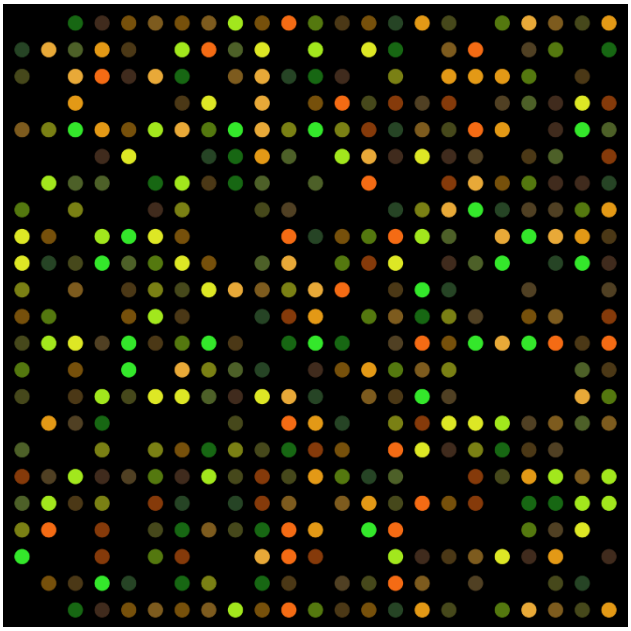
Faculdade de Ciências da Universidade de Lisboa
Departamento de Estatística e Investigação Operacional, CEAUL

Microarray data: Análise de dados de tecidos da veia safena em pacientes cardíacos



Disciplina de Métodos Estatísticos em Genética
Junho de 2020

Martim Almeida fc55066
João Damião fc55619

Índice

Introdução

Os *microarrays* são uma técnica utilizada na genómica que permite analisar a expressão génica em diferentes condições experimentais. O seu princípio de funcionamento reside em colocar fragmentos de DNA, ex: regiões codificantes de genes, num *chip* que contém compartimentos com sondas de DNA previamente preparadas. Em cada compartimento ocorre transcrição do DNA em mRNA num processo denominado como transcrição, que antecede a tradução para proteínas, sendo por isso esta molécula (mRNA) muito utilizado para medir a expressão génica. O mRNA é no entanto instável e por esse motivo é convertido para cDNA (DNA complementar) através de reações enzimáticas mediada pela transcriptase reversa [1]. O cDNA é marcado com dois fluoróforos, Cy3 e Cy5, e após clivagem do cDNA com endonucleases, vai estabelecer ligação com as sondas de DNA nos "poços", permitindo fixar os fragmentos. A intensidade de emissão dos fluoróforos é medida através de computadores, produzindo dados respetivos à intensidade de emissão [2]. Podem ser também medidos e considerados valores de *background* que correspondem a um ruído de emissão dos fluoróforos, o que pode interferir na análise da intensidade da emissão, sendo por isso necessário por vezes ajustar os valores de intensidade com os valores de *background*. A Figura 1 ilustra o processo experimental inerente à técnica de *microarray* de dois canais.

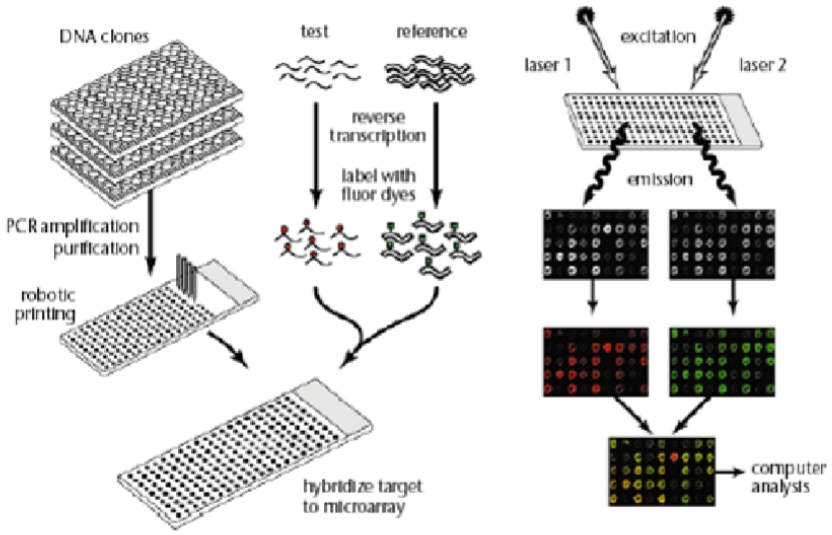


Figura 1 – Procedimento experimental na técnica de *microarray* de dois canais

Os dados resultantes do contexto experimental a que nos propomos analisar provém de amostras de tecido de veia safena de três doentes com uma patologia cardíaca associada. Cada uma amostra de tecido foi sujeita a duas condições experimentais explicitadas como regime arterial e venoso. Em cada uma das condições experimentais foi medido a expressão de 2994 genes, num *microarray* de dois canais, sendo o regime venoso associado com Cy3 e o regime arterial associado com o vermelho.

No contexto deste problema, o objetivo deste trabalho prende-se com determinar os genes mais e menos diferencialmente expressos, com a utilização da ferramenta R e bibliotecas pertinentes para o efeito.

Desenvolvimento

1. Análise individual ao primeiro paciente

A informação do primeiro doente foi guardada num ficheiro *chip1.txt* de onde se obteve um *array* com 2994 IDs. Um excerto da informação no ficheiro relativo ao primeiro paciente está representado na Figura 2. A biblioteca *genArise*[1] em R[2] permite corrigir o *background*, normalizar, filtrar e identificar genes diferencialmente expressos num só *array* de dois canais. Com base nesta biblioteca, procedeu-se assim à análise completa do *array* referente ao primeiro doente.

ID	Art	Ven	BgArt	BgVen
…	…	…	…	…
Id1192	3484.730111	2716.285714	965.944444	1374.15812
Id1193	2866.54152	2820.299281	997.219048	1415.209402
Id1194	26019.97111	12429.78425	1130.161905	1413.109524
…	…	…	…	…

Figura 2 – Excerto do ficheiro *chip1.txt* contendo o *array* do primeiro paciente

1.1. R Studio e biblioteca genArise

Para realização da análise dos dados, seu processamento e representação gráfica, optou-se pela utilização do RStudio[3], um IDE (*Integrated Development Environment*) para a linguagem de programação R, particularmente direcionada à análise estatística.

Na análise individual do primeiro doente, optou-se pela biblioteca genArise. Este *package* contém funções específicas para leitura, representação e análise de dados de *microarray*. Assim, esta biblioteca é indicada para as etapas pretendidas, como correção de *background*, normalização e seleção de genes diferencialmente expressos. Excertos do código utilizado para a realização da análise ao primeiro paciente serão incluídos ao longo do relatório. O código simplificado bem como a descrição das funções da biblioteca genArise utilizadas podem ser consultados em anexo.

1.2. Diagrama de dispersão, MA plot e RI plot

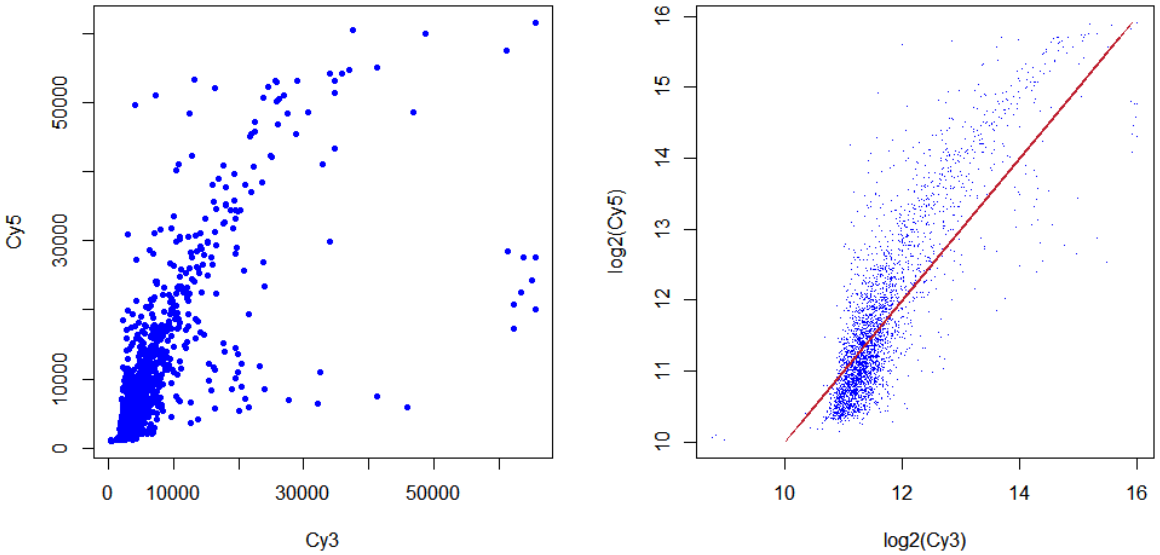
A primeira etapa consiste em extrair a informação do ficheiro referente ao primeiro doente e representá-la. O ficheiro *chip1.txt* contém os níveis de expressão dos genes do tecido submetido ao regime Arterial (*Art*), dos genes do tecido submetido ao regime venoso (*Ven*) e respetivos valores de *background* (*BgArt* e *BgVen*) – ver Figura 2. Sabe-se ainda que a amostra *Art* corresponde ao canal vermelho (ou Cy5) e a amostra *Ven* ao canal verde (ou Cy3). Assim, é necessário extrair as colunas referentes aos dois canais do *microarray Art* e *Ven*, bem como as colunas referentes ao *background* e índices, e guardar num objeto do tipo *Spot* da nossa biblioteca genArise, ideal para o efeito.

