PEC1

Javier Amado Bouza

29 de abril, 2020

Table of Contents

El link al repositorio de GitHub es:https://github.com/amadobouza/amadobouza_PEC1_ADO.git

1. Abstract

En la búsqueda urgente de nuevas dianas terapéuticas frente al cáncer de próstata resistente a castración, los científicos al cargo del estudio Terada (2010) cuyos resultados están almacenados en GEO como GSE21887 estudiaron tumores obtenidos de ratas mediante la técnica xenograft. Entre otras técnicas analizaron los perfiles de expresión génica en varias condiciones.

2. Objetivos

Estudiar las diferencias en la expresión génica entre 3 estadíos diferentes en la evolución del cáncer de próstata.

3. Materiales y métodos

3.1 Naturaleza de los datos

Los datos se obtuvieron por descarga desde la base de datos GEO, utilizando el número de acceso GSE21887[1]. Estos datos se encontraban en forma de 12 archivos CEL(uno por cada array), ya que son derivados de experimentos realizados con arrays de la marca Affymetrix. Concretamente del modelo Affymetrix Human Genome U133 Plus 2.0 Array.

Se trata de un estudio con 1 solo factor, que es el cáncer de próstata. En este experimento se utilizan tres niveles, que se corresponden con el estadío del tumor cuando se extrae para el análisis. El primero de los niveles es el crecimiento dependiente de andrógeno (AR). El segundo de los niveles es el nadir de la regresión inducido por castración (ND). Y por último el tercero de los niveles es el del recrecimiento resistente a castración (CR). Para cada uno de los 3 niveles se usaron 4 réplicas biológicas lo que nos da un total de 12 unidades experimentales.

3.2 Métodos utilizados durante el procedimiento de análisis.

Para este estudio nos hemos basado en el "pipeline" sugerido en el pdf de esta PEC.

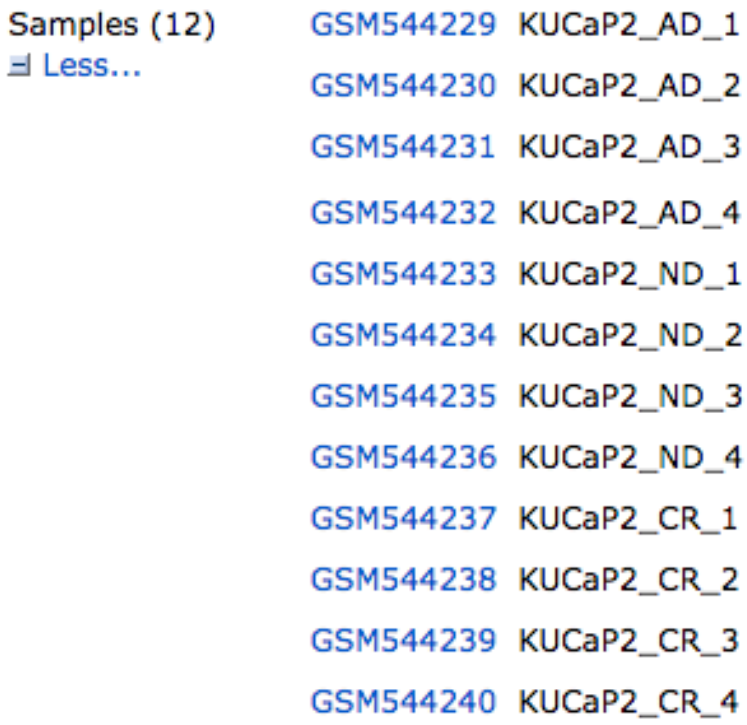
3.2.1 Identificar qué grupos hay y a qué grupo pertenece cada muestra

En este caso existen 3 grupos:

[1] https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE21887

- AR - Microarrays con muestras de tumores en niveles de crecimiento dependiente de andrógeno.
- ND - Microarrays con muestras de tumores en la fase nadir de la regresión inducida por castracion.
- CR - Microarrays con muestras de tumores en la fase de crecimiento resistente a la castración.

Los códigos de los arrays hibridados en este experimento son los siguientes (Fig. 1):



Arrays de este experimento

Podemos observar que los cuatro primeros arrays pertenecen al grupo AD, los cuatro siguientes pertenecen al grupo ND, y os cuatro últimos pertenecen al grupo CR.

3.2.2 Descarga de los archivos CEL y carga de los mismos en RStudio

Una vez identificados los grupos a los que pertenece cada array, pinchamos en el link para descargar el archivo comprimido en el que se encuentran los archivos CEL de cada uno de los 12 arrays utilizados en este estudio.

Se trata de archivos CEL ya que son los resultantes de leer las intensidades de arrays de Affymetrix.

Este procedimiento es fundamental para empezar el análisis de datos de microarrays con R y Bioconductor.

Descomprimimos esos datos, y los cargamos en el entorno de RStudio utilizando la función ReadAffy. El resultado es un objeto de tipo affybatch con todos los datos contenidos en los 12 arrays.

3.2.3 Control de calidad de los datos crudos

Sometemos los datos del affybatch obtenido anteriormente a un control de calidad realizado por la función arrayQualityMetrics.

Este control es importante porque nos proporciona una serie de resultados gráficos en los que podemos ver cómo se distribuyen los datos de los arrays, y nos marca los arrays que cumplen las características para ser considerados defectuosos.

3.2.4 Normalización

Para este procedimiento utilizamos el método RMA presente en el paquete affy.

Se trata del método estándar para los arrays de la marca Affymetrix, como los utilizados en este experimento.

Este método realiza 3 procedimientos:

- Ajusta el ruído de fondo
- Realiza una normalización por cuantiles, de los valores del logaritmo en base 2 de cada intensidad ajustada por el ruído de fondo.
- Estima las intensidades de cada gen, separadamente para cada conjunto de sondas. Para ello utiliza la técnica conocida como median polish.

El resultado final es una matriz en formato expressionSet con los datos normalizados y sumarizados, y en escala logarítmica.

3.2.5 Control de calidad de los datos normalizados

Utilizando de nuevo la función arrayQualityMetrics, en este caso sobre el expressionSet resultante de la aplicación de la función RMA. Hemos realizado un control de calidad para ver si los datos tienen una estructura correcta.

Decidí hacer este paso porque daba un extra de control sobre la forma de los datos obtenidos tras la normalización.

3.2.6 Filtrado de los genes menos expresados

Este paso nos limpia nuestro expressionset normalizado, de genes que no superan un umbral de expresión, y que por lo tanto no se espera que estén diferencialmente expresados.

Para ello utilizamos la función nsFilter del paquete genefilter.

Además la configuramos para que elimine los datos de los probesets que no tienen datos de identificadores génicos asociados.

Con ello obtenemos un expressionset filtrado sólo con los genes más variables, y que tienen datos asociados.

3.2.7 Identificación de genes diferencialmente expresados

El fin último de este análisis de microarrays es el de identificar genes diferencialmente expresados.

Para identificar estos genes diferencialmente expresados entre los distintos grupos del estudio utilizamos las funciones disponibles en la librería limma. Éstas utilizan modelos lineales para realizar los contrastes.

Se seleccionó esta librería ya que el número de grupos es mayor de dos, y esta librería facilita el trabajo en este caso. Los pasos a seguir fueron los siguientes:

1. Creamos la matriz de diseño del estudio siguiento los datos de los grupos previamente comentados.
2. Aplicando ese diseño, especificamos los contrastes que vamos a realizar, y los guardamos en una matriz de contraste.
3. Finalmente, con los datos normalizados de microarrays, y las matrices de diseño y contraste es posible realizar inferencia sobre los parámetros del modelo, como puede ser la obtención del listado de genes diferencialmente expresados.

