

Unidad 2. Estudios Transcriptómicos Masivos: Análisis de Microarrays

Francisco J. Romero-Campero - email: fran@us.es - web: <http://www.cs.us.es/~fran>

Dpto CCIA - Unidad de Desarrollo Vegetal del IBVF

Máster Genética Molecular y Biotecnología

Universidad de Sevilla

Lectura de los datos brutos.

El paquete **affy** proporciona las funciones básicas para manipular datos de microarrays de **affymetrix**.

```
library(affy)
```

```
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
## 
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## 
## The following object is masked from 'package:stats':
## 
##     xtabs
## 
## The following objects are masked from 'package:base':
## 
##     anyDuplicated, append, as.data.frame, as.vector, cbind,
##     colnames, do.call, duplicated, eval, evalq, Filter, Find, get,
##     intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unlist, unsplit
## 
## Loading required package: Biobase
## Welcome to Bioconductor
## 
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

Los datos de microarray que vamos a analizar corresponden al estudio **Long et al. (2010) The bHLH transcription factor POPEYE regulates response to iron deficiency in Arabidopsis roots**. Plant Cell 22(7):2219-36. Estos datos se encuentran disponibles de forma pública en la base de datos **Gene Expression Omnibus (GEO)** identificados con el número de acceso **GSE21582**.

Fijamos el espacio de trabajo a la carpeta que contenga los datos de microarrays. Siempre es aconsejable guardar en la misma carpeta el script de análisis y los datos analizados. Si se sigue este consejo basta fijar el espacio de trabajo a la localización donde se encuentra el script o source file. Para ellos usar el menu **Session** y la opción **Set Working Directory** seguida de **To Source File Location**.

Utilizamos la función **ReadAffy** para cargar o leer los datos brutos de microarrays (fluorescencia sin procesar) contenidos en la correspondiente carpeta. Los datos brutos deben estar contenidos en ficheros con **formato CEL**. Se recomienda fijar el argumento **verbose** a TRUE para que muestra información por pantalla sobre la lectura de los datos.

```
microarray.raw.data <- ReadAffy(verbose=TRUE)
```

```
## 1 reading /home/fran/Dropbox/asignaturas/master_genetica_bioteecnologia/clases/microarrays/GSE21582_R
## Reading in : /home/fran/Dropbox/asignaturas/master_genetica_bioteecnologia/clases/microarrays/GSE21582_R
```

Si evaluamos la variable que almacena los datos brutos obtenemos información sobre el tamaño del microarray, el diseño de la placa, el número de muestras y el número de transcritos (o sondas) que contiene la placa correspondiente.

```
microarray.raw.data
```

```
## Creating a generic function for 'nchar' from package 'base' in package 'S4Vectors'
## AffyBatch object
## size of arrays=712x712 features (20 kb)
## cdf=ATH1-121501 (22810 affyids)
## number of samples=8
## number of genes=22810
## annotation=ath1121501
## notes=
```

Si sólo estamos interesados en obtener el diseño de placa podemos usar la función **cdfName**.

```
cdfName(microarray.raw.data)
```

```
## [1] "ATH1-121501"
```

Control de la Calidad.

Los paquetes **simpleaffy** y **affyPLM** contienen funciones para el análisis de la calidad de microarrays.

```
library(simpleaffy)

## Loading required package: genefilter
##
## Attaching package: 'genefilter'
##
## The following object is masked from 'package:base':
##
##     anyNA
##
## Loading required package: gcrma
```

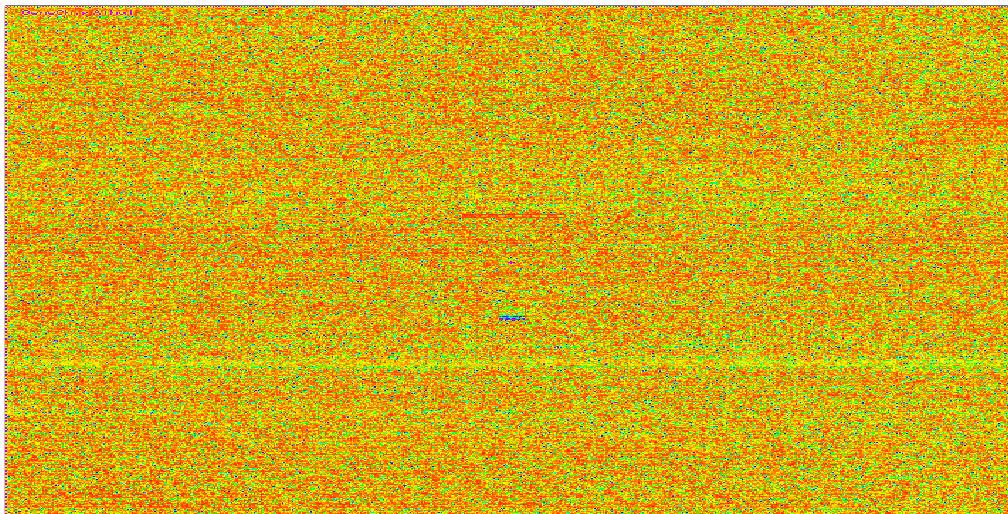
```
library(affyPLM)
```

```
## Loading required package: preprocessCore
```

La función **image** nos permite visualizar una reproducción de la foto tomada por el scanner de la fluorescencia de la placa de microarray. Usando esta visualización podemos determinar si se han producido daños físicos durante el manejo de la placa de microarrays.

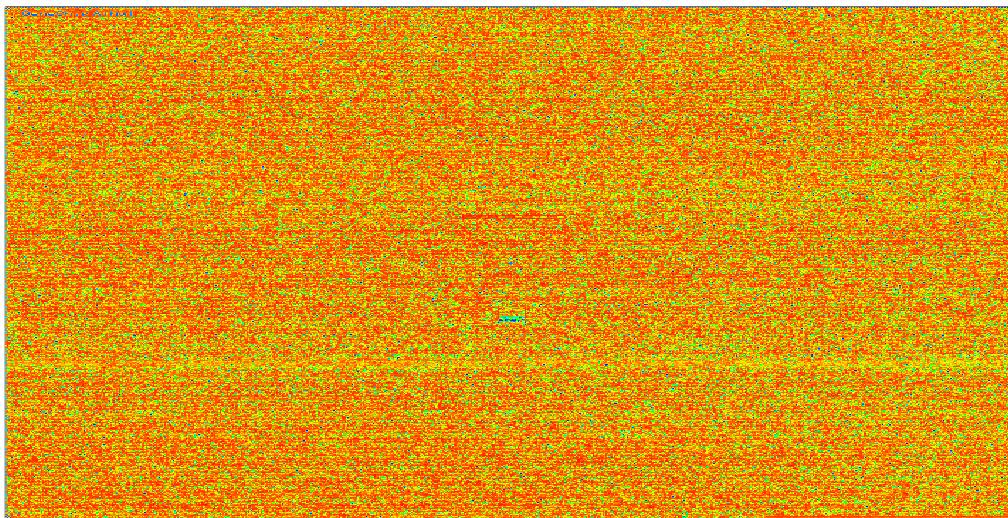
```
image(microarray.raw.data[,1],col=rainbow(100))
```

GSM535984.CEL.gz



