Durana_Aritz_ADO_PEC1

Aritz Durana

8/4/2020

Contents

1 ABSTRACT

Este informe esta basado en un estudio de Kim *et al.* (2019) sobre el efecto de posibles antagonistas del factor de transcripción HNF4α, relacionado con tumores gástricos en humanos. Se ha medido la actividad biológica de este antagonista y sus derivados en diferentes lineas celulares tumorales. Los resultados muestran que el derivado *Para* de BI6105 es el único capaz de inhibir algunas rutas oncogénicas.

Todos los ficheros con los datos y el código utilizados para generar este informe se pueden encontrar en un repositorio de github.[1]

2 OBJETIVOS

El principal objetivo de este trabajo es estudiar los efectos en la señalización antineoplásica de un conocido inhibidor del factor de transcripción HNF4α y de sus derivados.

Para ello se medirá la actividad biológica mediante la eficacia antimitogénica frente a varias lineas celulares de cáncer gástrico. Los genes expresados diferencialmente se seguirán por análisis de las rutas de señalización oncogénicas.

Por otra parte, el objetivo de esta PEC1 es la realización de un análisis de datos de microarray de una manera lo más fiel posible a la realidad. De este modo, es posible aplicar los conocimientos teoricos adquiridos a un caso práctico.

3 MATERIALES Y MÉTODOS

3.1 Software

Para la realización de este informe se ha utilizado R (versión 3.6.2) y el interfaz RStudio. Los paquetes específicos para el análisis de datos de microarrays se han obtenido del proyecto Bioconductor (versión 3.6) que engloba los paquetes para análisis de datos ómicos más habituales.

3.2 Datos

Los datos públicos utilizados para este estudio se han conseguido de la base de datos Gene Expression Omnibus (GEO) con el código GSE114626. Se trata de un experimento de arrays de perfil de expresión en el que se utilizan arrays Human Gene 2.0 ST de Affymetrix (código de plataforma: GPL16686). En concreto son 20 muestras de arrays que se generan a partir de un diseño experimental de dos factores: agentes inhibidores (4 niveles) y líneas celulares (5 niveles).

3.3 Workflow y métodos

El procedimiento general del análisis se puede resumir en los siguientes pasos. En cada paso se resume el tipo de tarea realizado y los métodos utilizados.

1. Preparación del entorno: Creación de directorios de trabajo.
2. Obtención de datos: De la base de datos GEO con `getGEO()`. Creación del fichero *targets.csv*.
3. Lectura de datos: Generación de un objeto ExpressionSet (`rawData`) a partir de los ficheros *.CEL*.
4. Control de calidad de datos crudos: Tanto a nivel numérico (`arrayQualityMetrics()`) como gráfico (PCA, boxplot, cluster).

[1] https://github.com/adurana/ADO_PEC1

5. Normalización de datos: Usando el método RMA para generar un ExpressionSet con datos normalizados (`eset_rma`).
6. Control de calidad de datos normalizados: Análogo a lo realizado con los datos crudos.
7. Detección de genes más variables: De manera gráfica mediante un plot de desviaciones estándar.
8. Filtrado de genes: Filtrado de los genes menos variables (`nsFilter()`) y generación de un ExpressionSet con datos filtrados (‘eset_filtered).
9. Control de calidad de datos filtrados: Análogo a lo realizado con los datos crudos.
10. Diseño experimental: Creación de la matriz de diseño y de la matriz de contraste.
11. Modelización y selección de genes: Ajuste del modelo lineal (`limma()`) y selección de genes.
12. Listado de genes diferencialmente expresados: Obtención de listas de genes (`topTable()`) para cada una de las hipótesis a contrastar.
13. Anotación de genes: Uso de ficheros de anotación para completar los listados de genes.
14. Visualización de la expresión diferencial: Volcano plot.
15. Comparaciones múltiples: Diagramas de Venn y heatmaps.
16. Significación biológica de los resultados: Uso del método *Gene Enrichment Analysis* y del paquete `ReactomePA` para ayudar a interpretar los resultados desde un punto de vista biológico.
17. Resumen de resultados: Listado de los ficheros de resultados generados.

4 RESULTADOS Y DISCUSIÓN

En esta sección se detallarán los resultados obtenidos en el análisis siguiendo el workflow descrito anteriormente. El código utilizado para realizar el análisis no se mostrará y solo se mostrarán los resultados más relevantes. Todo el código empleado para la realización del análisis se podrá encontrar en el apéndice que se encuentra al final de este informe además de en el repositorio de github mencionado anteriormente.[1] Hay que señalar que gran parte del análisis esta basado en uno similar disponible en los materiales de la asignatura.[2]

4.1 Preparación del entorno

Antes de descargar los datos y comenzar con el análisis es recomendable preparar el entorno de trabajo. Tras elegir el directorio de trabajo `workingDir`, se crearon los directorios para los datos `dataDir` y los resultados `resultsDir`.

4.2 Obtención de datos

Como se ha mencionado anteriormente, los datos se obtuvieron de la base de datos de GEO. Para ello se realizó una busqueda para acotar los resultados con ciertos parámetros de interés como tipo de experimento, organismo, fecha de publicación y otros (Figure 1). De entre los resultados de esta búsqueda se eligió el estudio GSE114626 de Kim *et al.* (2019).

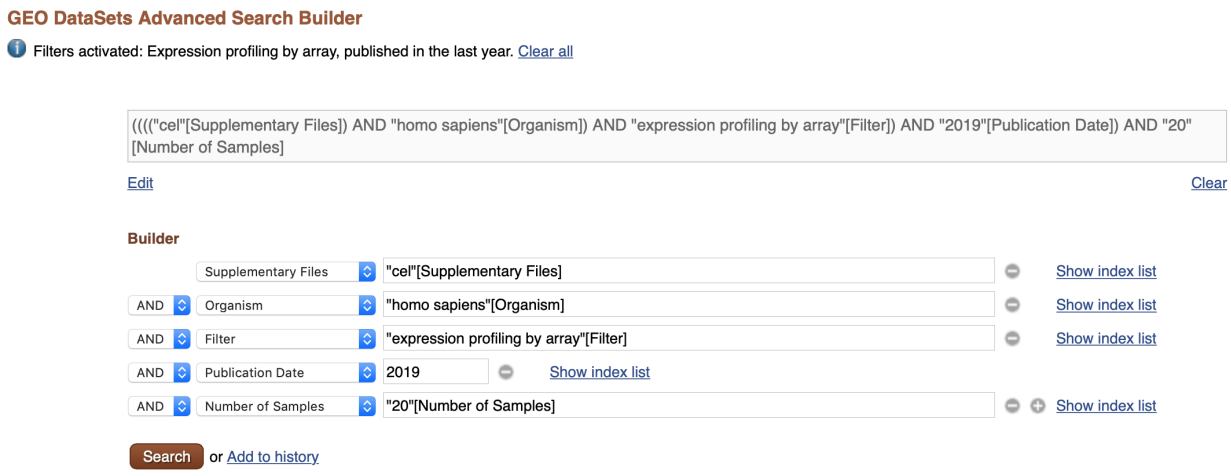

Figure 1: Parámetros de busqueda en GEO

Para la obtención de los datos se utilizó el paquete `GEOquery`, en concreto la función `getGEO()` para descargar los datos del experimento y `getGEOSuppFiles()` para descargar los ficheros *.CEL*.

A pesar de intentar generarlo automáticamente a partir de la información descargada con `getGEO()`, al final opté por crear el fichero *targets.csv* de manera manual.

En el estudio se observa el efecto de 3 agentes inhibidores (Ortho, Meta y Para) y un control (DMSO) sobre la expresión génica en cinco líneas celulares tumorales (SNU601,SNU216, SNU668, AGS y MKN1). Esto da como resultado 20 muestras diferentes que son las que se resumen en el fichero *targets.csv* (Table 1). Las columnas de este fichero son *FileName* para el nombre del fichero *.CEL*, *CellType* para la línea celular, *Agent* para el agente inhibidor y *ShortName* para el código de cada muestra.

[2] https://github.com/ASPteaching/Omics_Data_Analysis-Case_Study_1-Microarrays

Table 1: Contenido del fichero *targets.csv* para el estudio GSE114626

FileName	CellType	Agent	ShortName
GSM3146128_SNU_601_DMSO_HuGene-2_0-st_.CEL	SNU601	DMSO	SNU601_DMSO
GSM3146129_SNU_601_ortho_HuGene-2_0-st_.CEL	SNU601	Ortho	SNU601_Ortho
GSM3146130_SNU_601_para_HuGene-2_0-st_.CEL	SNU601	Para	SNU601_Para
GSM3146131_SNU_601_Meta_HuGene-2_0-st_.CEL	SNU601	Meta	SNU601_Meta
GSM3146132_SNU_216_DMSO_HuGene-2_0-st_.CEL	SNU216	DMSO	SNU216_DMSO
GSM3146133_SNU_216_ortho_HuGene-2_0-st_.CEL	SNU216	Ortho	SNU216_Ortho
GSM3146134_SNU_216_para_HuGene-2_0-st_.CEL	SNU216	Para	SNU216_Para
GSM3146135_SNU_216_Meta_HuGene-2_0-st_.CEL	SNU216	Meta	SNU216_Meta
GSM3146136_SNU668_DMSO_HuGene-2_0-st_.CEL	SNU668	DMSO	SNU668_DMSO
GSM3146137_SNU668_Ortho_HuGene-2_0-st_.CEL	SNU668	Ortho	SNU688_Ortho
GSM3146138_SNU668_Para_HuGene-2_0-st_.CEL	SNU668	Para	SNU688_Para
GSM3146139_SNU668_Meta_HuGene-2_0-st_.CEL	SNU668	Meta	SNU688_Meta
GSM3146140_AGS_DMSO_HuGene-2_0-st_.CEL	AGS	DMSO	AGS_DMSO
GSM3146141_AGS_Ortho_HuGene-2_0-st_.CEL	AGS	Ortho	AGS_Ortho
GSM3146142_AGS_Para_HuGene-2_0-st_.CEL	AGS	Para	AGS_Para
GSM3146143_AGS_Meta_HuGene-2_0-st_.CEL	AGS	Meta	AGS_Meta
GSM3146144_MKN1_DMSO_HuGene-2_0-st_.CEL	MKN1	DMSO	MKN1_DMSO
GSM3146145_MKN1_Ortho_HuGene-2_0-st_.CEL	MKN1	Ortho	MKN1_Ortho
GSM3146146_MKN1_Para_HuGene-2_0-st_.CEL	MKN1	Para	MKN1_Para
GSM3146147_MKN1_Meta_HuGene-2_0-st_.CEL	MKN1	Meta	MKN1_Meta

4.3 Lectura de datos

Para poder trabajar con los datos descargados, es necesario leerlos y guardarlos como un objeto en *R*. Para ello son necesarios los paquetes `Biobase`, `oligo` y el paquete de anotaciones `pd.hugene.2.0.st` debido al array utilizado en este caso. A continuación se leyeron los ficheros *.CEL* con la función `list.celfiles()` y el fichero *targets.csv* con la función `read.AnnotatedDataFrame()`. Estos dos objetos se combinaron para generar el objeto ExpressionSet `rawData` utilizando la función `read.celfiles()`. Tras modificar los nombres de las columas, se obtuvo el objeto `rawData` listo para su análisis.