Con cada comparación realizada entre grupos, y con la función toptable obtendremos un dataset con los genes diferencialmente expresados, donde entre otros datos se almacenan los códigos de los pocillos del array, el logaritmo del ratio o razón de expresiones (mide cuántas más veces está expresado un gen en una condición frente a la otra), y el p-valor del valor de logFC obtenido.

Tras este paso obtenemos un objeto del tipo MArrayLM, que acumula los valores previamente comentados.

3.2.8 Anotación de los resultados

De poco sirve obtener un listado de pocillos cuya secuencia contenida sufre un cambio de expresión en la condición B si la comparamos con la A.

Para que estos datos de logFC tengan una interpretación más sencilla debemos anotar qué secuencia concreta está en estos pocillos.

Para esto diseñamos una función que asocia los códigos de los pocillos de nuestro objeto MArrayLM a la información almacenada en las bases de datos genómicas. El nexo de unión es una base de datos específica para nuestro modelo de microarray que nos descargaremos utilizando el instalador del proyecto bioconductor. Enlazándola con nuestra nueva función, esta base de datos asociará el nombre del pocillo con el nombre del gen contenido en ella, su símbolo, o su entrezID (que son las tres informaciones que se extrajeron en nuestro estudio).

Así, finalmente obtenemos un data.frame por cada comparación hecha. En el que además del número de pocillo se han incorporado tres nuevas columnas, que contienen el nombre del gen asociado al pocillo, su símbolo, y su entrezID.

3.2.9 Comparación entre distintas comparaciones

Con el objeto MArrayLM del paso 3.2.7, y utilizando las funciones decideTests y VennDiagram del paquete limma anotamos y contamos los genes que se han seleccionado como diferencialmente expresados en cada comparación.

Así podemos ver de un vistazo cuántos genes están diferencialmente expresados en las comparaciones hechas, y cuántos de ellos son comunes a las tres hechas en este caso.

La función decideTests nos da una tabla con en la que las filas son el número total de genes que han incrementado su expresión, los que tienen la misma expresión entre los grupos comparados, y los genes que han disminuido su expresión. Y las columnas son cada una de las tres comparaciones realizadas.

La función VennDiagram como su nombre indica nos proporciona un diagrama de Venn donde los genes diferencialmente expresados de cada comparación se encuentran dentro de una elipse, y nos da también cuáles son comunes dos a dos, y los que son comunes a las tres comparaciones.

3.2.10 Análisis de significación biológica

Se ha realizado este paso porque una vez hechas las listas de los genes diferencialmente expresados, y la comparación entre las distintas comparaciones, un paso que facilita mucho la interpretación de los datos es el análisis de la significación biológica. Este análisis nos da información de la función biológica, procesos biológicos, o rutas moleculares en las que se encuentran involucrados los genes diferencialmente expresados en las comparaciones realizadas.

Para este análisis he utilizado las funciones del paquete ReactomePA.

Como primer paso obtendremos la lista de genes que serán analizados. En este paso nos interesa tener un número elevado de genes, así que utilizaremos un FDR elevado. En este caso utilizamos un FDR <0,15, y no realizamos ningún filtro de fold-change.

El análisis también requiere que todos los genes a analizar tengan su entrezID.

Los resultados obtenidos en este análisis son los siguientes:

- Un archivo .csv con un resumen de todas las rutas enriquecidas en este estudio, y sus estadísticas asociadas.
- Un gráfico de barras con las rutas enriquecidas con mejores estadísticas. La altura del gráfico de barras se corresponde con el número de genes de nuestro análisis que están relacionados con esta ruta. Además, las rutas están ordenadas según su significación estadística.

- Un gráfico que muestra las redes de las rutas enriquecidas, y la relación entre los genes incluidos.

4 Resultados

4.1 Control de calidad de los datos crudos

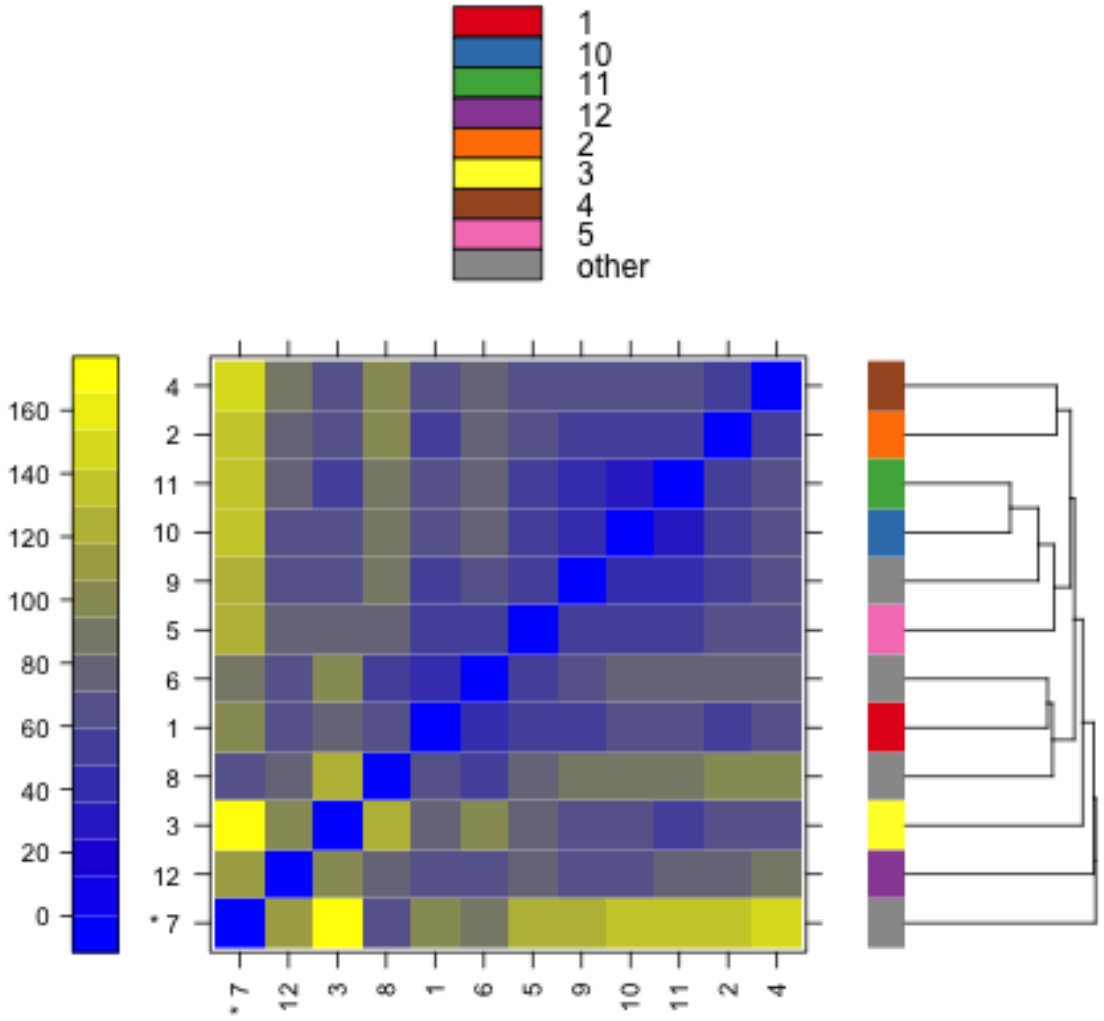
El primer resultado que nos ofrece la función arrayQualityMetrics es una tabla resumen con los nombres de los arrays, fecha de su lectura, y los resultados de las distintas pruebas de detección de arrays defectuosos(Fig. 2).

