

Análisis de datos Ómicos - Primera prueba de evaluación continua

Maria P. Plaza

16/04/2020

Análisis de Microarrays

0. Abstract

En el experimento analizado se investigaron los patrones de expresión génica relacionados con el sistema inmune en los tejidos del hígado que rodean los tejidos de carcinoma hepatocelular (HCC) en etapa temprana y los agentes quimiopreventivos que podrían alterar estos patrones para prevenir la tumorigénesis hepática. Con ayuda de un análisis de microarrays tratamos de crear clasificaciones, establecer asociaciones con la línea celular de origen y ayudar en el pronóstico, identificando los genes que determinan esa posible clasificación.

El artículo se encuentra en el siguiente enlace: <https://www.ncbi.nlm.nih.gov/pubmed/31344396>

1. Introducción

La cirrosis y la inflamación crónica preceden el desarrollo del carcinoma hepatocelular (CHC) en aproximadamente el 80% de los casos. Como se ha indicado, se estudian los patrones de expresión génica relacionados con el sistema inmune en los tejidos del hígado que rodean los HCC en etapa temprana y los agentes quimiopreventivos que podrían alterar estos patrones para prevenir la tumorigénesis hepática. Para ello, se recogieron tejidos hepáticos recién congelados de 22 ratones (3 de control, 9 tratados con vehículo y 10 tratados con nintedanib), y se extrajo el ARN. Los ratones C57BL/6J recibieron una inyección única de N-nitrosodietilamina seguida de dosis semanales de tetracloruro de carbono para inducir fibrosis hepática y tumorigénesis. A los ratones se les dio por vía oral el inhibidor múltiple de tirosina nintedanib o vehículo (controles); Se recolectaron tejidos hepáticos y se realizaron análisis de histología, transcriptoma y proteínas. También analizaron transcriptomas de tejidos hepáticos recolectados de ratones en una dieta alta en grasas con deficiencia de colina, que desarrollaron inflamación hepática crónica y tumores, se les administró aspirina y clopidogrel por vía oral o el agente antiinflamatorio sulindac frente a ratones en una dieta de alimentación (control).

2. Objetivos

El objetivo principal que se persigue con el análisis de expresión genética en tumores de ratón es estimar el posible error de clasificación mediante expresión genética de nuevos casos. Trataremos por tanto de responder a la cuestión de encontrar genes diferencialmente expresados entre dos o más condiciones o a lo largo del tiempo y descubrir patrones de expresión característicos.

Los objetivos específicos que se persiguen nos llevan a cubrir un análisis completo de microarrays de expresión genética:

- Control de calidad de microarrays de expresión genética
- Pre procesado de microarrays
- Normalización
- Control de calidad de datos pre-procesados
- Selección de genes diferenciados por expresión
- Clasificación y evaluación de tejidos tumorales por firmas genéticas

3. Materiales y Métodos

3. 1. Obtención y lectura de datos

Vamos a utilizar los datos libremente disponibles en GEO de un estudio del transcriptoma de *Mus musculus*. Se puede acceder a los datos desde el siguiente link: <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE125975>

El articulo escogido se denomina: *An Immune Gene Expression Signature Associated With Development of Human Hepatocellular Carcinoma Identifies Mice That Respond to Chemopreventive Agents*

Base de datos

Se tienen las muestras de ARN de tejidos hepáticos de 22 ratones (3 de control, 9 tratados con vehículo y 10 tratados con nintedanib). El análisis del transcriptoma del genoma completo se obtuvo usando un Chip HT MG-430 (Affymetrix) de acuerdo con los protocolos del fabricante. [CDF: HTMG430PM_Mm_ENTREZG, Brainarray version 19]

La cantidad total de genes que contiene la base de datos es de 13392 genes en 22 muestras.

Usando placas de microarray de affymetrix HTMG430PM se obtuvieron los siguientes ficheros .CEL

Samples (22) GSM3587187 QP-4 Control Vehicle GSM3587188 QP-49 Control Vehicle GSM3587189 QP-50 Control Vehicle GSM3587190 QP-76 Model Vehicle GSM3587191 QP-77 Model Vehicle GSM3587192 QP-78 Model Vehicle GSM3587193 QP-79 Model Vehicle GSM3587194 QP-80 Model Vehicle GSM3587195 QP-81 Model Vehicle GSM3587196 QP-92 Model Vehicle GSM3587197 QP-93 Model Vehicle GSM3587198 QP-94 Model Vehicle GSM3587199 QP-82 Model Nintedanib GSM3587200 QP-83 Model Nintedanib GSM3587201 QP-84 Model Nintedanib GSM3587202 QP-85 Model Nintedanib GSM3587203 QP-86 Model Nintedanib GSM3587204 QP-87 Model Nintedanib GSM3587205 QP-88 Model Nintedanib GSM3587206 QP-89 Model Nintedanib GSM3587207 QP-90 Model Nintedanib GSM3587208 QP-91 Model Nintedanib

Todos los archivos, así como el código R empleado se encuentran en el siguiente repositorio GIT:

https://github.com/mariaplaza/PEC1_DatosOmicos

```
[1] "Genes y Muestras"
```

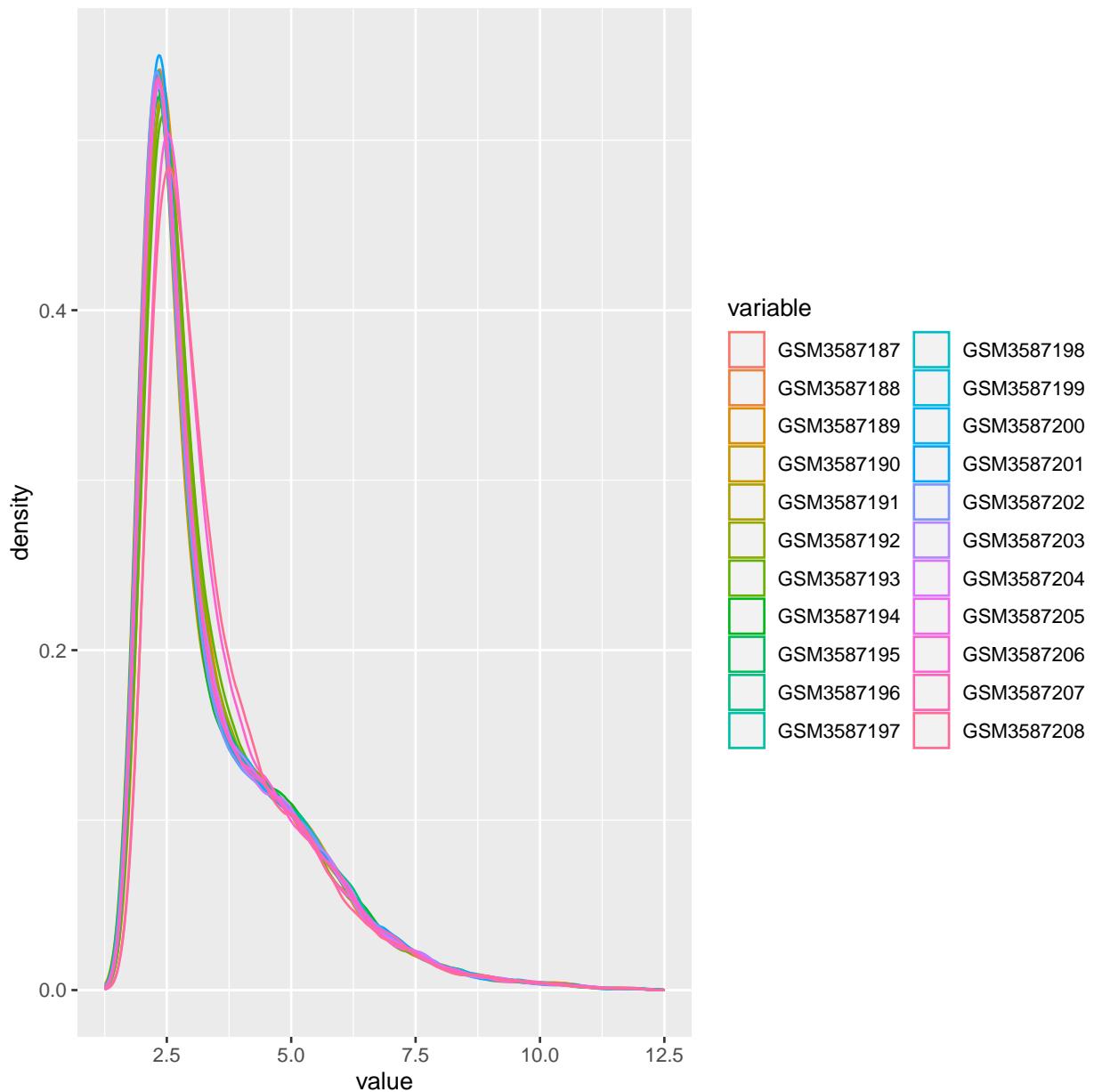
```
[1] 13392      22
```

3. Pasos llevados a cabo.

Exploración

En primer lugar realizamos una exploración de los datos con los que trabajamos. Para ello los representamos en varios tipos de gráficos. Además comprobamos que son comparables entre sí. Podemos analizar la distribución global de la fluorescencia en los distintos microarrays utilizando boxplots (gráficos de cajas) y histogramas usando las funciones *boxplot* y *hist* o *ggplot*

