UNIVERSIDADE DA CORUÑA

Advanced Statistics | Master's in Bioinformatics Applied to Health Sciences

Study of Differential Gene Expression in Microarrays

Author:

Babih Velázquez De Burnay

April 2023

1. **Regarding the experiment associated with the dataset:**
   a) **Summarize its objective.**
   The main objective of the experiment associated with this dataset is to identify differentially methylated promoter regions of altered genes that are relevant in cancer, specifically in mouse hepatocellular carcinoma, since such methylations are an important marker for early detection of this disease. To achieve this, different microarrays were used to identify the promoters and to measure gene expression, in addition to using three different groups of mice: some were used as controls, others were called spontaneous as they spontaneously developed hepatocellular carcinoma, and finally, others were treated with a Ginkgo biloba extract (GBE), a natural supplement used in China as traditional medicine due to its anti-oxidant and anti-cancer properties.

   b) **What type of microarray is used?**
   To identify differentially methylated promoters, the Roche NimbleGen® mouse DNA methylation microarray was used. On the other hand, to check for differential gene expression in controls, spontaneous, and treatment groups, the Affymetrix Mouse Genome 430 2.0 GeneChip arrays were used.

2. **Import the data with the affy package from Bioconductor and save them in an object named gse132575. Answer the following questions:**
   a) **What class is the gse132575 object? Is it an S3 or S4 class? How many slots does the class have?**
   gse132575 gse132575 is an S4 object of the formal class AffyBatch. This class has 10 slots.

```
########################## Ejercicio 2.a) ##############################
class(gse132575)  # De que clase es el objeto
isS4(gse132575)  # si es de clase S4
slots <- getSlots('AffyBatch')  # Obtener slots
length(slots)  # Número de slots
```

   b) **Give an example of an empty slot in the gse132575 object (i.e., one that contains no information).**
   The slot in the gse132575 object that is empty is AnnotatedDataFrame.

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

   c) **Show the content of the phenoData slot using various access functions. Specifically, show the names of the microarrays with sampleNames(), and if they are excessively long (for example, if they are the original CEL.gz file names), change them to shorter ones.**
   The names of the microarrays were already shortened before being introduced into RStudio.

```
> sampleNames(gse132575@phenoData)
 [1] "GSM3876418.CEL.gz" "GSM3876420.CEL.gz" "GSM3876422.CEL.gz" "GSM3876424.CEL.gz" "GSM3876426.CEL.gz"
 [6] "GSM3876428.CEL.gz" "GSM3876430.CEL.gz" "GSM3876432.CEL.gz" "GSM3876434.CEL.gz" "GSM3876436.CEL.gz"
[11] "GSM3876439.CEL.gz" "GSM3876441.CEL.gz" "GSM3876443.CEL.gz" "GSM3876445.CEL.gz" "GSM3876447.CEL.gz"
[16] "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"
 [6] "GSM3876428.CEL.gz" "GSM3876430.CEL.gz" "GSM3876432.CEL.gz" "GSM3876434.CEL.gz" "GSM3876436.CEL.gz"
[11] "GSM3876439.CEL.gz" "GSM3876441.CEL.gz" "GSM3876443.CEL.gz" "GSM3876445.CEL.gz" "GSM3876447.CEL.gz"
[16] "GSM3876449.CEL.gz" "GSM3876451.CEL.gz" "GSM3876453.CEL.gz"
```

d) Could the assayData slot be empty? Why? What function is used to access the content of the assayData slot? Using this function, show part of the content of the slot.

The assayData slot cannot be empty as it contains the microarray data, i.e., the fluorescence intensity data. To access the content of this slot and view only the first lines, you can do it in three different ways:

```
head(exprs(gse132575))
head(intensity(gse132575))
head(assayData(gse132575)[["exprs"]])
```

First lines of the content of assayData:

```
> 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
```

e) Can methods from the eSet class be applied to the gse132575 object? Why? If so, give an example.

Yes, methods from the eSet class can be applied to the gse132575 object (first line of the script in Figure X) since it is an object of the AffyBatch class, which inherits from the eSet class defined in the biobase package (second line of the script in the figure). Thus, methods defined for that class, such as phenoData used previously, can be applied.

```
> extends("AffyBatch", "eSet")
[1] TRUE
> showMethods("phenoData")
Function: phenoData (package Biobase)
object="AffyBatch"
    (inherited from: object="eSet")
object="eSet"
```

3. Regarding the gse132575 object from Question 2 and using the appropriate accessor functions, answer the following questions:

a) How many intensity measurements are there in each microarray?

Each microarray records 1,004,004 intensity measurements, in total there are 18,072,072 measurements (18 x 1,004,404).

```
> nrow(exprs(gse132575))  # Número de medidas de intensidad en cada microarray
[1] 1004004
> length(exprs(gse132575))  # Número total de medidas de intensidad
[1] 18072072
```

b) How many probe pairs does each microarray have? How many probes?

```
> length(probeNames(gse132575)) # Numero de pares de sondas
[1] 496468
> length(probeNames(gse132575))*2  # Número de sondas
[1] 992936
```

c) Does the number of probes match the number of intensity measurements? Why?