	array	sampleNames	*1	*2	*3	*4	*5	*6	sample	ScanDate
☐	1	GSM544229.CEL.gz							1	04/11/07 16:13:37
☐	2	GSM544230.CEL.gz							2	04/11/07 16:26:51
☐	3	GSM544231.CEL.gz							3	04/12/07 11:56:42
☐	4	GSM544232.CEL.gz							4	04/12/07 12:08:08
☐	5	GSM544233.CEL.gz							5	04/11/07 16:38:41
☐	6	GSM544234.CEL.gz							6	04/12/07 12:20:11
☐	7	GSM544235.CEL.gz	x		x	x		x	7	04/12/07 12:31:30
☐	8	GSM544236.CEL.gz							8	04/12/07 15:03:18
☐	9	GSM544237.CEL.gz							9	04/11/07 16:53:02
☐	10	GSM544238.CEL.gz							10	04/12/07 15:16:08
☐	11	GSM544239.CEL.gz							11	04/12/07 15:29:20
☐	12	GSM544240.CEL.gz							12	04/12/07 15:40:44

Tabla resumen arrayQualityMetrics de datos no normalizados

Podemos ver que el array número 7 incumple 4 de las 6 métricas.

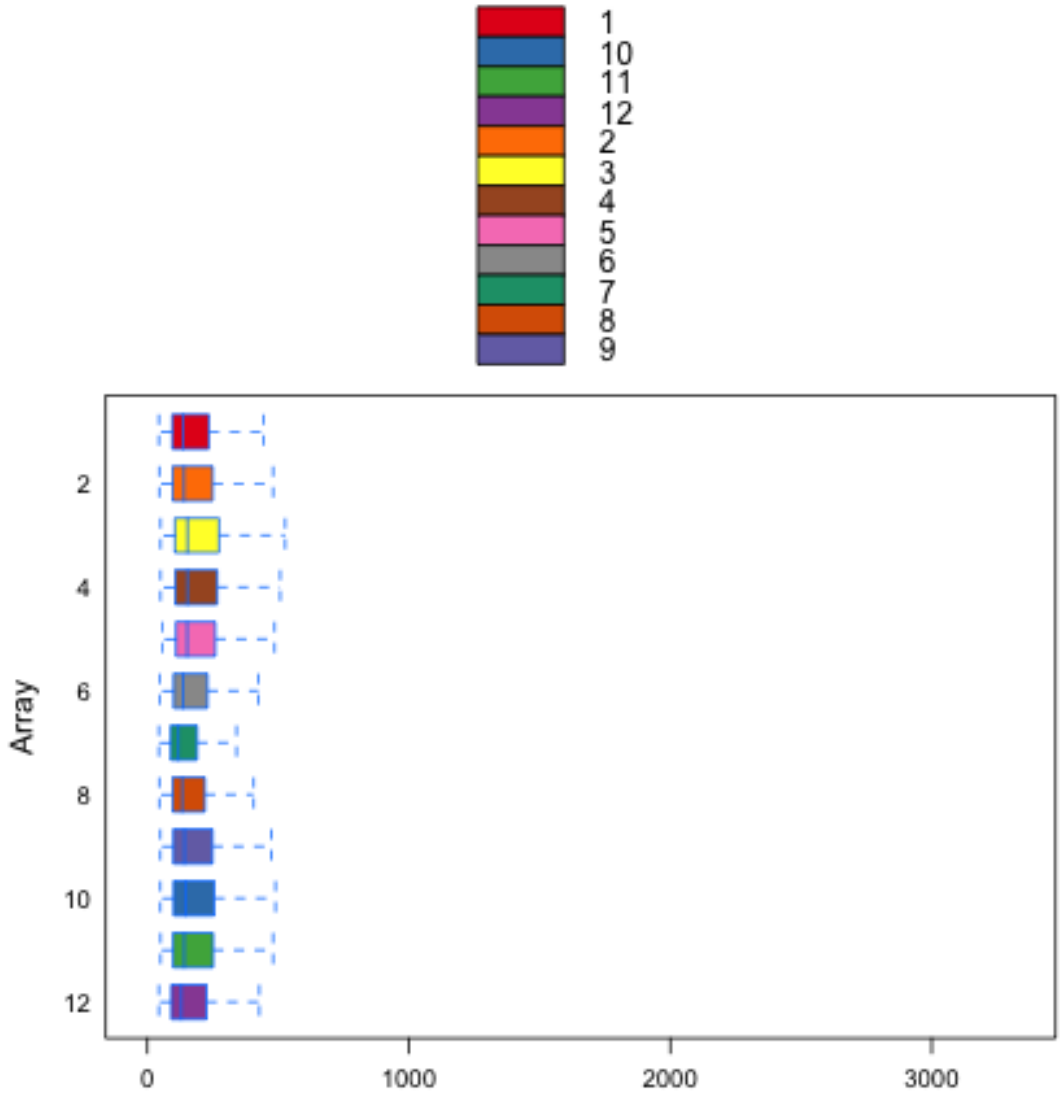
Contiuamos con un heatmap de las distancias entre arrays (Fig.3).



Heatmap de las distancias entre arrays

El gráfico nos muestra que el array número 7 está bastante distanciado de los otros 11. Por lo que podría tratarse de un array defectuoso.

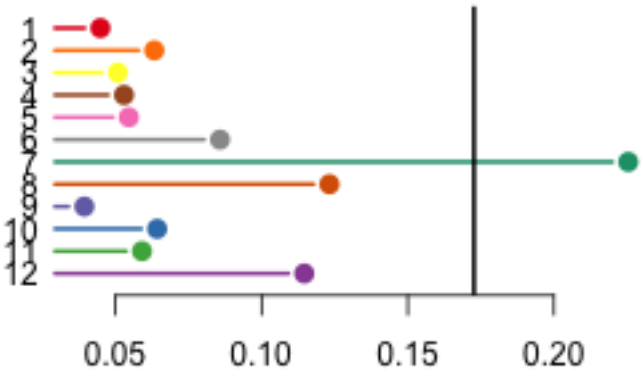
Obtenemos también un boxplot que muestra la distribución de los datos de cada array (Fig.4).



Boxplot de los valores de los arrays

Podemos ver también en este gráfico que el array 7 tiene unos valores que se alejan ligeramente de los del resto. Pero no es suficiente para que con este criterio sea considerado un outlier.

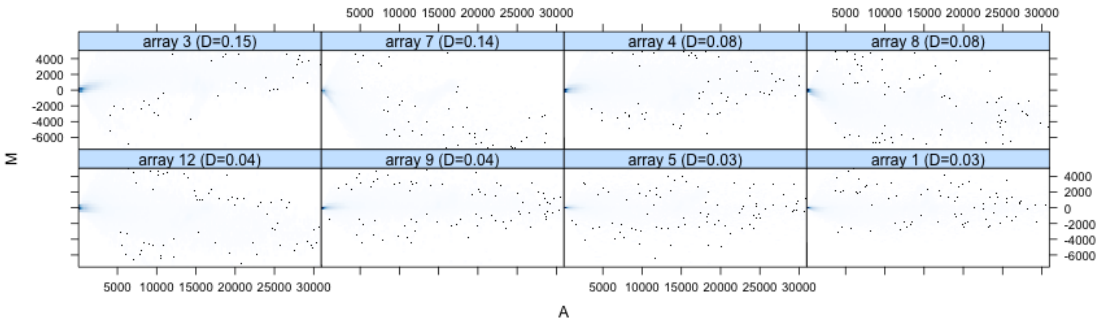
El siguiente gráfico de control de calidad es el de la detección de outliers utilizando la expresión relativa en escala logarítmica (RLE) (Fig. 5).



