PEC 1 Análisis de Datos Ómicos

Mónica Hortal Foronda

25/04/2020

Repositorio GitHub: *https://github.com/monicahortal/PEC-1-Analisis-de-Datos-Omicos.git*

1. Abstract

En estudios previos se ha observado que la inmunosupresión mejora la inmunidad antitumoral. En este estudio se evalúa la eficacia de la inhibición del receptor del factor 1 estimulante de colonias (CSF-1R) BLZ945 y del anticuerpo frente a PD1/PDL1 Nivolumab, así como la de ambas estrategias en conjunto como terapia antitumoral. Para analizar la eficacia de las distintas estrategias, se realizan arrays de expresión a cuatro grupos de muestras (control, en presencia de BLZ945, en presencia de anticuerpo frente a PD1/PDL1 y con ambos tratamientos), lo que permite ver cuáles son los genes diferencialmente expresados entre grupos. Para realizar este análisis, utilizaremos el paquete Bioconductor dentro del programa R.

2. Objetivos

Con este estudio, el objetivo es conocer si el inhibidor de CSF-1R BLZ945 y la inhibición mediante anticuerpos de PD1/PDL1 son una buena estrategia para realizar tratamientos antitumorales. Para ello, analizaremos las diferencias en los niveles de expresión de los distintos genes. En nuestro caso, ocupamos un rol técnico en este estudio, por lo que nuestro objetivo será realizar el análisis que permita detectar estas diferencias en la expresión de los genes en células de ratón con los distintos tratamientos.

3. Materiales y métodos

3.1. Naturaleza de los datos, tipo de experimento, diseño experimental y tipo de microarrays utilizados

Para analizar las diferencias de expresión génica con los distintos tratamientos se ha utilizado el array de Affymetrix Mouse Gene 2.1 ST Array [MoGene-2_1-st]. Se han utilizado un total de 16 de estos arrays, 4 para cada uno de los 4 grupos.

Los datos obtenidos de se han guardado en ficheros con formato .CEL, uno por cada array, y están disponibles en la base de datos Gene Expression Omnibus (GEO), perteneciente al NCBI. Estos son los ficheros que procesaremos para realizar el análisis.

Los datos que vamos a utilizar se encuentran disponibles en el número de acceso **GSE79485** y pertenecen al estudio publicado como **Eissler N, Mao Y, Brodin D, Reuterswärd P et al. Regulation of myeloid cells by activated T cells determines the efficacy of PD-1 blockade. Oncoimmunology 2016;5(12):e1232222.**

3.2. Software

Para realizar el análisis debemos instalar el programa **R statistical software** que podemos descargar desde la página de R Project (https://www.r-project.org/).Para facilitar su uso podemos descargar la interfaz RStudio desde su página web (https://www.rstudio.com/).

A la hora de realizar este análisis necesitaremos funciones que no están disponibles en la instalación básica de R, por lo que necesitamos instalar algunas librerías. La mayoría de los que necesitamos son del proyecto Bioconductor.

3.3. Análisis de los datos

El orden del procedimiento es el siguiente:

- Preparación del entorno.
- Creación del fichero *targets*.
- Descarga de las librerías y lectura de los ficheros .CEL.
- Control de calidad de los datos brutos.
- Normalización de los datos y control de calidad de los datos normalizados.
- Batch detection.
- Identificación de genes diferencialmente expresados y filtrado.
- Diseño de la matriz, definición de comparaciones y estimación del modelo.
- Obtención de listados de los genes diferencialmente expresados.
- Anotación de genes y visualización de la expresión diferencial.
- Comparación entre distintas comparaciones.
- Mapas de calor.
- Significación biológica de los resultados obtenidos.

3.3.1. Preparación del entorno:

Para manejar los distintos archivos que van a formar parte del análisis, creamos distintas carpetas. En primer lugar, creamos la carpeta **"ADO"**, que será nuestro *working directory*. Su ruta es "*C:/Users/monic/Desktop/ADO*". Dentro de esta carpeta creamos otras dos: **"data"**, donde guardamos los ficheros .CEL y el archivo *targets* y **"results"**, que es donde se enviarán los resultados obtenidos durante el análisis de microarrays. Estas carpetas se han creado desde el explorador de archivos de Windows.

```
> setwd("C:/Users/monic/Desktop/ADO")
```

3.3.2. Creación del fichero *targets*

Para el análisis de los datos son necesarios los archivos .CEL que hemos descargado y el fichero *targets*, que contiene la información sobre los grupos y las variables. Este archivo nos permite relacionar el nombre de cada archivo .CEL con su condición en el experimento.

En este archivo, hemos creado las siguientes columnas:

- **FIleName:** Contiene el nombre de los archivos .CEL.
- **Group:** Indica el grupo al que pertenece (control, BLZ, PD1_PDL1 y combo).
- **BLZ:** Indica si se ha añadido BLZ en esa muestra.
- **PD1_PDL1:** Indica si se ha añadido PD1_PDL1 en esa muestra.
- **ShortName:** Nombre de la muestra que se utilizará para las leyendas de los gráficos.

```
> targets <- read.csv("C:/Users/monic/Desktop/ADO/data/targets.csv", sep=";")
> knitr::kable(targets, booktabs = TRUE,
+   caption = 'Tabla 1: contenido del fichero targets utilizado en este
análisis')
```

Tabla 1: Contenido del fichero targets utilizado en este análisis

FileName	Group	BLZ	PD1_PDL1	ShortName
GSM2095868	ctrl	NO	NO	ctrl_1
GSM2095869	ctrl	NO	NO	ctrl_2
GSM2095870	ctrl	NO	NO	ctrl_3
GSM2095871	ctrl	NO	NO	ctrl_4
GSM2095872	BLZ	YES	NO	BLZ_1
GSM2095873	BLZ	YES	NO	BLZ_2
GSM2095874	BLZ	YES	NO	BLZ_3
GSM2095876	PD1_PDL1	NO	YES	PD1_PDL1_1
GSM2095877	PD1_PDL1	NO	YES	PD1_PDL1_2
GSM2095878	PD1_PDL1	NO	YES	PD1_PDL1_3
GSM2095879	PD1_PDL1	NO	YES	PD1_PDL1_4
GSM2095880	combo	YES	YES	combo_1
GSM2095881	combo	YES	YES	combo_2
GSM2095882	combo	YES	YES	combo_3
GSM2095883	combo	YES	YES	combo_4

Cabe señalar que, una vez realizado el control de calidad de los datos brutos, encontramos que una de las muestras (el array 8, correspondiente a la muestra BLZ_4), no cumplía los requisitos de calidad al tener asteriscos en las tres columnas. Por tanto, retiramos su archivo .CEL de la carpeta "data" y eliminamos su línea del archivo *targets*. El proceso que se muestra de ahora en adelante es sin esta muestra, únicamente con 3 arrays en el grupo BLZ.

3.3.3. Descarga de las librerías y lectura de los ficheros .CEL.

Para poder realizar el análisis debemos instalar los paquetes necesarios que no están disponibles en la instalación básica. Los paquetes estándar se descargan desde el repositorio CRAN con la función `install.packages`, mientras los paquetes de Bioconductor se descargan con la función `install()` del paquete `BiocManager`.

En primer lugar, instalamos Bioconductor:

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
+     install.packages("BiocManager")
> BiocManager::install()
```

A continuación, descargamos los paquetes:

```
> install.packages("knitr")
> install.packages("colorspace")
> install.packages("gplots")
> install.packages("ggplot2")
> install.packages("ggrepel")
> install.packages("htmlTable")
> install.packages("prettydoc")
> install.packages("devtools")
> install.packages("BiocManager")
> BiocManager::install("oligo")
> BiocManager::install("pd.mogene.2.1.st")
> BiocManager::install("arrayQualityMetrics")
> BiocManager::install("pvca")
> # NOT NEEDED UNTIL ANALYSES ARE PERFORMED
> BiocManager::install("limma")
> BiocManager::install("genefilter")
> BiocManager::install("mogene21sttranscriptcluster.db")
> BiocManager::install("annotate")
> BiocManager::install("org.Mm.eg.db")
> BiocManager::install("ReactomePA")
> BiocManager::install("reactome.db")
```

El siguiente paso es leer los datos brutos (archivos CEL) y almacenarlos en una variable, que en este caso hemos llamado **rawData**. Cargamos el paquete `oligo` para leer los archivos .CEL e indicamos la ruta para la carpeta "data".

```
> library(oligo)
> celFiles <- list.celfiles("C:/Users/monic/Desktop/ADO/data", full.names =
TRUE)
> library(Biobase)
> my.targets <-
read.AnnotatedDataFrame(file.path("C:/Users/monic/Desktop/ADO/data","targets.
csv"),
+                                     header = TRUE, row.names = 1,
+                                     sep=";")
> rawData <- read.celfiles(celFiles, phenoData = my.targets)
```

```
> my.targets@data$ShortName->rownames(pData(rawData))
> colnames(rawData) <-rownames(pData(rawData))
>
> head(rawData)

GeneFeatureSet (storageMode: lockedEnvironment)
assayData: 1 features, 15 samples
  element names: exprs
protocolData
  rowNames: ctrl_1 ctrl_2 ... combo_4 (15 total)
  varLabels: exprs dates
  varMetadata: labelDescription channel
phenoData
  rowNames: ctrl_1 ctrl_2 ... combo_4 (15 total)
  varLabels: Group BLZ PD1_PDL1 ShortName
  varMetadata: labelDescription channel
featureData: none
experimentData: use 'experimentData(object)'
Annotation: pd.mogene.2.1.st
```

3.3.4. Control de calidad de los datos brutos:

En este paso vamos a analizar si los datos de los distintos arrays tienen la calidad suficiente para introducirlos en el análisis y normalizarlos. Este paso es importante ya que arrays sin la suficiente calidad podrían alterar el proceso de normalización e introducir mucho ruido.

Para el control de calidad utilizamos el paquete `ArrayQualityMetrics`.