4.4 Control de calidad de datos crudos

En este paso del análisis se comprobará si los datos crudos tienen calidad aceptable para continuar con el análisis o ha habido algún tipo de problema serio a la hora de recoger los datos o de realizar los experimentos.

Este control de calidad se realizó de manera gráfica pero también de una manera más exhaustiva utilizando el paquete `arrayQualityMetrics`. Este paquete realiza diferentes controles de calidad e indica si alguna de las muestras se puede considerar un *outlier* en alguna categoria (Figure 2).

En este resumen se observa que hay dos muestras (AGS_DMSO y AGS_Ortho) que cuentan con dos outliers cada una. Esto, de momento, no deberia ser problema para conseguir normalizar los datos correctamente pero si surgiera algún problema durante el análisis posterior habría que revisar estas dos muestras.

En cuanto al análisis gráfico de los datos crudos, se decidio realizar un boxplot (Figure 3), un cluster jerárquico (Figure 4) y un análisis de componentes principales (PCA, Figure 5). Para este último plot, se creo una función específica `plotPCA3()` cuyo código se encuentra, como el resto, en el apéndice al final del informe.

En el boxplot se observa que la distribución de la intensidad en las muestras es bastante similar en todos los casos por lo que no parece que haya problemas con los datos.

	array	sampleNames	*1	*2	*3	CellType	Agent	ShortName
☐	1	SNU601_DMSO				SNU601	DMSO	SNU601_DMSO
☐	2	SNU601_Ortho				SNU601	Ortho	SNU601_Ortho
☐	3	SNU601_Para				SNU601	Para	SNU601_Para
☐	4	SNU601_Meta				SNU601	Meta	SNU601_Meta
☐	5	SNU216_DMSO				SNU216	DMSO	SNU216_DMSO
☐	6	SNU216_Ortho				SNU216	Ortho	SNU216_Ortho
☐	7	SNU216_Para				SNU216	Para	SNU216_Para
☐	8	SNU216_Meta				SNU216	Meta	SNU216_Meta
☐	9	SNU668_DMSO				SNU668	DMSO	SNU668_DMSO
☐	10	SNU688_Ortho				SNU668	Ortho	SNU688_Ortho
☐	11	SNU688_Para				SNU668	Para	SNU688_Para
☐	12	SNU688_Meta				SNU668	Meta	SNU688_Meta
☑	13	AGS_DMSO	x	x		AGS	DMSO	AGS_DMSO
☑	14	AGS_Ortho	x	x		AGS	Ortho	AGS_Ortho
☐	15	AGS_Para				AGS	Para	AGS_Para
☐	16	AGS_Meta				AGS	Meta	AGS_Meta
☐	17	MKN1_DMSO				MKN1	DMSO	MKN1_DMSO
☐	18	MKN1_Ortho				MKN1	Ortho	MKN1_Ortho
☐	19	MKN1_Para				MKN1	Para	MKN1_Para
☐	20	MKN1_Meta				MKN1	Meta	MKN1_Meta

Figure 2: Resumen de resultados del fichero index.html para el análisis por arrayQualityMetrics de los datos crudos

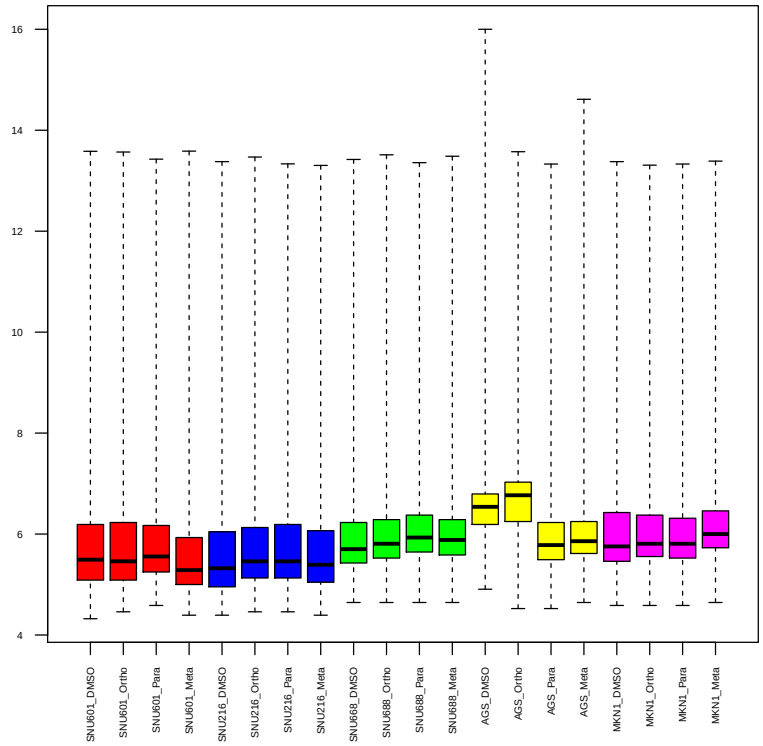

Figure 3: Boxplot de intensidad de arrays: datos crudos

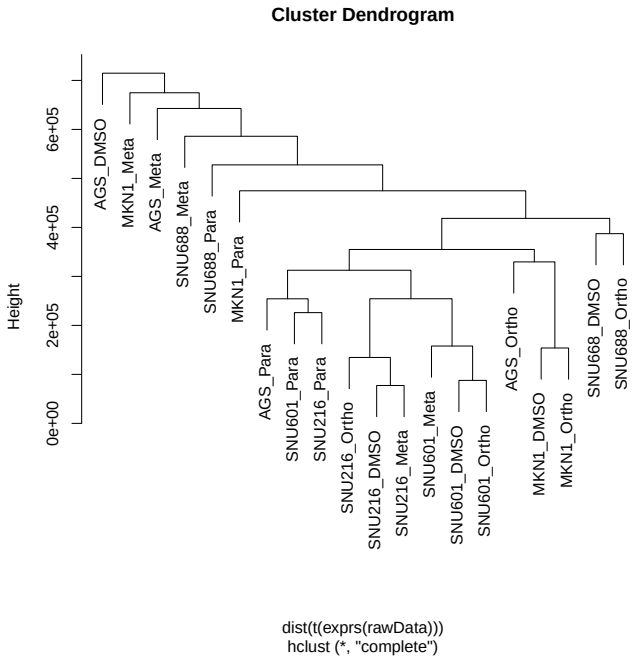

Figure 4: Cluster jerárquico de datos crudos

En cuanto al cluster, se observa que no hay una jerarquia clara entre las muestras. Esto puede ser debido a que estamos tratando con datos crudos.

En el plot de componentes principales (PCA) se observa que los dos primeros componentes explican practicamente el mismo porcentaje de variabilidad con un 16.2% y un 14.9% respectivamente. Esto indica que el factor *Agent* y el factor *CellType* tienen una importancia similar en la variabilidad de los datos. Además son porcentajes bastante bajos, lo que quiere decir que hay muchos más factores que no se contemplan pero que tiene una influencia grande en la variabilidad. Hay que señalar que exceptuando dos muestras, el resto se encuentran muy cerca lo que indica que son muy similares entre si.

Con estos resultados obtenidos del control de calidad en los datos crudos, se decidió que los datos eran lo suficientemente buenos como para poder continuar el análisis.

4.5 Normalización de datos

La etapa de normalización se basa en la necesidad de hacer los arrays más comparables entre sí minimizando la variabilidad de las muestras debida a razones técnicas o de análisis. El proceso de normalización pretende asegurar que las diferencias de intensidades presentes en los arrays sea debido sólo a razones biológicas y que las observaciones sean reflejo de la expresión diferencial de los genes. En este caso se ha utilizado para el análisis el método Robust Multichip Analysis (RMA) desarrollado por Irizarry *et al.* (2003) que es el más habitual en la actualidad.

Para normalizar los datos se creó un nuevo objeto ExpressionSet `eset_rma` con la función `rma()`.

4.6 Control de calidad de datos normalizados

En este punto se realizó un análisis similar al realizado para los datos crudos.

En los resultados de `arrayQualityMetrics()` se observó una mejora de los indicadores de calidad con respecto a los datos crudos dado que ya no hay ninguna muestra con dos outliers (Figure 6).

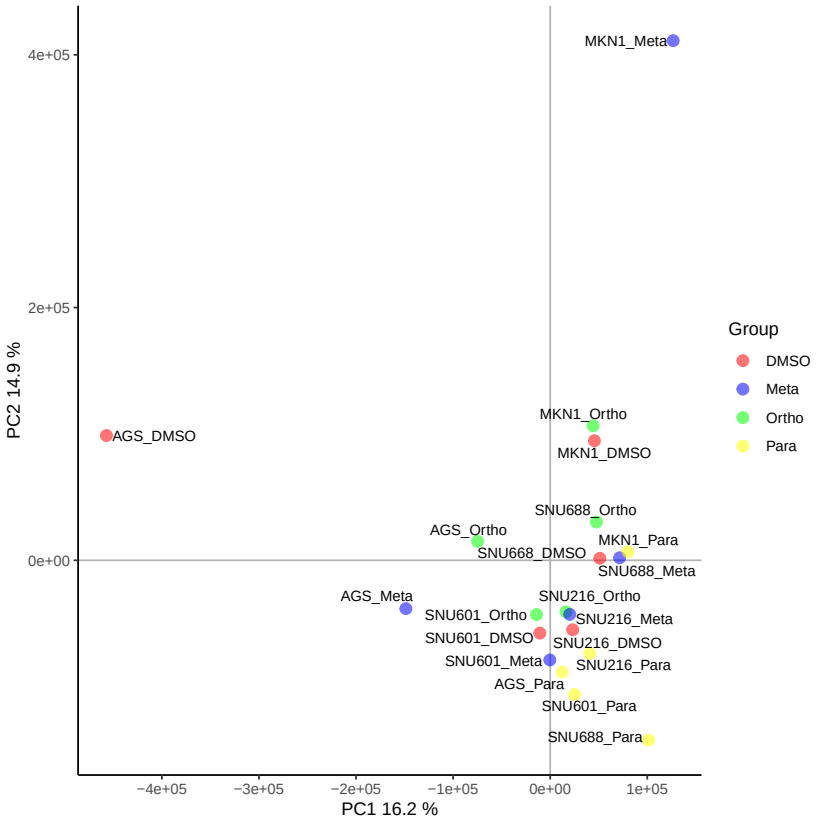

Figure 5: Análisis de componentes principales de datos crudos

	array	sampleNames	*1	*2	*3	CellType	Agent	ShortName
☐	1	SNU601_DMSO				SNU601	DMSO	SNU601_DMSO
☐	2	SNU601_Ortho				SNU601	Ortho	SNU601_Ortho
☐	3	SNU601_Para				SNU601	Para	SNU601_Para
☐	4	SNU601_Meta				SNU601	Meta	SNU601_Meta
☐	5	SNU216_DMSO				SNU216	DMSO	SNU216_DMSO
☐	6	SNU216_Ortho				SNU216	Ortho	SNU216_Ortho
☐	7	SNU216_Para				SNU216	Para	SNU216_Para
☐	8	SNU216_Meta				SNU216	Meta	SNU216_Meta
☐	9	SNU668_DMSO				SNU668	DMSO	SNU668_DMSO
☐	10	SNU688_Ortho				SNU668	Ortho	SNU688_Ortho
☐	11	SNU688_Para				SNU668	Para	SNU688_Para
☐	12	SNU688_Meta				SNU668	Meta	SNU688_Meta
☐	13	AGS_DMSO				AGS	DMSO	AGS_DMSO
☐	14	AGS_Ortho				AGS	Ortho	AGS_Ortho
☑	15	AGS_Para		x		AGS	Para	AGS_Para
☐	16	AGS_Meta				AGS	Meta	AGS_Meta
☐	17	MKN1_DMSO				MKN1	DMSO	MKN1_DMSO
☐	18	MKN1_Ortho				MKN1	Ortho	MKN1_Ortho
☑	19	MKN1_Para	x			MKN1	Para	MKN1_Para
☑	20	MKN1_Meta		x		MKN1	Meta	MKN1_Meta

Figure 6: Resumen de resultados del fichero index.html para el análisis por arrayQualityMetrics de los datos normalizados

Se realizó el mismo análisis gráfico que para los datos crudos. En el boxplot (Figure 7) se observa que la distribución de la intensidad en las muestras ha mejorado con respecto a los datos crudos, lo que es indicativo de que la normalización ha funcionado correctamente.