```
image(microarray.raw.data[,2],col=rainbow(100))
```

GSM535985.CEL.gz



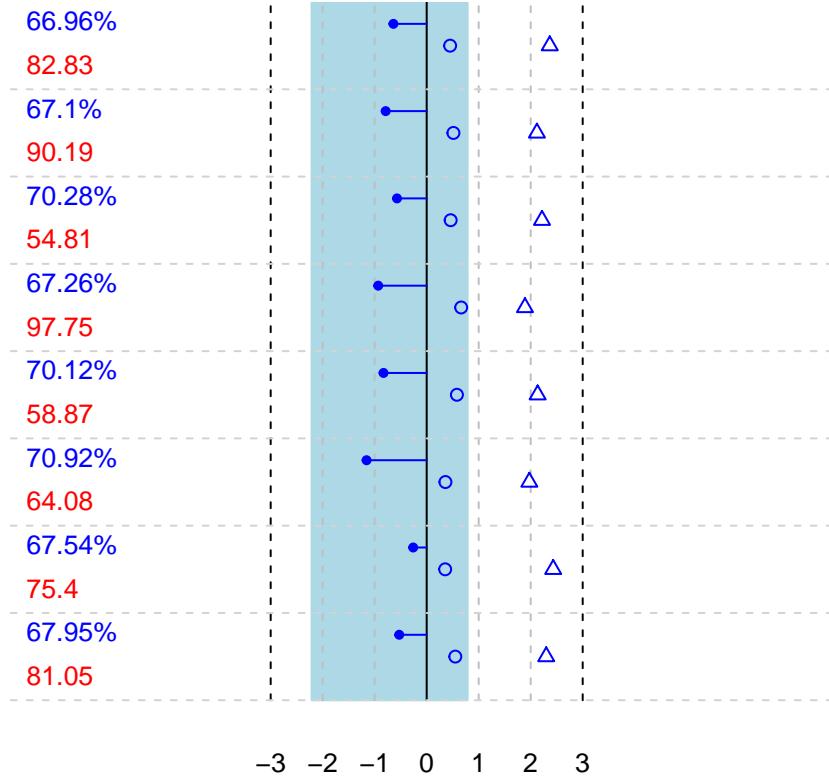
La función **qc** aplicada sobre los datos crudos nos permite realizar un análisis de la calidad. Un resumen gráfico del análisis de la calidad se puede representar con la función **plot**.

```
quality.analysis <- qc(microarray.raw.data)
plot(quality.analysis)
```

△ actin3/actin5
○ gapdh3/gapdh5

GSM535991.CEL.gz	66.96%	82.83
GSM535990.CEL.gz	67.1%	90.19
GSM535989.CEL.gz	70.28%	54.81
GSM535988.CEL.gz	67.26%	97.75
GSM535987.CEL.gz	70.12%	58.87
GSM535986.CEL.gz	70.92%	64.08
GSM535985.CEL.gz	67.54%	75.4
GSM535984.CEL.gz	67.95%	81.05

QC Stats



Cada fila del gráfico QC representa una muestra. El primer valor numérico se llama *porcentaje de detección* y corresponde al porcentaje de sondas para el cual se ha detectado fluorescencia. Se espera que estos valores sean

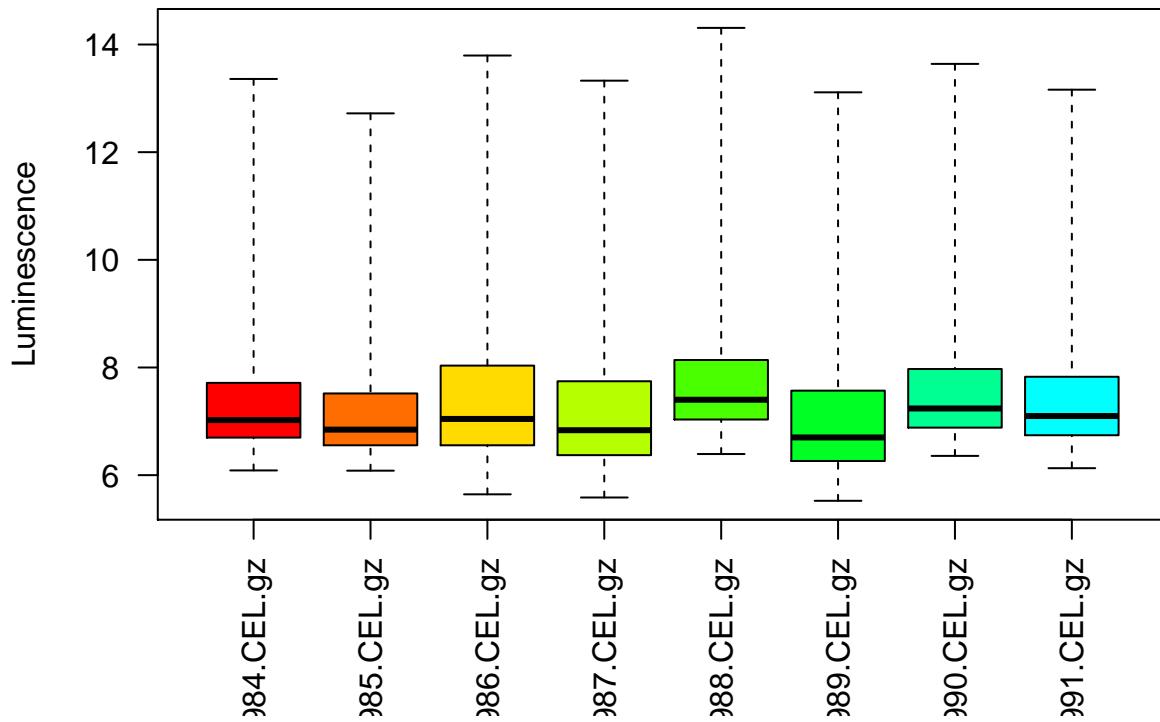
similares en todas las muestras. El segundo valor numérico hace referencia a la *fluorescencia del fondo* medida en posiciones del microarray donde no se encuentra ninguna sonda de nucleótidos. Esta fluorescencia de fondo mide la hibridación inespecífica de cDNA a la superficie del microarray. Típicamente la fluorescencia del fondo aparece marcada como problemática ya que presenta valores dispares en las distintas muestras. Sin embargo, este valor puede considerarse como un “blanco” que es fácilmente corregible durante el preprocesamiento de los datos. Los símbolos de círculo y triángulo representan la proporción de fluorescencia entre los extremos 3' y 5' de las sondas control para la actina y el GAPDH. Debido a los procesos de degradación del RNA y de síntesis del cDNA a partir del RNA siempre se espera que el extremo 3' de los transcritos esté sobrerepresentado con respecto al extremo 5'. Sin embargo, si se han dado procesos masivos de degradación del RNA y de incompletitud durante la síntesis del cDNA el extremo 3' estará muy altamente sobrerepresando con respecto al extremo 5' lo que indicará problemas en la calidad.

Preprocesamiento y Estimación de los Niveles de Expresión génicos.

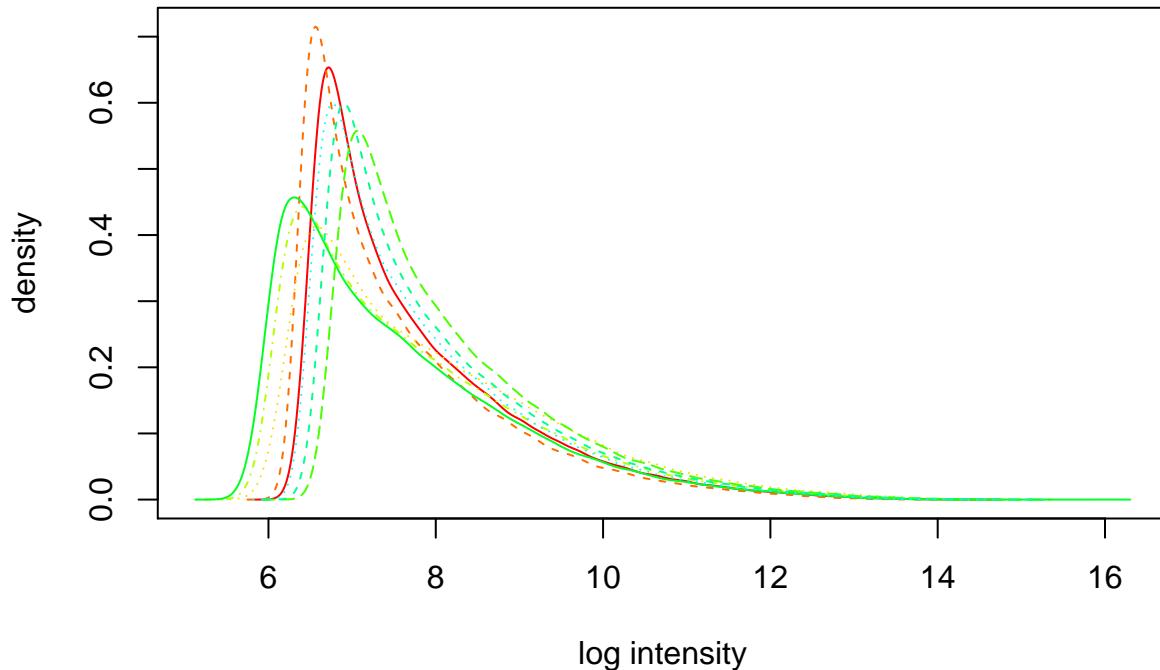
En todo experimento existen dos fuentes de variabilidad. Por una parte tendremos la variabilidad biológica que estamos interesados en estudiar. Por otra parte tendremos a variabilidad experimental producida por factores técnicos. La variabilidad experimental se suele denominar ruido y no es deseada. Típicamente en todo análisis de datos se realiza un preprocesamiento de los datos brutos que busca eliminar el ruido o variabilidad experimental mientras se mantiene la variabilidad biológica.

Antes de realizar ningún preprocesamiento de los datos comprabamos si son comparables utilizando como descriptores globales de la distribución de los niveles de expresión boxplot (cajas y bigotes) e histogramas.