```
> library(arrayQualityMetrics)
> arrayQualityMetrics(rawData)
```

Para analizar los resultados, abrimos la carpeta QCDir.Raw que se ha creado al analizar los datos brutos. En ella, encontramos entre los archivos uno llamado *index.html* que contiene el resumen del análisis. En primer lugar, vemos una tabla que contiene todos los archivos .CEL indicados en el fichero *targets* y nos proporciona información sobre su calidad. Esta tabla tiene tres columnas que indican distintos parámetros de calidad y debemos descartar aquellos arrays que tengan un asterisco en las tres columnas.

	array	sampleNames	*1	*2	*3	Group	BLZ	PD1_PDL1	ShortName
☐	1	ctrl_1				ctrl	NO	NO	ctrl_1
☐	2	ctrl_2				ctrl	NO	NO	ctrl_2
☐	3	ctrl_3				ctrl	NO	NO	ctrl_3
☐	4	ctrl_4				ctrl	NO	NO	ctrl_4
☐	5	BLZ_1				BLZ	YES	NO	BLZ_1
☐	6	BLZ_2				BLZ	YES	NO	BLZ_2
☐	7	BLZ_3				BLZ	YES	NO	BLZ_3
☐	8	PD1_PDL1_1				PD1_PDL1	NO	YES	PD1_PDL1_1
☐	9	PD1_PDL1_2				PD1_PDL1	NO	YES	PD1_PDL1_2
☐	10	PD1_PDL1_3				PD1_PDL1	NO	YES	PD1_PDL1_3
☐	11	PD1_PDL1_4				PD1_PDL1	NO	YES	PD1_PDL1_4
☐	12	combo_1				combo	YES	YES	combo_1
☐	13	combo_2			x	combo	YES	YES	combo_2
☐	14	combo_3				combo	YES	YES	combo_3
☐	15	combo_4	x			combo	YES	YES	combo_4

Tabla 2: Información sobre la calidad de los arrays del estudio.

Como se ha explicado anteriormente, la muestra BLZ_4 fue eliminada tras un primer control de calidad para mejorar la calidad general del experimento. El proceso que se muestra es la repetición una vez eliminado este array.

Continuamos el control de calidad analizando la contribución de las componentes principales (PCA).

```
> library(ggplot2)
> library(ggrepel)
> plotPCA3 <- function (datos, labels, factor, title, scale,colores, size =
1.5, glineas = 0.25) {
+   data <- prcomp(t(datos),scale=scale)
+   # plot adjustments
+   dataDf <- data.frame(data$x)
+   Group <- factor
+   loads <- round(data$sdev^2/sum(data$sdev^2)*100,1)
+   # main plot
+   p1 <- ggplot(dataDf,aes(x=PC1, y=PC2)) +
+     theme_classic() +
+     geom_hline(yintercept = 0, color = "gray70") +
+     geom_vline(xintercept = 0, color = "gray70") +
+     geom_point(aes(color = Group), alpha = 0.55, size = 3) +
+     coord_cartesian(xlim = c(min(data$x[,1])-5,max(data$x[,1])+5)) +
+     scale_fill_discrete(name = "Group")
+   # avoiding labels superposition
+   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("Principal Component Analysis for: ",title,sep=" "))+
+     theme(plot.title = element_text(hjust = 0.5)) +
+     scale_color_manual(values=colores)
+ }

> plotPCA3(exprs(rawData), labels = targets$ShortName, factor =
targets$Group,
+          title="Raw data", scale = FALSE, size = 3,
+          colores = c("red", "blue", "green", "yellow"))
```

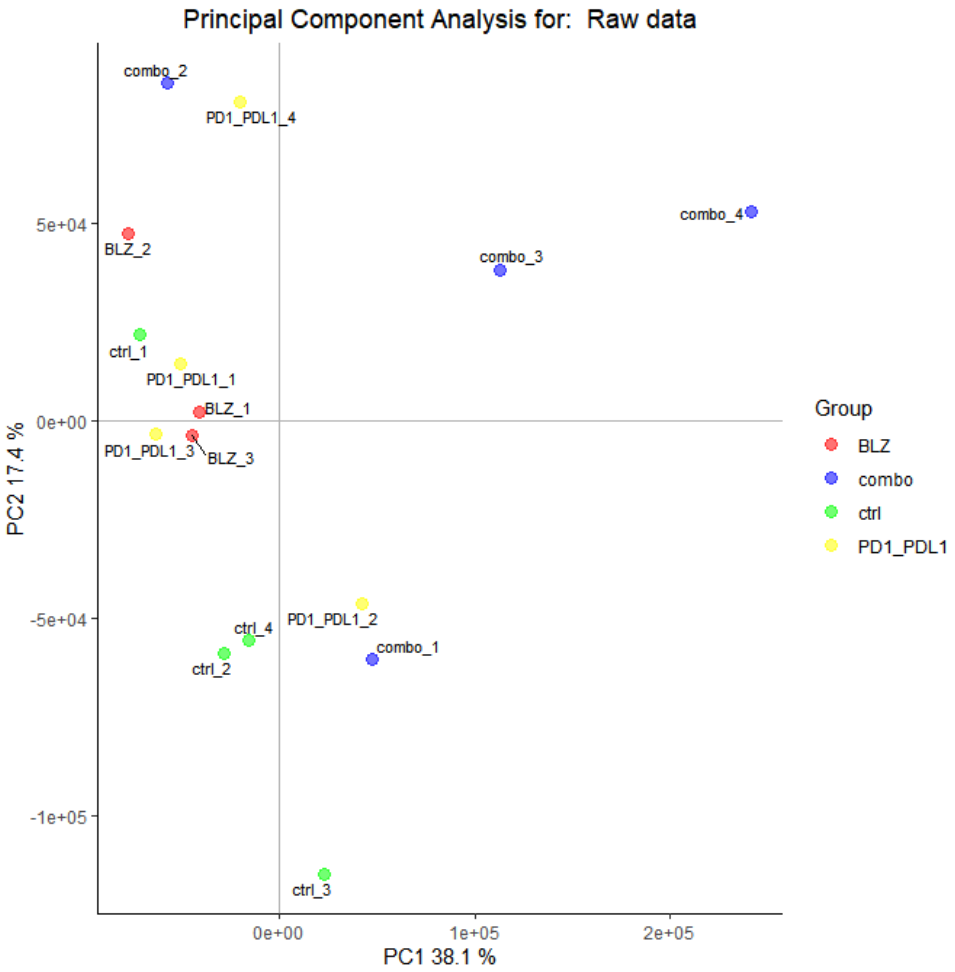

Gráfico 1: Visualización de las dos componentes principales para los datos brutos.

La primera componente principal explica el 38.1% de la variabilidad total de las muestras y la segunda componente principal, el 17.4%, como podemos observar en la gráfica. Viendo la distribución de los puntos en el gráfico es difícil saber qué variable corresponde a cada componente y no podríamos distinguir en qué eje se representa BLZ y PD1_PDL1, ya que los grupos en los que se ha añadido uno de los dos compuestos están muy juntos en la gráfica.

También podemos observar la distribución de intensidad de las matrices usando boxplots.

```
> boxplot(rawData, cex.axis=0.5, las=2,  which="all",
+         col = c(rep("red", 4), rep("blue", 3), rep("green", 4),
rep("yellow", 4)),
+         main="Distribution of raw intensity values")
```

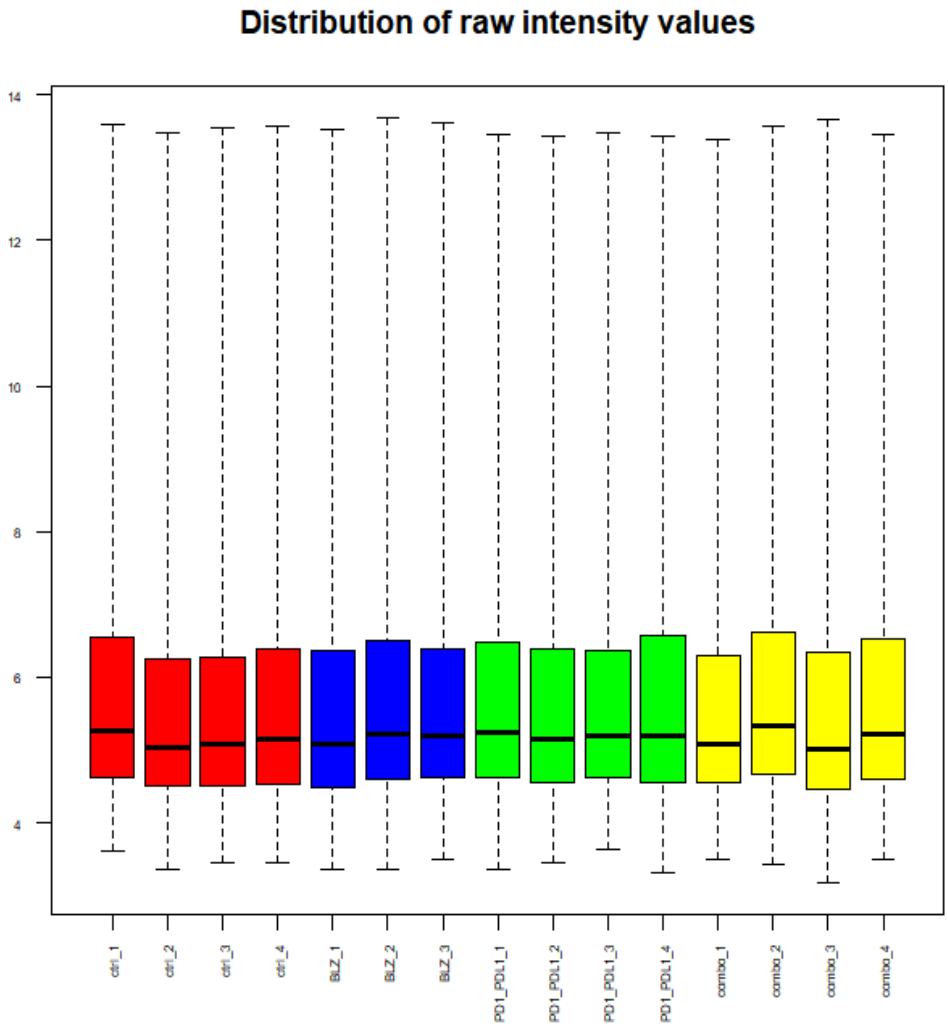

Gráfico 2: Boxplot para la intensidad de los arrays con datos brutos (Raw Data).

Vemos que hay variaciones de intensidad entre los distintos arrays, pero éstas son pequeñas y aceptables tratándose de los datos brutos. A continuación, procedemos a la normalización de los datos para eliminar estas diferencias.

3.3.5. Normalización de los datos y control de calidad de los datos normalizados:

Para poder comparar los arrays entre sí hemos de reducir la variabilidad en las muestras que no se deba a razones biológicas y evitar que haya sesgos artificiales debidos a problemas técnicos.

