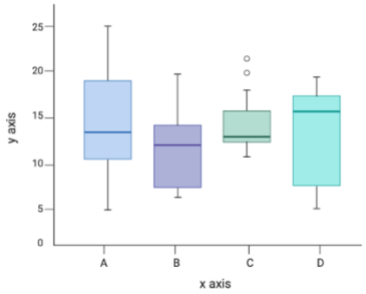
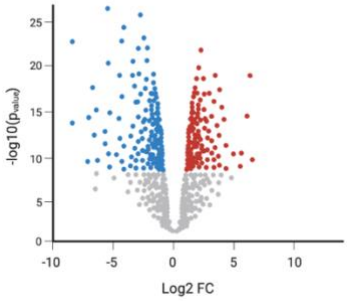
Master Universitario en Bioinformática para Ciencias de la Salud
Facultad de Informática
UNIVERSIDADE DA CORUÑA





Caso práctico.

MÉTODOS ESTADÍSTICOS AVANZADOS EN BIOINFORMÁTICA.

Pedro Sánchez García
MÁSTER UNIVERSITARIO EN BIOINFORMÁTICA PARA CIENCIAS DE LA SALUD.
PROFESOR: DR. IGNACIO LÓPEZ DE ULLIBARRI.

ÍNDICE DE CONTENIDO.

ENUNCIADO DEL CASO PRÁCTICO.

Descargar de la base de datos GEO del NCBI (https://www.ncbi.nlm.nih.gov/gds) el conjunto de datos de microarrays de expresión génica con código de acceso GSE132575. Asimismo, descargar de la Biblioteca de la UDC el artículo asociado a los datos:

Kovi RC, Bhusari S, Mav D, Shah RR, Ton TV, Hoenerhoff MJ, Sills RC, Pandiri AR (2019). Genome-wide promoter DNA methylation profiling of hepatocellular carcinomas arising either spontaneously or due to chronic exposure to Gikgo biloba extract (GBE) in B6C3F1/N mice. *Archives of Toxicology,* 93 (8): 2219-2235. https://doi.org/10.1007/s00204-019-02505-7.

Debe descargarse también el 'Supplementary material' del artículo. La descarga del artículo y datos suplementarios puede hacerse accediendo a la Biblioteca de la UDC *desde fuera de la UDC* (https://www.udc.es/es/biblioteca/recursos_informacion/acceso_fora_udc), proceso que requiere *Autentificarse*. Una vez en la web de la revista, se recomienda usar el doi, 10.1007/s00204-019-02505-7, como término de búsqueda.

PREGUNTA 1.

En relación con el experimento al que está asociado el conjunto de datos:

A) Resumir su objetivo.

Estamos situados en un contexto de intensas investigaciones a nivel mundial en diversos tipos de cáncer. Algunas de ellas siguen una vertiente centrada en las propiedades y efectos de extractos que nos proporciona la Tierra desde los inicios de las civilizaciones. Un suplemento natural muy conocido es el de *Ginkgo biloba*, empleado por la medicina tradicional de la antigua China. Sus propiedades antioxidantes, anticlastogénicas y de regulación génica en redes de señalización celular han despertado el interés en investigaciones como la de Kovi *et al.* (2019). En este caso, se centran en el carcinoma hepatocelular, en concreto, evaluando la relevancia del tratamiento con un extracto de *Gingko biloba* en el desarrollo y evolución del carcinoma asociada a metilaciones de ADN, que conforman un interesante biomarcador para la detección temprana de este proceso espontáneo o debido a la exposición de compuestos tóxicos.

En líneas generales, mediante el uso de la línea de ratones B6C3F1/N, se analizó el perfil de metilación en los promotores de un amplio número de genes relevantes en cáncer. Para ello, se estableció un grupo control, un grupo correspondiente a la aparición espontánea del carcinoma hepatocelular y el grupo de ratones expuestos al extracto de *Gingko biloba* (dosis de 2000 mg/kg). Esta investigación, en el marco de análisis de expresión génica mediante microarrays, representa un avance en la comprensión de mecanismos epigenéticos implicados en los tumores espontáneos y relacionados con tratamientos de compuestos químicos.

B) ¿Qué tipo de microarray se utiliza?

Para realizar los análisis de los perfiles de metilación, se empleó el microarray *Roche NimbleGen mouse DNA methylation microarray '3x720K Promoter Plus CpG Plus RefSeq Promoter' (Roche NimbleGen, Madison, Wisconsin*). Se trata de un microarray de metilación que consta de 20.404 regiones promotoras, 22.881 transcritos y 15.988 islas CpG, permitiendo la determinación de aquellas regiones CpG de promotores metilados entre los grupos distinguidos para el diseño experimental del proyecto.

Por otro lado, para los análisis de expresión génica en los diferentes grupos, se utilizó el microarray *Affymetrix Mouse Genome 430 2.0 GeneChip arrays (Affymetrix, Santa Clara, CA).* Atendiendo a la información proporcionada por la casa comercial *ThermoFisher Scientific (https://www.thermofisher.com/order/catalog/product/900498)*, se trata de un array de un canal que representa en torno a 14.000 genes de ratón caracterizados, lo que permite su uso para proyectos relacionados con múltiples enfermedades. Cabe destacar que las secuencias del array se han seleccionado de GenBank, dbEST y RefSeq, mejorando posteriormente el diseño del array con el ensamblaje más reciente del genoma de ratón. En consecuencia, gracias a este array, se logra una cobertura adecuada de estos genes, sensibilidad y reproducibilidad.

PREGUNTA 2.

Importar los datos con el paquete affy de Bioconductor y guardarlos en un objeto de nombre gse132575. Responder a las siguientes cuestiones:

A) ¿De qué clase es el objeto gse132575? ¿Es una clase de tipo S3 o S4? ¿Cuántos *slots* tiene la clase?

En primer lugar, atendiendo al resumen que se obtiene al introducir el nombre del objeto gse132575 creado en la línea de órdenes, debemos tener en cuenta que se trata de un objeto de la clase AffyBatch. Se comprueba y confirma que es de la clase de tipo S4, es decir, un tipo de clase que se enmarca en la Programación Orientada a Objetos de Bioconductor, con una serie de métodos("funciones") que actúan sobre los objetos de esas clases, a través del acceso directo a cada "slot" en los que se compartimentan estos objetos.

```
> getSlots("AffyBatch")
            cdfName                nrow                ncol           assayData           phenoData
        "character"           "numeric"           "numeric"         "AssayData" "AnnotatedDataFrame"
        featureData      experimentData          annotation        protocolData    .__classVersion__
"AnnotatedDataFrame"             "MIAxE"         "character" "AnnotatedDataFrame"           "Versions"
```

Con respecto a este último aspecto, determinamos que la clase Affybatch consta de 10 slots principales: *cdfName, nrow, ncol, assayData, phenoData, featureData, experimentData, annotation, protocolData* y *classVersion*.

B) Dar el ejemplo de un *slot* del objeto gse132575 que esté vacío (esto es, que no contenga ninguna información).

Se emplea el método de función de acceso homónima para analizar los diferentes *slots* obtenidos anteriormente. De este modo, planteando *nombre_del_slot(gse132575)* en la línea de órdenes, podemos determinar que el *slot* correspondiente a featureData se encuentra vacío:

```
> featureData(gse132575)
An object of class 'AnnotatedDataFrame': none
```

C) Mostrar el contenido del *slot* phenoData empleando diversas funciones de acceso. En concreto, mostrar los nombres de los microarrays con sampleNames() y, si son excesivamente largos (por ejemplo, si son los nombres de los archivos CEL.gz originales), cambiarlos por otros más cortos.

En este caso, además de la función homónima, se lleva a cabo también un procedimiento directo en el que se emplea el operador "@" en una orden *gse132575@nombre_del_slot*. Se obtuvieron los siguientes resultados:

```
> gse132575@phenoData ## Acceso mediante el operador @
An object of class 'AnnotatedDataFrame'
  sampleNames: GSM3876418.CEL.gz GSM3876420.CEL.gz ... GSM3876453.CEL.gz (18 total)
  varLabels: treatment rep
  varMetadata: labelDescription
> phenoData(gse132575) ## Acceso a través de la función de acceso homónima
An object of class 'AnnotatedDataFrame'
  sampleNames: GSM3876418.CEL.gz GSM3876420.CEL.gz ... GSM3876453.CEL.gz (18 total)
  varLabels: treatment rep
  varMetadata: labelDescription
```

Ahora bien, para mostrar los nombres de los microarrays con *sampleNames*, se plantea el acceso con la función de acceso homónima *sampleNames(gse132575)* y con *sampleNames(phenoData(gse132575)),* lo que conduce a los siguientes resultados, respectivamente:

```
> sampleNames(gse132575)
 [1] "GSM3876418.CEL.gz" "GSM3876420.CEL.gz" "GSM3876422.CEL.gz" "GSM3876424.CEL.gz" "GSM3876426.CEL.gz" "GSM3876428.CEL.gz"
 [7] "GSM3876430.CEL.gz" "GSM3876432.CEL.gz" "GSM3876434.CEL.gz" "GSM3876436.CEL.gz" "GSM3876439.CEL.gz" "GSM3876441.CEL.gz"
[13] "GSM3876443.CEL.gz" "GSM3876445.CEL.gz" "GSM3876447.CEL.gz" "GSM3876449.CEL.gz" "GSM3876451.CEL.gz" "GSM3876453.CEL.gz"

> sampleNames(phenoData(gse132575))
 [1] "GSM3876418.CEL.gz" "GSM3876420.CEL.gz" "GSM3876422.CEL.gz" "GSM3876424.CEL.gz" "GSM3876426.CEL.gz" "GSM3876428.CEL.gz"
 [7] "GSM3876430.CEL.gz" "GSM3876432.CEL.gz" "GSM3876434.CEL.gz" "GSM3876436.CEL.gz" "GSM3876439.CEL.gz" "GSM3876441.CEL.gz"
[13] "GSM3876443.CEL.gz" "GSM3876445.CEL.gz" "GSM3876447.CEL.gz" "GSM3876449.CEL.gz" "GSM3876451.CEL.gz" "GSM3876453.CEL.gz"
```

Cabe destacar que, en ambos casos, se muestran unos nombres de archivo CEL.gz más cortos, pues en el transcurso de la primera fase de generación del archivo *phenoGSE132575*, donde se recogen los datos fenotípicos asociados al experimento, se tomó la decisión de abreviar los nombres.