```
boxplot(microarray.raw.data,col=rainbow(14),las=2,ylab="Luminescence")
```



```
hist(microarray.raw.data,col=rainbow(14))
```



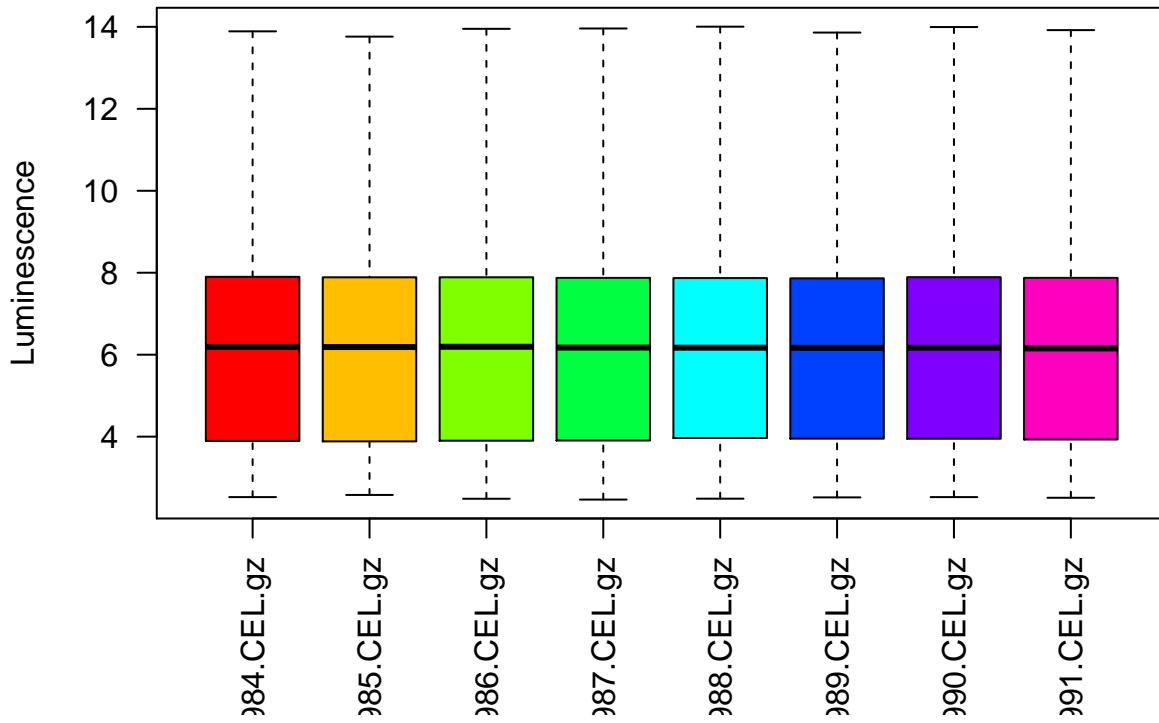
Para lograr que todos los microarrays sean comparables utilizamos el **Robust Multiarray Analysis** que realiza corrección de la fluorescencia del fondo, normalización, sumación y estimación de los niveles de expresión en **log2**. Este algoritmo realiza el correspondiente preprocesamiento de los datos brutos de microarrays.

```
microarray.processed.data <- rma(microarray.raw.data)
```

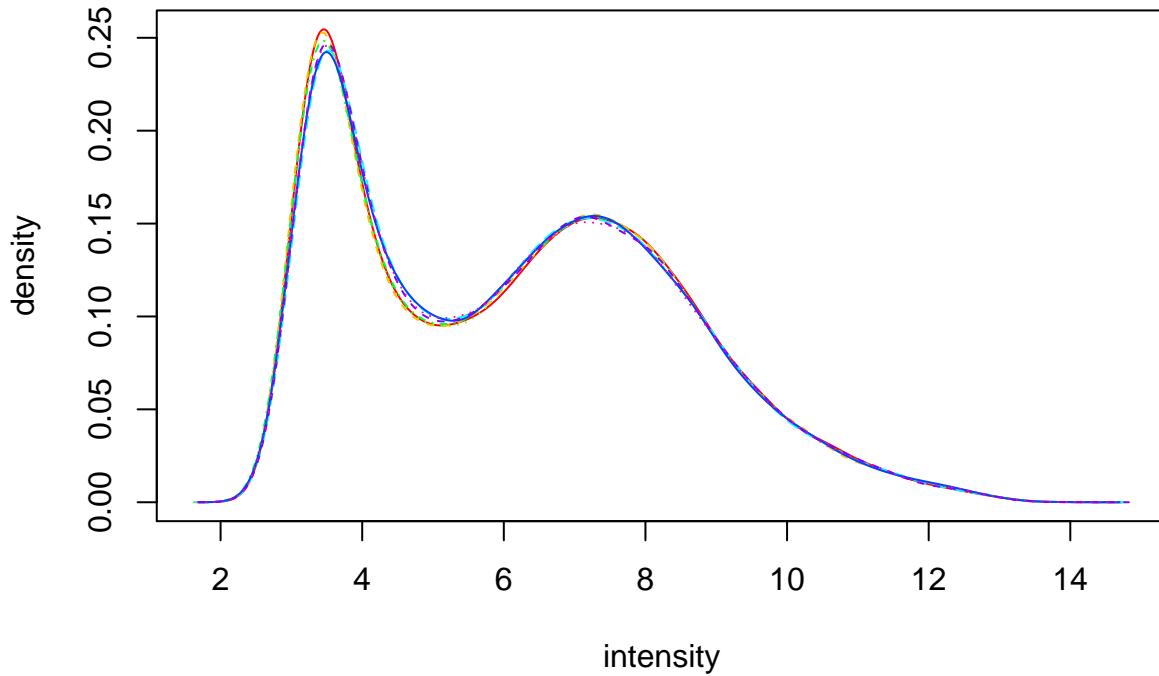
```
## Background correcting
## Normalizing
## Calculating Expression
```

Una vez preprocesados los datos comprobamos si son comparables.