```
> eset_rma <- rma(rawData)

Background correcting
Normalizing
Calculating Expression
```

Una vez que hemos normalizado los datos, volvemos a realizar el control de calidad con el paquete `arrayQualityMetrics`. En este caso, en lugar de analizar *rawData*, analizamos *eset_rma*.

```
> arrayQualityMetrics(eset_rma, outdir =
file.path("C:/Users/monic/Desktop/ADO/results", "QCDir.Norm"), force=TRUE)
```

	array	sampleNames	*1	*2	*3	Group	BLZ	PD1_PDL1	ShortName
☐	1	ctrl_1				ctrl	NO	NO	ctrl_1
☐	2	ctrl_2				ctrl	NO	NO	ctrl_2
☐	3	ctrl_3				ctrl	NO	NO	ctrl_3
☐	4	ctrl_4				ctrl	NO	NO	ctrl_4
☐	5	BLZ_1				BLZ	YES	NO	BLZ_1
☐	6	BLZ_2				BLZ	YES	NO	BLZ_2
☐	7	BLZ_3				BLZ	YES	NO	BLZ_3
☐	8	PD1_PDL1_1				PD1_PDL1	NO	YES	PD1_PDL1_1
☐	9	PD1_PDL1_2				PD1_PDL1	NO	YES	PD1_PDL1_2
☐	10	PD1_PDL1_3				PD1_PDL1	NO	YES	PD1_PDL1_3
☐	11	PD1_PDL1_4				PD1_PDL1	NO	YES	PD1_PDL1_4
☐	12	combo_1				combo	YES	YES	combo_1
☐	13	combo_2			x	combo	YES	YES	combo_2
☐	14	combo_3				combo	YES	YES	combo_3
☐	15	combo_4	x			combo	YES	YES	combo_4

Tabla 3: Información sobre la calidad de los arrays del estudio con los datos normalizados.

Vemos que todos los arrays tienen una calidad aceptable para introducirlos en el estudio.

```
> plotPCA3(exprs(eset_rma), labels = targets$ShortName, factor =
targets$Group,
+          title="Normalized data", scale = FALSE, size = 3,
+          colores = c("red", "blue", "green", "yellow"))
```

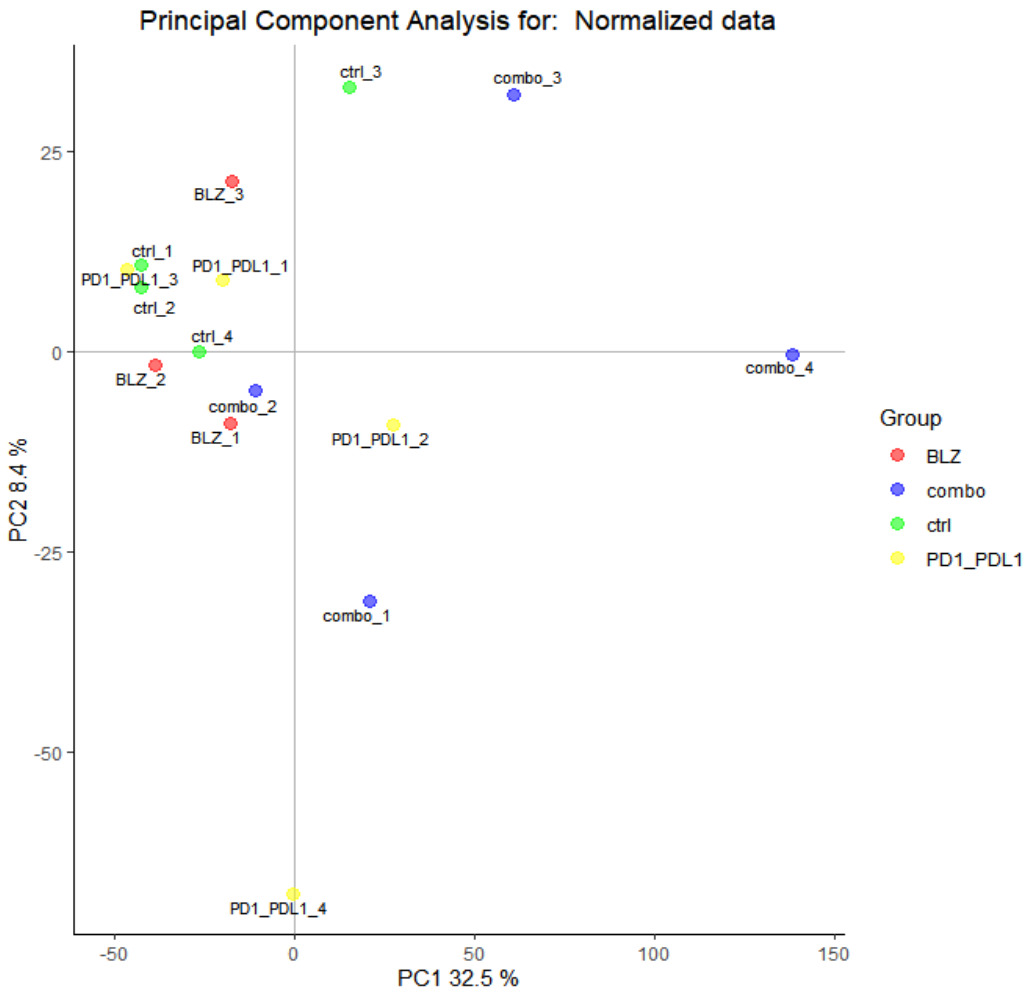

Gráfico 3: Visualización de las dos componentes principales para los datos normalizados.

Una vez normalizados los datos vemos que la primera componente principal representa el 32.5% de la variabilidad total. Este porcentaje ha disminuído respecto al obtenido con los datos brutos. Observando la distribución de los puntos en el gráfico es difícil saber qué variable corresponde a cada componente. Observamos que el grupo ctrl está a la izquierda del gráfico y el grupo combo está a la derecha, en líneas generales. Sin embargo, si observamos la distribución arriba-abajo, no vemos diferencias claras. Esto se debe a que la segunda componente principal solo explica el 8.4% de la variabilidad.

```
> boxplot(eset_rma, cex.axis=0.5, las=2,  which="all",
+          col = c(rep("red", 4), rep("blue", 3), rep("green", 4),
rep("yellow", 4)),
+          main="Boxplot for arrays intensity: Normalized Data")
```

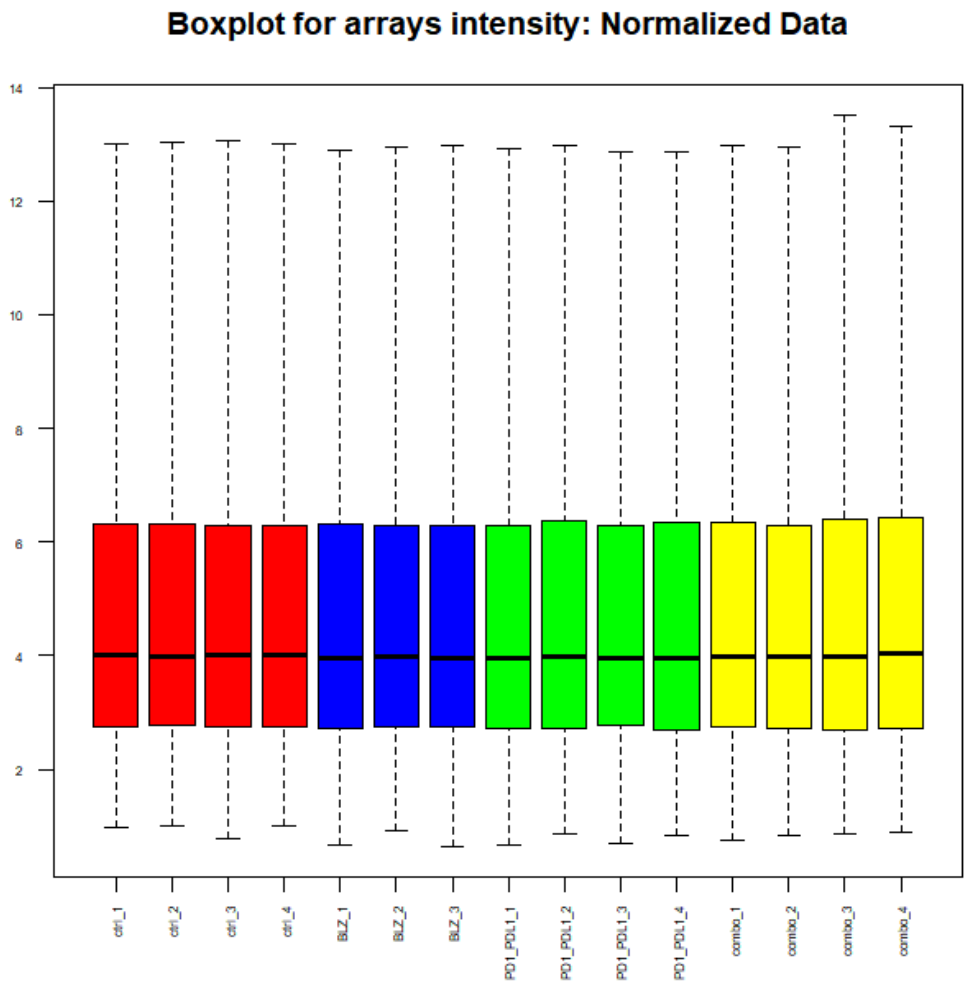

Gráfico 4: Boxplot para la intensidad de los arrays con datos normalizados.

En este boxplot vemos la distribución de las intensidades normalizadas de las muestras. Que todas estén a la misma altura nos indica que la normalización ha funcionado, por lo que podemos continuar con el análisis.

3.3.6. Batch detection:

Los resultados de microarrays de expresión génica pueden verse afectados por diferencias en variables no biológicas como pueden ser el uso de reactivos de distintos lotes, distintos operadores, diferencias de tiempo y lugar, etc. Para identificar estos efectos, vamos a utilizar el análisis de PVCA, por sus siglas en inglés *Principal variation component analysis*.