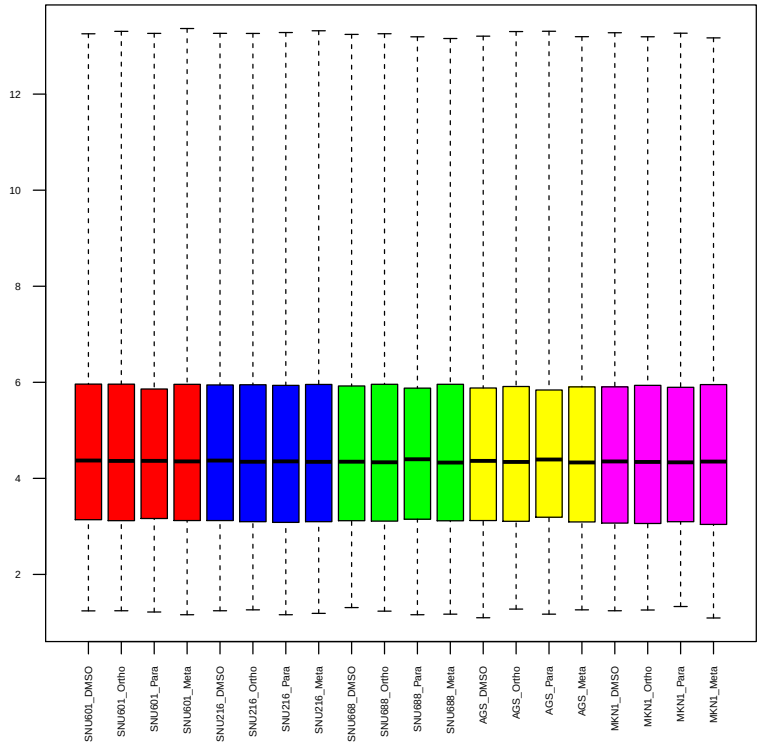

Figure 7: Boxplot de intensidad de arrays: datos normalizados

En cuanto al cluster (Figure 8), se observa que las muestras se encuentran más organizadas que en caso de los datos crudos. Parece que hay cierta jerarquización en torno a las muestras que utilizan el agente *Para*.

En el plot de componentes principales (Figure 9) se observa que las muestras *Ortho*, *Meta* y *DMSO* para cada una de las líneas celulares se encuentran agrupadas mientras que las muestras *Para* están muy alejadas del resto de las muestras para cada una de la líneas celulares. Esto es indicativo del comportamiento singular de las muestras que utilizan el agente *Para*. Además hay que señalar que los porcentajes de variabilidad han aumentado ligeramente pero que prácticamente no hay diferencia con los observados para los datos crudos.

4.7 Detección de genes más variables

Es esperable que los genes diferencialmente expresados muestren ciertas diferencias entre muestras por lo que la varianza de estos genes debería ser superior a la de los genes que no expresan diferencialmente. Una manera sencilla de observar esta diferencia es la representación de la desviación estándar de todos los genes (Figure 10) de modo que se pueda decidir a partir de que porcentaje de desviación estándar un grupo de genes se puede considerar como diferencialmente expresado. Las líneas verticales del gráfico representan los percentiles 90% y 95%.

4.8 Filtrado de genes

El filtrado de genes con poca variabilidad o con variabilidad atribuible a efectos aleatorios pueden ser filtrados de la población de genes total para un aumento de la potencia del análisis (Hackstadt 2009). Para realizar esto se utiliza la función `nsFilter()` del paquete `genefilter` que permite filtrar genes utilizando un umbral

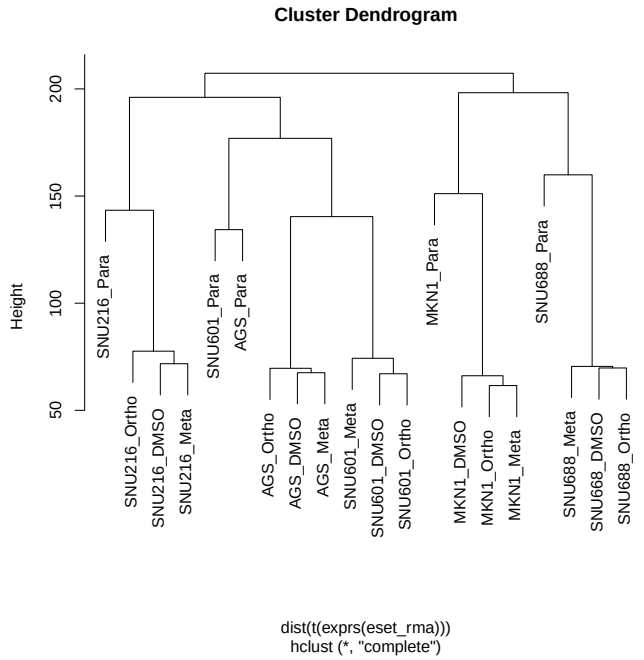

Figure 8: Cluster jerárquico de datos normalizados

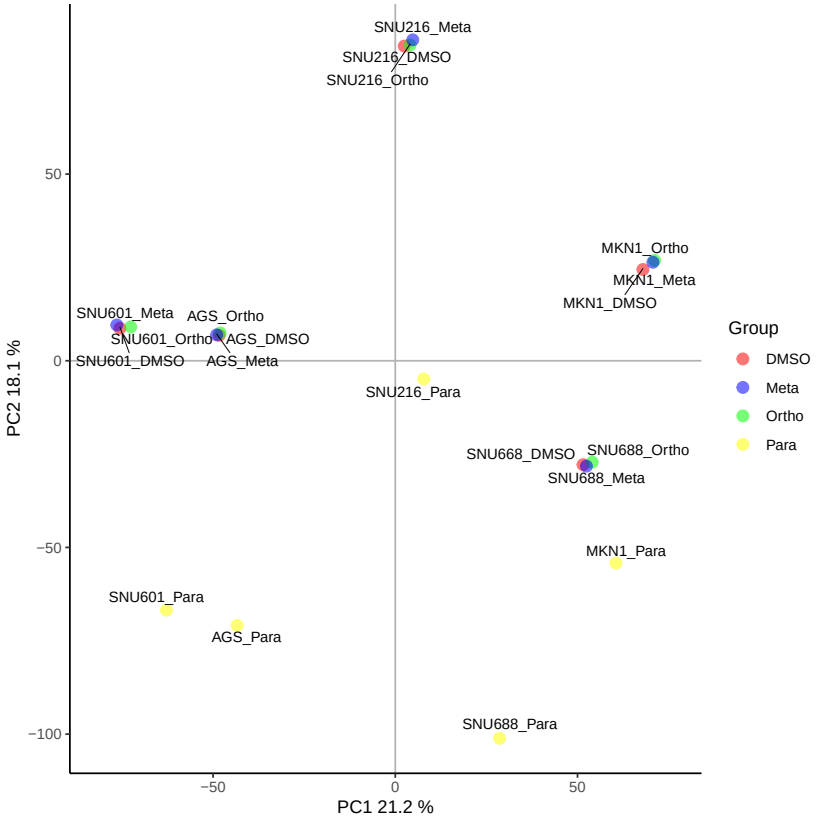

Figure 9: Análisis de componentes principales de datos normalizados

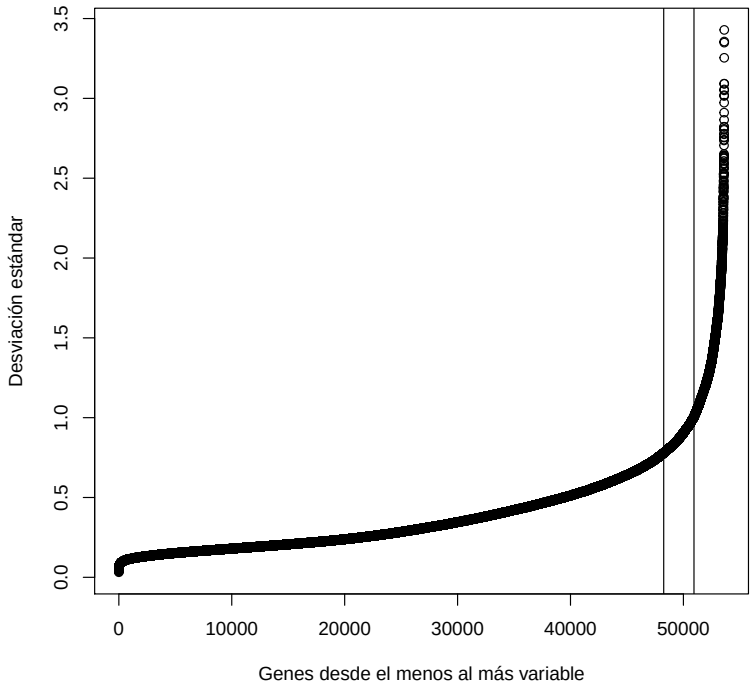

Figure 10: Distribución de la variabilidad de todos los genes

de variabilidad. Además el uso de un paquete de anotaciones, como `hugene20sttranscriptcluster.db` para este caso, permite desechar los genes sin un identificador asociado. El objeto ExpressionSet creado se ha guardado como `eset_filtered`.

4.9 Control de calidad de datos filtrados

De modo idéntico al caso de los datos crudos y normalizados, se realizó un análisis de los datos filtrados.

En los resultados de `arrayQualityMetrics()` (Figure 11) se observó una mejora de los indicadores de calidad dado que no hay ninguna muestra con outliers.

En el boxplot de los datos filtrados (Figure 12) no se observan cambios significativos.

