Análisis estadístico de microarrays

Carla Riera Segura

3 de Mayo de 2020

Índice

Resumen

En este documento se presenta el análisis de datos de microarrays, de un estudio público, utilizando paquetes R y Bioconductor. El flujo de trabajo comienza con los datos sin procesar y pasa por una serie de pasos como son la identificación y lectura de los datos sin procesar, control de calidad, normalización, filtraje, selección de genes diferencialmente expresados, anotación génica, comparación de listas seleccionadas y análisis de significancia biológica. Los datos y el código para el análisis se proporcionan en un repositorio de github[1].

Objetivos

De cada paciente estudiado (4 en total), primero se clasificaron los macrófagos peritoneales (PM) en dos tipos según su tamaño (SPM y LPM) para ver cómo reaccionaban ante la infección bacteriana del E.Coli (LPS).

Así pues, el objetivo principal fue comparar qué genes activan los SPMs y los LPMs (por separado) ante la respuesta bacteriana, además de caracterizar la heterogeneidad de los macrófagos peritoneales en la cirrosis hepática descompensada.

[1] https://github.com/carlariera/AnalisisMicroarrays

Materiales

Los datos utilizados se cargaron en el Omnibus de expresión génica (GEO). El conjunto de datos seleccionado corresponde a un estudio realizado por Stengel S y T (2020) y se identifica con el número de acceso **GSE124878**.

El nivel de superfície de CD206 se utilizó para identificar PMs inflamatorios, maduros y residentes en pacientes con cirrosis. Los subconjuntos de PM (SPM y LPM) se clasificaron por citometría de flujo y las células clasificadas se sembraron en placas de cultivo celcular y se trataron con 10 ng/ml de lipopolisacárido de E.Coli (LPS) durante 3h o se dejaron sin tratar.

El experimento comparó, en 4 pacientes, los dos subconjuntos de PMs, macrófagos peritoneales pequeños (SPMs) y macrófagos peritoniales grandes (LPMs) con dos tratamientos distintos, sin tratamiento (Control) o después del tratamiento durante 3h con lipopolisacárido de E.Coli (LPS). Es decir, era un diseño factorial 2x2 (tipo de PM y tipo de tratamiento) con dos niveles cada uno (pequeño y grande para tipo de PM; control y LPS para tipo de tratamiento).

El tamaño de la muestra del experimento es de 16 muestras, cuatro réplicas de cada grupo.

Los microarrays utilizados para este experimento fueron del tipo *Clariom S Human HT* de Affymetrix.

Métodos

Antes de comenzar, debemos organizar en distintas carpetas el espacio desde donde trabajermos, con el fin de evitar perderse ya que se administrarán una gran cantidad de archivos.

Así pues, creamos la carpeta **AnalisisMicroarrays** que será el directorio de trabajo y dentro de ella creamos dos carpetas: una llamada **data** donde guardaremos todos los archivos *.CEL* y el archivo *targets* con información sobre las covariables; y otra llamada **results** donde enviaremos todos los resultados obtenidos en el análisis de microarrays.

```
# Definición del directorio
setwd(".")
```

Identificación y lectura de los datos

Como se ha detallado anteriormente, tenemos 16 muestras, divididas en 4 grupos distintos: LPMs Control (LPM.Control), SPMs Control (SPM.Control), LPMs con tratamiento LPS (LPM.LPS) y SPM con tratamiento LPS (SPM.LPS); con 4 réplicas de cada uno.

Para realizar cualquier análisis, debemos contar con los archivos *.CEL* (los cuales hemos descargado del GEO) y del archivo *targets*, el cual debe relacionar el nombre de cada archivo *.CEL* con su condición en el experimento.

Preparación del archivo *targets*

Nuestro archivo *targets* contiene 5 columnas. *FileName* contiene el nombre exacto de los archivos *.CEL*; *Group* contiene de manera resumida las condiciones del experimento para esa muestra; *PM_Subset* indica el tipo conjunto de macrófagos peritoniales; *Treatment* el tipo de tratamiento; y *ShortName* contiene una etiqueta corta con la información más relevante de la muestra.

```
# Archivo targets
targets <- read.csv2("./data/targets.csv", header = TRUE, sep = ";")
knitr::kable(targets, booktabs = TRUE, caption = 'Contenido del archivo targets utilizado para el análi
```

Cuadro 1: Contenido del archivo targets utilizado para el análisis

FileName	Group	PM_Subset	Treatment	ShortName
GSM3557819_A3849_01.CEL	SPM.Control	SPM	Control	SPM.Control.1
GSM3557820_A3849_02.CEL	SPM.LPS	SPM	LPS	SPM.LPS.1
GSM3557821_A3849_03.CEL	LPM.Control	LPM	Control	LPM.Control.1
GSM3557822_A3849_04.CEL	LPM.LPS	LPM	LPS	LPM.LPS.1
GSM3557823_A3849_05.CEL	SPM.Control	SPM	Control	SPM.Control.2
GSM3557824_A3849_06.CEL	SPM.LPS	SPM	LPS	SPM.LPS.2
GSM3557825_A3849_07.CEL	LPM.Control	LPM	Control	LPM.Control.2
GSM3557826_A3849_08.CEL	LPM.LPS	LPM	LPS	LPM.LPS.2
GSM3557827_A3849_09.CEL	SPM.Control	SPM	Control	SPM.Control.3
GSM3557828_A3849_10.CEL	SPM.LPS	SPM	LPS	SPM.LPS.3
GSM3557829_A3849_11.CEL	LPM.Control	LPM	Control	LPM.Control.3
GSM3557830_A3849_12.CEL	LPM.LPS	LPM	LPS	LPM.LPS.3
GSM3557831_A3849_13.CEL	SPM.Control	SPM	Control	SPM.Control.4
GSM3557832_A3849_14.CEL	SPM.LPS	SPM	LPS	SPM.LPS.4
GSM3557833_A3849_15.CEL	LPM.Control	LPM	Control	LPM.Control.4
GSM3557834_A3849_16.CEL	LPM.LPS	LPM	LPS	LPM.LPS.4

Lectura de los archivos .CEL

Para leer los archivos *.CEL* de la carpeta usamos la función *list.celfiles()*. Posteriormente, con la función *read.AnnotatedDataFrame()* asociamos la información almacenada en los archivos *.CEL* con el archivo *targets* en una sola variable llamada *rawData* (datos crudos, sin procesar). El archivo creado es conocido como *ExpressionSet* y en él se almacena toda la información disponible sobre el experimento.

```
# Creación del objeto ExpressionSet
library(oligo)
celFiles <- list.celfiles("./data", full.names = TRUE)
library(Biobase)
my.targets <- read.AnnotatedDataFrame(file.path("./data","targets.csv"), header = TRUE, row.names = 1, s
rawData <- read.celfiles(celFiles, phenoData = my.targets)
```

Una vez creado el *ExpressionSet*, procedemos a cambiar el nombre largo de las muestras en etiquetas más cortas (columna *ShortName* del archivo *targets*) con el fin de facilitar el procedimiento y la interpretación.

```
# Cambio del nombre de las filas del ExpressionSet
my.targets@data$ShortName->rownames(pData(rawData))
colnames(rawData) <-rownames(pData(rawData))

head(rawData)
```

```
## ExpressionFeatureSet (storageMode: lockedEnvironment)
## assayData: 1 features, 16 samples
##   element names: exprs
## protocolData
##   rowNames: SPM.Control.1 SPM.LPS.1 ... LPM.LPS.4 (16 total)
##   varLabels: exprs dates
##   varMetadata: labelDescription channel
## phenoData
##   rowNames: SPM.Control.1 SPM.LPS.1 ... LPM.LPS.4 (16 total)
##   varLabels: Group PM_Subset Treatment ShortName
##   varMetadata: labelDescription channel
## featureData: none
```

```
## experimentData: use 'experimentData(object)'
## Annotation: pd.clariom.s.human.ht
```

Control de calidad de los datos crudos

Para conocer la calidad de los datos sin procesar, procedemos a realizar el análisis de calidad de los datos crudos utilizando la función *arrayQualityMetrics()*.

```
# Análisis de calidad de los datos crudos
library(arrayQualityMetrics)
arrayQualityMetrics(rawData)
```

Vamos a echarle un vistazo al archivo generado *index.html*:

	array	sampleNames	*1	*2	*3	Group	PM_Subset	Treatment	ShortName
☐	1	GSM3557819_A3849_01.CEL				SPM.Control	SPM	Control	SPM.Control.1
☑	2	GSM3557820_A3849_02.CEL	x		x	SPM.LPS	SPM	LPS	SPM.LPS.1
☐	3	GSM3557821_A3849_03.CEL				LPM.Control	LPM	Control	LPM.Control.1
☐	4	GSM3557822_A3849_04.CEL				LPM.LPS	LPM	LPS	LPM.LPS.1
☐	5	GSM3557823_A3849_05.CEL				SPM.Control	SPM	Control	SPM.Control.2
☐	6	GSM3557824_A3849_06.CEL				SPM.LPS	SPM	LPS	SPM.LPS.2
☑	7	GSM3557825_A3849_07.CEL			x	LPM.Control	LPM	Control	LPM.Control.2
☐	8	GSM3557826_A3849_08.CEL				LPM.LPS	LPM	LPS	LPM.LPS.2
☐	9	GSM3557827_A3849_09.CEL				SPM.Control	SPM	Control	SPM.Control.3
☐	10	GSM3557828_A3849_10.CEL				SPM.LPS	SPM	LPS	SPM.LPS.3
☑	11	GSM3557829_A3849_11.CEL			x	LPM.Control	LPM	Control	LPM.Control.3
☐	12	GSM3557830_A3849_12.CEL				LPM.LPS	LPM	LPS	LPM.LPS.3
☐	13	GSM3557831_A3849_13.CEL				SPM.Control	SPM	Control	SPM.Control.4
☐	14	GSM3557832_A3849_14.CEL				SPM.LPS	SPM	LPS	SPM.LPS.4
☐	15	GSM3557833_A3849_15.CEL				LPM.Control	LPM	Control	LPM.Control.4
☐	16	GSM3557834_A3849_16.CEL				LPM.LPS	LPM	LPS	LPM.LPS.4

Figura 1: Aspecto de la tabla resumen del fichero index.html, producida por el paquete arrayQualityMetrics sobre los datos crudos

Podemos observar que hay tres muestras (2, 7 y 11) que han sido marcadas con una cruz como valores atípicos. De hecho, la muestra 2, en particular, presenta 2 cruces. Sin embargo, ya que ninguna de ellas ha sido marcada 3 veces (punto en el que debe revisarse la exclusión de una muestra para mejorar la calidad), podemos afirmar que los datos crudos de este experimento tienen suficiente calidad para la normalización.