The number of probes does not match the number of intensity measurements as there are far fewer probes than measurements. This may be because some probes are used for other functions in the microarrays, such as internal control of nonspecific hybridizations (MM probes).

d) How many probe pair sets does each microarray have?

There are 45,101 probe pair sets.

```
> length(featureNames(gse132575))
[1] 45101
```

4. Write an R function named dePosicionATipoDeSonda that allows you to know if a position on the microarray corresponds to a PM or MM probe. Specifically, the function will take as arguments an AffyBatch object and the number corresponding to a position on the microarray, and the output will be the string "PM", "MM", or "None" depending on whether there is a PM probe, an MM probe, or no probe at that position, respectively. Example execution:

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

Function:

```
dePosicionATipoDeSonda <- function(obj_affy, num){
  if (length(which(sapply(pmindex(obj_affy), function(x) any(x == num)))) != 0){
      return('PM')
      } else if (length(which(sapply(mmindex(obj_affy), function(x) any(x == num)))) != 0){
      return('MM')
      } else {return('Ninguno')}
  }
```

Execution:

```
> dePosicionATipoDeSonda(gse132575, 471259)
[1] "MM"
> dePosicionATipoDeSonda(gse132575, 3)
[1] "Ninguno"
> dePosicionATipoDeSonda(gse132575, 3765)
[1] "PM"
```

5. Create the R object dni as indicated in the auxiliary file generarConDNI.R, which contains the necessary R code (this file is downloaded from the Virtual Campus, 'Prueba Práctica' block). The dni object will contain the digits of the student's DNI without the letter.

Next, run the following R code (also found in the auxiliary file generarConDNI.R) to obtain the name of a probe pair set:

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

Regarding this probe pair set:

a) How many PM probes does it consist of?

The probe pair set 1427186_a_at consists of 198 PM probes.

```
> length(pm(gse132575, featureNames(gse132575)[id5]))  # Número de sondas PM
[1] 198
```

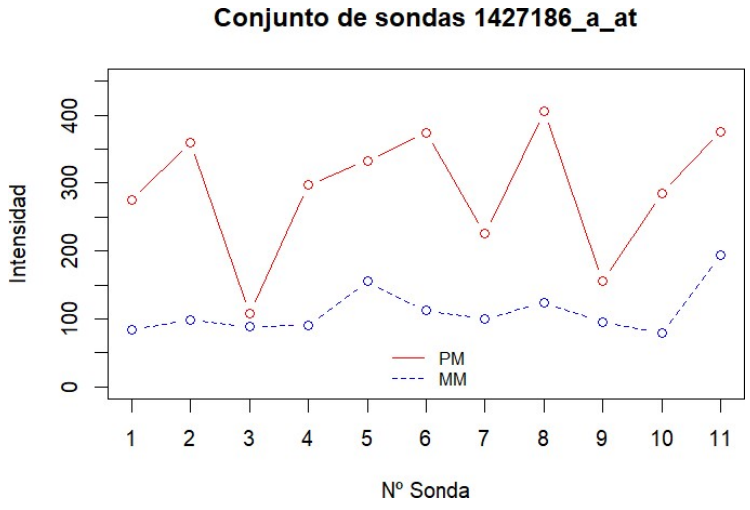
b) Obtain the intensity measurements of the PM probes from the microarray of the sample with identifier GSM3876426 (file GSM3876426_8353_473_CTL_Rep5_Mouse430_2_. CEL.gz).

```
> pm(gse132575[,'GSM3876426.CEL.gz'],
+    featureNames(gse132575)[id5])
             GSM3876426.CEL.gz
1427186_a_at1               275
1427186_a_at2               360
1427186_a_at3               108
1427186_a_at4               297
1427186_a_at5               333
1427186_a_at6               374
1427186_a_at7               226
1427186_a_at8               406
1427186_a_at9               155
1427186_a_at10              285
1427186_a_at11              376
```

c) Obtain the intensity measurements of the MM probes from the microarray in section b).

```
> mm(gse132575[,'GSM3876426.CEL.gz'],
+    featureNames(gse132575)[id5])
             GSM3876426.CEL.gz
1427186_a_at1                84
1427186_a_at2                99
1427186_a_at3                89
1427186_a_at4                91
1427186_a_at5               155
1427186_a_at6               113
1427186_a_at7               100
1427186_a_at8               124
1427186_a_at9                96
1427186_a_at10               79
1427186_a_at11              194
```

d) With an appropriate graph, show the numerical relationship between the intensity measurements from sections b) and c). Are the intensity measurements of the PM probes always higher than those of the corresponding MM probes?



In general, the intensity measurements of the PM probes tend to be higher than those of the MM probes, as observed in the graph. However, in this case, for probe numbers 3 and 9, the intensities are similar.

e) Do all probe pair sets on the microarray have the same number of probe pairs as the studied set? Justify your answer.