Detección de outliers mediante RLE

Este gráfico nos muestra los valores del estadístico de Kolmogorov-Smirnov para los valores de la expresión relativa en escala logarítmica. Se ha calculado un umbral para los datos de este experimento Ra = 0,173 (línea vertical negra), que el array 7 supera. Pudiendo de este modo ser considerado un array defectuoso. También podemos observar que los valores de los arrays 8 y 12 son bastante mayores que los del resto de arrays, pero sin llegar a cruzar la línea que los marcaría como posibles arrays defectuosos.

Un gráfico muy informativo en los datos de microarrays son los MA plots (Fig. 6).



MA plot

En este caso se nos muestran dos informaciones. En primer lugar los valores deberían estar distribuidos alrededor del eje M=0. Lo que sugiere que un procedimiento de normalización no le vendría mal a nuestros datos. Continuando con las observaciones, no podemos inferir que exista ningún artefacto causado por el background en los datos de los 8 arrays que se nos muestran. La segunda información de que nos muestra el gráfico es el valor numérico del estadístico Da. Siendo este valor mayor en el array 3 que en el 7. Además ninguno de los arrays superó el umbral de Da>0,15 que es la frontera para marcarlo como posible array defectuoso.

4.2 Control de calidad de los datos normalizados

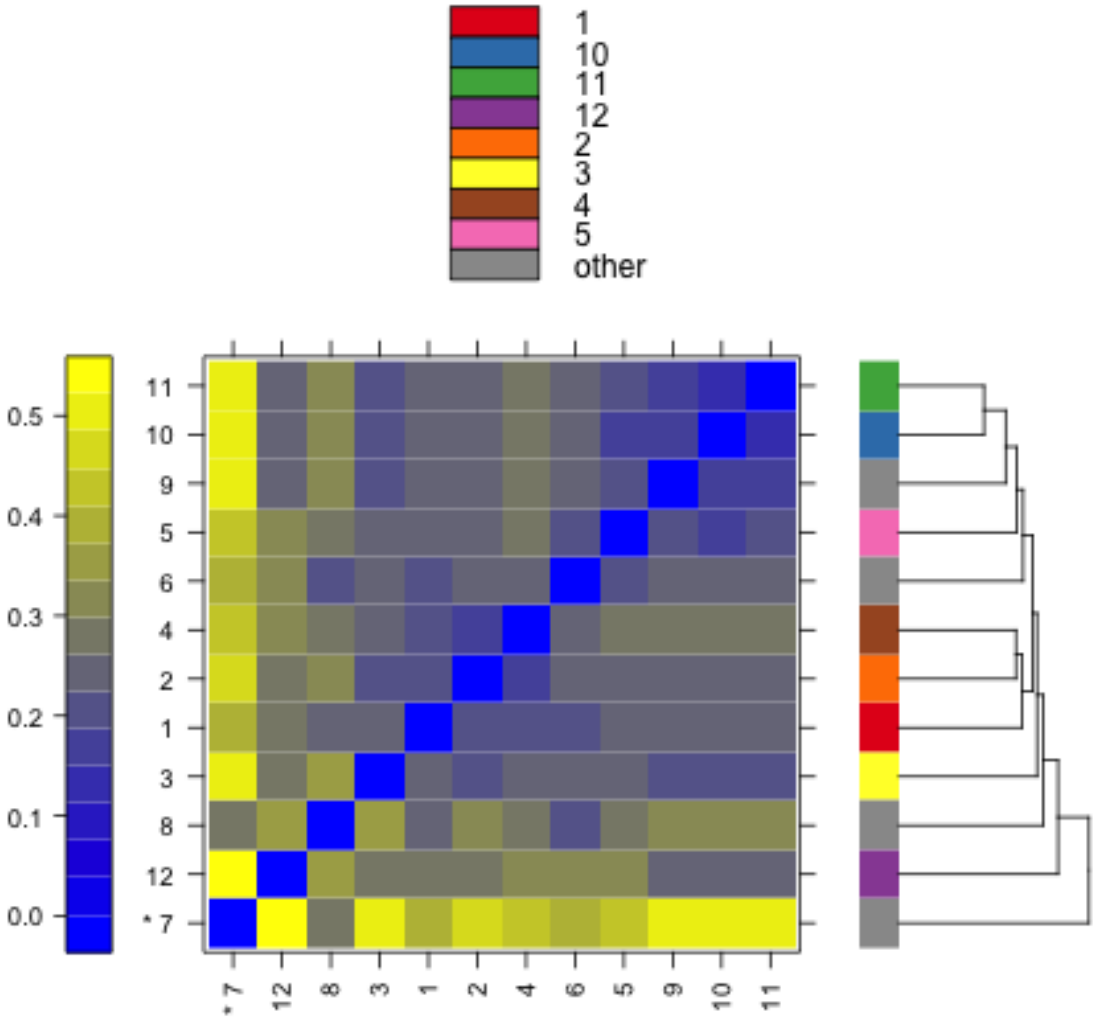
El reporte de arrayQualityMetrics comienza de nuevo con la tabla resumen de arrayQualityMetrics (Fig. 7).

	array	sampleNames	*1	*2	*3	sample	ScanDate
☐	1	GSM544229.CEL.gz				1	04/11/07 16:13:37
☐	2	GSM544230.CEL.gz				2	04/11/07 16:26:51
☐	3	GSM544231.CEL.gz				3	04/12/07 11:56:42
☐	4	GSM544232.CEL.gz				4	04/12/07 12:08:08
☐	5	GSM544233.CEL.gz				5	04/11/07 16:38:41
☐	6	GSM544234.CEL.gz				6	04/12/07 12:20:11
☐	7	GSM544235.CEL.gz	x	x		7	04/12/07 12:31:30
☐	8	GSM544236.CEL.gz				8	04/12/07 15:03:18
☐	9	GSM544237.CEL.gz				9	04/11/07 16:53:02
☐	10	GSM544238.CEL.gz				10	04/12/07 15:16:08
☐	11	GSM544239.CEL.gz				11	04/12/07 15:29:20
☐	12	GSM544240.CEL.gz				12	04/12/07 15:40:44

Heatmap de las distancias entre arrays

Nuevamente el array número 7 es el marcado como diferente, en este caso por 2 de las tres métricas.