A continuación, vamos a obtener un análisis más completo de los componentes principales para los datos sin procesar.

```
library(ggplot2)
library(ggrepel)
plotPCA3 <- function (datos, labels, factor, title, scale, colores, size = 1.5, glineas = 0.25) {
  data <- prcomp(t(datos),scale=scale)
  # Ajustes del plot
  dataDf <- data.frame(data$x)
  Group <- factor
  loads <- round(data$sdev^2/sum(data$sdev^2)*100,1)
  # Plot principal
  p1 <- ggplot(dataDf,aes(x=PC1, y=PC2)) +
```

```
  theme_classic() +
  geom_hline(yintercept = 0, color = "gray70") +
  geom_vline(xintercept = 0, color = "gray70") +
  geom_point(aes(color = Group), alpha = 0.55, size = 3) +
  coord_cartesian(xlim = c(min(data$x[,1])-5,max(data$x[,1])+5)) +
  scale_fill_discrete(name = "Grupo")
# Evitación de la superposición de etiquetas
p1 + geom_text_repel(aes(y = PC2 + 0.25, label = labels),segment.size = 0.25, size = size) +
  labs(x = c(paste("PC1",loads[1],"%")),y=c(paste("PC2",loads[2],"%"))) +
  ggtitle(paste("Análisis de componentes principales para",title,sep=" "))+
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_manual(values=colores)
}
```

Mostramos el diagrama de dispersión de los dos componentes principales realizado sobre los datos crudos.

```
plotPCA3(exprs(rawData), labels = targets$ShortName, factor = targets$Group,
         title="datos crudos", scale = FALSE, size = 3,
         colores = c("brown", "green", "orange", "purple"))
```

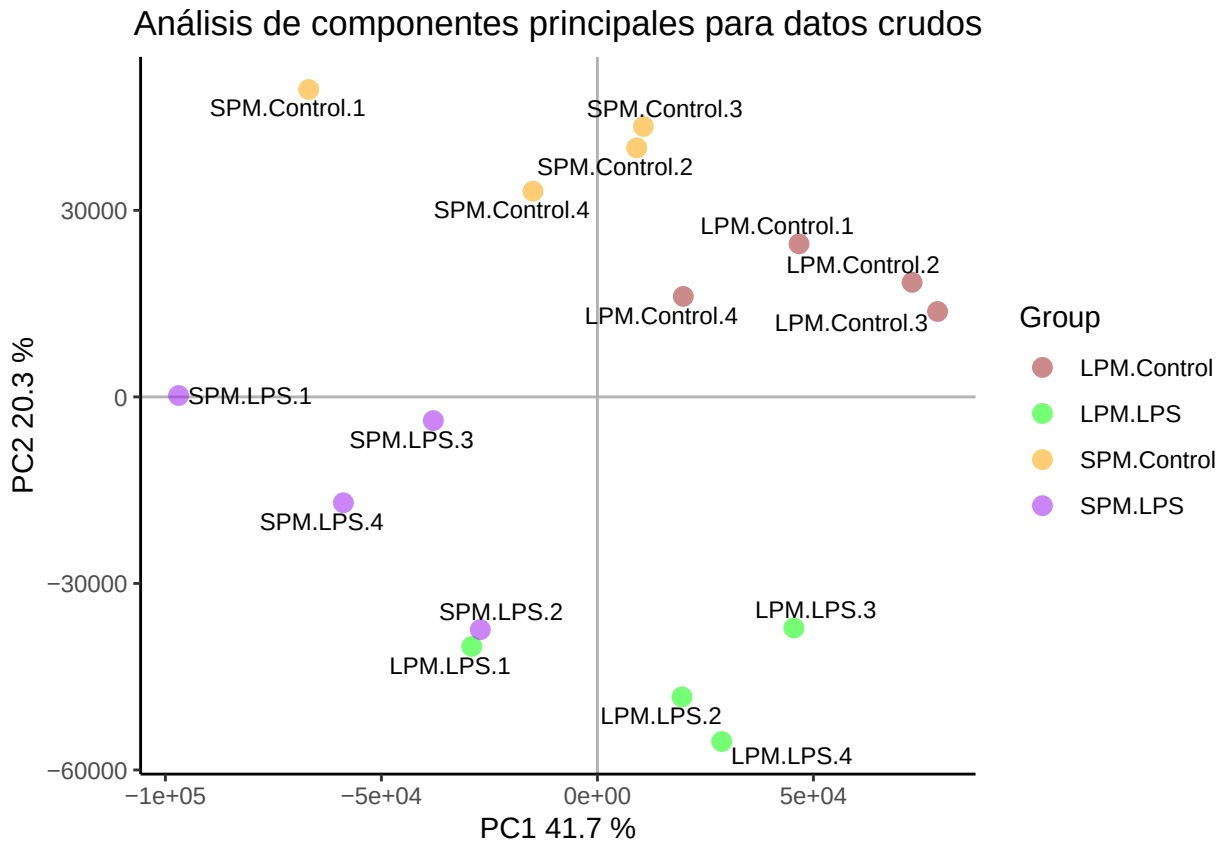

Figura 2: Visualización de los dos componentes principales para datos crudos

El primer componente de la PCA (*Principal Component Analysis*) representa el 41.7 % de la variabilidad total de las muestras, la cual se debe principalmente a la condición del tipo de PMs ya que las muestras de macrófagos peritoneales grandes (LPMs) se encuentran a la derecha y las muestras de macrófagos pequeños (SPMs) están a la izquierda. Cabe destacar que las muestras 2 y 3 del grupo SPM.Control se encuentran fuera del que sería su cuadrante, y lo mismo sucede con la muestra 1 del grupo LPM.LPS.

El segundo componente de la PCA representa el 20.3 % de la variabilidad total de las muestras, la cual se debe a la condición del tipo de tratamiento ya que las muestras que no han sido sometidas a tratamiento (Control) se encuentran en la parte superior y las muestras sometidas a tratamiento (LPS) están en la parte inferior.

Vamos ahora a visualizar la distribución de intensidad de las matrices usando bloxplots o diagramas de caja.

```
boxplot(rawData, cex.axis=0.5, las=2,  which="all",
         col = c(rep("brown", 4), rep("green", 4), rep("orange", 4), rep("purple", 4)),
         main="Distribución de los valores de intensidad de los datos crudos")
```

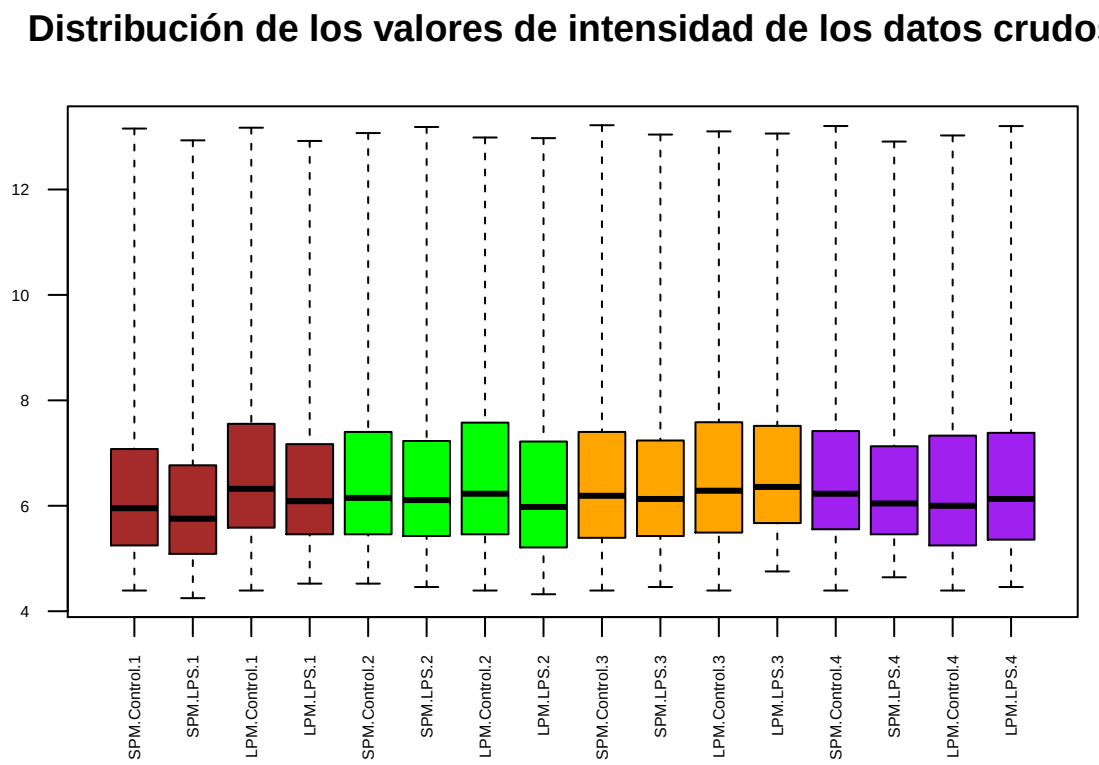

Figura 3: Distribución de las intensidades de los datos crudos

Podemos observar una ligera variación de intensidad entre los arrays, aunque era esperable ya que estamos tratando con los datos brutos.

Normalización de los datos

Una vez hemos realizado el control de calidad vamos a proceder a normalizar los datos y sumarizarlos para conseguir que las matrices sean comparables entre ellas y tratar de reducir, y si es posible eliminar, toda la variabilidad en las muestras que no se deba a razones biológicas.