```
> #load the library
> library(pvca)
> pData(eset_rma) <- targets
> #select the threshold
> pct_threshold <- 0.6
```

```
> #select the factors to analyze
> batch.factors <- c("BLZ", "PD1_PDL1")
> #run the analysis
> pvcaObj <- pvcaBatchAssess (eset_rma, batch.factors, pct_threshold)

> #plot the results
> bp <- barplot(pvcaObj$dat, xlab = "Effects",
+   ylab = "Weighted average proportion variance",
+   ylim= c(0,1.1),col = c("mediumorchid"), las=2,
+   main="PVCA estimation")
> axis(1, at = bp, labels = pvcaObj$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)
```

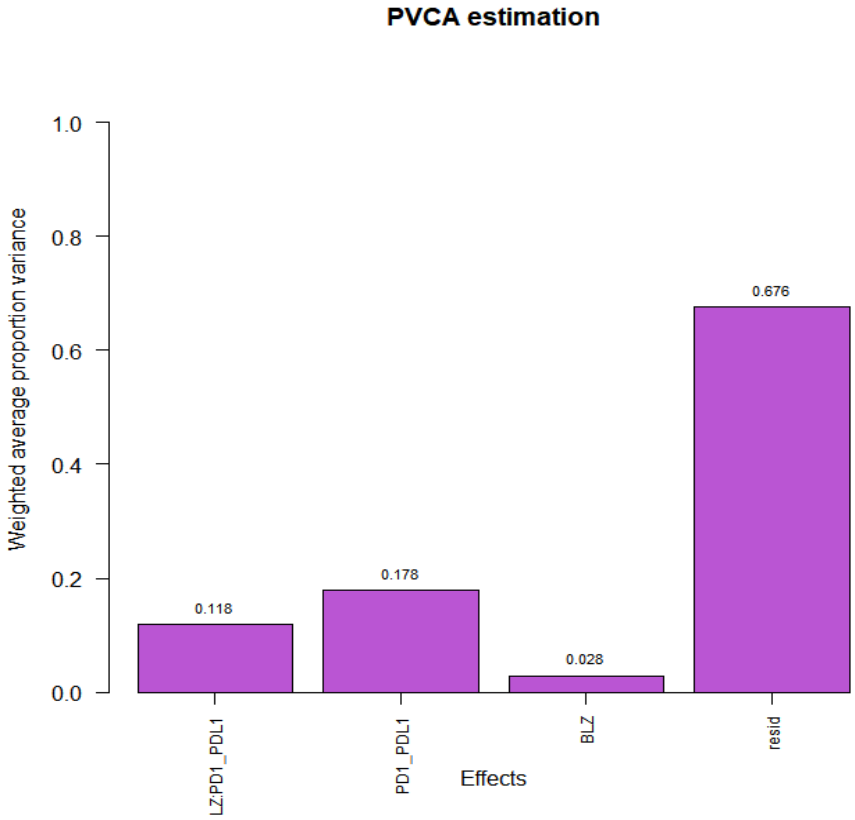

Gráfico 5: Importancia relativa de los factores del estudio en la variabilidad no biológica.

Tras este análisis obtenemos un diagrama de barras para las fuentes de variación incluidas en el análisis. Su tamaño relativo indica el porcentaje de variabilidad atribuible a cada fuente. En nuestro caso, la principal fuente de variación en las muestras es la condición *resid*.

3.3.7. Identificación de genes diferencialmente expresados y filtrado:

A continuación, vamos a detectar los genes que más expresión diferencial tienen. En el gráfico, se representan las desviaciones estándar de todos los genes ordenados de menor a mayor valor. Los genes más variables son aquellos con una desviación estándar superior al 90-95% de todas las desviaciones estándar. Estos valores se marcan con las líneas verticales.

```
> sds <- apply (exprs(eset_rma), 1, sd)
> sds0<- sort(sds)
> plot(1:length(sds0), sds0, main="Distribution of variability for all
genes",
+      sub="Vertical lines represent 90% and 95% percentiles",
+      xlab="Gene index (from least to most variable)", ylab="Standard
deviation")
> abline(v=length(sds)*c(0.9,0.95))
```

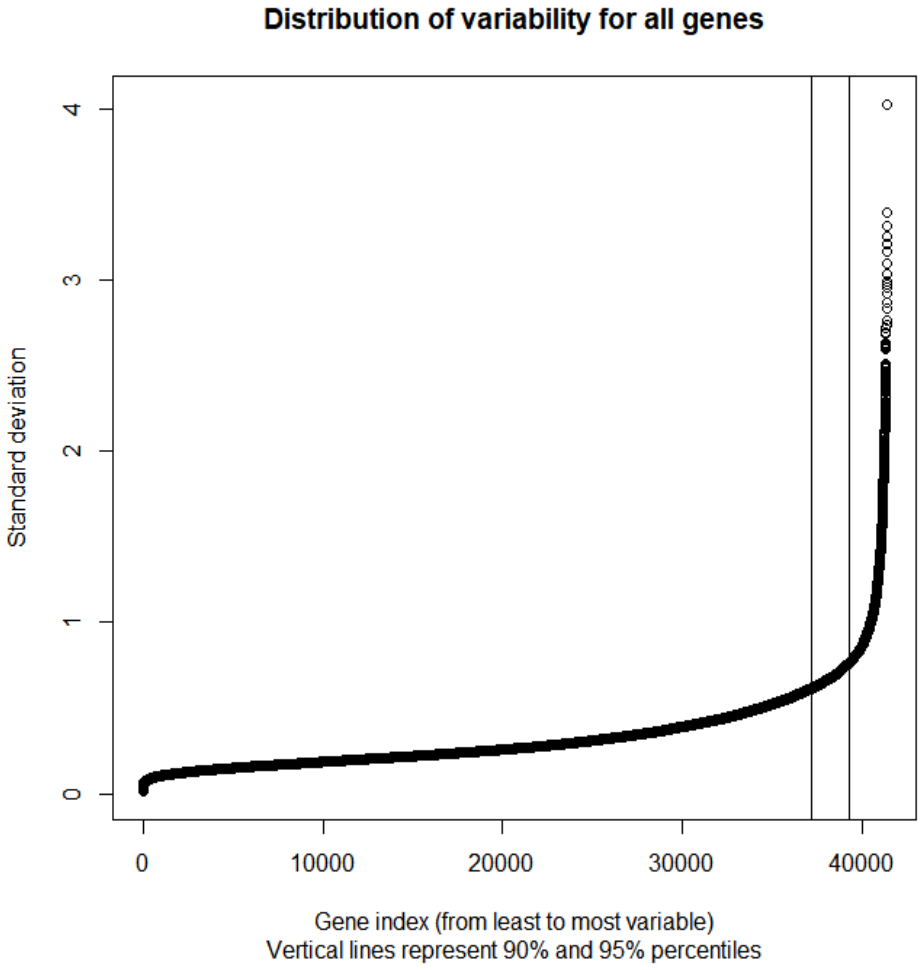

Gráfico 6: Valores de las desviaciones estándar en las muestras para todos los genes ordenados de menor a mayor.

Ahora debemos filtrar aquellos genes cuya variabilidad se puede atribuir a la variación aleatoria en lugar de a diferencias biológicas. Este método es útil para reducir el número de pruebas que se realizan.

Para ello, utilizamos la función `nsFilter` del paquete de Bioconductor `genefilter`, que permite eliminar genes en función de un umbral de variabilidad y nos devuelve los valores filtrados y un informe de los resultados. También utilizamos el paquete de anotaciones correspondiente al modelo de array en el que se ha hecho el experimento, en este caso, `mogene21sttranscriptcluster.db`.

```
> library(genefilter)
> library(mogene21sttranscriptcluster.db)
> annotation(eset_rma) <- "mogene21sttranscriptcluster.db"
> filtered <- nsFilter(eset_rma,
+                      require.entrez = TRUE, remove.dupEntrez = TRUE,
+                      var.filter=TRUE, var.func=IQR, var.cutoff=0.75,
+                      filterByQuantile=TRUE, feature.exclude = "^AFFX")

> print(filtered$filter.log)

$numDupsRemoved
[1] 672

$numLowVar
[1] 17986

$numRemoved.ENTREZID
[1] 16692

> eset_filtered <-filtered$eset
```

Después de filtrar, quedan 5995 genes. Los genes que quedan están almacenados en la variable *eset_filtered.*

Por último, guardamos los datos filtrados y normalizados, que son el punto de partida para futuros análisis. De esta forma, podríamos regresar a ellos para revisar algunos valores si lo deseamos.

