Análisis de datos de microarrays con R y Bioconductor
Selección de genes diferencialmente expresados

Alex Sánchez

May 25, 2020

Contents

1 Introducción y objetivos

Este documento presenta un ejemplo de análisis de microarrays realizado con R y Bioconductor. Su objetivo no es mostrar como realizar el mejor análisis de microarrays posible sino realizar una presentación didáctica que sirva de introducción completa al tema.

1.1 Estructura del documento

Típicamente un trabajo científico o técnico se organiza en varias partes: *Introducción*, *Material y Métodos*, *Resultados* y *Discusión*. Dado que el objetivo de este estudio es demostrar la aplicación de los métodos de análisis explicados en la asignatura esta estructura se relajará. Después de la introducción se realizará una breve descripción del proceso general y los métodos utilizados en cada paso. A continuación se presentarán los resultados obtenidos en cada fase del análisis. Finalmente se presentará una breve discusión sobre las posibles limitaciones encontradas.

1.2 Objetivos

El objetivo de este trabajo es realizar un análisis de datos de microarrays para encontrar genes diferencialmente expresados entre varios tipos de tumores de cancer de mama clasificados en tres grupos: apocrinos (APO), basales (BAS) y luminales (LUMI).

Esta clasificación se basa en la resistencia de los tumores a los receptores de estrógenos y de andrógenos.

- Los tumores clasificados como "APO" son negativos para los receptores de estrógenos (ER-) y positivos para los receptores de andrógenos (AR+).
- los clasificados como "LUMI" son (ER+) y (AR+) y
- los clasificados como "BAS" son (ER-) y (AR-).

Es conveniente destacar que, de hecho, el objetivo de la práctica difiere del objetivo del artículo en que se basa. Mientras que el artículo se busca *descubrir* los tres grupos con los que trabajamos en este ejercicio se pretende caracterizarlos a través de los gene que se expresan de forma distinta entre ellos. No deja de

ser un problema del tipo "el huevo y la gallina", es decir: *¿Puesto que se trata de grupos distintos encontramos genes que se expresan diferencialmente entre ellos o dado que hay genes que se expresan diferencialmente podemos deducir que son grupos distintos?*

2 Materiales y métodos

2.1 Etapas del analisis

El análisis se ha realizado siguiendo las pautas descritas en los capítulos 6 a 11 de unos materiales para un curso de análisis de microarrays `http://ueb.vhir.org/cursos/Microarrays` y resumidos en la figura 1.

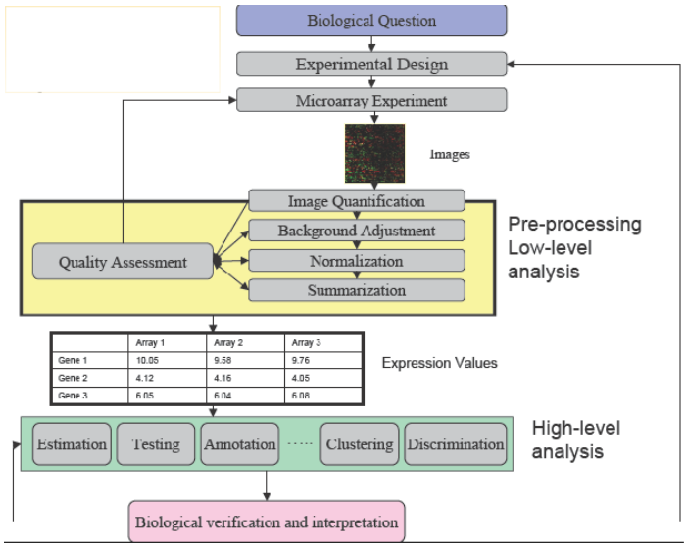


Figure 1: The Microarray Analysis Process

2.2 Métodos de análisis

Antes de hibridar los microarrays[1] se comprobó la calidad del cRNA de cada muestra. Sólo las que demostraron una calidad suficiente –presumiblemente un valor de "Bio-analyser"[2] igual o superior a 7– se sometieron a análisis posterior..

Los valores "crudos" de expresión obtenidos directamente de los archivos CEL se preprocesa utilizando el método de RMA ([7]), un proceso de tres pasos que integra la corrección de fondo, la normalización y el resumen de los valores del grupo de sondas en un único valor de expresión "absoluta". Dichos valores normalizados fueron la base para todos los análisis.

[1] Esta primera parte es ficticia en tanto que no hemos tenido acceso al proceso de elaboración de los microarrays, por lo que he escrito lo que suele ponerse en esta parte

[2] Sistema para el análisi de la calidad del RNA

Antes de la selección de genes los valores normalizados se sometieron a un filtraje no específico para eliminar los genes de baja variabilidad (los genes cuyo rango intercuartil entre todas las muestras no superó un umbral mínimo).

La selección de genes diferencialmente expresados entre condiciones experimentales se basó en el método desarrollado por Smyth y otros [11].

Este método extiende el análisis de la varianza clásico utilizando métodos Bayesianos empíricos para combinar la información de cada gen individual con la de todos los genes restantes para obtener mejores estimaciones de error. Esto es de gran utilidad en análisis de microarrays, un contexto en el que el tamaño de las muestras es a menudo pequeño lo que puede dar lugar a estimaciones de los errores erráticos y, en consecuencia, p–valores que no son de fiar.

Los genes más relevantes de cada comparación se resaltaron utilizando "volcano-plots", que organizan los genes a lo largo de dos dimensiones que podemos considerar de importancia biológica y estadística. El eje horizontal representa el cambio medio de expresión entre los dos grupos (en una escala logarítmica, por lo que la regulación hacia arriba y abajo aparecen simétrica), y el segundo (vertical) representa el "menos logaritmo del p-valor" por lo que los genes cuyo p-valor asociado sea inferior aparecen más arriba. El primer eje indica el impacto biológico del cambio, y el segundo indica la evidencia estadística, o la fiabilidad de dicho cambio.

Con el fin de hacer frente a la problemática derivada del hecho de que muchas pruebas (una por cada gen) se realizan simultáneamente, se realizo un ajuste de p–valores para obtener control sobre la tasa de falsos positivos usando el método de Benjamini y Hochberg ([1]).

Los genes seleccionados como diferencialmente expresados se agruparon para buscar patrones comunes de expresión entre condiciones experimentales. Para ello se utilizaron mapas de colores o "Heatmaps" que realizan una agrupación jerárquica de los genes y/o las muestras y la representan mediante una gama de colores apropiada, de forma que valores altos o bajos se corresponden a colores distintos de la gama escogida.

Las listas de genes diferencialmente expresados se anotaron en diversas bases de datos (Entrez, Unigene, Gene Ontology, KEGG, ...) utilizando los paquetes de anotación para microarrays de affymetrix disponibles en el proyecto Bioconductor (`http://bioconductor.org`).

Para contribuir a la interpretación biológica de los resultados se realizó un análisis de enriquecimiento [5, 3] que busca establecer si las categorias funcionales de los genes seleccionados aparecen entre estos genes con mayor (o menor) frecuencia que entre todos los del genoma. De ser así se indica que la lista de genes se encuentra "enriquecida" en estas funcionalidades, o lo que es lo mismo que los procesos afectados por las diferencias son éstos.

2.2.1 Herramientas y procedimientos bioinformáticos de análisis

Los análisis estadísticos se realizaron utilizando el lenguaje estadístico *R*y las librerías desarrolladas para el análisis de microarray en el proyecto de Bioconductor (`www.bioconductor.org`). Para más detalle sobre los métodos descritos en esta sección puede consultarse [6].

El código siguiente se utilizó para instalar los paquetes de Bioconductor necesarios para el análisis.

```
## [1] "Package Biobase already installed"
## [1] "Package affy already installed"
## [1] "Package arrayQualityMetrics already installed"
## [1] "Package genefilter already installed"
## [1] "Package multtest already installed"
## [1] "Package limma already installed"
## [1] "Package hgu133a.db already installed"
## [1] "Package annotate already installed"

## Warning:  Package 'KEGG.db' is deprecated and will be removed from
Bioconductor
##  version 3.12

## [1] "Package annaffy already installed"
## [1] "Package hwriter already installed"
## [1] "Package gplots already installed"
## [1] "Package GOstats already installed"
## [1] "Package GSA already installed"
```

3 Obtención y lectura de los datos

3.1 Los datos para el análisis

Los datos en que se basa el estudio se han obtenido a partir de tumores de mama ("advanced/inflammatory breast tumours") y fueron tomados antes del tratamiento de pacientes enrolados en un estudio clínico (EORTC 10994).

Los microarrays se prepararon a partir de RNA total extraído de secciones de 25 mm de biopsias y amplificados mediante el procedimiento "Eberwine T7 procedure" siguiendo el protocolo indicado por Affymetrix para pequeñas muestras.

Los datos de los microarrays se encuentran en Gene Expression Omnibus (GEO) con el número de serie GSE1561. Puede accederse a ellos en el siguiente enlace `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE1561`. El artículo de Farmer ([4]) *et alt.*(`http://www.ncbi.nlm.nih.gov/pubmed/15897907`) describe el estudio.

3.1.1 Localización de los datos

Como es habitual en este curso supondremos que trabajamos en un directorio escogido por nosotros y cuya localización se asigna a la variable `workingDir`.

Asumiremos que los datos se encuentran en un subdirectorio del anterior, denominado "data" que se almacenará en la variable `dataDir` y que los resultados se almacenarán en un directorio "results" cuyo nombre completo se almacenará en la variable `resultsDir`.

```
workingDir <-getwd()
dataDir <-file.path(workingDir, "data")
resultsDir <- file.path(workingDir,"results")
celfilesDir <- file.path(workingDir,"celfiles")
```

```
setwd(workingDir)
```

3.1.2 Selección de muestras para el análisis

Este análisis se ha basado en un subconjunto de muestras del estudio original. Las muestras se han obtenido de la lista original mediante un pequeño script de *R*.

De la dirección `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE1561` es posible descargar un archivo comprimido con los 49 archivos ".cel". Con el fin de simplificar el análisis se decidió seleccionar aleatoriamente 5 arrays de cada grupo utilizando a partir de la información contenida en el archivo: "Asignacion_de_Muestras_A_Grupos.csv" elaborado a partir de la información sobre el diseño experimental contenida en: `http://www.ncbi.nlm.nih.gov/geo/gds/profileGraph.cgi?gds=1329`

```
# muestras <- read.csv2(file.path(dataDir, "Asignacion_de_muestras_a_grupos.csv"),  head=T
# misMuestras <- as.character (muestras£Sample)
# paraAnalisis <- c(sample(misMuestras[1:6], 5),
#                   sample(misMuestras[7:20], 5),
#                   sample(misMuestras[23:48], 5))
# alAnalisis <-muestras[muestras£Sample %in% paraAnalisis,]
# write.table(alAnalisis, file=file.path(dataDir, "targets.txt"),
#             sep="\t", row.names=FALSE, quote=FALSE)
```

3.1.3 Lectura de los datos

La lectura de datos se lleva a cabo utilizando las clases y métodos definidas en los paquetes `Biobase` y `affy` de Bioconductor.