Para ello utilizaremos el método RMA, el cual implica 3 etapas: corrección de fondo, normalización para hacer que los valores de los arrays sean comparables y resumen de las diversas sondas asociadas a cada grupo de sondas para dar un único valor.

```
# Normalización de los datos
eset_rma <- rma(rawData)
```

```
## Background correcting
## Normalizing
## Calculating Expression
```

Control de calidad de los datos normalizados

Para conocer la calidad de los datos normalizados vamos a realizar el análisis de calidad utilizando la función *arrayQualityMetrics()* tal y como se ha hecho anteriormente.

```
# Análisis de calidad de los datos normalizados
arrayQualityMetrics(eset_rma, outdir = file.path("./results", "QCDir.Norm"), force = TRUE)
```

Vamos a echarle un vistazo al archivo generado *index.html*:

	array	sampleNames	*1	*2	*3	Group	PM_Subset	Treatment	ShortName
☑	1	SPM.Control.1	x			SPM.Control	SPM	Control	SPM.Control.1
☑	2	SPM.LPS.1	x			SPM.LPS	SPM	LPS	SPM.LPS.1
☐	3	LPM.Control.1				LPM.Control	LPM	Control	LPM.Control.1
☐	4	LPM.LPS.1				LPM.LPS	LPM	LPS	LPM.LPS.1
☐	5	SPM.Control.2				SPM.Control	SPM	Control	SPM.Control.2
☐	6	SPM.LPS.2				SPM.LPS	SPM	LPS	SPM.LPS.2
☐	7	LPM.Control.2				LPM.Control	LPM	Control	LPM.Control.2
☐	8	LPM.LPS.2				LPM.LPS	LPM	LPS	LPM.LPS.2
☐	9	SPM.Control.3				SPM.Control	SPM	Control	SPM.Control.3
☐	10	SPM.LPS.3				SPM.LPS	SPM	LPS	SPM.LPS.3
☐	11	LPM.Control.3				LPM.Control	LPM	Control	LPM.Control.3
☐	12	LPM.LPS.3				LPM.LPS	LPM	LPS	LPM.LPS.3
☐	13	SPM.Control.4				SPM.Control	SPM	Control	SPM.Control.4
☐	14	SPM.LPS.4				SPM.LPS	SPM	LPS	SPM.LPS.4
☐	15	LPM.Control.4				LPM.Control	LPM	Control	LPM.Control.4
☐	16	LPM.LPS.4				LPM.LPS	LPM	LPS	LPM.LPS.4

Figura 4: Aspecto de la tabla resumen del fichero index.html, producida por el paquete arrayQualityMetrics sobre los datos normalizados

Podemos observar que ahora hay 2 muestras (1 y 2) que han sido marcadas con una cruz como valores atípicos, y por tanto, concluimos que los datos normalizados tienen suficiente calidad.

Tal y como hemos realizado en apartados previos, vamos a obtener un análisis más completo de los componentes principales para los datos normalizados.

```
plotPCA3(exprs(eset_rma), labels = targets$ShortName, factor = targets$Group,
         title="datos normalizados", scale = FALSE, size = 3,
         colores = c("brown", "green", "orange", "purple"))
```

Ahora el primer componente de la PCA representa el 33 % de la variabilidad total, la cual cosa supone una disminución con respecto a la PCA realizada para datos crudos. De la misma manera que con el PCA para datos sin procesar, las muestras son separadas a derecha e izquierda por el tipo de PMs, y arriba y abajo por el tipo de tratamiento. Podemos observar que las muestras 2 y 3 del grupo SPM.Control han quedado más agrupadas con las otras de su grupo con la normalización de los datos, y lo mismo ha sucedido con la muestra 1 del grupo LPM.LPS.

Vamos ahora a visualizar la distribución de intensidad de las matrices usando bloxplots o diagramas de caja.

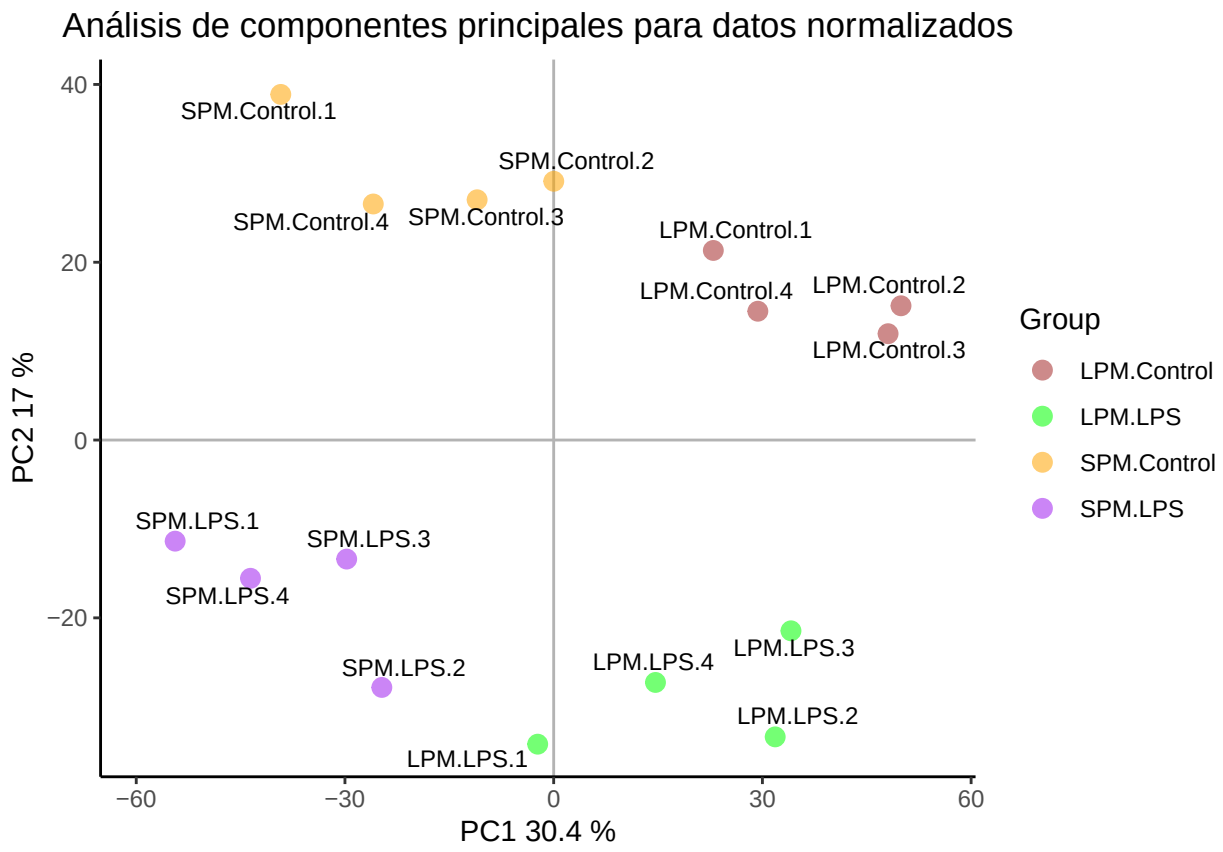

Figura 5: Visualización de los dos componentes principales para datos normalizados

```
boxplot(eset_rma, cex.axis=0.5, las=2,  which="all",
         col = c(rep("brown", 4), rep("green", 4), rep("orange", 4), rep("purple", 4)),
         main="Distribución de los valores de intensidad de los datos normalizados")
```

```
## Warning in .local(x, ...): Argument 'which' ignored (not meaningful for
## ExpressionSet)
```

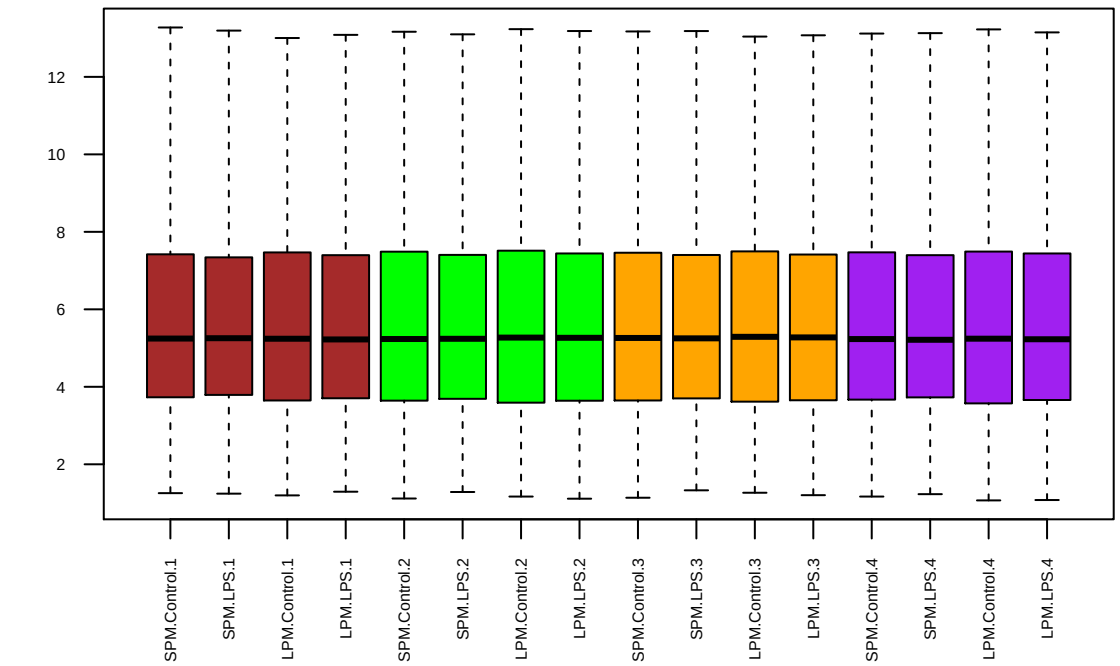

Figura 6: Distribución de las intensidades de los datos normalizados

Como era de esperar, la distribución de todos los arrays tiene un aspecto muy similar, la cual cosa sugiere que la normalización ha funcionado bien.

Filtraje no específico

Realizada la normalización, puede realizarse un filtraje no específico con el fin de eliminar genes que constituyen básicamente ruido. Bien porque sus señales son muy bajas o bien porque apenas varían entre condiciones, por lo que no aportan nada a la selección de genes diferencialmente expresados.

Con el fin de detectar y eliminar los errores introducidos por variaciones experimentales del tiempo y el lugar (efectos por lotes) vamos a realizar un análisis de componentes de variación principal (PVCA). Este análisis estima la fuente y la proporción de variación en dos pasos: análisis de componentes principales y análisis de componentes de varianza.

