Trabajo Teórico Práctico 9 Procesamiento completo para la selección de genes candidatos a estar expresados diferencialmente utilizando **Limma**

1 Dos grupos, comparación directa.

1.1 Lectura de datos a partir de los archivos resultantes del procesamiento de imágenes del programa de análisis de imágenes GenePix.

Como ya hemos visto los datos de intensidades generados por el programa GenePix están guardados en archivos con extensión gpr

Para evitar escribir todo el "path" es más conveniente cambiar el working directory a la carpeta donde se encuentran los archivos .gpr resultantes del procesamiento de imágenes: desde el menú principal: **`file -> change dir.. -> (Browse)`**
`archivo -> cambiar dir.. -> (Paginar)`

```
C:\Archivos de programa\R\R-2.5.0\library\beta7\beta7
```

Luego cargar la librería limma

```
> library(limma)
```

Utilizamos la función **`readTargets`** para leer la lista del RNA target hibridado a cada canal de cada arreglo, y los nombres de los archivos que contienen la información de las intensidades.

```
#  targets <- readTargets("targets.txt") #en general
```

```
> targets <- readTargets("TargetBeta7.txt")
```

```
> targets
             FileNames SubjectID  Cy3  Cy5
6Hs.195.1 6Hs.195.1.gpr         1 b7 - b7 +
6Hs.168     6Hs.168.gpr         3 b7 + b7 -
6Hs.166     6Hs.166.gpr         4 b7 + b7 -
6Hs.187.1 6Hs.187.1.gpr         6 b7 - b7 +
6Hs.194     6Hs.194.gpr         8 b7 - b7 +
6Hs.243.1 6Hs.243.1.gpr        11 b7 + b7 -
```

La siguiente función se utilizará para fijar un filtro de manera que cualquier spot marcado con (flag de) -99 o menos tenga peso cero.

```
  > f <- function(x) as.numeric(x$Flags > -99)
```

Para ver otras funciones disponibles para dar pesos de acuerdo con la calidad del spot:

```
  >?QualityWeights
```

Para leer los datos de intensidades utilizamos la función **read.maimages**. Qué tipo de objeto genera la función? Qué componentes tiene? Utilice el help

```
  >  ?read.maimages
```

Los datos leídos con **read.maimages** difieren un poco de los leídos con **read.GenePix** pues **read.maimages** toma intensidades medias para el foreground de cada spot mientras que **read.GenePix** toma las medianas. Esta diferencia no debería ser importante en este caso.

Los nombres de los archivos guardados en **targets** permiten identificar la información de intensidades contenida en los archivos de salida del procesamiento de imágenes. En este caso son archivos con extensión **gpr**.

```
> RG <- read.maimages(targets$FileName, source="genepix", wt.fun=f)
```

Mediante la opción **source="genepix"** se obtienen los valores del foreground para cada spot del arreglo que provienen de las columnas

- `F635: Mean` (Cy5) Es la media de las intensidades del canal rojo
- `F532: Mean` (Cy3) Es la media de las intensidades del canal verde

de cada archivo .gpr. y para las intensidades del background se utilizan las medianas:

- `B635 Median` (Cy5)
- `B532 Median` (Cy3)

Qué produce la opción **'source="genepix.median"** Mire el help**.**

```
> summary(RG)
        Length Class       Mode
R       139104 -none-      numeric
G       139104 -none-      numeric
Rb      139104 -none-      numeric
Gb      139104 -none-      numeric
weights 139104 -none-      numeric
targets      1 data.frame  list
genes        5 data.frame  list
source       1 -none-      character
printer      6 PrintLayout list
```

Veamos los pesos

```
> summary(RG$weights)
```

¿Los primeros 10 genes tienen peso = ...?

```
> RG$weights[1:10,]
```

Veamos la estructura del microarreglo

```
> RG$printer
```

1.2 Gráficos

Gráficos espaciales

Los gráficos de imágenes para cada uno de los arreglos se obtienen utilizando la función **`imageplot`**. Mediante las siguientes instrucciones obtenemos los gráficos del background y las intensidades de los canales rojo y verde del primer arreglo:

```
x11()
par(mfrow=c(2,1))
imageplot(log2(RG$Rb[,1]), RG$printer, low="white", high="red")
imageplot(log2(RG$Gb[,1]), RG$printer, low="white", high="green")

x11()
par(mfrow=c(2,1))
 imageplot(log2(RG$R [,1]), RG$printer, low="white", high="red")
 imageplot(log2(RG$G [,1]), RG$printer, low="white", high="green")
```

MA-plots.