Una forma cómoda de leer los datos y, al mismo tiempo, asignar a cada muestra los valores de las covariables (por ejemplo el grupo para el analisis) consiste en crear un pequeño archivo de texto, que suele denominarse `targets.txt` y que contiene la identificacion de cada archivo con la asignación de cada muestra a cada condición experimental. El archivo targets para este caso tiene el aspecto que se muestra en la tabla siguiente.

```
## Loading required package:  xtable
```

	Sample	Ids	SampleIDs	Group
GSM26878.CEL	GSM26878	PF14 EnPnT2N1G2	PF14	A
GSM26883.CEL	GSM26883	PF19 EnPuT4N0Gu	PF19	A
GSM26887.CEL	GSM26887	PF23 EnPnT2N0G2	PF23	A
GSM26903.CEL	GSM26903	PF39 EnPuT4N0Gu	PF39	A
GSM26910.CEL	GSM26910	PF46 EnPnT4N1G3	PF46	A
GSM26888.CEL	GSM26888	PF24 EnPnTiN0G3	PF24	B
GSM26889.CEL	GSM26889	PF25 EnPnT3N2G2	PF25	B
GSM26892.CEL	GSM26892	PF28 EnPnT2N1G3	PF28	B
GSM26898.CEL	GSM26898	PF34 EnPnT3N1G3	PF34	B

GSM26906.CEL	GSM26906	PF42	EnPnT2N2G3	PF42	B
GSM26879.CEL	GSM26879	PF15	EpPnTiN1G3	PF15	L
GSM26896.CEL	GSM26896	PF32	EpPnT3N1G2	PF32	L
GSM26897.CEL	GSM26897	PF33	EpPnTiN0G2	PF33	L
GSM26907.CEL	GSM26907	PF43	EpPpT2N1G2	PF43	L
GSM26911.CEL	GSM26911	PF47	EpPpT3N1G3	PF47	L

Table 1: Archivo targets.txt con la asignación a cada muestra de su condición experimental

El contenido del archivo `targets` se utiliza en la lectura de los datos y la creación del objeto `rawData` de la clase `affybatch` que contendrá las intensidade "crudas" de cada archivo .CEL

```
require(affy)
sampleInfo <- read.AnnotatedDataFrame(file.path(dataDir,"targets.txt"),
    header = TRUE, row.names = 1, sep="\t")
fileNames <- rownames(pData(sampleInfo))
rawData <- read.affybatch(filenames=file.path(celfilesDir,fileNames),
                          phenoData=sampleInfo)
show(rawData)
```

```
## Warning:  replacing previous import 'AnnotationDbi::tail' by 'utils::tail'
when loading 'hgu133acdf'
## Warning:  replacing previous import 'AnnotationDbi::head' by 'utils::head'
when loading 'hgu133acdf'
##
```

```
## AffyBatch object
## size of arrays=712x712 features (29 kb)
## cdf=HG-U133A (22283 affyids)
## number of samples=15
## number of genes=22283
## annotation=hgu133a
## notes=
```

Este objeto es la base para todos los análisis que se realizarán.

4 Preprocesado: Exploración, Control de Calidad y Normalización

Los datos procedentes de la lectura de los microarrays se denominan datos "crudos" y deben ser pre-procesados de diversas formas antes de analizarlos.

"Preprocesado" es un término genérico que engloba varios procesos

- Exploración y control de calidad de los datos.
- Normalización y resumen (llamdo "sumarización") de los valores de las sondas de cada grupo de sondas.

- Filtrado no específico para eliminar el efecto de genes que no se expresan o bien no se expresan de forma distinta entre los grupos.

A su vez la exploración y el control de calidad contempla:

1. Exploraciones estadísticas estándar.
2. Técnicas de control de calidad desarrolladas específicamente para datos de microarrays.

4.1 Exploración y visualización

La exploración de los datos suele basarse en técnicas univariantes como los histogramas o los diagramas de caja o en técnicas multivariantes como los análisis de conglomerados ("clusters"), de distancias o de análisis de componentes principales.

4.1.1 Exploración estadística de los datos

Un gráfico de densidad –mal llamado en este contexto, histograma– permite hacerse una idea de si las distribuciones de los distintos arrays son similares en forma y posición.

Un diagrama de cajas muestra la misma información facilitando más la comparación entre distribuciones.

Una representación multivariante como la ofrecida por los gráficos de las componentes principales suele resultar muy informativa de posibles problemas como efectos batch o presencia de valores "raros". El gráfico de componentes principales muestra los datos en dimensión reducida de forma que cada componente explica una dimensión de mayor variabilidad e independiente de la siguiente. Representando el porcentaje de variabilidad explicada para cada componente tenemos una medida de la importancia de los grupos que se puedan visualizar. Si la suma de los porecentajes es alta, por ejemplo superior al 50% las conclusiones que de ellos se deriven seran más fiables que con valores bajos, por ejemplo inferiores al 30% de varianza explicada.

```
plotPCA <- function ( X, labels=NULL, colors=NULL, dataDesc="", scale=FALSE)
{
  pcX<-prcomp(t(X), scale=scale) # o prcomp(t(X))
  loads<- round(pcX$sdev^2/sum(pcX$sdev^2)*100,1)
  xlab<-c(paste("PC1",loads[1],"%"))
  ylab<-c(paste("PC2",loads[2],"%"))
  if (is.null(colors)) colors=1
  plot(pcX$x[,1:2],xlab=xlab,ylab=ylab, col=colors,
       xlim=c(min(pcX$x[,1])-10, max(pcX$x[,1])+10))
  text(pcX$x[,1],pcX$x[,2], labels, pos=3, cex=0.8)
  title(paste("Plot of first 2 PCs for expressions in", dataDesc, sep=" "), cex=0.8)
}
```

Un enfoque alternativo al del PCA aunque relacionado es realizar un análisis basado en distancias. Podemos hacerlo calculando y visualizando la matriz de distancias mediante un mapa de colores o mediante un agrupamiento jerárquico seguido de un dendrograma.

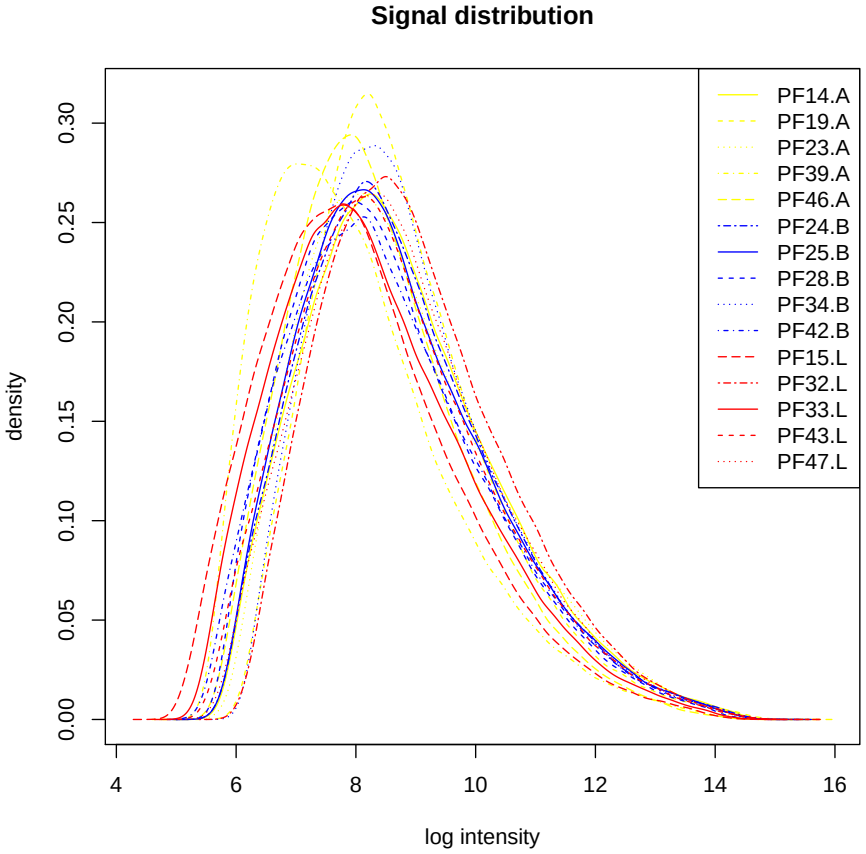


Figure 2: Distribución de las expresiones de cada array. Aunque los gráficos muestran una cierta heterogeneidad entre muestras el aspecto de todas ellas es compatible con la "normalidad" es decir no sugieren que pueda haber algún problema en los datos.

```
boxplot(rawData, cex.axis=0.6, col=colores, las=2, names=sampleNames,
        main="Signal distribution for selected chips")
```

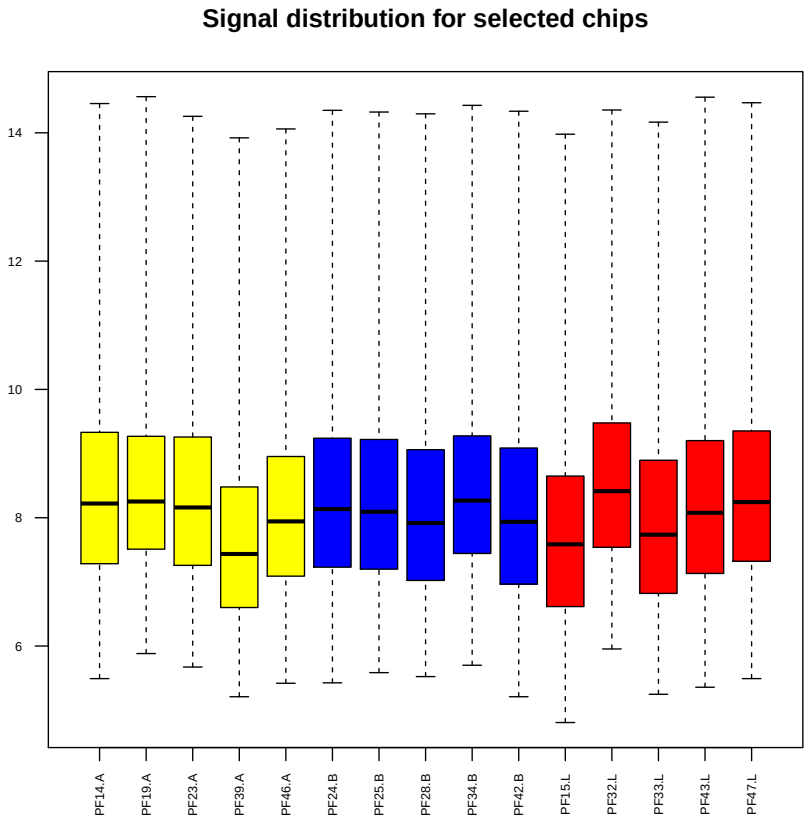


Figure 3: El diagrama de cajas muestra, de forma similar al histograma, cómo las distribuciones de los datos son relativamente similares. Hay heterogeneidad pero no se muestra ninguna diferencia sistemática, lo que sugiere que es conveniente normalizar peró no parece que hayan arrays problemáticos

```
plotPCA(exprs(rawData), labels=sampleNames, dataDesc="selected samples")
```