Primero vamos a observar si las muestras fueron procesadas en el mismo día utilizando el comando *get.celfile.dates()*.

```
library(affyio)
get.celfile.dates(celFiles)
```

```
##  [1] "2018-07-10" "2018-07-10" "2018-07-10" "2018-07-10" "2018-07-10"
##  [6] "2018-07-10" "2018-07-10" "2018-07-10" "2018-07-10" "2018-07-10"
```

```
## [11] "2018-07-10" "2018-07-10" "2018-07-10" "2018-07-10" "2018-07-10"
## [16] "2018-07-10"
```

Partiendo del conocimiento que todas las muestras se procesaron el mismo día (10 de julio de 2018), procedemos a realizar el análisis PVCA.

```
library(pvca)
pData(eset_rma) <- targets
pct_threshold <- 0.6
batch.factors <- c("PM_Subset", "Treatment")
pvcaObj <- pvcaBatchAssess (eset_rma, batch.factors, pct_threshold)
```

Graficamos los resultados del análisis, donde cada barra corresponde a cada fuente de variación incluida en el análisis y su tamaño indica el porcentaje de variabilidad atribuible a cada fuente.

```
bp <- barplot(pvcaObj$dat, xlab = "Efectos",
  ylab = "Variación de la proporción promedio ponderada",
  ylim= c(0,0.6),col = c("mediumorchid"), las=2,
  main="Estimación PVCA")
axis(1, at = bp, labels = pvcaObj$label, cex.axis = 0.75, las=2)
values = pvcaObj$dat
new_values = round(values , 3)
text(bp,pvcaObj$dat,labels = new_values, pos=3, cex = 0.7)
```

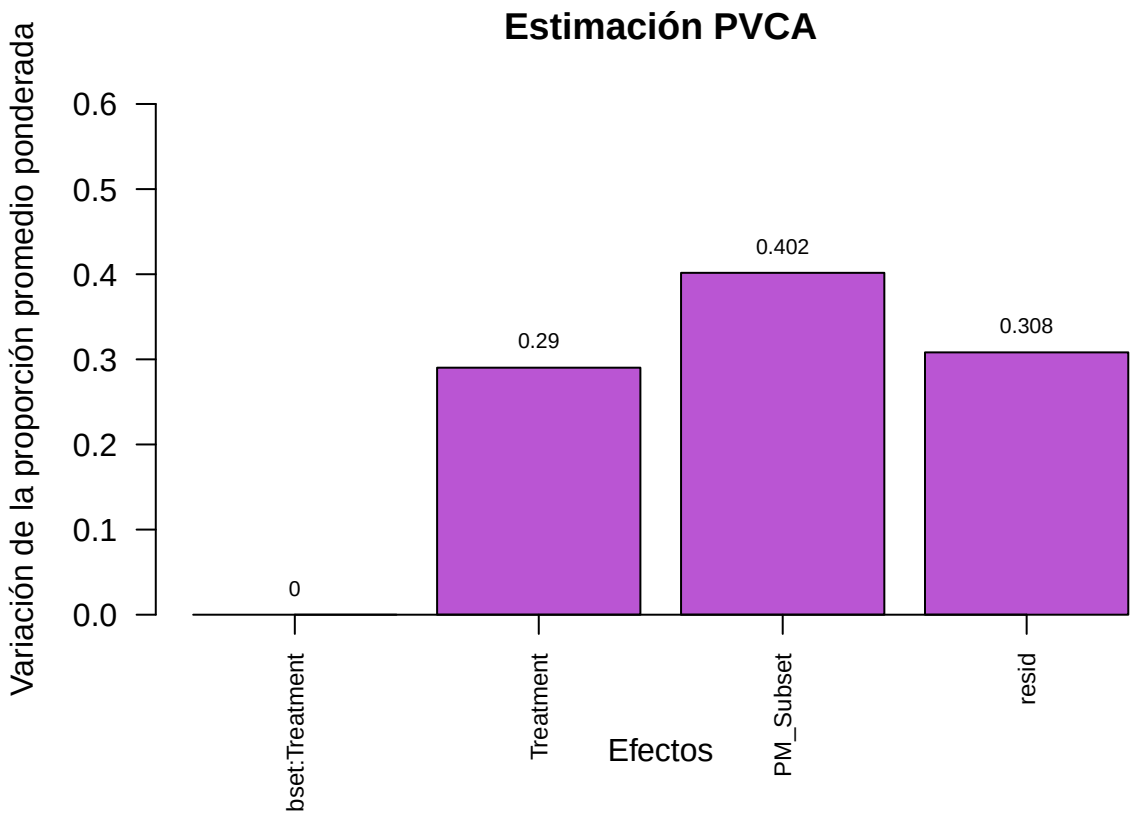

Figura 7: Importancia relativa de los diferentes factores - tipo de PMs, tratamiento e interacción - que afectan a la expresión génica

Podemos observar que la principal fuente de variación en las muestras es la condición del tipo de PM (SPM o LPM).

Detección de los genes más variables

Cuando un gen se expresa de manera diferencial, se espera que haya una cierta diferencia entre los grupos y, por lo tanto, la varianza general del gen será mayor que la de aquellos que no tienen expresión diferencial.

Así pues, vamos a trazar la variabilidad general de todos los genes, donde se representan las desviaciones estándar de todos ellos ordenados de menor a mayor valor.

```
sds <- apply (exprs(eset_rma), 1, sd)
sds0<- sort(sds)
plot(1:length(sds0), sds0, main="Distribución de variabilidad para todos los genes",
     sub="Las líneas verticales representan los percentiles 90% y 95%",
     xlab="Índice de genes (de menos a más variable)", ylab="Desviación estándar")
abline(v=length(sds)*c(0.9,0.95))
```

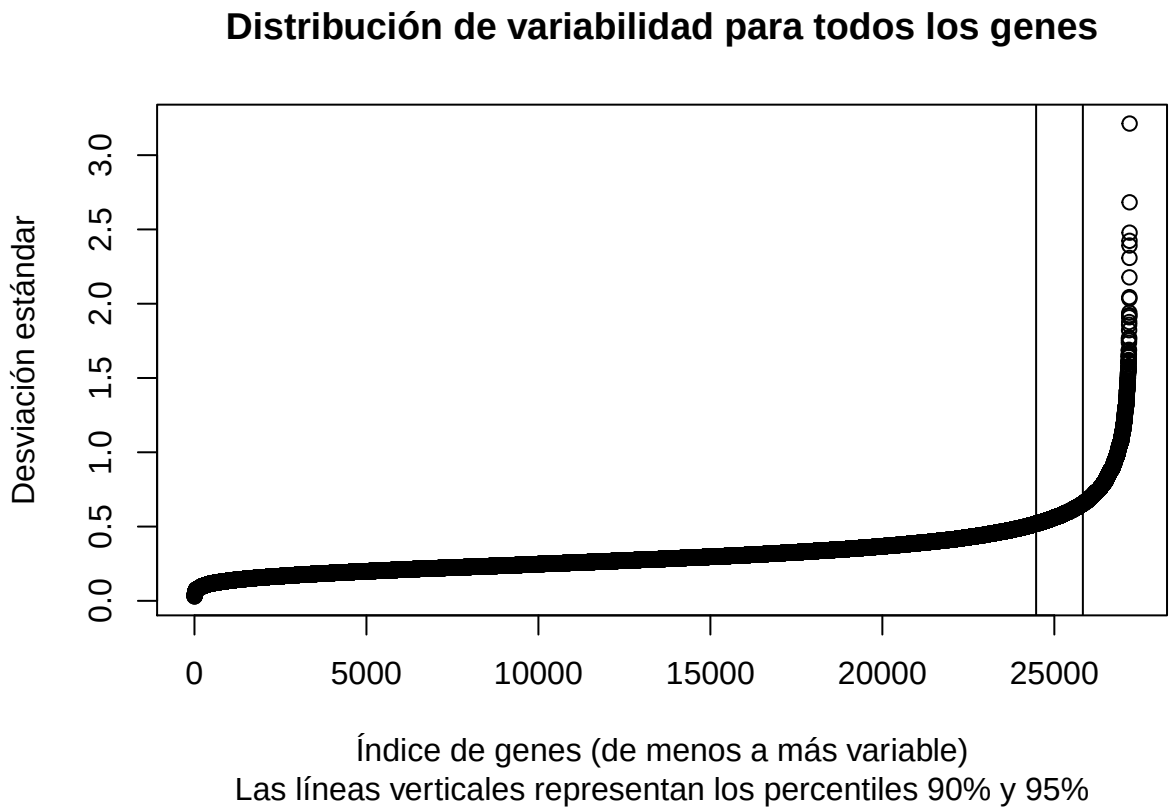

Figura 8: Valores de las desviaciones estándar de todas las muestras para todos los genes ordenados de menor a mayor

El gráfico muestra que los genes más variables son aquellos con una desviación estándar superior al 90-95 % de todas las desviaciones estándar.

Filtración de los genes menos variables

Vamos a filtrar aquellos genes que no se espera que se expresen diferencialmente. Para ello, podemos utilizar la función *nsFilter()* que permite eliminar los genes que, o bien varían poco, o bien no se dispone de anotación para ellos.

```
library(genefilter)
library(clariomdhumantranscriptcluster.db)
```

```
annotation(eset_rma) <- "clariomdhumantranscriptcluster.db"
filtered <- nsFilter(eset_rma, require.entrez = TRUE, remove.dupEntrez = TRUE,var.filter=TRUE, var.func=
```

La función *nsFilter()* devuelve los valores filtrados en un objeto *ExpressionSet* y un informe de los resultados del filtraje.

```
print(filtered$filter.log)
```

```
## $numDupsRemoved
## [1] 86
##
## $numLowVar
## [1] 13894
##
## $numRemoved.ENTREZID
## [1] 8577
```

```
eset_filtered <- filtered$eset
```

Al inicio del estudio contábamos con 27189 genes, valor que ha quedado reducido a 4632, después de eliminar todos los genes que no se expresan diferencialmente (27189 - 86 - 13894 - 8577 = 4632). Los genes que quedan después de filtrar han sido almacenados en la variable *eset_filtered*.