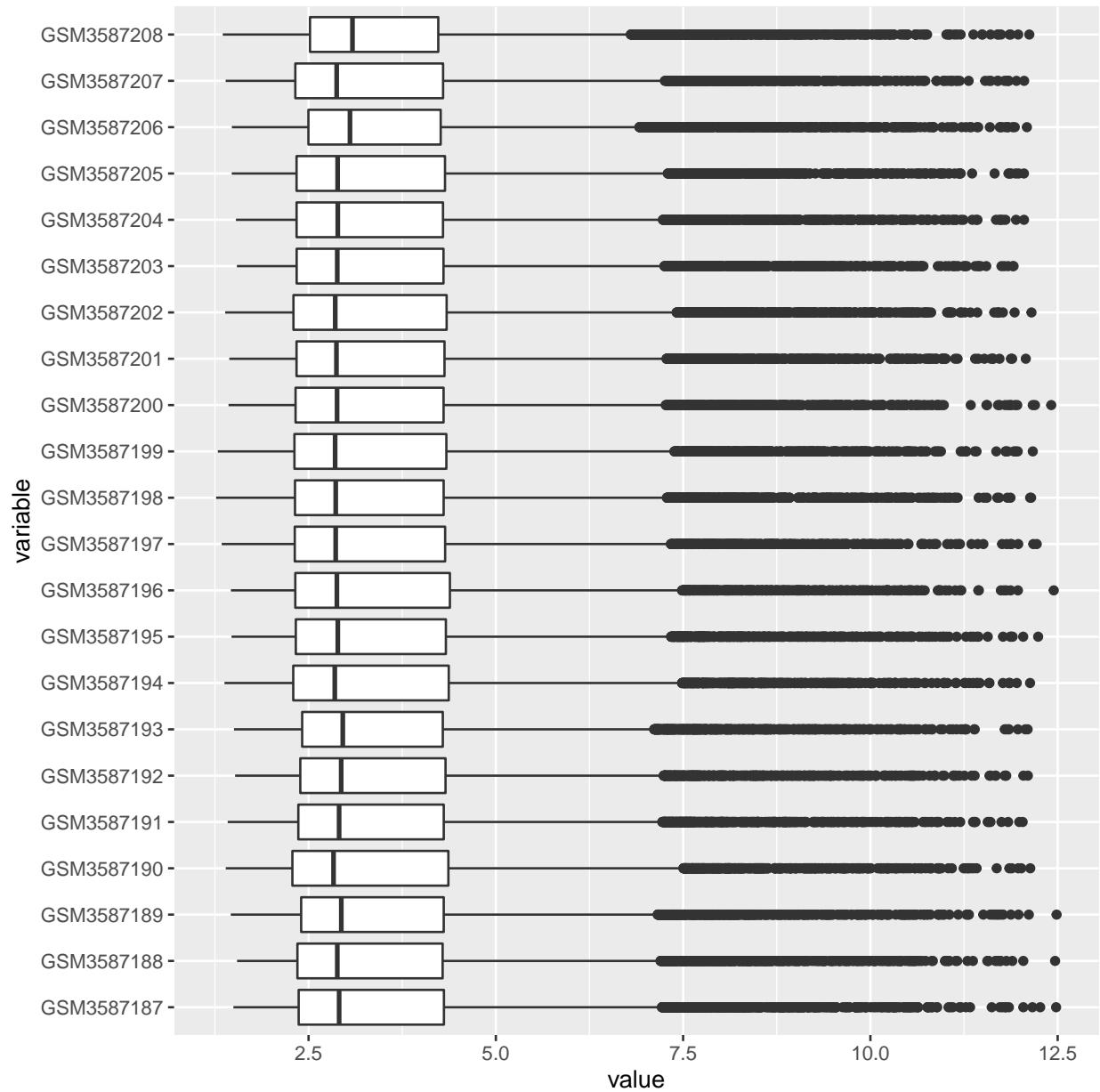
HISTOGRAMA



Según la imagen anterior, algunas muestras discrepan de la media. Además la forma extraña que vemos a la derecha, en el value = 5, puede indicar un error sistemático.

BOXPLOTS

Podemos ademas representar las muestras en diagramas de cajas o Boxplots, mostrando la distribución de los niveles de expresión a lo largo de las muestras. Como vemos, hay algunas que se desvían del comportamiento general:

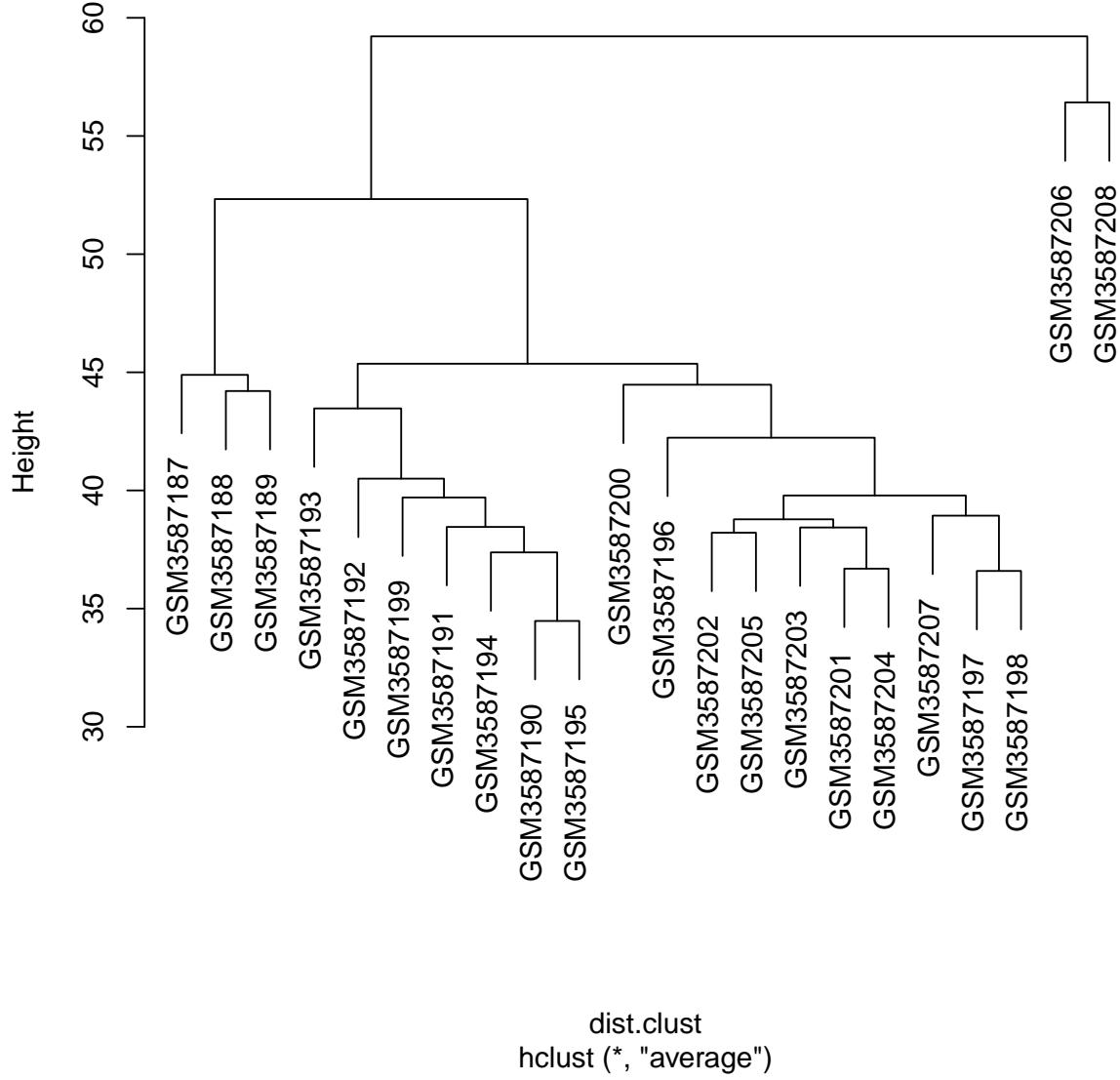


HIERARCHICAL CLUSTERING ANALYSIS

Finalmente un cluster jerárquico seguido de un dendrograma nos puede ayudar a hacernos una idea de si las muestras se agrupan por condiciones experimentales

```
Call:  
hclust(d = dist.clust, method = "average")  
  
Cluster method : average  
Distance : euclidean  
Number of objects: 22
```

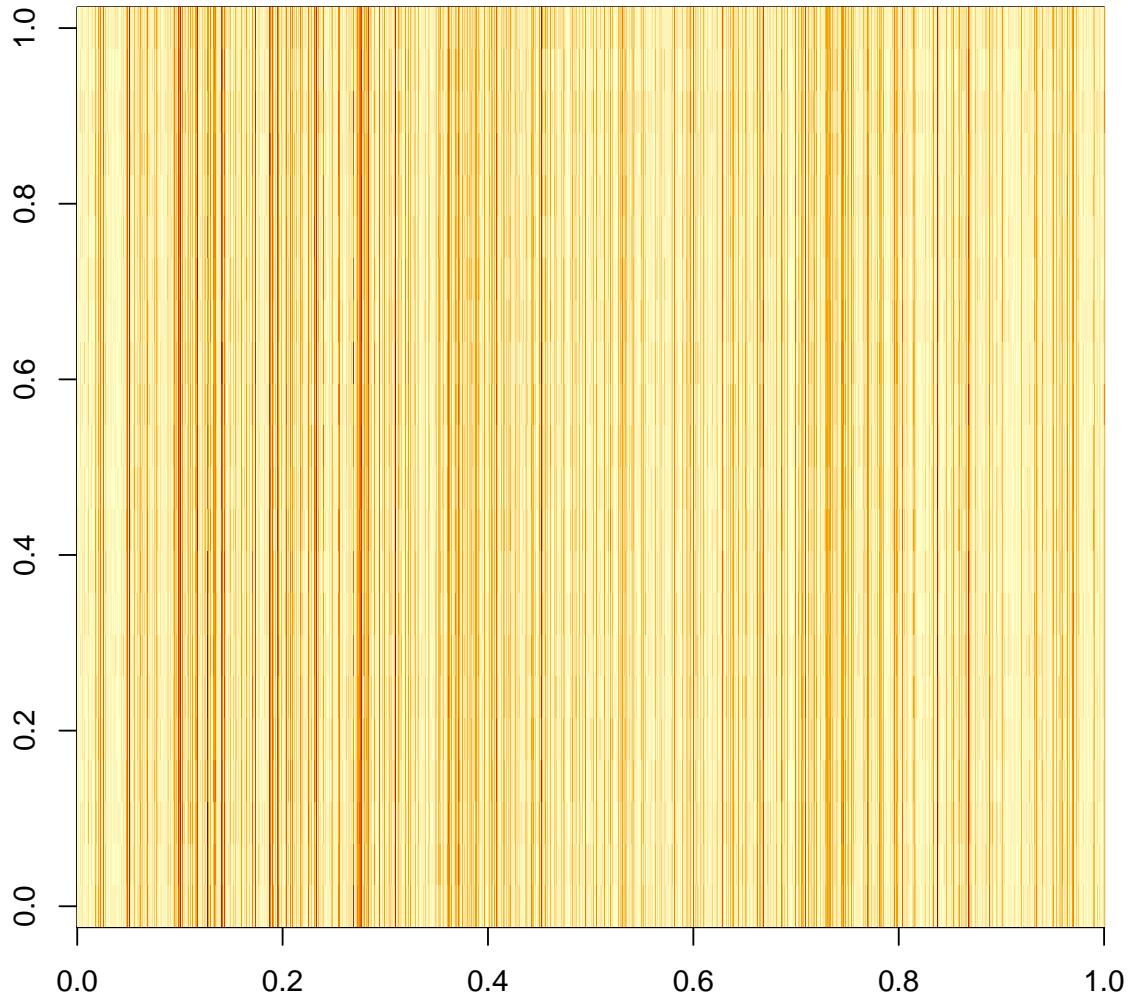
Cluster Dendrogram



Estos gráficos nos han mostrado que:

1. Hay al menos dos valores atípicos en el conjunto de datos
2. La distribución de las intensidades en cada matriz (boxplots) ilustra la necesidad de un paso de normalización.

También es posible visualizar la imagen que representa cada archivo CEL (en este ejemplo, visualizamos solo la primera diapositiva del conjunto de datos). De esta manera, es posible identificar problemas técnicos que ocurren eventualmente solo en una región de la matriz.