Para visualizarmos graficamente as intensidades observadas nos dois canais Art e Ven, podemos utilizar gráficos distintos. Um diagrama de dispersão que confronta as intensidades

cruas entre os dois canais é a forma mais simples de visualizar estes dados – Figura 3a. Denotando este rácio de expressão por T_k,

$$T_k = R_k / G_k \quad (1)$$

onde R_k representa a intensidade para a amostra (red) e G_k para a referência (green). Contudo, é comum aplicar o logaritmo a estas intensidades – Figura 3b. Além de algumas vantagens de representação, a transformação logarítmica elimina a inconsistência nos espaços de regulação [3]. Os genes que são up-regulated terão rácios de expressão de 1 a +∞, por exemplo se R/G=2G/G=2, enquanto que os genes down-regulated terão fatores entre 0 e 1, por exemplo com R/G=R/2R=0.5. Assim, o domínio da regulação passa de]0, +∞ [para]- ∞, +∞ [e passa a ser contínuo com equilíbrio em 0 em vez de 1.



a) intensidades naturais b) intensidades logarítmicas

Figura 3 – Diagramas de dispersão inicial para o primeiro paciente

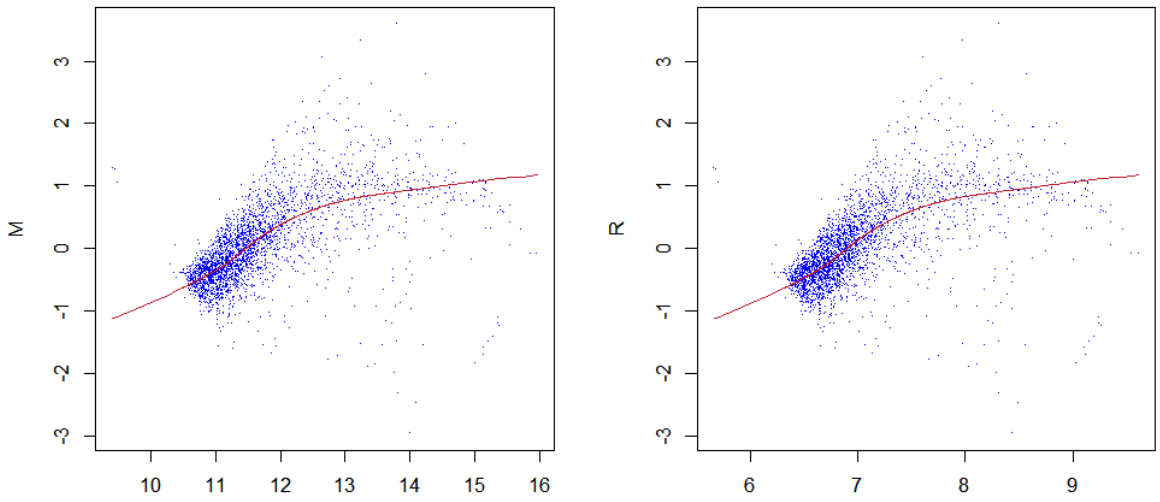
Existem ainda outras formas de representar a relação entre os dois canais. Dois dos mais relevantes e que podem ser encontrados na biblioteca genArise são o *MA-plot* e o *RI-plot*. O MA-plot – Figura 4a - confronta o rácio logarítmico entre R e G com a média de intensidades logarítmicas. Analiticamente, pode escrever-se:

$$\begin{cases} M = \log_2(R/G) = \log_2(R) - \log_2(G) \\ A = \frac{1}{2}\log_2(RG) = \frac{1}{2}(\log_2(R) + \log_2(G)) \end{cases} \quad (2)$$

onde R é a intensidade de Cy5 e G de Cy3. Similar ao MA-plot, o RI-plot – Figura 4b - compara grandezas semelhantes, apresentando-se como uma representação alternativa:

$$\begin{cases} R = \log_2(R/G) = \log_2(R) - \log_2(G) \\ I = \log(RG) = \log(R) + \log(G) \end{cases} \quad (3)$$

Estes gráficos estão na verdade relacionados com o gráfico Cy3/Cy5 da Figura 3a, podendo ser obtidos por uma rotação de 45º e uma ampliação conveniente. Se os genes se expressassem igualmente nos dois canais, nestas duas últimas representações a nuvem de pontos deveria alinhar-se horizontalmente em y=0.



a) MA-plot b) RI-plot

Figura 4 – Diagramas iniciais MA e RI para o primeiro paciente

O facto de os valores de expressão não se disporem numa linha horizontal no *MA* ou *RI-plot* ou, paralelamente, não se disporem numa reta com declive praticamente unitário nos diagramas de dispersão, indica a necessidade de pré-processamento ou normalização [4].

Para importar os dados do ficheiro *chip*1.txt, utilizou-se a função read.*Spot* e guardou-se os dados *microarray* num objeto do tipo *Spot*. Para gerar os gráficos das Figuras 3 e 4, além da função plot do R, utilizou-se *cys.plot*, *ma.plot* e *ri.plot* da biblioteca genArise. O código para esta parte é apresentado na Figura 5 e as funções utilizadas detalhadas na secção de Anexo.

```
# INSTALLING AND IMPORTING genArise PACKAGE

if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

BiocManager::install("genArise")

library(genArise)

# READING DATA FILE AND CREATING SPOT OBJECT

file_name<- "data/chip1.txt"

#patient1 <- read.dataset(file.name = 'chip1.txt', cy3=3, cy5=2, ids=1, symdesc=NULL, zscore=NULL, type=5, header = TRUE, sep="\t")
# os dados n?o s?o do tipo RI nem MA e por isso n?o conseguem ser lidos pelo read.dataset usa-se read.table que converte em tabela
patient1_table <- read.table(file_name, header = TRUE, sep="\t", quote = "", dec = ".")
#mas pode se usar read.spot para criar logo classe Spot

patient1 <- read.spot(file_name, cy3=3, cy5=2, bg.cy3=5, bg.cy5=4, ids=1, header = TRUE, sep="\t")
patient1

# a) representar graficamente as intensidades observadas nos dois canais atrav?s do diagrama de dispers?o e do MA plot

par(mfrow=c(1,2), bg='black')
plot(patient1_table[,3], patient1_table[,2],pch=20, col="yellow", xlab = "Cy3", ylab = "Cy5") # dispersion
cys.plot(patient1, col="yellow") #dispersion with log

par(mfrow=c(1,2), bg='white')
ma.plot(patient1, "yellow")
ri.plot(patient1, "yellow")

```

Figura 5 – excerto do código utilizado na inicialização e impressão dos diagramas

1.3. Remoção de *background*

Uma primeira fase na análise computacional de um *microarray* consiste em técnicas de processamento de imagem para extração de características. Em concreto, este processo compreende a conversão de imagens com intensidades de hibridização diferentes obtidas pelo

scanner em valores numéricos qualitativos para cada canal em cada característica. Deste processo resulta o conjunto de características contendo pelo pelo menos valores de intensidade: o da própria característica e a do *background*.

Na fase seguinte, pode então tratar-se dos dados: limpá-los ou transformá-los por forma a poderem ser processados ou interpretados. Este processo inclui, por exemplo, remover características incorretas ou calcular o logaritmo das intensidades, como descrito anteriormente. Outra etapa importante no tratamento dos dados é a remoção do *background*.

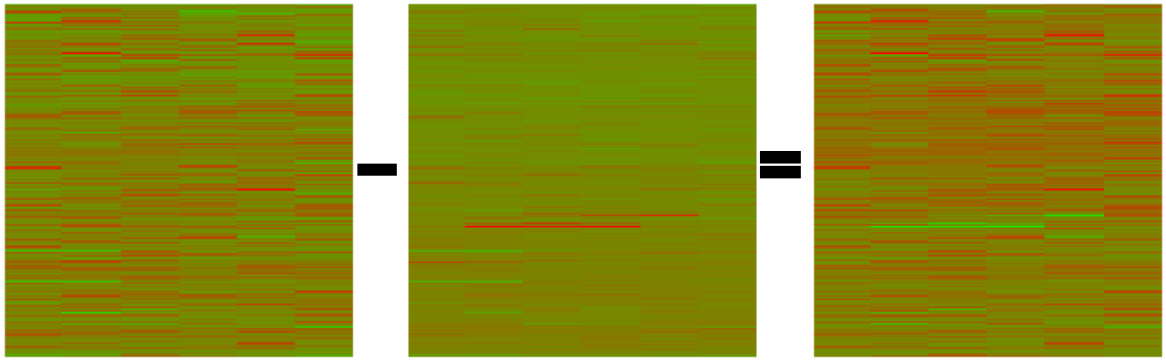


Figura 6 – Representação da subtração de *background*

O *background* representa a intensidade de fundo e deve portanto ser subtraído ao valor de intensidade dessa característica. A Figura 6 foi extraída com base nos dados do primeiro paciente. O número de colunas e linhas por *array* não é relevante e esta representação é meramente ilustrativa para o efeito. Dos três *arrays* que consitutem a Figura 6, o primeiro guarda os valores de intensidades iniciais, utilizados para representar as Figuras 3 e 4; a segunda imagem guarda os valores de *background*, mais baixos que os da primeira imagem; a última imagem resulta da subtração dos valores do *background* às intensidades das características. É necessário ter alguns cuidados quando se remove o *background* já que pode acontecer este ter intensidade superior à da característica resultando numa diferença com valor negativo. Para os dados desta experiência, esta questão não foi um problema.

Esta subtração é realizada fazendo uso da função *bg.correct* da biblioteca genArise. O código utilizado para esta etapa encontra-se na Figura 7. Foram também obtidos novamente gráficos de representação da relação entre os dois canais: diagrama de dispersão logarítmica e *MA-plot* – Figura 8a e 8b respetivamente.

