Análisis de microarrays *Clariom D Human* en `limma`

Adam Casas

Compilado: 23 de julio del 2021

Contents

1 Introducción

En este documento se muestra cómo analizar los archivos con extensión .CEL resultantes de secuenciar muestras biológicas en los microchips *Clariom D Human* de Affymetrix. Llevaremos a cabo este análisis en R 3.6 con el paquete de Bioconductor **oligo** versión 1.50.0. La versión de Biocondcutor empleada es la 3.10.

Figure 1: Microchip Clariom D Human y sus correspondientes reactivos.

2 Carga de datos y librerías

Primero debemos instalar oligo. Téngase en cuenta que en R 3.6 se instala la versión 1.50.0 de oligo, el cual depende de los paquetes ff versión 2.2.0 y RSQLite versión 2.1.4. Instalar versiones posteriores ocasionará fallos en la carga de oligo y en los análisis posteriores:

```
# Instalación oligo en R 3.6:
BiocManager::install("oligo")

# Paquete ff versión 2.2.0: (en el portatil la 2.2.14 funciona, en el sobremesa no)
remove.packages("ff")

install.packages("https://cran.r-project.org/src/contrib/Archive/ff/ff_2.2-0.tar.gz",
                 repos=NULL, type = "source")

installed.packages()

# Paquete RSQLite versión 2.1.4:
install.packages("https://cran.r-project.org/src/contrib/Archive/RSQLite/RSQLite_2.1.4.tar.gz",
                 repos=NULL, type = "source")
```

Comenzamos cargando oligo y paquetes accesorios:

```
library("oligo")
library("ggplot2")
library("dplyr")
library("FactoMineR") # PCA
library("rgl") # 3D plots
library("Rtsne") # t-SNE
library("uwot") # UMAP
library("limma") # DEG Analysis
```

Acto seguido definimos el directorio de trabajo en la carpeta donde se encuentran los archivos de secuenciado.

```
directorio_trabajo <- "../Archivos secuenciado/"
```

Listamos y cargamos los archivos de secuenciado en el objeto de tipo HTAFeatureSet denominado datos_crudos_microarrays. La carga de los archivos .CEL se hace mediante la función read.celfiles, la cual lee y carga los archivos por orden alfabético de sus nombres (*i.e.* primero carga el archivo 52CNT_(Clariom_D_Human).CEL, luego 52GEN_(Clariom_D_Human).CEL, luego 53CNT_(Clariom_D_Human).CEL, ... etc):

```
# Obtenemos la ruta completa de los archivos con el argumento `full.names = T`
ruta_completa_archivos_secuenciado <- list.files(path = directorio_trabajo,
                                                 pattern = "*.CEL", full.names = T)

# Cargamos datos
datos_crudos_microarrays <- read.celfiles(filenames = ruta_completa_archivos_secuenciado)
```

```
## Reading in : ../Archivos secuenciado/52CNT_(Clariom_D_Human).CEL
## Reading in : ../Archivos secuenciado/52GEN_(Clariom_D_Human).CEL
## Reading in : ../Archivos secuenciado/53CNT_(Clariom_D_Human).CEL
## Reading in : ../Archivos secuenciado/53GEN_(Clariom_D_Human).CEL
## Reading in : ../Archivos secuenciado/54CNT_(Clariom_D_Human).CEL
## Reading in : ../Archivos secuenciado/54GEN_(Clariom_D_Human).CEL
## Reading in : ../Archivos secuenciado/55CNT_(Clariom_D_Human).CEL
## Reading in : ../Archivos secuenciado/55GEN_(Clariom_D_Human).CEL
```

```
# Observamos el tipo de objeto que hemos cargado
summary(datos_crudos_microarrays)
```

```
##        Length         Class          Mode
##             1 HTAFeatureSet            S4
```

```
# Observamos las sondas y muestras que tenemos
dim(datos_crudos_microarrays)
```

```
## Features  Samples
##  6892960        8
```