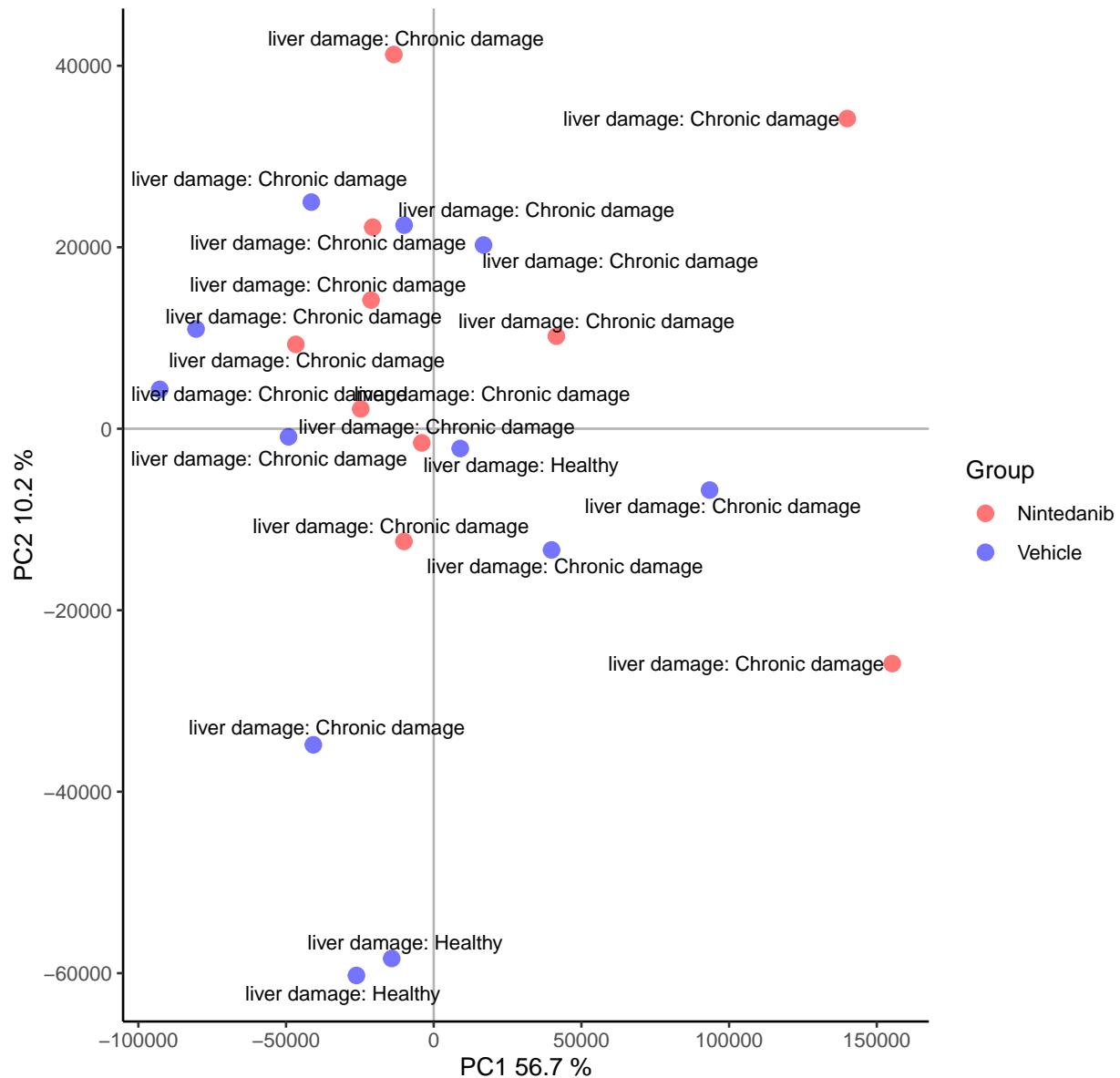


Control de calidad

El siguiente paso consiste en realizar un control de la calidad de los datos de microarrays con los que vamos a trabajar para comprobar que no ha habido problemas de hibridación, degradación del RNA o daño físico en la placa de microarray. La función *arrayQualityMetrics* produce un informe de métricas de calidad de nuestra matriz de datos, que guardamos dentro de nuestra carpeta de resultados:

y vemos los datos en un grafico PCA:

Principal Component Analysis for: Raw data



En el análisis de la calidad de los microarrays de nuestro ejemplo no detectamos problemas graves de calidad. Sin embargo, tanto en el boxplot como en el histograma observamos que la distribución global de los niveles de fluorescencia no es comparable ya que presentan distintos valores medios, mínimos y máximos. Por lo tanto es necesario realizar un preprocesamiento para obtener una estimación de los niveles de expresión de los distintos genes basadas en la fluorescencia de los microarrays.

Para ello, se realiza la corrección de la fluorescencia de fondo para eliminar señal no específica, la normalización entre los distintos microarrays para eliminar diferencias técnicas y una estimación de los niveles de expresión calculando la media de la fluorescencia de las distintas sondas que representan un mismo gen. Un algoritmo que realiza este preprocesamiento es el Robust Multiarray Average (RMA). La función *rma* implementa este algoritmo. Esta función recibe como entrada un objeto con los datos crudos de la lectura de los microarrays y devuelve una estimación en escala log2 de los niveles de expresión.

Normalización

La normalización supone la corrección de dos o más muestras antes de comparar sus valores de expresión. Suele constar de tres pasos: ! Corrección del fondo (background): Estimar y eliminar la intensidad de ruido de fondo ! Normalización global o local: Asegurar que la mayoría de las sondas varíen igual ! Sumarización: Conversión de sondas o conjuntos de sondas a transcritos o genes

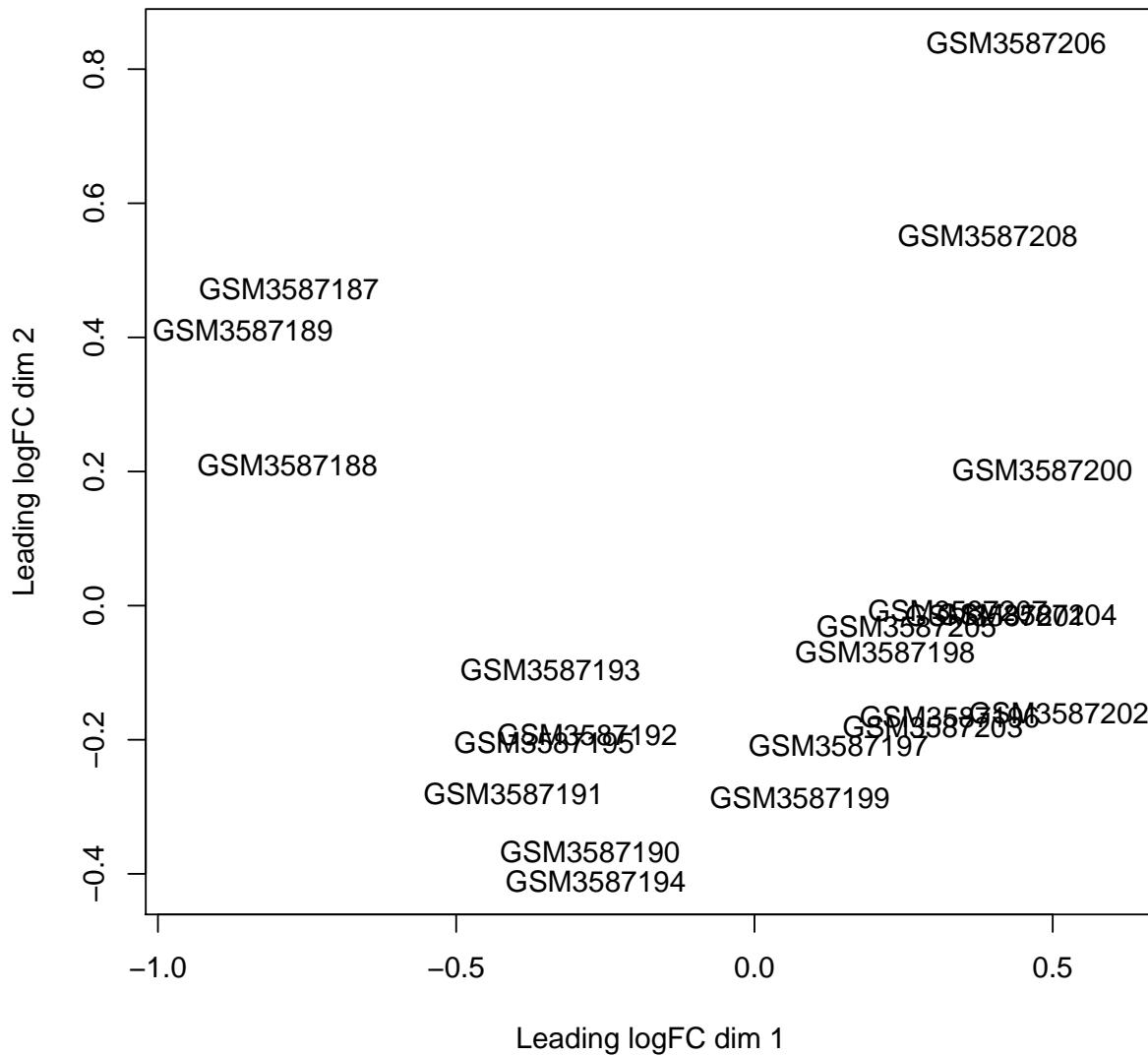
En primer lugar empleamos el Robust Multiarray Analysis: Método para realizar la corrección de fondo, normalización y sumarización en chips de Affymetrix. Tiene una precisión mucho mayor que MAS 5.0 (el método de Affymetrix para preprocesar sus chips). Además: ! Corrección de fondo sin contar MM ! Normalización por cuantiles ! Estimación por median polish

Background correcting

Normalizing

Calculating Expression

Podemos realizar la normalización con otro tipo de funciones, con el paquete *limma* y/o *edgeR*.



GSM3587187	GSM3587188	GSM3587189	GSM3587190	GSM3587191	GSM3587192	GSM3587193
1.0006153	1.0003829	1.0019194	0.9982851	1.0010305	1.0022102	1.0030345
GSM3587194	GSM3587195	GSM3587196	GSM3587197	GSM3587198	GSM3587199	GSM3587200
0.9974079	0.9994986	0.9983949	0.9981566	0.9982010	0.9971785	0.9973989
GSM3587201	GSM3587202	GSM3587203	GSM3587204	GSM3587205	GSM3587206	GSM3587207
1.0014825	0.9981018	1.0003769	0.9999061	0.9987041	1.0032609	0.9992955
GSM3587208						
1.0052075						

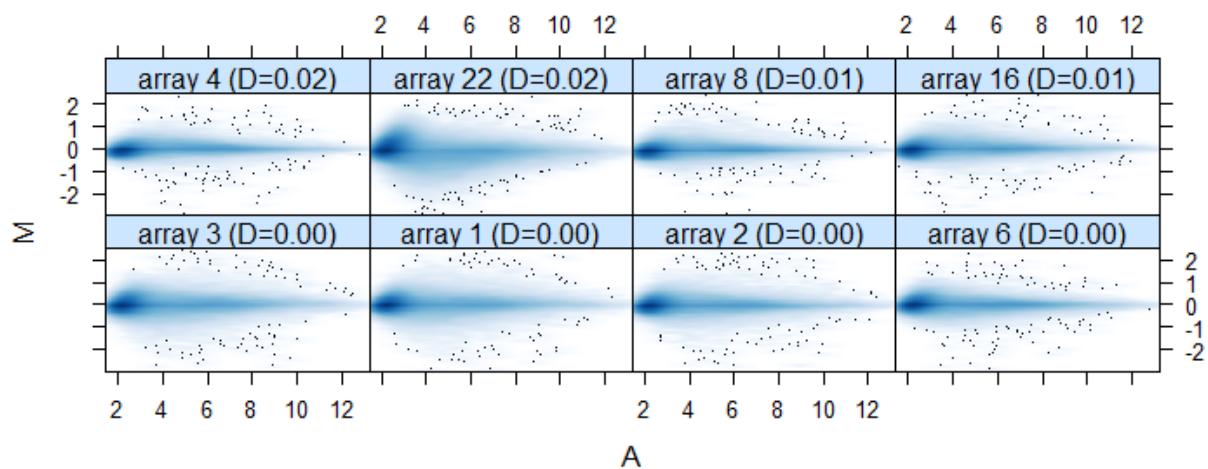
	GSM3587187	GSM3587188	GSM3587189	GSM3587190	GSM3587191	GSM3587192
100009600_at	2.992425	2.566677	2.992425	2.872160	3.249477	3.471569
100012_at	2.034567	2.244631	2.106121	2.264301	2.112395	2.060698
100017_at	5.442850	4.775964	5.301385	5.350172	4.735635	5.478662
100019_at	4.280375	4.023198	4.093205	3.708929	3.658147	4.381778
100034675_at	1.952049	2.370992	2.145451	2.727788	2.247678	2.374035