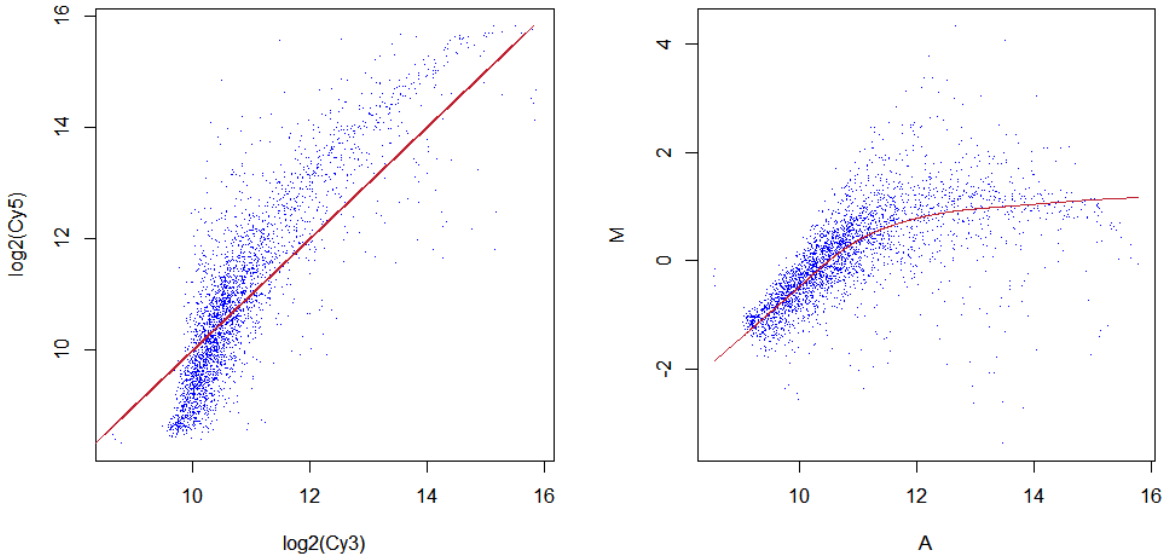
```
# b) background subtraction


par(mfrow=c(1,3), bg='white')
datos <- attr(patient1, "spotData")# Extract spot data
M <- log(datos$Cy3, 2) - log(datos$Cy5, 2)
K <- log(datos$BgCy3, 2) - log(datos$BgCy5, 2)
imageLimma(z = M, row = 499, column = 6, meta.row = 1, meta.column = 1, low = NULL, high = NULL)
imageLimma(z = K, row = 499, column = 6, meta.row = 1, meta.column = 1, low = NULL, high = NULL)
imageLimma(z = M-K, row = 499, column = 6, meta.row = 1, meta.column = 1, low = NULL, high = NULL)

bg_corrected_patient1 <- bg.correct(patient1)

par(mfrow=c(1,2))
cys.plot(bg_corrected_patient1, "yellow") #dispersion with log
ma.plot(bg_corrected_patient1, "yellow") #ma plot
#ri.plot(bg_corrected_patient1) #ri plot
```

Figura 7 – excerto do código utilizado para remoção de *background*



a) Diagrama de dispersão logarítmica b) *MA-plot*

Figura 8 – Diagrama de dispersão logarítmica e *MA-plot* após remoção de *background*

As linhas a vermelho são, para a Figura 8a, a reta x=y e, para a Figura 8b, a regressão não-linear *loess* (ou *locally weighted ploynomial regression*). No primeiro caso, como explicado anteriormente, esta reta representa um alvo, já que é desejável que os pontos no gráfico de dispersão se disponham ao longo desta reta como sinal de equilíbrio na sua relação. Quanto à Figura b, esta regressão é obtida através dos sucessivos pontos médios da regressão linear para uma dada janela deslizante, com vizinhança ε. Seria desejável paralelamente que esta reta, obtida por *loess*, se aproximasse de uma reta horizontal em y=0: isto significaria que os pontos se dispunham simetricamente em relação a este eixo, ou seja que os seus níveis de expressão dos respetivos genes em cada um dos canais eram semelhantes. A forma das curvas de pontos faz notar fortes não-linearidades, possivelmente introduzidas pela diferença de eficiência na marcação com fluoróforos verde e vermelho e na sua deteção pelo scanner. Isto pede um procedimento adequado de normalização dos dados.

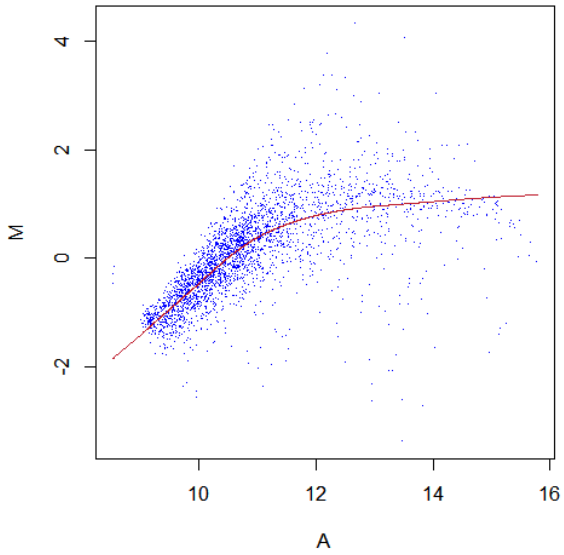
1.4. Normalização dos dados

Para medir expressão diferencial entre duas amostras distintas, é necessário comparar as intensidades de Cy3 e Cy5 em igual medida, ou seja é necessário que as intensidades estejam em equilíbrio. A normalização intra-*array* é assim o processo de eliminar possíveis fontes de erro sistemático ou *bias* introduzidos pelo processo experimental para permitir que duas amostras sejam devidamente comparadas.

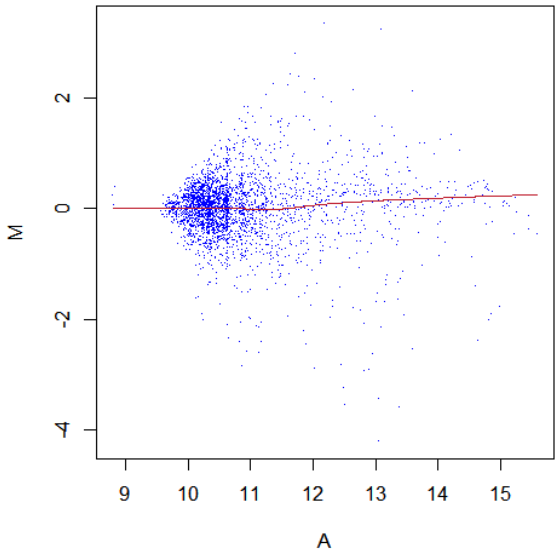
Algumas formas de normalização intra-*array* são regressão linear Cy5/Cy3, regressão linear *M/A* e regressão *loess M/A*, esta última sendo particularmente indicada quando a relação entre os dois canais é não-linear. A normalização por regressão *loess M/A* pega no *MA-plot* e na curva de regressão *loess* (ver, por exemplo, Figura 8a) e para cada ponto calcula uma nova posição de acordo com:

rácio normalizado = rácio logarítmico da regressão – rácio logarítmico original (4)

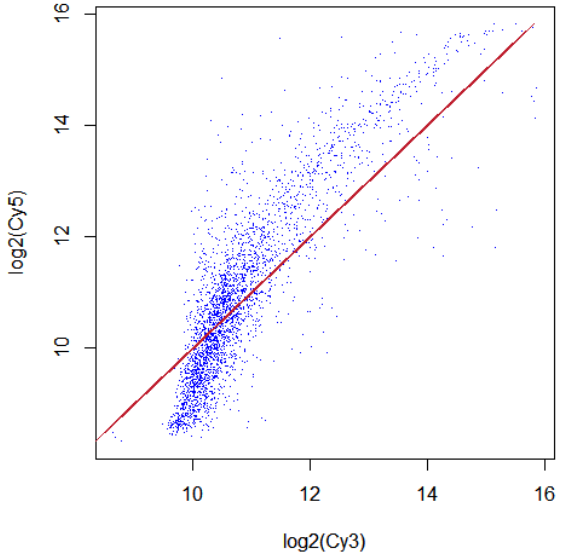
Este processo mapeia a nuvem de pontos, até aqui em torno da curva de regressão *loess*, para zonas próximas de y=0, onde ficará a nova curva de regressão uma vez terminada a normalização – Figura 9a e 9b. É apresentado também o resultado da normalização nos gráficos Cy5/Cy3 onde, como era de esperar, os pontos passam a dispor-se numa reta de declive unitário – Figura 9c e 9d.



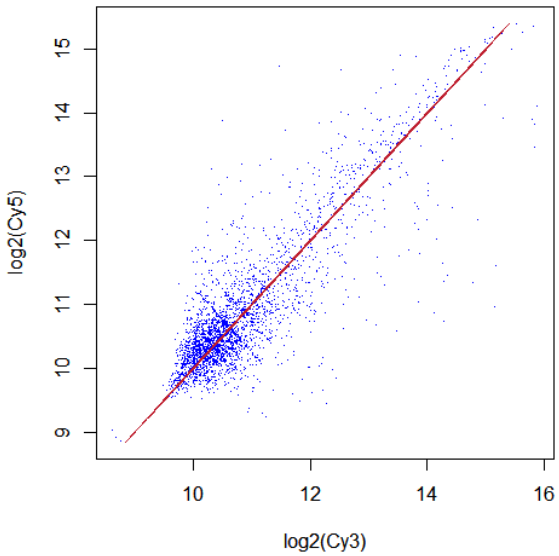
a) diagrama de dispersão original



b) diagrama de dispersão normalizado



c) diagrama de dispersão original



d) diagrama de dispersão normalizado

Figura 9 – efeito da normalização no diagrama de dispersão e MA-plot

Existem vários métodos de normalização intra-*array*, muitas vezes originando resultados distintos. Ao nível da biblioteca genArise existem dois métodos de normalização, *grid.norm* e *global.norm*, tendo sido escolhido o segundo referente à normalização intra *array*. A descrição desta função encontra-se disponível em anexo. Inclui-se o código utilizado para normalizar os dados e imprimir os gráficos na Figura 10.

```
# c) normalizing the array:
# non linear (loess) regression of log ratio against average intensity
#1- for each feature construct A = avergae log intensity ; M = log ratio
#2- produce MA plot
#3 - apply loess regression to the data
#4- for each feature calculate NormalizedRatio=FittedLogRatio-RawLogRatio

normalized_bg_patient1 <- global.norm(mySpot = bg_corrected_patient1)

par(mfrow=c(1,2))
cys.plot(bg_corrected_patient1, "yellow") #dispersion with log
cys.plot(normalized_bg_patient1, "yellow") #dispersion with log
ma.plot(bg_corrected_patient1, "yellow") #ma plot
ma.plot(normalized_bg_patient1, "yellow") #ma plot
```

Figura 10 – excerto do código utilizado na normalização dos dados

1.5. Cálculo do Z-score

O Z-score é uma medida do quão longe um valor está da média μ do conjunto em que a distância é medida em desvios padrão σ, segundo a fórmula

$$z = \frac{x - \mu}{\sigma} \tag{5}$$

A função Zscore da biblioteca genArise permite identificar genes diferencialmente expressos. A função utiliza uma janela deslizante em redor de cada ponto para calcular a média e desvio padrão no seu interior e define o Z-score normalmente onde o Z mede o número de desvios padrão que cada ponto dista da média. Assim, procedeu-se ao cálculo do Z-score para cada gene através deste comando e construiu-se o gráfico adequado (*MA-plot*) com recurso à função *zscore.plot*, representado na Figura 11. A função inclui um código de cores responsável pelos diferentes tons na imagem: cor-de-rosa para Z<1, amarelo para 1<Z<1.5, vermelho para 1.5<Z<2 e preto para Z>2.