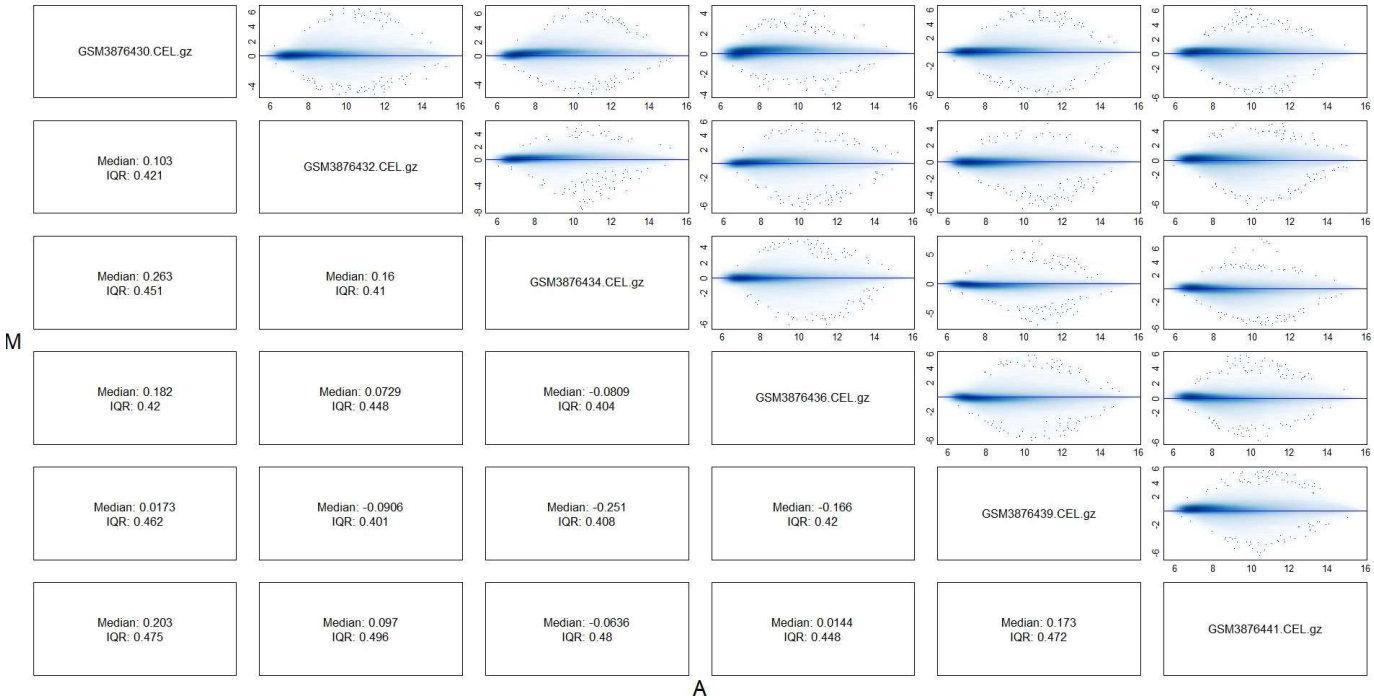
Not all probe pair sets on the microarray have the same number of probe pairs. If we count the number of probe pairs in each set, some have 8, 9, 10, 11, 20, or 21 pairs.

```
> table(table(probeNames(gse132575)))

    8     9    10    11    20    21
    1     5    20 45032    40     3
```

6. The labels "Espontáneo" (spontaneous) and "Tratamiento" (treatment) refer to the samples in the experiment corresponding to mice that developed hepatocellular carcinoma spontaneously or after treatment with Ginkgo biloba extract, respectively. The other samples are controls. By running the following R code (contained in the auxiliary file generarConDNI.R), the object id6 will contain either the string "Spontaneous" or "Treatment" chosen at random:

> set.seed(dni)
> id6 <- sample(c("Espontáneo", "Tratamiento"), size = 1)
> id6
Answer the following sections:

a) Plot an MA diagram comparing the PM probe intensity measurements of all microarray pairs corresponding to the value of id6.



b) By visually examining the diagram from section a), which microarray pair corresponds to the highest absolute value of the median of the M values? What is the value of that median?

Considering both the plot and the obtained median values, the microarray pair with the highest absolute median value is GSM3876430 and GSM38476434, with a median of 0.263.

c) Instead of answering the questions in section b) by visually examining the diagram from section a), write R code to perform the calculations automatically.