```
> write.csv(exprs(eset_rma),
file="C:/Users/monic/Desktop/ADO/results/normalized.Data.csv")
> write.csv(exprs(eset_filtered),
file="C:/Users/monic/Desktop/ADO/results/normalized.Filtered.Data.csv")
> save(eset_rma, eset_filtered,
file="C:/Users/monic/Desktop/ADO/results/normalized.Data.Rda")
```

3.3.8. Diseño de la matriz, definición de comparaciones y estimación del modelo:

Para seleccionar genes diferencialmente expresados utilizaremos el paquete `limma`, que utiliza modelos lineales para realizar esta tarea.

En primer lugar, creamos la matriz de diseño, en la que se indica a qué grupo pertenece cada muestra. Esta matriz tiene tantas filas como muestras y tantas columnas como grupos.

```
> if (!exists("eset_filtered")) load
(file="C:/Users/monic/Desktop/ADO/results/normalized.Data.Rda")

> library(limma)
> designMat<- model.matrix(~0+Group, pData(eset_filtered))
> colnames(designMat) <- c("BLZ", "combo", "ctrl", "PD1_PDL1")
> print(designMat)

   BLZ combo ctrl PD1_PDL1
1    0     0    1        0
2    0     0    1        0
3    0     0    1        0
4    0     0    1        0
5    1     0    0        0
6    1     0    0        0
7    1     0    0        0
8    0     0    0        1
9    0     0    0        1
10   0     0    0        1
11   0     0    0        1
12   0     1    0        0
13   0     1    0        0
14   0     1    0        0
15   0     1    0        0
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$Group
[1] "contr.treatment"
```

Una vez que tenemos la matriz, podemos definir las comparaciones que queremos hacer. Estas comparaciones, llamadas contrastes, se representan por un 1 y un -1 en las filas de los grupos que se comparan y por 0 en el resto.

En este trabajo queremos observar el efecto de agregar el anticuerpo frente a PD1/PDL1 en presencia y ausencia del inhibidor BLZ. También queremos probar si hay interacción al juntar el anticuerpo frente a PD1/PDL1 y BLZ. Esto se puede ver haciendo las tres comparaciones que se describen a continuación:

```
> cont.matrix <- makeContrasts (BLZvscombo = BLZ-combo,
+                               ctrlvsPD1_PDL1 = ctrl-PD1_PDL1,
+                               INT = (BLZ-combo) - (ctrl-PD1_PDL1),
+                               levels=designMat)
> print(cont.matrix)
          Contrasts
Levels     BLZvscombo ctrlvsPD1_PDL1 INT
  BLZ               1              0   1
  combo            -1              0  -1
  ctrl              0              1  -1
  PD1_PDL1          0             -1   1
```

La matriz de contraste se define para tres comparaciones: efecto del anticuerpo frente a PD1/PDL1 en ausencia o presencia de BLZ e interacción entre dicho anticuerpo y BLZ.

Una vez que se han definido la matriz de diseño y los contrastes, podemos estimar el modelo y los contrastes y realizar las pruebas de significación estadística. Para ello, volvemos a utilizar el paquete `limma` que utiliza modelos empíricos de Bayes.

```
> 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"
```

3.3.9. Obtención de listados de los genes diferencialmente expresados.

Para obtener listados de los genes diferencialmente expresados utilizamos la función `topTable` del paquete `limma`. Este listado se ordena según el p-valor obtenido para cada gen, de menor a mayor. Por tanto, aquellos genes que aparecen arriba son los que tienen una mayor expresión diferencial.

Para cada gen, los parámetros estadísticos que se muestran son:

- `logFC`: diferencia media entre grupos.
- `AveExpr`: expresión promedio de todos los genes en la comparación.
- `t`: t-test para la comparación.
- `P.Value`: p-valor.
- `adj.P.Val`: p-valor ajustado.
- `B`: B-statistic

A continuación, podemos ver las primeras filas de cada `topTable`.

Para la comparación 1 (BLZvscombo): genes que varían sus niveles de expresión con la adición del anticuerpo frente a PD1/PDL1 en presencia de BLZ945:

```
> topTab_BLZvscombo <- topTable (fit.main, number=nrow(fit.main),
coef="BLZvscombo", adjust="fdr")
> head(topTab_BLZvscombo)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
17450501	-3.603563	5.316604	-5.537609	0.0000561	0.1489982	1.6378802
17470976	-1.714192	5.306210	-4.941946	0.0001752	0.1489982	0.7530925
17483615	-2.267989	6.015220	-4.722603	0.0002694	0.1489982	0.4122470
17312637	-1.540237	3.503515	-4.719460	0.0002711	0.1489982	0.4073083
17527661	-1.742880	5.410426	-4.692495	0.0002859	0.1489982	0.3648704
17491193	-2.167735	5.866347	-4.690983	0.0002868	0.1489982	0.3624870

Para la comparación 2 (ctrlvsPD1_PDL1): genes que varían sus niveles de expresión con la adición del anticuerpo frente a PD1/PDL1 en ausencia de BLZ945:

```
> topTab_ctrlvsPD1_PDL1 <- topTable (fit.main, number=nrow(fit.main),
coef="ctrlvsPD1_PDL1", adjust="fdr")
> head(topTab_ctrlvsPD1_PDL1)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
17527666	0.9146316	3.892206	4.287842	0.0006418	0.9002548	-4.183626
17316373	0.7486733	2.646915	4.005748	0.0011373	0.9002548	-4.213102
17454692	-1.0463449	4.583908	-3.878144	0.0014758	0.9002548	-4.227148
17364176	1.5465575	4.627353	3.748010	0.0019265	0.9002548	-4.241928
17430833	0.9342524	7.701533	3.661421	0.0023011	0.9002548	-4.252016
17449725	-1.8057120	5.092797	-3.646117	0.0023746	0.9002548	-4.253820

Para la comparación 3 (INT): genes que se comportan de manera diferente entre la comparación 1 y 2:

```
> topTab_INT  <- topTable (fit.main, number=nrow(fit.main), coef="INT",
adjust="fdr")
> head(topTab_INT)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
17442526	1.250823	5.343071	4.044612	0.0010508	0.5901391	-4.408747
17527661	-1.963809	5.410426	-3.879833	0.0014707	0.5901391	-4.417320
17367510	2.028738	8.573531	3.795276	0.0017486	0.5901391	-4.421860
17331438	-1.307566	6.338055	-3.783065	0.0017929	0.5901391	-4.422524
17410435	1.328570	5.984329	3.723373	0.0020264	0.5901391	-4.425796
17448595	-1.928065	4.938469	-3.658681	0.0023141	0.5901391	-4.429395

Como podemos observar, la primera columna de cada tabla contiene la identificación del fabricante (Affymetrix) para cada conjunto de sondas. El siguiente paso consiste en identificar a qué gen corresponde a cada ID de Affymetrix. Este proceso se llama anotación.

3.3.10. Anotación de genes y visualización de la expresión diferencial:

En la anotación se busca información para asociar los identificadores que hemos obtenido en la tabla anterior, que son los que nos da el array, con el símbolo del gen, el identificador ENTREZ o la descripción del gen.

```
> 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)
+ }
```

Después, guardamos esta información en archivos .csv.

```
> topAnnotated_BLZvscombo <- annotatedTopTable(topTab_BLZvscombo,
+ anotPackage="mogene21sttranscriptcluster.db")
> topAnnotated_ctrlvsPD1_PDL1 <- annotatedTopTable(topTab_ctrlvsPD1_PDL1,
+ anotPackage="mogene21sttranscriptcluster.db")
> topAnnotated_INT <- annotatedTopTable(topTab_INT,
+ anotPackage="mogene21sttranscriptcluster.db")
> write.csv(topAnnotated_BLZvscombo,
file="C:/Users/monic/Desktop/ADO/results/topAnnotated_BLZvscombo.csv")
> write.csv(topAnnotated_ctrlvsPD1_PDL1,
file="C:/Users/monic/Desktop/ADO/results/topAnnotated__ctrlvsPD1_PDL1.csv")
> write.csv(topAnnotated_INT,
file="C:/Users/monic/Desktop/ADO/results/topAnnotated_INT.csv")
```

La anotación hace que las tablas sean más comprensibles. En la tabla que vemos a continuación se muestran las anotaciones agregadas a los resultados “topTable” para la comparación “BLZvscombo”.

	PROBEID <chr>	SYMBOL <chr>	ENTREZID <chr>	GENENAME <chr>
1	17210904	Oprk1	18387	opioid receptor, kappa 1
2	17210953	St18	240690	suppression of tumorigenicity 18
3	17210984	Pcmtd1	319263	protein-L-isoaspartate (D-aspartate) O-methyltransferase domain containing 1
4	17211004	Adhfe1	76187	alcohol dehydrogenase, iron containing, 1
5	17211043	Sgk3	170755	serum/glucocorticoid regulated kinase 3

5 rows

Para visualizar la expresión diferencial global podemos utilizar gráficos de "volcanes", donde podemos ver si hay muchos o pocos genes expresados diferencialmente. En el eje X se representan los cambios en los niveles de expresión mientras en el eje Y se representa el logaritmo negativo en base 10 del p-valor.