D) ¿Podría estar vacío el *slot* assayData? ¿Por qué? ¿Qué función se utiliza para acceder al contenido del *slot* assayData? Empleando dicha función, mostrar *parte* del contenido del *slot.*

El *slot* assayData hace referencia a un aspecto de gran relevancia, pues posee los datos correspondientes a las intensidades de fluorescencia en el microarray con el que se está trabajando. De esta forma, en base a la naturaleza del *slot*, no sería posible que estuviese vacío.

Debemos tener en cuenta que, si usásemos los métodos de acceso vistos anteriormente para el acceso al *slot* assayData, se alcanzarían los siguientes resultados:

```
> assayData(gse132575)
<environment: 0x7f8712860478>
> gse132575@assayData
<environment: 0x7f8712860478>
```

Con esto, se obtienen el identificador del *environment (entorno)* que contiene las intensidades del microarray, es decir, la dirección donde se ubica dicho contenido. Para acceder a esto último, se verifica mediante *ls(assayData(gse132575),* que en este caso nos lleva a una matriz *exprs* con todas las intensidades de fluorescencia registradas en el microarray.

```
> ls(assayData(gse132575))
[1] "exprs"
> head(assayData(gse132575)[["exprs"]])
  GSM3876418.CEL.gz GSM3876420.CEL.gz GSM3876422.CEL.gz GSM3876424.CEL.gz GSM3876426.CEL.gz GSM3876428.CEL.gz
1               156               197               166               221               160               142
2             25459             25241             26995             26734             27731             26017
3               314               261               317               299               266               304
4             25752             26618             27316             27830             28607             26550
5               193               106               112               108               124               132
6               189               199               146               155               115               146
  GSM3876430.CEL.gz GSM3876432.CEL.gz GSM3876434.CEL.gz GSM3876436.CEL.gz GSM3876439.CEL.gz GSM3876441.CEL.gz
1               401               129               121               202               126               167
2             24235             24504             26368             28687             28679             18709
3               451               235               303               282               247               308
4             25886             24998             26837             28388             29649             19484
5               119               108               110               151               104               101
6               129               131               117               128               114               139
  GSM3876443.CEL.gz GSM3876445.CEL.gz GSM3876447.CEL.gz GSM3876449.CEL.gz GSM3876451.CEL.gz GSM3876453.CEL.gz
1               233               234               265               282               134               135
2             21934             20566             17860             18076             21973             23663
3               268               351               291               385               184               282
4             22313             20931             17905             18194             21751             24337
5               127               154                97               105                81               121
6               152               263               201               251               149               172
```

Posteriormente, se emplea la función *head(assayData(gse132575)[["exprs"]])*, pues teniendo en cuenta el elevado número de filas, *head* nos permite observar la cabecera del contenido con 6 filas.

E) ¿Se pueden aplicar métodos de la clase eSet al objeto gse132575? ¿Por qué? En caso afirmativo, poner un ejemplo.

Sí es posible aplicar métodos de la clase eSet al objeto gse132575, pues la clase AffyBatch hereda de esa clase eSet, la cual está destinada a metadatos de proyectos de estas características. Un ejemplo corresponde con el método phenoData(), en relación a datos fenotípicos por las variables aportadas por el experimentador. Si efectuamos el siguiente procedimiento: *getMethod("phenoData", signature = "AffyBatch")* obtendremos un error, pues la clase Affybatch por sí sola no contiene ese método.

En cambio, con *getMethod("phenoData", signature = "eSet"),* veremos que sí se distingue:

```
> getMethod("phenoData", signature = "AffyBatch") # Conduce a error
Error in getMethod("phenoData", signature = "AffyBatch") :
  no method found for function 'phenoData' and signature AffyBatch
> getMethod("phenoData", signature = "eSet")
Method Definition:

function (object)
object@phenoData
<bytecode: 0x7f870ef7cfa0>
<environment: namespace:Biobase>

Signatures:
        object
target  "eSet"
defined "eSet"
```

Este aspecto es muy interesante, pues pone de manifiesto la relevancia del concepto de herencia, que resulta de utilidad para trabajo rutinario en el entorno de Bioconductor.

PREGUNTA 3.

En relación con el objeto gse132575 de la Pregunta 2 y empleando las funciones de acceso adecuadas, responder las siguientes cuestiones:

A) ¿Cuántas medidas de intensidad hay en cada uno de los microarrays?

Empleando *nrow(assayData(gse132575)[["exprs"]])* o *nrow(intensity(gse132575))*, vemos que el número de intensidades es considerablemente elevado, pues hay 1.004.004 medidas de intensidad.

B) ¿Cuántos pares de sondas tiene cada microarray? ¿Cuántas sondas?

El número de pares de sondas se obtiene empleando la función probeNames() a través de *length(probeNames(gse132575)*, a través de la cual, podemos acceder al número de elementos del contenido. En este caso, alcanzamos que el número de pares de sondas es de 496.468.

Posteriormente, en lo que respecta a las sondas de cada microarray, se hace el producto de *length(probeNames(gse132575) x 2*, alcanzando 992.936 sondas.

C) ¿Coincide el número de sondas con el número de medidas de intensidad? ¿Por qué?

En base a los resultados obtenidos, podemos apreciar que el número de sondas es inferior al de lecturas de intensidad en los microarrays, lo que se debe fundamentalmente a que hay sondas destinadas a otras funciones en el microarray, tales como el control de ruido e hibridaciones inespecíficas.

D) ¿Cuántos conjuntos de pares de sondas tiene cada microarray?

Para responder a esta cuestión, es necesaria la carga del paquete ath1121501cdf, con información proporcionada por Affymetrix en los archivos CDF (*Chip Definition File*). El conjunto de pares de sondas hace referencia al número de objetos contenidos en el entorno del paquete empleado, por lo que se utiliza *length(ls(ath1121501cdf)* y se alcanza un valor de 45.101 conjuntos de pares de sondas.

PREGUNTA 4.

Escribir una función de R, de nombre dePosicionATipoDeSonda, que permita saber si una posición sobre el microarray corresponde a una sonda PM o MM. En concreto, la función tendrá como *argumentos* un objeto de clase AffyBatch y el número correspondiente a una posición sobre el microarray, y la *salida* será la cadena "PM", la cadena "MM" o la cadena "Ninguno" según que en esa posición haya, respectivamente, una sonda PM, una sonda MM o no haya ningún tipo de sonda. Ejemplo de ejecución:

```
> dePosicionATipoDeSonda(gse132575, 471259)
[1] "MM"
```

Se ha planteado la siguiente función, donde se indicaría en primer lugar, el conjunto de conjunto de pares de sondas. En este caso, se muestra para el conjunto "1415670_at", de modo que, con las correspondientes condiciones, se verifica si la posición b que se establece, corresponde a una sonda PM o MM en el objeto a, que será de clase AffyBatch.

```
pm <- pmindex(gse132575)[["1415670_at"]]
mm <- mmindex(gse132575)[["1415670_at"]]
dePosicionATipoDeSonda <- function(a, b) {
  if(b %in% pm) {
    print('PM')
  } else if(b %in% mm) {
    print('MM')
  } else {
    print('Ninguno')}}
```

Además, cabe destacar que se es importante la evaluación de que no haya ningún tipo de sonda, lo que se muestra con la salida "Ninguno".

PREGUNTA 5.

Crear el objeto de R dni como se indica en el archivo auxiliar generarConDNI.R, que contiene el código de R necesario (este archivo se descarga del Campus Virtual, bloque 'Prueba Práctica'). El objeto dni contendrá los dígitos del DNI sin letra del estudiante. A continuación, ejecutar el siguiente código de R (que también se encuentra en el archivo auxiliar generarConDNI.R), para obtener el nombre de un conjunto de pares de sondas:

```
> set.seed(dni)
> id5 <- sample(length(featureNames(gse132575)), size = 1)
> featureNames(gse132575)[id5]
```

En relación con este conjunto de pares de sondas:

A) ¿Cuántas sondas PM lo componen?

En el presente proyecto, mediante el archivo auxiliar generarConDNI.R, se ha alcanzado el conjunto de pares de sondas "1433854_at". Con el fin de analizar cuántas sondas PM componen este conjunto de pares de sondas, se pueden emplear dos opciones: En primer lugar, con *pm(gse132575, "1433854_at"),* donde el resultado proporcionado es acerca de las intensidades PM de los conjuntos de pares de sondas en cada uno de los microarrays. En él, apreciamos que hay 10 conjuntos de pares de sondas PM que lo componen.

```
> pm(gse132575, "1433854_at")
           GSM3876418.CEL.gz GSM3876420.CEL.gz GSM3876422.CEL.gz GSM3876424.CEL.gz GSM3876426.CEL.gz GSM3876428.CEL.gz
1433854_at1               158               122               150               136               110               115
1433854_at2               178               152               179               141               113               149
1433854_at3               265               181               265               175               151               176
1433854_at4               242               277               286               191               212               177
1433854_at5               331               317               304               221               220               191
1433854_at6               301               308               294               211               229               195
1433854_at7              1363              1336              1459              1109               989               935
1433854_at8               679               499               726               456               411               443
1433854_at9               483               430               542               291               304               321
1433854_at10              360               263               388               221               214               246
```

```
> probeset(gse132575, "1433854_at")
$`1433854_at`
ProbeSet object:
  id=1433854_at
  pm= 10 probes x  18  chips
```

Tal y como se refleja en la figura anterior, la otra opción para la búsqueda es mediante el método probeset(), donde *probeset(gse132575, "1433854_at")* muestra, a través de un breve resumen descriptivo, el número de sondas PM en los microarrays con los que se está trabajando.

B) Obtener las medidas de intensidad de las sondas PM del microarray de la muestra con identificador GSM3876426.