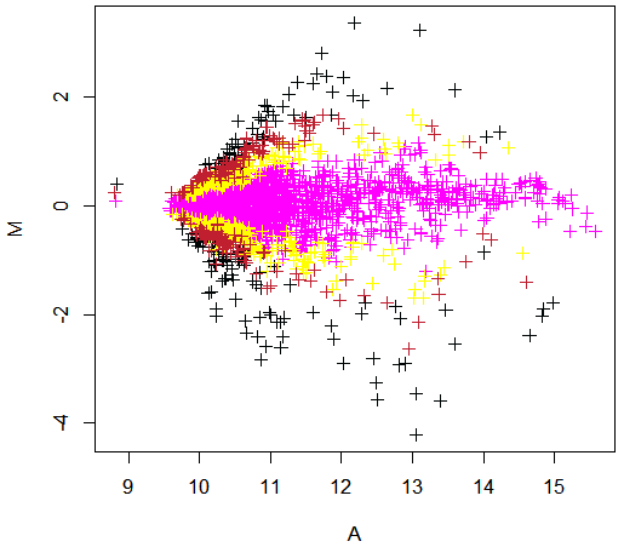


Figura 11 – MA-plot com o Z-score de cada gene

Neste gráfico podemos observar que existem vários pontos, ou seja genes expressos, com Z-score superior a 2. O valor de *cutoff* para identificar os genes diferencialmente expressos, depende do contexto do problema e da intenção do investigador, a biblioteca GenArise considera os três valores acima mencionados (1, 1.5 e 2). Tendo em conta o considerável

número de genes com Z-score acima de 2, pode-se optar por este valor, obtendo uma análise mais conservadora.

Gene_ID	Z_score
Id428	3.863192
Id2037	3.595207
Id372	3.444038
Id233	3.258511
Id2830	3.142282

Gene_ID	Z_score
Id336	-3.816216
Id1626	-3.895506
Id476	-4.043949
Id2748	-4.239489
Id1362	-4.333539

Tabela 1– 5 genes up e down regulated

Observando os Z-scores, verifica-se que existem 811 genes com o módulo do Z-score de expressão superior a 1; 356 superior a 1.5 e 156 superior a 2. Dos 156 genes que consideramos como diferencialmente expressos (*cutoff* = 2), 60 são *upregulated*, ou seja com valores Cy5 superiores e 96 são *downregulated*, valores de Cy3 superiores. Os 5 genes mais expressos e os respetivos Z-scores podem ser encontrados na tabela 1. O código em R utilizado nesta secção encontra-se na Figura 12.

```
# d) Zscore

z_normalized_bg_patient1 <- Zscore(normalized_bg_patient1, type="ma")
par(mfrow=c(1,1))
data(z_normalized_bg_patient1)
Zscore.plot(z_normalized_bg_patient1)

### tabela Zscores
cut_off <- 2

z_scores <- z_normalized_bg_patient1@dataSets$Zscore
count=0
for(i in 1:2994){
  if(abs(z_scores[i])>=cut_off) count=count+1
}

ids <- z_normalized_bg_patient1@dataSets$Id
tabela <- data.frame("Gene_ID" = ids,
                     "Z_score" = z_scores)

tabela_cutoff <- subset(tabela, subset = abs(Z_score) > cut_off)

tabela_cutoff <- tabela_cutoff[order(-tabela_cutoff$Z_score),]

```

Figura 12 – excerto do código utilizado no cálculo do Z-score e dos genes diferencialmente expressos

2. Análise aos três pacientes

2.1 Normalização intra-*array*

Tal como na primeira etapa, leu-se a informação relativa aos *chip*s 2 e 3, juntamente já com a informação do *chip* 1 e procedeu-se à criação do *Spot*. De igual forma ao *chip* 1, aplicaram-se também os procedimentos de remoção do *background* e da normalização intra-*array* com a função *global.norm* da biblioteca GenArise (figuras 13 e 14).

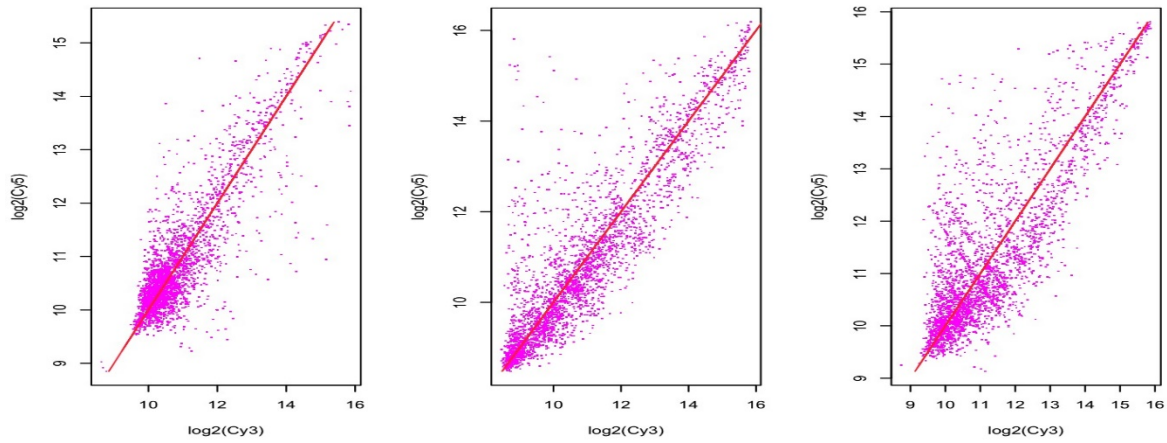


Figura 13 –Cys plot dos 3 *chip*s normalizados

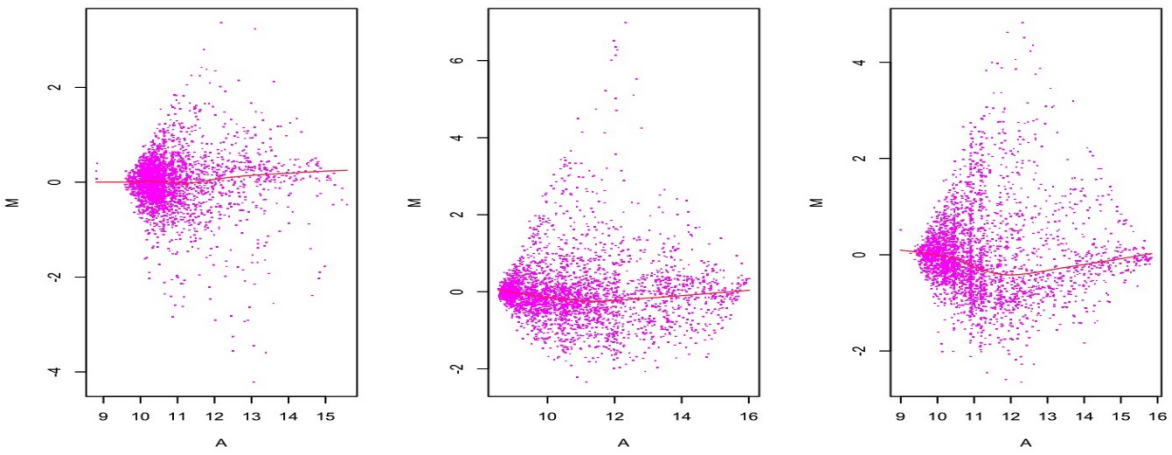


Figura 14 –MA plot dos 3 *chip*s normalizados

É importante referir, pela figura 14 que a normalização pareceu ter sido menos eficiente para o *chip*3 pois a reta de regressão *loess* para apresenta regiões com valor absoluto de declive superior quando comparado com esta mesma reta do *chip* 1 e 2, sendo que o ideal era estar o mais concordante com a reta y=0. Na Figura 15 pode ler-se o código utilizado para a normalização.

```
## e) normalizar cada um dos 3 arrays

## corrigir o background
bg_corrected_patient1 <- bg.correct(spot1)
bg_corrected_patient2 <- bg.correct(spot2)
bg_corrected_patient3 <- bg.correct(spot3)

## normalizar como no primeiro procedimento
normalized_bg_patient1 <- global.norm(mySpot = bg_corrected_patient1)
normalized_bg_patient2 <- global.norm(mySpot = bg_corrected_patient2)
normalized_bg_patient3 <- global.norm(mySpot = bg_corrected_patient3)

par(mfrow=c(1,3))
cys.plot(bg_corrected_patient1) #dispersion with log
cys.plot(bg_corrected_patient2) #dispersion with log
cys.plot(bg_corrected_patient3) #dispersion with log
ma.plot(normalized_bg_patient1) #ma plot
ma.plot(normalized_bg_patient2) #ma plot
ma.plot(normalized_bg_patient3) #ma plot