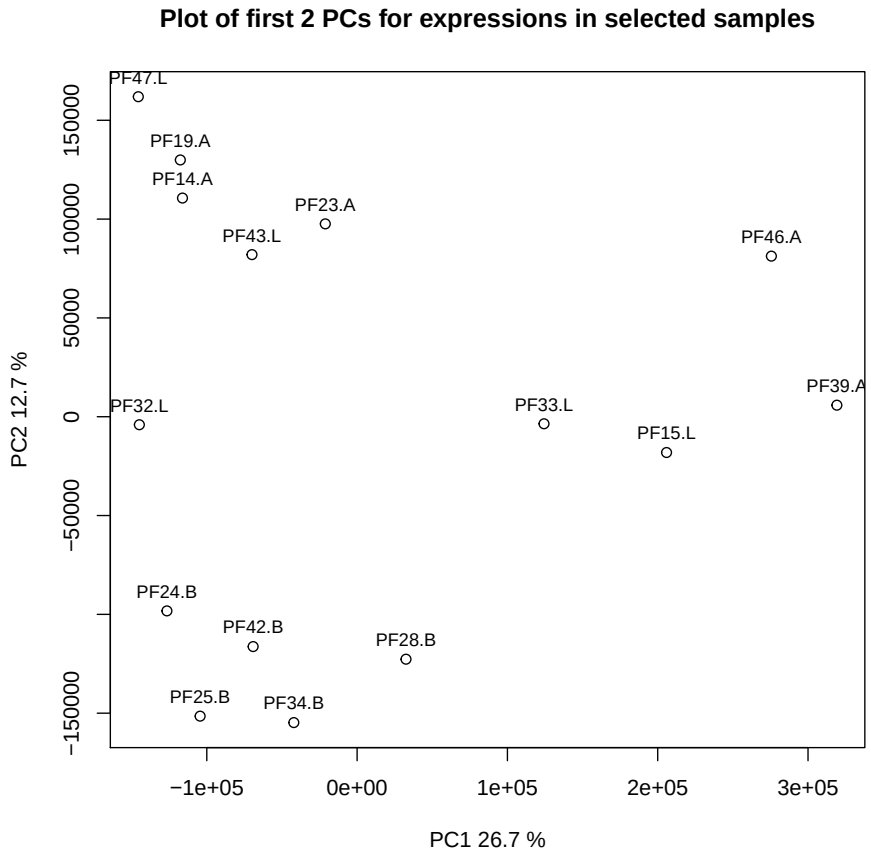


Figure 4: Representación de las dos primeras componentes de un análisi de components principales. Los grupos aparecen algo separados pero el porcentaje de variabilidad explicado por las dos primeras componentes sugiere que puede no ser muy explicativas

```
manDist <-  dist(t(exprs(rawData)))
heatmap (as.matrix(manDist),  col=heat.colors(16))
```

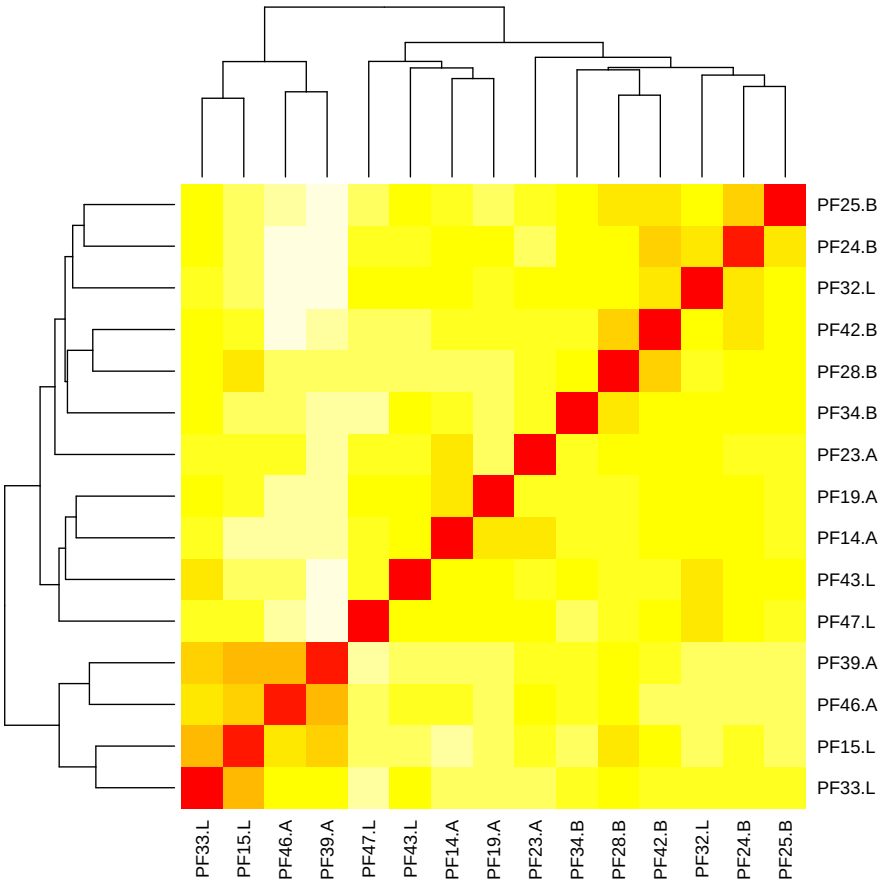


Figure 5: Mapa de colores de las distancias (euclídeas) entre los distintos arrays

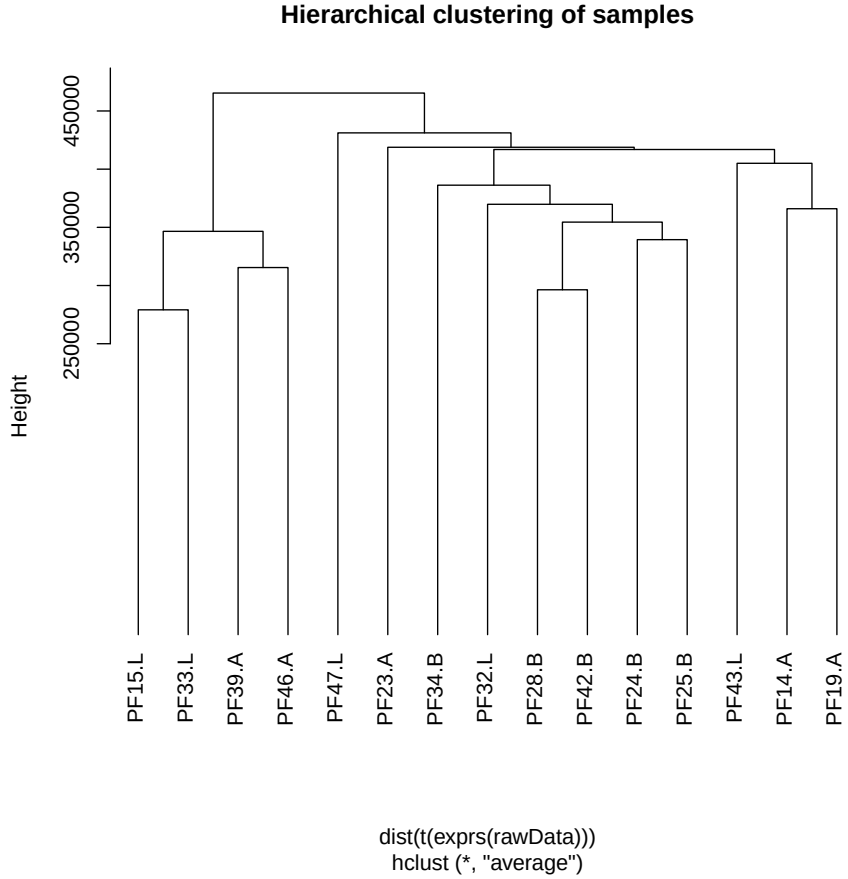


Figure 6: Dendrograma resultante de un agrupamiento jerárquico entre las muestras. Como en otras representaciones (Heatmaps, PCA) se observa una cierta agrupación de muestras de tipos similares aunque también una cierta heterogeneidad, atribuïble probablemente a la diferencia entre pacientes

Un cluster jerárquico seguido de un dendrograma nos puede ayudar a hacernos una idea de si las muestras se agrupan por condiciones experimentales.

Si lo hacen es bueno, pero si no, no es necesariamente indicador de problemas, puesto que es un gráfico basado en todo los datos.

4.1.2 Control de calidad (2): métodos específicos para microarrays

El paquete `arrayQualitMetrics` encapsula todos los análisis anteriores, y alguno más, facilitando su ejecución e incluso su interpretación. La instrucción `arrayQualityMetrics` lleva a cabo todos los análisi de golpe y genera un informe de resultados con ayudas a la interpretación y a la detección de arrays problemáticos.

```
stopifnot(require(arrayQualityMetrics))
arrayQualityMetrics(rawData,
                    intgroup = "Group",
                    outdir = file.path(resultsDir, "arrayQuality"),
                    force=TRUE)

## The report will be written into directory 'C:/Users/Alexandre/Dropbox
(VHIR)/SotaCV/Ejemplo_de_MDA_con_Bioconductor/results/arrayQuality'.
```

4.1.3 Archivos con los resultados del control de calidad

Los resultados del control de calidad realizado con `arrayQualityMetrics` se encuentran accesibles a traves del archivo `index.html` contenido en el subdirectorio creado al invocarlo -en este caso denominado `arrayQuality`.

4.2 Normalizacion y Filtraje

Una vez realizado el control de calidad se procede a normalizar los datos y sumarizarlos.

La normalización puede hacerse por distintos métodos (MAS5, VSN, RMA, GCRMA, ...) pero en este caso se utilizará el método RMA que es sin duda el más utilizado entre arrays de affymetrix.

El procesado mediante RMA implica un proceso en tres etapas:

- Corrección de fondo (el RMA hace precisamente esto).
- Normalización para hacer los valores de los arrays comparables.
- Summarización de las diversas sondas asociadas a cada grupo de sondas para dar un único valor.

```
stopifnot(require(affy))
eset_rma <- rma(rawData)

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

Un boxplot de los valores normalizados muestra que los valores han quedado claramente en una escala común en donde se pueden comparar.

4.2.1 Filtraje

La selección de genes diferencialmente expresados se ve afectada por el número de genes sobre la que la hacemos. Cuántos más son, mayor será el ajuste necesario de p-valores (como se verá a continuación) lo que conllevará que podamos acabar descartando erróneamente más genes.

Por este motivo interesa reducir el total de genes que examinamos, es decir si podemos descartar los genes que, razonablemente no van a estar diferencialmente expresados podremos reducir el total sin apenas riesgo de descartar genes