Cálculo de los valores M y A

Los valores de M y A pueden ser calculados utilizando la función **`normalizeWithinArrays`**. Con la opción **`method="none"`** calcula los valores de M y A crudos (no normalizados).

```
 MA <- normalizeWithinArrays(RG, method="none")
```

o equivalentemente

```
 MA <- MA.RG(RG)
```

La función **`MA.RG`** convierte una **`'RGList'`** sin logaritmizar en un objeto '**MAList**'.

`'MA.RG(objeto)'` es equivalente a
`'normalizeWithinArrays(objeto,method="none")'`

`'RG.MA(objeto)'` convierte en forma inversa un objeto **`'MAList'`** a un objeto **`'RGList'`** con las intensidades sin logaritmizar.

Con las instrucciones anteriores no hemos normalizado pero por defecto se ha realizado la sustracción del background al foreground..

Pueden elegirse otras opciones de corrección por background utilizando la función `backgroundCorrect` sobre una `RGList`. Por ejemplo si no queremos realizar la corrección por background:

```
> RGnobg <- backgroundCorrect(RG, method="none")
> MAnobg <- MA.RG(RGnobg)
```

Gráficos MA

Para graficar los datos crudos M y A para el primer arreglo se utiliza:

```
> plotMA(MA, array=1) #con sustracción del background
> plotMA(MAnobg, array=1) #sin corrección por background
```

Para los demás arreglos se modifica el argumento `array` . Por ejemplo para el segundo arreglo `array=2`.

Gráficos MA individuales, para cada print-tip group

Estos gráficos incluyen la curva loess que se utilizará para la normalización

```
> plotPrintTipLoess(MA)
```

1.3 Corrección por background - beta7

La siguiente corrección por background "normexp" o "rma"es recomendada por la guía de usuarios de limma para los datos de GenePix

```
RGrma <- backgroundCorrect(RG, method="rma")
```

1.4 Normalización por print-tip - beta7

La normalización

```
> MA <- normalizeWithinArrays(RG)
```

es equivalente a

```
> RGb <- backgroundCorrect(RG, method="subtract")
> MA <- normalizeWithinArrays(RGb)
```

Como hemos visto antes de normalizar, para los datos de GenePix es recomendable realizar la corrección “normexp” o “rma”

```
> RGrma <- backgroundCorrect(RG, method="rma")
> MA <- normalizeWithinArrays(RGrma)
```

1.5 Ajuste de un modelo lineal - beta7

Especificación del modelo

Una vez que los datos se han normalizados, se ajusta un modelo lineal a los valores M de cada gen. Recordemos qué modelo lineal ajustamos. Para cada gen, tendremos tantas ecuaciones como microarreglos componen el experimento. Sea $y_i = M_i = \log(R / G)$ para el arreglo i. Para beta7 serán 6 ecuaciones. Se trata de un experimento con intercambio de tintes (dye-swap) de acuerdo con la tabla siguiente

Cy3	Cy5
b7 -	b7 +
b7 +	b7 -
b7 +	b7 -
b7 -	b7 +
b7 -	b7 +
b7 +	b7 -

por lo tanto las ecuaciones son

$$
\begin{aligned}
Y_1 &= \beta + \varepsilon \\
Y_2 &= -\beta + \varepsilon \\
Y_3 &= -\beta + \varepsilon \\
Y_4 &= \beta + \varepsilon \\
Y_5 &= \beta + \varepsilon \\
Y_6 &= -\beta + \varepsilon
\end{aligned}
$$

El modelo anterior se puede escribir vectorialmente como

$$Y = X\,\beta + \varepsilon$$

ó

$$E(Y) = X\,\beta$$

donde X es la matriz de diseño (en este caso de 6 filas y 1 columna) y está dada por `(1,-1,-1,1,1,-1)` transpuesto. Los -1 corresponden a los arreglos con dye- swap.

En general cada fila de la matriz de diseño corresponde a un arreglo del experimento, en este caso son 6 filas = 6 arreglos. En un diseño directo de microarreglos de dos colores,

como es el que estamos desarrollando se necesitan tantas columnas como fuentes de variabilidad de RNA menos una. En este caso son 2 -1 = 1 columna.

¿Qué representa β en este modelo?

Ajuste del modelo lineal

El modelo se ajusta utilizando la función **`lmFit,`** por defecto realiza un ajuste por cuadrados mínimos ordinario. Tiene opciones para incluir una estructura de covarianzas de los errores y un ajuste robusto. Mire el help.