```

Figura 15– excerto do código utilizado na normalização dos dados dos 3 pacientes

2.2 Normalização por *centering*

De seguida é pedido para realizar a normalização entre-*arrays*, passo necessário para poder comparar os 3 *arrays* correspondentes aos *chip*s. Existem vários métodos a considerar para esta normalização de entre eles o scaling (subtrair a cada valor do *array* normalizado previamente a sua média) e o *centering* (subtrair a média e dividir pelo desvio padrão). No enunciado é sugerido o *centering*.

```
# f) Normalizar entre arrays e boxplots

par(mfrow=c(1,3))

#equivalente a:
#boxplot(log2(patient1@spotData$Cy5/patient1@spotData$Cy3),col=rainbow(6))
#boxplot(log2(patient2@spotData$Cy5/patient2@spotData$Cy3),col=rainbow(6))
#boxplot(log2(patient3@spotData$Cy5/patient3@spotData$Cy3),col=rainbow(6))
M_array <- array(c(log2(patient1@spotData$Cy5/patient1@spotData$Cy3),
                   log2(patient2@spotData$Cy5/patient2@spotData$Cy3),
                   log2(patient3@spotData$Cy5/patient3@spotData$Cy3)),
                   dim = c(2994,3))

boxplot(M_array,col=rainbow(6))

M_array_bg_normalized <- array(c(log2(normalized_bg_patient1@spotData$Cy5/normalized_bg_patient1@spotData$Cy3),
                                 log2(normalized_bg_patient2@spotData$Cy5/normalized_bg_patient2@spotData$Cy3),
                                 log2(normalized_bg_patient3@spotData$Cy5/normalized_bg_patient3@spotData$Cy3)),
                                 dim = c(2994,3))

boxplot(M_array_bg_normalized,col=rainbow(6))

median_patients=c(0,0,0)
mad_patients=c(0,0,0)
centered_M_array= matrix(0, nrow = 2994, ncol = 3)
for(j in 1:3){
  median_patients[j]= median(M_array[,j])
  mad_patients[j]=mad(M_array[,j])
  for(i in 1:2994){
    centered_M_array[i,j]=(M_array[i,j]-median_patients[j])/mad_patients[j]
  }
}

boxplot(centered_M_array,col=rainbow(6))

# confirmar q apos normalizacao, a media/mediana deve ser 0 e a std/mad 1
median_patients_pos=c(0,0,0)
mad_patients_pos=c(0,0,0)
for(j in 1:3){
  median_patients_pos[j]= median(centered_M_array[,j])
  mad_patients_pos[j]=mad(centered_M_array[,j])
}

```

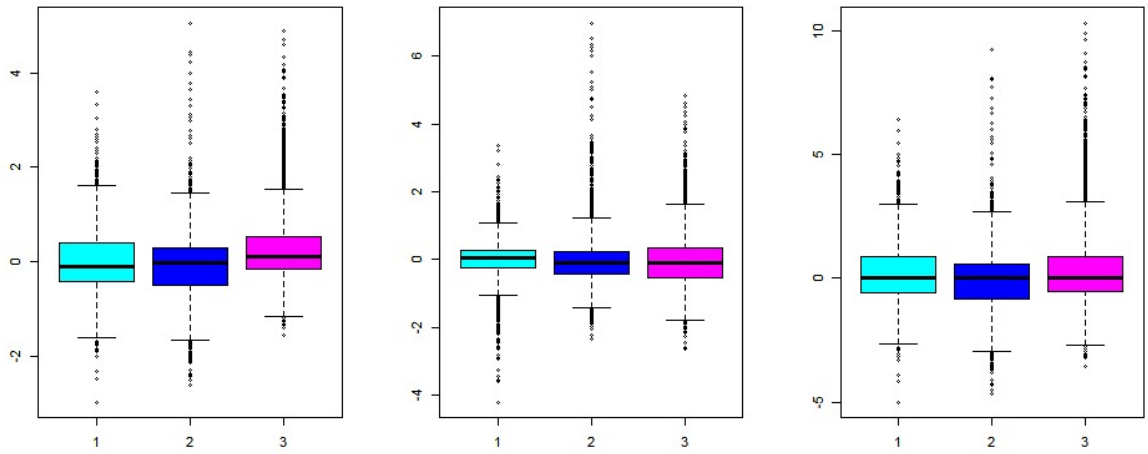
Figura 16– excerto do código utilizado no *centering* dos dados

O *centering* é uma forma muito utilizada de normalização entre-*arrays* já que garante que a média e desvio padrão de todas as distribuições ficam iguais. É também comum, em vez da média e desvio padrão, utilizar a mediana e o desvio absoluto médio já que formam uma solução mais robusta à presença de *outliers* ou a ditribuições que fujam à Gaussiana [5]. Assim, o método de *centering* consiste analitacamente em, para cada medida x_i do *array*, mapear:

$$x_i \rightarrow (x_i - \bar{x}) / s_x \qquad (6)$$

onde $\bar{x}$ é a média (ou mediana) do *array* e s_x o seu desvio padrão (ou desvio absoluto médio). O código utilizado para implementar esta secção encontra-se na Figura 16.

Uma excelente forma de representar os dados de *microarrays* diferentes para comparação e análise é o *box plot*. Este tipo gráfico permite comparar as distribuições de intensidades logarítmicas e rácios logarítmicos de genes de diferentes *microarrays* e permite retirar assim relevantes conclusões sobre a necessidade de normalização entre eles. Os gráficos da Figura 17 acompanham a relação entre os 3 *arrays* ao ínicio, depois da normalização da alínea anterior e depois da normalização por *centering*.



a) dados não-normalizados b) após normalização intra-*array* c) após *centering*

Figura 17 – *boxplot*s referente aos dados dos 3 pacientes

No *box plot* os dados estão representados entre os traços horizontais limítrofes superior e inferior: fora desta área os *outliers* são representados por pontos e dentro desta área os dados dividios em quatro regiões, os quartis. A caixa colorida corresponde aos dois quartis centrais, sendo que com um traço mais escuro está representada a mediana. Ora inicialmente, como é possível ver pela Figura 17a os três *chip*s referentes a cada um dos pacientes têm distribuições muito diferentes, com medianas, máximos e mínimos distintos. Com a normalização entre-*arrays* por *centering*, cada caixa (box) deve passar a ter mediana em zero e entre elas devem ter o mesmo tamanho, como de facto acontece na Figura 17c. Com a normalização intra-*array* isto também se deve verificar, o que neste caso não é tão notório e que portanto não exclui a necessidade à mesma do outro tipo de normalização. No caso da normalização por scaling, a risca preta da mediana estaria à mesma alinhada para os três *arrays* mas a amplitude interquartil poderia não ter o mesmo tamanho.

É também possível verificar o sucesso da normalização analiticamente. Como se pode ver na Figura 16, foi realizado um teste aos dados normalizados para aferir a mediana e o MAD (desvio absoluto médio) tendo se obtido para as três amostras uma mediana nula e um MAD unitário, como esperado. Pode ainda interpretar-se que o M-value (eixo y) está próximo de zero desde início já que a maioria dos genes não é diferencialmente expressa. A percentagem de genes diferencialmente expressos deverá estar nos extremos do *box plot*, como *outliers*.

Apenas assim se encerra o processo de normalização dos dados e se pode passar ao pós-processamento e análise estatística.

2.3 Z-scores para identificação de genes diferencialmente expressos

De seguida, à semelhança do processo realizado em 1.5, procedeu-se ao cálculo do Z-score para cada gene de cada *chip* e utilizou-se estes valores para identificar os genes diferencialmente expressos em cada amostra. Para o cálculo dos valores de Z-score e sua representação recorreu-se às funções *Zscore()* e *Zscore.plot()* da biblioteca genArise respetivamente. Os gráficos resultantes estão dispostos na Figura 18.

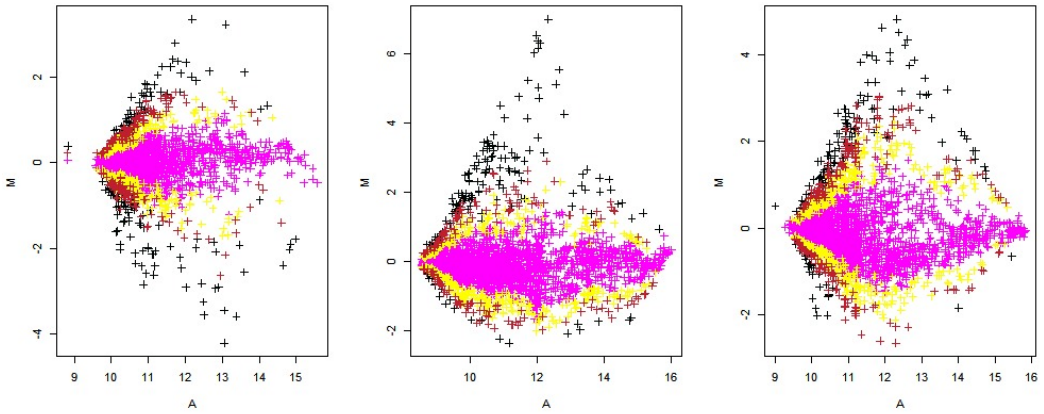


Figura 18 – *MA-plot* com o Z-score de cada gene para os 3 pacientes

Relembra-se pela expressão (5) que o Z-score constitui um indicador na identificação de genes diferencialmente expressos já que o Z mede o número de desvios padrão que cada ponto dista da média. Pelos gráficos da Figura 18 podemos observar que existem vários pontos (em baixa percentagem) com Z-score superior a 2, a preto. Para o valor de *cutoff* para identificar os genes diferencialmente expressos, que novamente depende do contexto do problema, volta a utilizar-se então esse valor que resultará num número de genes diferencialmente expressos distinto para cada amostra.

ANÁLISE Z-SCORE	Genes diferencialmente expressos				Z-score mais elevado		Z-score mais baixo	
	total	%	up	down	GeneID	Z-score	GeneID	Z-score
Paciente 1	156	5.21	60	96	ID428	3.8632	ID1362	-4.3335
Paciente 2	156	5.21	132	24	ID2333	5.3633	ID2882	-2.6029
Paciente 3	142	4.74	114	28	ID2405	4.0049	ID899	-4.3239

Tabela 2– resultados obtidos pela análise do Z-score

Os resultados ilustrados na Tabela 2 condizem com a avaliação qualitativa que se pode tirar da Figura 18. Graficamente, valores de rácio logarítmico M próximos de zero correspondem a genes que não são diferencialmente expressos, sendo que estes encontrar-se-ão mais afastados de y=0. De facto, estabelecer o valor de *cutoff* a 2 (análise relativamente conservadora) leva a ser considerada uma percentagem de aproximadamente 5% dos genes como diferencialmente expressos em cada um, cerca de 150. No primeiro caso, a maioria deles é *downregulated*, ou seja com expressão de Cy5 superior à de Cy3, enquanto que para o segundo e terceiro paciente verfica-se o contrário.