With the code created to obtain the microarray pair with the highest absolute median, the same result is obtained as visually, i.e., the pair GSM3876430 and GSM38476434.

```
> for (i in 1:length(gse132575_SPNT)){
+   for (j in 1:length(gse132575_SPNT)){
+     median <- abs(median(log2(pm(gse132575_SPNT[,i]))- log2(pm(gse132575_SPNT[,j]))))
+     medians <- c(medians, median)
+   }
+ }
> max(max(medians))
[1] 0.2630344
```

7. By running the following R code (found in the auxiliary file generarConDNI.R), the object id7 will store the identifier of a probe set. In this code, the dni object is the same as in Questions 5 and 6, and the + symbol indicates a line break for page width.

```
> 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
```

Answer the following sections:

a) Using Bioconductor annotation resources and detailing the R code used, find the correspondence of the identifier stored in id7 with the genes in the Entrez Gene and GenBank NCBI databases (https://www.ncbi.nlm.nih.gov/).

Using the following R code, it was found that the gene associated with id7 (1441604_at) is Esd with id 13885.

```
getSlots("ExpressionSet")
annotation(gse132575)  # Cargar el paquete que nos indique aquí

library(mouse4302.db)  # Cargar paquete
library(annotate)  # Cargar librería recomendada

ls("package:mouse4302.db")  # Listar el contenido del paquete
# Nos quedamos con "mouse4302SYMBOL" para obtener el símbolo del gen y con
# "mouse4302ENTREZID" para obtener el ENTREZ ID del gen

get(id7, envir = mouse4302SYMBOL)  # Obtener símbolo gen
get(id7, envir = mouse4302ENTREZID)  # Obtener entrez id del gen
```

b) Using NCBI resources, answer the following questions: On which chromosome is the gene located? On which strand of the chromosome? What are its coordinates? Finally, summarize the function of the protein encoded by the gene.

Using the NCBI website, it was found that the Esr gene (Entrez gene =13885) is located on chromosome 14 on the positive strand at coordinates 74969591-74988205. The protein it encodes is an S-formylglutathione hydrolase that activates hydrolase activity, acting on ester bonds. It participates in the catabolic processes of formaldehyde in cytoplasmic vesicles and the cytosol.

8. With the subset of microarrays referred to at the end of question 7, i.e., the subset formed by the control samples and those corresponding to the value of id6, create an object named gse132575.sub. Starting from this object:

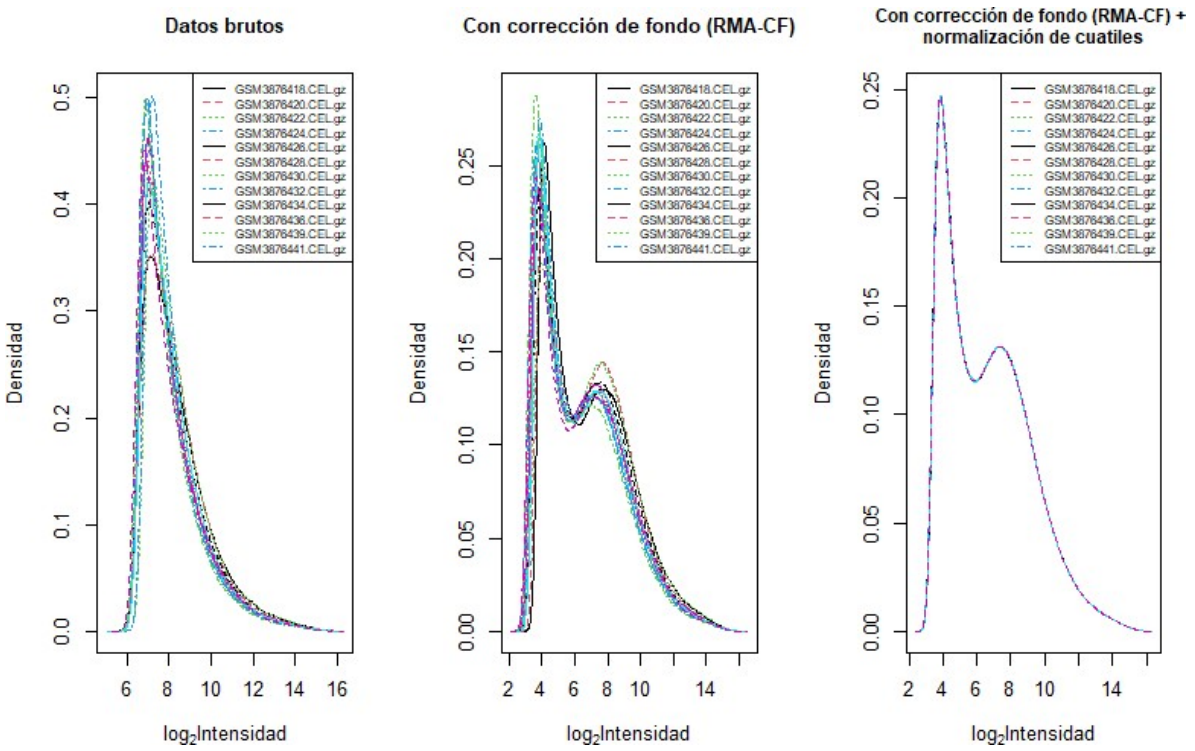
a) Perform background correction considering only the PM probes.

```
########################## Ejercicio 8.a) ####################################
gse132575.sub.corr <- bg.correct(gse132575.sub,
                                 method = 'rma') # Corrección de fondo solo para las sondas PM
```

b) After performing background correction, carry out quantile normalization, considering only the PM probes.

```
########################## Ejercicio 8.b) ####################################
gse132575.sub.corr.cuart <- normalize(gse132575.sub.corr,  # Normalización de cuantiles
                                      method = 'quantiles', # solo para sondas PM
                                      type = 'pmonly')
```

c) Graphically represent estimates of the density of raw, background-corrected, and normalized intensities. Interpret the graphs.