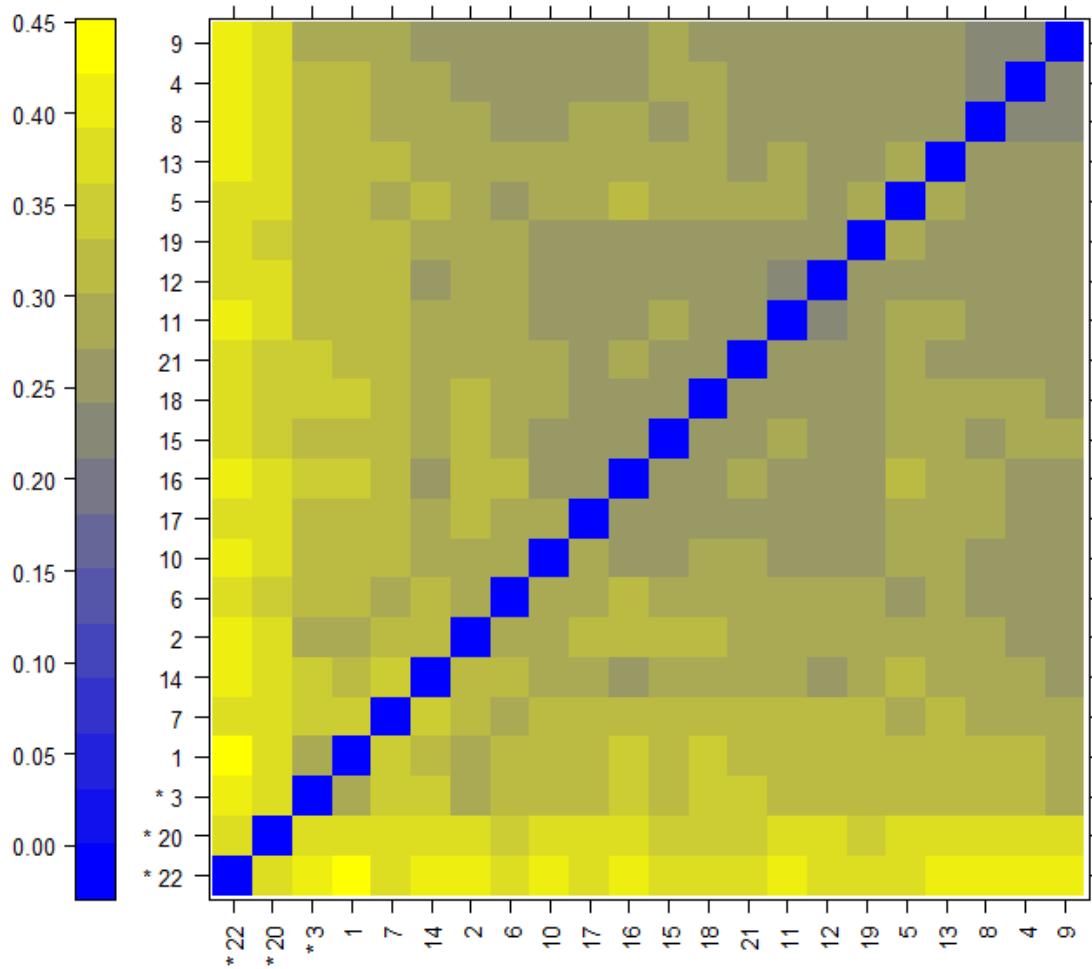
100034739_at	1.969100	1.761193	2.226607	1.828196	1.909139	2.122422
	GSM3587193	GSM3587194	GSM3587195	GSM3587196	GSM3587197	GSM3587198
100009600_at	2.664682	2.422216	2.742301	2.652172	2.637908	2.709667
100012_at	1.943948	1.985288	2.204389	2.204596	2.329772	2.072051
100017_at	4.904846	5.196591	5.367091	4.989992	5.249435	5.180066
100019_at	3.345184	3.799712	3.952343	4.118994	4.082502	3.974870
100034675_at	2.355263	3.122760	2.527103	2.147679	2.500051	2.161019
100034739_at	1.849819	2.287199	2.269439	2.426231	2.143186	2.188145
	GSM3587199	GSM3587200	GSM3587201	GSM3587202	GSM3587203	GSM3587204
100009600_at	3.570557	2.793264	2.899228	2.578091	2.746432	2.890687
100012_at	2.040693	2.047483	1.920341	2.121895	2.174334	2.022428
100017_at	5.101483	5.088051	4.998591	4.983577	5.275112	4.993089
100019_at	4.176549	3.678023	4.207287	4.275201	4.417371	4.221848
100034675_at	2.456386	2.307226	2.364352	2.380996	2.218104	3.326487
100034739_at	2.758540	2.681238	2.324663	2.107863	1.839197	2.120801
	GSM3587205	GSM3587206	GSM3587207	GSM3587208		
100009600_at	3.107076	3.755808	2.723219	3.205439		
100012_at	2.063782	1.960947	2.167812	1.934829		
100017_at	4.752626	4.719848	5.408127	5.292625		
100019_at	3.836021	4.488508	3.721000	4.455525		
100034675_at	2.356992	2.351102	2.382482	3.915802		
100034739_at	2.289795	1.845371	2.146163	2.171648		

Control de calidad de datos normalizados

Empleamos de nuevo la función *arrayQualityMetrics* para producir un informe de métricas para comprobar la calidad de la matriz normalizada y guardamos la tabla creada en la carpeta de resultados.

Se obtiene además unos graficos representativos del informe.



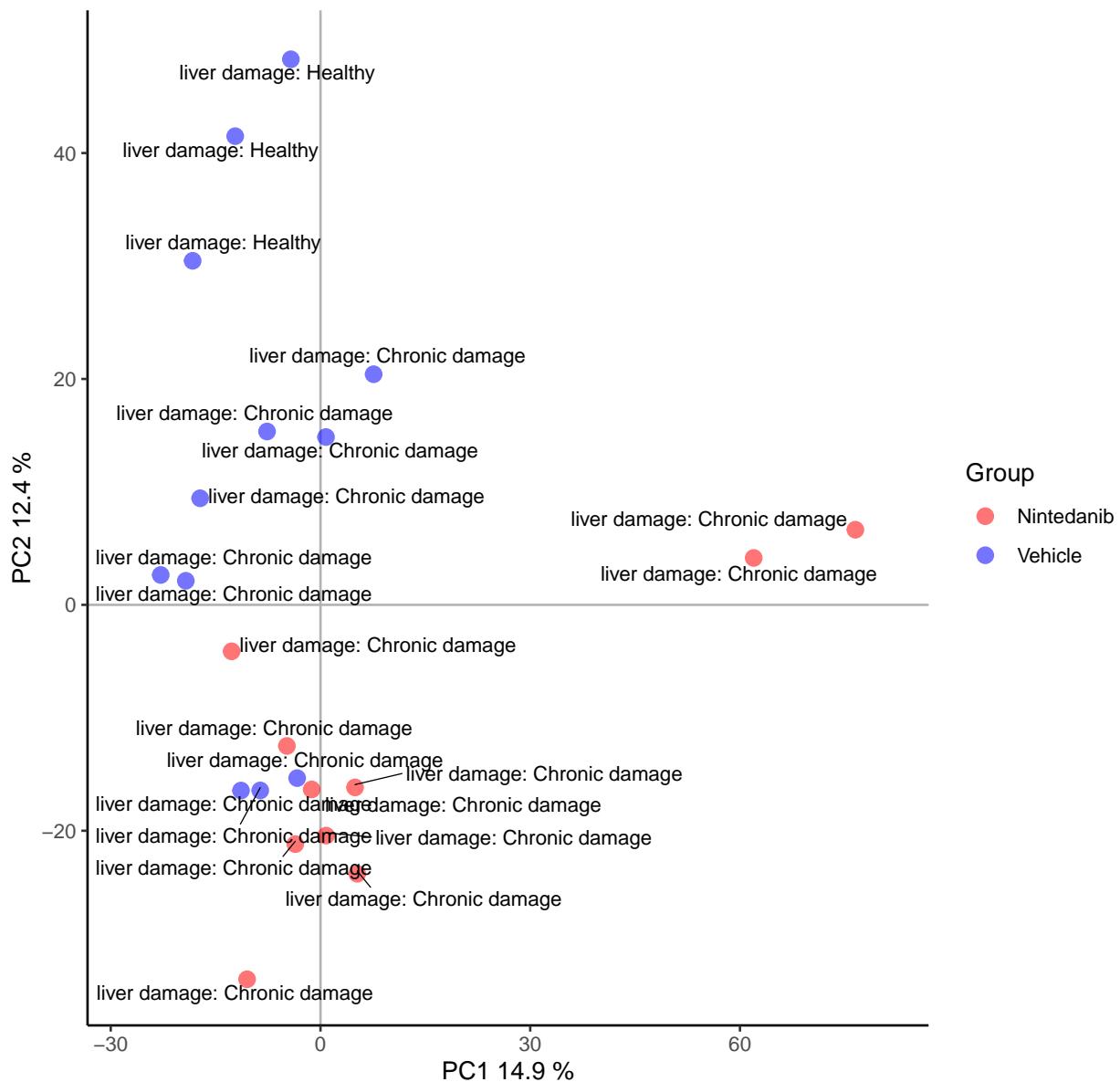


PCA y boxplot

Tras este preprocesamiento podemos observar en los correspondientes boxplot y PCA que los niveles de expresión son comparables entre las distintas muestras.