La matriz **`design`** indica cuáles arreglos son dye-swaps.

```
> fit1 <- lmFit(MA, design=c(1,-1,-1,1,1,-1))

>  names(fit1)
 [1] "coefficients"     "stdev.unscaled"   "sigma"        "df.residual"
 [5] "cov.coefficients" "pivot"            "method"            "design"
 [9] "genes"            "Amean"

> summary(fit1)
                 Length Class      Mode
coefficients     23184  -none-     numeric
stdev.unscaled   23184  -none-     numeric
sigma            23184  -none-     numeric
df.residual      23184  -none-     numeric
cov.coefficients     1  -none-     numeric
pivot                1  -none-     numeric
method               1  -none-     character
design               6  -none-     numeric
genes                5  data.frame list
Amean            23184  -none-     numeric
```

Los coeficientes son los M estimados para cada gen.

La función **`eBayes`** calcula el estadístico t moderado (Lönnstedt and Speed 2001)

$$t^* = \frac{\sqrt{n}M}{\sqrt{a + s^2}}$$

con M = $\frac{\sum_{i=1}^{n} Y_i}{n}$, $n = 6$ en este caso y un p-valor, para cada gen.

```
> fit <- eBayes(fit1)
> summary(fit)
                 Length Class      Mode
coefficients     23184  -none-     numeric
stdev.unscaled   23184  -none-     numeric
sigma            23184  -none-     numeric
df.residual      23184  -none-     numeric
cov.coefficients     1  -none-     numeric
```

```
pivot                1  -none-     numeric
method               1  -none-     character
design               6  -none-     numeric
genes                5  data.frame list
Amean            23184  -none-     numeric
df.prior             1  -none-     numeric
s2.prior             1  -none-     numeric
var.prior            1  -none-     numeric
proportion           1  -none-     numeric
s2.post          23184  -none-     numeric
t                23184  -none-     numeric  Estadístico t*
p.value          23184  -none-     numeric
lods             23184  -none-     numeric
F                23184  -none-     numeric
F.p.value        23184  -none-     numeric
```

Para obtener el estadístico t usual de la salida de la función eBayes:

```
> Estadístico.t.usual <- fit$coef / fit$stdev.unscaled / fit$sigma
```

Genes ordenados según su evidencia de estar expresados diferencialmente

Para ver los estadísticos de los 10 primeros genes, utilizamos la función **`topTable`** . Los p.valores están ajustado de acuerdo con el método de Benjamini y Hochberg (1995) , **`adjust.method="BH",`** para controlar el FDR

```
> topTable(fit,coef=1,number=10,genelist=fit$genes,adjust.method="BH",
sort.by="B",resort.by=NULL)
```

1.6 Lectura de datos - swirl- Programa de análisis de imágenes Spot.

Igual que antes, primero hay que cambiar el working directory a la carpeta donde se encuentran los archivos resultantes del procesamiento de imágenes: desde el menú principal: **`file -> change dir.. -> (browse)`**

En este ejemplo tenemos que ir al subdirectorio swirldata:

C:\Archivos de programa\R\ ..\Library\marray\swirldata

```
> library(limma)
> targets <- readTargets("SwirlSample.txt")
```

```
> targets
         Names slide.number experiment.Cy3 experiment.Cy5   date
1 swirl.1.spot         81           swirl      wild type 2001/9/20
2 swirl.2.spot         82      wild type          swirl 2001/9/20
3 swirl.3.spot         93           swirl      wild type 2001/11/8
```

```
4 swirl.4.spot         94      wild type          swirl 2001/11/8
```

```
> RG <- read.maimages(targets$Names, source="spot")
```

Para ver el contenido

```
> RG
```

Por defecto se utilizan las columnas **Rmean** y **Gmean** de cada archivo como intensidades del foreground y `morphR` y `morphG` se utilizan como intensidades del background.

Aquí también **RG** es un objeto de tipo **RGList** que contiene la intensidades del foreground y el background para cada uno de los canales rojo y verde para cada gen (spot) y para cada arreglo. .