```
boxplot(microarray.processed.data, col=rainbow(8), las=2, ylab="Luminescence")
```



```
hist(microarray.processed.data, col=rainbow(8))
```



La estimación de los niveles de expresión realizada por RMA puede extraerse como una matriz usando la función **exprs**. Las columnas de esta matriz corresponden a los ficheros de datos brutos .CEL. Por lo tanto, las columnas representan muestras y las filas sondas o transcritos.

```
expression.level <- exprs(microarray.processed.data)
head(expression.level)
```

```

##          GSM535984.CEL.gz GSM535985.CEL.gz GSM535986.CEL.gz
## 244901_at      4.352067      4.512782      3.890223
## 244902_at      4.958582      5.029459      4.878251
## 244903_at      6.021612      6.688565      5.665589
## 244904_at      5.403892      5.552970      5.408837
## 244905_at      3.753489      3.851467      3.884735
## 244906_at      6.080130      6.207847      6.174497
##          GSM535987.CEL.gz GSM535988.CEL.gz GSM535989.CEL.gz
## 244901_at      4.052286      4.022519      4.479939
## 244902_at      5.024956      4.832249      5.269997
## 244903_at      6.499780      5.817047      6.537776
## 244904_at      5.459914      5.245839      5.351673
## 244905_at      3.801065      3.850620      4.139859
## 244906_at      6.179443      6.102840      6.562424
##          GSM535990.CEL.gz GSM535991.CEL.gz
## 244901_at      4.089720      4.610809
## 244902_at      4.738543      5.131795
## 244903_at      5.516187      6.453257
## 244904_at      5.498372      5.564439
## 244905_at      4.032634      3.859356
## 244906_at      6.123178      6.277115

```

Por defecto, el nombre de las columnas coincide con el nombre del correspondiente fichero CEL. Estos nombres no son informativos y conducen fácilmente a error. Por lo tanto, renombramos apropiadamente las columnas con el nombre de las muestras.

```

sampleID <- c("WT_with_Fe_1", "WT_with_Fe_2", "WT_no_Fe_1", "WT_no_Fe_2", "pye_with_Fe_1", "pye_with_Fe_2", "pye_no_Fe_1", "pye_no_Fe_2")
colnames(expression.level) <- sampleID
head(expression.level)

```

```

##          WT_with_Fe_1 WT_with_Fe_2 WT_no_Fe_1 WT_no_Fe_2 pye_with_Fe_1
## 244901_at      4.352067      4.512782      3.890223      4.052286      4.022519
## 244902_at      4.958582      5.029459      4.878251      5.024956      4.832249
## 244903_at      6.021612      6.688565      5.665589      6.499780      5.817047
## 244904_at      5.403892      5.552970      5.408837      5.459914      5.245839
## 244905_at      3.753489      3.851467      3.884735      3.801065      3.850620
## 244906_at      6.080130      6.207847      6.174497      6.179443      6.102840
##          pye_with_Fe_2 pye_no_Fe_1 pye_no_Fe_2
## 244901_at      4.479939      4.089720      4.610809
## 244902_at      5.269997      4.738543      5.131795
## 244903_at      6.537776      5.516187      6.453257
## 244904_at      5.351673      5.498372      5.564439
## 244905_at      4.139859      4.032634      3.859356
## 244906_at      6.562424      6.123178      6.277115

```

Calculamos los valores medios de expresión para cada genotipo/condición sumando las correspondientes columnas y dividiendo por el número de réplicas.

```

wt.with.fe <- (expression.level[, "WT_with_Fe_1"] + expression.level[, "WT_with_Fe_2"])/2
wt.no.fe <- (expression.level[, "WT_no_Fe_1"] + expression.level[, "WT_no_Fe_2"])/2
pye.with.fe <- (expression.level[, "pye_with_Fe_1"] + expression.level[, "pye_with_Fe_2"])/2
pye.no.fe <- (expression.level[, "pye_no_Fe_1"] + expression.level[, "pye_no_Fe_2"])/2

```

Creamos una matriz que contenga por columna la expresión media para cada condición o genotipo. Nombramos las filas con el nombre de las sondas (transcritos) y la columnas con la condición o genotipo.

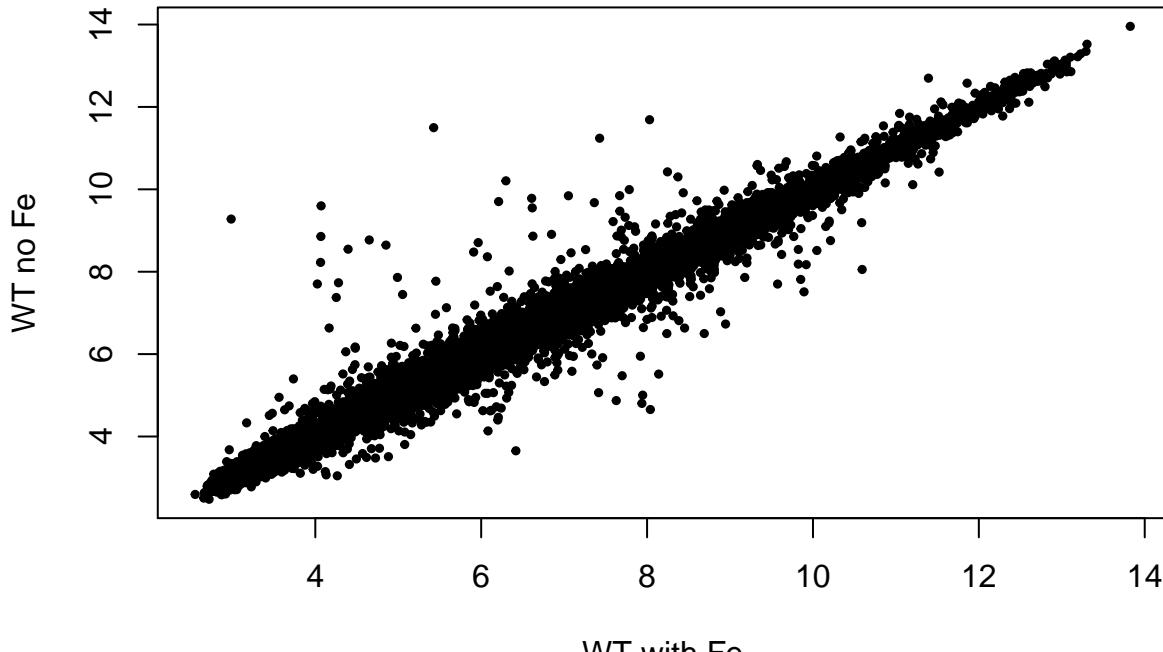
```
mean.expression <- matrix(c(wt.with.fe,wt.no.fe,pye.with.fe,pye.no.fe),ncol=4)
conditions.id <- c("WT_with_Fe","WT_no_Fe","pye_with_Fe","pye_no_Fe")
rownames(mean.expression) <- names(wt.with.fe)
colnames(mean.expression) <- conditions.id
head(mean.expression)