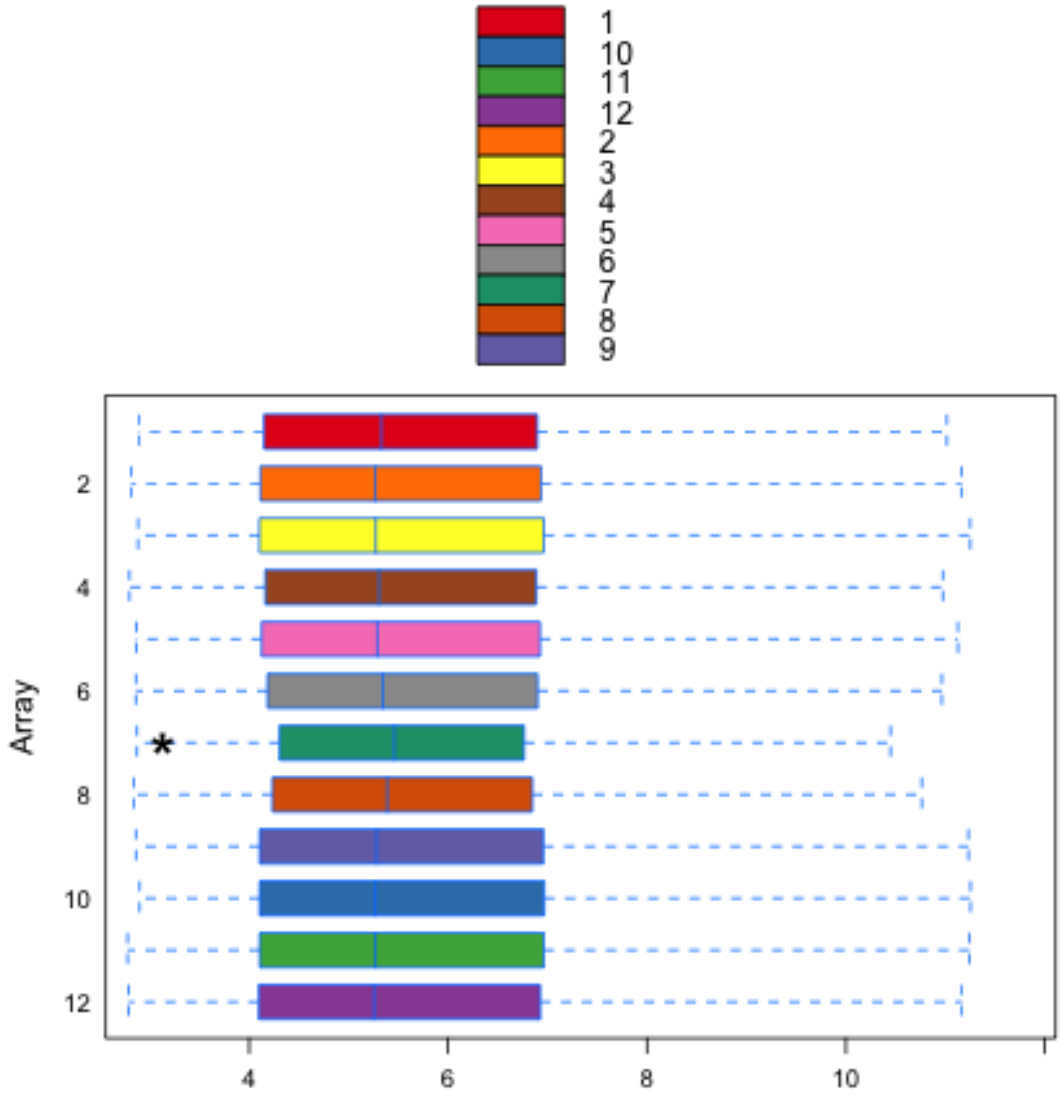
Vamos a observar ahora el heatmap de los datos normalizados (Fig. 8).



Heatmap de las distancias entre arrays

El array 7 claramente es el más diferente al resto.

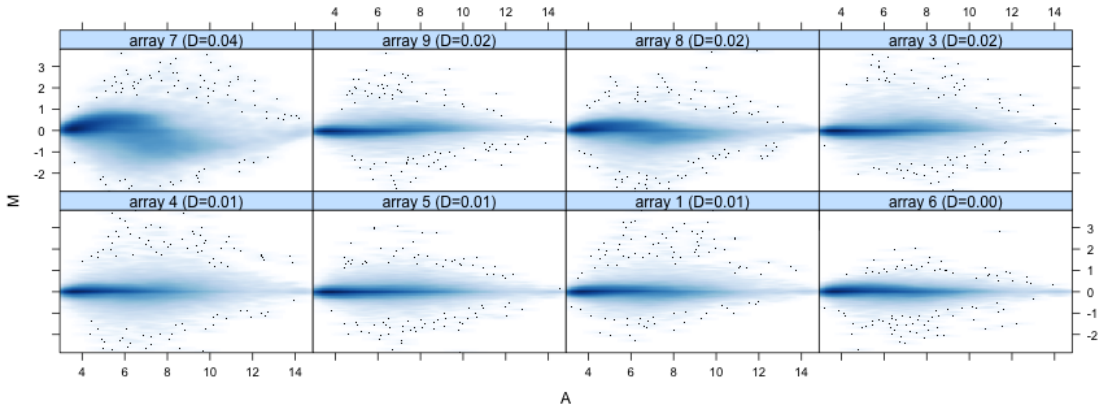
El siguiente gráfico es el boxplot (Fig. 9).



Boxplot de los arrays normalizados

En el boxplot hecho con los datos sin normalizar, el array 7 no aparecía marcado como un outlier. Pero tras la normalización, la distribución de los datos de este array no supera esta prueba.

Para finalizar con los gráficos de datos normalizados, vamos a ver el MA plot (Fig. 10).



MA plot de los arrays normalizados

El array número 7 muestra tanto la gráfica más dispersa como el valor del estadístico Da más elevado. Pero no es considerado un array defectuoso con este criterio.

4.3 Filtrado de los genes menos variables

A continuación mostramos el resultado del filtrado de los genes menos variables.

```
## $numDupsRemoved
## [1] 21738
##
## $numLowVar
## [1] 15130
##
## $numRemoved.ENTREZID
## [1] 12753
##
## $feature.exclude
## [1] 10
```

Vemos aquí desglosado por categorías el número de genes filtrados.

4.4 Identificación de genes diferencialmente expresados

Ahora que ya hemos reducido el número de genes con el fin de ganar potencia, procedemos a identificar los genes cuya expresión ha variado más en cada una de las comparaciones. De cada una de ellas mostramos los 6 cuyo p-valor es más pequeño (Figs. 11, 12 y 13).

```
##               logFC  AveExpr         t      P.Value    adj.P.Val
B
## 213920_at 2.858124 5.349648 13.819634 6.509913e-10 3.283600e-06
12.554250
## 212789_at 3.683034 8.858018 12.086215 4.125320e-09 1.040406e-05
10.980721
```

```
## 227725_at 3.572040 5.464033 11.002343 1.468297e-08 2.359284e-05
9.858653
## 221854_at 1.938753 7.522012 10.804614 1.870963e-08 2.359284e-05
9.641134
## 223529_at 3.876587 7.290373 10.574891 2.490397e-08 2.512312e-05
9.383163
## 226726_at 2.541581 8.768476  9.047485 1.900992e-07 1.598101e-04
7.513481

##               logFC  AveExpr         t      P.Value    adj.P.Val
B
## 228731_at -6.004366 5.410900 -17.23465 2.893897e-11 8.180413e-08
15.31532
## 213920_at  3.535993 5.349648  17.09727 3.243622e-11 8.180413e-08
15.22509
## 221854_at  2.517308 7.522012  14.02888 5.280512e-10 8.878301e-07
12.91757
## 227725_at  4.234467 5.464033  13.04270 1.451398e-09 1.830212e-06
12.03810
## 233220_at -3.183225 6.061729 -12.48798 2.638514e-09 2.661733e-06
11.50876
## 212789_at  3.642750 8.858018  11.95402 4.791729e-09 4.028247e-06
10.97380

##                 logFC  AveExpr          t      P.Value    adj.P.Val
B
## 228731_at   -5.832997 5.410900 -16.742763 4.370933e-11 2.204699e-07
13.958167
## 233220_at   -2.885126 6.061729 -11.318521 1.003687e-08 2.531299e-05
9.871098
## 223529_at   -3.316201 7.290373  -9.046220 1.904390e-07 3.201915e-04
7.349968
## 1560469_at  -2.114055 4.440994  -8.639499 3.411947e-07 4.302465e-04
6.830794
## 220180_at   -2.140047 5.986990  -8.285953 5.753191e-07 5.202354e-04
6.360890
## 201744_s_at -3.827316 5.101076  -8.237401 6.188367e-07 5.202354e-04
6.294972
```

4.5 Comparación entre distintas comparaciones

Para tener una visión más global, es interesante ver cuál es el total de genes que han cambiado en cada comparación. Los mostramos a continuación (Fig. 14).

```
##        AD - ND AD - CR ND - CR
## Down        75     249     422
## NotSig    4600    4633    4581
## Up         369     162      41
```

También es posible mostrarlos como diagrama de Venn para así ver cuáles son comunes a varias comparaciones (Fig. 15).