In the previous graphs, it is observed that performing only background correction or quantile normalization halves the density of intensities in the microarrays. However, performing only background correction does not prevent the appearance of samples that differ from the mean (curves with different peaks) but does adjust the effect of nonspecific hybridizations and system noise by reducing the density.

Additionally, performing the different corrections produces a small relief that does not appear in the raw data, which may be due to systematic errors. On the other hand, representing the data with background correction and quantile normalization causes all samples to follow the same curve, resulting in good data for analysis as certain errors due to physical problems with the microarrays, reagent defects, markers, etc., are eliminated.

9. **9. Starting from the object gse132575.sub, preprocess the microarrays in a single step using the rma() function of the affy package. How would the exact same preprocessing be performed by the RMA method with the expresso() function? What is the reason for the differences in computational efficiency of the two functions?**

To do the same preprocessing as is done by the rma() function, the following parameters must be defined in expresso():

```
gse132575.sub.rma <- rma(gse132575.sub)  # Preprocesamiento con rma
gse132575.sub.expr <- expresso(gse132575.sub, bgcorrect.method = 'rma',  # Preprocesamiento expresso
                               normalize.method = 'quantiles', pmcorrect.method = 'pmonly',
                               summary.method = 'medianpolish')
```

Thus, rma() and expresso() produce the same results. However, expresso() needs more time to perform the executions since part of its code is in C language and not just R, as is the case with rma(). For this reason, it has a higher computational overhead and is therefore less efficient.

10. **Starting from the already preprocessed gse132575.sub object created in Question 9, perform a study of differential gene expression between mice from the control samples and those corresponding to the value of id6 (see Question 6). To this end, use the procedures implemented in the multtest and limma packages, among others. Compare the results obtained with those in the article by Kovi et al (2019) associated with the experiment.**

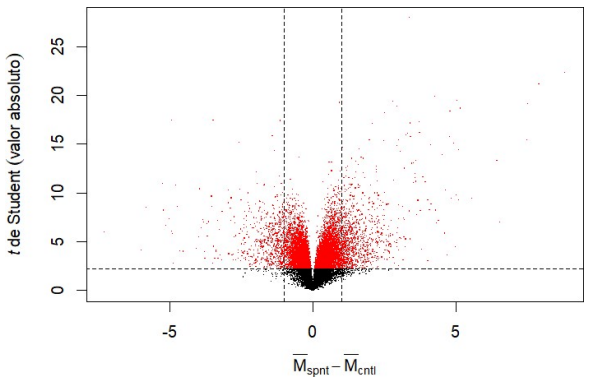
To conduct a study of differential gene expression between mice from the control and spontaneous samples, the following steps must be performed.

First, a volcano plot is obtained for the t-student statistics of the spontaneous samples to see the genes that are differentially expressed. For this, both the mean and the t-student of the spontaneous samples are calculated.

```
spnt <- which(gse132575.sub.rma$treatment == 'SPNT')

M.media <- apply(exprs(gse132575.sub.rma), 1, function(x) mean(x[spnt]) - mean(x[-spnt]))
t.student <- apply(exprs(gse132575.sub.rma), 1, function(x) t.test(x[spnt],
                 x[-spnt])$statistic)
```

```
# Diagrama de volcán
par(mfrow = c(1, 1))
plot(M.media, abs(t.student), xlab = expression(bar(M)[spnt] - bar(M)[salv]),
     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")
```

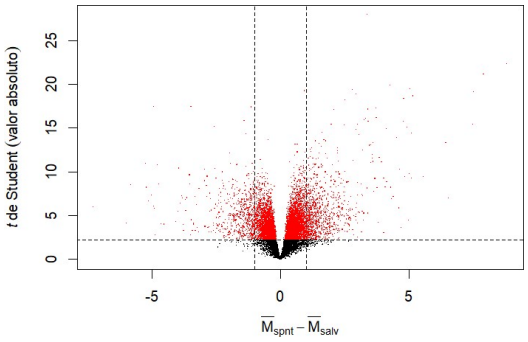
Thus, the volcano plot on the right was obtained, in which those genes that show significantly different expression in red, i.e., a p-value lower than 0.05, are observed.



However, not all these genes are expected to be differentially expressed since some have a small denominator that makes them significant in the plot. Therefore, those genes must be discarded. To do this, only those in which the interquartile range (IQR) of the expression measure across the microarrays is greater than the median of the set of IQRs of all genes and in which the expression measure in more than 10% of microarrays is greater than the median of the set of medians of all are selected.