##           WT_with_Fe WT_no_Fe pye_with_Fe pye_no_Fe
## 244901_at    4.432425 3.971254   4.251229  4.350265
## 244902_at    4.994020 4.951604   5.051123  4.935169
## 244903_at    6.355089 6.082685   6.177411  5.984722
## 244904_at    5.478431 5.434376   5.298756  5.531406
## 244905_at    3.802478 3.842900   3.995240  3.945995
## 244906_at    6.143988 6.176970   6.332632  6.200146
```

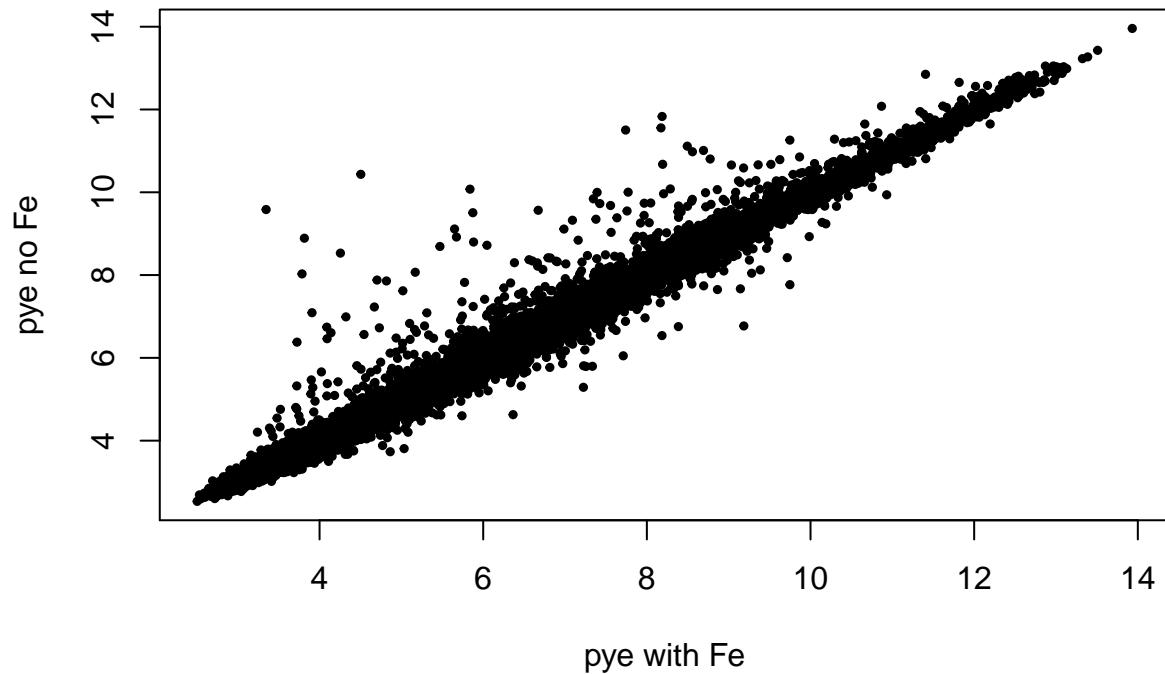
Para realizar una previsualización comparativa entre los transcriptomas de dos genotipos/condiciones diferentes se suelen usar gráficos de dispersión o scatterplots. Este tipo de gráficos nos permite obtener una visión global de la comparación entre genotipos/condiciones. Cada punto representa un transcripto, la coordenada x representa el nivel expresión en el primer genotipo/condición mientras que la coordenada y representa el nivel de expresión en el segundo genotipo/condición. Por lo tanto, genes cuyos transcriptos estén representados por puntos localizados en la diagonal no se expresan de forma diferencial. Genes cuyos transcriptos estén representados por puntos localizados en la parte superior a la diagonal estarán activados en la segunda condición/genotipo con respecto a la primera. Genes cuyos transcriptos estén representados por puntos localizados en la parte inferior a la diagonal estarán reprimidos en la segunda condición/genotipo con respecto a la primera.

Gráficos de dispersión o scatterplots que muestren una gran dispersión indica efectos notables en el transcriptoma. Mientras que poca dispersión indica efectos leves. Por lo tanto, en los casos donde se observe mucha dispersión se suelen fijar criterios restrictivos para la determinación de genes expresados de forma diferencial.

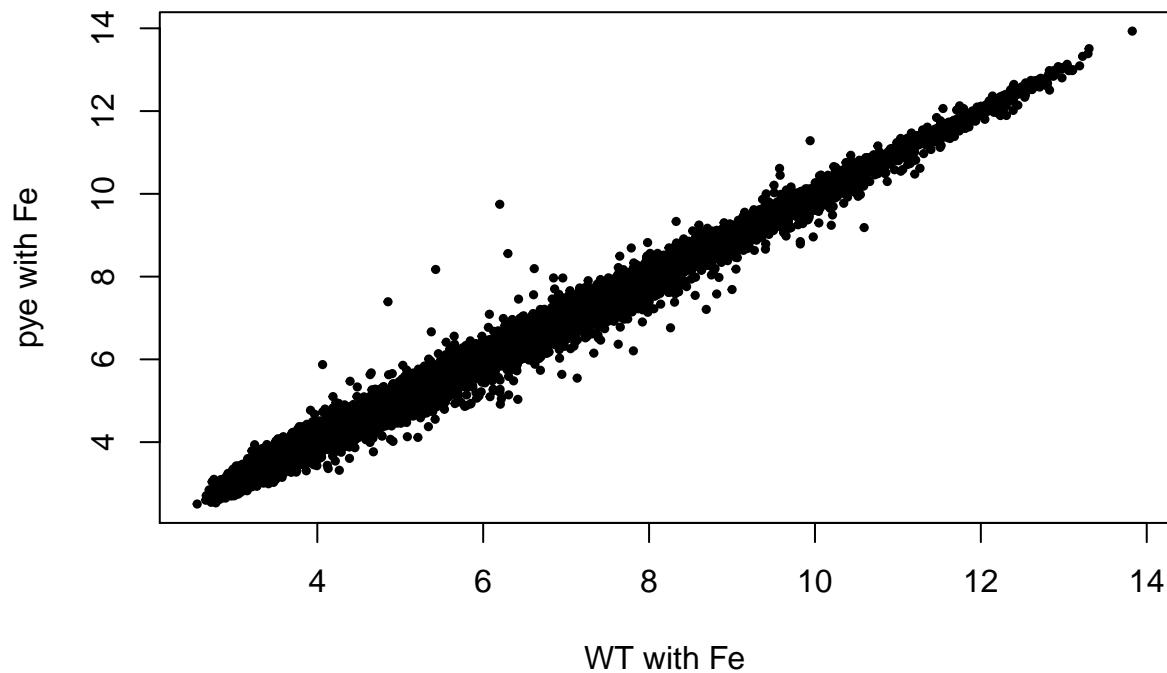
```
plot(wt.with.fe,wt.no.fe,xlab="WT with Fe",ylab="WT no Fe",pch=19,cex=0.5)
```



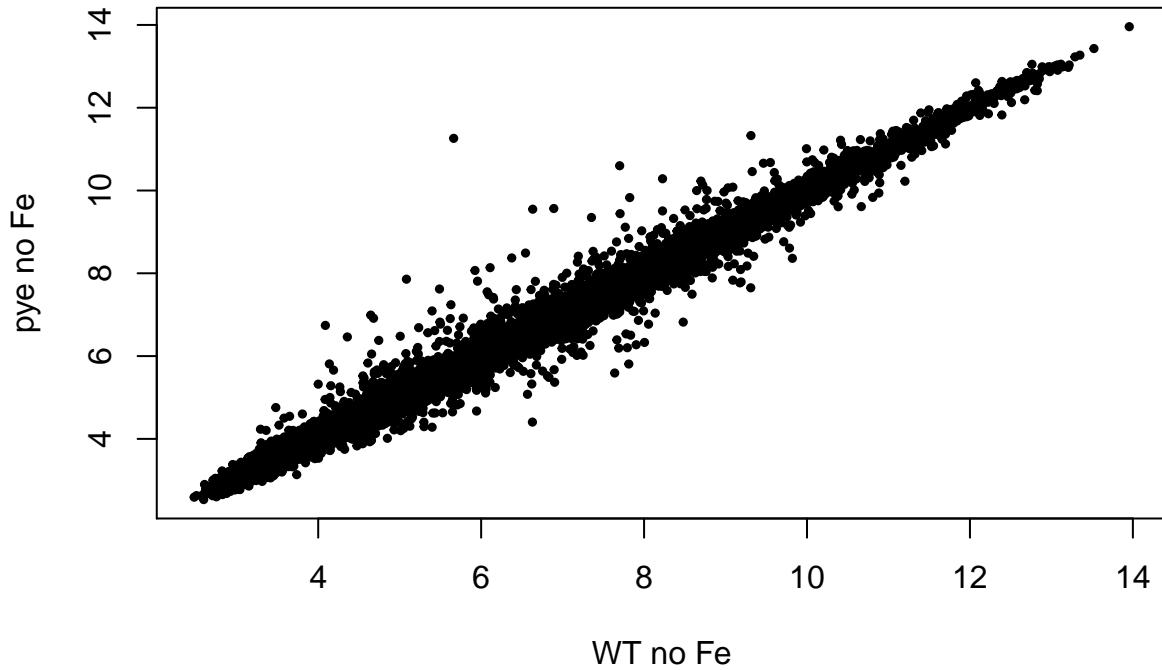
```
plot(pye.with.fe,pye.no.fe,xlab="pye with Fe",ylab="pye no Fe",pch=19,cex=0.5)
```