En cuanto al cluster jerárquico (Figure 13), se observa una jeraquización mucho más elevada que en el caso de los datos normalizados. En concreto, y como parecian apuntar los datos normalizados, todas las muestras con agente *Para* se encuentran en un cluster mientras que el resto de las muestras se organizan por líneas celulares.

En el plot de componentes principales de datos filtrados (Figure 14) se observa un comportamiento muy similar al observado en el cluster: las muestras *Para* se encuentran separadas del resto de las muestras que se agrupan por líneas celulares.

Antes de continuar con el análisis y debido a que los datos filtrados son el punto de partida de diferentes análisis, suele ser conveniente guardarlos por si hubiera que volver a ellos. Para ello se crean los ficheros *normalized.Data.csv* para datos normalizados, *filtered.Data.csv* para datos filtrados y *normalized.filtered.Data.Rda* para el conjunto de los datos normalizados y filtrados como un objeto *R*.

	array	sampleNames	*1	*2	*3	CellType	Agent	ShortName
☐	1	SNU601_DMSO				SNU601	DMSO	SNU601_DMSO
☐	2	SNU601_Ortho				SNU601	Ortho	SNU601_Ortho
☐	3	SNU601_Para				SNU601	Para	SNU601_Para
☐	4	SNU601_Meta				SNU601	Meta	SNU601_Meta
☐	5	SNU216_DMSO				SNU216	DMSO	SNU216_DMSO
☐	6	SNU216_Ortho				SNU216	Ortho	SNU216_Ortho
☐	7	SNU216_Para				SNU216	Para	SNU216_Para
☐	8	SNU216_Meta				SNU216	Meta	SNU216_Meta
☐	9	SNU668_DMSO				SNU668	DMSO	SNU668_DMSO
☐	10	SNU688_Ortho				SNU668	Ortho	SNU688_Ortho
☐	11	SNU688_Para				SNU668	Para	SNU688_Para
☐	12	SNU688_Meta				SNU668	Meta	SNU688_Meta
☐	13	AGS_DMSO				AGS	DMSO	AGS_DMSO
☐	14	AGS_Ortho				AGS	Ortho	AGS_Ortho
☐	15	AGS_Para				AGS	Para	AGS_Para
☐	16	AGS_Meta				AGS	Meta	AGS_Meta
☐	17	MKN1_DMSO				MKN1	DMSO	MKN1_DMSO
☐	18	MKN1_Ortho				MKN1	Ortho	MKN1_Ortho
☐	19	MKN1_Para				MKN1	Para	MKN1_Para
☐	20	MKN1_Meta				MKN1	Meta	MKN1_Meta

Figure 11: Resumen de resultados del fichero index.html para el análisis por arrayQualityMetrics de los datos fitrados

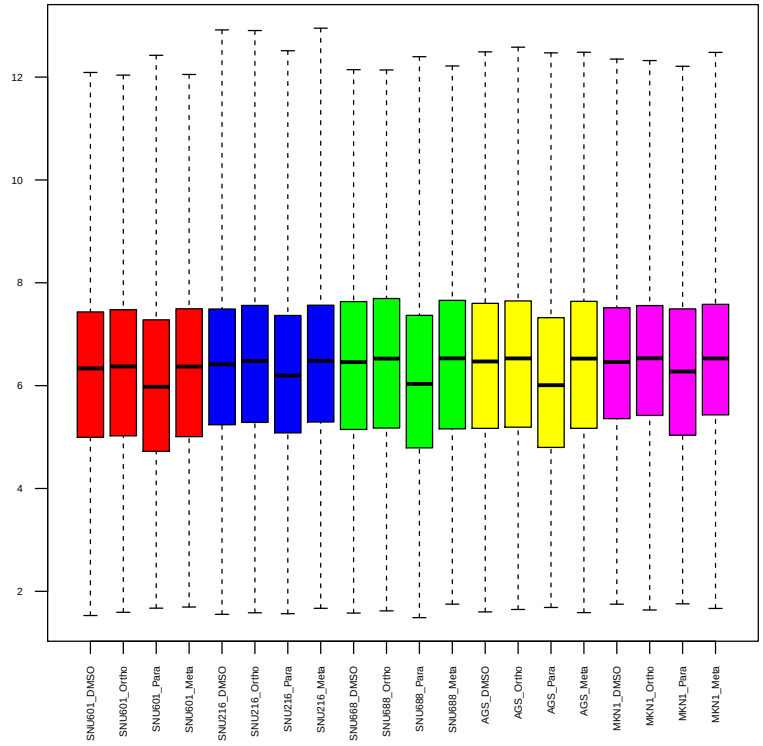

Figure 12: Boxplot de intensidad de arrays: datos filtrados

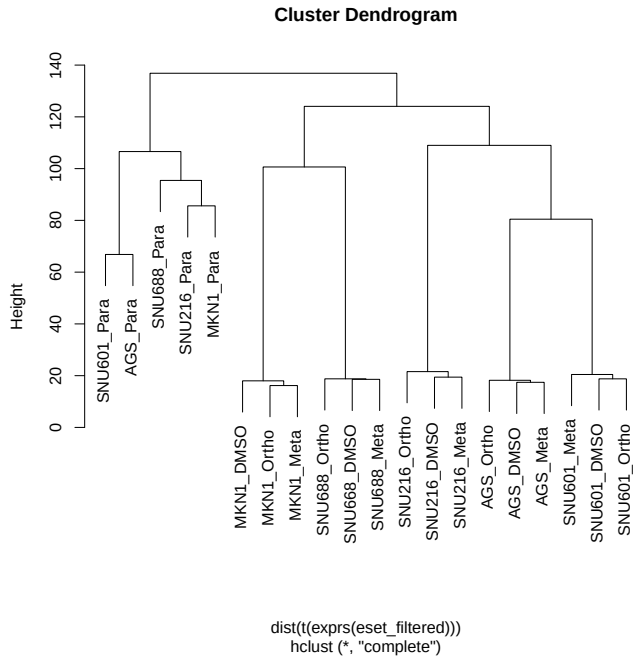

Figure 13: Cluster jerárquico de datos filtrados

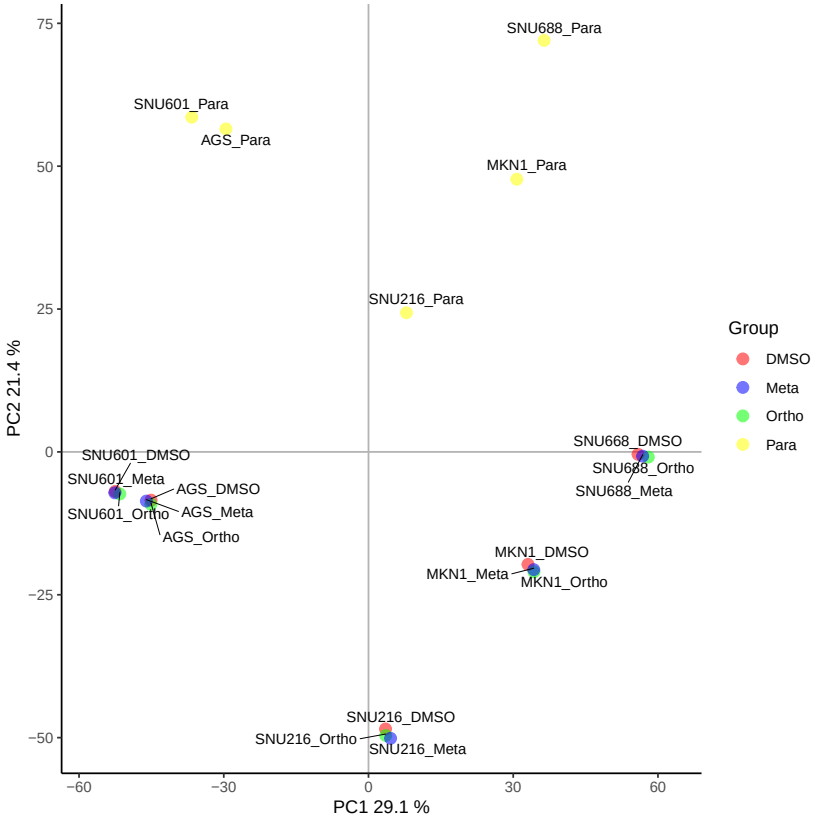

Figure 14: Análisis de componentes principales de datos filtrados

4.10 Diseño experimental

Seleccionar genes diferencialmente expresados consiste básicamente en realizar algún tipo de test o de comparación estadística entre diferentes grupos. Existes numerosos métodos que permiten hacerlo pero el más utilizado es el método de modelos lineales para microarrays implementado en el paquete `limma` desarrollado por Smyth (2004).

El primer paso de este método es crear la matriz de diseño (`designMat`), que simplemente describe las condiciones experimentales aplicables a cada muestra. Un valor 1 significa que la muestra (filas) pertenece a un grupo concreto (columnas), mientras que el valor 0 significa que no pertenece a ese grupo. En este caso se ha generado la matriz agrupando las muestras por agente inhibidor y se ha almacenado en la variable `designMat`.

```
> designMat
```

```
             DMSO Meta Ortho Para
SNU601_DMSO     1    0     0    0
SNU601_Ortho    0    0     1    0
SNU601_Para     0    0     0    1
SNU601_Meta     0    1     0    0
SNU216_DMSO     1    0     0    0
SNU216_Ortho    0    0     1    0
SNU216_Para     0    0     0    1
SNU216_Meta     0    1     0    0
SNU668_DMSO     1    0     0    0
SNU688_Ortho    0    0     1    0
SNU688_Para     0    0     0    1
SNU688_Meta     0    1     0    0
AGS_DMSO        1    0     0    0
AGS_Ortho       0    0     1    0
AGS_Para        0    0     0    1
AGS_Meta        0    1     0    0
MKN1_DMSO       1    0     0    0
MKN1_Ortho      0    0     1    0
MKN1_Para       0    0     0    1
MKN1_Meta       0    1     0    0
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$Agent
[1] "contr.treatment"
```

El segundo paso es crear la matriz de contrastes. Esta matriz se usa para definir los contrastes entre diferentes muestras sobre los que se quiere realizar una comparación. En el fondo, no es otra cosa que definir las hipótesis nulas que interese estudiar. En este caso, y debido a que aparentemente el agente *Para* tiene un comportamiento diferente, los contrastes a estudiar serían los siguentes:

- Para - Ortho = *PvsO*
- Para - Meta = *PvsM*
- Para - DMSO = *PvsD*

Estos contrastes se han almacenado en la variable `cont.matrix`.

```
> cont.matrix
```

```
       Contrasts
Levels  PvsO PvsM PvsD
  DMSO     0    0   -1
  Meta     0   -1    0
  Ortho   -1    0    0
  Para     1    1    1
```

4.11 Modelización y selección de genes

Para ajustar el modelo se utilizó la función `lmFit()` del paquete `limma` utilizando el ExpressionSet de datos filtrados y las dos matrices del diseño. Además también se añaden al análisis modelos empiricos bayesianos que mejoran la estimación de los errores (Smyth 2004). Este método proporciona p-valores ajustados que permiten controlar el porcentaje de falsos positivos al usar el método de Benjamini y Hochberg (Benjamini 1995). Todo este análisis se almacena en el objeto `fit.main`.