Para este planteamiento, se varía la búsqueda mediante pm(), con *pm(gse132575[, "GSM3876426.CEL.gz"]*, de forma que le indicamos con una sintaxis para indexar, que deseamos obtener los datos correspondientes a la columna de la muestra con el identificador propuesto.

```
> pm(gse132575[, "GSM3876426.CEL.gz"], "1433854_at")
           GSM3876426.CEL.gz
1433854_at1                110
1433854_at2                113
1433854_at3                151
1433854_at4                212
1433854_at5                220
1433854_at6                229
1433854_at7                989
1433854_at8                411
1433854_at9                304
1433854_at10               214
```

Como resultado, obtenemos la tabla reflejada en la figura anterior, con un aspecto simplificado para conocer las medidas de intensidad que son de interés.

C) Obtener las medidas de intensidad de las sondas MM del microarray del apartado *b).*

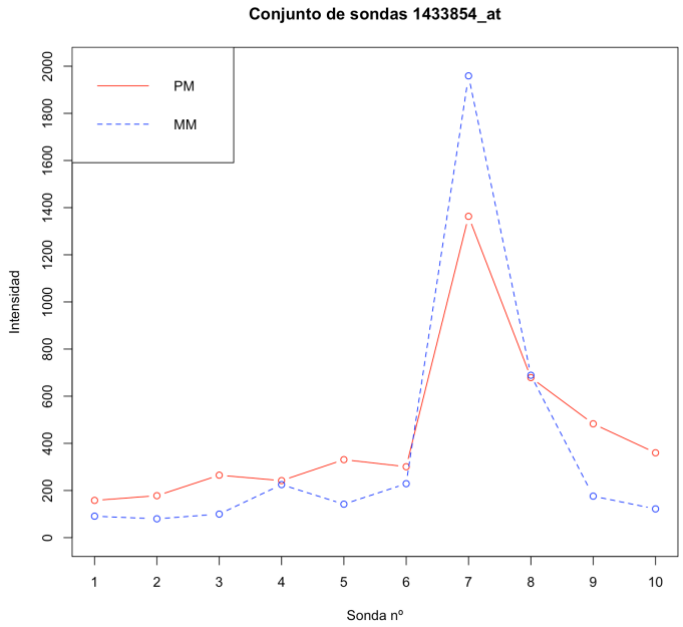
Para las medidas de intensidad de las sondas MM, simplemente se modifica la búsqueda empleando mm(), de tal forma que se plantea *mm(gse132575[, "GSM3876426.CEL.gz"]*:

```
> mm(gse132575[, "GSM3876426.CEL.gz"], "1433854_at")
           GSM3876426.CEL.gz
1433854_at1                124
1433854_at2                 92
1433854_at3                 83
1433854_at4                126
1433854_at5                109
1433854_at6                220
1433854_at7               1555
1433854_at8                460
1433854_at9                131
1433854_at10               110
```

En base al resultado obtenido, cabe destacar que, en términos generales, los valores de las medidas de intensidad de las sondas MM siguen una tendencia similar al caso de sondas PM.

D) Con un gráfico adecuado, mostrar la relación numérica entre las medidas de intensidad de los apartados *b)* y *c).* ¿Son siempre mayores las medidas de intensidad de las sondas PM que las de las correspondientes sondas MM?

Para determinar una correcta visualización de las intensidades de todas las sondas de un determinado conjunto de sondas, en este caso, 1433854_at, se lleva a cabo un gráfico de líneas, donde se enfrenta cada sonda del conjunto a analizar con la correspondiente medida en la lectura de intensidad de fluorescencia.



Tal y como podemos observar en el gráfico obtenido, la tendencia es que las medidas de intensidad son ligeramente superiores en el caso de las sondas PM en comparación con las sondas MM, aunque podemos destacar que se produce una situación contraria en el caso de la sonda 7. Debemos tener en cuenta que la situación ideal es una tendencia donde todas las intensidades de las sondas PM en un conjunto sean superiores a las de las sondas MM. No obstante, lo habitual es que haya algún valor con una situación diferente debido a un valor más elevado de intensidad en las sondas MM, lo que se podría asociar fundamentalmente a un problema de hibridaciones inespecíficas, un aspecto que se debe detectar para las posteriores fases del procedimiento experimental.

Por otra parte, es importante destacar que, en base al comportamiento de las intensidades, resulta imposible plantear sustracción como un método para corrección de fondo asociada a hibridaciones inespecíficas. De esta forma, los métodos de corrección de fondo empleados habitualmente no emplean las sondas MM.

E) ¿Todos los conjuntos de pares de sondas del microarray tienen el mismo número de pares de sondas que el conjunto de pares de sondas estudiado? Justificar la respuesta.

No, pues si procedemos de nuevo a la aplicación de *pmindex(gse132575)*, veremos que el número de pares de sondas es variable. A continuación, se muestra una imagen representativa del análisis realizado:

```
$`1460745_at`
 [1] 556420 678384 664608 267979 181721 173649 220190 770557 576466 769127 874895

$`1460746_at`
 [1] 951625 720637 610678  87897 436281 205862 809257  11764 548652 472367 949538

$`AFFX-18SRNAMur/X00686_3_at`
 [1] 699035 802635 775542 889500 545008 913584  95381 604425 989131 893277 662595 388573 434164 387869 175781 154012 404757
[18]  99731 447982 819216
```

En concreto, se puede apreciar que el número presenta variaciones entre 8, 9, 10, 11, 20 y 21.

PREGUNTA 6.

Con las etiquetas "Espontáneo" y "Tratamiento" se hará referencia a las muestras del experimento correspondientes a ratones que desarrollaron carcinoma hepatocelular espontáneamente o tras el tratamiento con extracto de *Gingko biloba*, respectivamente. Las demás muestras del experimento son controles. Ejecutando el siguiente código de R (contenido en el archivo auxiliar generarConDNI.R)), se guardará en el objeto id6 una de las cadenas "Espontáneo" o "Tratamiento" elegida al azar:

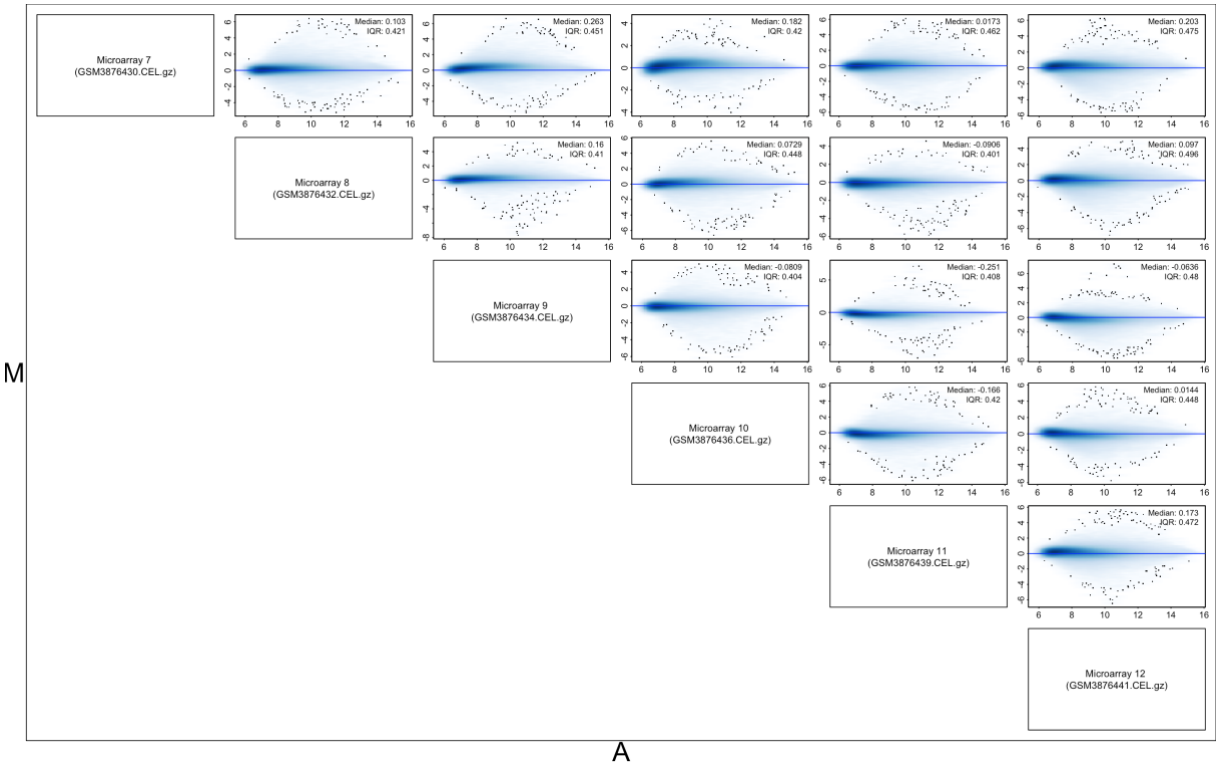
```
> set.seed(dni)
> id6 <- sample(c("Espontáneo", "Tratamiento"), size = 1)
> id6
```

En el código anterior, el objeto dni es el mismo de la Pregunta 5. Responder los apartados siguientes:

A) Representar un diagrama MA que compare las medidas de intensidad de las sondas PM de todos los pares de microarrays correspondientes al valor de id6.