```
boxplot(eset_rma,main="RMA", names=sampleNames, cex.axis=0.7, col=colores,las=2)
```

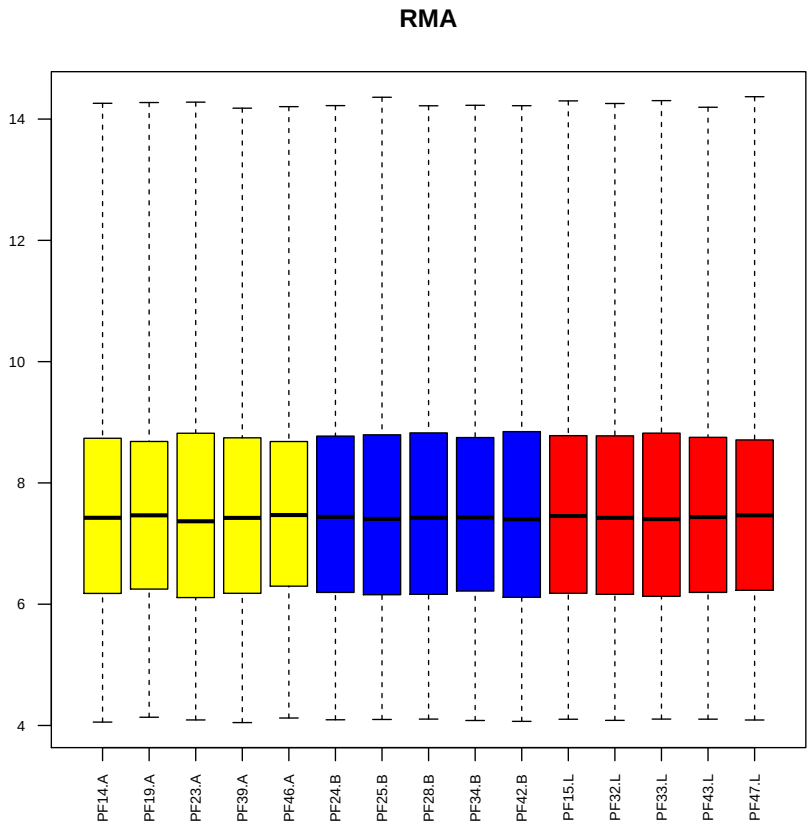
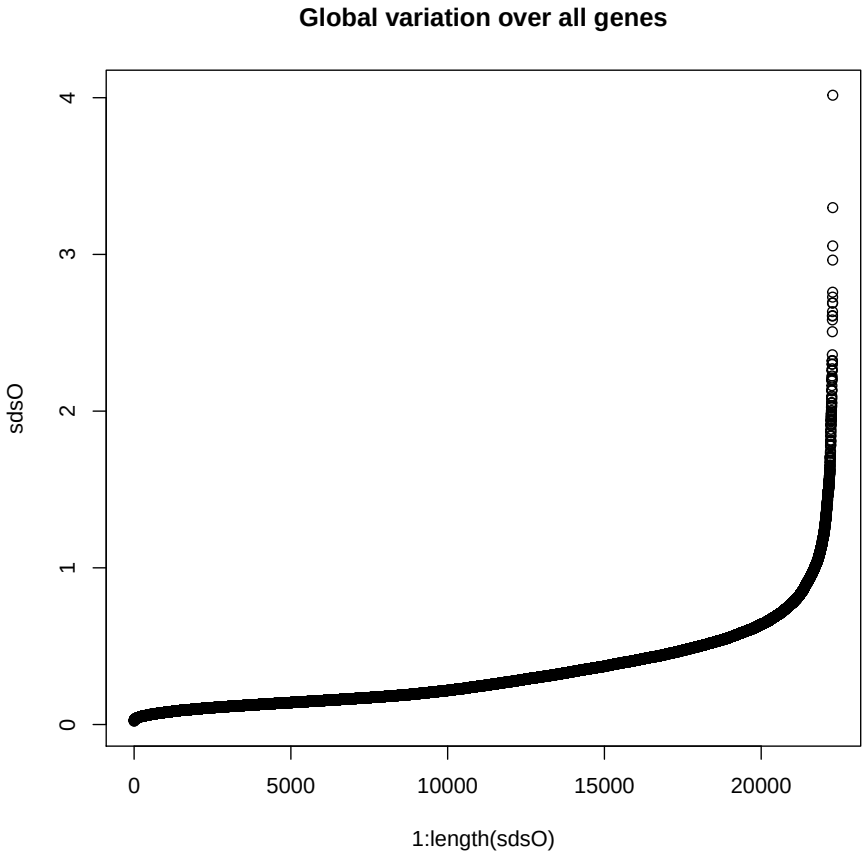


Figure 7: El diagrama de cajas después de normalizar muestra como dicho proceso ha puesto los valores en una escala completamente comparable. *Esto no significa que si hubieran errores los habría arreglado* por lo que dicho proceso debe hacerse tras descartar la ausencia de arrays problemáticos

interesantes. Para ello podemos fijarnos, por ejemplo en la variabilidad global de cada gen, es decir sin tener en cuenta los grupos. La idea será que si un gen está diferencialmente expresado es preciso que haya una cierta diferencia entre los grupos, y por lo tanto la varianza global del gen será mayor que la de aquellos que no presentan expresión diferencial. Si hacemos un gráfico de la variabilidad de todos los genes, deberíamos poder diferenciar los genes que varian más (es decir que presentan expresión diferencial) de los que no la presentan.

```
sds <- apply (exprs(eset_rma), 1, sd)
sds0<- sort(sds)
plot(1:length(sds0), sds0, main="Global variation over all genes")
```



El filtraje *no específico* permite eliminar los genes que varían poco entre condiciones o que deseamos quitar por otras razones como por ejemplo que no disponemos de anotación para ellos. La función `nsFilter` permite eliminar los genes que, o bien varían poco, o bien no se dispone de anotación para ellos.

Si al filtrar deseamos usar las anotaciones, o la falta de ellas, como criterio de filtraje debemos disponer del correspondiente paquete de anotaciones.

```
require(genefilter)
filtered <- nsFilter(eset_rma, require.entrez=TRUE,
         remove.dupEntrez=TRUE, var.func=IQR,
         var.cutoff=0.5, var.filter=TRUE,
         filterByQuantile=TRUE, feature.exclude="^AFFX")
```

La función `nsFilter` devuelve los valores filtrados en un objeto `expressionSet` y un informe de los resultados del filtraje.

```
names(filtered)

## [1] "eset"       "filter.log"

class(filtered$eset)

## [1] "ExpressionSet"
## attr(,"package")
## [1] "Biobase"

print(filtered$filter.log)

## $numDupsRemoved
## [1] 7399
##
## $numLowVar
## [1] 6201
##
## $numRemoved.ENTREZID
## [1] 2472
##
## $feature.exclude
## [1] 10

eset_filtered <-filtered$eset
```

Podemos grabar el objeto `eset_rma` y los datos filtrados para su posterior uso.

4.2.2 Archivos de resultados normalizados

El resultado de la normalización y el filtraje es un objeto `expressionSet` almacenado en un archivo binario `datos.normaliados.Rda`, que será la base para todos los estudios posteriores.

Es muy común también que los investigadores deseen ver los datos normalizados por lo que también se pueden guardar en un archivo de texto para facilitar su posterior inspección. El archivo `Datos.Normalizados.Filtrados.csv2` contiene los datos normalizados y filtrados separados por ";" para no generar problemas de formato con el punto decimal en ingles o español.

```
shortNames<- paste(pData(eset_rma)$Group, pData(eset_rma)$SampleIDs,sep=".")
sum(rownames(pData(eset_filtered))!=colnames(exprs(eset_filtered)))

## [1] 0

colnames(exprs(eset_filtered))<- colnames(exprs(eset_rma))<- shortNames

## Error in (function (od, vd) :  object and replacement value dimnames
differ

write.csv2(exprs(eset_rma), file.path(resultsDir, "Datos.Normalizados.csv2"))
write.csv2(exprs(eset_filtered), file.path(resultsDir, "Datos.Normalizados.Filtrados.csv2"
save(eset_rma, eset_filtered, file=file.path(resultsDir, "datos.normalizados.Rda"))
```

5 Selección de genes diferencialmente expresados

Como en las etapas anteriores la selección de genes diferencialmente expresados (GDE) puede basarse en distintas aproximaciones, desde la t de Student, tests de permutaciones o multitud de variantes.

Empezaremos realizando una comparación entre los grupos A y B que nos permitirá introducir algunos conceptos como el ajuste de p-valores para continuar luego con una aproximación más general, la utilización del modelo lineal que facilita la realización simultanea de más de dos comparaciones.

5.1 Seleccion básica de genes en una comparación entre dos grupos

Empezamos considerando el caso sencillo: Buscamos genes diferencialmente expresados entre dos condiciones utilizando un test clásico como el test$-t$.

5.1.1 Cálculo de estadísticos de test y p–valores

Una aproximación "naïf" a este análisis podría hacerse creando una función que calcule el test t de Student y extraiga el p–valor y el valor de test. Aplicando esta función sobre toda la matiz de expresión y ordenando los valores resultantes de menor a mayor p–valor tendríamos una lista de candidatos a genes diferencialmente expresados.

Una versión algo mejorada de esta aproximación puede obtenerse aplicando la función `rowttests` del paquete genefilter que calcula los estadísticos de test para cada fila de la matriz de expresión y devuelve, para cada gen los valores del estadístico t, el p–valor, los grados de libertad la diferencia de medias.

Escribiendo `? rowttests` se puede ver qué opciones hay disponibles.

La llamada a la función se hace pasándole el objeto y un campo tipo factor que diferencíe los dos grupos.

Crearemos un subconjunto de los datos tomando tan solo los grupos A y B.

```
eset2 <- eset_filtered[,1:10]
pData(eset2)
```

```
##                Sample             Ids SampleIDs Group Apocrine.grade
## GSM26878.CEL GSM26878 PF14 EnPnT2N1G2      PF14     A              3
## GSM26883.CEL GSM26883 PF19 EnPuT4N0Gu      PF19     A              2
## GSM26887.CEL GSM26887 PF23 EnPnT2N0G2      PF23     A              3
## GSM26903.CEL GSM26903 PF39 EnPuT4N0Gu      PF39     A              3
## GSM26910.CEL GSM26910 PF46 EnPnT4N1G3      PF46     A              3
## GSM26888.CEL GSM26888 PF24 EnPnTiN0G3      PF24     B              2
## GSM26889.CEL GSM26889 PF25 EnPnT3N2G2      PF25     B              1
## GSM26892.CEL GSM26892 PF28 EnPnT2N1G3      PF28     B              2
## GSM26898.CEL GSM26898 PF34 EnPnT3N1G3      PF34     B              1
## GSM26906.CEL GSM26906 PF42 EnPnT2N2G3      PF42     B             na
##              AR.repeat.length
## GSM26878.CEL               18
## GSM26883.CEL               18
## GSM26887.CEL               18
## GSM26903.CEL               17
## GSM26910.CEL               19
## GSM26888.CEL               NA
## GSM26889.CEL               21
## GSM26892.CEL               22
## GSM26898.CEL               17
## GSM26906.CEL               16
```

```
dim(exprs(eset2))
```

```
## [1] 6201   10
```

Observemos que esta instrucción ha creado un subconjunto tanto de las covariables como de los datos!!!

```
stopifnot(require(genefilter))
teststat <-rowttests(eset2, "Group")
head(teststat)
```

```
##              statistic         dm     p.value
## 216705_s_at -1.2091620 -0.2994326 0.261120186
## 212607_at   -4.3506604 -1.1042224 0.002443108
## 209028_s_at -1.7677938 -0.4968775 0.115072401
## 202382_s_at -3.4956834 -0.6426245 0.008130368
## 203256_at   -0.3900989 -0.1973068 0.706644376
## 209451_at   -2.2829535 -0.8503465 0.051831823
```

Ordenando los genes de menor a mayor p–valor tenemos una lista de genes *candidatos* a ser considerados diferencialmente expresados.

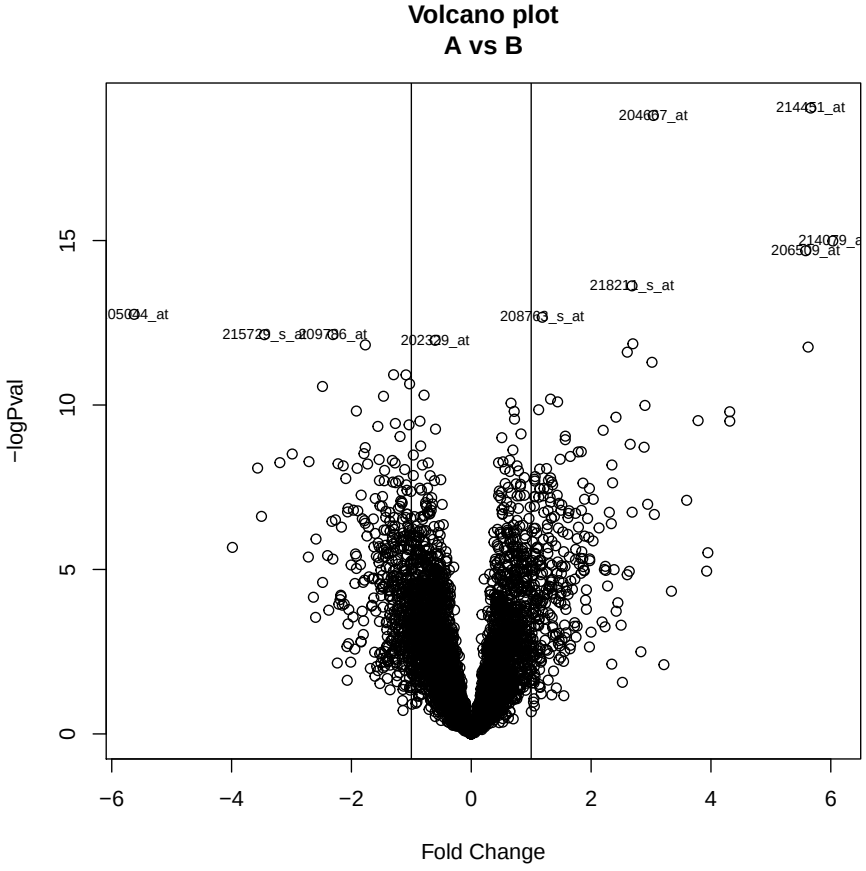
```
topDown<-order(teststat$p.value)
ranked<-teststat[topDown,]
```