```
plot(wt.with.fe,pye.with.fe,xlab="WT with Fe",ylab="pye with Fe",pch=19,cex=0.5)
```



```
plot(wt.no.fe,pye.no.fe,xlab="WT no Fe",ylab="pye no Fe",pch=19,cex=0.5)
```



Selección de Genes Expresados de Forma Diferencial.

El paquete **limma** (Linear Models for Microarray Analysis) proporciona las funciones necesarias para determinar los genes expresados de forma diferencial (DEGs).

```
library(limma)
```

```
##  
## Attaching package: 'limma'  
##  
## The following object is masked from 'package:BiocGenerics':  
##  
##     plotMA
```

Generamos una matriz que represente el diseño experimental. Cada condición/genotipo se identifica con un número entero. En nuestro caso tenemos seis condiciones/genotipos: tipo silvestre con hierro (1), tipo silvestre sin hierro (2), popeye con hierro (3) y popeye sin hierro (4). Seguidamente, se marca cada muestra con el número que identifica cada condición/genotipo. De esta forma, cada identificador se repite el número de réplicas de la condición/genotipo correspondiente (no necesariamente de forma consecutiva). Usamos la función **model.matrix** con la sintaxis siguiente para especificar el diseño experimental. Las columnas de esta matrix se nombran con los nombres de cada condición, WT_with_Fe, WT_no_Fe, pye_with_Fe y pye_no_Fe. Muy importante, a partir de este punto sólo podremos referirnos a cada condición/genotipo usando exactamente los nombres anteriores.

```
experimental.design <- model.matrix(~ -1+factor(c(1,1,2,2,3,3,4,4)))  
colnames(experimental.design) <- c("WT_with_Fe", "WT_no_Fe", "pye_with_Fe", "pye_no_Fe")
```

A continuación, ajustamos la estimación de los niveles de expresión de cada gen a un modelo lineal teniendo en cuenta el diseño experimental. Este paso fundamentalmente se calcula la media de las réplicas en cada condición.

```
linear.fit <- lmFit(expression.level, experimental.design)
```

Para determinar los genes expresados de forma diferencial debemos especificar los contrastes (comparaciones entre condiciones) que se van a considerar. Típicamente se suelen considerar comparaciones entre condiciones donde sólo se ha dado un cambio genotípico o de condición para poder establecer de forma inequívoca una relación causal. En nuestro ejemplo buscamos determinar los genes que ven su expresión afectada por la deficiencia de hierro en el genotipo WT y pye. Además queremos determinar el efecto del gen POPEYE. Por lo tanto, realizaremos todas las siguientes comparaciones:

- WT no Fe / WT with Fe
- pye no Fe / pye with Fe
- WT with Fe / pye with Fe
- WT no Fe / pye no Fe

Para especificar los contrastes a realizar utilizamos la función *makeContrasts* que recibe como entrada los contrastes a realizar separados por comas y especificados con los nombres de las dos condiciones correspondientes separadas por un guión -. También recibe el argumento levels, un vector con el nombre de las condiciones:

```
contrast.matrix <- makeContrasts(WT_no_Fe-WT_with_Fe,pye_no_Fe-pye_with_Fe,WT_with_Fe-pye_with_Fe,WT_no_
```

Calculamos el fold-change y los p-valores correspondientes para cada gen en cada uno de los contrastes especificados utilizando las funciones *contrasts.fit* y *eBayes*.

```
contrast.linear.fit <- contrasts.fit(linear.fit, contrast.matrix)
contrast.results <- eBayes(contrast.linear.fit)
```

Con la función *topTable* podemos obtener un marco de datos con información sobre la expresión diferencial de los genes. Esta función recibe como entrada la salida de *eBayes*, el número de genes a mostrar (argumento number) y el identificador de la comparación a tener en cuenta (argumento coef). El parametro sort.by nos permite ordenar las filas del marco de datos según el fold-change o el p-valor.