El archivo .gal contiene información sobre los genes y la estructura del arreglo.

```
RG$genes <- readGAL("fish.gal")
```

```
> summary( RG$genes )
     Block            Row           Column           ID                Name
 Min.   : 1.00   Min.   : 1.0   Min.   : 1.00   Length:8448        Length:8448
 1st Qu.: 4.75   1st Qu.: 6.0   1st Qu.: 6.75   Class :character   Class
:character
 Median : 8.50   Median :11.5   Median :12.50   Mode  :character   Mode
:character
 Mean   : 8.50   Mean   :11.5   Mean   :12.50
 3rd Qu.:12.25   3rd Qu.:17.0   3rd Qu.:18.25
 Max.   :16.00   Max.   :22.0   Max.   :24.00
```

La matriz **RG$genes** contiene en sus primeras 3 columnas la información del bloque y fila y columna dentro de cada bloque del gen. De aquí se infiere la estructura del arreglo

```
RG$printer <- getLayout(RG$genes)
```

Ahora toda la información necesaria está guardada en nuestro objeto de tipo **RGList**. Tenemos un resumen con

```
> RG
```

Ejercicio para los datos Swirl

a) Normalizar
b) Defina la matriz de diseño
c) Ajuste el modelo lineal
d) Obtenga los estadísticos t moderados
e) Halle los 5 genes que ofrecen más evidencia de estar diferencialmente expresados de acuerdo con los estadísticos t moderados

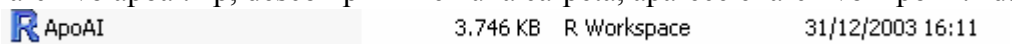
2 Selección de genes diferencialmente expresados. Dos grupos, comparación indirecta.

Dos muestras independientes

Consideramos un estudio en el cual dos tipos de muestras de RNA son comparadas a través de una muestra común de referencia . En este caso el análisis de los log-ratios corresponde a la comparación de medias para dos muestras independientes.

Para desarrollar este ejemplo hay que bajar los datos de
http://bioinf.wehi.edu.au/marray/jsm2005/apoai.zip

Es el archivo apoai.zip, descomprimir en una carpeta, aparece el archivo ApoAI.Rdata



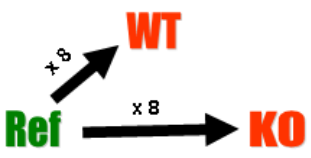
2.1 Descripción de los datos

Objetivo. Los datos son de un estudio sobre el metabolismo de lípidos (Callow et al 2000). Es conocido que el gen de la apolipoproteina AI (ApoAI) juega un papel pivotal en el metabolismo de la lipoproteina de alta densidad (HDL). Los ratones a los que se les ha eliminado (knocked out) el gen de ApoAI tienen muy bajos niveles de colesterol HDL. El objetivo de este experimento es determinar cómo la deficiencia de ApoAI afecta la acción de otros genes del hígado con la idea de que esto puede ayudar a determinar los caminos moleculares por los cuales opera ApoAI.

Hibridaciones. El experimento compara 8 ratones ApoAI knockout con 8 normales (wild type) C57BL/6 ("black six"), son los ratones de control. Para cada uno de estos 16 ratones, el mRNA target fue obtenido del tejido de hígado y etiquetado utilizando un tinte Cy5. El RNA de cada ratón fue hibridado a un microarreglo separado. El RNA de referencia común a todos los arreglos fue etiquetado con el tinte Cy3. El RNA de referencia fue obtenido realizando una mezcla (pool) del RNA extraído de los 8 ratones de control.

Cantidad de Arreglos	Rojo (Cy5)	Verde (Cy3)
8	Ratones Wild Type "black six" (**WT**)	Referencia (**Ref**)
8	ApoAI Knockout (**KO**)	Referencia (**Ref**)

El diseño del experimento puede ser descripto mediante el siguiente diagrama:



Este experimento es un poco viejo. Actualmente sería inusual que se utilicen esta cantidad de arreglos replicados sin incluir ningún intercambio de tintes (dye-swaps). En arreglos de dos colores el sesgo del tinte sigue siendo un problema. Por esta razón la normalización es siempre necesaria.

Este es un ejemplo de una única comparación utilizando una referencia común. El hecho que la comparación se realiza utilizando una referencia común en vez de realizarla directamente, como en el experimento de swirl, tenemos dos muestras para cada gen en vez de una muestra.

2.2 Lectura de los datos

Dos formas de lectura de los datos

- clickear en el ícono  que se ha generado en esa carpeta.