Comprobamos el valor de genes restantes después del filtraje con el siguiente código.

```
# Genes restantes después del filtraje
dim(exprs(eset_filtered))[1]
```

```
## [1] 4632
```

Almacenaje de datos normalizados y filtrados

Procedemos a guardar los datos filtrados normalizados para futuros análisis.

```
write.csv(exprs(eset_rma), file="./results/normalized.Data.csv")
write.csv(exprs(eset_filtered), file="./results/normalized.Filtered.Data.csv")
save(eset_rma, eset_filtered, file="./results/normalized.Data.Rda")
```

Selección de genes diferencialmente expresados

Para seleccionar los genes diferencialmente expresados de este estudio, vamos a utilizar la aproximación presentada por Smyth del paquete *limma*. Ésta se basa en la utilización del modelo lineal general, combinada con un método para obtener una estimación mejorada de la varianza.

Creación de la matriz de diseño

El primer paso para el análisis basado en modelos lineales es la creación de la matriz de diseño.

```
if (!exists("eset_filtered")) load (file="./results/normalized.Data.Rda")
```

La matriz de diseño estará compuesta por 16 filas (muestras del estudio) y por 4 columnas (grupos). La variable *Group* es una combinación de las dos condiciones experimentales: SPM/LPM y Control/LPS, que se representan conjuntamente como un factor de 4 niveles. + SPM.Control (presencia de LPMs y sin tratamiento) + SPM.LPS (presencia de LPMs y sin tratamiento) + LPM.Control (presencia de LPMs y sin tratamiento) + LPM.LPS (presencia de LPMs y con tratamiento LPS)

```
library(limma)
designMat<- model.matrix(~0+Group, pData(eset_filtered))
```

```
colnames(designMat) <- c("LPM.Control", "LPM.LPS", "SPM.Control", "SPM.LPS")
print(designMat)
```

```
##    LPM.Control LPM.LPS SPM.Control SPM.LPS
## 1            0       0           1       0
## 2            0       0           0       1
## 3            1       0           0       0
## 4            0       1           0       0
## 5            0       0           1       0
## 6            0       0           0       1
## 7            1       0           0       0
## 8            0       1           0       0
## 9            0       0           1       0
## 10           0       0           0       1
## 11           1       0           0       0
## 12           0       1           0       0
## 13           0       0           1       0
## 14           0       0           0       1
## 15           1       0           0       0
## 16           0       1           0       0
## attr(,"assign")
## [1] 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Group
## [1] "contr.treatment"
```

Definición de comparaciones con la matriz de contrastes

Ahora vamos a crear la matriz de contrastes para describir las comparaciones entre grupos.

En este estudio queremos verificar el efecto de la aplicación o no del tratamiento con E.Coli (Control vs LPS) por separado para los SPMs y los LPMs. Además, también queremos comprobar si existe diferencia entre el tipo de macrófagos peritoneales. Esto se puede hacer haciendo tres comparaciones que se describen a continuación:

```
cont.matrix <- makeContrasts (SPM.LPSvsControl = SPM.LPS-SPM.Control,
                              LPM.LPSvsControl = LPM.LPS-LPM.Control,
                              PM = SPM.Control-LPM.Control,
                              levels=designMat)
print(cont.matrix)
```

```
##              Contrasts
## Levels        SPM.LPSvsControl LPM.LPSvsControl PM
##   LPM.Control                0               -1 -1
##   LPM.LPS                    0                1  0
##   SPM.Control               -1                0  1
##   SPM.LPS                    1                0  0
```

En definitiva, la matriz de contraste se ha definido para realizar tres comparaciones: + Aplicación del tratamiento LPS en macrófagos peritoneales pequeños (SPMs) + Aplicación del tratamiento LPS en macrófagos peritoneales grandes (LPMs) + Diferencia entre el tipo de macrófago (SPM vs LPM)

Estimación del modelo y selección de genes

Una vez definida la matriz de diseño y los contrastes, podemos pasar a estimar el modelo, estimar los contrastes y realizar las pruebas de significación, para cada gen y cada comparación, si pueden considerarse

diferencialmente expresados.

Toda la información relevante para una mayor exploración de los resultados la almacenamos en un objeto R de la clase MArrayLM definida en el paquete *limma*, al que se nombra como *fit.main*.

```
library(limma)
fit<-lmFit(eset_filtered, designMat)
fit.main<-contrasts.fit(fit, cont.matrix)
fit.main<-eBayes(fit.main)
class(fit.main)
```

```
## [1] "MArrayLM"
## attr(,"package")
## [1] "limma"
```

Obtención de listas de genes expresados diferencialmente

Para obtener una lista de genes ordenadors de más a menos diferencialmente expresados para cada contraste, podemos hacer uso de la función *topTable()*.

Para la comparación 1 (SPM.LPSvsControl): genes que cambian su expresión entre LPS y Control en macrófagos peritoneales pequeños.

```
topTab_SPM.LPSvsControl <- topTable (fit.main, number=nrow(fit.main), coef="SPM.LPSvsControl", adjust="
head(topTab_SPM.LPSvsControl)
```

```
##                       logFC   AveExpr        t      P.Value    adj.P.Val
## TC1800007471.hg.1 2.300106  9.230860 11.89240 1.112957e-09 4.295274e-06
## TC0700006890.hg.1 4.621228  9.244300 11.45581 1.969076e-09 4.295274e-06
## TC0500008307.hg.1 3.279997  5.821030 11.19755 2.781913e-09 4.295274e-06
## TC0100016715.hg.1 2.900772  9.705994 10.86027 4.409970e-09 5.106745e-06
## TC0100006483.hg.1 2.947474  8.191151 10.50997 7.199910e-09 6.669996e-06
## TC1000008397.hg.1 3.008548 10.269101 10.13165 1.239630e-08 7.681833e-06
##                         B
## TC1800007471.hg.1 12.40918
## TC0700006890.hg.1 11.87600
## TC0500008307.hg.1 11.55130
## TC0100016715.hg.1 11.11645
## TC0100006483.hg.1 10.65143
## TC1000008397.hg.1 10.13337
```

Para la comparación 2 (LPM.LPSvsControl): genes que cambian su expresión entre LPS y Control en macrófagos peritoneales grandes.

```
topTab_LPM.LPSvsControl <- topTable (fit.main, number=nrow(fit.main), coef="LPM.LPSvsControl", adjust="
head(topTab_LPM.LPSvsControl)
```

```
##                       logFC   AveExpr        t      P.Value    adj.P.Val
## TC0500008307.hg.1 4.356883  5.821030 14.87392 3.377825e-11 1.564608e-07
## TC1800007471.hg.1 2.617737  9.230860 13.53467 1.498473e-10 3.470464e-07
## TC0100016715.hg.1 3.255227  9.705994 12.18732 7.641186e-10 1.179799e-06
## TC0100006483.hg.1 3.247831  8.191151 11.58097 1.669086e-09 1.794603e-06
## TC1100012134.hg.1 5.098410  6.412998 11.24159 2.621587e-09 1.794603e-06
## TC1000008397.hg.1 3.337669 10.269101 11.24000 2.627198e-09 1.794603e-06
##                         B
## TC0500008307.hg.1 15.68643
## TC1800007471.hg.1 14.32914
## TC0100016715.hg.1 12.81217
```

```
## TC0100006483.hg.1 12.07402
## TC1100012134.hg.1 11.64464
## TC1000008397.hg.1 11.64260
```

Para la comparación 3 (PM): genes que se cambian su expresión entre SPM y LPM.

```
topTab_PM  <- topTable (fit.main, number=nrow(fit.main), coef="PM", adjust="fdr")
head(topTab_PM)
```

```
##                         logFC  AveExpr         t      P.Value    adj.P.Val
## TC0200006677.hg.1 -3.265149 6.172736 -9.642633 2.558705e-08 4.527133e-05
## TC2000007089.hg.1 -2.594413 5.363168 -9.408936 3.651379e-08 4.527133e-05
## TC0300010775.hg.1  3.035489 4.720077  9.326627 4.144652e-08 4.527133e-05
## TC0100015856.hg.1  4.471536 7.179823  9.248513 4.677630e-08 4.527133e-05
## TC0100008101.hg.1 -2.027201 4.515460 -9.161487 5.356966e-08 4.527133e-05
## TC1700010618.hg.1 -2.097731 5.460852 -9.099171 5.906426e-08 4.527133e-05
##                           B
## TC0200006677.hg.1 9.358200
## TC2000007089.hg.1 9.026534
## TC0300010775.hg.1 8.908026
## TC0100015856.hg.1 8.794732
## TC0100008101.hg.1 8.667556
## TC1700010618.hg.1 8.575869
```

Anotación génica

La anotación génica consiste en adivinar qué gen corresponde a cada ID de Affymetrix (primera columna de las tablas).

Debido a que hay 3 tablas, vamos a ejecutar la siguiente función para hacerlo más simple.

```
annotatedTopTable <- function(topTab, anotPackage)
{
  topTab <- cbind(PROBEID=rownames(topTab), topTab)
  myProbes <- rownames(topTab)
  thePackage <- eval(parse(text = anotPackage))
  geneAnots <- select(thePackage, myProbes, c("SYMBOL", "ENTREZID", "GENENAME"))
  annotatedTopTab<- merge(x=geneAnots, y=topTab, by.x="PROBEID", by.y="PROBEID")
return(annotatedTopTab)
}
```

```
topAnnotated_SPM.LPSvsControl <- annotatedTopTable(topTab_SPM.LPSvsControl,
anotPackage="clariomdhumantranscriptcluster.db")
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
topAnnotated_LPM.LPSvsControl <- annotatedTopTable(topTab_LPM.LPSvsControl,
anotPackage="clariomdhumantranscriptcluster.db")
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
topAnnotated_PM <- annotatedTopTable(topTab_PM,
anotPackage="clariomdhumantranscriptcluster.db")
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
write.csv(topAnnotated_SPM.LPSvsControl, file="./results/topAnnotated_SPM.LPSvsControl.csv")
write.csv(topAnnotated_LPM.LPSvsControl, file="./results/topAnnotated_LPM.LPSvsControl.csv")
write.csv(topAnnotated_PM, file="./results/topAnnotated_PM.csv")
```

Cuadro 2: Anotaciones agregadas a los resultados "topTable" para la comparación "SPM.LPSvsControl"