```
WT.with.no.Fe <- topTable(contrast.results, number=22810,coef=1,sort.by="logFC")
head(WT.with.no.Fe)
```

```
##          logFC    AveExpr      t    P.Value   adj.P.Val       B
## 253502_at 6.292455 6.298557 33.853364 8.018263e-11 1.099382e-06 12.79502
## 254550_at 6.069725 9.163291 9.415532 5.773803e-06 1.848275e-03  4.57266
## 257135_at 5.528440 7.151721 29.302962 2.919267e-10 1.506046e-06 12.19495
## 255524_at 4.788395 6.407028 28.950314 3.252976e-10 1.506046e-06 12.14013
## 245692_at 4.162611 6.917671 20.633406 6.630683e-09 1.102313e-05 10.32704
## 249535_at 4.151186 6.776116 20.586516 6.765623e-09 1.102313e-05 10.31311
```

Cuando se comparan los transcriptomas de dos genotipos diferentes o de un mismo genotipo bajo distintas condiciones existen diversos métodos para determinar genes expresados de forma diferencial o differentially expressed genes (DEGs) en inglés:

Método basado en el fold-change (factor de proporcionalidad):

Se fija un umbral para el fold-change típicamente 2, 4 u 8 que en log2 corresponde a 1, 2 ó 3. Los DEGs son aquellos que incrementan (o decrementan) su expresión por encima de dicho umbral (por debajo de menos

dicho umbral). Este método es biológicamente interpretable de forma directa y no requiere un alto número de réplicas biológicas. Se aplica especialmente a estudios con organismo modelos donde no son necesarias muchas réplicas.

Para la selección de DEGs extraemos el fold change e identificadores de cada sonda.

```
fold.change.WT.with.no.Fe <- WT.with.no.Fe[["logFC"]]
genes.ids.WT.with.no.Fe <- rownames(WT.with.no.Fe)
```

Fijamos un umbral de 2 que corresponde a genes que se expresan más del doble o menos de la mitad. Ya que nuestros datos están transformados por log2 en nuestro código usamos $\log_2(2) = 1$

```
activated.genes.WT.with.no.Fe.1 <- genes.ids.WT.with.no.Fe[fold.change.WT.with.no.Fe > 1]
repressed.genes.WT.with.no.Fe.1 <- genes.ids.WT.with.no.Fe[fold.change.WT.with.no.Fe < - 1]

length(activated.genes.WT.with.no.Fe.1)

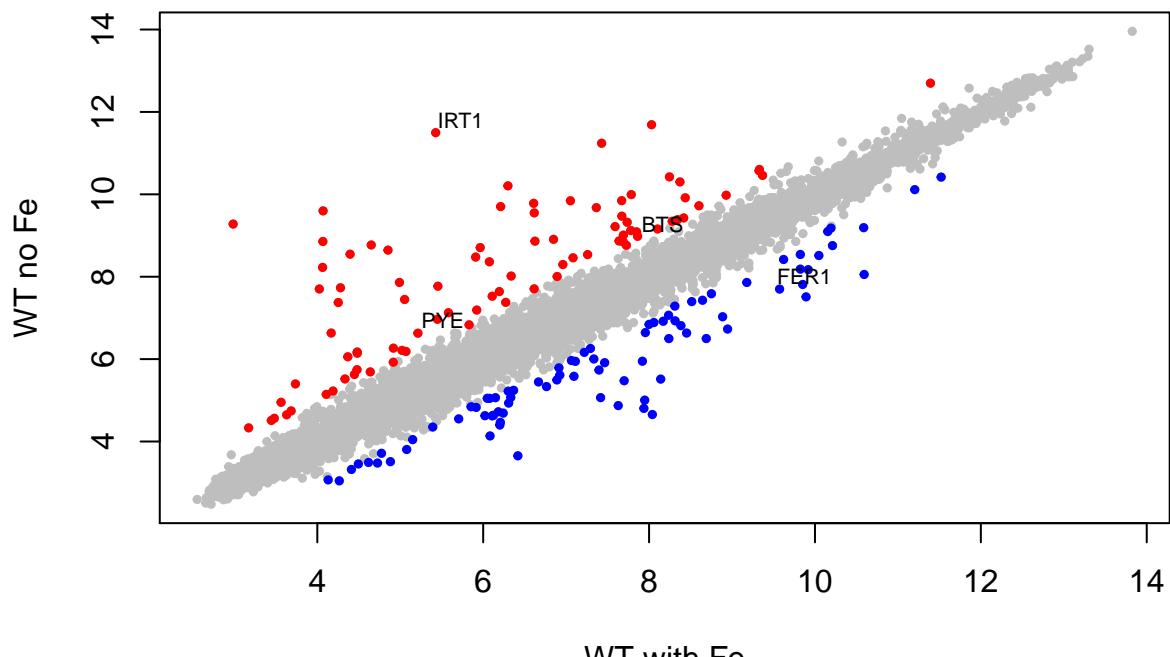
## [1] 87

length(repressed.genes.WT.with.no.Fe.1)

## [1] 83
```

Los gráficos que se usan principalmente para representar los DEGs según el criterio del fold-change son los gráficos de dispersión o scatterplots.

```
plot(wt.with.fe,wt.no.fe,pch=19,cex=0.5,col="grey",xlab="WT with Fe",ylab="WT no Fe")
points(wt.with.fe[activated.genes.WT.with.no.Fe.1],wt.no.fe[activated.genes.WT.with.no.Fe.1],pch=19,cex=0.5)
points(wt.with.fe[repressed.genes.WT.with.no.Fe.1],wt.no.fe[repressed.genes.WT.with.no.Fe.1],pch=19,cex=0.5)
text(wt.with.fe["254550_at"]+0.3,wt.no.fe["254550_at"]+0.3,"IRT1", col="black", cex=0.7)
text(wt.with.fe["252427_at"]+0.3,wt.no.fe["252427_at"]+0.3,"PYE", col="black", cex=0.7)
text(wt.with.fe["257062_at"]+0.3,wt.no.fe["257062_at"]+0.3,"BTS", col="black", cex=0.7)
text(wt.with.fe["251109_at"]+0.3,wt.no.fe["251109_at"]+0.3,"FER1", col="black", cex=0.7)
```



Anotación de los genes expresados de forma diferencial Podemos extraer información sobre los genes expresados de forma diferencial. La librería o paquete *annaffy* nos proporciona funciones para generar listas de DEGs en formato html o txt con la anotación disponible sobre los distintos genes.

```
library(annaffy)

## Loading required package: GO.db
## Loading required package: AnnotationDbi
## Loading required package: stats4
## Loading required package: GenomeInfoDb
## Loading required package: S4Vectors
## Loading required package: IRanges
##
## Attaching package: 'IRanges'
##
## The following object is masked from 'package:simpleaffy':
##
##     members
##
## Loading required package: DBI
##
## Loading required package: KEGG.db
##
## KEGG.db contains mappings based on older data because the original
##   resource was removed from the the public domain before the most
##   recent update was produced. This package should now be
##   considered deprecated and future versions of Bioconductor may
##   not have it available. Users who want more current data are
##   encouraged to look at the KEGGREST or reactome.db packages
```

La función *affTableAnn* nos permite generar una tabla con DEGs y su anotación. Las funciones *saveHTML* y *saveText* escriben la tabla anterior en formato HTML y txt.