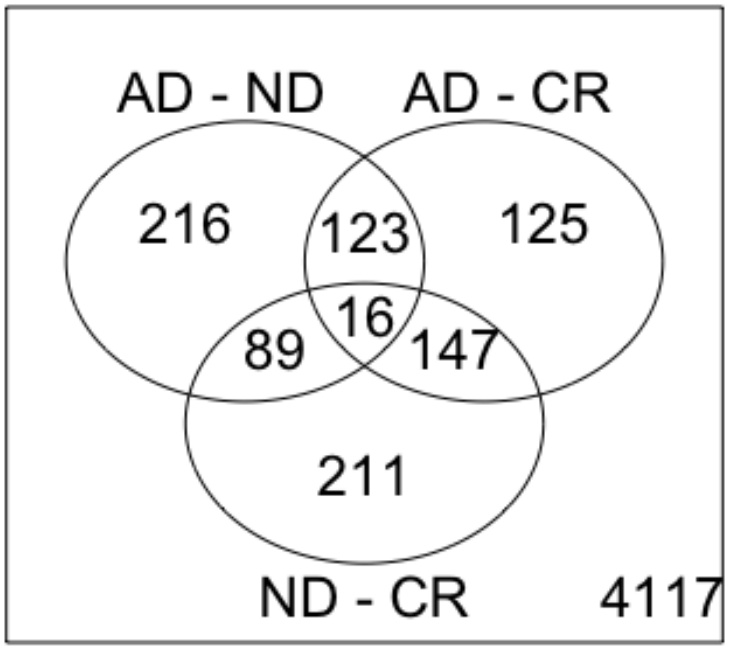
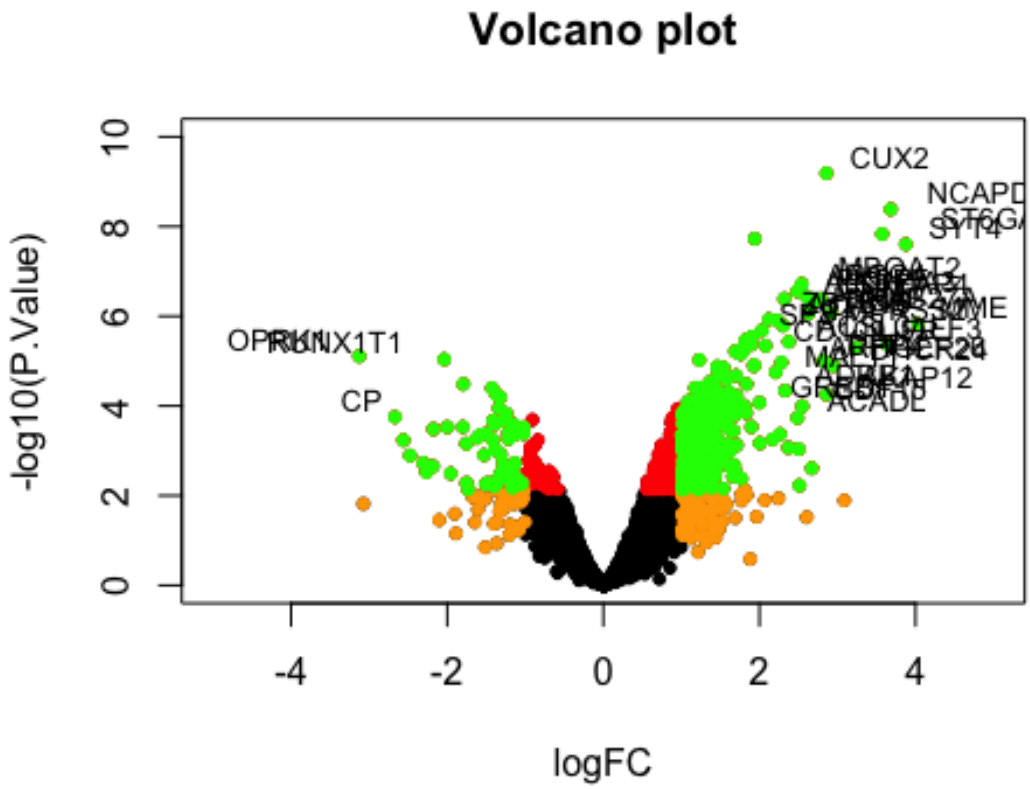
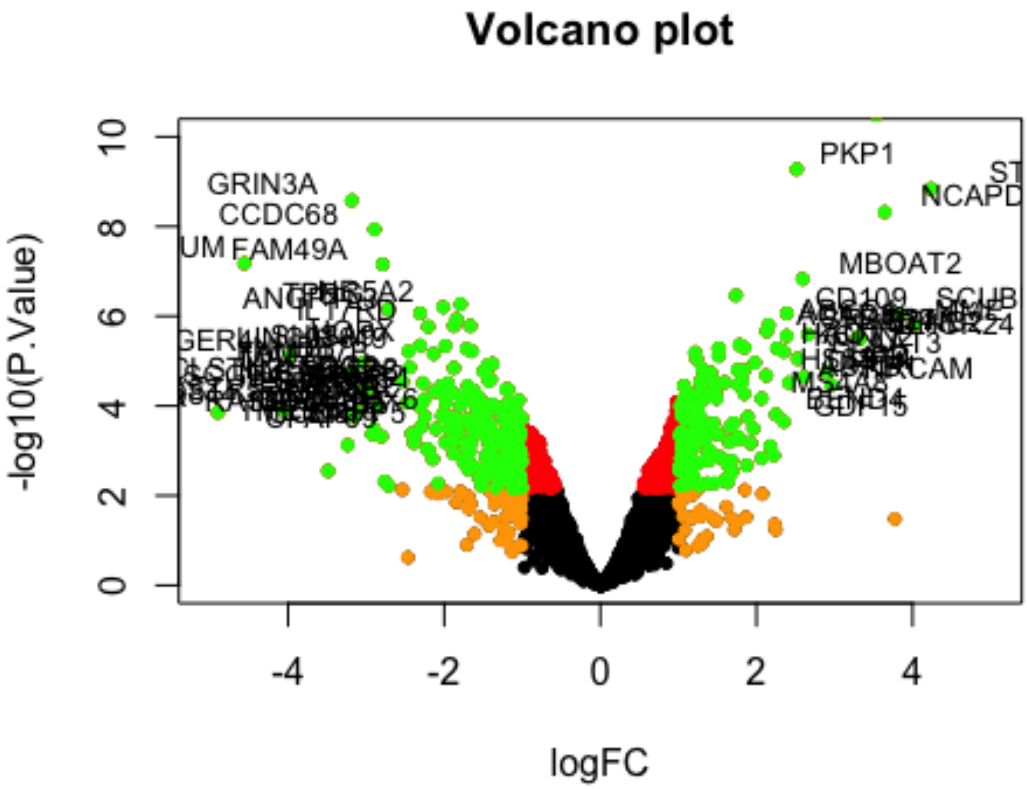


Diagrama de Venn del número de genes diferencialmente expresados en cada comparación, también se muestran aquellos que coinciden entre una y otra