PROBEID	SYMBOL	ENTREZID	GENENAME
TC0100006483.hg.1	ISG15	9636	ISG15 ubiquitin like modifier
TC0100006497.hg.1	SCNN1D	6339	sodium channel epithelial 1 delta subunit
TC0100006501.hg.1	CPTP	80772	ceramide-1-phosphate transfer protein
TC0100006516.hg.1	VWA1	64856	von Willebrand factor A domain containing 1
TC0100006565.hg.1	SKI	6497	SKI proto-oncogene

La siguiente tabla muestra las anotaciones agregadas a los resultados “topTable” para la comparación “SPM.LPSvsControl”.

```
##             PROBEID SYMBOL ENTREZID                                  GENENAME
## 1 TC0100006483.hg.1  ISG15     9636               ISG15 ubiquitin like modifier
## 2 TC0100006497.hg.1 SCNN1D     6339   sodium channel epithelial 1 delta subunit
## 3 TC0100006501.hg.1   CPTP    80772       ceramide-1-phosphate transfer protein
## 4 TC0100006516.hg.1   VWA1    64856 von Willebrand factor A domain containing 1
## 5 TC0100006565.hg.1    SKI     6497                           SKI proto-oncogene
```

Visualización de la expresión diferencial

Para obtener una visualización de la expresión diferencial global se puede emplear un gráfico volcán.

```
library(clariomdhumantranscriptcluster.db)
geneSymbols <- select(clariomdhumantranscriptcluster.db, rownames(fit.main), c("SYMBOL"))
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
SYMBOLS<- geneSymbols$SYMBOL
volcanoplot(fit.main, coef=1, highlight=4, names=SYMBOLS,
            main=paste("Genes expresados diferencialmente", colnames(cont.matrix)[1], sep="\n"))
  abline(v=c(-1,1))
```

Los genes cuyo logaritmo negativo es superior a 0 y cuyo Log2 Fold Change es, en valor absoluto, superior a 1, son candidatos a estar diferencialmente expresados.

Comparación entre distintas comparaciones

Dado que se han realizado varias comparaciones, nos puede resultar importante ver qué genes cambian simultáneamente en más de una comparación. Para ello podemos utilizar la función *decidetests*, la cual devuelve una tabla que llamaremos *res*.

```
library(limma)
res <- decideTests(fit.main, method="separate", adjust.method="fdr", p.value=0.1, lfc=1)
```

Para cada gen y cada comparación contiene un 1 (si el gen está sobreexpresado o *up* en esta condición), un 0 (si no hay cambio significativo) o un -1 (si está *down* regulado).

```
sum.res.rows<-apply(abs(res),1,sum)
res.selected<-res[sum.res.rows!=0,]
print(summary(res))
```

```
##        SPM.LPSvsControl LPM.LPSvsControl   PM
## Down                 47              26  396
## NotSig             4340            4356 4102
## Up                  245             250  134
```

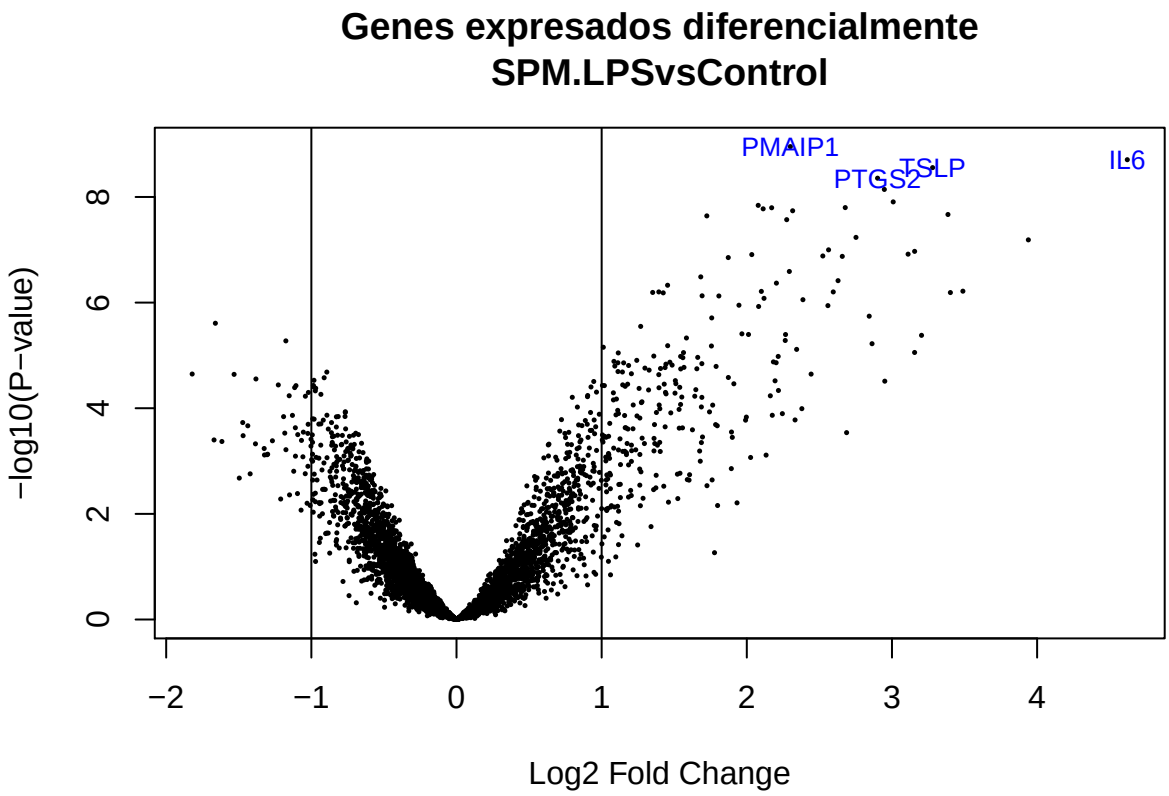

Figura 9: Gráfico volcán para la comparación entre LPS y Control en mácrofagos pequeños. Los nombres de los 4 genes principales se muestran en el gráfico

Estos resultados se pueden visualizar en un diagrama de Venn, el cual muestra cuántos de estos genes hay en común entre las comparaciones dos a dos o con las tres.

```
vennDiagram (res.selected[,1:3], cex=0.7)
title("Genes en común entre las tres comparaciones\n Genes seleccionados con FDR < 0.1 and logFC > 1")
```

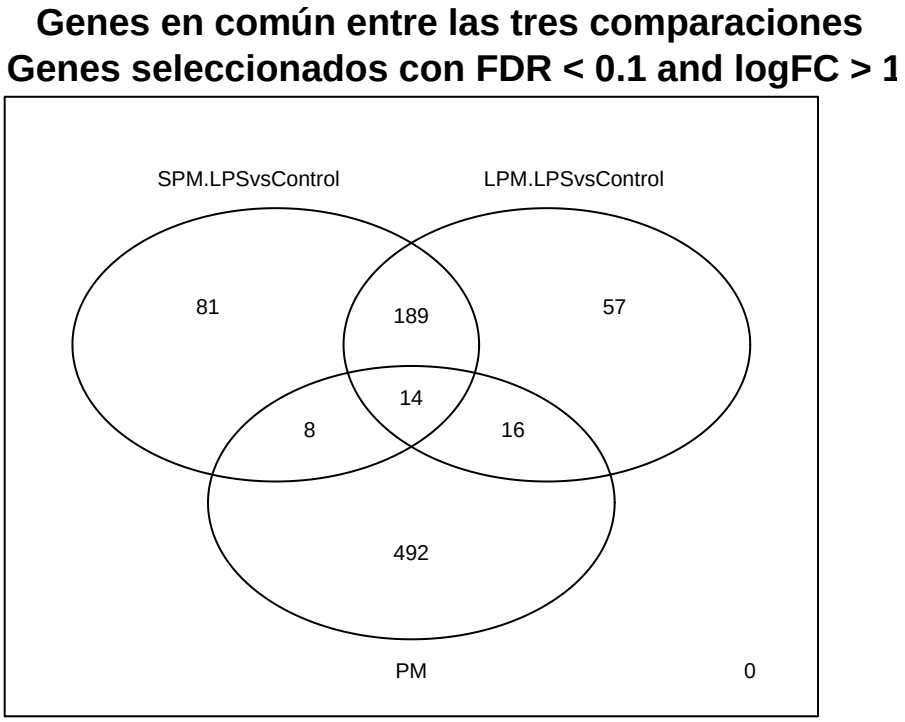

Figura 10: Gráfico de los genes en común entre las tres comparaciones realizadas (diagrama de Venn)

Heatmaps

Para visualizar los genes que han sido seleccionados como diferencialmente expresados, podemos utilizar un mapa de calor.

```
probesInHeatmap <- rownames(res.selected)
HMdata <- exprs(eset_filtered)[rownames(exprs(eset_filtered)) %in% probesInHeatmap,]

geneSymbols <- select(clariomdhumantranscriptcluster.db, rownames(HMdata), c("SYMBOL"))
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
SYMBOLS<- geneSymbols$SYMBOL
rownames(HMdata) <- SYMBOLS
write.csv(HMdata, file = file.path("./results/data4Heatmap.csv"))
```

```
my_palette <- colorRampPalette(c("blue", "red"))(n = 299)
library(gplots)
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:IRanges':
##
##     space
```

```
## The following object is masked from 'package:S4Vectors':
##
##     space
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
heatmap.2(HMdata,
          Rowv = FALSE,
          Colv = FALSE,
          main = "Genes diferencialmente expresados \n FDR < 0,1, logFC >=1",
          scale = "row",
          col = my_palette,
          sepcolor = "white",
          sepwidth = c(0.05,0.05),
          cexRow = 0.5,
          cexCol = 0.9,
          key = TRUE,
          keysize = 1.5,
          density.info = "histogram",
          ColSideColors = c(rep("brown",4),rep("green",4), rep("orange",4), rep("purple",4)),
          tracecol = NULL,
          dendrogram = "none",
          srtCol = 30)
```

