **tres niveles principales**:

1. **Cambios Transcripcionales:**
    - Se activan genes proinflamatorios y relacionados con la inmunidad innata (RNA-seq).
    - Se observan modificaciones en la expresión de factores de transcripción clave.
2. **Cambios Epigenéticos y Cromatínicos:**

    - Se detecta mayor accesibilidad en regiones promotoras/enhancers mediante snATAC-seq.
    - Aumento en marcas epigenéticas activadoras como H3K27ac (ChIP-seq).
    - Cambios en la conectividad promotor-enhancer que afectan la transcripción a largo plazo.
3. **Cambios Funcionales en la Respuesta Inmune:**
    - Mayor producción de citoquinas inflamatorias (IL-6, TNF-α) tras reestimulación con LPS.
    - Alteración en la diferenciación de monocitos/macrófagos.
    - Modulación de la respuesta inflamatoria en modelos de endotoxemia y sepsis.

En conjunto, la estimulación induce una **reprogramación inmunológica duradera** que modifica la respuesta de las células a estímulos secundarios, reflejando el fenómeno de **entrenamiento inmune**.

Como podemos observar con el análisis de enriquecimiento, la ruta metabólica más sobrerrepresentada según la base de datos KEGG es la ruta metabólica del coronavirus-COVID-19, lo cual tiene bastante sentido, ya que estamos trabajando con células del sistema inmune.

La segunda ruta metabólica más representada en la base de datos KEGG es la vía metabólica de la MAPK, la cual interfiere en la actividad de ciertas proteínas, debido a su papel regulador en la transcripción de las células.

También tenemos vías sobreexpresadas de infecciones por *Escherichia coli* o algún herpes, y podemos observar, además, que existe una sobreexpresión de la regulación errónea de la transcripción en células cancerosas.

En cuanto a la base de datos MSIGDB, tenemos que el proceso biológico más representado es 'HALLMARK TNF SIGNALING VIA NFKB'. Esto es:

- **TNFA (Tumor Necrosis Factor Alpha)** es una citocina proinflamatoria clave en la respuesta inmune e inflamatoria.
- **NFKB (Nuclear Factor Kappa B)** es un factor de transcripción que regula la expresión de genes involucrados en inflamación, inmunidad, proliferación celular y apoptosis.
- **Esta vía de señalización** se activa cuando TNFA se une a su receptor (TNFR), desencadenando una cascada de eventos que culmina en la activación de NFKB.

- **Los genes de esta firma** están asociados con la activación de la respuesta inflamatoria y procesos como la respuesta inmune innata y adaptativa.

**Importancia en Biología y Medicina:**

- Está relacionada con enfermedades inflamatorias, autoinmunes y cáncer.
- Su activación excesiva puede contribuir a enfermedades como artritis reumatoide, enfermedad inflamatoria intestinal y ciertos tipos de cáncer.

Tiene sentido porque estamos trabajando con células del sistema inmune.

| **Paquete (Versión)** | **Origen** | **Enlace** |
|---|---|---|
| R (4.4.2) | | |
| dplyr (1.1.4) | CRAN | https://cran.r-project.org/package=dplyr |
| Stringr(1.5.1) | CRAN | https://cran.r-project.org/package=stringr |
| ggplot2 (3.5.1) | CRAN | https://cran.r-project.org/package=ggplot2 |
| ComplexHeatmap(2.22.0) | Bioconductor | https://bioconductor.org/packages/ComplexHeatmap |
| ggpubr(0.6.0) | CRAN | https://cran.r-project.org/package=ggpubr |
| RColorBrewer(1.1.3) | CRAN | https://cran.r-project.org/package=RColorBrewer |
| edgeR(4.4.1) | Bioconductor | https://bioconductor.org/packages/edgeR |
| biomaRt(2.62.0) | Bioconductor | https://bioconductor.org/packages/biomaRt |
| AnnotationDbi(1.68.0) | Bioconductor | https://bioconductor.org/packages/AnnotationDbi |
| limma(3.62.1) | Bioconductor | https://bioconductor.org/packages/limma |
| org.Hs.eg.db(3.20.0) | Bioconductor | https://bioconductor.org/packages/org.Hs.eg.db |
| msigdb(1.14.0) | Bioconductor | https://bioconductor.org/packages/msigdb |
| clusterProfiler(4.14.4) | Bioconductor | https://bioconductor.org/packages/clusterProfiler |
| fgsea(1.32.0) | Bioconductor | https://bioconductor.org/packages/fgsea |
| enrichplot(1.26.5) | Bioconductor | https://bioconductor.org/packages/enrichplot |
| msigdbr(7.5.1) | Bioconductor | https://bioconductor.org/packages/msigdbr |