  Esto es iniciar R desde allí. Vemos que objetos tenemos

  ```
  > objects()
  [1] "RG"
  ```

  Cargamos la librería limma

  ```
  > library (limma)
  ```

Otra forma

- Desde el R nuevamente, cambiar el working directory a la carpeta donde se encuentran los datos:

  ```
  file -> change dir.. -> (browse)
  ```

  y ejecutar la instrucción:

  ```
  > load("ApoAI.RData")
  ```

  La función **load** se utiliza cuando un objeto de R se ha guardado con la función **save,** por ejemplo**:**

  ```
  > save(cinturón, file="cinturón.Rdata")
  > load("cinturón.Rdata")
  ```

Ya tenemos los datos del experimento en un objeto **RGlist** llamado **RG**

```
> RG
> summary(RG)
```

2.3 Normalización

Normalización por defecto, print tip loess

```
> MA <- normalizeWithinArrays(RG)
```

2.4 Definición de la matriz de diseño

Recordemos qué modelo lineal estamos ajustando para cada gen:

$$E(Y) = X \beta$$

Donde

- Y es el vector de los log ratios normalizados, de un gen, de cada uno de los 16 arreglos
- E(Y) es el valor esperado de Y
- X es la matriz de diseño
- β es el vector de los log ratios para estimar los correspondientes M que aparecerá en la columna de la lista final de los genes expresados diferencialmente dada por **`topTable`**. Los log ratios estimados también son llamados "coefficients", "parameters" and "log fold changes".

Este experimento tiene tres tipos de RNA:

- Referencia (**Ref**)
- Wild Type (**WT**)
- Knockout (**KO**)

Es suficiente estimar para cada gen dos log ratios en el modelo lineal.

Estimaremos dos parámetros por lo tanto nuestra matriz de diseño tendrá dos columnas. Elegiremos los dos parámetros que constituyen el vector $\beta = (\beta_1, \beta_2)$ de manera que:

- β_1 corresponde a los log ratios que comparan los niveles de expresión de **WT vs. Ref**
- β_2 corresponde a los log ratios que comparan los niveles de expresión de **KO vs. WT** (es el parámetro de interés en este caso)

Aunque otras parametrizaciones son posibles estamos utilizando una que nos permite estimar el contraste de interés (**KO vs. WT**) directamente del ajuste del modelo lineal en vez de estimarlo posteriormente como un contraste (esto es una combinación lineal de los parámetros estimados del modelo lineal).

Utilizaremos la siguiente matriz de diseño:

$$X = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Cada fila de la matriz corresponde a un arreglo del experimento, en este caso son 16 filas = 16 arreglos y cada columna corresponde a cada coeficiente que es utilizado para describir las fuentes de RNA. En un experimento de dos colores con una referencia común se necesitan tantos coeficientes como fuentes de RNA se tengan.

Veremos por qué la primera columna corresponde al parámetro β_1 de "**WT vs. Ref**" y la segunda para el parámetro β_2 de "**KO vs. WT**". Las primeras 8 filas de la matriz de diseño se corresponderán con las ecuaciones de los primeros 8 arreglos en los que se hibrida **WT** RNA con **Ref** RNA en esa ecuación aparecerá únicamente β_1 es por esto que aparece un '1' en la columna de **WT vs. Ref**. Para los últimos 8 arreglos en los que se hibrida RNA **KO** con RNA **Ref** le corresponde la suma de los dos parámetros.

Justificación de la interpretación de los coeficientes en este ejemplo

De acuerdo con la matriz de diseño estamos modelando mediante las siguientes ecuaciones:

$$E(\log(\frac{\text{Wt}}{\text{Ref}})) = \beta_1 \qquad \text{para los arreglos } 1 \text{ a } 8$$

$$E(\log(\frac{\text{KO}}{\text{Ref}})) = \beta_1 + \beta_2 \qquad \text{para los arreglos } 9 \text{ a } 16$$

Las ecuaciones anteriores pueden expresarse de la siguiente manera:

$$E(Y_i) = \beta_1 x_{1i} + \beta_2 x_{2i} \text{ donde } \begin{matrix} x_{1i} = 1 & x_{2i} = 0 & \text{si } i = 1,...,8 \\ x_{1i} = 1 & x_{2i} = 1 & \text{si } i = 9,...,16 \end{matrix}$$

¿Qué mide β_2?

Restando las dos ecuaciones

$$E(\log(\frac{\text{KO}}{\text{Ref}})) = \beta_1 + \beta_2$$

–

$$E(\log(\frac{\text{Wt}}{\text{Ref}})) = \beta_1$$

tenemos que

$$E(\log(\frac{\text{KO}}{\text{Wt}})) = \beta_2$$

Por lo tanto β_2 mide el log ratio de la intensidad del gen entre el RNA del KO y el RNA del Wt que es directamente el contraste de interés.