La gráfica muestra un diagrama de volcán para la comparación entre la adición o no del anticuerpo frente a PD1/PDL1 en presencia de BLZ. Los nombres de los 4 genes principales (es decir, los primeros cuatro genes en la tabla superior) se muestran en la gráfica.

```
> library(mogene21sttranscriptcluster.db)
> geneSymbols <- select(mogene21sttranscriptcluster.db, rownames(fit.main),
c("SYMBOL"))
> SYMBOLS<- geneSymbols$SYMBOL
> volcanoplot(fit.main, coef=1, highlight=4, names=SYMBOLS,
+             main=paste("Differentially expressed genes",
colnames(cont.matrix)[1], sep="\n"))
>   abline(v=c(-1,1))
```

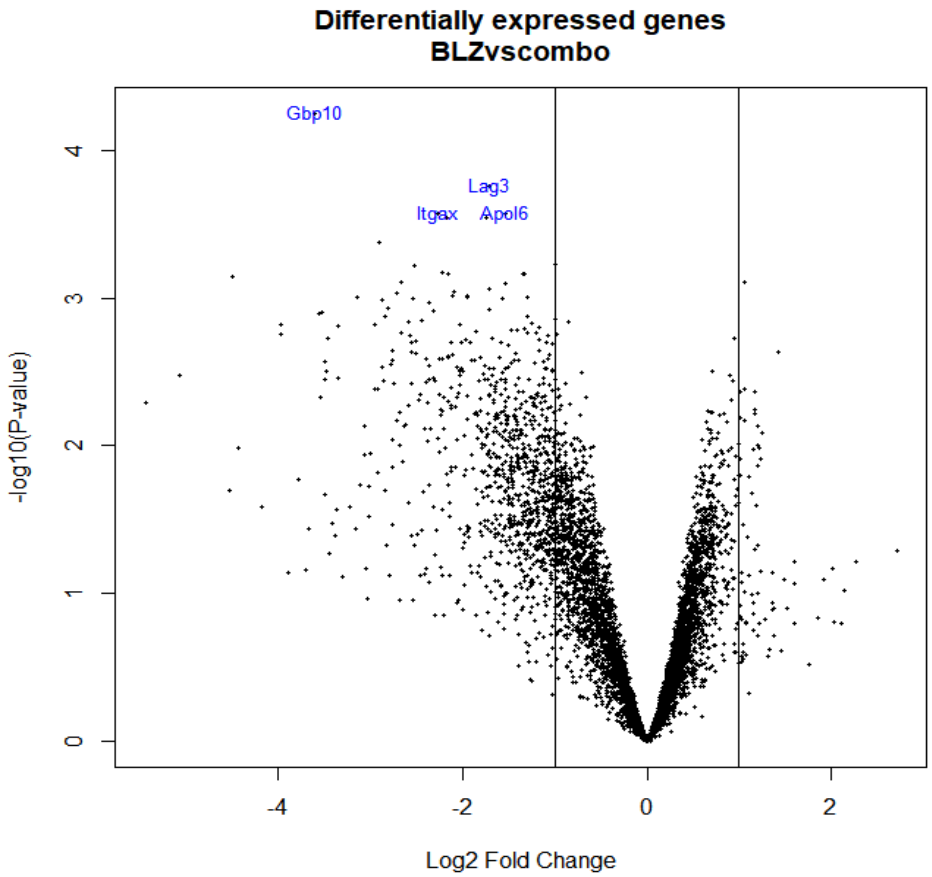

Gráfico 7: Diagrama de volcán para la comparación entre la adición o no del anticuerpo frente a PD1/PDL1 en presencia de BLZ945.

3.3.11. Comparación entre distintas comparaciones:

En este punto del estudio es interesante conocer si algunos genes se han seleccionado como diferencialmente expresados en más de una comparación. A veces, los genes biológicamente relevantes serán aquellos que se seleccionan en una de ellas. En otros casos, serán más interesantes aquellos que se seleccionan en todas las comparaciones.

Utilizamos las funciones `decideTests` y `VennDiagram` del paquete `limma` para anotar y contar los genes seleccionados en cada comparación.

En este caso, no hemos obtenido ningún gen diferencialmente expresado con los criterios estipulados, por lo que para continuar el estudio he modificado el p-valor, incrementándolo hasta 0.15.

```
> library(limma)
> res<-decideTests(fit.main, method="separate", adjust.method="fdr",
p.value=0.15, lfc=1)

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

       BLZvscombo ctrlvsPD1_PDL1  INT
Down          238              0    0
NotSig       5748           5995 5995
Up              9              0    0
```

Estos resultados se pueden visualizar en un diagrama de Venn, que representa el número de genes que se han denominado expresados diferencialmente en cada comparación con un límite dado. La figura muestra cuántos de estos genes son compartidos por una o más selecciones.

```
> vennDiagram (res.selected[,1:3], cex=0.9)
> title("Genes in common between the three comparisons\n Genes selected with
FDR < 0.15 and logFC > 1")
```

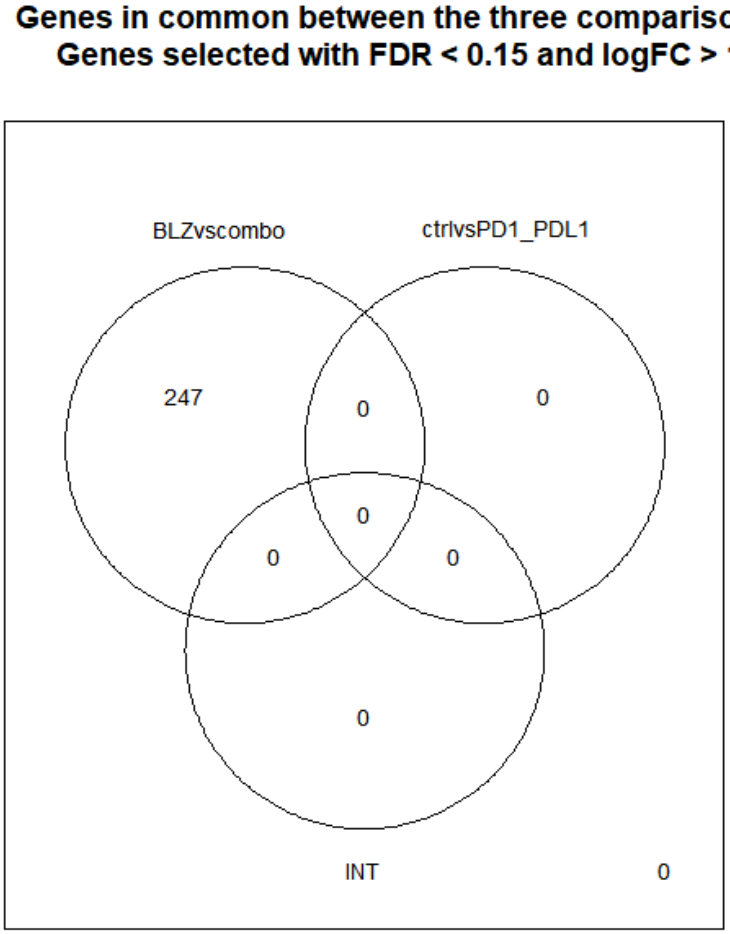

Gráfico 8: Diagrama de Venn que muestra cuántos genes son compartidos por una o más selecciones.

Al igual que hemos observado en la tabla, solo hay genes diferencialmente expresados en una comparación.

3.3.12. Mapas de calor:

Los mapas de calor sirven para visualizar los valores de expresión de genes expresados diferencialmente. En las columnas están representadas las muestras y, en las filas, los distintos genes. Estos genes se pueden agrupar jerárquicamente para observar variaciones de la expresión en grupos de genes relacionados entre sí.

```
> probesInHeatmap <- rownames(res.selected)
> HMdata <- exprs(eset_filtered)[rownames(exprs(eset_filtered)) %in%
probesInHeatmap,]
>
> geneSymbols <- select(mogene21sttranscriptcluster.db, rownames(HMdata),
c("SYMBOL"))
> SYMBOLS<- geneSymbols$SYMBOL
> rownames(HMdata) <- SYMBOLS
```

Con los datos seleccionados se puede generar un mapa de calor con o sin agrupamiento de genes y/o muestras.

```
> my_palette <- colorRampPalette(c("blue", "red"))(n = 299)
> library(gplots)
>
> heatmap.2(HMdata,
+           Rowv = FALSE,
+           Colv = FALSE,
+           main = "Differentially expressed genes \n FDR < 0,15, 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("red",4),rep("blue",3), rep("green",4),
rep("yellow",4)),
+           tracecol = NULL,
+           dendrogram = "none",
+           srtCol = 30)
```