4.12 Listado de genes diferencialmente expresados

La función `topTable()` implementada en el paquete `limma` devuelve una lista de genes en orden ascendente de p-valores para cada contraste, por lo que se puede considerar que los primeros genes de esta lista son los más diferencialmente expresados. Se muestran las primeras filas de los listados `topTab_PvsO` para el contraste *PvsO*, `topTab_PvsM` para el contraste *PvsM* y `topTab_PvsD` para el contraste *PvsD*.

```
> head(topTab_PvsO)
```

```
             logFC  AveExpr         t      P.Value    adj.P.Val        B
16877019 -3.276193 7.109180 -13.45278 2.390140e-11 1.572712e-07 15.93631
16705159 -4.248789 6.708237 -12.85607 5.330811e-11 1.753837e-07 15.20340
17061662 -3.857884 7.933510 -12.19448 1.344171e-10 2.948215e-07 14.34978
16702571 -3.145559 6.514993 -11.68238 2.826225e-10 4.649140e-07 13.65767
16799793 -3.591773 8.113105 -11.41139 4.230537e-10 5.567386e-07 13.27984
16904780 -3.108524 5.638834 -11.25859 5.327973e-10 5.843011e-07 13.06315
```

```
> head(topTab_PvsM)
```

```
             logFC  AveExpr         t      P.Value    adj.P.Val        B
16877019 -3.268322 7.109180 -13.42046 2.494415e-11 1.641325e-07 15.89210
16705159 -4.156472 6.708237 -12.57673 7.840500e-11 2.579524e-07 14.84401
17061662 -3.830971 7.933510 -12.10941 1.518220e-10 3.329961e-07 14.23290
16702571 -3.145093 6.514993 -11.68065 2.833452e-10 4.510728e-07 13.65192
16904780 -3.189583 5.638834 -11.55218 3.427605e-10 4.510728e-07 13.47393
16799793 -3.548159 8.113105 -11.27282 5.214232e-10 5.718274e-07 13.08053
```

```
> head(topTab_PvsD)
```

```
             logFC  AveExpr         t      P.Value    adj.P.Val        B
16877019 -3.209383 7.109180 -13.17845 3.443509e-11 2.265829e-07 15.58869
16705159 -4.071896 6.708237 -12.32082 1.123177e-10 3.642590e-07 14.50427
```

```
17061662 -3.811231 7.933510 -12.04701 1.660755e-10 3.642590e-07 14.14229
16702571 -3.165309 6.514993 -11.75573 2.536979e-10 4.173330e-07 13.74843
16799793 -3.557672 8.113105 -11.30305 4.980980e-10 6.554970e-07 13.11781
16904780 -3.017683 5.638834 -10.92958 8.825244e-10 9.678350e-07 12.57996
```

Se observa que tanto los p-valores como los p-valores ajustados son singificativos (<0.05), lo que significa que habría que rechazar las tres hipótesis nulas propuestas. Esto implica que el efecto del agente inhibidor *Para* en la expresión diferencial de genes es diferente al efecto de los otros agentes.

4.13 Anotación de genes

El siguiente paso consiste en añadir información adicional a los genes seleccionados previamente. Este proceso se denomina anotación y básicamente trata de combinar la información presente en los arrays y la información de cada gen almacenada en diferentes bases de datos a partir del identificador de los probesets del array.

Para llevar a cabo esta anotación, se define la función `annotatedTopTable()` que sirve para añadir los datos de cada gen al listado de los genes seleccionados. Posteriormente, se guardaron las listas anotadas en ficheros *.csv* para cada una de las comparaciones.

A continuación se muestran las listas anotadas para cada uno de los contrastes.

```
> head(topAnnotated_PvsO)
```

```
   PROBEID SYMBOL ENTREZID                                      GENENAME
1 16657594  ISG15     9636              ISG15 ubiquitin like modifier
2 16657598   AGRN   375790                                     agrin
3 16657654 MIR429   554210                                microRNA 429
4 16657767   VWA1    64856 von Willebrand factor A domain containing 1
5 16658184 TPRG1L   127262          tumor protein p63 regulated 1 like
6 16658536   PER3     8863                period circadian regulator 3
       logFC  AveExpr          t      P.Value    adj.P.Val         B
1 -0.2622283 7.100654 -0.4964045 6.251358e-01 7.757542e-01 -6.586130
2 -1.0215985 7.386909 -2.1799934 4.164777e-02 1.420999e-01 -4.513017
3 -0.2615470 2.934123 -0.7874617 4.404403e-01 6.341570e-01 -6.396835
4 -0.7223445 6.821458 -2.5344660 1.992265e-02 8.424872e-02 -3.842165
5  1.5165118 7.384093  2.7315336 1.301878e-02 6.180633e-02 -3.447340
6 -2.0764685 5.805498 -9.6866212 6.595069e-09 2.410864e-06 10.671817
```

```
> head(topAnnotated_PvsM)
```

```
   PROBEID SYMBOL ENTREZID                                      GENENAME
1 16657594  ISG15     9636              ISG15 ubiquitin like modifier
2 16657598   AGRN   375790                                     agrin
3 16657654 MIR429   554210                                microRNA 429
4 16657767   VWA1    64856 von Willebrand factor A domain containing 1
5 16658184 TPRG1L   127262          tumor protein p63 regulated 1 like
6 16658536   PER3     8863                period circadian regulator 3
       logFC  AveExpr          t      P.Value    adj.P.Val         B
1 -0.1466874 7.100654 -0.2776827 7.841667e-01 8.802311e-01 -6.669322
2 -1.0784461 7.386909 -2.3013006 3.250610e-02 1.198936e-01 -4.286304
3 -0.2147087 2.934123 -0.6464418 5.254987e-01 7.028011e-01 -6.495072
```

```
4 -0.7167249 6.821458 -2.5147489 2.077755e-02 8.756648e-02 -3.877471
5  1.5454225 7.384093  2.7836075 1.161646e-02 5.790629e-02 -3.337470
6 -2.0798835 5.805498 -9.7025522 6.420146e-09 2.346920e-06 10.696313
```

```
> head(topAnnotated_PvsD)
```

```
   PROBEID SYMBOL ENTREZID                                    GENENAME
1 16657594  ISG15     9636               ISG15 ubiquitin like modifier
2 16657598   AGRN   375790                                      agrin
3 16657654 MIR429   554210                                microRNA 429
4 16657767   VWA1    64856 von Willebrand factor A domain containing 1
5 16658184 TPRG1L   127262          tumor protein p63 regulated 1 like
6 16658536   PER3     8863                period circadian regulator 3
       logFC  AveExpr          t      P.Value    adj.P.Val         B
1 -0.1608041 7.100654 -0.3044059 7.640303e-01 8.744183e-01 -6.654168
2 -1.0538717 7.386909 -2.2488613 3.620428e-02 1.310863e-01 -4.377071
3 -0.4401477 2.934123 -1.3251900 2.003648e-01 3.981880e-01 -5.831830
4 -0.6734384 6.821458 -2.3628709 2.861018e-02 1.108871e-01 -4.163854
5  1.7062999 7.384093  3.0733791 6.100755e-03 3.632784e-02 -2.724294
6 -2.0032676 5.805498 -9.3451429 1.181955e-08 3.535119e-06 10.108049
```

4.14 Visualización de la expresión diferencial

Para visualizar la expresión diferencial de manera general se pueden utilizar los volcano plots. Estos gráficos muestran con claridad los genes con un gran fold-change, esto es, con p-valores muy altos y por tanto con probabilidades elevadas de expresarse diferencialmente. En el eje abscisa se representa el efecto biológico al representar los cambios de expresión en escala logarítmica. En cambio, el eje ordenada representa el efecto estadístico al representar el logaritmo negativo de los p-valores. Por tanto, los genes más alejados del centro del gráfico son los más significativos, esto es, los primeros de la lista de selección de genes. Se muestran los tre gráficos para cada una de las comparaciones: *PvsO* (Figure 15), *PvsM* (Figure 16) y *PvsD* (Figure 17).

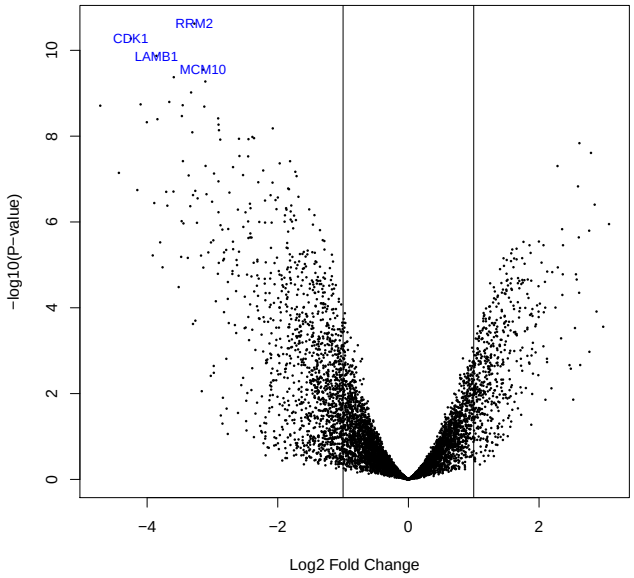

Figure 15: Genes expresados diferencialmente para el contraste PvsO

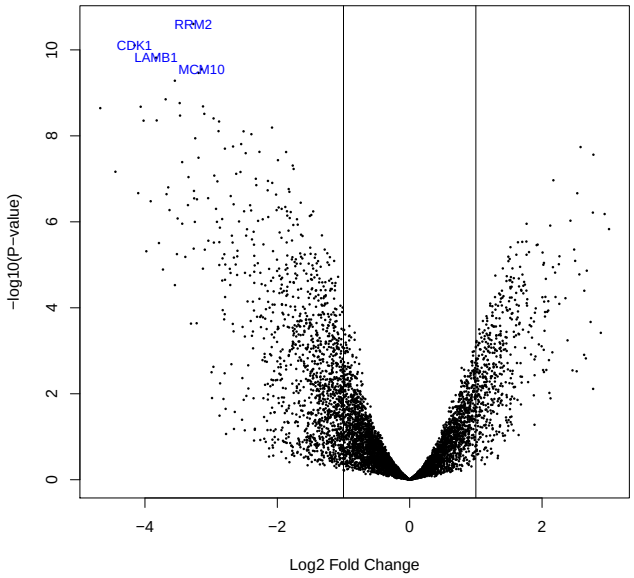

Figure 16: Genes expresados diferencialmente para el contraste PvsM

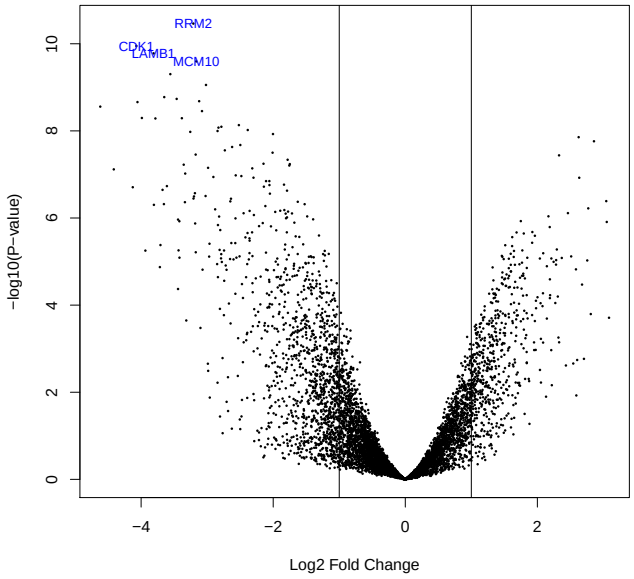

Figure 17: Genes expresados diferencialmente para el contraste PvsD

4.15 Comparaciones múltiples

En estudios en los que se realizan varias comparaciones es interesante conocer qué genes se han seleccionado en cada contraste. En ocasiones, y dependiendo del estudio, pueden interesar los genes seleccionados solo en una de las comparaciones o los que se hayan seleccionado en todas. La función `decideTests()` del paquete `limma` permite hacer estas comparaciones múltiples. A continuación se muestran el número de genes diferencialemente expresados para las tres comparaciones.

```
       PvsO PvsM PvsD
Down    866  849  785
NotSig 5295 5313 5313
Up      419  418  482
```

Una manera muy común de representar los resultados de estas compraciones es mediante diagramas de Venn (Figure 18).