Definición de la matriz de diseño en R

Hemos visto cómo definir matrices en R, en este caso la matriz de diseño puede definirse de la siguiente manera:

```
> design <- matrix(c(rep(1,16),rep(0,8),rep(1,8)),ncol=2)
> colnames(design) <- c("WT-Ref","KO-WT")
> design
```

Una vez que hemos definido la matriz de diseño lo que sigue es igual que en los casos anteriores de una muestra. Si se tienen más muestras el procedimiento es idéntico salvo que cambiará la matriz de diseño.

2.5 Ajuste del modelo lineal

Cuadrados mínimos ordinario

```
> fit <- lmFit(MA,design=design)
> names(fit)
```

Ajuste robusto

```
 > fitrob <- lmFit(MA,design=design, method="robust")
> names(fitrob)
```

2.6 Cálculo de los estadísticos moderados - Empirical Bayes

Cuadrados mínimos ordinario

```
> fit <- eBayes(fit)
> names(fit)
```

Ajuste robusto

```
> fitrob <- eBayes(fitrob)
> names(fitrob)
```

2.7 Despliegue de la lista de genes

Utilizamos la función **topTable** para obtener la lista de los genes con mayor evidencia de estar expresados diferencialmente entre las muestras de RNA Knockout y Wild-Type. En teoría el gen anulado (knockout gene), ApoAI no debería estar expresado dando un log fold change de menos infinito. A pesar que el valor M del gen ApoAI en topTable no tiene demasiado sentido biológico su alto ranking muestra que está sistemáticamente regulado hacia abajo (down-regulated) en estas replicaciones biológicas.

Ejercicio: Explore los argumentos de topTable en detalle con **?topTable**. Por defecto los genes se ordenan de acuerdo con el estadístico B (log odds de expresión diferencial, Lonnstedt and Speed 2003), pero también se pueden utilizar el estadístico t moderado ó el p-valor p-value.

Cuadrados mínimos ordinario

```
> topTable(fit,coef="KO-WT")
 > numGenes <- nrow(RG$genes)
 > completeTableKOvsWT <- topTable(fit,coef="KO-WT",number=numGenes)
```

Ajuste robusto

```
> topTable(fitrob,coef=2) # nos interesa β2

 > numGenes <- nrow(RG$genes)
 > completeTableKOvsWTrob <-
           topTable(fitrob,coef=2,number=numGenes)
```

Exportamos los resultados, cada uno a un archivo.

Cuadrados mínimos ordinario

```
 > write.table(completeTableKOvsWT,file="KOvsWTgenes.xls",sep="\t",
quote=FALSE,col.names=NA)
```

La extensión ".xls" es utilizada de manera que el archivo de texto delimitado por tabulaciones (tab-delimited text) pueda ser abierto por el Excel dobleclickeando sobre su ícono.

Ajuste robusto

```
> write.table(completeTableKOvsWTrob,file="KOvsWTgenesRob.xls",
sep="\t", quote=FALSE,col.names=NA)
```

2.8 M A plot utilizando los coeficientes del modelo lineal

Utilizamos un M A plot para visualizar cuales genes han sido seleccionados como expresados diferencialmente en base al estadístico B que es el ordenamiento por defecto de la función **topTable**. Los genes así seleccionados pueden no ser los más extremos respecto de sus fold changes promedio ya que los estadísticos dividen por la variabilidad entre arreglos replicados (s^2, ver teórica).

Analice y ejecute cada una de las instrucciones siguientes

```
 M <- fit$coefficients[,"KO-WT"]
 A <- fit$Amean
 ord <- order(fit$lods[,"KO-WT"],decreasing=TRUE)
 top10 <- ord[1:10]
 plot(A,M,pch=16,cex=0.1)
text(A[top10],M[top10],labels=substring(fit$genes[top10,"NAME"],1,5),
     cex=0.8,col="blue")
```

Referencias

Callow, M. J., Dudoit, S., Gong, E. L., Speed, T. P., and Rubin, E. M. (2000).

Microarray expression profiling identifies genes with altered expression in HDL deficient mice. *Genome Research* **10**, 2022-2029.
(http://www.genome.org/cgi/content/full/10/12/2022)

Guía Limma: http://bioinf.wehi.edu.au/limma/usersguide.pdf

Lab 4 - Differential Expression and Linear Modeling using limma
Ben Bolstad. August 6-7, 2005

Lonnstedt I, Speed T. Replicated microarray data. STAT SINICA. Volume 12. Pages 31-46. Jan 2002