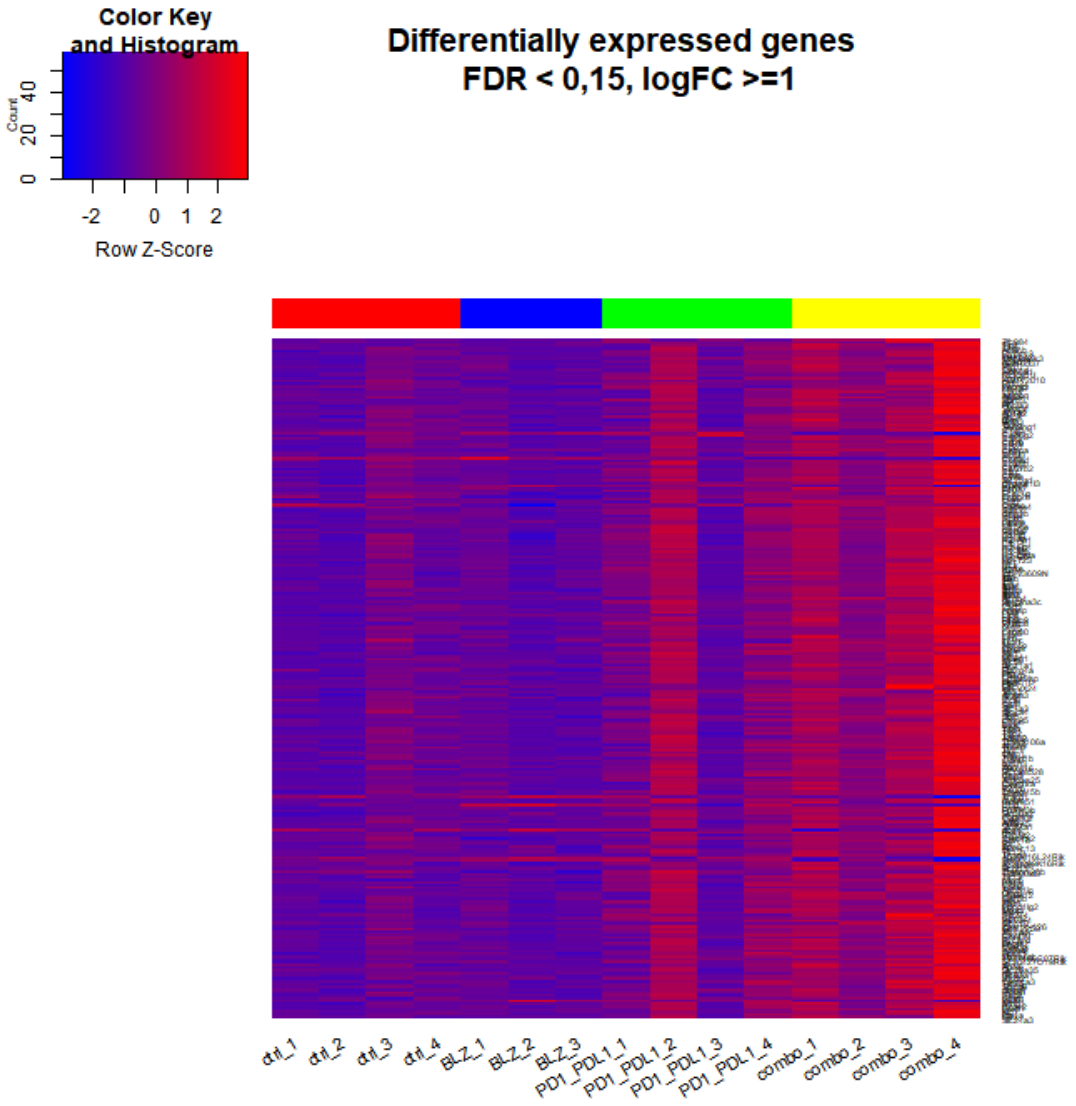

Gráfico 9: Mapa de calor para los datos de expresión sin agrupaciones de muestras y/o genes.

Este mapa de calor incluye todos los genes seleccionados con los mismos criterios descritos anteriormente (FDR <0.15 y logFC> 1) donde no se realiza la agrupación de genes y muestras.

```
> heatmap.2(HMdata,
+           Rowv = TRUE,
+           Colv = TRUE,
+           dendrogram = "both",
+           main = "Differentially expressed genes \n FDR < 0,15, 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("red",4),rep("blue",3), rep("green",4),
rep("yellow",4)),
+            tracecol = NULL,
+            srtCol = 30)
```

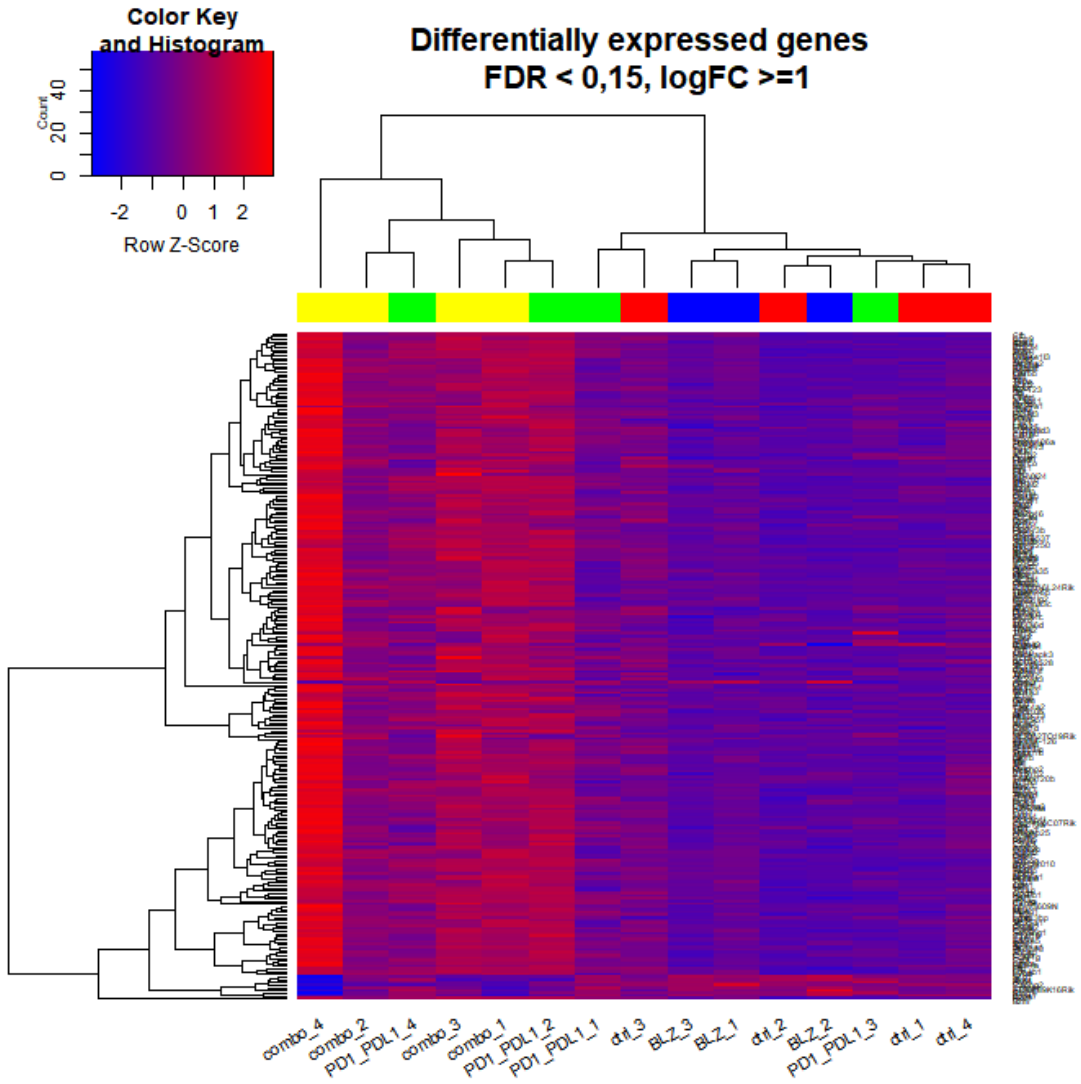

Gráfico 10: Mapa de calor para los datos de expresión con agrupaciones por genes (en las filas) y por muestras (en las columnas) por su similaridad.

En este otro mapa de calor, los genes y las muestras se han agrupado, lo que nos da una visión mucho más clara de los resultados obtenidos, ya que hay grupos de genes que tienen una variación conjunta de sus niveles de expresión.

3.3.13. Significación biológica de los resultados obtenidos:

Una vez que hemos obtenido la lista de genes que están diferencialmente expresados en las distintas condiciones del estudio, debemos interpretar los resultados. Para ello, realizamos un análisis de enriquecimiento básico con el paquete `ReactomePA` que nos puede enfocar la interpretación.

En este análisis, buscamos si hay rutas moleculares, funciones, procesos biológicos, etc. que relacionen los genes que estén diferencialmente expresados.

Para que esta parte del estudio nos dé resultados fiables, necesitamos incluir más genes. Por ello, se recomienda poner como límite de FDR <0.15. En este caso ya lo habíamos hecho en pasos anteriores, por lo que continuamos con los mismos criterios.

En el primer paso, preparamos la lista de listas de genes que se analizarán:

```
> listOfTables <- list(BLZvscombo = topTab_BLZvscombo,
+                      ctrlvsPD1_PDL1  = topTab_ctrlvsPD1_PDL1,
+                      INT = topTab_INT)
> 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(mogene21sttranscriptcluster.db, selectedIDs,
c("ENTREZID"))
+   EntrezIDs <- EntrezIDs$ENTREZID
+   listOfSelected[[i]] <- EntrezIDs
+   names(listOfSelected)[i] <- names(listOfTables)[i]
+ }
> sapply(listOfSelected, length)

    BLZvscombo ctrlvsPD1_PDL1            INT
           275              0              0
```

Para hacer este análisis necesitamos tener los identificadores de Entrez para todos los genes que tienen al menos una anotación en Gene Ontology.

```
> mapped_genes2GO <- mappedkeys(org.Mm.egGO)
> mapped_genes2KEGG <- mappedkeys(org.Mm.egPATH)
> mapped_genes <- union(mapped_genes2GO , mapped_genes2KEGG)
```

El análisis de significación biológica se aplicará solo a las dos primeras listas.

Los resultados obtenidos en el análisis de significación biológica son:

- Un archivo *.csv* con un resumen de las rutas enriquecidas y las estadísticas asociadas.
- Un diagrama de barras con las mejores rutas enriquecidas. (Gráfico 11).
- Una trama con una red de las vías enriquecidas y la relación entre los genes incluidos (Gráfico 12).