Quando comparamos os resultados e sobrepomos as três listas de genes diferencialmente expressos em cada paciente, é ainda possível observar aqueles com esse comportamento nos três. Neste caso, e para *cutoff* de 2, nenhum gene se mostrou diferencialmente expresso para os três pcientes. O código utilizado nesta secção encontra-se na Figura 19.

```
# g) Zscore for all patients

z_normalized_bg_patient2 <- Zscore(normalized_bg_patient2, type="ma")
z_normalized_bg_patient3 <- Zscore(normalized_bg_patient3, type="ma")
par(mfrow=c(1,3))

Zscore.plot(z_normalized_bg_patient1)

Zscore.plot(z_normalized_bg_patient2)

Zscore.plot(z_normalized_bg_patient3)

### tabela Zscores
cut_off <- 2

z_scores <- z_normalized_bg_patient2@dataSets$Zscore
count_2=0
up_count_2=0
for(i in 1:2994){
  if(abs(z_scores[i])>=2) count_2=count_2+1
  if(z_scores[i]>=2) up_count_2=up_count_2+1
}
ids_2 <- z_normalized_bg_patient2@dataSets$Id
tabela_2 <- data.frame("Gene_ID" = ids_2,"Z_score" = z_scores)
tabela_cutoff_2 <- subset(tabela_2, subset = abs(Z_score) > cut_off)
tabela_cutoff_2 <- tabela_cutoff_2[order(-tabela_cutoff_2$Z_score),]

z_scores <- z_normalized_bg_patient3@dataSets$Zscore
count_3=0
up_count_3=0
for(i in 1:2994){
  if(abs(z_scores[i])>=2) count_3=count_3+1
  if(z_scores[i]>=2) up_count_3=up_count_3+1
}
ids_3 <- z_normalized_bg_patient3@dataSets$Id
tabela_3 <- data.frame("Gene_ID" = ids_3,"Z_score" = z_scores)
tabela_cutoff_3 <- subset(tabela_3, subset = abs(Z_score) > cut_off)
tabela_cutoff_3 <- tabela_cutoff_3[order(-tabela_cutoff_3$Z_score),]

```

Figura 19– excerto do código utilizado na identificação de Z-scores e genes diferencialmente expressos para os 3 pacientes

2.4 *Package* Limma: método bayesiano

Tendo feito a análise pelo *package* genArise com o cálculo dos Z-scores, é indicado também a utilização do método bayesiano Lönnstedt e Speed para fazer a identificação dos genes diferencialmente expressos. Este método é considerado mais estável e com potência melhorada sobretudo para experiências com número reduzido de amostras, no contexto de *microarrays* de dois canais, como o caso deste exemplo. O método baseia-se em calcular uma nova estatística de teste, B_i, que consiste no logaritmo dos odds à posteriori, dado pela expressão (7):

$$B_i = ln\frac{P(I_i = 1|M_i)}{P(I_i = 0|M_i)} = ln\frac{pf_{I_i=1}(M_i)}{(1-p)pf_{I_i=0}(M_i)} \quad (7)$$

Sendo que:

$$M_{ij}|\mu_i, \sigma_i \sim N(\mu_i, \sigma_i)$$

$$I_i = \begin{cases} 0 & \text{if, } \mu_i = 0 \\ 1 & \text{if, } \mu_i \neq 0 \end{cases}$$

p é a proporção de genes considerados como diferencialmente expressos, $M_{ij} = (M_{i1}, \dots, M_{1j})$ e considera-se um gene i como diferencialmente expresso se $B_i > 0$.

O *package* limma contém funções para analisar vários tipos de dados de expressão génica, contendo precisamente métodos para ultrapassar o problema de pequeno número de amostras [6]. Tal como o *package* genArise, permite analisar *microarrays* de dois canais, corrigir o *background* e normalizar os dados, intra e inter *arrays*. Este *package* contém também a abordagem Bayesiana para determinar os genes diferencialmente expressos, utilizando o método acima especificado, sendo por isso útil para o problema em questão. É importante notar que este *package* tem a sua própria metodologia para organizar os dados, que difere substancialmente da metodologia empregada pelo *package* genArise, deste modo não foi possível aplicar diretamente nos dados (*chip* 1, 2, 3.txt) as funções indicadas na documentação. Para contornar esta questão foi feito o seguinte procedimento, tendo por base teórica a documentação das funções do *package* limma (normalizeWithin*Arrays*, normalizeBetween*Arrays*):

1. Criar um objeto "MAlist"
    a. Contendo o *array* A com o *background* corrigido
    b. Contendo o *array* B com o *background* corrigido e com os dados já normalizados previamente pelo global.norm(), do *package* GenArise.
2. Normalizar entre-*arrays* o objeto MAlist, usando a função própria do *package* limma (normalizeBetween*Arrays*).
3. Aplicar a função lmFit e de seguida o eBayes, prosseguindo com a análise.

```
132 library(limma)
133
134 A_array <- array(c(0.5*log2(patient1@spotData$Cy5*patient1@spotData$Cy3),
135                     0.5*log2(patient2@spotData$Cy5*patient2@spotData$Cy3),
136                     0.5*log2(patient3@spotData$Cy5*patient3@spotData$Cy3)),dim = c(2994,3))
137
138 ## Segundo a informação das funções normalize withinArray e normalizBetweeArray do limma:
139 ## Utilizar o M array e o A array  ja com o background corrigido e o M_array normalizado ja anteriormente
140 A_array_bg_corrected <- array(c(0.5*log2(bg_corrected_patient1@spotData$Cy5*bg_corrected_patient1@spotData$Cy3),
141                                 0.5*log2(bg_corrected_patient2@spotData$Cy5*bg_corrected_patient2@spotData$Cy3),
142                                 0.5*log2(bg_corrected_patient3@spotData$Cy5*bg_corrected_patient3@spotData$Cy3)),dim = c(2994,3))
143
144 MA <- new("MAList", list("M"= M_array_bg_normalized, "A" = A_array_bg_corrected))
145
146 ## De seguida é necessario normalizar entre arrays e para isso pode-se utilizar a função do limma
147 MA <- normalizeBetweenArrays(MA , method = "scale")
148
149
150 #design <- c(-1,1,-1,1)
151 # -1 corresponds to dye-swap
152 MA_list_fit <- lmFit(MA)
153 MA_list_eBayes <- eBayes(MA_list_fit, proportion = 0.05)
```

Figura 20– excerto do código utilizado para o *package* limma

A função *lmFit* vai estimar a variabilidade nos dados a partir do objeto *MAList*, criando um modelo linear para cada gene. De seguida, a função *eBayes* irá calcular estatísticas t moderadas e a estatística B que representa o *log odds* da expressão diferencial. Por último, a utilização da função *topTable* permite produzir uma tabela com os genes ordenados pela estatística pretendida, por defeito ordenados pelo valor B, que é o pretendido neste contexto.

O *volcanoplot* permite-nos obter uma representação gráfica do *log fold changes* vs o *log odds* à posteriori, B – Figura 21. Estas funções utilizadas encontram-se especificadas em Anexo.

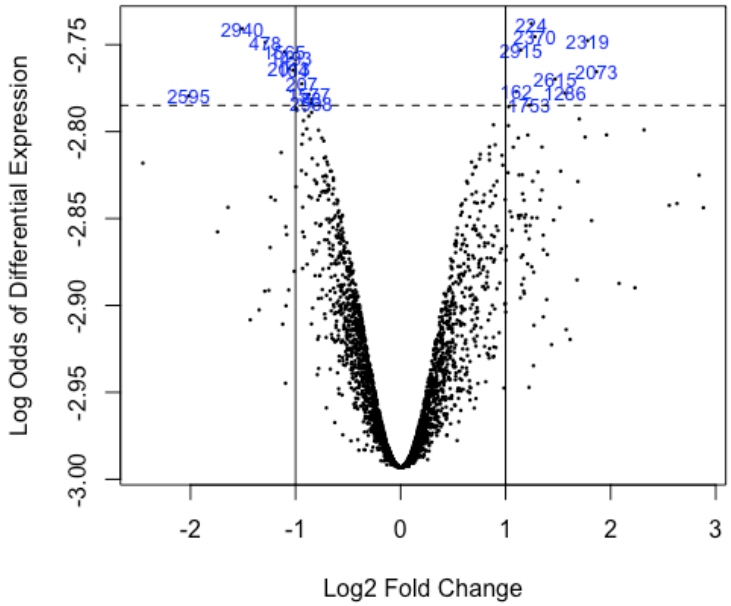


Figura 21– Gráfico (*volcanoplot*) resultante do método bayesiano

Foi escolhido que iriam ser considerar os 20 genes mais diferencialmente expressos, sendo por isso o valor de *cutoff* o valor de significância associado ao 20º gene escolhido, ou seja, B =-2.784904. O exp do B dá-nos o odds da expressão diferencial, ou seja exp(-2.784904) = 0.06173502, e retirando esta quantidade a uma unidade, obtemos a probabilidade de o gene não ser diferencialmente expresso, logo 1- 0.06173502= 0.939. Este valor indica-nos que existe uma probabilidade de cerca de 0.939 de o 20º gene considerado não ser diferencialmente expresso, sendo um valor muito perto de 1.

Na Tabela 3 encontram-se representados os resultados desta análise. O valor p ajustado resulta do método Benjamini-Hochberg, como escolhido na função topTable.