```
activated.genes.WT.with.no.Fe.1.table <- aafTableAnn(activated.genes.WT.with.no.Fe.1, "ath1121501.db", )

## Loading required package: ath1121501.db
## Loading required package: org.At.tair.db

## Warning in chkPkgs(chip): The ath1121501.db package does not appear to
## contain annotation data.

saveHTML(activated.genes.WT.with.no.Fe.1.table, file="activated_genes_WT_with_no_Fe_1.html")
saveText(activated.genes.WT.with.no.Fe.1.table, file="activated_genes_WT_with_no_Fe_1.txt")

repressed.genes.WT.with.no.Fe.1.table <- aafTableAnn(repressed.genes.WT.with.no.Fe.1, "ath1121501.db", )
saveHTML(repressed.genes.WT.with.no.Fe.1.table, file="repressed_genes_WT_with_no_Fe_1.html")
saveText(repressed.genes.WT.with.no.Fe.1.table, file="repressed_genes_WT_with_no_Fe_1.txt")
```

Método que combina inferencia estadística con fold-change:

Para aplicar este método es necesario tener un alto número de réplicas biológicas. Para cada gen y para cada pareja de genotipos/condiciones a comparar se formula un contraste de hipótesis sobre igualdad de medias.

Normalmente este contraste de hipótesis utiliza un estadístico similar a la t-student. Se fija un nivel de significancia y se calcula el correspondiente p-valor (y p-valor corregido para el testeo múltiple o q-valor). Si dicho p-valor (o q-valor) es menor que el nivel de significancia y además el correspondiente fold-change cumple el umbral se asume que el correspondiente gen se expresa de forma diferencial en los genotipos/condiciones estudiadas.

Para la selección de DEGs según este criterio extraemos el fold change, el p-valor ajustado y los identificadores de cada sonda.

```
fold.change.WT.with.no.Fe <- WT.with.no.Fe[["logFC"]]
p.value.WT.with.no.Fe <- WT.with.no.Fe[["adj.P.Val"]]
genes.ids.WT.with.no.Fe <- rownames(WT.with.no.Fe)
```

Fijamos como umbral de significancia un p-valor menor a 0.05 y como umbral para el fold change 2 (que corresponde a 1 en log2).

```
activated.genes.WT.with.no.Fe.2 <- genes.ids.WT.with.no.Fe[fold.change.WT.with.no.Fe > 1 & p.value.WT.w
repressed.genes.WT.with.no.Fe.2 <- genes.ids.WT.with.no.Fe[fold.change.WT.with.no.Fe < -1 & p.value.WT.w
```

```
length(activated.genes.WT.with.no.Fe.2)
```

```
## [1] 80
```

```
length(repressed.genes.WT.with.no.Fe.2)
```

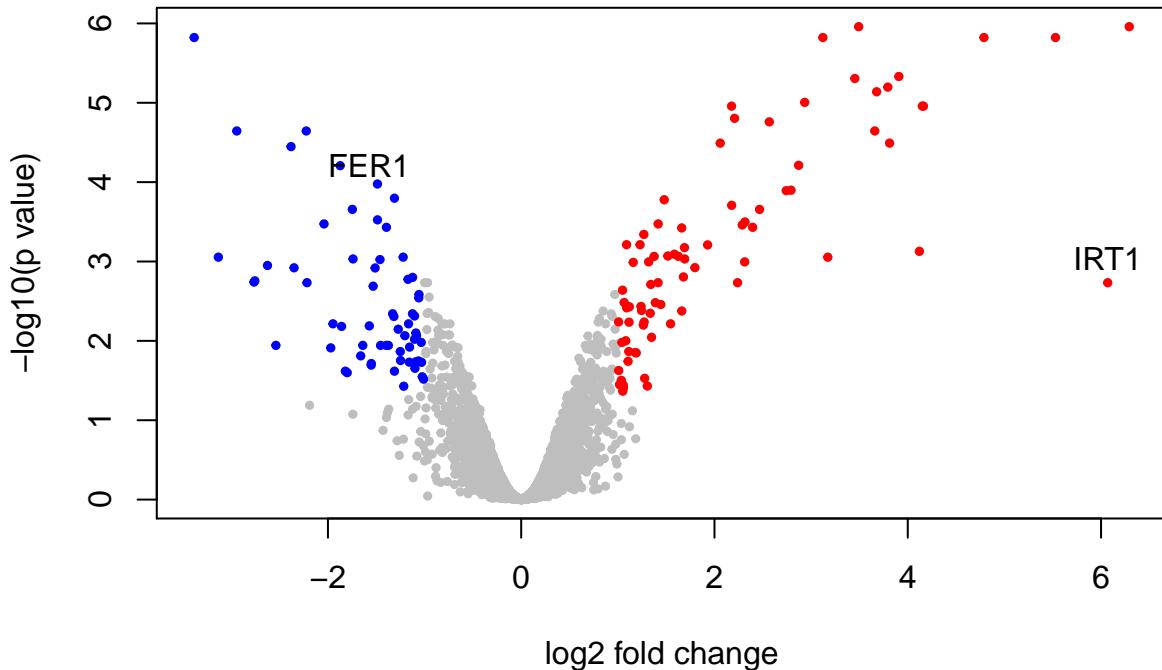
```
## [1] 63
```

Los volcano plots nos permiten representar en un único gráfico el fold change y la significancia estadística. Estos gráficos se utilizan para representar los DEGs cuando se usa el método que combina fold-change e inferencia estadística.

```
names(fold.change.WT.with.no.Fe) <- genes.ids.WT.with.no.Fe
log.p.value.WT.with.no.Fe <- -log10(p.value.WT.with.no.Fe)
names(log.p.value.WT.with.no.Fe) <- genes.ids.WT.with.no.Fe

plot(fold.change.WT.with.no.Fe,log.p.value.WT.with.no.Fe,pch=19,cex=0.5,col="grey",ylab="-log10(p value")
points(fold.change.WT.with.no.Fe[activated.genes.WT.with.no.Fe.2],log.p.value.WT.with.no.Fe[activated.g
points(fold.change.WT.with.no.Fe[repressed.genes.WT.with.no.Fe.2],log.p.value.WT.with.no.Fe[repressed.g

text(fold.change.WT.with.no.Fe["254550_at"],log.p.value.WT.with.no.Fe["254550_at"]+0.3,"IRT1", col="bla
text(fold.change.WT.with.no.Fe["251109_at"]+0.3,log.p.value.WT.with.no.Fe["251109_at"],"FER1", col="bla
```



Mapas de Calor

Los mapas de calor nos permiten visualizar los niveles de expresión de los DEGs en todas las condiciones analizadas utilizando un código de colores. Esta visualización global nos permite identificar genes con comportamientos similares en las condiciones analizadas.

Para realizar un mapa de calor el primer paso consiste en recopilar todos los genes expresados de forma diferencial en un vector y eliminar repeticiones.

```
complete.DEGs <- c(diff.genes.1[["activated.genes"]],diff.genes.1[["repressed.genes"]],diff.genes.2[["activated.genes"]],diff.genes.2[["repressed.genes"]])
complete.DEGs <- unique(complete.DEGs)
length(complete.DEGs)
```

A continuación, extraemos los niveles de expresión de todos los DEGs en las condiciones/genotipos de interés y normalizamos los datos teniendo en cuenta que la función scale normaliza por columnas y nosotros queremos hacerlo por filas. Calculamos con la función t la traspuesta de nuestra matriz.

```
DEG.expression <- mean.expression[complete.DEGs,c("WT_with_Fe","WT_no_Fe","pye_with_Fe","pye_no_Fe")]
normalized.DEG.expression <- t(scale(t(DEG.expression)))
```

Con la función heatmap.2 del paquete gplots construimos el correspondiente heatmap.

```
library(gplots)
help(heatmap.2)
heatmap.2(normalized.DEG.expression,Colv=FALSE,dendrogram="row",labRow=c()),density.info="none",trace=
```