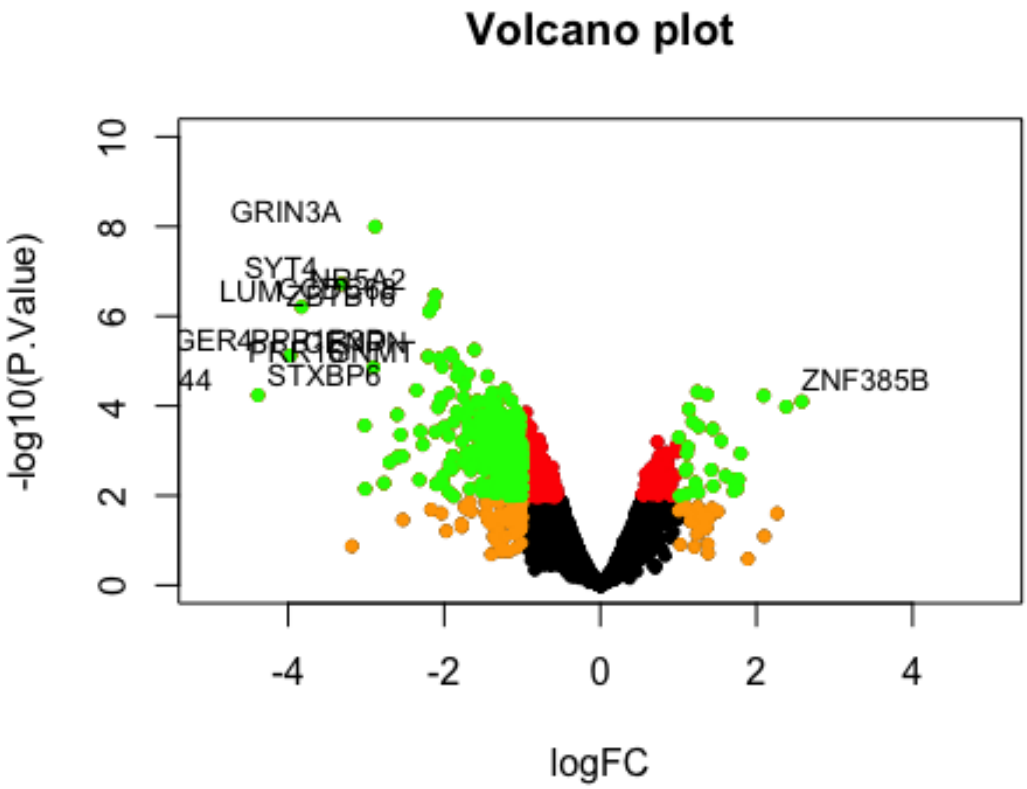
Una manera muy visual de ver los datos expresión diferencial de microarrays son los volcano plots. Aquí vemos uno por cada comparación realizada. Para que la información sea más visual, se han categorizado según su p-valor y log2FC. Además, los más variables aparecen con su nombre (Figs. 16, 17 y 18).



Volcano plot de la comparación 1



Volcano plot de la comparación 2



Volcano plot de la comparación 3

4.6 Análisis de la significación biológica

Para finalizar, realizaremos un gráfico que muestra las redes bioquímicas enriquecidas según los resultados de este estudio (Fig. 19).

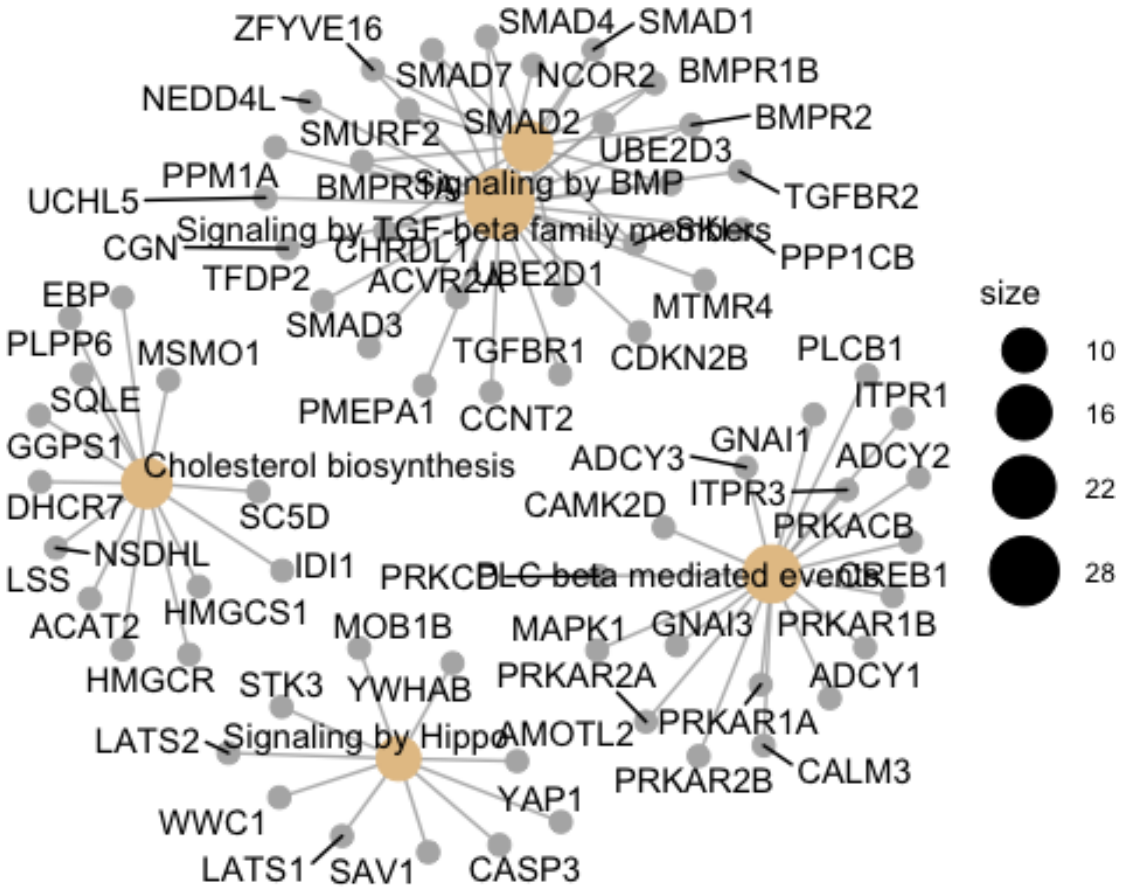


Figura 16. Redes de rutas enriquecidas en este estudio

5 Discusión

El número de muestras de nuestro estudio es correcto según **web**, ya que es el número mínimo para obtener un poder de 0.8 con un nivel de significación de 0,05 y una diferencia de medias de 2.

Con respecto al array número 7, que algunos indicadores para detectar muestras erróneas daban como outlier, se decidió no sacarlo porque por una parte el número de muestras sería inferior al mínimo calculado para tener poder estadístico. Y por otra parte, en este estudio se buscaba diferencias en la expresión, y al eliminarlo estaríamos perdiendo fuentes de variabilidad. Precisamente para facilitar la variabilidad en las muestras, a costa de inroducir algunos resultados erróneos tampoco se realizó filtraje no específico.

Debido a todo lo indicado previamente, en siguientes fases se podría hacer RT-qPCR para afinar los resultados de nuestro estudio.