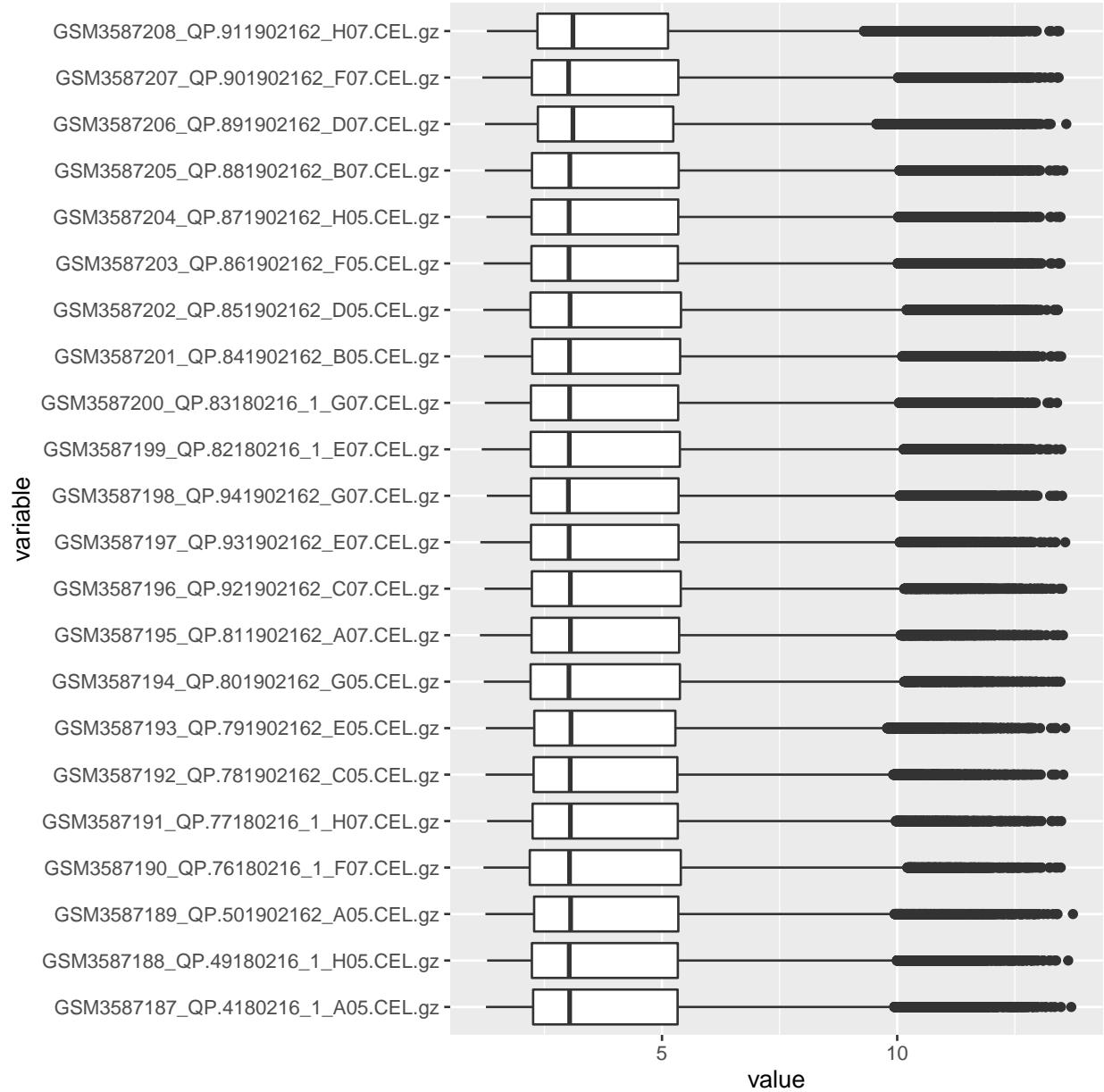
Visualización de los dos primeros componentes principales para datos normalizados.

Principal Component Analysis for: Normalized data

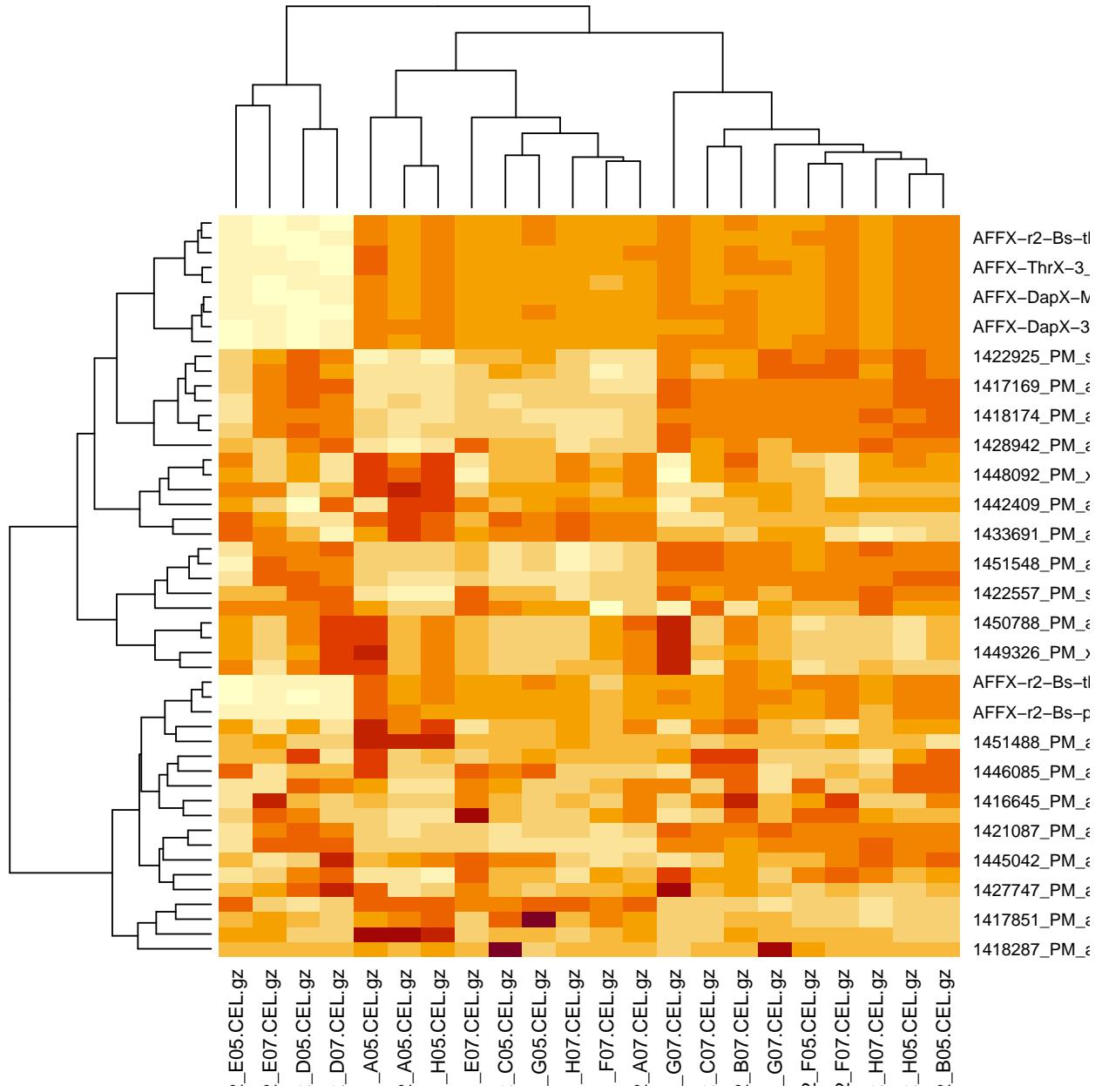


Ahora el primer componente representa el 14.9% de la variabilidad total. El porcentaje de variabilidad explicada ha disminuido con respecto a la PCA realizada en datos sin procesar. Del mismo modo que con el PCA con datos sin procesar, separa las muestras del nivel Nintedanib a la izquierda, y las muestras del control a la derecha. Es importante tener en cuenta que hay alguna muestra del grupo control(vehicle) que agrupa cerca de Nintedanib y viceversa. Podría ser un problema de etiquetado incorrecto de las muestras que deben verificarse con el laboratorio que ha procesado las muestras.

La figura siguiente muestra un diagrama de caja múltiple que representa la distribución de las intensidades normalizadas a lo largo de todas las muestras. Todas las gráficas de caja tienen el mismo aspecto. Esto sugiere que la normalización ha funcionado bien. Sin embargo, es importante tener en cuenta que RMA incluye un paso ("normalización cuantil") donde la distribución empírica de todas las muestras se establece en los mismos valores. Como consecuencia, se espera que los diagramas de caja sean idénticos o al menos muy similares.



Podemos además realizar un Heatmap que nos ayude a entender la relación entre las muestras:

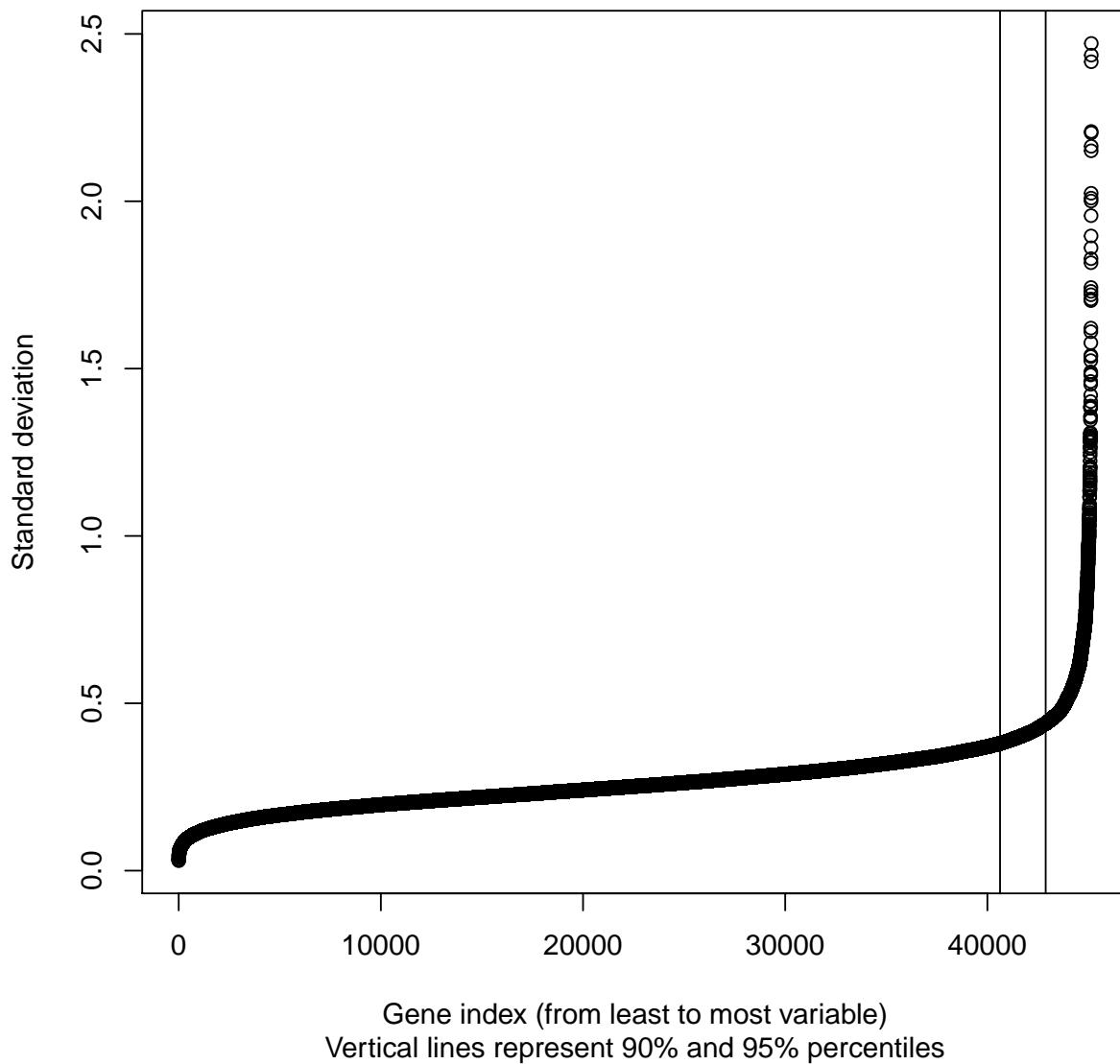


Detectar genes más variables

La selección de genes expresados diferencialmente se ve afectada por la cantidad de genes en los que la hacemos. Cuanto mayor sea el número, mayor será el ajuste necesario de los valores de p , lo que nos llevará a terminar abordando más genes.

Si un gen se expresa de manera diferencial, se espera que haya una cierta diferencia entre los grupos y, por lo tanto, la varianza general del gen será mayor que la de aquellos que no tienen expresión diferencial. Trazar la variabilidad general de todos los genes es útil para decidir qué porcentaje de genes muestra una variabilidad que puede atribuirse a otras causas que no sean la variación aleatoria. La siguiente figura representa las desviaciones estándar de todos los genes ordenados de menor a mayor valor. El gráfico muestra que los genes más variables son aquellos con una desviación estándar superior al 90-95% de todas las desviaciones estándar.

