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.

```
> 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

# leer el archivo "TargetBeta7.txt" de la librería beta 7 usando la
función file.choose()

> targets <- readTargets(file.choose())

> 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 -
```

GenePix produce una columna llamada “Flags” que vale cero para cualquier spot “normal” y valores negativos crecientes en valor absoluto para diferente clases de problemas. 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**.

```
> dirección.beta7 <- "C:\\Archivos de programa\\R\\R-
2.10.1\\library\\beta7\\beta7"

> RG <- read.maimages(targets$FileName, source="genepix", path =
dirección.beta7, wt.fun=f)
```

Otra forma de leer las imágenes, dando peso cero a cualquier spot con el valor de Flag menor que -50

```
> RG <- read.maimages(targets$FileName,source="genepix", path =
dirección.beta7, wt.fun=wtflags(weight=0,cutoff=-50))
```

Los pesos quedan guardados en la componente "weights" del objeto RGList:

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="normexp")
MArma <- MA.RG(RGrma)

plotMA(MArma, array=1) #con background por "normexp"
```

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=" normexp")
> MA <- normalizeWithinArrays(RGrma)# normalización por print
tip y background por "normexp"
> plotMA(MA, array=1) #con background por "normexp"
> plotPrintTipLoess(MA)
```

Falta ver si es necesaria una normalización entre arreglos

```
# veamos si es necesario una normalización entre arreglos
> boxplot(MA$M~col(MA$M),names=colnames(MA$M))
```

Los arreglos muestran diferentes dispersiones de los valores M luego es aconsejable una normalización por escala.

```
> MA <- normalizeBetweenArrays(MA,method="scale")
> boxplot(MA$M~col(MA$M),names=colnames(MA$M))
```

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 $Y_i = M_i = \log (Ri / Gi)$ 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

$$Y_1 = \beta + \varepsilon$$
$$Y_2 = -\beta + \varepsilon$$
$$Y_3 = -\beta + \varepsilon$$
$$Y_4 = \beta + \varepsilon$$
$$Y_5 = \beta + \varepsilon$$
$$Y_6 = -\beta + \varepsilon$$

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. Se estima la media de M para cada gen.

```
> 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
```

En fit1 **"coefficients"**son simplemente el promedio de los M para cada gen y sigma es el desvío estándar correspondiente.

```
> fit1

An object of class "MArrayLM"

$coefficients

[1] -0.32943195 -0.35760022 -0.03858380 -0.01335241 -0.01949076

23179 more elements ...

......
```

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

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

con $\overline{M} = \frac{\sum_{i=1}^{n} Y_i}{n}$, $n = 6$ en este caso y un p-valor, para cada gen (no está indicado el subíndice para cada gen).

El estadístico B es equivalente para el propósito de ordenar los genes al estadístico t* moderado, valores de B mayores que cero corresponden a una chance mayor que 50-50 de que el gen en cuestión esté expresado diferencialmente (Smyth 2003)

```
> 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
```

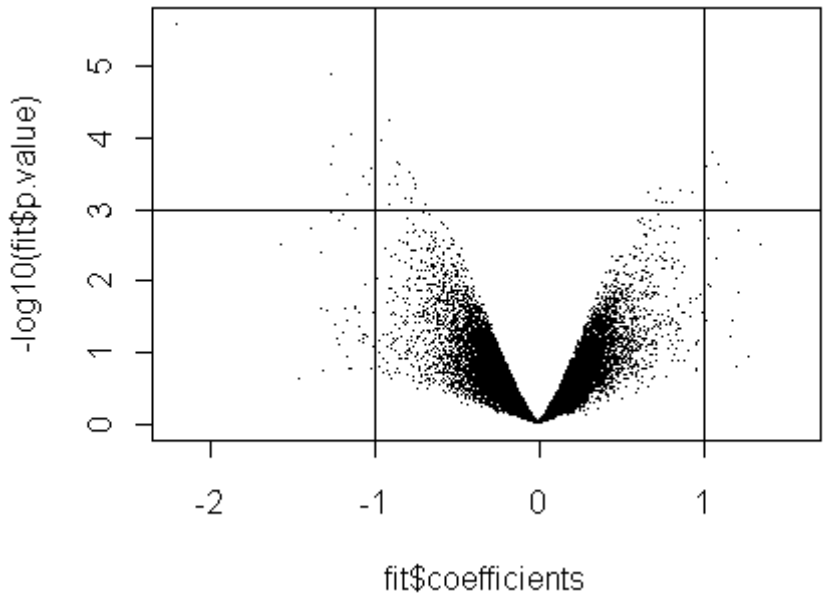
```
s2.prior             1  -none-     numeric
var.prior            1  -none-     numeric
proportion           1  -none-     numeric
s2.post          23184  -none-     numeric
t                23184  -none-     numeric  Estadístico t*
p.value          23184  -none-     numeric
lods             23184  -none-     numeric
F                23184  -none-     numeric
F.p.value        23184  -none-     numeric
```

Según la documentación de "Limma" para obtener el estadístico t usual de la salida de la función eBayes:

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

Volcano plot

```
plot( fit$coefficients, -log10(fit$p.value),pch='.')
abline(h=3,v=c(1,-1))
```



¿Qué significan las rectas verticales y la recta horizontal del "volcano plot"?

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)
```

The `sort.by` argument specifies the criterion used to select the top genes. The choices are: `"logFC"` to sort by the (absolute) coefficient representing the log-fold-change; `"A"` to sort by average expression level (over all arrays) in descending order; `"T"` or `"t"` for absolute t-statistic; `"P"` or `"p"` for p-values; or `"B"` for the `lods` or B-statistic.

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

Referencias

Smyth, G. K., Yang, Y.-H., Speed, T. P. (2003). Statistical issues in cDNA microarray data analysis. *Methods in Molecular Biology* **224**, 111-136. [PubMed ID 12710670]