6 Apéndice con el código de R utilizado en el estudio

```
#Carga de los archivos CEL
rawData <- ReadAffy(celfile.path =
"/Volumes/Coisas/Javier/Master/2020/2/Análisis_datos_omicos/PEC1/Data/GSE
21887_RAW")
#arrayqualitymetrics de los datos crudos
arrayQualityMetrics(rawData,
                    outdir =
"/Volumes/Coisas/Javier/Master/2020/2/Análisis_datos_omicos/PEC1/Results/
QCrawData",
                    intgroup = as.vector(colnames(rawData@phenoData)),
                    force = T, reporttitle = "Informe_calidad_rawData")

## The report will be written into directory
'/Volumes/Coisas/Javier/Master/2020/2/Análisis_datos_omicos/PEC1/Results/
QCrawData'.

## Warning in maximumLevels(fac, n = length(colors)): A factor was
provided with 12 levels, but the colour map used here has only 9 colours.
For the purpose of colouring, levels 9 ('6') to 12 ('9') are being
collapsed. Please consider grouping together some of the levels of your
factor of interest to reduce the number of levels, this might improve the
legibility of the plots.

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
```

```
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
```

```
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
```

```
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'

## Warning in KernSmooth::bkde2D(x, gridsize = nbin, bandwidth =
bandwidth):
## Binning grid too coarse for current (small) bandwidth: consider
increasing
## 'gridsize'
```

```
#RMA
eset <- affy::rma(rawData)
#arrayqualitymetrics de los datos normalizados
arrayQualityMetrics(eset,
                    outdir =
"/Volumes/Coisas/Javier/Master/2020/2/Análisis_datos_omicos/PEC1/Results/
QCeset",
                    intgroup = as.vector(colnames(eset@phenoData)),
                    force = T, reporttitle = "Informe_calidad_eset")
```

```
## The report will be written into directory
'/Volumes/Coisas/Javier/Master/2020/2/Análisis_datos_omicos/PEC1/Results/
QCeset'.

## Warning in maximumLevels(fac, n = length(colors)): A factor was
provided with 12 levels, but the colour map used here has only 9 colours.
For the purpose of colouring, levels 9 ('6') to 12 ('9') are being
collapsed. Please consider grouping together some of the levels of your
factor of interest to reduce the number of levels, this might improve the
legibility of the plots.

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
```

```
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

## Warning in svgStyleAttributes(style, svgdev): Removing non-SVG style
attribute
## name(s): subscripts, group.number, group.value

#base de datos correspondiente a nuestro array
annotation(eset) <- "hgu133plus2.db"
#Filtrado de genes
filtrado <- nsFilter(eset, require.entrez = T, remove.dupEntrez = T,
var.filter = T, var.func = IQR, var.cutoff = 0.75, filterByQuantile = T,
feature.exclude = "^AFFX")
#Nuestro nuevo expressionset con los datos filtrados
eset_f <- filtrado$eset
#Creación de la matriz de diseño
diseño <- model.matrix(~ 0+factor(c(1,1,1,1,2,2,2,2,3,3,3,3)))
```

```r
rownames(diseño) <- as.vector(rownames(eset_f@phenoData))
colnames(diseño) <- c("AD", "ND", "CR")
#Ajuste del modelo lineal a cada gen dada una matriz de diseño
fit <- lmFit(object = eset_f, design = diseño)
#Creación de la matriz de contraste
contraste <- limma::makeContrasts(AD-ND,AD-CR, ND-CR, levels= diseño)
#Computamos los coeficientes estimados y los errores estándar para un
grupo de contrastes
fit2 <- contrasts.fit(fit, contraste)
#Calculamos estadísticos de Bayes para la expresión diferencial
fit2 <- eBayes(fit2)
#Genes más variables por cada comparación
top_genes1 <- topTable(fit2, number=nrow(fit2), coef = 1 , adjust="BH")
top_genes2 <- topTable(fit2, number=nrow(fit2), coef = 2 , adjust="BH")
top_genes3 <- topTable(fit2, number=nrow(fit2), coef = 3 , adjust="BH")
#Hacemos test múltiples entre genes y contrastes y filtramos los
resultados
resultados<-decideTests(fit2, method="separate", adjust.method="BH",
p.value=0.1, lfc=1)
sumabs<-apply(abs(resultados),1,sum)
rescero<-resultados[sumabs!=0,]
#Diagrama de venn de los resultados
vennDiagram(resultados)
#Función para anotar los probesets
annotatedTopTable <- function(topTab, anotPackage)
{
  topTab <- cbind(PROBEID=rownames(topTab), topTab)
  myProbes <- rownames(topTab)
  thePackage <- eval(parse(text = anotPackage))
  geneAnots <- AnnotationDbi::select(thePackage, myProbes, c("SYMBOL",
"ENTREZID", "GENENAME"))
  annotatedTopTab<- merge(x=geneAnots, y=topTab, by.x="PROBEID",
by.y="PROBEID")
  return(annotatedTopTab)
}
#Utilizamos la función para anotar nuestros resultados
anotado1 <- annotatedTopTable(topTab = top_genes1,
anotPackage="hgu133plus2.db")

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

anotado2 <- annotatedTopTable(topTab = top_genes2,
anotPackage="hgu133plus2.db")

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

anotado3 <- annotatedTopTable(topTab = top_genes3,
anotPackage="hgu133plus2.db")

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

```r
#Preparamos los genes que van a ser analizados
listOfTables <- list(ADvsND = top_genes1,
                     ADvsCR  = top_genes2,
                     NDvsCR = top_genes3)
listOfSelected <- list()
for (i in 1:length(listOfTables)){
  topTab <- listOfTables[[i]]
  whichGenes<-topTab["adj.P.Val"]<0.15
  selectedIDs <- rownames(topTab)[whichGenes]
  EntrezIDs<- AnnotationDbi::select(hgu133plus2.db, selectedIDs,
c("ENTREZID"))
  EntrezIDs <- EntrezIDs$ENTREZID
  listOfSelected[[i]] <- EntrezIDs
  names(listOfSelected)[i] <- names(listOfTables)[i]
}

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

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

sapply(listOfSelected, length)
#Obtenemos el entrezID de los genes analizados
mapped_genes2GO <- mappedkeys(org.Hs.egGO)
mapped_genes2KEGG <- mappedkeys(org.Hs.egPATH)
mapped_genes <- union(mapped_genes2GO , mapped_genes2KEGG)
#Con los datos anteriores realizamos el análisis de significación
biológica
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)

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

  if (length(rownames(enrich.result@result)) != 0) {
  write.csv(as.data.frame(enrich.result),

file=paste0("./results/","ReactomePA.Results.",comparison,".csv"),
```

```
row.names = FALSE)

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

  pdf(file =
paste0("./results/","ReactomePAcnetplot.",comparison,".pdf"))
    print(cnetplot(enrich.result, categorySize = "geneNum", schowCategory
= 15, vertex.label.cex = 0.75))
  dev.off()
  }
}
#Gráfico de redes enriquecidas
cnetplot(enrich.result, categorySize = "geneNum", schowCategory = 15,
vertex.label.cex = 0.75)
#Código para el volcano plot. El código es el mismo en cada caso, lo que
cambia es el dataset.
attach(anotado1)
with(anotado1, plot(logFC, -log10(P.Value), pch=20, main="Volcano plot",
xlim=c(-5,5), ylim = c(0,10)))
with(subset(anotado1, adj.P.Val<.05 ), points(logFC, -log10(P.Value),
pch=20, col="red"))
with(subset(anotado1, abs(logFC)>1), points(logFC, -log10(P.Value),
pch=20, col="orange"))
with(subset(anotado1, adj.P.Val<.05 & abs(logFC)>1), points(logFC, -
log10(P.Value), pch=20, col="green"))
with(subset(anotado1, adj.P.Val<.01 & abs(logFC)>2), textxy(logFC, -
log10(P.Value), labs=SYMBOL, cex=.8))
detach(anotado1)
```

Bibliografía

Terada, Naoki et al. 2010. "Identification of Ep4 as a Potential Target for the Treatment of Castration-Resistant Prostate Cancer Using a Novel Xenograft Model." Cancer Research.