Con el id6 = "Espontáneo" obtenido, se representa el diagrama MA con las medidas de intensidad de las sondas PM en los pares de microarray. Para ello, es preciso el uso de la función MAplot(), con ajustes del código visto en el programa teórico de la materia, con el fin de adaptarlo al presente trabajo. A continuación, se muestra el aspecto del código empleado y el diagrama MA obtenido, respectivamente:

```
par(mfrow = c(6, 6), mar= c(1, 1 , 1, 1), oma = c(2, 2, 0, 0))
for (i in 7:12) {
    for(j in 7:12) {
        if (j <= i) {
            plot(1, 1, type = "n", axes = FALSE)
            if (i == j) {
                box()
                text(1, 1, labels = paste("Microarray", i, paste("\n(", sampleNames(gse132575)[i], ")", sep = "")), cex = 1.1)
            }
        }
        else {
            ma.plot(A = (log2(pm(gse132575[, i])) + log2(pm(gse132575[, j])))/2, M = log2(pm(gse132575[, i])) - log2(pm(gse132575[, j])),
                    add.loess = FALSE, plot.method = "smoothScatter",
                    cex = 0.9, cex.main = 0.9, xlab = "", ylab = "", tcl = NA, mgp = c(3, 0.2, 0))
        }
    }
}
mtext("A", side = 1, line = 1.1, outer = TRUE, cex = 2)
mtext("M", side = 2, line = 0.1, outer = TRUE, cex = 2, las = 1)
box("inner")
```



B) Examinando visualmente el diagrama del apartado a), ¿a qué par de microarrays le corresponde el mayor valor absoluto de la mediana de los valores de M? ¿Cuál es el valor de dicha mediana?

En base a los resultados alcanzados, visualmente podemos determinar que el par de microarrays 7 y 9, es decir, GSM3876430 y GSM3876434, presentan el valor absoluto de mediana más elevado en M, que es de 0.263.

C) En vez de responder a las preguntas del apartado b) mediante un examen visual del diagrama del apartado a), escribir código de R que haga los cálculos automáticamente.

Se plantea un código donde se genera un vector en el que se añaden los valores de la mediana en términos absolutos de M. Las iteraciones comprenden los diferentes pares de microarrays analizados, de modo que en la variable M se asignan los valores calculados de la mediana en términos absolutos, los cuales se añaden posteriormente al vector generado:

```
vector_final = c()
for (i in 7:12) {
  for(j in 7:12) {
      M  <- abs(median(log2(pm(gse132575[, i])) - log2(pm(gse132575[, j]))))
      vector_final <- c(vector_final, M)
    }
  }
print(max(vector_final))
```

Finalmente, se muestra por pantalla el mayor valor absoluto de la mediana de los valores de M.

PREGUNTA 7.

Ejecutando el siguiente código de R (código que se encuentra en el archivo auxiliar generarConDNI.R), se guardará en el objeto id7 el identificador de un conjunto de sondas. En dicho código, el objeto dni es el mismo de las Preguntas 5 y 6, y el símbolo '+' indica que la línea está cortada para que entre en el ancho de página.

```
> set.seed(dni)
> nombres.id <- c("1415789_a_at", "1415861_at", "1416240_at", "1417612_at",
+   "1421320_a_at", "1421899_a_at", "1423997_at", "1425128_at","1428553_at",
+   "1428689_at", "1429840_at", "1430649_at", "1434268_at", "1435718_at",
+   "1437060_at", "1439922_at", "1440278_at", "1440867_at", "1441604_at",
+   "1441876_x_at", "1442171_at", "1452055_at", "1455720_at", "1456500_at",
+   "1456753_at", "1457629_at", "1459034_at", "1459825_x_at")
> id7 <- sample(nombres.id, size = 1)
> id7
```

Responder los apartados siguientes:

A) Usando los recursos de anotación de Bioconductor y detallando el código de R utilizado, averiguar la correspondencia del identificador guardado en id7 con los genes de las bases de datos Entrez Gene y GenBank del NCBI (https://www.ncbi.nlm.nih.gov/).

Con la ejecución del código proporcionado, se alcanzó el id7 = "1456753_at". En primer lugar, se accede al *slot* annotation del objeto gse132575 del proyecto. En base al resultado obtenido de "mouse4302", se procede a la instalación del paquete de Bioconductor mouse4302.db. Además, se carga el paquete annotate de Bioconductor. Mediante el uso del componente which por medio de la instrucción *which(featureNames(gse132575) == '1456753_at')*, podemos conocer la posición del conjunto de sondas en el microarray, que en este caso es la 41.048, por lo que se invoca a la función get() con la posición como identificador y el nombre del recurso, es decir "mouse4302".

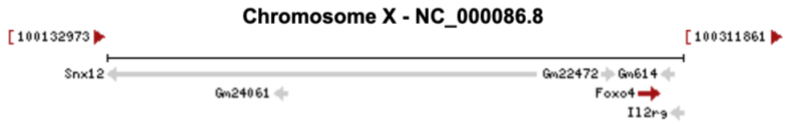
Por medio de *get(featureNames(gse132575)[41048], envir = mouse4302SYMBOL*, obtenemos que se trata del gen Foxo4. Además, cabe destacar que se puede emplear la instrucción alternativa *getSYMBOL(featureNames(gse132575)[41048], 'mouse4302')* para la resolución de la pregunta.

Por otro lado, para tener el identificador del gen en la base de datos Entrez Gene, simplemente planteamos la instrucción *get(featureNames(gse132575)[41048], envir = mouse4302ENTREZID*, que nos conduce al identificador 54601.

B) Usando los recursos del NCBI, responder a las siguientes cuestiones: ¿En qué cromosoma está localizado el gen? ¿En qué hebra del cromosoma? ¿Qué coordenadas tiene? Finalmente, resumir la función de la proteína codificada por el gen.

Para la resolución de este apartado, se combinan recursos de anotación de Bioconductor en relación con registros en la base de datos Gene del NCBI, así como el acceso a la dirección para llevar a cabo las consultas correspondientes sobre el gen Foxo4. Empleando los componentes CHR, CHRLOC y CHRLOCEND en la función get(), por medio de *get(featureNames(gse132575)[41048], envir = <mouse4302CHR o mouse4302CHRLOC o mouse4302CHRLOCEND>)* apreciamos que el

gen se localiza en el cromosoma X, en concreto en la hebra + atendiendo a la breve representación gráfica mostrada en la página, comprendiendo las posiciones 101254527..101260873:



El gen analizado codifica para la proteína Foxo4 *(forkhead box O4)*, cuya función principal es la intervención y facilitar la actividad de unión de secuencias específicas de ADN, lo que la hace versátil, pues resulta fundamental en diversos procesos tales como la regulación transcripcional, regulación de la proliferación celular y el control de daños en ADN en la fase G2 de la mitosis. Además, los conocimientos actuales muestran que la proteína Foxo4 se expresa en estructuras del sistema nervioso central, sistema genitourinario y órganos sensoriales, por lo que podemos concluír que se trata de una proteína muy relevante en diversos procesos en el caso del ratón, al igual que su ortóloga FOXO4 de humanos.

PREGUNTA 8.

Con el subconjunto de microarrays al que se alude en el recuadro al final de la pregunta 7, es decir, el subconjunto formado por las muestras control y las correspondientes al valor de id6, crear el objeto de nombre gse132575.sub. Partiendo de dicho objeto:

A) Hacer una corrección de fondo, considerando sólo las sondas PM.

La creación del subconjunto se lleva a cabo seleccionando las filas correspondientes a las muestras control y de tratamiento espontáneo presentes en el archivo de datos fenotípicos *phenoGSE132575,* donde en este caso se trata de las filas 1 a 12. Por tanto, planteamos la instrucción *gse132575[,1:12]*. Posteriormente, se utiliza la función bg.correct() de affy con la instrucción *bg.correct(gse132575.sub, method = 'rma')*, a través de la cual se efectúa una corrección de fondo por el método RMA-CF, en el que sólo se encuentran involucradas las sondas PM.

B) Tras haber realizado la corrección de fondo, llevar a cabo una normalización de cuantiles, considerando sólo las sondas PM.

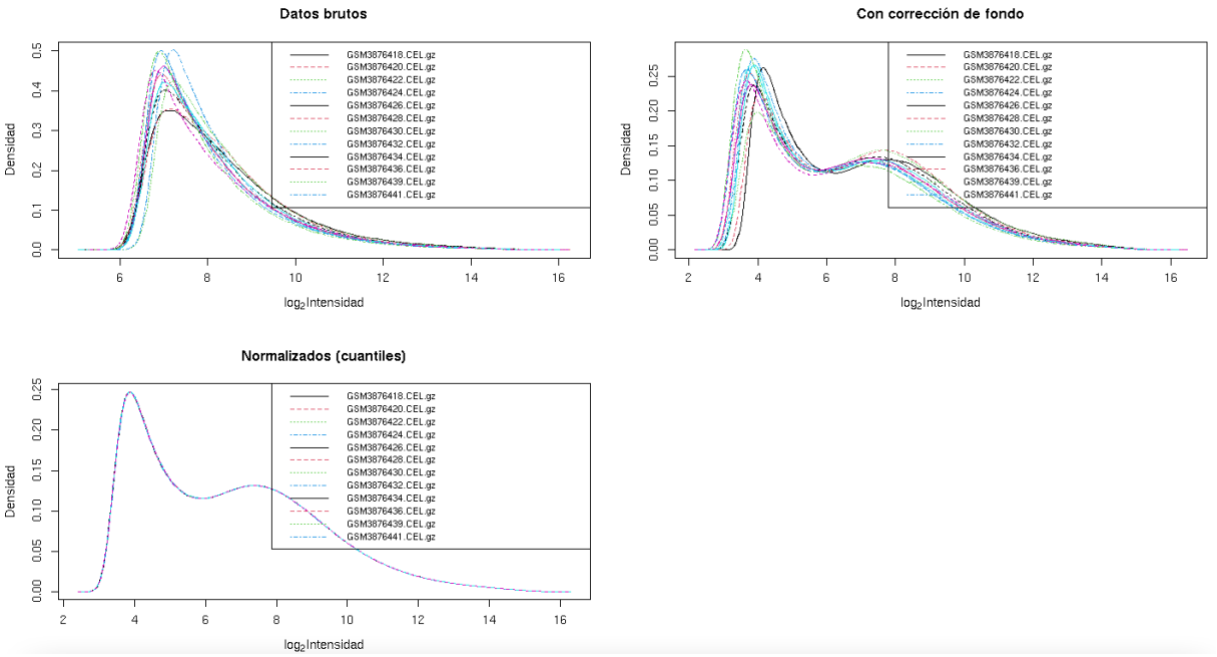
La normalización de cuantiles se lleva a cabo por la llamada a la función normalize(), estableciendo el valor de "quantiles" al correspondiente argumento method: *normalize(gse132575.cor, method = 'quantiles', type = 'pmonly')*. De esta forma, cabe destacar que en el proceso se consideran las sondas PM.