```
print(top10<-ranked[1:10,])
```

```
##             statistic        dm      p.value
## 214451_at    25.81132  5.665059 5.445929e-09
## 204667_at    25.08796  3.038344 6.819410e-09
## 214079_at    15.42509  6.032815 3.102260e-07
## 206509_at    14.84966  5.581047 4.166328e-07
## 218211_s_at  12.92792  2.682284 1.212997e-06
## 205044_at   -11.54100 -5.626206 2.884239e-06
## 208763_s_at  11.40712  1.184755 3.151220e-06
## 209786_at   -10.63889 -2.308988 5.335935e-06
## 215729_s_at -10.63444 -3.452290 5.352779e-06
## 202329_at   -10.39560 -0.602802 6.348918e-06
```

Volcano plots Un gráfico muy útil para hacerse una idea de si hay muchos o pocos genes cambiados es el *volcano plot* que representa la significación de cada gen, medida por *"menos el logaritmo del p–valor"* frente al cambio de expresión medido por el `fold--change` o diferencia de medias, que puede considerarse equivalente a una expresión relativa siempre que los datos se encuentren en una escala logarítmica ("el logaritmo del cociente es la diferencia de logaritmos").

```
x<-ranked$dm; y<--log(ranked$p.value)
plot(x, y, xlab="Fold Change", ylab ="-logPval",
     main="Volcano plot\nA vs B")
abline(v=-1);abline(v=1);
text (x[1:10], y[1:10],rownames(ranked)[1:10], cex=0.7)
```



Alternativas al test t de Student La utilización del test-t se basa en la suposición de normalidad de los datos que deberíamos verificar, lo cual resulta difícil porque, gen a gen, se dispone sólo de 10 puntos.

A falta de dicha comprobación podríamos usar aproximaciones menos restrictivas como tests de permutaciones similares a los incluïdos en el paquete `multtest`.

En este ejemplo y con fines ilustrativos aceptaremos que los p–valores obtenidos nos permite realizar una ordenación válida de los genes de más a menos diferencialmentee expresados.

Aún así subsisten varios problemas que deberíamos considerar:

- ¿Cómo podemos seleccionar los genes que deseamos considerar "expresados diferencialmente"?
- ¿Cómo podemos establecer una línea de corte que garantice un control similar ded error tipo I al de los análisis estadísticos tradicionales?
- ¿Como podemos, por lo menos, obtener estimaciones decente de los porcentajes de falsos positivos y falsos negativos?

5.1.2 Selección de los genes [más] expresados diferencialmente

Suponiendo que consideramos válidas las condiciones para aplicar el test–t podemos clasificar los genes por sus p–valores y seleccionar algunos de los genes de la parte superior de la lista (es decir los que tienen menor p-valor)

```
selectedNaif <-ranked[ranked$p.value<0.01,]
nrow(selectedNaif)

## [1] 647
```

Sin embargo, el hecho de que estamos haciendo miles de pruebas al mismo tiempo puede inducir a error. Una posibilidad es ajustar los p–valores de tal manera que se obtenga algún tipo de *control de la tasa de error global*, es decir la probabilidad de obtener falsos positivos, no con una sino con todas las comparaciones.

Otra posibilidad es estimar la *tasa de falsos positivos o "False Discovery Rate" (FDR)*, el porcentaje de falsos positivos entre los genes considerados positivos.

En este caso nos centraremos en la primera opción, ajustar los p–valores.

5.1.3 Ajuste de p–valores para comparaciones múltiples

El ajuste para pruebas múltiples de los p–valores permite obtener un control global de la probabilidad de considerar equivocadamente un gen como diferencialmente expresado.

Es decir, si seleccionamos los genes cuyos p–valores ajustados, p^* están por debajo de un umbral determinado, ($p^* \leq \alpha$), significa que en lugar de hacer las pruebas con un error tipo I de α las hacemos con una FDR (o FWER) esperada igual o inferior a α.

Hay muchos métodos para ajustar los p–valores. El paquete multtest proporciona funciones sencillas que permiten realizar los ajustes más usuales.

```
stopifnot(require(multtest))
#procs <- c("Bonferroni","Holm","Hochberg","SidakSS","SidakSD","BH", "BY")
procs <- c("Bonferroni","BH")
adjPvalues <- mt.rawp2adjp(ranked$p.value, procs)
names(adjPvalues)

## [1] "adjp"    "index"   "h0.ABH"  "h0.TSBH"

ranked.adjusted<-cbind(ranked[,c(1,3)], adjPvalues$adjp[,-1])
ranked.adjusted[1:10,]

##             statistic      p.value   Bonferroni           BH
## 214451_at    25.81132 5.445929e-09 3.377021e-05 2.114358e-05
## 204667_at    25.08796 6.819410e-09 4.228716e-05 2.114358e-05
## 214079_at    15.42509 3.102260e-07 1.923711e-03 6.412371e-04
## 206509_at    14.84966 4.166328e-07 2.583540e-03 6.458851e-04
## 218211_s_at  12.92792 1.212997e-06 7.521793e-03 1.504359e-03
## 205044_at   -11.54100 2.884239e-06 1.788517e-02 2.791531e-03
```

```
## 208763_s_at  11.40712 3.151220e-06 1.954071e-02 2.791531e-03
## 209786_at   -10.63889 5.335935e-06 3.308813e-02 3.688065e-03
## 215729_s_at -10.63444 5.352779e-06 3.319258e-02 3.688065e-03
## 202329_at   -10.39560 6.348918e-06 3.936964e-02 3.721422e-03
```

Como puede verse el número de genes seleccionado si fijáramos el mismo punto de corte sería inferior a si no lo hiciéramos Con los p–valores ajustados podemos usar puntos de corte de 0.01 o 0.05 y, si el ajuste controla la tasa de falsos positivos, este valor puede subir hasta 0.25 puesto que lo que indicaría es que si declaráramos diferencialmente expresados los genes con un p–valor ajustado menor o igual que aquel esperaríamos que, de cada 100 que verificáramos se encontraran 25 falsos positivos, lo que puede resultar perfectamente aceptable en estudios exploratorios.

```
selectedAdjusted<-ranked.adjusted[ranked.adjusted$BH<0.01,]
nrow(selectedAdjusted)

## [1] 19

selectedAdjusted2<-ranked.adjusted[ranked.adjusted$BH<0.05,]
nrow(selectedAdjusted2)

## [1] 225

selectedAdjusted2<-ranked.adjusted[ranked.adjusted$BH<0.25,]
nrow(selectedAdjusted2)

## [1] 1877
```

5.2 Análisis basado en modelos lineales

Como se ha indicado en la introducción a este apartado, cuando hay más de dos grupos la aproximación natural es realizar algun tipo de análisis de la varianza.

En este ejemplo, se aplicará la aproximación desarrollada por Smyth *et al.* (2004) basado en la utilización del *modelo lineal general* combinada con un método para obtener una estimación mejorada de la varianza. Dicho método se conoce con el mismo nombre del paquete: `limma` por *linear models for microarray data*.

5.2.1 Matrices de diseño y de contrastes

El primer paso para el análisis basado en modelos lineales es crear la matriz de diseño. Básicamente se trata de una tabla que describe la asignación de cada muestra a un grupo. Tiene tantas filas como muestras y tantas columnas como grupos (si solo se considera un factor)Cada fila contiene un uno en la columna del grupo al que pertenece la muestra y un cero en las restantes.

La matriz de contrastes esse utiliza para describir las comparaciones entre grupos. Consta de tantas columnas como comparaciones y tantas filas como grupos (es decir como columnas de la matriz de diseño). Una comparación entre grupos –lamada "contraste"– se representa con un "1" y un "-1" en las filas de los

grupos a comparar y ceros en las restantes. Si varios grupos intervinieran en la comparación se tendría tantos coeeficientes como grupos con la única restricción de que su suma sería cero.

La matriz de diseño puede definirse manualmente o a partir de un factor creado específicamente para ello.

Manualmente, seria:

```
design<-matrix(
            c(1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,1,1,1,1,1),
            nrow=15,byrow=F)
colnames(design)<-c("A", "B", "L")
rownames(design) <-  sampleNames
print(design)
```

```
##        A B L
## PF14.A 1 0 0
## PF19.A 1 0 0
## PF23.A 1 0 0
## PF39.A 1 0 0
## PF46.A 1 0 0
## PF24.B 0 1 0
## PF25.B 0 1 0
## PF28.B 0 1 0
## PF34.B 0 1 0
## PF42.B 0 1 0
## PF15.L 0 0 1
## PF32.L 0 0 1
## PF33.L 0 0 1
## PF43.L 0 0 1
## PF47.L 0 0 1
```

Las comparaciones que nos interesan son las diferencias, dos a dos entre cada tipo de tumor lo que puede hacerse con los contrastes siguientes:

```
require(limma)
cont.matrix <- makeContrasts (
      AvsB = B-A,
      AvsL = L-A,
      BvsL = L-B,
      levels=design)
print(cont.matrix)
```

```
##       Contrasts
## Levels AvsB AvsL BvsL
##      A   -1   -1    0
##      B    1    0   -1
##      L    0    1    1
```

5.2.2 Estimación del modelo y selección de genes

Una vez definida la matriz de diseño y los contrastes podemos pasar a estimar el modelo, estimar los contrastes y realizar las pruebas de significación que nos indiquen, para cada gen y cada comparación, si puede considerarse diferencialmente expresado.

El método implementado en `limma` amplía el análisis tradicional utilizando modelos de Bayes empíricos para combinar la información de toda la matriz de datos y de cada gen individual y obtener estimaciones de error mejoradas.

El análisis proporciona los estadísticos de test habituales como `Fold--change` t-moderados o p-valores ajustados que se utilizan para ordenar los genes de mas a menos diferencialmente expresados.

A fin de controlar el porcentaje de falsos positivos que puedan resultar del alto numero de contrastes realizados simultaneamente los p–valores se ajustan de forma que tengamos control sobre la tasa de falsos positivos utilizando el metodo de Benjamini y Hochberg ([1]).

La funcion `topTable` genera para cada contraste una lista de genes ordenados de mas a menos diferencialmente expresados.