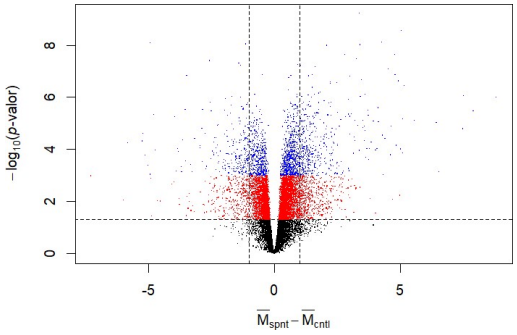
```
library(genefilter)

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

In this way, the number of probe sets was reduced from 45,101 to 17,032, as can be seen in the volcano plot on the right. The main difference between both plots is that there is greater separation in the central area due to filtering out genes that added noise to the analysis. However, the number of genes with significant differential expression does not seem to change.



Apart from the t-student analysis, it is interesting to simultaneously evaluate the magnitude of the fold change with the p-value of each gene, since it is important to detect those genes with a large fold change and statistically significant differences.

As seen in the plot on the right, the genes in blue have a very small p-value, less than 0.001, while those in red have a p-value of 0.05. Thus, the genes of most interest for analysis are those at the extremes and top of the plot (they have fold change).



So far, we have relied on the t-student statistic, but this may poorly approximate the true null distribution. Therefore, a solution is to use the **moderated t-statistic**, a variant of the t-statistic based on empirical Bayes analysis. To use this statistic, the following steps must be performed::

```
pData(gse132575.sub.rma)

matriz.type <- matrix(c(rep(1, 12), ## Efecto medio
                        rep(rep(c(0,1), each = 6), 1)), ## Efecto del espontáneo
                        nrow = 12, ncol = 2, byrow = FALSE)
colnames(matriz.type) <- c("intercept", "Espontáneo")
rownames(matriz.type) <- sampleNames(phenoData(gse132575.sub.rma.filt))
matriz.type

library(limma)
gse132575.type.lmFit <- lmFit(gse132575.sub.rma.filt, design = matriz.type)
gse132575.type.eBayes <- eBayes(gse132575.type.lmFit)  # Obtener estadístico t-moderado
                                                       # para hipótesis nula en la que
                                                       # el test de igualdad de medias de
                                                       # los grupos control y espontáneo
```

With the above steps, a linear regression model is fitted to obtain the moderated t-statistic for the null hypothesis, i.e., for the test of equality of means of the control and spontaneous groups, in addition to obtaining the number of p-values less than 0.05 and 0.001.

```
> head(gse132575.type.eBayes$t)  # t-moderado
           intercept Espontáneo
1415670_at 161.52477  4.5975586
1415672_at 200.06132  3.0121672
1415673_at  49.05218  5.3750622
1415677_at  74.31361 -0.8352755
1415680_at  92.65659  1.3024333
1415682_at 106.88323 -1.7496049
> head(gse132575.type.eBayes$p.value)  # p-value
              intercept   Espontáneo
1415670_at 7.455558e-23 0.0005001577
1415672_at 4.625497e-24 0.0100039528
1415673_at 3.862336e-16 0.0001265169
1415677_at 1.778167e-18 0.4186524034
1415680_at 1.016056e-19 0.2153746700
1415682_at 1.590855e-20 0.1037427851
>
> sum(gse132575.type.eBayes$p.value[, "Espontáneo"] < 0.05)  # 9035 p-values menores que 0.05
[1] 9035
> sum(gse132575.type.eBayes$p.value[, "Espontáneo"] < 0.001) # 2810 p-values menores que 0.001
[1] 2810
```

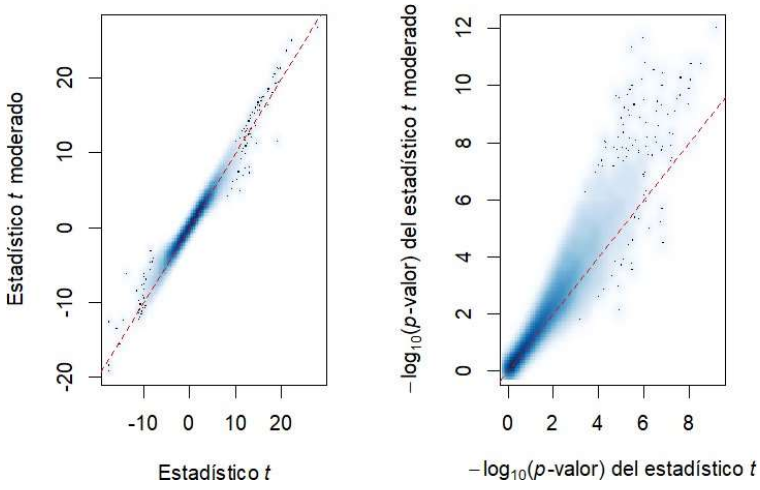
The specific identity of the probe sets, as well as their ordered p-values, are shown in the following image:

```
> 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
```

Once the results for both the t-student and moderated t-statistics have been seen, a comparison between the two is made to see if the latter actually improves the data.