C) Representar gráficamente estimaciones de la densidad de las intensidades brutas, con corrección de fondo y normalizadas. Interpretar los gráficos.

La generación de los histogramas con las estimaciones de densidad de intensidades requiere el planteamiento del siguiente código:

```
par(mfrow = c(2,2))
hist(gse132575.sub, which = "pm", xlab = expression(paste(log[2], "Intensidad")),
     ylab = "Densidad", main = "Datos brutos")
legend("topright", legend = sampleNames(gse132575.sub), col = 1:4, lty = 1:4, cex = 0.7)
hist(gse132575.cor, which = "pm", xlab = expression(paste(log[2], "Intensidad")),
     ylab = "Densidad", main = "Con corrección de fondo")
legend("topright", legend = sampleNames(gse132575.sub), col = 1:4, lty = 1:4, cex = 0.7)
hist(gse132575.normcuant, which = "pm", xlab = expression(paste(log[2], "Intensidad")),
     ylab = "Densidad", main = "Normalizados (cuantiles)")
legend("topright", legend = sampleNames(gse132575.sub), col = 1:4, lty = 1:4, cex = 0.7)
```

Con respecto al código, cabe destacar que se emplea la función hist() con métodos definidos para la clase AffyBatch. Para cada uno de los conjuntos, se muestra la estimación de la densidad con el logaritmo en base 2 de las intensidades, así como etiquetas en los ejes, junto con los nombres de cada uno de los microarrays por medio de un código de colores y líneas en la esquina superior derecha de cada representación gráfica:



En base a los resultados obtenidos con las representaciones gráficas, podemos destacar que, con la corrección de fondo, hay una separación en la escala de la representación, con la distinción de un pico de intensidad en torno a un valor de 8. Esto último resulta de interés en el proyecto, pues se podría tratar de un artefacto, aunque no conformaría un problema considerable, pues con la corrección de fondo, podemos apreciar con mayor claridad el resto de las intensidades dada la separación lograda. Finalmente, con respecto a los datos normalizados, las diferentes densidades estimadas son iguales, de modo que se observa una superposición en el gráfico.

PREGUNTA 9.

Partiendo del objeto gse132575.sub, preprocesar los microarrays en un solo paso empleando la función rma() del paquete affy. ¿Cómo se efectuaría exactamente el mismo preprocesamiento por el método RMA con la función expresso()? ¿A qué se deben las diferencias en eficiencia computacional de las dos funciones?

Con la función rma, el preprocesamiento en un solo paso resulta sencillo y eficiente, pues se plantea la instrucción *rma(gse132575.sub)*, donde se pasa el subconjunto de microarrays tratado anteriormente. A grandes rasgos, se obtienen los resultados de manera muy rápida.

En cambio, con la ejecución de la función expresso(), el preprocesamiento resulta más lento e ineficiente. Además, la llamada a la función implica indicar explícitamente los argumentos, es decir, el método RMA, normalización por cuantiles, tener en cuenta las sondas PM y las medianas:

```
gse132575.expresso <- expresso(gse132575.sub, bgcorrect.method = "rma", normalize.method = "quantiles",
                               pmcorrect.method = "pmonly", summary.method = "medianpolish")
```

De esta forma, podemos asociar las diferencias en la eficiencia computacional a que la función rma() presenta el código de R, mientras que la expreso, a pesar de caracterizarse por una amplia parte de código de R, destina código en C a procedimientos entre los que se encuentran aquellos que consumen más recursos computacionales. En consecuencia, tal y como se ha mostrado en este proyecto, la función rma() es notablemente eficiente frente a la expresso() para el preprocesamiento de microarrays.

PREGUNTA 10.

Partiendo del objeto gse132575.sub ya preprocesado creado en la Pregunta 9, realizar un estudio de la expresión génica diferencial entre ratones de las muestras control y de las muestras correspondientes al valor de id6 (ver Pregunta 6). Con tal fin, utilizar los procedimientos implementados en los paquetes multtest y limma, entre otros. Comparar los resultados obtenidos con los del artículo de Kovi *et al.* (2019) asociado al experimento.

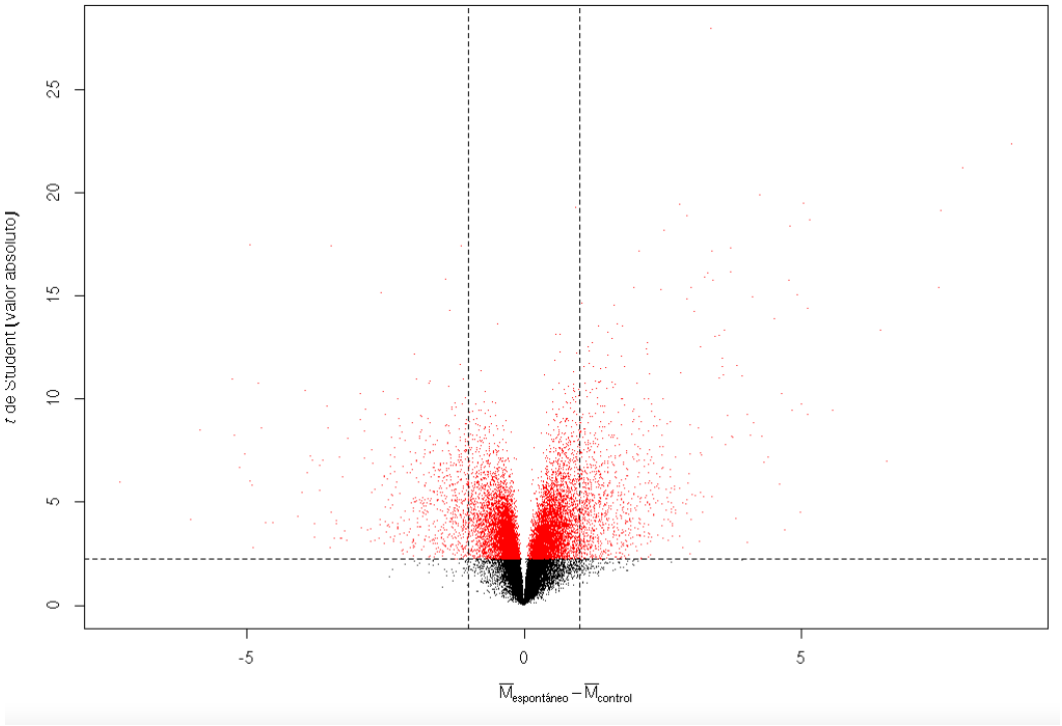
Este estudio de expresión génica diferencial comprende un amplio número de fases, donde se analizarán con rigurosidad los resultados alcanzados, con el fin de evaluar la posible expresión génica diferencial entre ratones de muestras control y de muestras de tratamiento espontáneo. En primer lugar, a partir del objeto gse132575.rma preprocesado, se generan dos subconjuntos de datos correspondientes a las muestras control y de tratamiento espontáneo. Para ello, se emplea el componente which, con las instrucciones *gse132575.rma$treatment == 'control'* y *gse132575.rma$treatment == 'spnt'* para cada uno de los casos.

Posteriormente, se efectúa un diagrama de volcán como una ilustración de ciertos detalles relevantes en el problema a tratar, que además nos permite identificar genes diferencialmente expresados. Se emplea el siguiente código:

```
M.media <- apply(exprs(gse132575.rma), 1, function(x) mean(x[espontáneo]) - mean(x[-espontáneo]))
t.student <- apply(exprs(gse132575.rma), 1, function(x) t.test(x[espontáneo], x[-espontáneo])$statistic)
plot(M.media, abs(t.student), xlab = expression(bar(M)[espontáneo] - bar(M)[control]),
     ylab = expression(italic(t)~de~Student~(valor~absoluto)), pch = ".")
abline(v = c(-1, 1), lty = "dashed")
abline(h = qt(0.975, 10), lty = "dashed")
points(M.media[abs(t.student) >  qt(0.975, 10)], abs(t.student[abs(t.student) > qt(0.975, 10)]), pch = ".", col = "red")
```

Como resultado, en el diagrama se enfrenta para cada conjunto de sondas, el valor absoluto del estadístico t y la diferencia de las intensidades medias de los 2 grupos analizados. A grandes rasgos, se destacan las líneas verticales en la posición de donde la diferencia de las intensidades medias

entre los 2 grupos es igual a ±1. También, es importante destacar la línea horizontal en el percentil 97.5 de la t de Student y el uso de color rojo con el fin de facilitar la identificación de aquellos genes con diferencias de expresión estadísticamente significativas.



En el diagrama alcanzado, tal y como se muestra en la figura anterior, apreciamos numerosos genes en los que el estadístico de la t de Student resulta significativo. No obstante, resulta interesante destacar que esto puede deberse fundamentalmente a que: (a) son genes con un denominador reducido en la estimación de la desviación típica y (b) son genes con valor notablemente elevado en el numerador, es decir, asociado a la diferencia de medias.