```
topTab_AvsB <- topTable (fit.main, number=nrow(fit.main), coef="AvsB", adjust="fdr")
topTab_AvsL <- topTable (fit.main, number=nrow(fit.main), coef="AvsL", adjust="fdr")
topTab_BvsL  <- topTable (fit.main, number=nrow(fit.main) , coef="BvsL", adjust="fdr")
```

Los volcano-plots permiten visualizar si hay muchos o pocos genes con un gran fold-change y significativamente expresados o si este número es bajo. Estos gráficos representa en abscisas los cambios de expresión en escala logarítmica y en ordenadas el "menos logaritmo" del p-valor o alternativamente el estadístico B ("log-odds").

5.2.3 Principales genes expresados diferencialmente

Si consideráramos genes expresados diferencialmente aquellos que tienen un p–valor ajustado inferior a 0.05 o 0.01 el número de genes diferencialmente expresado en cada caso sería:

Los 10 genes más expresados diferencialmente en cada comparación se muestran en las tablas siguientes:

6 Post-procesado de las listas de genes obtenidas

Una vez obtenidas las listas de genes diferencialmente expresados pueden llevarse a cabo todo tipo de análisis sobre ellas, generalmente encaminados a facilitar la interpretación de los resultados.

Entre estas exploraciones –que podemos llamar genericamente "post-procesado de las listas" se encuentra

- La anotación de las listas de genes en diversas bases de datos.
- La comparación entre las listas para determinar que genes cambian simultaneamente -o no- en varias comparaciones (o cuales cambian en una comparación pero no en otra).

- La visualización de todos los genes seleccionados en varias comparaciones para, de forma similar a lo anterior, detectar grupos de genes con patrones de cambio similares –o distintos– entre distintas comparaciones.
- EL análisis de significación biológica de las listas mediante análisis de enriquecimiento o mediante "gene set analysis" para detectar si las listas se encuentran enriquecidas en genes asociados a funciones o procesos biológicos determinados.

6.1 Comparaciones múltiples

Cuando se realizan varias comparaciones a la vez puede resultar importante ver que genes cambian simultáneamente en más de una comparación. Si el número de comparaciones es alto también puede ser necesario realizar un ajuste de p-valores entre las comparaciones, distinto del realizado entre genes.

La función `decidetests` permite realizar ambas cosas. En este ejemplo no se ajustaran los p–valores entre comparaciones. Tan solo se seleccionaran los genes que cambian en una o más condiciones.

EL resultado del análisis es una tabla `res` que para cada gen y cada comparación contiene un 1 (si el gen esta sobre-expresado o "up" en esta condicion), un 0 (si no hay cambio significativo) o un -1 (si esta "down"-regulado).

Para resumir dicho análisis podemos contar qué filas tienen como mínimo una celda distinta de cero:

```
sum.res.rows<-apply(abs(res),1,sum)
res.selected<-res[sum.res.rows!=0,]
print(summary(res))
```

```
##        AvsB AvsL BvsL
## Down     90   11   19
## NotSig 6018 6176 6158
## Up       93   14   24
```

En vista de estos valores podemos aplicar otros criterios de selección, por ejemplo genes con un p-valor ajustado inferior a 0.05 y `log Fold change` mayor o igual a 1. *Este criterio combina a la vez la significación estadística y la significación biológica por lo que, en un estudio real sería probablemente el escogido.*

```
##        AvsB AvsL BvsL
## Down    192   36   55
## NotSig 5766 6121 6069
## Up      243   44   77
```

Un diagrama de Venn permite visualizar la tabla anterior sin diferenciar entre genes "up" o "down" regulados.

6.2 Anotación de resultados

La identificación de los genes seleccionados puede resultar más sencilla para el especialista en un campo si se utilizan nombres estándar como el simbolo del gen o "gene symbol". Con esta finalidad cada paquete de anotaciones tiene tablas

```
vennDiagram (res.selected[,1:3], main="Genes in common #1", cex=0.9)
```

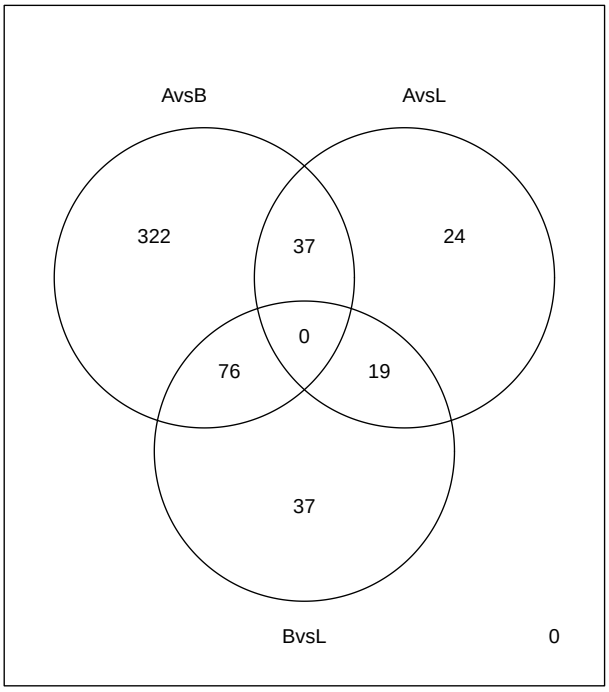


Figure 8: Número de genes seleccionado en cada comparación

de correspondencia entre los distintos tipos de identificadores, principalmente entre los del array y los de otras bases de datos.

Para saber que anotaciones estan disponibles debe cargarse el paquete y llamar la función del mismo nombre

```
require(hgu133a.db)
hgu133a()

## Quality control information for hgu133a:
##
##
## This package has the following mappings:
##
## hgu133aACCNUM has 22283 mapped keys (of 22283 keys)
## hgu133aALIAS2PROBE has 46188 mapped keys (of 124367 keys)
## hgu133aCHR has 19857 mapped keys (of 22283 keys)
## hgu133aCHRLENGTHS has 93 mapped keys (of 595 keys)
## hgu133aCHRLOC has 19711 mapped keys (of 22283 keys)
## hgu133aCHRLOCEND has 19711 mapped keys (of 22283 keys)
## hgu133aENSEMBL has 19565 mapped keys (of 22283 keys)
## hgu133aENSEMBL2PROBE has 13633 mapped keys (of 29048 keys)
## hgu133aENTREZID has 19860 mapped keys (of 22283 keys)
## hgu133aENZYME has 3056 mapped keys (of 22283 keys)
## hgu133aENZYME2PROBE has 892 mapped keys (of 975 keys)
## hgu133aGENENAME has 19860 mapped keys (of 22283 keys)
## hgu133aGO has 19230 mapped keys (of 22283 keys)
## hgu133aGO2ALLPROBES has 19454 mapped keys (of 22911 keys)
## hgu133aGO2PROBE has 14985 mapped keys (of 18177 keys)
## hgu133aMAP has 19798 mapped keys (of 22283 keys)
## hgu133aOMIM has 17279 mapped keys (of 22283 keys)
## hgu133aPATH has 7877 mapped keys (of 22283 keys)
## hgu133aPATH2PROBE has 229 mapped keys (of 229 keys)
## hgu133aPMID has 19794 mapped keys (of 22283 keys)
## hgu133aPMID2PROBE has 435275 mapped keys (of 632694 keys)
## hgu133aREFSEQ has 19769 mapped keys (of 22283 keys)
## hgu133aSYMBOL has 19860 mapped keys (of 22283 keys)
## hgu133aUNIGENE has 19822 mapped keys (of 22283 keys)
## hgu133aUNIPROT has 19326 mapped keys (of 22283 keys)
##
##
## Additional Information about this package:
##
## DB schema: HUMANCHIP_DB
## DB schema version: 2.1
## Organism: Homo sapiens
## Date for NCBI data: 2015-Sep27
## Date for GO data: 20150919
## Date for KEGG data: 2011-Mar15
## Date for Golden Path data: 2010-Mar22
## Date for Ensembl data: 2015-Jul16
```

Cada tabla de asociación puede consultarse de diversas formas,

- Con las funciones `get` o `mget`.
- Convirtiéndola en una tabla y extrayendo valores
- En algunos casos utilizando funciones específicas como `getSYMBOL` o `getEG` (por "Entrez Gene") cuando exitan.

Bioconductor dispone de algunos paquetes que permiten aprovechar esta funcionalidad anterior para obtener las anotaciones de cada gen y generar una tabla HTML con enlaces a algunas bases de datos.

De forma sencilla es posible obtener tablas con las anotaciones correspondientes a los genes seleccionados. Si se desea ser más ambicioso es posible generar tablas en las que se combinen hiperenlaces a las anotaciones con los resultados de la selección de genes.

Veamos las distintas posibilidades, en orden de complejidad creciente

6.2.1 Tablas de anotaciones sencillas

El paquete `annafy` permite de forma muy simple generar una tabla de anotaciones con hiperenlaces a las bases de datos para cada anotación seleccionada.

La instrucción siguiente crea una tabla con las anotaciones disponibles para los genes seleccionados en la sección de comparaciones múltiples.

```
require(annaffy)
genesSelected <- rownames(res.selected)
at <- aafTableAnn(genesSelected, "hgu133a.db")

## Warning in result_fetch(res@ptr, n = n):  SQL statements must be
issued with dbExecute() or dbSendStatement() instead of dbGetQuery()
or dbSendQuery().
## Warning in result_fetch(res@ptr, n = n):  SQL statements must be
issued with dbExecute() or dbSendStatement() instead of dbGetQuery()
or dbSendQuery().

saveHTML (at, file.path(resultsDir, "anotations.html"),
          "Annotations for selected genes")
```

6.2.2 Archivos de resultados con tablas hiperenlazables

La funcion `htmlpage` permite generar un archivo html a partir de un `data.frame` con una funcionalidad muy interesante: pdemos hacer que una o varias de sus columnas contengan hiperenlaces a alguna base de datos permitiendo acceder a gran cantidad de información sobre los genes de la lista.

Para aprender más sobre su uso se puede leer la viñeta `How to get pretty output` disponible en Bioconductor.

Mediante un bucle apropiado creamos una tabla anotada para cada comparación.

```
require(annotate)
require(hgu133a.db)
listOfTables <- list(AvsB = topTab_AvsB, AvsL = topTab_AvsL, BvsL = topTab_BvsL)
for (i in 1:length(listOfTables)){
  # Seleccionamos la "topTable"
  topTab <- listOfTables[[i]]
  # Escogemos los grupos de sondas a incluir en la tabla
  whichGenes<-topTab["P.Value"]<0.05
  selectedIDs <- rownames(topTab)[whichGenes]
  # Los convertimos a identificadores Entrez ("EG") y a Gene Symbols
  genes<- getEG(selectedIDs, "hgu133a.db")
  simbols <-getSYMBOL(selectedIDs, "hgu133a.db")
  # Haremos la columna de Entrez sea hiperenlazable
  paraEnlace <- list (misgenes=genes)
  # Preparamos el data.frame con el que se creará el archivo de resultados
  otherNames = data.frame(selectedIDs, simbols, topTab[whichGenes,-1])
  names(otherNames) = c("Affy ID", "Gene Symbol", colnames(topTab)[-1])
  # Invocamos la función "htmlpage"
  comparison <- names(listOfTables)[i]
  htmlpage(paraEnlace,
           filename =file.path(resultsDir,
           paste("Selected Genes in comparison ",comparison,".html", sep="")) ,
           title = paste("Diff. expressed genes in comparison ", comparison, sep=""),
           othernames = otherNames,
           table.head = c("Entrez IDs", names(otherNames)),
           table.center = TRUE,
           repository=list("en"))
}
```

Podemos tambien salvar una combinacion de la "topTable", las anotaciones y las expresiones.

```
require(annotate)

EntrezsA <- getEG (rownames(topTab_AvsB), annotation(eset_rma))
SymbolsA <- getSYMBOL (rownames(topTab_AvsB), annotation(eset_rma))
ExpressAndTop_AvsB <- cbind(SymbolsA, EntrezsA, topTab_AvsB, exprs(eset_filtered)[rownames
write.csv2(ExpressAndTop_AvsB, file.path(resultsDir, "ExpressAndTop_AvsB.csv2"))

EntrezsA <- getEG (rownames(topTab_AvsL), annotation(eset_rma))
SymbolsA <- getSYMBOL (rownames(topTab_AvsL), annotation(eset_rma))
ExpressAndTop_AvsL <- cbind(SymbolsA, EntrezsA, topTab_AvsL, exprs(eset_filtered)[rownames
write.csv2(ExpressAndTop_AvsL, file.path(resultsDir, "ExpressAndTop_AvsL.csv2"))

EntrezsA <- getEG (rownames(topTab_BvsL), annotation(eset_rma))
SymbolsA <- getSYMBOL (rownames(topTab_BvsL), annotation(eset_rma))
ExpressAndTop_BvsL <- cbind(SymbolsA, EntrezsA, topTab_BvsL, exprs(eset_filtered)[rownames
write.csv2(ExpressAndTop_BvsL, file.path(resultsDir, "ExpressAndTop_BvsL.csv2"))
```

6.3 Visualización de los perfiles de expresión

Tras seleccionar los genes diferencialmente expresados podemos visualizar las expresiones de cada gen agrupándolas para destacar los genes que se encuentran up o down regulados simultáneamente constituyendo *perfiles de expresión*.

Hay distintas formas de visualización pero aquí tan sólo se presenta el uso de mapas de color o `Heatmaps`.

En primer lugar seleccionamos los genes a visualizar: Se toman todos aquellos que han resultado diferencialmente expresados en alguna de las tres comparaciones.