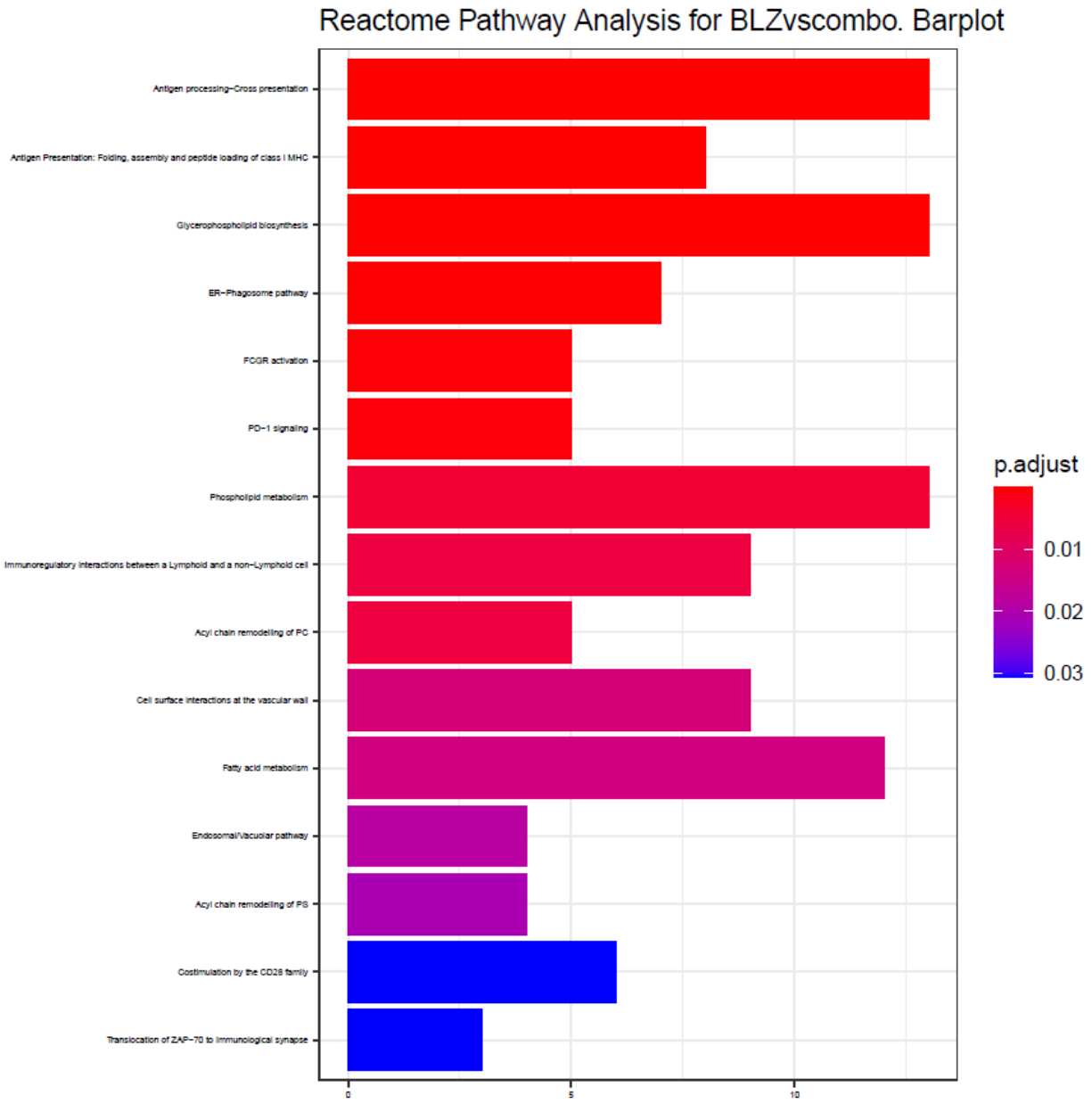

Gráfico 11: Diagrama de barras con las mejores rutas enriquecidas. La altura de cada barra (eje X) es el número de genes del análisis relacionados con esa ruta molecular. Las rutas están ordenadas en el eje Y por su significación estadística.

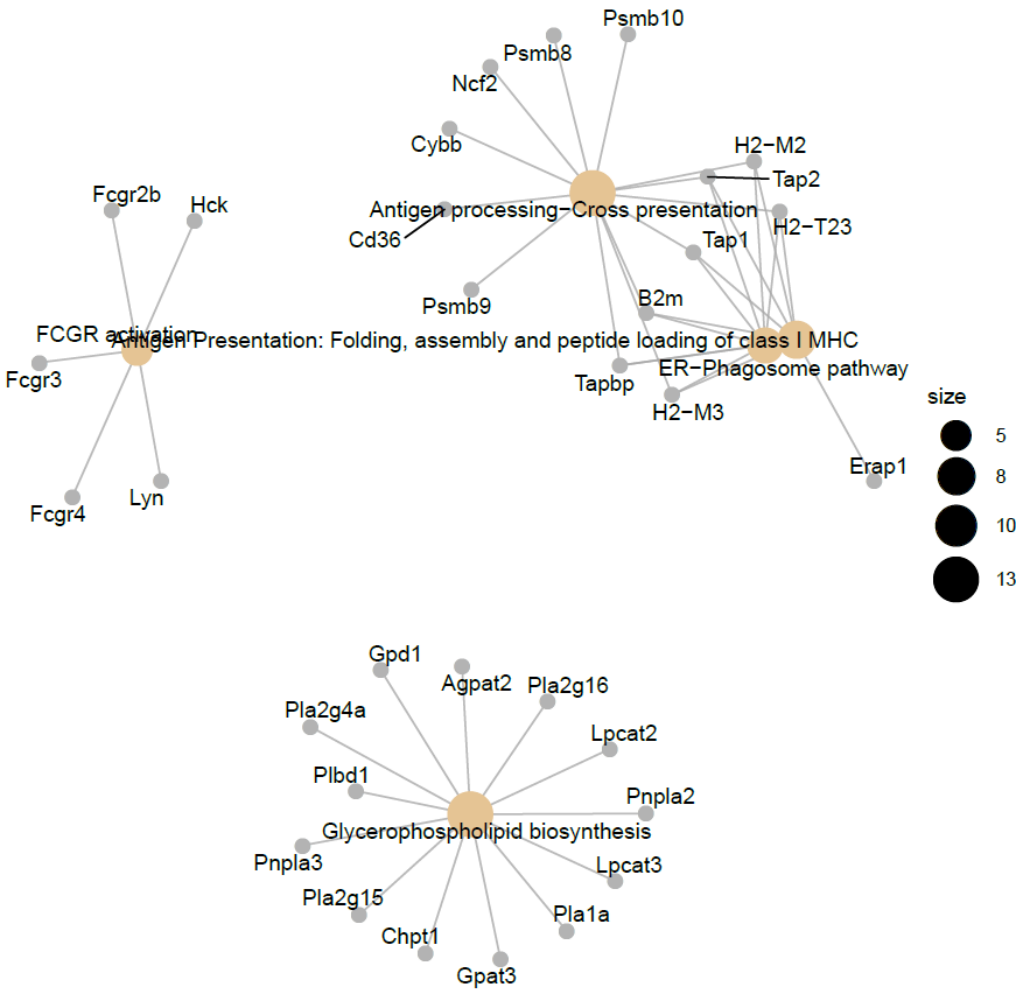

Gráfico 12: Red de las vías enriquecidas y relación entre los genes incluidos.

Tabla 4: Primeras líneas del archivo BLZvscombo.csv con los resultados del análisis con Reactome.

	Description	GeneRatio	BgRatio	pvalue	p.adjust
R-MMU-1236975	Antigen processing-Cross presentation	13/171	86/9065	6.38529533906818e-09	3.00747410470111e-06
R-MMU-983170	Antigen Presentation: Folding, assembly and peptide loading of class I MHC	8/171	36/9065	2.62574791086751e-07	5.2436548095131e-05
R-MMU-1483206	Glycerophospholipid biosynthesis	13/171	119/9065	3.33990752198287e-07	5.2436548095131e-05
R-MMU-1236974	ER-Phagosome pathway	7/171	29/9065	8.26816553108399e-07	9.73576491285139e-05

4. Resultados

En este estudio hemos encontrado que las rutas más relacionadas con los genes diferencialmente expresados son las del procesamiento de antígenos de presentación cruzada, el plegado, ensamblaje y carga de péptidos MHC-I en la presentación de antígenos, la biosíntesis de glicerofosfolípidos y los fagosomas a través del RE. Sin embargo, nos hemos visto obligados a modificar el p-valor ajustado (FDR) a <0.15 para obtener genes que se consideren diferencialmente expresados.

Los arrays utilizados en este estudio tenían una calidad aceptable según los controles realizados, excepto uno que tuvo que ser eliminado. Por tanto, podemos considerar que la calidad de los arrays no ha sido una fuente de error, ya que la normalización se realizó sin problemas.

5. Discusión y conclusiones

En el estudio que se presenta en este documento vemos que las rutas a las que pertenecen los genes diferencialmente expresados están relacionadas con el sistema inmune. Por tanto, los datos sí reflejarían la inhibición de estas rutas mediada por BLZ y el anticuerpo frente a PD1/PDL1. Además, la gran mayoría de los genes diferencialmente expresados están subexpresados (238 frente a 9 sobreexpresados) por el efecto de la inhibición.

Además, solamente encontramos genes diferencialmente expresados (incluso incrementando el FDR) en la comparación entre el grupo BLZ y el grupo combo, que es en el que se emplean ambos inhibidores.

Por tanto, los resultados de nuestro estudio son similares a los del estudio original (Eissler et al., 2016) en el que se concluyó que la combinación del inhibidor de CSF-1R BLZ945 con el bloqueo mediante anticuerpos de la señalización de PD1/PDL1 conduce a un aumento significativo de las quimiocinas inducidas por IFN-γ CXCL9, 10 y 11 en las células mieloides y no se encontraron resultados relevantes utilizando solamente una de las dos inhibiciones.

6. Apéndice

En el repositorio de GitHub *https://github.com/monicahortal/PEC-1-Analisis-de-Datos-Omicos.git* se pueden encontrar los siguientes documentos relacionados con este estudio:

- hortal_monica_ADO_PEC1.pdf: este mismo archivo.
- hortal_monica_ADO_PEC1.rmd: contiene el código R utilizado para realizar el informe.
- Las carpetas “data” y “results” con los archivos generados en el desarrollo del trabajo.

7. Referencias

Eissler N, Mao Y, Brodin D, Reuterswärd P et al. Regulation of myeloid cells by activated T cells determines the efficacy of PD-1 blockade. Oncoimmunology 2016;5(12):e1232222.