Estos últimos genes, son los que interesa detectar principalmente, pues nos centramos en los que presentan un impacto biológico, tratando de reducir el ruido asociado a aquellos genes que han resultado significativos, pero que, en realidad, carecen de impacto biológico a analizar. Para ello, se emplea el paquete genefilter con el fin de efectuar el filtrado y obtener un conjunto menor de genes que mejorarán la potencia en la multiplicidad de contrastes por el ajuste del p-valor. A continuación, se muestra el código empleado:

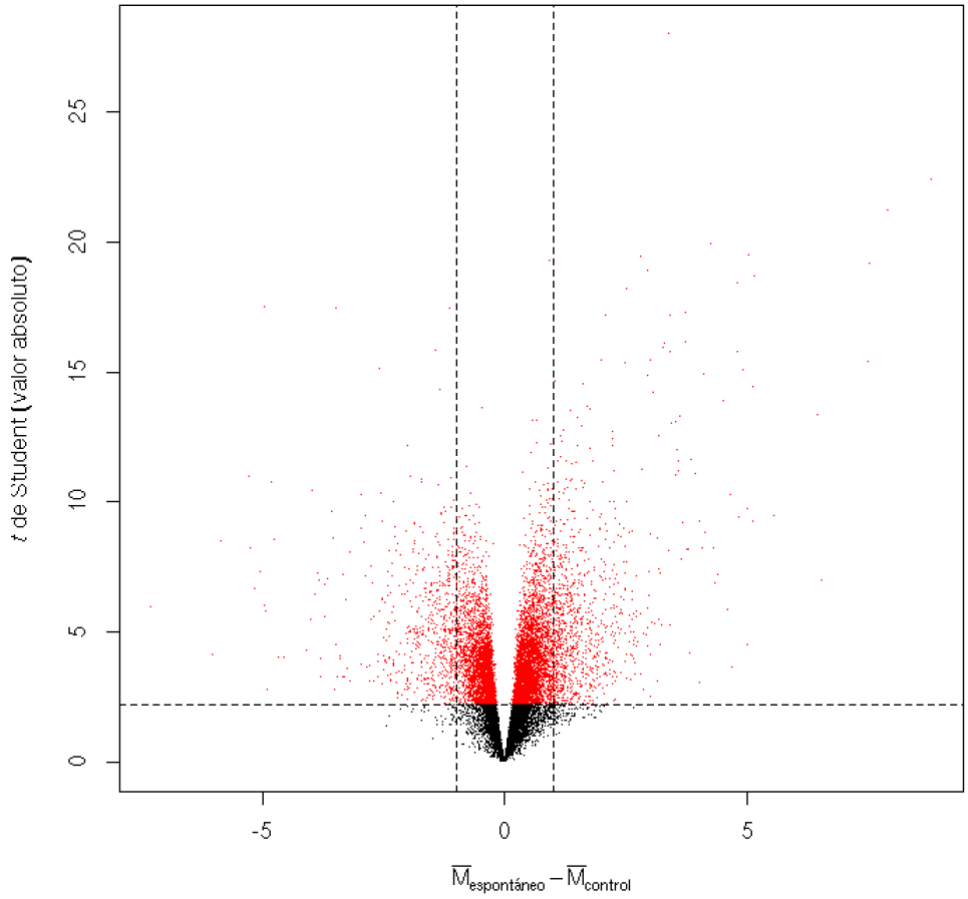
```
library(genefilter)
me1 <- median(apply(exprs(gse132575.rma), 1, IQR))
fun1 <- function(x) IQR(x) > me1
me2 <- median(apply(exprs(gse132575.rma), 1, median))
fun2 <- pOverA(0.1, me2)
funfiltro <- filterfun(fun1, fun2)
filt <- genefilter(exprs(gse132575.rma), funfiltro)
gse132575.rma.filt <- gse132575.rma[filt, ]
```

El fundamento del paquete empleado es la selección de genes con recorrido intercuartílico (IQR) de la medida de expresión en los microarrays superior a la mediana en todos los genes, así como seleccionar aquellos donde la medida de expresión superior al 10% de microarrays es mayor que la mediana del conjunto de medianas de la totalidad de genes.

Como resultado, si el objeto gse132575.rma contenía 45.101 conjuntos de sondas, el nuevo objeto gse132575.rma.filt presenta 17.032. A continuación, siguiendo el código empleado anteriormente, se procede a realizar un nuevo diagrama de volcán:

```
x11()
M.media <- apply(exprs(gse132575.rma.filt), 1, function(x) mean(x[espontáneo]) - mean(x[-espontáneo]))
t.student <- apply(exprs(gse132575.rma.filt), 1, function(x) t.test(x[espontáneo], x[-espontáneo])$statistic)
plot(M.media, abs(t.student), xlab = expression(bar(M)[espontáneo] - bar(M)[control]),
     ylab = expression(italic(t)~de~Student~(valor~absoluto)), pch = ".")
abline(v = c(-1, 1), lty = "dashed")
abline(h = qt(0.975, 10), lty = "dashed")
points(M.media[abs(t.student) >  qt(0.975, 10)], abs(t.student[abs(t.student) > qt(0.975, 10)]), pch = ".", col = "red")
```

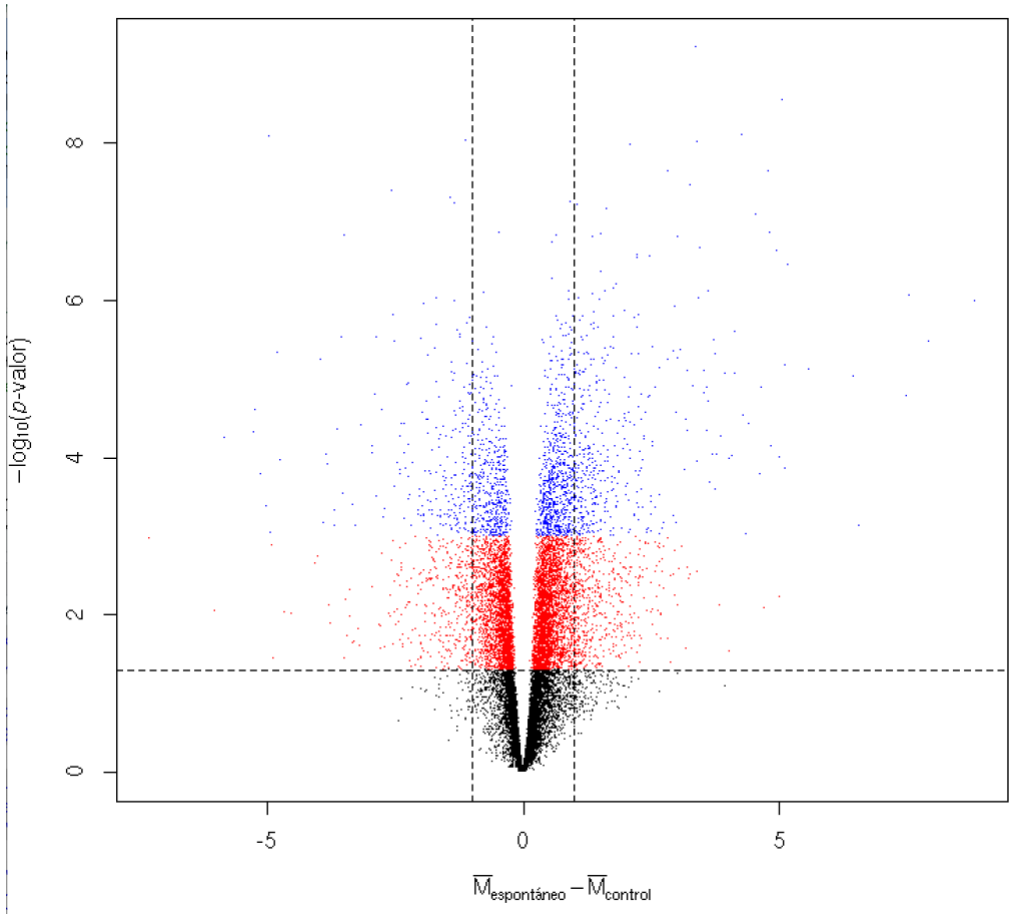
Se alcanza el siguiente resultado:



A grandes rasgos, con respecto al diagrama de volcán realizado anteriormente, se puede apreciar que no hay una reducción notable en el número de genes que resultan significativos. No obstante, hay una mayor separación en la región central del diagrama, como resultado de la eliminación de aquellos genes que aportaban ruido en los análisis de expresión génica diferencial.

En base a los análisis realizados, se ha llevado a cabo posteriormente, una variante del diagrama de volcán, representando $-\log_{10}$(p-valor) como la medida de la significación estadística en el proceso, frente a la diferencia de intensidades medias en tratamiento espontáneo y control. Resulta muy interesante la realización de esta variante, pues es más empleada y además proporciona otra visión del análisis. A continuación, se muestra el código empleado y el resultado alcanzado:

```
x11()
p.valor <- apply(exprs(gse132575.rma.filt), 1, function(x) t.test(x[espontáneo], x[-espontáneo])$p.value)
plot(M.media, -log10(p.valor), xlab = expression(bar(M)[espontáneo] - bar(M)[control]),
     ylab = expression(paste(-log[10],"(",italic(p),"-valor)")), pch = ".")
abline(v = c(-1, 1), lty = "dashed")
abline(h = -log10(0.05), lty = "dashed")
points(M.media[p.valor < 0.05], -log10(p.valor[p.valor < 0.05]), pch = ".", col = "red")
points(M.media[p.valor < 0.001], -log10(p.valor[p.valor < 0.001]), pch = ".", col = "blue")
```



En este caso, podemos apreciar que los puntos azules son aquellos genes que resultan estadísticamente significativos bajo un nivel de significación $\alpha = 0.001$. En cambio, los puntos rojos son aquellos genes estadísticamente significativos con un nivel de significación $\alpha = 0.05$. Con esto, tenemos una idea de que el proceso de fijar el nivel de significación influye notablemente en el número de genes que muestran expresión diferencial entre los tratamientos analizados.