Distribution of variability for all genes



Podemos empleamos la función “Mclust” del paquete mclust para aplicar el modelo de mezcla gaussiana a la matriz de expresión génica ingresada. Además, también se generan archivos que contienen información sobre la media, la varianza, la proporción de mezcla y la asignación gaussiana del conjunto de datos.

Algoritmos de selección de genes. Selección de genes expresados de forma diferencial

Una vez que hemos obtenido una estimación de los niveles de expresión de cada gen podemos pasar a determinar los genes que cambian su expresión debido a la inyección de N-nitrosodietilamina seguida de dosis semanales de tetracloruro de carbono.

Una vez preprocesados, tenemos dos tipos de análisis sobre datos de microarrays:

1. Estadística inferencial: determinar qué genes están expresados diferencialmente (DEGs) y si dicha expresión es significativa. Para ello creamos una matriz que indique que muestras son réplicas biológicas

de la misma condición, con la función *model.matrix*. Veamos solo las 6 primeras muestras.

La función *toptable* recibe como entrada la salida de eBayes, el número de genes a mostrar (argumento *number*) y el identificador de la comparación a tener en cuenta (argumento *coef*). Podemos ver la cabecera de este marco de datos con la función *head*. Veamos los primeros datos de esta tabla. Por ejemplo, para ver la información de expresión diferencial sobre todos los genes (13392) de la primera comparación:

	Length	Class	Mode			
coefficients	13392	-none-	numeric			
rank	1	-none-	numeric			
assign	2	-none-	numeric			
qr	5	qr	list			
df.residual	13392	-none-	numeric			
sigma	13392	-none-	numeric			
cov.coefficients	1	-none-	numeric			
stdev.unscaled	13392	-none-	numeric			
pivot	2	-none-	numeric			
genes	4	data.frame	list			
Amean	13392	-none-	numeric			
method	1	-none-	character			
design	44	-none-	numeric			
contrasts	2	-none-	numeric			
df.prior	1	-none-	numeric			
s2.prior	1	-none-	numeric			
var.prior	1	-none-	numeric			
proportion	1	-none-	numeric			
s2.post	13392	-none-	numeric			
t	13392	-none-	numeric			
df.total	13392	-none-	numeric			
p.value	13392	-none-	numeric			
lod5	13392	-none-	numeric			
F	13392	-none-	numeric			
F.p.value	13392	-none-	numeric			
	ID	adj.P.Val	P.Value	t	B	logFC
14245_at	14245_at	0.0007044322	7.920846e-08	7.560906	7.697798	0.2497512
17750_at	17750_at	0.0007044322	1.147368e-07	7.399594	7.379695	0.6627852
252838_at	252838_at	0.0007044322	1.578029e-07	7.262020	7.105046	0.2396449
17748_at	17748_at	0.0009840820	2.939313e-07	6.996494	6.566329	0.4648430
56458_at	56458_at	0.0029188100	1.304160e-06	6.374956	5.262532	0.1668160
170768_at	170768_at	0.0029188100	1.307711e-06	6.373838	5.260137	0.3447457
	SPOT_ID					
14245_at	14245					
17750_at	17750					
252838_at	252838					
17748_at	17748					
56458_at	56458					
170768_at	170768					

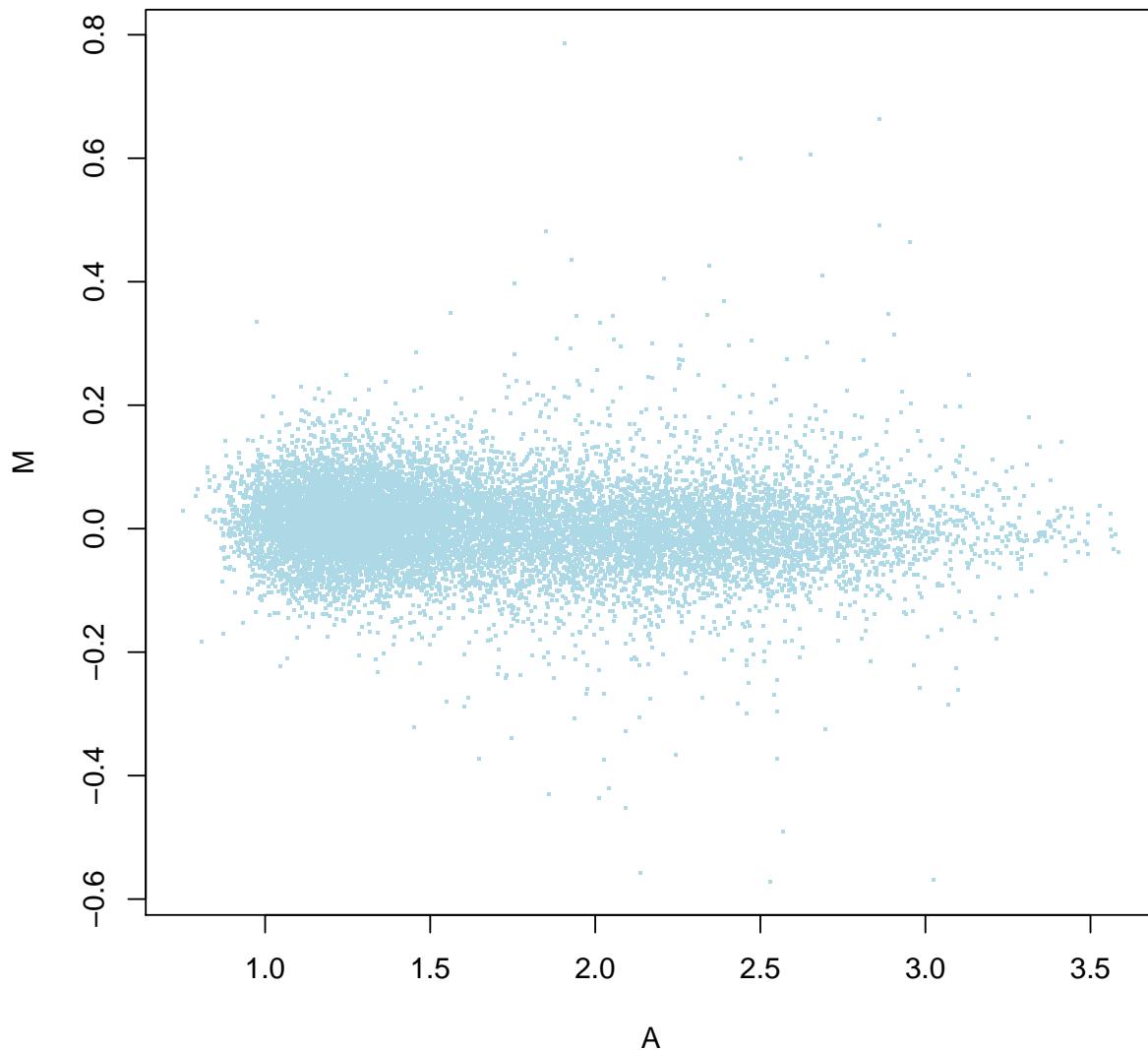
La primera columna ID contiene el identificador de la sonda que representa a cada gen en el microarrays. La columna logFC representa el fold-change en log2 entre las dos condiciones experimentales. La columna AveExpr presenta el valor medio de expresión de cada gen en todas las condiciones experimentales consideradas. La columna t muestra el p-valor del correspondiente constreñido de hipótesis basado en una t-student moderada. La columna P.Value es el p-valor correspondiente mientras que la columna adj.P.Value es el p-valor ajustado para controlar la proporción de falsos descubrimientos o falsos positivos (false discovery rate).

Por ejemplo, la sonda con identificador 14245_at que representa un gen cuya proteína tiene lipin 1 presenta un fold-change en log2 de 0.2497512. Por lo tanto, su nivel de expresión se ha incrementado en $2^{(0.2497512)} = 1.189$ veces más. Este incremento es altamente significativo según el valor del estadístico t-student de 7.560906 con un p-valor de 7.920846e-08 incluso con un p-valor ajustado para el false discovery rate de 0.0007044322.

Si queremos obtener todos los genes cuyo p-valor < 0.05, primero debemos almacenar en una variable todos los genes ordenados por el estadístico B.

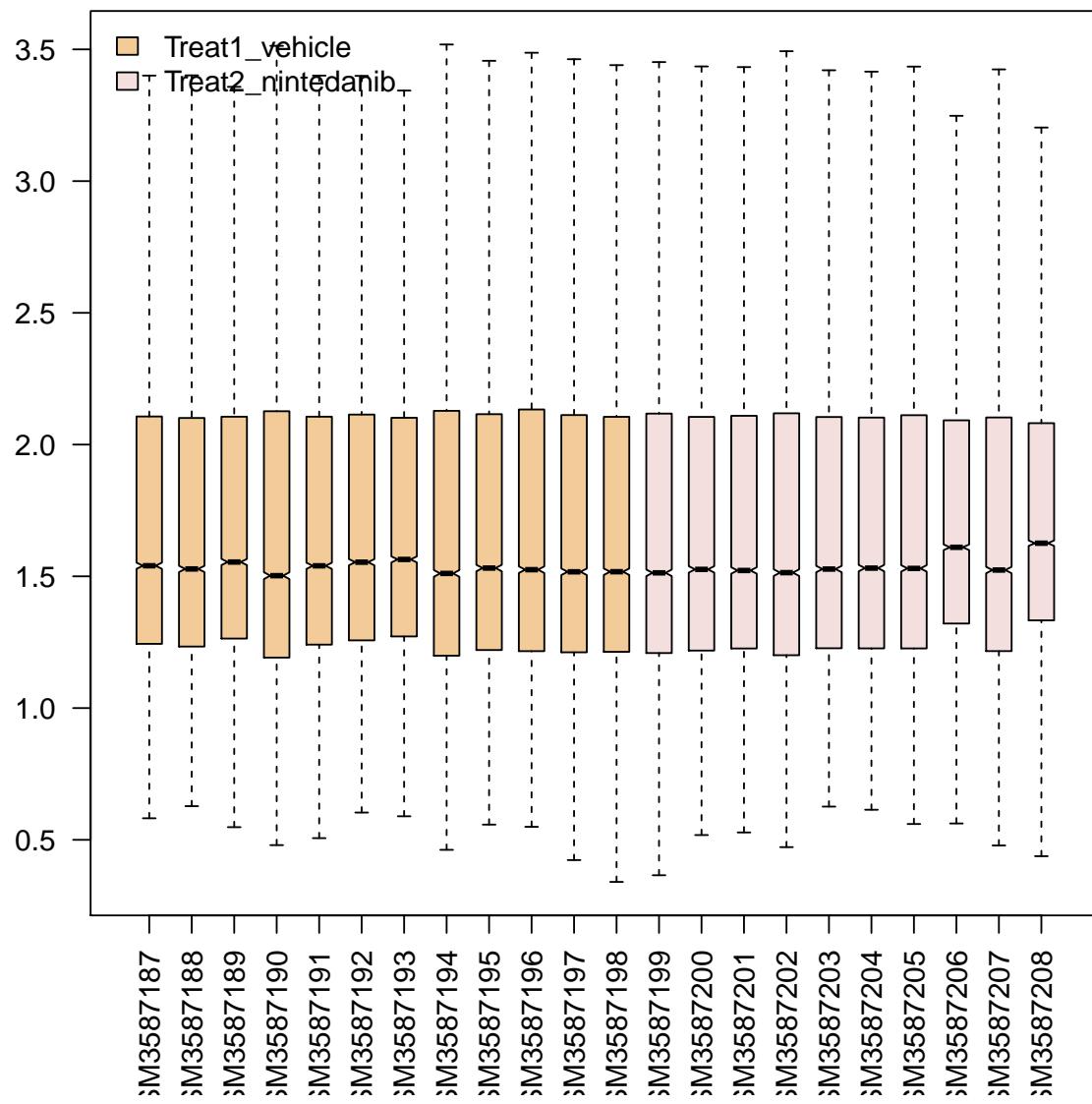
ID	adj.P.Val	P.Value	t	B	logFC	SPOT_ID	
14245_at	14245_at	0.0007044322	7.920846e-08	7.560906	7.697798	0.2497512	14245