```
probeNames<-rownames(res)
probeNames.selected<-probeNames[sum.res.rows!=0]
exprs2cluster <-exprs(eset_rma)[probeNames.selected,]
colnames(exprs2cluster)<-sampleNames
color.map <- function(grupo) {
  if (grupo=="A"){
    c<- "yellow"
  }else{
    if (grupo=="B"){
      c<- "red"
    }else{
      c<- "blue"
   }
  }
return(c)}
```

Para representar el Heatmap tan sólo necesitamos la matriz de datos resultante.

Si se desea realizar mapas de color más sofisticados puede utilizarse el paquete `gplots` que implementa una version mejorada en la función `heatmap.2`

6.4 Análisis de la significación biológica (1): Análisis de enriquecimiento

El análisis de significación biológica busca establecer si, dada una lista de genes seleccionados por estar diferencialmente expresados entre dos condiciones, las funciones y procesos biológicos que los caracterizan aparecen en esta lista con mayor frecuencia que entre el resto de genes analizados.

Se han desarrollad multitud de variantes de estos tipos de análisis ([8]) pero aquí utilizaremos el análisis básico de enriquecimiento tal como se describe en los trabajos de Gentleman ([5]) implementados en el paquete `GOstats` de Bioconductor.

El análisis se realiza sobre dos bases de datos de anotaciones, la "Gene Ontology" o la "Kyoto Encyclopedia of Genes and Genomes".

Los análisis de este tipo necesitan un número mínimo de genes para resultar fiables por lo que se incluiran en todos los genes con p–valores ajustados inferiores a 0.05 (sin filtrar por minimo "fold–change").

```
grupColors <- unlist(lapply(pData(eset_rma)$Group, color.map))
heatmap(exprs2cluster, col=rainbow(100), ColSideColors=grupColors, cexCol=0.9)
```

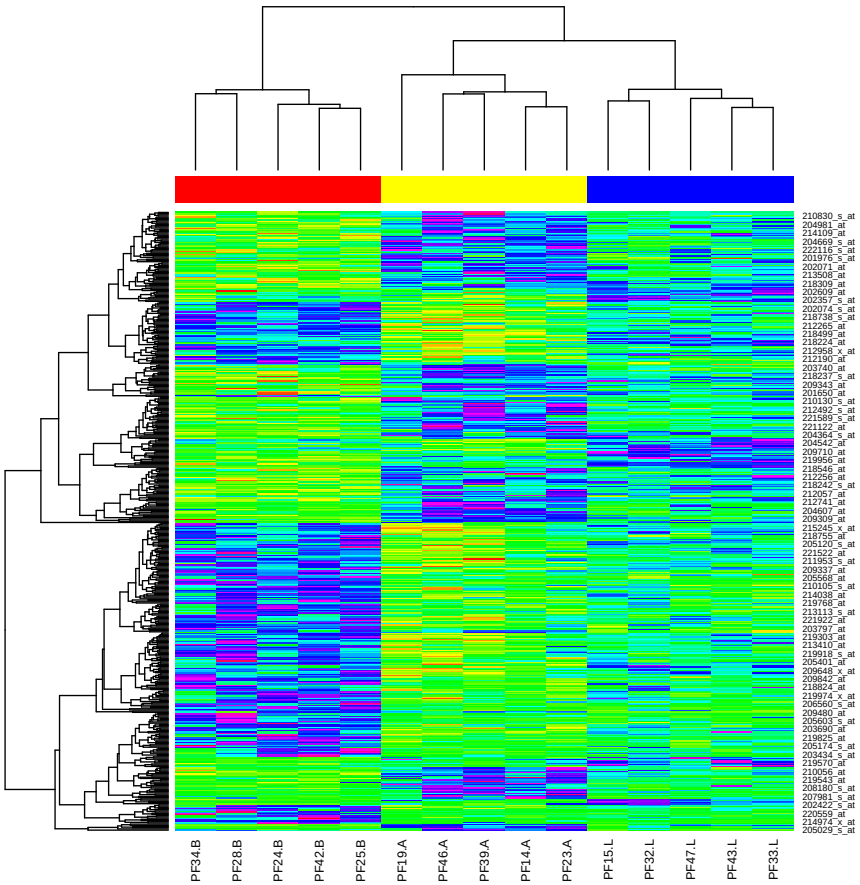


Figure 9: Mapa de colores basado en los genes seleccionados por estar diferencialmente expresados. Como puede verse los tumores Apocrinos y Luminales tienen perfiles de expresión más parecidos entre ellos que cada uno con los de tipo Basal

```
grupColors <- unlist(lapply(pData(eset_rma)$Group, color.map))
require("gplots")
heatmap.2(exprs2cluster,
          col=bluered(75), scale="row",
          ColSideColors=grupColors, key=TRUE, symkey=FALSE,
          density.info="none", trace="none", cexCol=1)
```

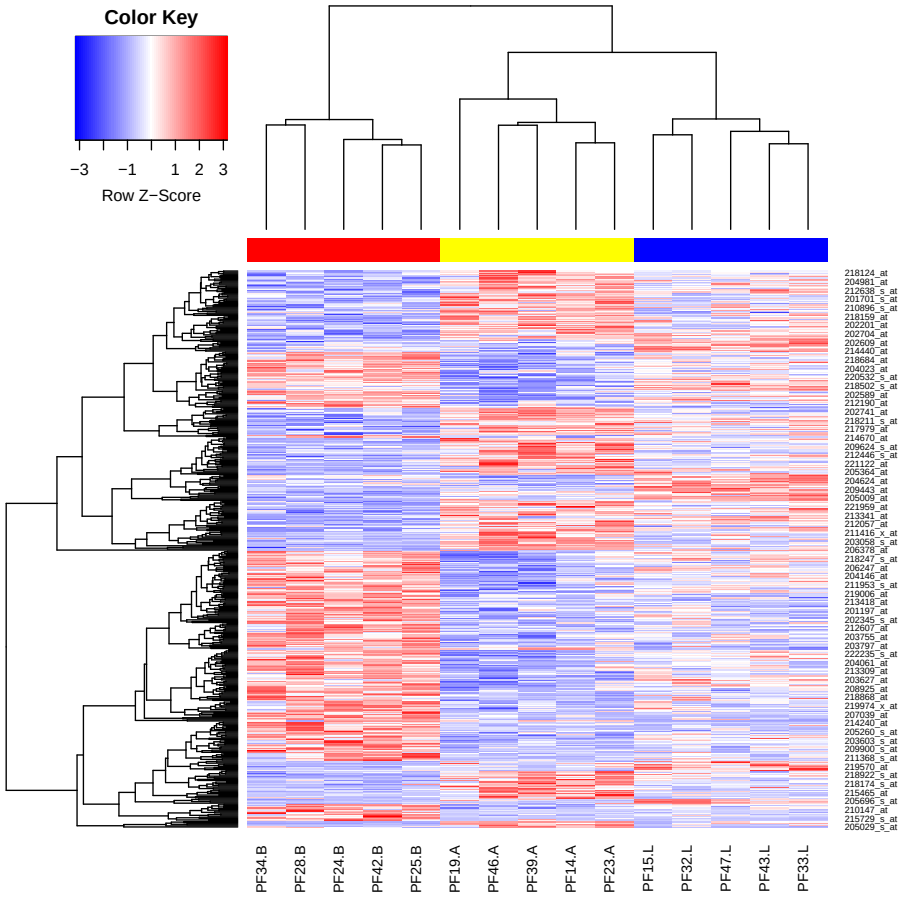


Figure 10: Mapa de colores mejorado basado en los genes seleccionados por estar diferencialmente expresados. Como puede verse los tumores Apocrinos y Luminales tienen perfiles de expresión más parecidos entre ellos que cada uno con los de tipo Basal

```
require(GOstats)
listOfTables <- list(AvsB = topTab_AvsB, AvsL = topTab_AvsL, BvsL = topTab_BvsL)
for (i in 1:length(listOfTables)){
  # Seleccionamos la "topTable"
  topTab <- listOfTables[[i]]
  # Definimos el universo de genes: todos los que se han incluido en el análisis
  # EL programa trabaja con identificadores "entrez" y no admite duplicados

  entrezUniverse = unique(getEG(rownames(topTab), "hgu133a.db"))

  # Escogemos los grupos de sondas a incluir en el análisis
  # Este análisis trabaja bien con varios centenares de genes
  # por lo que es habitual basarse en p-valores sin ajustar para incluirlos

  whichGenes<-topTab["adj.P.Val"]<0.05
  geneIds <-   unique(getEG(rownames(topTab)[whichGenes],"hgu133a.db"))

  # Creamos los "hiperparámetros" en que se basa el análisis
  GOparams = new("GOHyperGParams",
    geneIds=geneIds, universeGeneIds=entrezUniverse,
    annotation="org.Hs.eg.db", ontology="BP",
    pvalueCutoff=0.001, conditional=FALSE,
    testDirection="over")
  # KEGGparams = new("KEGGHyperGParams",
  #   geneIds=geneIds, universeGeneIds=entrezUniverse,
  #   annotation="org.Hs.eg.db",
  #   pvalueCutoff=0.01, testDirection="over")

  # Ejecutamos los análisis

  GOhyper = hyperGTest(GOparams)
  # KEGGhyper = hyperGTest(KEGGparams)

# Creamos un informe html con los resultados
   comparison = names(listOfTables)[i]
   GOfilename =file.path(resultsDir,
     paste("GOResults.",comparison,".html", sep=""))
   # KEGGfilename =file.path(resultsDir,
   #   paste("KEGGResults.",comparison,".html", sep=""))
  htmlReport(GOhyper, file = GOfilename, summary.args=list("htmlLinks"=TRUE))
  # htmlReport(KEGGhyper, file = KEGGfilename, summary.args=list("htmlLinks"=TRUE))
}
```

6.5 Análisis de la significación biológica (2): Expresión diferencial de conjuntos de genes

El análisis de enriquecimiento adolece de un problema consistente en que parte de una lista de genes que ha sido truncada por un punto de corte más o menos

arbitraria, lo que deja a muchos posibles genes interesantes fuera del análisis.

EL "Gene Set Enrichment Analysis" fue introducido para intentar soslayar esta característica. La idea venía a ser que, aunque a veces un gen dado pudiera no estar diferencialmente expresado, un grupo de genes relacionado con una característica médica (enfermedad) o biológica (proceso) podría tender a mostrar más diferencias entre los grupos en estudio que el resto de los genes. Es decir aunque gen a gen no hubiera expresión diferencial este método podría detectars diferencias a nivel del grupo de genes.

A partir del método original se han desarrollado múltiples variaciones. Aquí presentamos la propuesta por Efronn y Tibshirani.

Este método necesita

- Una matriz de expresión, con una fila por gen, y una columna por muestra.
- Un vector de condiciones experimentales.
- Un vector de nombres de los genes (se obtendrá del paquete de anotaciones).
- Una lista de grupos de genes o "genesets" (se extraerá de un archivo descargado: "PBTs_all_affy.csv").

Por simplicidad se analizará la primera comparación: "A vs B".