Una vez finalizada esta etapa, se procede al manejo del paquete limma de Bioconductor, que implementa el método *t-moderado*, conformando una variante del estadístico *t* basada en el análisis de Bayes. De esta forma, se evita la problemática de que la distribución *t* de Student aproxime inadecuadamente la distribución nula. Como primer paso, se necesita definir la matriz de diseño del experimento en base a la información de datos fenotípicos comentada anteriormente. Para ello, se emplea el siguiente código:

```
matriz.diseño <- model.matrix(~ factor(treatment, levels = c("control", "spnt")),
                              pData(gse132575.rma.filt))
colnames(matriz.diseño) <- c("intercept", "espontáneo")
matriz.diseño
```

Con la función model.matrix, establecemos los niveles de tratamiento, marcando la referencia del objeto gse132575.rma.filt que se está analizando. Con el siguiente código, se plantea el modelo de regresión lineal a través de la función lmFit() y los valores del estadístico *t-moderado* para llevar a cabo el contraste de igualdad de medias en la expresión de los grupos control y de tratamiento espontáneo:

```
library(limma)
gse132575.type.lmFit <- lmFit(gse132575.rma.filt, design = matriz.diseño)
gse132575.type.eBayes <- eBayes(gse132575.type.lmFit)
```

El objeto gse132575.type.eBayes presenta una estructura en la que distinguimos un componente t, una matriz con la columna "espontáneo" que contiene los valores de los estadísticos *t-moderados* del test que estamos tratando y el componente p.value, con una estructura de matriz con los *p*-valores de los estadísticos *t-moderados* con la variación en la distribución *t* de Student comentada anteriormente. Se ejecuta el siguiente código:

```
sum(gse132575.type.eBayes$p.value[, "espontáneo"] < 0.05)
```

Como resultado, vemos que existen 9035 *p*-valores de los estadísticos *t-moderados*. Podemos tener un conocimiento de la identidad de los conjuntos de sondas y de los *p*-valores por la función topTable de limma, donde a continuación se muestra el código empleado para visualizar la cabecera y el correspondiente resultado obtenido:

```
head(topTable(gse132575.type.eBayes, coef = "espontáneo",
              number = sum(gse132575.type.eBayes$p.value[, "espontáneo"] < 0.05),
              adjust.method = "none", sort.by = "p"))
```

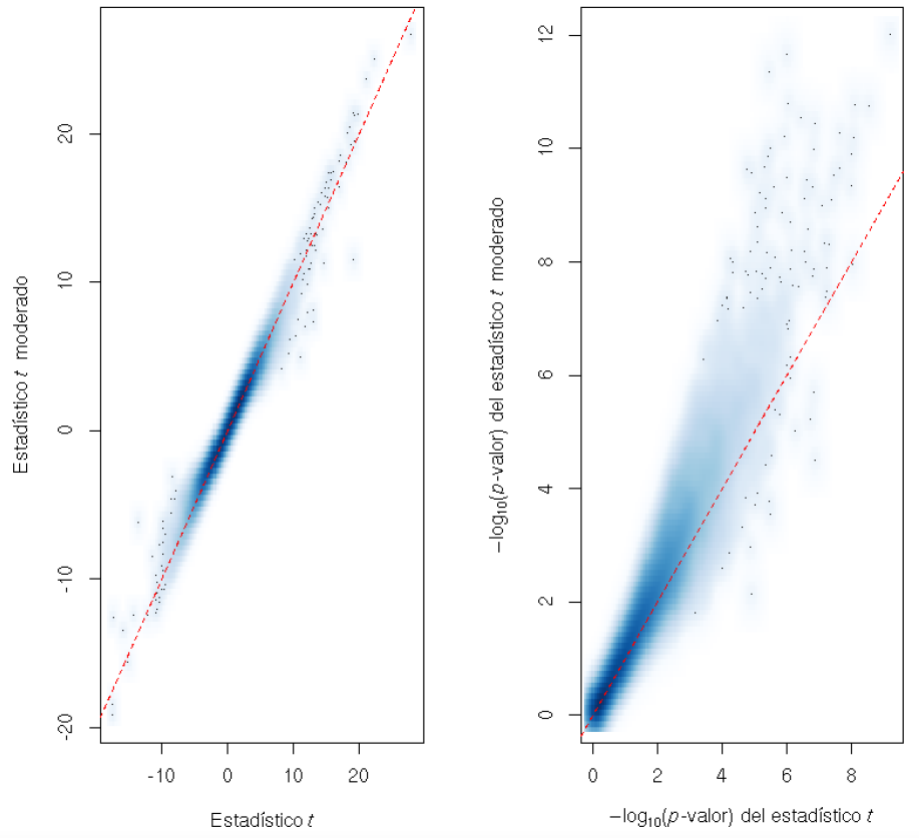
```
                logFC  AveExpr        t      P.Value    adj.P.Val        B
1460550_at   3.378314 7.242144 26.71352 9.635707e-13 9.635707e-13 18.70065
1416645_a_at 8.815539 8.406228 25.00678 2.238307e-12 2.238307e-12 18.05189
1448595_a_at 7.919543 8.013581 23.63624 4.588765e-12 4.588765e-12 17.48104
1424649_a_at 7.536050 9.050558 21.39846 1.621749e-11 1.621749e-11 16.43924
1415824_at   4.268034 7.050784 21.29090 1.728585e-11 1.728585e-11 16.38538
1448261_at   5.045404 9.410076 21.22744 1.795155e-11 1.795155e-11 16.35342
```

Resulta interesante la realización de una comparación gráfica en la que enfrentan los estadísticos *t* clásico y *t-moderado* y los *p*-valores, con el fin de verificar el manejo del estadístico *t-moderado* frente al clásico. Se procede a la ejecución del siguiente código:

```
x11()
par(mfrow = c(1, 2), mar = c(5, 5, 2, 2))
smoothScatter(t.student, gse132575.type.eBayes$t[, "espontáneo"], pch=".", xlab = expression(Estadístico~italic(t)),
              ylab = expression(Estadístico~italic(t)~~moderado))
abline(a = 0, b = 1, lty = "dashed", col = "red")
smoothScatter(-log10(p.valor), -log10(gse132575.type.eBayes$p.value[, "espontáneo"]),
              pch=".", xlab = expression(paste(-log[10],"(",italic(p),"-valor) del estadístico ",
                                               italic(t))), ylab = expression(paste(-log[10],
                                                                                    "(",italic(p),"-valor) del estadístico ",
                                                                                    italic(t), " moderado")))

abline(a = 0, b = 1, lty = "dashed", col = "red")
```

Como resultado, se obtiene el siguiente gráfico:



Apreciamos, en base a la línea roja marcada y a la nube de puntos, que el estadístico *t-moderado* se caracteriza por una menor variabilidad en comparación con el *t*-clásico. En consecuencia, queda claro que este análisis gráfico es de gran interés para apreciar la necesidad de usar el estadístico *t-moderado*.

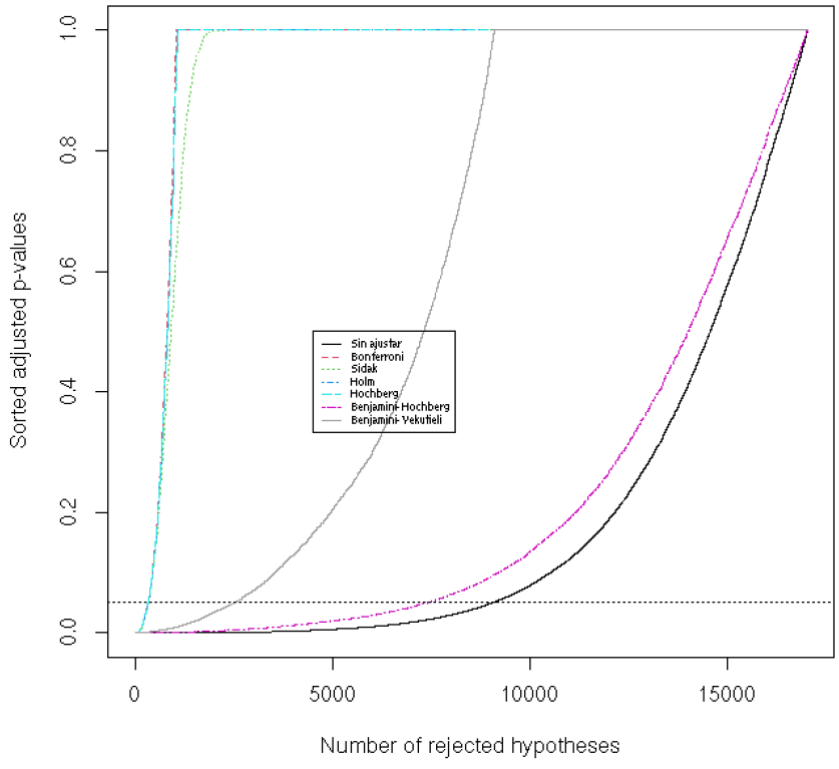
La siguiente fase del análisis de expresión génica diferencial que se está efectuando es la de multiplicidad de contrastes. Con el paquete multtest, en concreto, con la función mt.rawp2adjp(), se ajustan los *p*-valores por los procedimientos de Bonferroni, Sidak, Holm, Hochberg, Benjamin-Hochberg y Benjamini-Yekutieli. Se muestra el código empleado y los resultados, donde además de la función utilizada, se visualizan los *p*-valores más significativos y sus índices correspondientes:

```
library(multtest)
gse132575.type.pval <- mt.rawp2adjp(gse132575.type.eBayes$p.value[,  "espontáneo"],
                                    proc = c("Bonferroni", "SidakSS", "Holm", "Hochberg", "BH", "BY"))
gse132575.type.pval$adjp[1:11,]
gse132575.type.pval$index[1:11]
```

```
> gse132575.type.pval$adjp[1:11,]
              rawp   Bonferroni      SidakSS         Holm     Hochberg           BH           BY
 [1,] 9.635707e-13 1.641154e-08 1.641140e-08 1.641154e-08 1.641154e-08 1.641154e-08 1.693686e-07
 [2,] 2.238307e-12 3.812285e-08 3.812308e-08 3.812061e-08 3.812061e-08 1.906143e-08 1.967157e-07
 [3,] 4.588765e-12 7.815584e-08 7.815599e-08 7.814666e-08 7.814666e-08 2.605195e-08 2.688585e-07
 [4,] 1.621749e-11 2.762164e-07 2.762159e-07 2.761677e-07 2.761677e-07 5.095846e-08 5.258961e-07
 [5,] 1.728585e-11 2.944125e-07 2.944124e-07 2.943434e-07 2.943434e-07 5.095846e-08 5.258961e-07
 [6,] 1.795155e-11 3.057508e-07 3.057504e-07 3.056610e-07 3.056610e-07 5.095846e-08 5.258961e-07
 [7,] 2.850598e-11 4.855139e-07 4.855137e-07 4.853429e-07 4.853429e-07 6.935913e-08 7.157928e-07
 [8,] 3.654400e-11 6.224175e-07 6.224170e-07 6.221617e-07 6.221617e-07 7.780219e-08 8.029259e-07
 [9,] 5.395967e-11 9.190412e-07 9.190416e-07 9.186095e-07 9.186095e-07 9.947139e-08 1.026554e-06
[10,] 6.258067e-11 1.065874e-06 1.065874e-06 1.065311e-06 1.065311e-06 9.947139e-08 1.026554e-06
[11,] 6.424291e-11 1.094185e-06 1.094185e-06 1.093543e-06 1.093543e-06 9.947139e-08 1.026554e-06
> gse132575.type.pval$index[1:11]
 [1] 16890   665 12507  4585    95 12294  2120  4623  4833  1818 10309
```