ID	logFC	AveExpr	t	P.Value	adj.P.Val	B
Id224	1.244587	12.14529	4.683984	0.0039054	0.8988501	-4.388687
Id2940	-1.507473	13.16637	-4.568239	0.0043856	0.8988501	-4.391452
Id2370	1.278259	10.87905	4.387529	0.0052761	0.8988501	-4.396077
Id2319	1.779939	12.19074	4.303891	0.0057566	0.8988501	-4.398353
Id478	-1.287366	11.72450	-4.268985	0.0059716	0.8988501	-4.399330

Tabela 3– Informação obtida dos cinco genes com valor de B mais elevado

2.5 Análise e comparação entre os dois métodos

Na análise realizada pelo Z-score foram selecionados os genes diferencialmente expressos para cada paciente, numa escolha de *cutoff* de Z=2 e resultando em cerca de 5% dos genes totais. O número de genes assinalados foi relativamente semelhante entre pacientes não havendo contudo nenhum gene que se tenha manifestado diferencialmente expresso nos 3 pacientes.

Os resultados na análise das três amostras pelo método bayesiano Lönnstedt and Speed sugerem que não há genes estatisticamente diferencialmente expressos, pois o valor B mais elevado não é superior a zero, o que significa que o odds à posteriori é inferior a 1 e por isso tendemos a aceitar a hipótese que o gene não é diferencialmente expresso (tabela 3). Os valores p ajustados vão de encontro a esta hipótese, sendo valores muito superiores aos níveis usuais de significância. O facto de não haver genes estatisticamente significativos pode ser pelo facto de a variância poder não ser homongénea para cada *array*, o que aumenta a variabilidade total e torna difícil a inferência.

O método bayesiano baseia-se em informação à priori que neste caso foi escolhida através da análise anterior com os Zscores: fazendo a média do número de genes diferencialmente expressos considerados em cada uma das três amostras, obeteve-se um valor aproximado de 0.05, sendo incluído no código como *eBayes*(..., *proportion* = 0.05).

Além da divisão horizontal em função das *log odds* que permite distinguir os genes com B>0 (ou elevado) dos demais, pode determinar-se também o conjunto dos genes potencialmente diferencialmente expressos através da *log fold change*. Neste caso, o *cutoff* é vertical e marca-se em x=1, x=-1. Este valor provém de $\log_2 2=1$ e implicitamente do facto de indicar os níveis de expressão de Cy5 serem o dobro dos de controlo.

Ainda que os valores de B não sugiram presença de genes diferencialmente expressos, é conveniente considerar aqueles que apresentam maior probabilidade. Assim, os 20 genes selecionados como diferencialmente expressos correspondem aos valores B mais elevados, mas também na sua maioria estão fora do intervalo [-1, 1] de *log fold change* que é usado como um *cutoff* vertical no volcanoplot apresentado. Por norma deve-se considerar estes dois *cutoff* (vertical e horizontal) quando se analisa este gráfico para identificar os genes diferencialmente expressos, sobretudo em casos em que a variância é muito grande o que leva a problemas na sua estimação em amostras pequens [7]. A tabela completa com os 20 genes diferencialmente expressos encontra-se em anexo.

Conclusão

Neste trabalho propunhamo-nos a determinar os genes mais e menos diferencialmente expressos com base na análise *microarray* de amostras de tecido de veia safena de três doentes com uma patologia cardíaca associada.

Numa primeira fase, o foco foi a preparação dos dados para análise. Esta etapa consistiu, por um lado, em limpar e transformar os dados gerados pelas ferramentas de extração de características: subtração de *background* e transformação logarítmica. Por outro lado, incluiu também a normalização dos dados das amostras, tanto dentro do próprio *array* como entre os 3 *arrays*. Para isso foi utilizada a regressão não-linear (*loess*) e o *centering*, respetivamente. Foram ainda abordadas diversas formas de representação de dados como *Cys plot, MA-plot, RI-plot* e *box plot.*

Numa segunda etapa, uma vez tratados e normalizados os dados, utilizou-se dois métodos diferentes para averiguar o conjunto de genes diferencialmente expressos. Com a técnica de Z-score foi possível associar um valor de distância de cada gene à média da sua amostra, servindo para, através de uma fronteira, identificar aqueles que mais se afastavam para depois calcular a interseção dos três conjuntos de candidatos. Por outro lado, com o método *bayesiano* de *Lönnstedt e Speed*, foi possível confrontar *fold-change* e *t-statistic* num *volcano plot* de onde foi possível gerar uma hierarquização distinta dos genes consoante a probabilidade de expressão diferencial.

Acima de tudo, com esta análise, é relevante conseguir gerar um ranking para os nossos genes e identificar os que mais provavelmente são diferencialmente expressos. Para isto, é até desejável deixar alguma margem na seleção do topo da tabela, neste caso 20 genes, para aumentar a probabilidade de estarem a ser incluídos todos os diferencialmente expressos. Assim, não é grave estimar por excesso e ter alguns falsos positivos já que esta análise deve depois ser corroborada por testes experimentais.

Referências

[1] Kaliyappan, K., Palanisamy, M., Govindarajan, R., & Duraiyan, J. (2012). Microarray and its applications. *Journal of Pharmacy and Bioallied Sciences*, *4*(6), 310. https://doi.org/10.4103/0975-7406.100283

[2] Bumgarner, R. (2013). Overview of DNA Microarrays: Types, Applications, and Their Future. *Current Protocols in Molecular Biology*. https://doi.org/10.1002/0471142727.mb2201s101

[3] Babu, M. Madan. Chapter 11: An Introduction to Microarray Data Analysis https://www.mrc-lmb.cam.ac.uk/genomes/madanm/microarray/chapter-final.pdf

[4] Zhang, A. (2006). Vizualization of microarray data. In *Advanced Analysis of Gene Expression Microarray Data*. World Scientific.

[5] Stekel, D. (2003). Microarray Bioinformatics. https://doi.org/10.1017/cbo9780511615535

[6] Ritchie, M. E., Phipson, B., Wu, D., Hu, Y., Law, C. W., Shi, W., & Smyth, G. K. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research*, *43*(7). https://doi.org/10.1093/nar/gkv007

[7] Li, W. (2012). VOLCANO PLOTS IN ANALYZING DIFFERENTIAL EXPRESSIONS WITH mRNA MICROARRAYS. Journal of Bioinformatics and Computational Biology, 10(06), 1231003. https://doi.org/10.1142/s0219720012310038

Anexos

Anexo A: funções utilizadas da biblioteca genArise[1]

`bg.correct` *Background Correction*

Description

This function use the background data to eliminate unwanted effects in signal. The background correction establish the new Cy3 signal as Cy3 - BgCy3 and the new Cy5 as Cy5 - BgCy5.

Usage

```
bg.correct(mySpot)
```

Arguments

`mySpot`	Spot object for one microarray.

Value

Spot object with the background correction done.

Examples

```
data(Simon)
## background correction and save it in c.spot
c.spot <- bg.correct(Simon)
```

`cys.plot` *Data Visualization: log2(Cy3) vs log2(Cy5)*

Description

This function shows the plot of the values from the log Cy3 against log Cy5 intensities that belongs to an object of the Spot class.

Usage

```
cys.plot(mySpot, col = "green")
```

Arguments

`mySpot`	An Spot object
`col`	Color in which the points of the plot will be shown. This argument must be quoted and the possible values it can take are the same from the color funcion in the R base.

[1] retirado da documentação oficial disponível em http://www.bioconductor.org

Examples

```
data(Simon)
cys.plot(Simon)
```

`global.norm` *Global Normalization of Spot*

Description

This function normalize R and I values and fit the value of Cy5 from his argument. In this function the normalize algorithm will be applied to all observations to get the lowess factor and then fit Cy5 with this factor. The observations. The observations with values R = 0 are deleted because they have no change in their expression levels.

Usage

```
global.norm(mySpot)
```

Arguments

`mySpot`	A spot object

Value

A new spot object but normalized, It means with a different Cy5 that is the result of the fit with the lowess factor.

Examples

```
data(Simon)
# Background Correction
c.spot <- bg.correct(Simon)
#Normalized data
n.spot <- global.norm(c.spot)
```

`ma.plot` *Data Visualization: M vs. A plot*

Description

This function allows to plot **M -vs- A** in spot.

Usage

```
ma.plot(mySpot, col = "green")
```

Arguments

`mySpot`	Spot for one microarray.
`col`	color of points in graphic

Examples

```
data(Simon)
## plot the signals for spot.
ma.plot(Simon)
```

`read.spot` *Read Spot from File*

Description

Read all file, but only extract the interested columns and create a Spot object.

Usage

```
read.spot(file.name, cy3, cy5, bg.cy3, bg.cy5, ids, symdesc, header =
FALSE, sep = "\t", is.ifc = FALSE, envir)
```

Arguments

`file.name`	a connection or a character string giving the name of the file to read where each column represent the spot components.
`cy3`	column that represent Cy3.
`cy5`	column that represent Cy5.
`bg.cy3`	column that represent BgCy3.
`bg.cy5`	column that represent BgCy5.
`ids`	column that represent Id.
`symdesc`	(optional) identifier besides Id column.
`header`	the logical value of the header input file
`sep`	the separator in the inputfile
`is.ifc`	If is.ifc = TRUE this experiment was done in the Unit of Microarray from Cellular Phisiology Institute.
`envir`	Environment where are the genArise variables. You don't need to specify this argument.

Description

This function allows to plot **R-values** vs **I-values I-value** from a Spot object

Usage

```
ri.plot(mySpot, col = "green")
```

Arguments

mySpot	Spot Object
col	Color in which the pioints of the plot will be shown. This argment must be quoted and the possible values it can ake ares the same from the colors funcion in the R base package.

Examples

```
data(Simon)
ri.plot(Simon)
```

Description

This function identify differential expressed genes by calculating an intensity-dependent Z-score. This function use a sliding window to calculate the mean and standard deviation within a window surrounding each data point, and define a Z-score where Z measures the number of standard deviations a data point is from the mean.

Usage

```
Zscore(spot.object,type,window.size)
```

Arguments

spot.object	A spot object
type	Type of analysis: "ri" is for a R-I analysis and "ma" is for M-A analysis
window.size	Size of the sliding window

Value

A dataSet object with attributes Cy3, Cy5, Id, Z-score.