```
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")
```

In the image on the right, it is observed that the values of the moderated t-statistic are less variable than those of the classic t, and the p-values are smaller. Thus, we can conclude that the analysis with the moderated t yields better results than with the t-student for continuing the study.

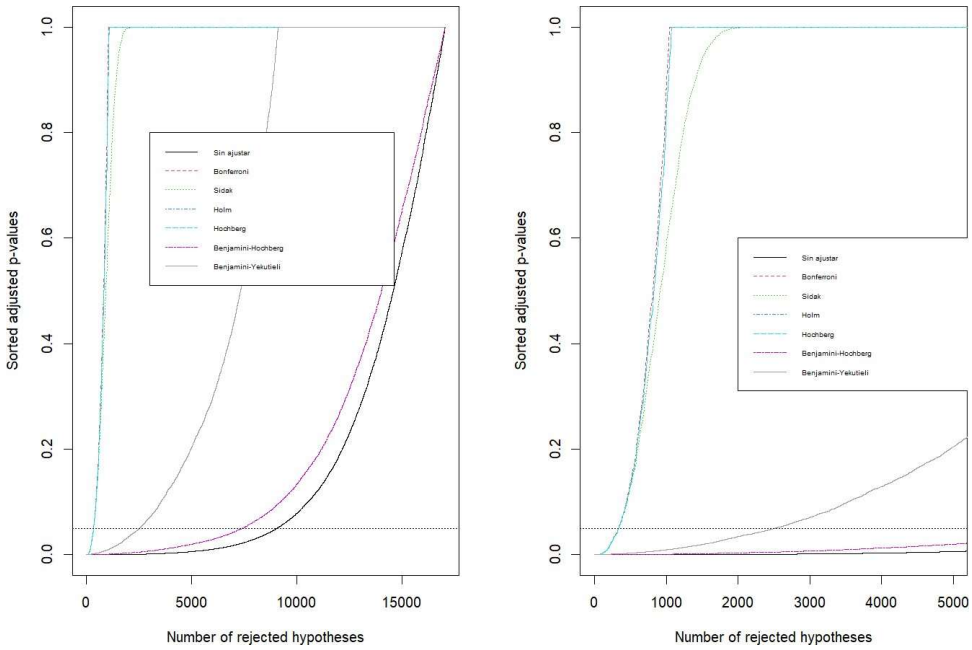


Now that we have seen that the moderated t-statistic is better than the t-student, we will perform a multiplicity analysis of the contrasts, adjusting the p-values using different procedures: Bonferroni, Sidak, Holm, Hochberg, Benjamini-Hochberg, and Benjamini-Yekutieli.

In the following image, the code used and the different p-values obtained for the various statistical procedures are shown.

```
> 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[1:10,] # Primeras 10 líneas de los p-values ajustados
             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
```

However, these values by themselves do not tell us much, so we will plot them to see the differences between them.



n the previous graphs, the relationship between the number of rejected hypotheses and the type I error rate is observed. It is seen that, except for the Benjamini-Hochberg adjustment, the rest of the procedures significantly reduce the number of differentially expressed genes. For example, the Hochberg adjustment reduces this number to about 450. However, this is far from the unadjusted data, so much information is lost, i.e., many genes. Therefore, to obtain differentially expressed genes without losing too much information, we will use the p-values obtained with the Benjamini-Hochberg procedure in the next steps, so we will have about 7,000 genes instead of 10,000.

Once the adjusted p-values are obtained, we will get the Affymetrix identifiers of the probe sets. In this way, all probe sets are obtained (a total of 7,397), but we do not know which genes they correspond to.

```
> featureNames(gse132575.sub.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"
   [7] "1419136_at"   "1424713_at"   "1425120_x_at" "1418572_x_at" "1438596_at"   "1439808_at"
  [13] "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"
  [25] "1435758_at"   "1417821_at"   "1449152_at"   "1452934_at"   "1429899_at"   "1429159_at"
  [31] "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"
  [43] "1437199_at"   "1448250_at"   "1417822_at"   "1451798_at"   "1430172_a_at" "1417125_at"
  [49] "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"
  [61] "1451230_a_at" "1424138_at"   "1422286_a_at" "1457403_at"   "1455390_at"   "1418918_at"
  [67] "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"
  [79] "1421223_a_at" "1416410_at"   "1448111_at"   "1448605_at"   "1420385_at"   "1429527_a_at"
```

Therefore, we must obtain their correspondence with the standard nomenclature systems of the NCBI. To do this, the following steps are performed:

```
> library(annotate)
> annotation(gse132575.sub.rma.filt)
[1] "mouse4302"
> library(mouse4302.db)
> ls("package:mouse4302.db")
 [1] "mouse4302"              "mouse4302.db"           "mouse4302_dbconn"       "mouse4302_dbfile"
 [5] "mouse4302_dbInfo"       "mouse4302_dbschema"     "mouse4302ACCNUM"        "mouse4302ALIAS2PROBE"
 [9] "mouse4302CHR"           "mouse4302CHRLENGTHS"    "mouse4302CHRLOC"        "mouse4302CHRLOCEND"
[13] "mouse4302ENSEMBL"       "mouse4302ENSEMBL2PROBE" "mouse4302ENTREZID"      "mouse4302ENZYME"
[17] "mouse4302ENZYME2PROBE"  "mouse4302GENENAME"      "mouse4302GO"            "mouse4302GO2ALLPROBES"
[21] "mouse4302GO2PROBE"      "mouse4302MAPCOUNTS"     "mouse4302MGI"           "mouse4302MGI2PROBE"
[25] "mouse4302ORGANISM"      "mouse4302ORGPKG"        "mouse4302PATH"          "mouse4302PATH2PROBE"
[29] "mouse4302PFAM"          "mouse4302PMID"          "mouse4302PMID2PROBE"    "mouse4302PROSITE"
[33] "mouse4302REFSEQ"        "mouse4302SYMBOL"        "mouse4302UNIPROT"
```

Thus, with the mouse4302.db package and the getSymbol() function, we can obtain the symbols of the most relevant genes from the NCBI Gene database.

```
> getSYMBOL(featureNames(gse132575.sub.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
       "Mtmr11"            "Afp"            "Bex1"          "Tspan8"          "Scd2"          "Cdh1"
     1419136_at      1424713_at     1425120_x_at     1418572_x_at      1438596_at      1439808_at
      "Akr1c18"        "Calml4"       "Ifi27l2b"       "Tnfrsf12a"       "Hectd2os"        "Ipcef1"
     1443866_at      1460329_at      1418571_at     1420018_s_at      1415823_at     1436879_x_at
        "Lrtm1"       "B4galt6"      "Tnfrsf12a"          "Tspan8"          "Scd2"           "Afp"
   1416473_a_at      1448469_at      1416474_at      1415822_at     1453181_x_at      1416808_at
       "Igdcc4"          "Nid1"        "Igdcc4"          "Scd2"        "Plscr1"          "Nid1"
     1435758_at      1417821_at      1449152_at      1452934_at      1429899_at      1429159_at
      "B4galt6"    "D17H6S56E-5"        "Cdkn2b"          "Tmc5"              NA         "Itih5"
     1415698_at      1428066_at      1416646_at      1436212_at      1441946_at      1420017_at
        "Golm1"       "Ccdc120"           "Afp"        "Tmem71"         "Itih5"        "Tspan8"
     1423515_at     1427932_s_at      1451486_at     1431292_a_at     1439440_x_at      1434726_at
        "Scn8a" "1200015M12Rik"       "Slc46a3"          "Twf2"          "Twf2"      "Catsperd"
     1437199_at      1448250_at      1417822_at      1451798_at     1430172_a_at      1417125_at
        "Dusp5"          "Clmp"    "D17H6S56E-5"         "Il1rn"              NA              NA
     1448350_at      1449375_at      1436591_at      1420908_at      1418345_at      1449049_at
          "Asl"        "Ces2a"         "Vsig10"         "Cd2ap"              NA          "Tlr1"
```

Once we have the most relevant genes (although some have not been identified, so they appear as NA), we will compare them with the 20 genes obtained in the experiment associated with the data. To do this, we will obtain the list of genes as follow:

```
> for (i in 1:20)
+   print(get(featureNames(gse132575.sub.rma.filt[gse132575.type.pval$index[
+     gse132575.type.pval$adjp[, "BH"] < 0.05], ])[i],
+   env = mouse4302SYMBOL))
[1] "Mtmr11"
[1] "Afp"
[1] "Bex1"
[1] "Tspan8"
[1] "Scd2"
[1] "Cdh1"
[1] "Akr1c18"
[1] "Calml4"
[1] "Ifi27l2b"
[1] "Tnfrsf12a"
[1] "Hectd2os"
[1] "Ipcef1"
[1] "Lrtm1"
[1] "B4galt6"
[1] "Tnfrsf12a"
[1] "Tspan8"
[1] "Scd2"
[1] "Afp"
[1] "Igdcc4"
[1] "Nid1"
```

In this way, comparing the first 20 genes, the only genes that appear both in the experiment and in this analysis are Tspan8, Lrtm1, and B4galt6. This may be because the authors used other criteria or different parameters when optimizing the analysis (for example, they used a fold change of 2).

It should be noted that Tspan8 is a tetraspanin that activates integrin binding activity, acting in the regulation of gene expression and spermatogenesis. On the other hand, Lrtm1 encodes leucine-rich repeats and transmembrane domains, allowing heparin binding activity and acting in the positive regulation of synaptic assembly. Finally, B4galt6 encodes a beta 1,4-galactosyltransferase that participates in the biosynthetic process of gangliosides through lactosylceramide and neurogenesis.