Estos resultados se puede complementar con un análisis gráfico, pues el objeto gse132575.type.pval almacena los *p*-valores ajustados y sin ajustar en el componente adjp. Por tanto, con la función mt.plot(), se genera el siguiente código para lograr el correspondiente resumen gráfico, mostrado posteriormente:

```
x11()
mt.plot(adjp = gse132575.type.pval$adjp, plottype = "pvsr", proc = c("Sin ajustar", "Bonferroni", "Sidak", "Holm", "Hochberg",
                                                                     "Benjamini-Hochberg", "Benjamini-Yekutieli"),
        leg = c(4500, 0.5), col = c(1:6, 8), lty = 1:7, cex = 0.5)
abline(h = 0.05, lty = "dotted")
```



En el gráfico generado, se representa la relación entre el número de hipótesis rechazadas y la tasa de error de tipo I. Debemos destacar que el número de genes para los que se determina expresión diferencial se reduce de manera considerable. Con el método de Benjamini-Hochberg, cuando la tasa de error de tipo I es 0.05, se identifican en torno a 7.500 genes. El número se reduce de manera drástica en el caso de los otros métodos, pues son más conservadores.

Como fase final del procedimiento, tras la visualización de los *p*-valores llevada a cabo anteriormente, se determina la identidad de los genes que se expresan diferencialmente entre los tratamientos. Para ello, siguiendo el método de Benjamini-Hochberg, se obtienen los identificadores de Affymetrix de los conjuntos de sondas siguiendo el siguiente código, que contiene diferentes aproximaciones para el acceso:

```
## Determinación de la identidad de los genes que se expresan diferencialmente:
## Directamente, con topTable():
rownames(topTable(gse132575.type.eBayes, coef = "espontáneo", number = Inf, p.value = 0.05, adjust.method = "BH", sort.by = "p"))
## Indirectamente, procesando el objeto creado por mt.rawp2adjp():
featureNames(gse132575.rma.filt[gse132575.type.pval$index[gse132575.type.pval$adjp[, "BH"] < 0.05],])
```

```
  [1] "1460550_at"   "1416645_a_at" "1448595_a_at" "1424649_a_at" "1415824_at"   "1448261_at"   "1419136_at"   "1424713_at"   "1425120_x_at"
 [10] "1418572_x_at" "1438596_at"   "1439808_at"   "1443866_at"   "1460329_at"   "1418571_at"   "1420018_s_at" "1415823_at"   "1436879_x_at"
 [19] "1416473_a_at" "1448469_at"   "1416474_at"   "1415822_at"   "1453181_x_at" "1416808_at"   "1435758_at"   "1417821_at"   "1449152_at"
 [28] "1452934_at"   "1429899_at"   "1429159_at"   "1415698_at"   "1428066_at"   "1416646_at"   "1436212_at"   "1441946_at"   "1420017_at"
 [37] "1423515_at"   "1427932_s_at" "1451486_at"   "1431292_a_at" "1439440_x_at" "1434726_at"   "1437199_at"   "1448250_at"   "1417822_at"
 [46] "1451798_at"   "1430172_a_at" "1417125_at"   "1448350_at"   "1449375_at"   "1436591_at"   "1420908_at"   "1418345_at"   "1449049_at"
 [55] "1449309_at"   "1435394_s_at" "1436755_at"   "1425601_a_at" "1426910_at"   "1418028_at"   "1451230_a_at" "1424138_at"   "1422286_a_at"
 [64] "1457403_at"   "1455390_at"   "1418918_at"   "1420541_at"   "1431644_a_at" "1433571_at"   "1416178_a_at" "1417126_a_at" "1441110_at"
 [73] "1450842_a_at" "1421183_at"   "1417879_at"   "1436033_at"   "1416930_at"   "1437112_at"   "1421223_a_at" "1416410_at"   "1448111_at"
 [82] "1448605_at"   "1420385_at"   "1429527_a_at" "1456295_at"   "1460243_at"   "1417267_s_at" "1417370_at"   "1436808_x_at" "1427747_a_at"
 [91] "1433531_at"   "1438852_x_at" "1450913_at"   "1417901_a_at" "1417953_at"   "1437064_at"   "1417392_a_at" "1429417_at"   "1450990_at"
[100] "1423721_at"   "1455266_at"   "1453238_s_at" "1416022_at"   "1421852_at"   "1423049_a_at" "1448319_at"   "1417277_at"   "1433482_a_at"
[109] "1448490_at"   "1449848_at"   "1433504_at"   "1416251_at"   "1457824_at"   "1451373_at"   "1456226_x_at" "1460276_a_at" "1436984_at"
[118] "1420906_at"   "1433985_at"   "1460318_at"   "1454903_at"   "1426300_at"   "1448272_at"   "1419586_at"   "1416026_a_at" "1435870_at"
```

Como resultado, se obtiene un amplio número de resultados (superior a 6.397), que corresponden con los genes de interés. No obstante, debemos tener conocimiento de la correspondencia con los sistemas de nomenclatura estándar, de forma que se cargan los paquetes annotate y mouse4302.db. A continuación, haciendo uso de la función getSYMBOL(), se accede a cada gen identificado que muestra expresión diferencial entre los tratamientos:

```
library(mouse4302.db)
library(annotate)
getSYMBOL(featureNames(gse132575.rma.filt[gse132575.type.pval$index[gse132575.type.pval$adjp[, "BH"] < 0.05], ]), "mouse4302")
```

```
   1460550_at    1416645_a_at    1448595_a_at    1424649_a_at      1415824_at      1448261_at      1419136_at      1424713_at    1425120_x_at
     "Mtmr11"          "Afp"           "Bex1"          "Tspan8"         "Scd2"          "Cdh1"         "Akr1c18"        "Calml4"      "Ifi27l2b"
 1418572_x_at      1438596_at      1439808_at      1443866_at      1460329_at      1418571_at      1420018_s_at    1415823_at    1436879_x_at
  "Tnfrsf12a"      "Hectd2os"        "Ipcef1"        "Lrtm1"        "B4galt6"     "Tnfrsf12a"        "Tspan8"         "Scd2"          "Afp"
 1416473_a_at      1448469_at      1416474_at      1415822_at      1453181_x_at    1416808_at      1435758_at      1417821_at    1449152_at
    "Igdcc4"          "Nid1"          "Igdcc4"         "Scd2"          "Plscr1"        "Nid1"         "B4galt6"    "D17H6S56E-5"      "Cdkn2b"
   1452934_at      1429899_at      1429159_at      1415698_at      1428066_at      1416646_at      1436212_at      1441946_at    1420017_at
      "Tmc5"              NA          "Itih5"         "Golm1"         "Ccdc120"         "Afp"         "Tmem71"         "Itih5"       "Tspan8"
   1423515_at      1427932_s_at    1451486_at      1431292_a_at    1439440_x_at    1434726_at      1437199_at      1448250_at    1417822_at
     "Scn8a"  "1200015M12Rik"       "Slc46a3"         "Twf2"          "Twf2"       "Catsperd"         "Dusp5"         "Clmp"    "D17H6S56E-5"
   1451798_at      1430172_a_at    1417125_at      1448350_at      1449375_at      1436591_at      1420908_at      1418345_at    1449049_at
     "Il1rn"             NA              NA            "Asl"         "Ces2a"         "Vsig10"         "Cd2ap"             NA          "Tlr1"
   1449309_at      1435394_s_at    1436755_at      1425601_a_at    1426910_at      1418028_at      1451230_a_at    1424138_at    1422286_a_at
```

La imagen anterior, correspondiente al resultado obtenido, pone de manifiesto los identificadores de todos aquellos genes de interés en cada conjunto de sondas que han mostrado expresión diferencial entre los tratamientos.

En base a los resultados que se han alcanzado en el análisis efectuado, se puede apreciar al comparar con los del artículo de Kovi *et al.* (2019), que el número de genes detectados con expresión diferencial es muy superior al determinado por los autores (1.632), pues establecieron un fold-change superior a 2.0 y un *p*-valor inferior a 0.05 como criterios conjuntos para determinar esta cuestión.

Debido a estos criterios más estrictos, en el artículo alcanzan un conjunto de 44 genes con expresión diferencial entre control y tratamiento espontáneo. Además, muestran el top 20 de los genes que presentan esta tendencia. Este aspecto resulta de interés, pues los genes que aparecen mostrados incluyen factores de transcripción con diversos roles (*Id3, Lrtm1)* y genes relacionados con señalización y metabolismo celular (*Cyps8b1, Cyp2f2, Cyp2c23)*. Si comparamos esta tabla con la salida mostrada en la última figura de este ejercicio, se pudo comprobar que hay coincidencia con los genes *Tspan8, Tlr1, Tmem71, Btg2* y *B4galt6*. Además, la ausencia de los otros genes que han mostrado un incremento en la expresión al comparar tratamientos como los mencionados, se podría asociar al criterio establecido por los autores.

En lo que se refiere a los genes de la tabla cuya expresión se reduce al comparar tratamientos, se producen coincidencias con respecto al presente trabajo en los genes *Gna14, Aldh5a1, Csrp3, Cyp8b1, Kcnk5, Lppos, Asl* y *Fam214a*. De nuevo, cabe destacar que la ausencia de otros genes se asocia al criterio empleado por los autores.

En general, resulta interesante este análisis comparativo, pues nos proporciona una visión general de genes relacionados con señalización, metabolismo y cáncer, donde algunos como el gen *Dusp5* presenta una participación en carcinomas hepatocelular, objeto de estudio en el proyecto.