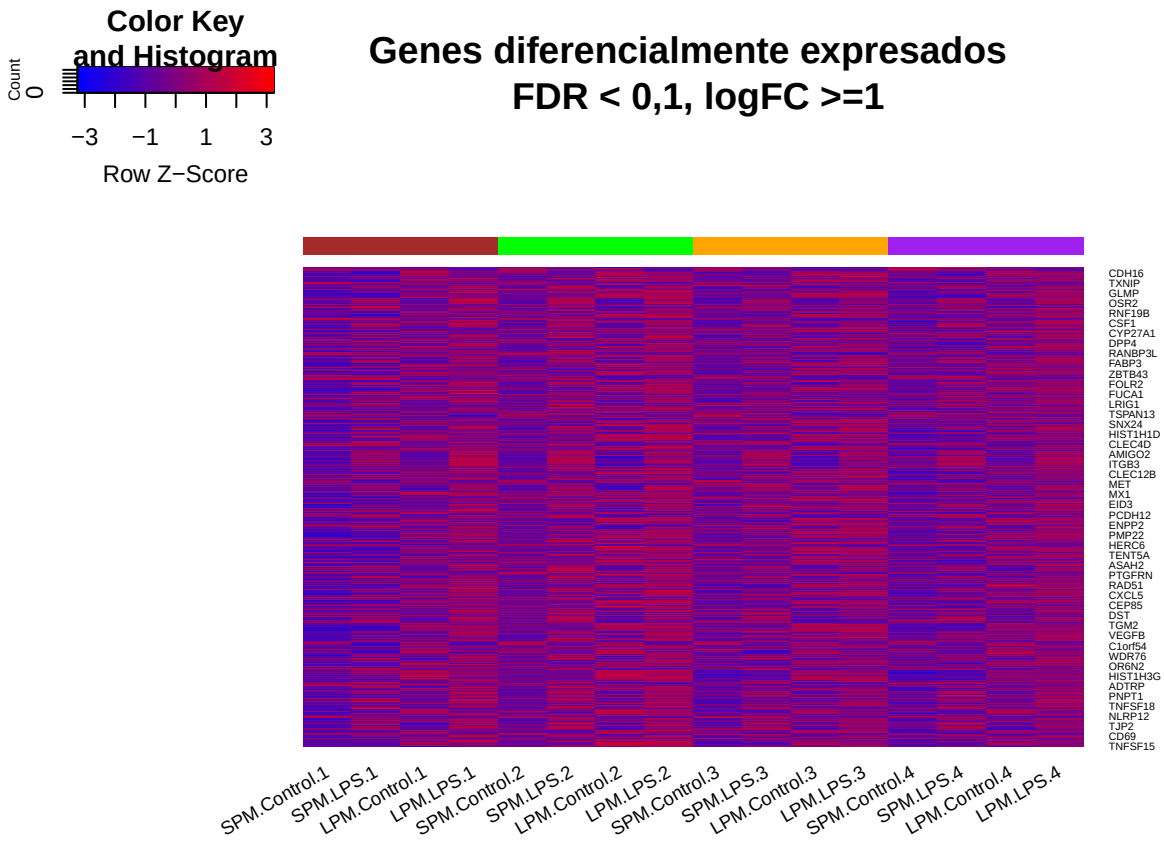

Figura 11: Mapa de calor para datos de expresión sin ninguna agrupación

A continuación, mostramos un mapa de calor donde los genes y las muestras se agrupan por fila y columna de manera similar.

```
heatmap.2(HMdata,
          Rowv = TRUE,
          Colv = TRUE,
          dendrogram = "both",
          main = "Genes diferencialmente expresados \n FDR < 0,1, logFC >=1",
          scale = "row",
          col = my_palette,
          sepcolor = "white",
          sepwidth = c(0.05,0.05),
          cexRow = 0.5,
          cexCol = 0.9,
          key = TRUE,
          keysize = 1.5,
          density.info = "histogram",
          ColSideColors = c(rep("brown",4),rep("green",4), rep("orange",4), rep("purple",4)),
          tracecol = NULL,
          srtCol = 30)
```

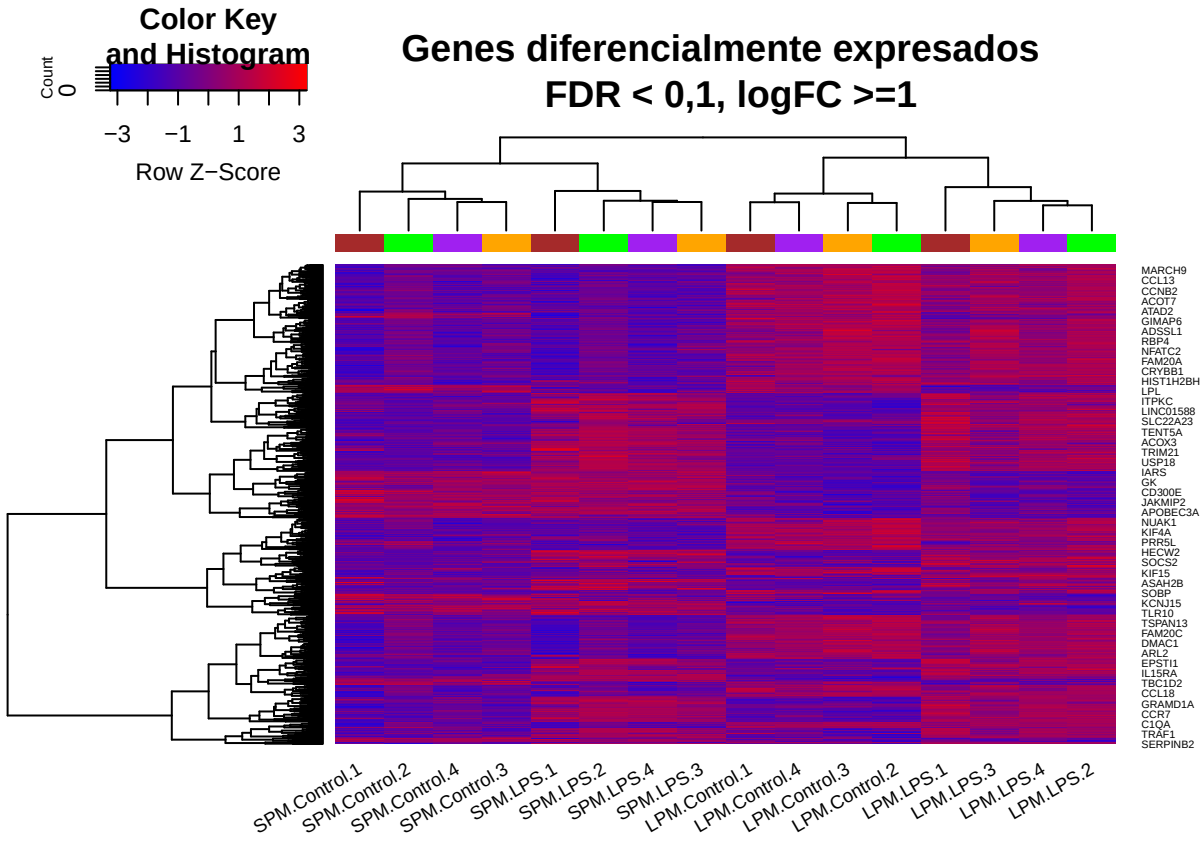

Figura 12: Mapa de calor para expresión de datos agrupando genes (filas) y muestras (columnas) por su similitud

Análisis de significación biológica

Primeramente, preparamos la lista de listas de genes que se analizarán.

```
listOfTables <- list(SPM.LPSvsControl = topTab_SPM.LPSvsControl,
                     LPM.LPSvsControl = topTab_LPM.LPSvsControl,
                     PM = topTab_PM)
listOfSelected <- list()
for (i in 1:length(listOfTables)){
  # select the toptable
  topTab <- listOfTables[[i]]
  # select the genes to be included in the analysis
  whichGenes <- topTab["adj.P.Val"]<0.15
  selectedIDs <- rownames(topTab)[whichGenes]
  # convert the ID to Entrez
  EntrezIDs<- select(clariomdhumantranscriptcluster.db, selectedIDs, c("ENTREZID"))
  EntrezIDs <- EntrezIDs$ENTREZID
  listOfSelected[[i]] <- EntrezIDs
  names(listOfSelected)[i] <- names(listOfTables)[i]
}
```

```
## 'select()' returned 1:1 mapping between keys and columns
## 'select()' returned 1:1 mapping between keys and columns
## 'select()' returned 1:1 mapping between keys and columns
```

```
sapply(listOfSelected, length)
```

```
## SPM.LPSvsControl LPM.LPSvsControl              PM
##             1129              955            2228
```

Una vez hemos obtenido la lista de genes que caracteriza la diferencia entre las condiciones, debemos interpretarla. Para ello, utilizaremos el análisis de enriquecimiento descrito en el paquete *ReactomePA* de Bioconductor. El análisis de significación biológica se aplicará solamente a las dos primeras listas.

```
library(ReactomePA)
```

```
##
```

```
## Registered S3 method overwritten by 'enrichplot':
##   method               from
##   fortify.enrichResult DOSE
```

```
## ReactomePA v1.28.0  For help: https://guangchuangyu.github.io/ReactomePA
##
## If you use ReactomePA in published research, please cite:
## Guangchuang Yu, Qing-Yu He. ReactomePA: an R/Bioconductor package for reactome pathway analysis and
```

```
listOfData <- listOfSelected[1:2]
comparisonsNames <- names(listOfData)
for (i in 1:length(listOfData)){
  genesIn <- listOfData[[i]]
  comparison <- comparisonsNames[i]
  enrich.result <- enrichPathway(gene = genesIn,
                                 pvalueCutoff = 0.05,
                                 readable = T,
                                 pAdjustMethod = "BH",
                                 organism = "human")

  cat("####################################")
  cat("\nComparison: ", comparison,"\n")
  print(head(enrich.result))
```

```
  if (length(rownames(enrich.result@result)) != 0) {
  write.csv(as.data.frame(enrich.result),
            file =paste0("./results/","ReactomePA.Results.",comparison,".csv"),
            row.names = FALSE)

  pdf(file=paste0("./results/","ReactomePABarplot.",comparison,".pdf"))
    print(barplot(enrich.result, showCategory = 15, font.size = 4,
            title = paste0("Reactome Pathway Analysis for ", comparison,". Barplot")))
  dev.off()

  pdf(file = paste0("./results/","ReactomePAcnetplot.",comparison,".pdf"))
    print(cnetplot(enrich.result, categorySize = "geneNum", schowCategory = 15,
          vertex.label.cex = 0.75))
  dev.off()
  }
}
```

```
## ####################################
## Comparison:  SPM.LPSvsControl
##                           ID                          Description GeneRatio
## R-HSA-909733   R-HSA-909733     Interferon alpha/beta signaling    28/688
## R-HSA-913531   R-HSA-913531                Interferon Signaling    48/688
## R-HSA-6783783 R-HSA-6783783            Interleukin-10 signaling    22/688
## R-HSA-877300   R-HSA-877300          Interferon gamma signaling    26/688
## R-HSA-449147   R-HSA-449147           Signaling by Interleukins    63/688
## R-HSA-380108   R-HSA-380108 Chemokine receptors bind chemokines    14/688
##                  BgRatio       pvalue     p.adjust       qvalue
## R-HSA-909733   69/10619 4.123931e-16 3.415055e-13 3.267689e-13
## R-HSA-913531  199/10619 6.407233e-16 3.415055e-13 3.267689e-13
## R-HSA-6783783  47/10619 1.635378e-14 5.811044e-12 5.560286e-12
## R-HSA-877300   92/10619 7.574064e-11 2.018488e-08 1.931386e-08
## R-HSA-449147  463/10619 1.141065e-08 2.432751e-06 2.327773e-06
## R-HSA-380108   48/10619 1.232245e-06 2.189288e-04 2.094816e-04
##
## R-HSA-909733
## R-HSA-913531
## R-HSA-6783783
## R-HSA-877300
## R-HSA-449147  IL6/TSLP/PTGS2/TNF/PELI1/CLCF1/SOCS3/CRLF2/CXCL1/IL10/CSF3/CCL20/IL23A/CXCL2/STAT1/CCL
## R-HSA-380108
##               Count
## R-HSA-909733     28
## R-HSA-913531     48
## R-HSA-6783783    22
## R-HSA-877300     26
## R-HSA-449147     63
## R-HSA-380108     14
```

```
## ####################################
## Comparison:  LPM.LPSvsControl
##                           ID                          Description GeneRatio
## R-HSA-913531   R-HSA-913531                Interferon Signaling    45/560
## R-HSA-6783783 R-HSA-6783783            Interleukin-10 signaling    22/560
## R-HSA-909733   R-HSA-909733     Interferon alpha/beta signaling    26/560
```