Examples

```
data(Simon)
# Background Correction
c.spot <- bg.correct(Simon)
#Normalized data
n.spot <- grid.norm(c.spot,23,24)
#Filter spot
f.spot <- filter.spot(n.spot)
#Replicate filtering
u.spot <- spotUnique(f.spot)
#Zscore analysis
s.spot <- Zscore(u.spot)
```

Description

This function allows to plot **R-values** vs **I-values** or **M-values** vs **A-values** for identifying differential expression.

Usage

```
Zscore.plot(dataSet.spot, Zscore.min, Zscore.max, all, col)
```

Arguments

`dataSet.spot`	Spot Object
`Zscore.min`	The lower value in a range, if Zscore.min = NULL then the file will contain all values bellow Zscore.max
`Zscore.max`	The greater value in a range, if Zscore.max = NULL then file will be contain all values above Zscore.min. Both values, Zscore.min and Zscore.max can not be NULL
`all`	Plot all the observations in four sets: $Z < 1$, $1 < Z < 1.5$, $1.5 < Z < 2$, $Z > 2$
`col`	Color in which the pioints of the plot will be shown where only the points from center are plot. This argument must be quoted and the possible values it can take are the same from the colors function in the R base package.

Examples

```
data(WT.dataset)
Zscore.plot(WT.dataset, Zscore.min = 1, Zscore.max = 2)
```

Anexo B: funções utilizadas da biblioteca limma[2]

---

eBayes *Empirical Bayes Statistics for Differential Expression*

Description

Given a microarray linear model fit, compute moderated t-statistics, moderated F-statistic, and log-odds of differential expression by empirical Bayes moderation of the standard errors towards a common value.

Usage

```
eBayes(fit, proportion = 0.01, stdev.coef.lim = c(0.1,4),
       trend = FALSE, robust = FALSE, winsor.tail.p = c(0.05,0.1))
treat(fit, lfc = log2(1.2), trend = FALSE, robust = FALSE, winsor.tail.p = c(0.05,0.1))
```

Arguments

`fit`	an `MArrayLM` fitted model object produced by `lmFit` or `contrasts.fit`. For `ebayes` only, `fit` can alternatively be an unclassed list produced by `lm.series`, `gls.series` or `mrlm` containing components `coefficients`, `stdev.unscaled`, `sigma` and `df.residual`.
`proportion`	numeric value between 0 and 1, assumed proportion of genes which are differentially expressed
`stdev.coef.lim`	numeric vector of length 2, assumed lower and upper limits for the standard deviation of log2-fold-changes for differentially expressed genes
`trend`	logical, should an intensity-trend be allowed for the prior variance? Default is that the prior variance is constant.
`robust`	logical, should the estimation of `df.prior` and `var.prior` be robustified against outlier sample variances?
`winsor.tail.p`	numeric vector of length 1 or 2, giving left and right tail proportions of `x` to Winsorize. Used only when `robust=TRUE`.
`lfc`	the minimum log2-fold-change that is considered scientifically meaningful

[2] retirado da documentação oficial disponível em https://www.bioconductor.org/

Description

Fit linear model for each gene given a series of arrays

Usage

```
lmFit(object, design=NULL, ndups=1, spacing=1, block=NULL, correlation, weights=NULL,
      method="ls", ...)
```

Arguments

object	A matrix-like data object containing log-ratios or log-expression values for a series of arrays, with rows corresponding to genes and columns to samples. Any type of data object that can be processed by getEAWP is acceptable.
design	the design matrix of the microarray experiment, with rows corresponding to arrays and columns to coefficients to be estimated. Defaults to the unit vector meaning that the arrays are treated as replicates.
ndups	positive integer giving the number of times each distinct probe is printed on each array.
spacing	positive integer giving the spacing between duplicate occurrences of the same probe, spacing=1 for consecutive rows.
block	vector or factor specifying a blocking variable on the arrays. Has length equal to the number of arrays. Must be NULL if ndups>2.
correlation	the inter-duplicate or inter-technical replicate correlation
weights	non-negative precision weights. Can be a numeric matrix of individual weights of same size as the object expression matrix, or a numeric vector of array weights with length equal to ncol of the expression matrix, or a numeric vector of gene weights with length equal to nrow of the expression matrix.
method	fitting method; "ls" for least squares or "robust" for robust regression
...	other optional arguments to be passed to lm.series, gls.series or mrlm

Description

Extract a table of the top-ranked genes from a linear model fit.

Usage

```
topTable(fit, coef=NULL, number=10, genelist=fit$genes, adjust.method="BH",
         sort.by="B", resort.by=NULL, p.value=1, lfc=0, confint=FALSE)
toptable(fit, coef=1, number=10, genelist=NULL, A=NULL, eb=NULL, adjust.method="BH",
         sort.by="B", resort.by=NULL, p.value=1, lfc=0, confint=FALSE, ...)
topTableF(fit, number=10, genelist=fit$genes, adjust.method="BH",
         sort.by="F", p.value=1, lfc=0)
topTreat(fit, coef=1, sort.by="p", resort.by=NULL, ...)
```

Arguments

fit	list containing a linear model fit produced by lmFit, lm.series, gls.series or mrlm. For topTable, fit should be an object of class MArrayLM as produced by lmFit and eBayes.
coef	column number or column name specifying which coefficient or contrast of the linear model is of interest. For topTable, can also be a vector of column subscripts, in which case the gene ranking is by F-statistic for that set of contrasts.
number	maximum number of genes to list
genelist	data frame or character vector containing gene information. For topTable only, this defaults to fit$genes.
A	matrix of A-values or vector of average A-values. For topTable only, this defaults to fit$Amean.
eb	output list from ebayes(fit). If NULL, this will be automatically generated.
adjust.method	method used to adjust the p-values for multiple testing. Options, in increasing conservatism, include "none", "BH", "BY" and "holm". See p.adjust for the complete list of options. A NULL value will result in the default adjustment method, which is "BH".
sort.by	character string specifying statistic to rank genes by. Possible values for topTable and toptable are "logFC", "AveExpr", "t", "P", "p", "B" or "none". (Permitted synonyms are "M" for "logFC", "A" or "Amean" for "AveExpr", "T" for "t" and "p" for "P".) Possibilities for topTableF are "F" or "none". Possibilities for topTreat are as for topTable except for "B".
resort.by	character string specifying statistic to sort the selected genes by in the output data.frame. Possibilities are the same as for sort.by.
p.value	cutoff value for adjusted p-values. Only genes with lower p-values are listed.
lfc	minimum absolute log2-fold-change required. topTable and topTableF include only genes with (at least one) absolute log-fold-changes greater than lfc. topTreat does not remove genes but ranks genes by evidence that their log-fold-change exceeds lfc.
confint	logical, should confidence 95% intervals be output for logFC? Alternatively, can take a numeric value between zero and one specifying the confidence level required.
...	For toptable, other arguments are passed to ebayes (if eb=NULL). For topTreat, other arguments are passed to topTable.

volcanoplot *Volcano Plot*

Description

Creates a volcano plot for a specified coefficient of a linear model.

Usage

```
volcanoplot(fit, coef = 1, style = "p-value", highlight = 0, names = fit$genes$ID, hl.col="blue",
            xlab = "Log2 Fold Change", ylab = NULL, pch=16, cex=0.35, ...)
```

Arguments

fit	an MArrayLM fitted linear model object.
coef	index indicating which coefficient of the linear model is to be plotted.
style	character string indicating which significance statistic to plot on the y-axis. Possibilities are "p-value" or "B-statistic".
highlight	number of top genes to be highlighted by name.
names	character vector of length nrow(fit) giving gene names. Only used if highlight > 0.
hl.col	color for the gene names. Only used if highlight > 0.
xlab	character string giving label for x-axis
ylab	character string giving label for y-axis
pch	vector or list of plotting characters.
cex	numeric vector of plot symbol expansions.
...	any other arguments are passed to plot

Details

A volcano plot displays log fold changes on the x-axis versus a measure of statistical significance on the y-axis. Here the significance measure can be -log(p-value) or the B-statistics, which give the posterior log-odds of differential expression.

The plot is optionally annotated with the names of the most significant genes.

Value

No value is returned but a plot is created on the current graphics device.

Author(s)

Gordon Smyth

See Also

An overview of presentation plots following the fitting of a linear model in LIMMA is given in 06.LinearModels.

Examples

```
#  See lmFit examples
```

Anexo C: Tabela completa com os vinte genes diferencialmente expressos

ID	logFC	AveExpr	t	P.Value	adj.P.Val	B
Id224	1.2445866	12.14529	4.683984	0.0039054	0.8988501	-2.738006
Id2940	-1.5074732	13.16637	-4.568239	0.0043856	0.8988501	-2.740771
Id2370	1.2782589	10.87905	4.387529	0.0052761	0.8988501	-2.745396
Id2319	1.7799392	12.19074	4.303891	0.0057566	0.8988501	-2.747672
Id478	-1.2873661	11.72450	-4.268985	0.0059716	0.8988501	-2.748649
Id2915	1.1449601	10.84709	4.118087	0.0070120	0.8988501	-2.753062
Id1565	-1.1062705	12.02856	-4.082405	0.0072870	0.8988501	-2.754152
Id1493	-1.0426521	12.04587	-3.969118	0.0082442	0.8988501	-2.757741
Id2013	-1.0625074	11.40783	-3.793434	0.0100220	0.8988501	-2.763705
Id104	-1.0246604	12.22081	-3.763541	0.0103655	0.8988501	-2.764771
Id2073	1.8655420	10.61306	3.741666	0.0106253	0.8988501	-2.765561
Id2615	1.4718021	11.29503	3.627071	0.0121107	0.8988501	-2.769838
Id207	-0.9398375	11.42615	-3.559286	0.0130980	0.8988501	-2.772482
Id162	1.1015361	12.12682	3.448038	0.0149188	0.8988501	-2.777013
Id1286	1.5639441	11.95920	3.426268	0.0153072	0.8988501	-2.777928
Id1577	-0.8729727	11.96545	-3.409810	0.0156082	0.8988501	-2.778627
Id2595	-2.0182427	12.16484	-3.388519	0.0160075	0.8988501	-2.779538
Id83	-0.8483215	12.24841	-3.344154	0.0168763	0.8988501	-2.781469
Id2568	-0.8503506	12.05297	-3.295349	0.0178930	0.8988501	-2.783641