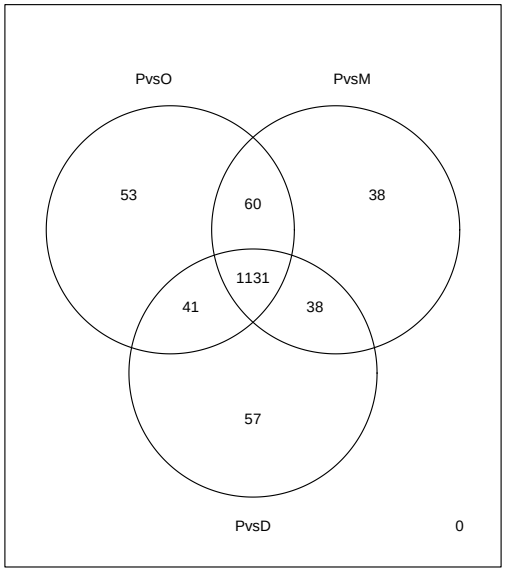

Figure 18: Genes en común de las tres comparaciones

En este caso se observa claramente que la amplia mayoria de los genes diferencialmente expresados se han seleccionado en las tres comparaciones.

Otra forma de visualizar estas comparaciones es mediante heatmaps. En estos gráficos se utiliza un paleta de colores para resaltar diferentes niveles de expresión. Estos mapas son interesantes dado que es posible ordenar los genes seleccionados mediante clusters jerárquicos lo que puede proporcionar una mayor comprensión biológica de los resultados.

En este caso (Figure 19) se observa como claramente las lineas celulares con el agente *Para* tienen un efecto diferente al resto. Esto se observa tanto en la organización del cluster como en el patrón de colores de los grupos.

4.16 Significación biológica de los resultados

Tras conseguir un listado de genes que caracterizan la diferencia entre dos condiciones, éste necesita ser interpretado. Esto requiere cierto conocimiento biológico del problema a estudiar pero métodos estadísicos como *Gene Enrichment Analysis* puede ser de ayuda a la hora de interpretar los resultados. Este método

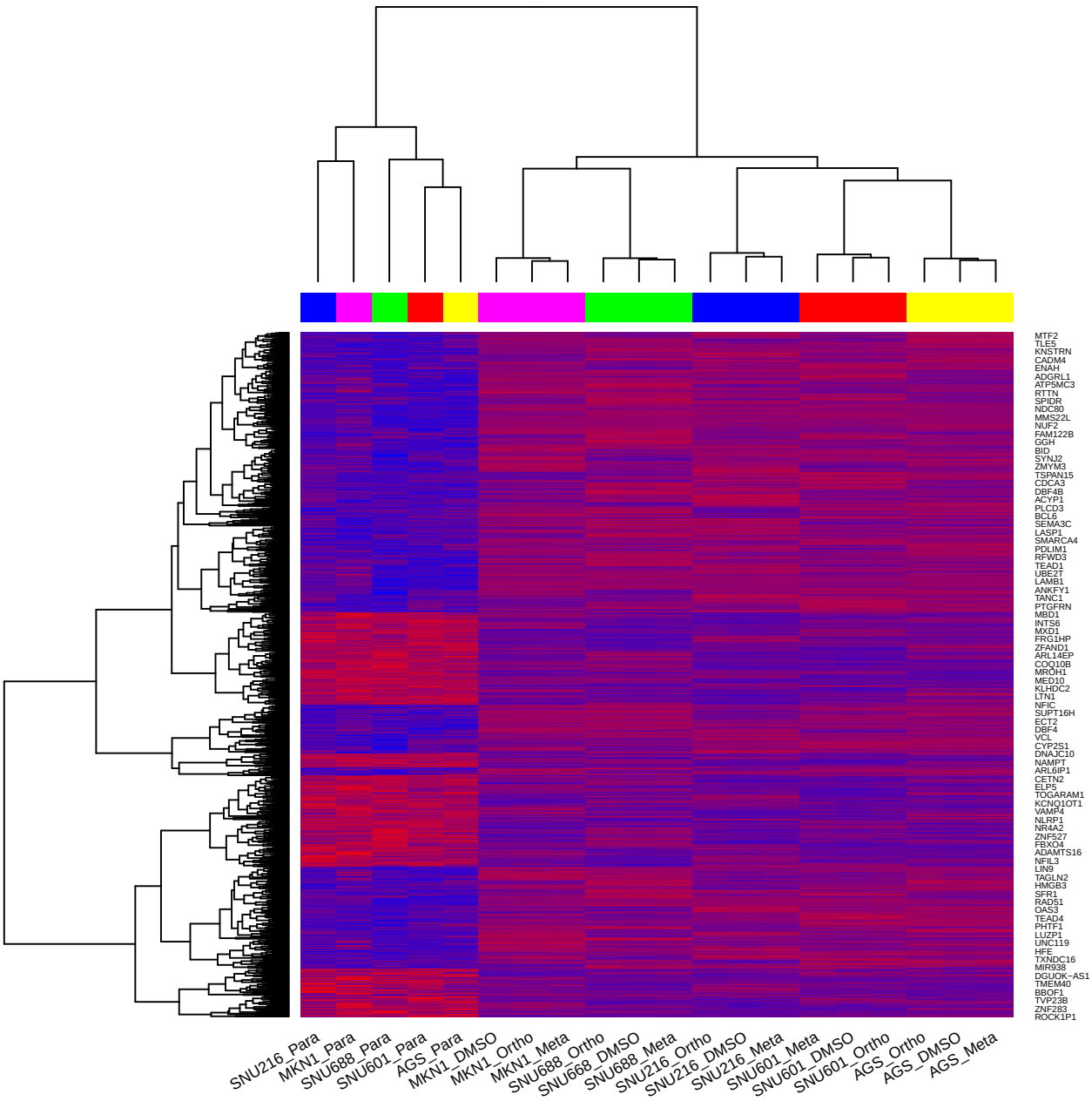

Figure 19: Genes expresados diferencialmente

se basa en que las funciones biológicas o las rutas moleculares de estos genes diferencialmente expresados se encuentren con más frecuencia en este listado de genes que en el resto de los genes no seleccionados.

En este caso, se utilizó el análisis básico de enriquecimiento implementado en el paquete `ReactomePA` utilizando su base de datos de anotaciones (https://reactome.org). Tras preparar la lista de genes a analizar, se añadieron los identificadores *Entrez* de los genes restantes para lo que se utilizó la anotación de *Gene Ontology*. Por último, se guardaron los ficheros con los resultados correspondientes.

A continuación se muestran las primeras filas de los listados enriquecidos para los contrastes *PvsO* (Table 2), *PvsM* (Table 3) y *PvsD* (Table 4).

Table 2: Inicio del listado de Reactome para la comparación PvsO

	Description	pvalue	p.adjust
R-HSA-69620	Cell Cycle Checkpoints	1.27094746756331e-22	1.60139380912977e-19
R-HSA-195258	RHO GTPase Effectors	2.10471705450959e-18	1.32597174434104e-15
R-HSA-68886	M Phase	1.63444072650905e-17	6.864651051338e-15
R-HSA-68877	Mitotic Prometaphase	8.10202062726186e-15	2.08986329006344e-12
R-HSA-194315	Signaling by Rho GTPases	8.29310829390252e-15	2.08986329006344e-12

Table 3: Inicio del listado de Reactome para la comparación PvsM

	Description	pvalue	p.adjust
R-HSA-69620	Cell Cycle Checkpoints	1.68764291316949e-21	2.1179918560277e-18
R-HSA-68886	M Phase	3.1351602593321e-17	1.9673130627309e-14
R-HSA-195258	RHO GTPase Effectors	6.19602469945521e-17	2.59200366593876e-14
R-HSA-68877	Mitotic Prometaphase	6.9294851716679e-16	2.1741259726108e-13
R-HSA-141424	Amplification of signal from the kinetochores	6.89935427709512e-15	1.4431149362924e-12

Table 4: Inicio del listado de Reactome para la comparación PvsD

	Description	pvalue	p.adjust
R-HSA-69620	Cell Cycle Checkpoints	3.23207653029119e-21	4.05625604551544e-18
R-HSA-68886	M Phase	4.7744456085484e-18	2.99596461936412e-15
R-HSA-195258	RHO GTPase Effectors	9.38029627411756e-18	3.92409060800584e-15
R-HSA-68877	Mitotic Prometaphase	1.69166028326347e-15	5.30758413873915e-13
R-HSA-141424	Amplification of signal from the kinetochores	4.82467307056422e-15	1.00916078392635e-12

Estos listados se pueden representar gráficamente mediante un barplot (Figure 20) o mediante un gráfico que representa las redes de las rutas enriquecidas y la relación de los genes (Figure 21). En ambos casos se muestran los resultados para la comparación *PvsO*.

4.17 Resumen de resultados

Con esto se podría dar por finalizado el estudio sobre los datos de microarrays pero es conveniente resumir los ficheros de resultados generados durante el análisis en un listado dado que en ocasiones se genera un gran número de ficheros (Table 5).