Cuadro 3: Primeras filas y columnas de los resultados Reactome sobre la comparación entre LPM en LPS y Control

	Description	GeneRatio	BgRatio	pvalue	p.adjust
R-HSA-913531	Interferon Signaling	45/560	199/10619	3.10786487103024e-17	2.97422668157594e-
R-HSA-6783783	Interleukin-10 signaling	22/560	47/10619	2.21361291934674e-16	8.96952216854335e-
R-HSA-909733	Interferon alpha/beta signaling	26/560	69/10619	2.81176243528005e-16	8.96952216854335e-
R-HSA-877300	Interferon gamma signaling	26/560	92/10619	7.21594104749765e-13	1.72641389561381e-

```
## R-HSA-877300   R-HSA-877300          Interferon gamma signaling    26/560
## R-HSA-449147   R-HSA-449147           Signaling by Interleukins    53/560
## R-HSA-380108   R-HSA-380108 Chemokine receptors bind chemokines    12/560
##                 BgRatio       pvalue     p.adjust       qvalue
## R-HSA-913531  199/10619 3.107865e-17 2.974227e-14 2.927936e-14
## R-HSA-6783783  47/10619 2.213613e-16 8.969522e-14 8.829921e-14
## R-HSA-909733   69/10619 2.811762e-16 8.969522e-14 8.829921e-14
## R-HSA-877300   92/10619 7.215941e-13 1.726414e-10 1.699544e-10
## R-HSA-449147  463/10619 6.418564e-08 1.228513e-05 1.209393e-05
## R-HSA-380108   48/10619 4.996746e-06 7.969810e-04 7.845768e-04
##
## R-HSA-913531                                                                ISG15/IFIT3/IFIT2/OAS
## R-HSA-6783783
## R-HSA-909733
## R-HSA-877300
## R-HSA-449147  TSLP/PTGS2/IL6/PELI1/SOCS3/TNF/MMP3/CLCF1/IL1R1/IL1A/CXCL10/CXCL2/CCL5/CXCL1/IL10RA/CSF
## R-HSA-380108
##               Count
## R-HSA-913531     45
## R-HSA-6783783    22
## R-HSA-909733     26
## R-HSA-877300     26
## R-HSA-449147     53
## R-HSA-380108     12
```

De esta manera hemos obtenido 3 ficheros como resultado del análisis de importancia biológica: + Archivo .csv con un resumen de todas las rutas enriquecidas y las estadísticas asociadas. + Un diagrama de barras con las mejores vías enriquecidas. La altura del gráfico de barras es el número de genes de nuestro análisis relacionados con esa vía. Además, las vías están ordenadas por significación estadística. + Una trama con una red de las vías enriquecidas y la relación entre los genes incluidos.

Vamos a mostrar la red producida a partir de los genes asociados en la comparación LPM en LPS y Control.

```
cnetplot(enrich.result, categorySize = "geneNum", schowCategory = 15, vertex.label.cex = 0.75)
```

Podemos observar que se han encontrado 5 vías enriquecidas.

Resultados

De los resultados obtenidos se puede concluir que existen diferencias entre los dos tipos de macrófagos diferenciados, es decir, entre los macrófagos peritoneales pequeños (SPM) y grandes (LPM). Como consecuencia de esta diferencia, la aplicación del tratamiento LPS actúa de manera distinta según el tipo de macrófago sobre el que se aplica. Específicamente, se han encontrado más genes diferencialmente expresados en las muestras de LPMs con la aplicación de LPS respecto el grupo control que las muestras de SPMs.

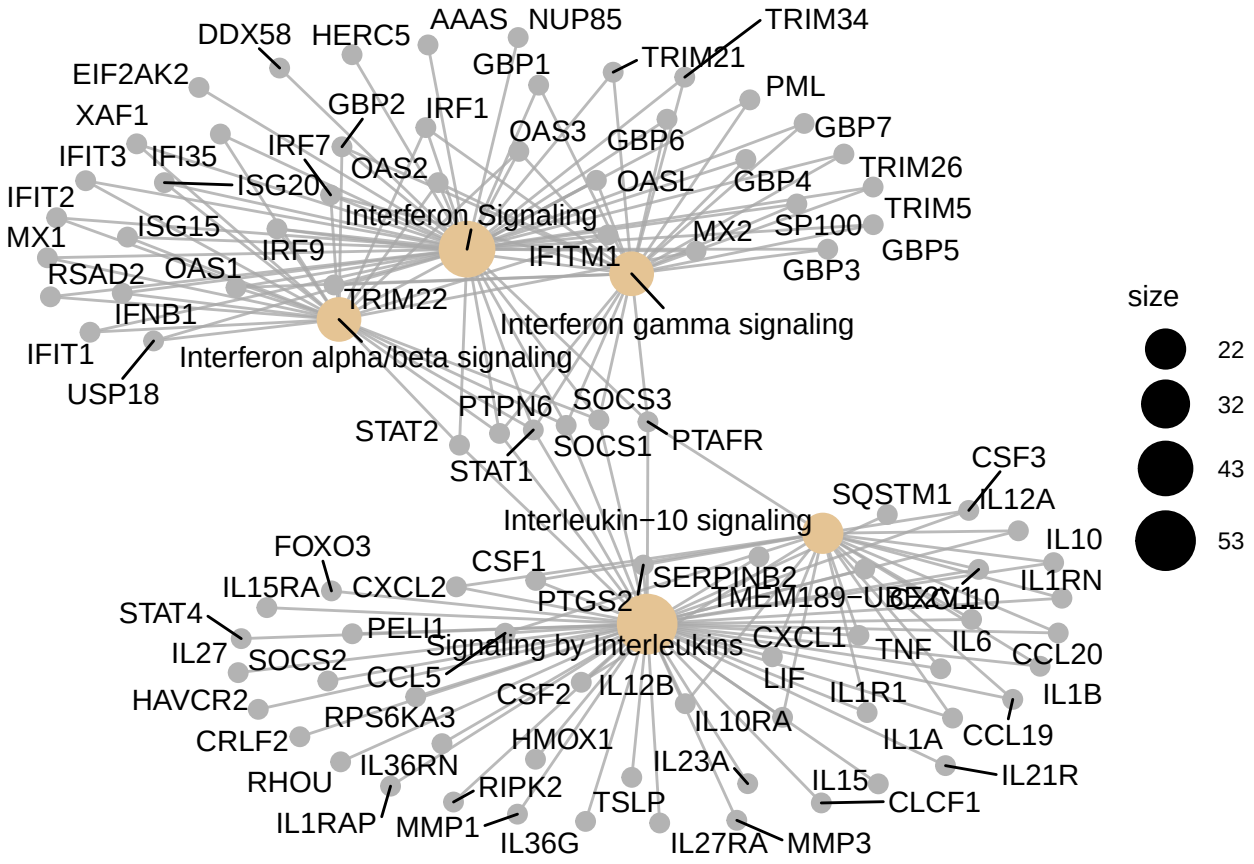

Figura 13: Red obtenida del análisis de enriquecimiento de Reactome en la lista obtenida de la comparación entre LPM en LPS y Control

Cuadro 4: Lista de archivos generados en el análisis

Lista de archivos
data4Heatmap.csv
normalized.Data.csv
normalized.Data.Rda
normalized.Filtered.Data.csv
ReactomePA.Results.LPM.LPSvsControl.csv
ReactomePA.Results.SPM.LPSvsControl.csv
ReactomePABarplot.LPM.LPSvsControl.pdf
ReactomePABarplot.SPM.LPSvsControl.pdf
ReactomePAcnetplot.LPM.LPSvsControl.pdf
ReactomePAcnetplot.SPM.LPSvsControl.pdf
topAnnotated_LPM.LPSvsControl.csv
topAnnotated_PM.csv
topAnnotated_SPM.LPSvsControl.csv

Resumen de los resultados

En la siguiente tabla se muestran los archivos con los resultados del análisis, lo cuales son muy útiles para discutir los resultados biológicamente.

Referencias

Stengel S, Lutz P, Quickert S, y Bruns T. 2020. «Peritoneal Level of CD206 Associates With Mortality and an Inflammatory Macrophage Phenotype in Patients With Decompensated Cirrhosis and Spontaneous Bacterial Peritonitis». https://doi.org/10.1053/j.gastro.2020.01.029.