Valores de M ajustados por eBayes



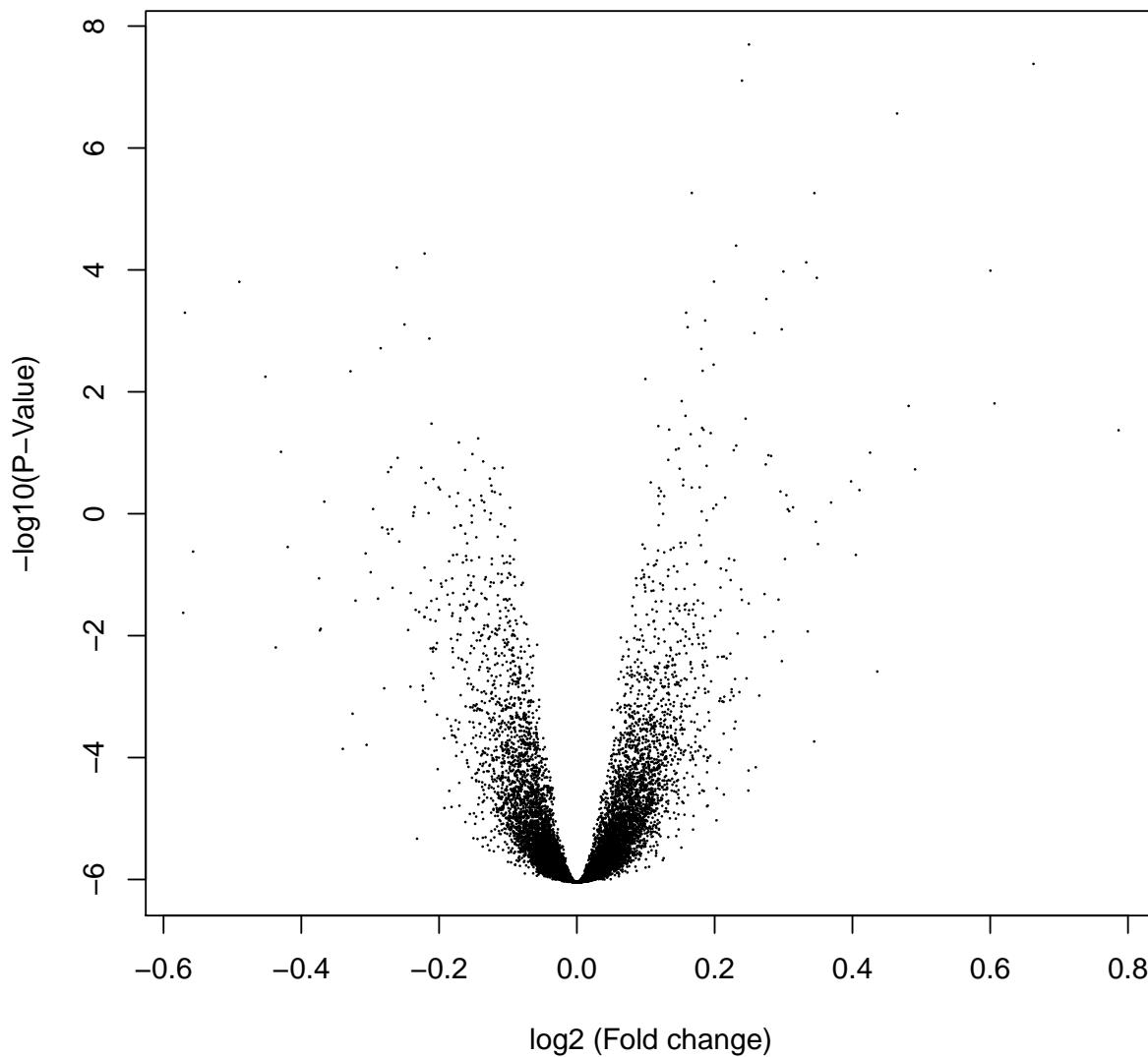
2. Estadística descriptiva: determinar grupos de genes que presentan patrones similares gráficos.

GSE125975/GPL21382 selected samples



Visualización de expresión diferencial: Volcano plot

Differentially expressed genes



La forma más simple de seleccionar genes expresados de forma diferencial se basa en el fold-change de sus niveles de expresión. Esto consiste en comprobar qué genes han visto sus niveles de expresión incrementados o disminuidos en un factor lo suficientemente grande. Esta metodología es interpretable biológicamente y es la más usada cuando se dispone de un bajo número de réplicas. Sin embargo, ignora cualquier análisis estadístico que mida la variabilidad y reproducibilidad de los resultados obtenidos.

Seleccionaremos como genes expresados de forma diferencial aquellos cuyos niveles de expresión presentan un fold-change en log2 superior a un umbral prefijado, típicamente 1, 2 ó 3 que corresponde a 2, 4 u 8 sobre los datos no transformados por log2. Adicionalmente, impondremos que esta diferencia sea significativa exigiendo p-valores menores a un umbral perfijado típicamente 0.05, 0.01 o 0.001. En nuestro ejemplo consideraremos genes expresados de forma diferencial como aquellos con un fold-change superior a 0.1 (0.2 en datos no transformados por log2) con una significancia del 99%.

Para realizar esta selección es necesario extraer del marco de datos los fold-changes, los exponentes de los correspondientes p-valores aplicando el $-\log_{10}$ y los identificadores de cada gen:

```
> fold.change <- tT[["logFC"]]
> log.p.value <- -log10(tT[["adj.P.Val"]])
> probe.names <- tT[["ID"]]
```

Determinamos como genes activados de forma diferencial aquellos cuyo fold-change es superior a 0.1 con una significancia mayor de 99% (p-valor menor 0.01 por lo tanto el correspondiente exponente obtenido como $-\log_{10}$ debe ser mayor de 0.2). De forma análoga determinamos como genes inhibidos de forma diferencial aquellos cuyo fold.change es inferior a -0.1 (esto corresponde a genes que ven su expresión disminuida en menos de la mitad) con una significancia mayor del 99%:

```
> activated <- (fold.change > 0.1) & (log.p.value > 0.2)
> inhibited <- (fold.change < -0.1) & (log.p.value > 0.2)
```

Los vectores de arriba son lógicos, es decir, están formados por valores TRUE y FALSE. Sólo los valores TRUE marcan genes activados o inhibidos diferencialmente. Para obtener los identificadores de estos genes accedemos al vector con todos los identificadores probe.names con ellos:

```
> activated.genes <- probe.names[activated]
> inhibited.genes <- probe.names[inhibited]
```

Podemos obtener el número de genes activados e inhibidos las muestras analizadas, calculando la longitud de los correspondientes vectores con la función length:

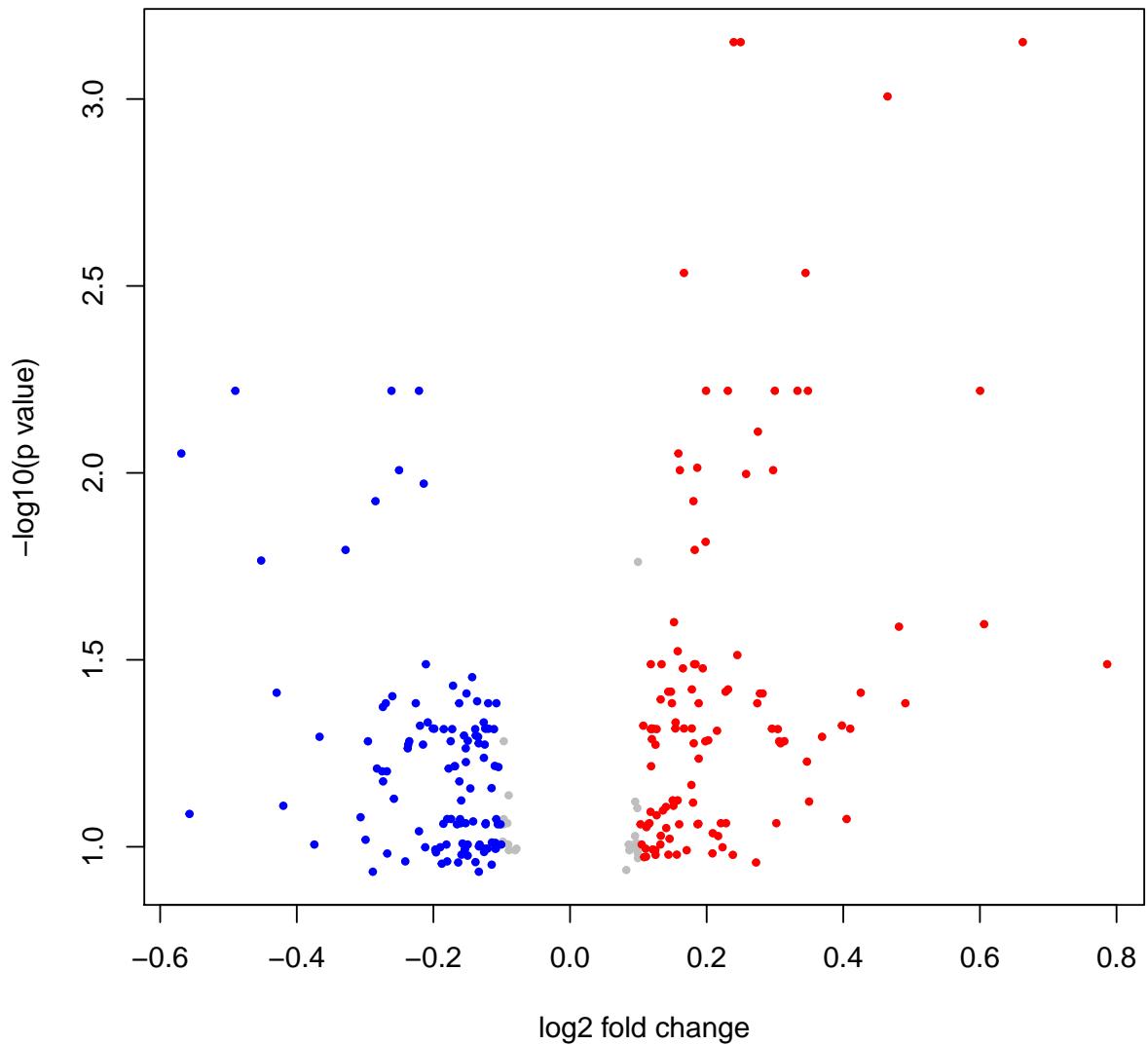
```
> length(activated.genes)
```

```
[1] 113
```

```
> length(inhibited.genes)
```

```
[1] 113
```

Típicamente para representar genes expresados de forma diferencial seleccionados según esta metodología que combina el fold-change con significancia estadística con volcano plots (gráficos de volcán).



En este gráfico podemos observar en azul los genes inhibidos más de la mitad en el tratamiento G1 con respecto a G2 con una significancia estadística del 99% mientras que en rojo se encuentran marcados los genes activados más del doble en G2 con respecto a G1 con una significancia estadística del 99%.