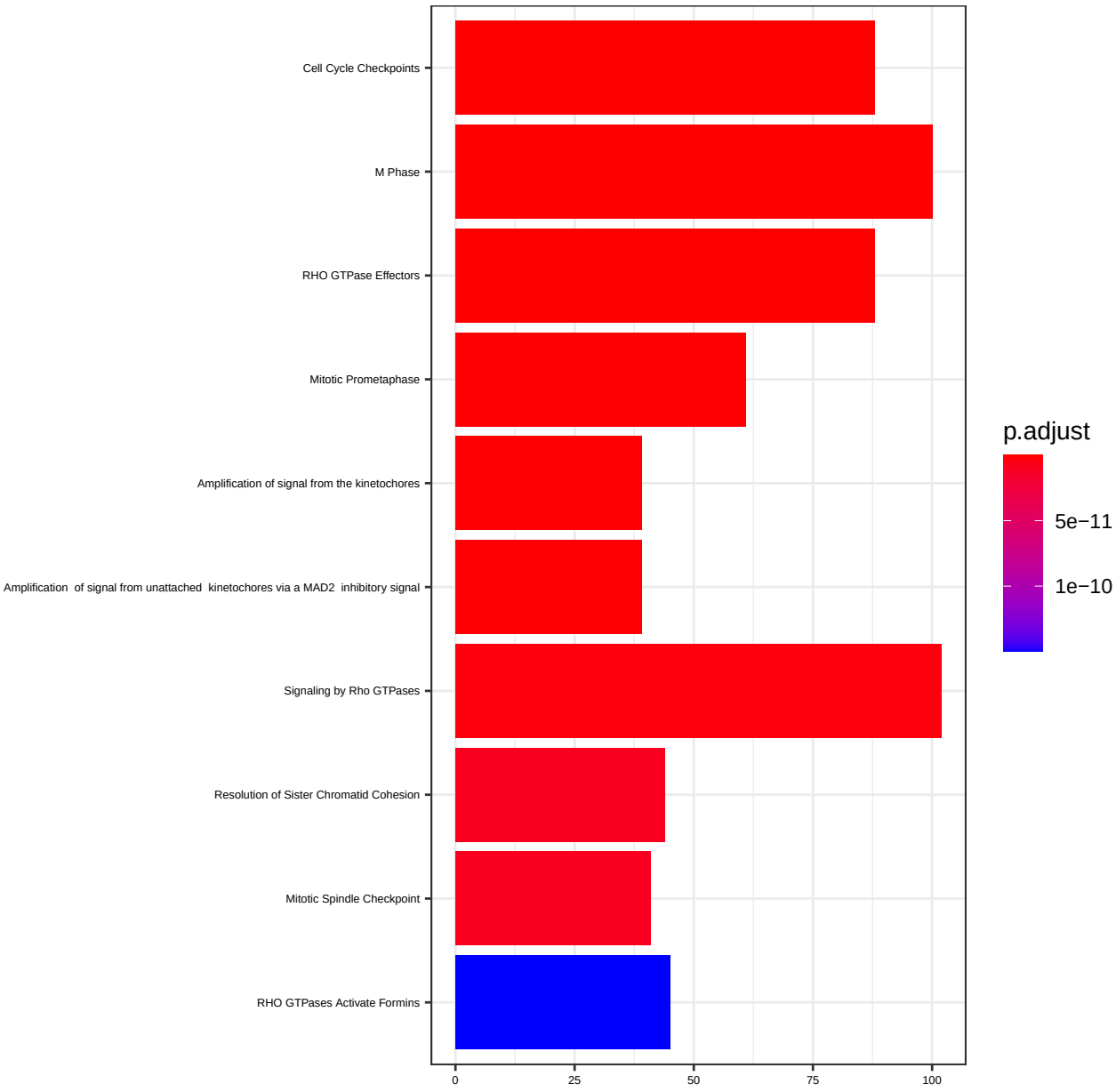

Figure 20: Barplot del análisis de ReactomePA para PvsO

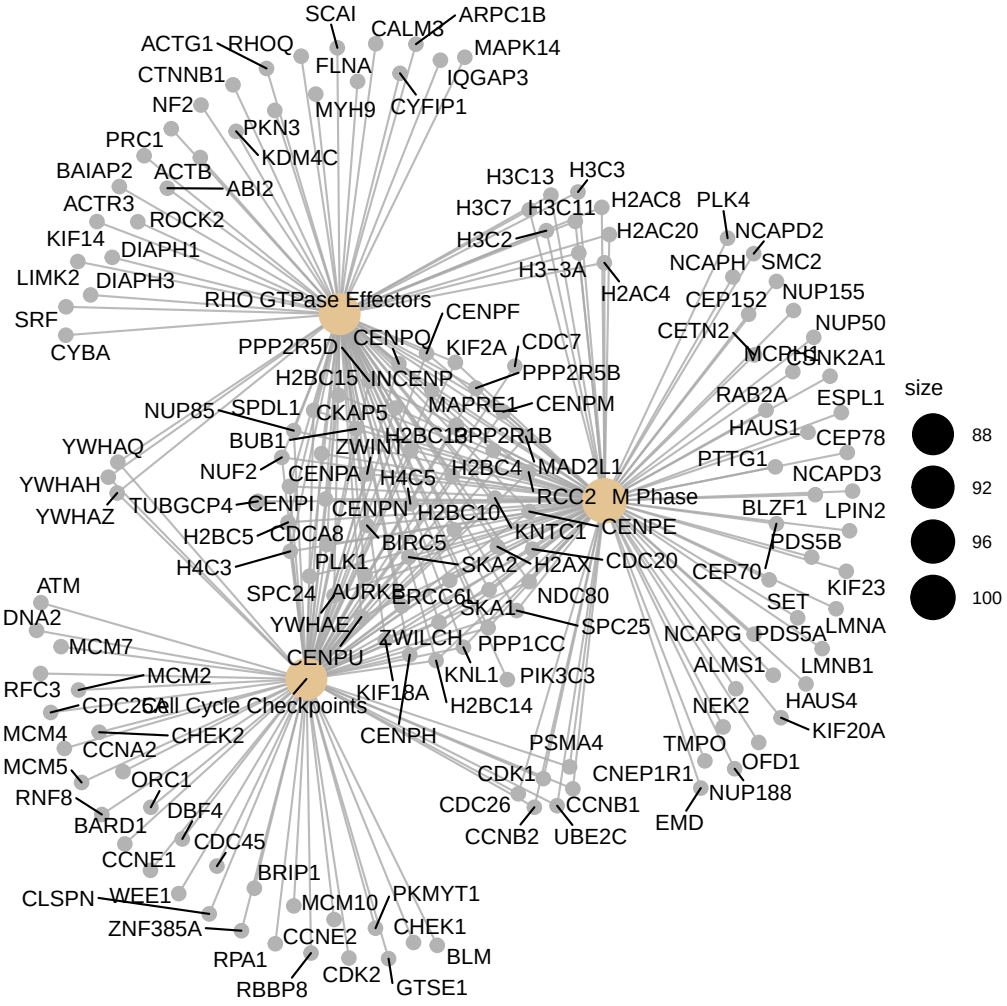

Figure 21: Red obtenida del analisis de enriquecimiento sobre la lista de la comparacion PvsO

Table 5: Listado de ficheros generados durante el análisis

ListadoFicheros
data4Heatmap.csv
filtered.Data.csv
normalized.Data.csv
normalized.filtered.Data.Rda
QCDir.Filt
QCDir.Norm
QCDir.Raw
ReactomePA.Results.PvsD.csv
ReactomePA.Results.PvsM.csv
ReactomePA.Results.PvsO.csv
ReactomePABarplot.PvsD.pdf
ReactomePABarplot.PvsM.pdf
ReactomePABarplot.PvsO.pdf
ReactomePAcnetplot.PvsD.pdf
ReactomePAcnetplot.PvsM.pdf
ReactomePAcnetplot.PvsO.pdf
topAnnotated_PvsD.csv
topAnnotated_PvsM.csv
topAnnotated_PvsO.csv

5 CONCLUSIÓN

El principal resultado del análisis ha sido que el efecto del agente inhibidor *Para* en la expresión diferencial de genes es muy diferente al de los otros agentes dado que los p-valores de las comparaciones han sido significativos. Este resultado está en concordancia con lo observado por los autores (Kim 2019).