Podemos ver el nombre de los microchips con el comando `sampleNames()`.

```
sampleNames(datos_crudos_microarrays)
```

```
## [1] "52CNT_(Clariom_D_Human).CEL" "52GEN_(Clariom_D_Human).CEL"
## [3] "53CNT_(Clariom_D_Human).CEL" "53GEN_(Clariom_D_Human).CEL"
## [5] "54CNT_(Clariom_D_Human).CEL" "54GEN_(Clariom_D_Human).CEL"
## [7] "55CNT_(Clariom_D_Human).CEL" "55GEN_(Clariom_D_Human).CEL"
```

Vamos a cambiar sus nombres por otros más claros:

```
sampleNames(datos_crudos_microarrays) <- c("Control_1", "Genisteina_1",
                                           "Control_2", "Genisteina_2",
                                           "Control_3", "Genisteina_3",
                                           "Control_4", "Genisteina_4")
sampleNames(datos_crudos_microarrays)
```

```
## [1] "Control_1"    "Genisteina_1" "Control_2"    "Genisteina_2" "Control_3"
## [6] "Genisteina_3" "Control_4"    "Genisteina_4"
```

Al cargar los archivos .CEL con read.celfiles(), se almacena en el bolsillo `datos_crudos_microarrays@assayData$exprs` la matriz de intensidades crudas de los fluoróforos Cy3 y Cy5 que hibridan con las distintas sondas de cada microarray. Esta matriz tiene tantas filas como sondas tiene el microarray, y tantas columnas como muestras hayamos secuenciado. Las funciones exprs() e intensity() del paquete oligo simplifican el acceso a la misma (ambas funciones devuelven exactamente el mismo *output*).

- Las filas corresponden a cada sonda individual del microarray, identificada por un nº que va desde el 1 hasta el nº máximo de sondas presentes en el mismo (en el caso de los microarrays *Clariom D Human*, estos contienen un máximo de 6.892.960 sondas).
- Las columnas son las muestras, en nuestro caso Control_1, Genisteina_1, Control_2... etc

```
# Obtenemos la matriz de intensidades cruda
intensidades <- oligo::exprs(datos_crudos_microarrays)

# Identificamos el nº de sondas presentes en cada microchip
dim(intensidades)[1]
```

```
## [1] 6892960
```

De acuerdo a la documentación de Affymetrix, las sondas de los microarrays de tipo Clariom D Human (Gene Arrays) están agrupadas en *transcription clusters* en vez de *probesets*.

Con el comando probeNames() obtenemos los *transcription clusters* a los que pertenecen las sondas de nuestro microchip, de manera que la primera sonda del microarray pertenece al *transcription cluster* TC1300007722.hg.1:

```
# Obtenemos y mostramos los transcription clusters a los que pertenece cada sonda
transcription_clusters <- probeNames(datos_crudos_microarrays)
head(transcription_clusters)
```

```
## [1] "TC1300007722.hg.1" "TC0300011139.hg.1" "TC1600006939.hg.1"
## [4] "TC0100018028.hg.1" "TC0500010071.hg.1" "TC1900010290.hg.1"
```

```
# Cada microarray de tipo Clariom D Human contiene 1.220.891 transcription
# clusters
length(transcription_clusters)
```

```
## [1] 1220891
```

```
# Obtenemos las sondas que pertenecen al transcription cluster TC1300007722.hg.1
which(transcription_clusters == "TC1300007722.hg.1")
```

```
## [1]       1  206732  334298  840938  990710 1101378
```

Si tenemos en cuenta la página web de ThermoFisher, podemos usar el centro de análisis NetAffx para obtener más información sobre dichos clusters. Por ejemplo, si nos fijamos en el segundo *transcription cluster*, TC0300011139.hg.1, vemos que las sondas de este cluster detectan la presencia/ausencia de expresión del pseudogen RNA5SP132 (identificador Ensembl:ENSG00000201595).