```
require(annotate)
require(hgu133a.db)
geneSets <- read.csv(file.path(dataDir, "PBTs_all_affy.csv"))

## gene sets a comparar amb cadascuna de les toptable
gsEntrez <- list()
for (i in 1:dim(geneSets)[2]) {
  gs <- as.character(geneSets[,i])
  gs <- gs[gs != ""]
  gsE <- getEG(gs, data = "hgu133a.db")
  gsEntrez[[colnames(geneSets[i])]] <- unique(gsE[!is.na(gsE)])
}

gsSymbol <- list()
for (i in 1:dim(geneSets)[2]) {
  gs <- as.character(geneSets[,i])
  gs <- gs[gs != ""]
  gsSymb <- getSYMBOL(gs,data = "hgu133a.db")
  gsSymbol[[colnames(geneSets[i])]] <- unique(gsSymb[!is.na(gsSymb)])
}
genesetsNames <- names(gsEntrez)
```

```
require(genefilter)
esetUnique <- featureFilter(eset_rma, require.entrez=TRUE, remove.dupEntrez=FALSE)
dim(exprs(esetUnique))

## [1] 19801    15
```

```
esetUnique <- featureFilter(eset_rma, require.entrez=TRUE, remove.dupEntrez=TRUE)
dim(exprs(esetUnique))

## [1] 12402    15

esetAB<- esetUnique[,pData(esetUnique)$Group %in% c("A","B")]
pData(esetAB)

##                Sample             Ids SampleIDs Group Apocrine.grade
## GSM26878.CEL GSM26878 PF14 EnPnT2N1G2      PF14     A              3
## GSM26883.CEL GSM26883 PF19 EnPuT4N0Gu      PF19     A              2
## GSM26887.CEL GSM26887 PF23 EnPnT2N0G2      PF23     A              3
## GSM26903.CEL GSM26903 PF39 EnPuT4N0Gu      PF39     A              3
## GSM26910.CEL GSM26910 PF46 EnPnT4N1G3      PF46     A              3
## GSM26888.CEL GSM26888 PF24 EnPnTiN0G3      PF24     B              2
## GSM26889.CEL GSM26889 PF25 EnPnT3N2G2      PF25     B              1
## GSM26892.CEL GSM26892 PF28 EnPnT2N1G3      PF28     B              2
## GSM26898.CEL GSM26898 PF34 EnPnT3N1G3      PF34     B              1
## GSM26906.CEL GSM26906 PF42 EnPnT2N2G3      PF42     B             na
##              AR.repeat.length
## GSM26878.CEL               18
## GSM26883.CEL               18
## GSM26887.CEL               18
## GSM26903.CEL               17
## GSM26910.CEL               19
## GSM26888.CEL               NA
## GSM26889.CEL               21
## GSM26892.CEL               22
## GSM26898.CEL               17
## GSM26906.CEL               16

xAB <- exprs(esetAB)
yAB <- ifelse(pData(esetAB)$Group=="A", 1, 2)
entrezs<-getEG(rownames(xAB), "hgu133a")
simbols <-getSYMBOL(rownames(xAB), "hgu133a")
```

Una vez recopilados los componentes necesarios se ejecuta la llamada al método **GSA**

```
require(GSA)
GSA.obj<-GSA(xAB, yAB, genenames=entrezs, genesets=gsEntrez,  resp.type="Two class unpaire

## perm= 10 / 100
## perm= 20 / 100
## perm= 30 / 100
## perm= 40 / 100
## perm= 50 / 100
## perm= 60 / 100
## perm= 70 / 100
## perm= 80 / 100
```

```
## perm= 90 / 100
## perm= 100 / 100

GSA.listsets(GSA.obj, geneset.names=genesetsNames, FDRcut=.5)

## $FDRcut
## [1] 0.5
##
## $negative
##      Gene_set Gene_set_name Score     p-value FDR
## [1,] "17"     "KT1.1"       "-0.612"  "0"     "0"
## [2,] "15"     "KT1"         "-0.3206" "0.04"  "0.26"
## [3,] "16"     "KT2"         "-0.409"  "0.06"  "0.26"
## [4,] "18"     "KT2.1"       "-0.3827" "0.11"  "0.3575"
##
## $positive
##       Gene_set Gene_set_name Score    p-value FDR
##  [1,] "14"     "IRITD5"      "0.3503" "0.05"  "0.325"
##  [2,] "2"      "QCMAT"       "0.6988" "0.07"  "0.325"
##  [3,] "8"      "IRRAT"       "0.4766" "0.09"  "0.325"
##  [4,] "13"     "IRITD3"      "0.2047" "0.1"   "0.325"
##  [5,] "7"      "ENDAT"       "0.2056" "0.15"  "0.39"
##  [6,] "3"      "GRIT1"       "0.4351" "0.18"  "0.39"
##  [7,] "1"      "QCAT"        "0.5897" "0.22"  "0.4086"
##  [8,] "6"      "IGT"         "0.2679" "0.28"  "0.4333"
##  [9,] "11"     "BAT"         "0.2612" "0.3"   "0.4333"
##
## $nsets.neg
## [1] 4
##
## $nsets.pos
## [1] 9
```

Si en vez de usar una lista "ad-hoc" de genesets se desea utilizar conjuntos públicos, pueden descargarse de sitios como el Broad Institute `http://software.broadinstitute.org/gsea/downloads.jsp` en formato `.gmt`.

En este ejemplo se utiliza el conjunto de grupos de genes definidos por los pathways de KEGG, aunque para simplificar la salida no se ha ejecutado.

```
geneset.obj <- GSA.read.gmt(file.path(dataDir,"c2.cp.kegg.v5.1.entrez.gmt"))
GSA.obj2 <- GSA(xAB, yAB, genenames=entrezs, genesets=geneset.obj$genesets,  resp.type="Tw
GSA.listsets (GSA.obj2, geneset.names=geneset.obj$geneset.names, FDRcut=.5)
```

7 Resumen de resultados y discusión

Los controles de calidad llevados a cabo en la primera parte de este estudio han permitido establecer que los datos con los que se ha trabajado eran de buena calidad, a pesar de presentar una cierta heterogeneidad que queda ilustrada en la incompleta agrupación de las muestras segun sus tipos de tumores.

Los análisis llevados a cabo sobre los datos normalizados y filtrados han permitido detectar un cierto número de genes diferencialmente expresados (ver tabla 2) que se prestan a estudios posteriores acerca de su significación biológica.

Table 2: Genes diferencialmente expresados basados en un cirte definido por un p-valor ajustado (FDR) inferior a 0.05 y un cambio de expresión de cómo mínimo 1.

	A vs B	A vs L	B vs L
Genes "down-regulados"	228	29	113
Genes "sobre-expresados"	174	27	111

La lista completas de genes seleccionados (ordenados de mayor a menor p–valor) pueden verse en los archivos "Selected genes in comparison AvsB.html", "Selected genes in comparison AvsL.html" y "Selected genes in comparison BvsL.html".

El análisis de significación biológica ha permitido detectar una serie de funciones y procesos biológicos que caracterizan las listas de genes seleccionadas en las bases de datos de anotaciones más populares, la Gene Ontology (GO) y la "Kyoto Enyclopedia of Genes and Genomes". Estas listas pueden verse en las tablas "GOResults.XXvsYY.html" y "KEGGResults.XXvsYY.html" (XXvsYY es AvsB, AvsL o BvsL) pero no se discuten aquí.

7.1 Discusión

El estudio que se ha presentado en este documento es un análisis estándar de microarrays y como tal adolece de sus ventajas e inconvenientes.

Como inconveniente podemos destacar dos cosas

- El tamaño de las muestras utilizadas es bastante limitado lo que determina que el estudio tenga poca potencia por lo que *probablemente* habrá menos reproducibilidad y más falsos negativos de los que sería deseable si se utilizara un mayor número de muestras.

- En cada paso del proceso se han tomado decisiones relativamente arbitrarias acerca de los métodos a seguir para la normalización, filtradom selección de genes, etc. La decisión de si estos métodos son los más adecuados o no es probablemente subjetiva (véase por ejemplo los artículos [2, 12]) por lo que sería interesante saber como cambian los resultados si se tomaran otras decisiones.

Los problemas anteriores no son, sin embargo, problemas de este estudio concreto, sino en general de los estudios basados en microarrays por lo que, limitaciones aparte, el estudio aportará probablemente información valiosa que permitirá un seguimiento posterior del problema.

Finalmente debe de tenerse en cuenta que cualquier gen que se acabe considerando realmente expresado diferencialmente se tendrá que verificar mediante otras técnicas como RT-qPCR por lo que este estudio debe de considerarse como un paso hacia el descubrimiento de genes candidatos, no como una fase definitiva.

References

[1] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B*, 57:289–300, 1995.

[2] Sung Choe, Michael Boutros, Alan Michelson, George Church, and Marc Halfon. Preferred analysis methods for affymetrix genechips revealed by a wholly defined control dataset. *Genome Biology*, 6(2):R16, 2005.

[3] S. Falcon and R. Gentleman. Using gostats to test gene lists for go term association. *Bioinformatics*, 23(2):257–258, 2007.

[4] Pierre Farmer, Herve Bonnefoi, Veronique Becette, Michele Tubiana-Hulin, Pierre Fumoleau, Denis Larsimont, Gaetan Macgrogan, Jonas Bergh, David Cameron, Darlene Goldstein, Stephan Duss, Anne-Laure Nicoulaz, Cathrin Brisken, Maryse Fiche, Mauro Delorenzi, and Richard Iggo. Identification of molecular apocrine breast tumours by microarray analysis. *Oncogene*, 24(29):4660–4671, July 2005. PMID: 15897907.

[5] R. Gentleman. Using go for statistical analysis. *Bioconductor's Compendiums (www.bioconductor.org)*, 2004.

[6] Robert Gentleman, Vince Carey, Wolfgang Huber, Rafael Irizarry, and Sandrine Dudoit. *Bioinformatics and Computational Biology Solutions using R and Bioconductor*. Springer, New York, 2005.

[7] R. A. Irizarry, B. Hobbs, F. Collin, Y. D. Beazer-Barclay, K. J. Antonellis, U. Scherf, and T. P. Speed. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, 4:249–264, 2003.

[8] P. Khatri and S. Drăghici. Ontological analysis of gene expression data: current tools, limitations, and problems. *Bioinformatics*, 18:3587–3595, 2005.

[9] Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physica Verlag, Heidelberg, 2002. ISBN 3-7908-1517-9.

[10] Friedrich Leisch. Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31, December 2002.

[11] Gordon K Smyth. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Stat Appl Genet Mol Biol*, 3:Article3, 2004.

[12] Qianqian Zhu, Jeffrey Miecznikowski, and Marc Halfon. Preferred analysis methods for affymetrix genechips. ii. an expanded, balanced, wholly-defined spike-in dataset. *BMC Bioinformatics*, 11(1):285, 2010.

Codigo R del análisis

El archivo `EjemploAnalisisMicroarrays.R` contiene el código R utilizado para realizar todos los análisis presentados en este informe.

El análisis se ha llevado a cabo utilizando el sistema `Sweave` ([10, 9]) de análisis reproducibles que integra el sistema LaTeXy el lenguaje *R*para generar a la vez los análisis y el informe de resultados por lo que el código puede contener alguna instrucción inusual cuando su objetivo es generar una tabla de LaTeX. Aparte de ésto es un código estándar en el que se ha procurado no usar funciones avanzadas sino básicamente lo que se ha discutido en el curso.