6 APÉNDICE

En este apartado se muestra en código utilizado a lo largo de todo el informe.

```
> #Preparación del entorno
>
> setwd("~/Desktop/Master UOC/Semestre 3/Asignaturas/
+       157-Analisis de datos omicos/PEC1/ADO_PEC1")
> workingDir <- getwd()
> dir.create("data")
> dir.create("results")
> dataDir <- file.path(workingDir, "data/")
> resultsDir <- file.path(workingDir, "results/")
>
> #Obtención de datos
>
> knitr::include_graphics("figuras/BusquedaGEO.png")
>
> library(GEOquery)
>
> my_gse <- getGEO(GEO = "GSE114626", destdir = dataDir)
> getGEOSuppFiles(GEO = "GSE114626", makeDirectory = F, baseDir = dataDir)
>
> targets <- read.csv2("./data/targets.csv", header = TRUE, sep = ";")
> knitr::kable(
+   targets, booktabs = TRUE,
+   caption = 'Contenido del fichero *targets.csv* para el estudio GSE114626')
>
> #Lectura de datos
>
> library(Biobase)
> library(oligo)
> library(pd.hugene.2.0.st)
>
> celFiles <- list.celfiles("./data", full.names = T) #get list of .cel files
> my.targets <- read.AnnotatedDataFrame(file.path("./data", "targets.csv"),
+                                       header = T, row.names = 1, sep = ";")
> rawData <- read.celfiles(celFiles, phenoData = my.targets) #reads .cel files
> pData(rawData)
> my.targets@data$ShortName -> rownames(pData(rawData)) #change names of the samples
> colnames(rawData) <- rownames((pData(rawData)))
> rawData #ExpressionSet: both .cel and target in one object
>
> #Control de calidad de datos crudos
>
> library(arrayQualityMetrics)
>
> arrayQualityMetrics(rawData, outdir = file.path("./results", "QCDir.Raw"))
>
> knitr::include_graphics("figuras/indexRaw.png")
>
> library(ggplot2)
> library(ggrepel)
```

```
>
> boxplot(rawData, cex.axis = 0.5, las=2, which="all", main="",
+         col = c(rep("red", 4), rep("blue", 4), rep("green", 4),
+                 rep("yellow", 4), rep("magenta", 4)))
>
> plot(hclust(dist(t(exprs(rawData)))))
>
> plotPCA <- function(datos, labels, factor, 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$x[,1])+5)) +
+     scale_fill_discrete(name = "Group")
+   #avoiding labels superposition
+   p1 + geom_text_repel(aes(y = PC2 + 0.25, label = labels),
+                        segment.size = 0.25, size = size) +
+     labs(x = c(paste("PC1", loads[1], "%")), y = c(paste("PC2", loads[2], "%"))) +
+     theme(plot.title = element_text(hjust = 0.5)) +
+     scale_colour_manual(values = colores)
+ }
>
> plotPCA(exprs(rawData), labels = my.targets@data$ShortName,
+         factor = my.targets@data$Agent,
+         scale = F, size = 3, colores = c("red", "blue", "green", "yellow"))
>
> #Normalización de datos
>
> eset_rma <- rma(rawData)
>
> #Control de calidad de datos normalizados
>
> arrayQualityMetrics(eset_rma, outdir = file.path("./results", "QCDir.Norm"))
>
> knitr::include_graphics("figuras/indexNorm.png")
>
> boxplot(eset_rma, cex.axis = 0.5, las=2, which="all",
+         col = c(rep("red", 4), rep("blue", 4), rep("green", 4),
+                 rep("yellow", 4), rep("magenta", 4)))
>
> plot(hclust(dist(t(exprs(eset_rma)))))
>
> plotPCA(exprs(eset_rma), labels = my.targets@data$ShortName,
+         factor = my.targets@data$Agent,
+         scale = F, size = 3, colores = c("red", "blue", "green", "yellow"))
```

```
>
> #Detección de los genes más variables
>
> sds <- apply(exprs(eset_rma), 1, sd)
> sds0 <- sort(sds)
> plot(1:length(sds0), sds0, xlab="Genes desde el menos al más variable",
+      ylab="Desviación estándar")
> abline(v=length(sds)*c(0.9,0.95))
>
> #Filtrado de genes
>
> library(genefilter)
> library(hugene20sttranscriptcluster.db)
>
> annotation(eset_rma) <- "hugene20sttranscriptcluster.db"
> filtered <- nsFilter(eset_rma,
+                      require.entrez = T, remove.dupEntrez = T,
+                      var.filter = T, var.func = IQR, var.cutoff = 0.75,
+                      filterByQuantile = T, feature.exclude = "^AFFX")
>
> eset_filtered <- filtered$eset
>
> #Control de calidad de datos filtrados
>
> arrayQualityMetrics(eset_filtered, outdir = file.path("./results", "QCDir.Filt"))
>
> knitr::include_graphics("figuras/indexFilt.png")
>
> boxplot(eset_filtered, cex.axis = 0.5, las=2, which="all",
+         col = c(rep("red", 4), rep("blue", 4), rep("green", 4),
+                 rep("yellow", 4), rep("magenta", 4)))
>
> plot(hclust(dist(t(exprs(eset_filtered)))))
>
> plotPCA(exprs(eset_filtered), labels = my.targets@data$ShortName,
+         factor = my.targets@data$Agent,
+         scale = F, size = 3, colores = c("red", "blue", "green", "yellow"))
>
> write.csv(exprs(eset_rma), file = "./results/normalized.Data.csv")
> write.csv(exprs(eset_filtered), file = "./results/filtered.Data.csv")
> save(eset_rma, eset_filtered, file = "./results/normalized.filtered.Data.Rda")
>
> #Diseño experimental
>
> library(limma)
>
> designMat <- model.matrix(~0+Agent, pData(eset_filtered))
> colnames(designMat) <- c("DMSO", "Meta", "Ortho", "Para")
>
> designMat
>
> cont.matrix <- makeContrasts(PvsO = Para-Ortho,
+                              PvsM = Para-Meta,
```

```
+                               PvsD = Para-DMSO,
+                               levels = designMat)
>
> cont.matrix
>
> #Modelización y selección de genes
>
> fit <- lmFit(eset_filtered, designMat)
> fit.main <- contrasts.fit(fit, cont.matrix)
> fit.main <- eBayes(fit.main)
>
> #Listado de genes diferencialmente expresados
>
> topTab_PvsO <- topTable(fit.main, number = nrow(fit.main),
+                         coef="PvsO", adjust="fdr")
>
> topTab_PvsM <- topTable(fit.main, number = nrow(fit.main),
+                         coef="PvsM", adjust="fdr")
>
> topTab_PvsD <- topTable(fit.main, number = nrow(fit.main),
+                         coef="PvsD", adjust="fdr")
>
> head(topTab_PvsO)
>
> head(topTab_PvsM)
>
> head(topTab_PvsD)
>
>
> #Anotación de genes
>
> 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_PvsO <- annotatedTopTable(topTab_PvsO,
+                                        anotPackage = "hugene20sttranscriptcluster.db")
>
> topAnnotated_PvsM <- annotatedTopTable(topTab_PvsM,
+                                        anotPackage = "hugene20sttranscriptcluster.db")
>
> topAnnotated_PvsD <- annotatedTopTable(topTab_PvsD,
+                                        anotPackage = "hugene20sttranscriptcluster.db")
>
> head(topAnnotated_PvsO)
> head(topAnnotated_PvsM)
> head(topAnnotated_PvsD)
```

```
>
> write.csv(topAnnotated_PvsO, file="./results/topAnnotated_PvsO.csv")
> write.csv(topAnnotated_PvsM, file="./results/topAnnotated_PvsM.csv")
> write.csv(topAnnotated_PvsD, file="./results/topAnnotated_PvsD.csv")
>
> #Visualización de la expresión diferencial
>
> library(hugene20sttranscriptcluster.db)
>
> geneSymbols <- select(hugene20sttranscriptcluster.db, rownames(fit.main), c("SYMBOL"))
> SYMBOLS <- geneSymbols$SYMBOL
>
> volcanoplot(fit.main, coef = 1, highlight = 4, names = SYMBOLS)
> abline(v=c(-1,1))
>
> volcanoplot(fit.main, coef = 2, highlight = 4, names = SYMBOLS)
> abline(v=c(-1,1))
>
> volcanoplot(fit.main, coef = 3, highlight = 4, names = SYMBOLS)
> abline(v=c(-1,1))
>
> #Comparaciones múltiples
>
> library(gplots)
>
> res <- decideTests(fit.main, method = "separate",
+                    adjust.method = "fdr", p.value = 0.1, lfc = 1)
> sum.res.rows <- apply(abs(res),1,sum)
> res.selected <- res[sum.res.rows!=0,]
> print(summary(res))
>
> vennDiagram(res.selected[,1:3], cex=0.8)
>
> probesInHeatmap <- rownames(res.selected)
> HMdata <- exprs(eset_filtered)[rownames(exprs(eset_filtered)) %in% probesInHeatmap,]
> geneSymbols <- select(hugene20sttranscriptcluster.db, rownames(HMdata), c("SYMBOL"))
> SYMBOLS <- geneSymbols$SYMBOL
> rownames(HMdata) <- SYMBOLS
> my_palette <- colorRampPalette(c("blue", "red"))(n =299)
>
> write.csv(HMdata, file=file.path("./results/data4Heatmap.csv"))
>
> heatmap.2(HMdata, Rowv = T, Colv = T,
+           scale = "row", col = my_palette, sepcolor = "white",
+           sepwidth = c(0.05,0.05), cexRow = 0.5, cexCol = 0.9,
+           key = F, density.info = "histogram",
+           ColSideColors = c(rep("red", 4), rep("blue", 4), rep("green", 4),
+                             rep("yellow", 4), rep("magenta", 4)),
+           tracecol = NULL, dendrogram = "both", srtCol = 30)
>
> #Significación biológica de los resultados
>
> library(org.Hs.eg.db)
```

```
> library(ReactomePA)
>
> listOfTables <- list(PvsO = topTab_PvsO,
+                      PvsM = topTab_PvsM,
+                      PvsD = topTab_PvsD)
>
> listOfSelected <- list()
>
> for(i in 1:length(listOfTables)){
+   topTab <- listOfTables[[i]]
+   whichGenes <- topTab["adj.P.Val"]<0.15
+   selectedIDs <- rownames(topTab)[whichGenes]
+   EntrezIDs <- select(hugene20sttranscriptcluster.db, selectedIDs, c("ENTREZID"))
+   EntrezIDs <- EntrezIDs$ENTREZID
+   listOfSelected[[i]] <- EntrezIDs
+   names(listOfSelected)[i] <- names(listOfTables)[i]
+ }
>
> sapply(listOfSelected, length)
>
> mapped_genes2GO <- mappedkeys(org.Hs.egGO)
> mapped_genes2KEGG <- mappedkeys(org.Hs.egPATH)
> mapped_genes <- union(mapped_genes2GO, mapped_genes2KEGG)
>
> listOfData <- listOfSelected[1:3]
> comparisonsNames <- names(listOfData)
> universe <- mapped_genes
>
> 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",
+                                  universe = universe)
+ }
>
> if(length(rownames(enrich.result@result))!=0) {
+   write.csv(as.data.frame(enrich.result),
+             file=paste0("./results/", "ReactomePA.Results.", comparison, ".csv"),
+             row.names = F)
+   pdf(file=paste0("./results/", "ReactomePABarplot.", comparison, ".pdf"))
+   pdf(file=paste0("./results/", "ReactomePAcnetplot.", comparison, ".pdf"))
+ }
>
> Tab.react.PvsO <- read.csv2(file.path("./results/ReactomePA.Results.PvsO.csv"),
+                             sep = ",", header = TRUE, row.names = 1)
>
> Tab.react.PvsO <- Tab.react.PvsO[1:5, c(1,4,5)]
> knitr::kable(Tab.react.PvsO, booktabs = TRUE,
+              caption = "Inicio del listado de Reactome para la comparación PvsO")
```

```
>
> Tab.react.PvsM <- read.csv2(file.path("./results/ReactomePA.Results.PvsM.csv"),
+                             sep = ",", header = TRUE, row.names = 1)
>
> Tab.react.PvsM <- Tab.react.PvsM[1:5, c(1,4,5)]
> knitr::kable(Tab.react.PvsM, booktabs = TRUE,
+              caption = "Inicio del listado de Reactome para la comparación PvsM")
>
> Tab.react.PvsD <- read.csv2(file.path("./results/ReactomePA.Results.PvsD.csv"),
+                             sep = ",", header = TRUE, row.names = 1)
>
> Tab.react.PvsD <- Tab.react.PvsD[1:5, c(1,4,5)]
> knitr::kable(Tab.react.PvsD, booktabs = TRUE,
+              caption = "Inicio del listado de Reactome para la comparación PvsD")
>
> comparison <- comparisonsNames[1]
>
> barplot(enrich.result, showCategory=10, font.size=5)
>
> cnetplot(enrich.result, categorySize="geneNum", showCategory = 3,
+          vertex.label.cex = 0.2)
>
> #Resumen de resultados
>
> listOfFiles <- dir("./results/")
> knitr::kable(
+   listOfFiles, booktabs = TRUE,
+   caption = "Listado de ficheros generados durante el análisis",
+   col.names="ListadoFicheros"
+ )
```

REFERENCIAS

Benjamini, Y., Y.; Hochberg. 1995. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society. Series B (Methodological)* 57 (1): 289–300. https://www.jstor.org/stable/2346101?seq=1.

Hackstadt, A.M., A.J.; Hess. 2009. "Filtering for Increased Power for Microarray Data." *BMC Bioinformatics* 10: 11. doi:10.1186/1471-2105-10-11.

Irizarry, B, R.A.; Hobbs. 2003. "Exploration, Normalization, and Summaries of High Density Oligonucleotide Array Probe Level Data." *Biostatistics* 4 (2): 249–64. doi:10.1093/biostatistics/4.2.249.

Kim, H.J.; Lim, J.; Eom. 2019. "Differential Effects, on Oncogenic Pathway Signalling, by Derivatives of the Hnf α Inhibitor Bi6015." *British Journal of Cancer* 120: 488–98. doi:10.1038/s41416-018-0374-5.

Smyth, G.K. 2004. "Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments." *Statistical Applications in Genetics and Molecular Biology* 3 (1): 1–25. doi:10.2202/1544-6115.1027.