Además, el paquete Biobase incluye también numerosas funciones que se pueden ejecutar sobre los objetos de tipo HTAFeatureSet para obtener informacion contenida en el mismo, tales como abstracts, anotaciones, ... etc.

```
# Anotación usada para nuestro objeto `HTAFeatureSet`
Biobase::annotation(datos_crudos_microarrays)
```

```
## [1] "pd.clariom.d.human"
```

```
# Acceder a los datos del experimento del objeto `HTAFeatureSet` (en nuestro
# caso no está anotado)
Biobase::experimentData(datos_crudos_microarrays)
```

```
## Experiment data
##   Experimenter name:
##   Laboratory:
##   Contact information:
##   Title:
##   URL:
##   PMIDs:
##   No abstract available.
```

En relación a lo dicho en el párrafo anterior, las gráficas que generemos a lo largo del protocolo de análisis pueden usar información ubicada en el bolsillo `datos_crudos_microarrays@phenoData@data`, por lo que podemos añadir metadatos de interés con el comando pData() de Biobase tal que así:

```
# Renombramos la columna "index" a "muestra"
colnames(pData(datos_crudos_microarrays)) <- "muestra"

# Añadimos el factor "Condición" a los metadatos de nuestro estudio
pData(datos_crudos_microarrays)$condicion <- as.factor(rep(c("Control", "Genisteina"),
                                                           times = 4))

# Visualizamos los metadatos
pData(datos_crudos_microarrays)
```

```
##              muestra  condicion
## Control_1          1    Control
```

```
## Genisteina_1       2 Genisteina
## Control_2          3    Control
## Genisteina_2       4 Genisteina
## Control_3          5    Control
## Genisteina_3       6 Genisteina
## Control_4          7    Control
## Genisteina_4       8 Genisteina
```

3 Métricas de calidad

3.1 Efecto lote

Para analizar el efecto lote, podemos usar las fechas en las que se secuenciaron los microarrays. El comando get.celfile.dates() del paquete affyio devuelve la fecha de dicho proceso.

```
affyio::get.celfile.dates(ruta_completa_archivos_secuenciado)
```

```
## [1] "2018-03-13" "2018-03-13" "2018-03-13" "2018-03-13" "2018-03-13"
## [6] "2018-03-13" "2018-03-13" "2018-03-13"
```

Vemos que los microarrays se secuenciaron el mismo día. Adicionalmente, el programa GeneChip Command Console 6.0 de ThermoFisher (compañía propietaria de Affymetrix) aportó la hora del secuenciado, el cual duró 6 horas en total (11AM-5PM).

En vista de que el secuenciado se realizó en un sólo lote (el mismo día y en horas consecutivas), concluimos que la probabilidad de observar varianza debida al ruido técnico del efecto lote es mínima.

3.2 Análisis QC

El programa TAC 4.0.2, propiedad también de ThermoFisher, muestra un resumen muy claro de las métricas de calidad de los microarrays.

File Name count: 8	Labeling Controls Threshold	Hybridization Controls Threshold	Pos vs Neg AUC Threshold	Condition	pos vs neg auc
52CNT_(Clariom_D_Human).sst-r...	Pass	Pass	Pass	Control	0,96
52GEN_(Clariom_D_Human).sst-r...	Pass	Pass	Pass	Genisteina	0,96
53CNT_(Clariom_D_Human).sst-r...	Pass	Pass	Pass	Control	0,95
53GEN_(Clariom_D_Human).sst-r...	Pass	Pass	Pass	Genisteina	0,96
54CNT_(Clariom_D_Human).sst-r...	Pass	Pass	Pass	Control	0,96
54GEN_(Clariom_D_Human).sst-r...	Pass	Pass	Pass	Genisteina	0,96
55CNT_(Clariom_D_Human).sst-r...	Pass	Pass	Pass	Control	0,95
55GEN_(Clariom_D_Human).sst-r...	Pass	Pass	Pass	Genisteina	0,96

Figure 2: Las muestras pasan todos los controles de calidad