Con estos parámetros el número de genes que ven afectada su expresión debido a la medicación recibida son 757.

4. Resultados

Una vez que se obtiene una lista de genes que caracteriza la diferencia entre dos condiciones, debe interpretarse. La prueba Chi-cuadrado que obtenemos en la lista de genes, nos arroja un estadístico $p < 0.001$ por lo que puede concluirse que existe una fuerte asociación entre la técnica de preprocessamiento empleada y el patrón de expresión predominante en los genes seleccionados como diferencialmente expresados. Sin embargo, son pocos los genes obtenidos y expresados de forma diferencial (aproximadamente un 6%) A la vista de los resultados obtenidos se puede afirmar que hay una relación leve entre el método utilizado en el preprocessamiento y patrón

dominante.

7. Apéndice: Código R

```
> # Para acceder a GEO Sample (GSM), GEO Series (GSE) (listas de archivos GSM que juntos  
> # forman un solo experimento) o GEO Dataset (GDS), empleamos la función getGEO() que  
> # devuelve una lista de ExpressionSets.  
>  
> library(GEOquery)  
> library(affy)  
> gset <- getGEO("GSE125975", GSEMatrix =TRUE, AnnotGPL=FALSE)  
> if (length(gset) > 1) idx <- grep("GPL21382", attr(gset, "names")) else idx <- 1  
> gset <- gset[[idx]]  
> # make proper column names to match toptable  
> fvarLabels(gset) <- make.names(fvarLabels(gset))  
>  
> dim(exprs(gset))  
> pheno.data <- pData(gset)  
>  
> # tambien podemos obtener los datos RAW. descargando los datos directamente de la  
> # pagina a nuestra carpeta de trabajo, despues usamos affy pa leer los datos.  
>  
> celpath <- "./GSE125975_RAW"  
> data.raw = ReadAffy(celfile.path=celpath)  
>  
> # la getGEOSuppFiles() función creará un directorio dentro del directorio de  
> # trabajo actual para almacenar los datos sin procesar. Aquí, las rutas de los  
> # archivos descargados (a menudo con una extensión .tar) se almacenan en un marco  
> # de datos llamado filePaths.  
>  
> #filePaths = getGEOSuppFiles("GSE125975")  
> #filePaths  
>  
> #HISTOGRAMA  
>  
> library(reshape)  
> library(ggplot2)  
> exprs0 <- exprs(gset)  
> df = data.frame(gene = featureNames(gset),exprs0)  
> df1 = melt(df,id=c("gene"))  
> ggplot(df1,aes(x=value,colour=variable,group=variable)) +  
+ geom_density(kernel = "epanechnikov",fill=NA) ## Estimadores densidad  
>  
>  
> # BOXPLOTS  
>  
> ggplot(df1,aes(x=variable,y = value)) + geom_boxplot() +  
+ coord_flip()  
>  
> # HIERARCHICAL CLUSTERING ANALYSIS*  
>  
> dist.clust <- dist(t(exprs(gset)))  
> hclust (dist.clust, "average")
```

```

> plot(hclust (dist.clust, "average"))
>
> # Image
> image(exprs(gset))
>
> ## Control de calidad
>
> library(arrayQualityMetrics)
> arrayQualityMetrics(data.raw, outdir = file.path("./results", "reportquality_dataRAW"))
>
> # PCA
>
> library(ggplot2)
> library(ggrepel)
>
> # Creamos la función necesaria
> plotPCA <- 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$x[,1])+5)) +
+     scale_fill_discrete(name = "Group")
+   # avoiding labels superposition
+   p1 + geom_text_repel(aes(y = PC2 + 0.25, label = labels),segment.size = 0.25,
+                         size = size) +
+     labs(x = c(paste("PC1",loads[1],"%")),y=c(paste("PC2",loads[2],"%"))) +
+     ggtitle(paste("Principal Component Analysis for: ",title,sep=" ")) +
+     theme(plot.title = element_text(hjust = 0.5)) +
+     scale_color_manual(values=colores)
+ }
>
> # lo dibujamos
> plotPCA(exprs(data.raw), labels = pheno.data$characteristics_ch1,
+           factor = pheno.data$treatment:ch1`,
+           title="Raw data", scale = FALSE, size = 3,
+           colores = c("red", "blue", "green", "yellow"))
>
> ## Normalización
>
> eset_rma <- rma(data.raw)
>
> # Podemos realizar la normalización con otro tipo de funciones, con el
> # paquete *limma* y/o *edgeR*.
>

```

```

> library(limma)
> library(edgeR)
>
> plotMDS(gset) #Esta función grafica las muestras en un diagrama de dispersión
> # bidimensional para que las distancias en el gráfico se aproximen a los cambios
> # típicos de pliegue log2 entre las muestras.
>
> calcNormFactors(gset) # Calcula los factores de normalización para escalar los datos.
>
> library(limma)
> # La siguiente función normaliza las intensidades de expresión para que las
> # intensidades o las proporciones logarítmicas tengan distribuciones similares
> # en un conjunto de matrices
> exprs1 = normalizeBetweenArrays(exprs(gset),method = "quantile")
> head(exprs1)
>
> # O bien podemos hacer que coincidan las medianas de los distintos arrays con
> #exprs2 = normalizeBetweenArrays(exprs(gset),method = "scale")
> #head(exprs2)
>
> ## Control de calidad de datos normalizados
>
> qc_rma <- arrayQualityMetrics(eset_rma, outdir = file.path("./results",
+                                         "QCraw.Norm"), force=TRUE)
>
> head(qc_rma)
>
> # PCA y boxplot
>
> plotPCA(exprs(eset_rma), labels = pheno.data$characteristics_ch1,
+          factor = pheno.data$treatment:ch1`,
+          title="Normalized data", scale = FALSE, size = 3,
+          colores = c("red", "blue", "green", "yellow"))
>
> exprs1 <- exprs(eset_rma)
> df.norm = data.frame(gene = featureNames(eset_rma),exprs1)
> df2 = melt(df.norm,id=c("gene"))
>
> ggplot(df2,aes(x=variable,y = value)) + geom_boxplot() +
+ coord_flip()
>
> #heatmap
> rsd <- apply(exprs(eset_rma), 1, sd)
> sel <- order(rsd, decreasing = TRUE)[1:50]
> heatmap(exprs(eset_rma)[sel, ])
>
> ## Detectar genes más variables
>
> 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))
>
> # BiocManager :: install("mclust")
> # library(mclust)
> # mclustplot <- Mclust(eset_rma) # Esta función devuelve un objeto que
> # contiene el número de genes / sondas para seleccionar durante el proceso
> # de selección de características.
> # summary(mclustplot)
> # head(mclustplot)
> # plot(mclustplot)
>
> ## Algoritmos de selección de genes. Selección de genes expresados de
> # forma diferencial
>
> # nombres de grupo para todas las muestras
> gsms <- "0000000000001111111111"
> sml <- c()
> for (i in 1:nchar(gsms)) { sml[i] <- substr(gsms,i,i) }
>
> # log2 transform
> exprs(gset) <- log2(exprs(gset))
>
> # configurar los datos y proceder con el análisis
> sml <- paste("G", sml, sep="") # set group names
> fl <- as.factor(sml)
> gset$description <- fl
> design <- model.matrix(~ description + 0, gset)
> colnames(design) <- levels(fl) # Definición de la configuración experimental:
> # la matriz de diseño
> fit <- lmFit(gset, design) # Estimación del modelo y selección de genes.
>
> # El primer paso para poder seleccionar los genes expresados de forma diferencial
> # consiste en crear una matriz que contenga el diseño experimental, es decir, una
> # matriz que indique que muestras son réplicas biológicas de la misma condición.
> # Esto lo realizamos con la función model.matrix. Seguidamente es aconsejable
> # nombrar las columnas de esta matriz con el nombre de las condiciones o puntos
> # en la serie temporal
>
> cont.matrix <- makeContrasts(G1-G0, levels=design) # contrastar
> fit2 <- contrasts.fit(fit, cont.matrix) # Multiple comparisons
> fit2 <- eBayes(fit2, 0.01) # Para ver los 10 primeros genes expresados
> # diferencialmente, ordenados por el estadístico B, se utilizará la función topTable.
> summary(fit2)
>
> # El paquete limma implementa la función topTable que contiene, para un
> # contraste dado, una lista de genes ordenados del valor p más pequeño al más
> # grande que puede considerarse como expresado de mayor a menor diferencial.
> # Para cada gen se proporcionan las siguientes estadísticas:
>
> tT <- topTable(fit2, adjust="fdr", sort.by="B", number=250)
>
> tT <- subset(tT, select=c("ID","adj.P.Val","P.Value","t","B","logFC","SPOT_ID"))
> #Obtaining lists of differentially expressed genes

```

```

> # write.table(tT, file=stdout(), row.names=F, sep="\t")
> # write.csv(tT, file= "./results/data_analisys.csv", sep = ",")
>
> #####
> # Boxplot
>
> # order samples by group
> ex <- exprs(gset)[ , order(sml)]
> sml <- sml[order(sml)]
> fl <- as.factor(sml)
> labels <- c("Treat1_vehicle","Treat2_nintedanib")
>
> # set parameters and draw the plot
> palette(c("#f2cb98","#f4fdf", "#AABBCC"))
> dev.new(width=4+dim(gset)[[2]]/5, height=6)
> par(mar=c(2+round(max(nchar(sampleNames(gset)))/2),4,2,1))
> title <- paste ("GSE125975", '/', annotation(gset), " selected samples", sep = '')
>
> boxplot(ex, boxwex=0.6, notch=T, main=title, outline=FALSE, las=2, col=fl)
> legend("topleft", labels, fill=palette(), bty="n")
>
> # Visualización de expresión diferencial: Volcano plot**
>
> plot(fit2$coef, fit2$lods, pch = 16, cex = 0.2,
+       xlab = "log2 (Fold change)", ylab = "-log10(P-Value)",
+       main=paste("Differentially expressed genes"))
> ord <- order(fit2$lods, decreasing = TRUE)
> top8 <- ord[1:8]
> text(fit2$coef[top8], fit2$lods[top8], labels = fit2$genes[top8,
+ "Name"], cex = 0.8, col = "blue")
>
> menores <- subset(tT, P.Value < 0.005, c("ID","adj.P.Val",
+                                         "P.Value","t","B","logFC","SPOT_ID"))
> menores[1,]
>
> # Luego podemos observar visualmente el gen de mayor expresión
> # diferencial de la siguiente forma:
> M<- fit2$coefficients
> A<- fit2$Amean
> plot (A, M, pch=". ", col="lightblue", cex=2,
+       main="Valores de M ajustados por eBayes")
> points(tT$A[1], tT$M[1], pch="*", col=2,cex=1);text(11.2,2.64,tT[1,5])
>
> fold.change <- tT[["logFC"]]
> log.p.value <- -log10(tT[["adj.P.Val"]])
> probe.names <- tT[["ID"]]
>
> activated <- (fold.change > 0.1) & (log.p.value > 0.2)
> inhibited <- (fold.change < - 0.1) & (log.p.value > 0.2)
>
> activated.genes <- probe.names[activated]
> inhibited.genes <- probe.names[inhibited]
>

```

```
> length(activated.genes)
> length(inhibited.genes)
>
> plot(fold.change,log.p.value,pch=19,cex=0.5,col="grey",
+       ylab="-log10(p value)",xlab="log2 fold change")
> points(fold.change[activated],log.p.value[activated],pch=19,cex=0.5,col="red")
> points(fold.change[inhibited],log.p.value[inhibited],pch=19,cex=0.5,col="blue")
```