T2_ExprDiferencial

Francisco Martínez Picó

10/05/2020

Cargando paquetes

En primer lugar, se cargan los paquetes que se van a utilizar:

- *Biobase*: este paquete para trabajar con Bioconductor contiene estructuras estandarizadas de datos para representar información genómica. Es decir, es necesario para trabajar con ExpressionSet.
- *affy*: nos permite trabajar y analizar datos obtenidos por Microarrays de Affymetrix.
- *genefilter*: se utilizará para realizar los t-test por filas.
- *limma*: paquete utilizado para el análisis de la expresión génica en datos de microarray utilizando modelos lineales.
- *ReportingTools*: se utiliza para crear el informe con los genes significativos.
- *franciscomartinez*: para utilizar las funciones que crean las URL a partir de los ID Entrez y Ensembl.

```
pacman::p_load(Biobase)
pacman::p_load(affy)
pacman::p_load(genefilter)
pacman::p_load(limma)
pacman::p_load(ReportingTools)
pacman::p_load(franciscomartinez)
```

Dataset

Seguidamente, se cargarán los datos obtenidos como producto del procesado de la Tarea 1 (T1_Microarray).

```
data('gse69762', package = 'franciscomartinez')
```

Los datos pueden encontrarse en GEO: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE69762

Comparación gen a gen: test de la t

Tenemos datos relativos a la expresión de genes bajo dos condiciones: **pacientes enfermos con una IBD (Enfermedad de Crohn, CD)** y **pacientes sanos control (CONTROL)**. Suponemos una distribución normal de los datos con varianzas iguales (siendo las medias distintas).

```
tt = genefilter::rowttests(gse69762, pData(gse69762)[,'type'])
head(tt)
```

Cada una de las filas de 'tt' corresponde a un gen. La columna '*statistic*' muestra el valor del estadístico t del contraste de hipótesis para la igualdad de medias. La segunda columna '*dm*' muestra la diferencia de medias, mientras que la tercera columna '*p.value*' muestra el p-valor del contraste de hipótesis.

Se obtiene por tanto un p-valor por cada gen. Observamos cuántos genes se expresarían diferencialmente entre ambos grupos si fijaramos un alpha = 0.05 (siendo alpha la cota de error tipo I, esto es rechazar la hipótesis nula H0 siendo esta cierta).

```
table(tt[,'p.value'] < 0.05)
```

Como resultado vemos que 4.862 genes se expresarían diferencialmente entre los pacientes con enfermedad de Crohn (**CD**) y pacientes sanos (**CONTROL**).

Comparaciones múltiples

Sin embargo, la metodología anterior está diseñada para hacer uno o unos pocos contrastes de hipótesis, mientras que aquí estamos haciendo miles al trabajar con miles de genes. Esto quiere decir que estamos cometiendo muchos errores de tipo I.

Por lo tanto, cuando estamos llevando a cabo numerosos contrastes de hipótesis se trabaja con **tasas de error**. Son extensiones del único error tipo I que se nos plantea cuando realizamos un solo contraste.

Family Wise Error Rate (FWER)

La tasa FWER se define para tratar de no rechazar una hipótesis nula cuando esta es cierta. Se trata de un criterio muy exigente cuando tenemos un número de contraste de hipótesis muy elevado (como es el caso del análisis de datos ómicos). Consecuentemente, se harán test muy conservadores en los que no se rechazará la hipótesis nula Hi a no ser que la evidencia sea muy grande. Perderemos potencia en los contrastes (donde la potencia es la probabilidad de rechazar la hipotesis nula cuando realmente es falsa).

False Discovery Rate (FDR)

Se observó que la tasa FWER era demasiado exigente. La tasa FDR surgió al tratar de conseguir procedimientos más potentes sin que por ello dejaramos de tener una tasa de error razonable controlada. Nos da la proporción de test que hemos rechazado erróneamente. Es decir, esta tasa sería la proporción de hipótesis nulas que son ciertas y que son rechazadas.

p-valores ajustados

Para cada contraste Hi se tiene un p-valor pi. Cuando consideramos todos los contrastes simultáneamente, los valores de alpha serán distintos para cada contraste (y por ello se debería ir comprobando uno a uno si pi menor o igual a alpha). Por ello, se ajustan todos los p-valores (siendo alpha el valor que especificamos para controlar alguna de las tasas de error tipo I, la tasa de error).

Método que controla la FWER: método de Bonferroni

```
p0 = tt[,'p.value']
p.bonferroni = p.adjust(p0, method = 'bonferroni')
new_tt = data.frame(tt,p.bonferroni)
head(new_tt)
```

Fijando un alpha de 0.05, vemos que genes son significativos bajo el p-valor ajustado por el método de Bonferroni:

```
significativos.bonferroni = which(p.bonferroni < 0.05)
head(new_tt[significativos.bonferroni,])
```

¿Cuántos de estos genes eran significativos con el p-valor crudo y cuántos lo son con el p-valor ajustado por el método de Bonferroni?

```
significativos.p0 = which(p0 < 0.05)
length(significativos.p0)
length(significativos.bonferroni)
```

4.862 genes tenían una expresión diferencial con el p-valor crudo y 7 lo son con el p-valor ajustado con el método de Bonferroni. Se puede comprobar que cuando trabajamos con p-valores ajustados con el método de bonferroni (y por tanto trabajamos con tasas FWER) se rechazan muy pocas hipótesis nulas, es decir, este test es muy conservador.

Método que controla la FDR: método de Benjamini y Hochberg (BH)

```
p0 = tt[,'p.value']
p.BH = p.adjust(p0, method = 'BH')
new_tt = data.frame(tt, p.bonferroni, p.BH)
head(new_tt)
```

Fijando un alpha de 0.05, vemos que genes son significativos bajo el p-valor ajustado por el método de BH:

```
significativos.BH = which(p.BH < 0.05)
head(new_tt[significativos.BH,])
```

¿Cuántos de estos genes eran significativos con el p-valor crudo, cuántos lo eran con el p-valor ajustado por el método de bonferroni, y cuántos lo son con el p-valor ajustado por el método de BH?

```
length(significativos.p0)
length(significativos.bonferroni)
length(significativos.BH)
```

4.862 genes tenían una expresión diferencial con el p-valor crudo, 7 lo eran con el p-valor ajustado con el método de Bonferroni y, utilizando el p-valor ajustado con el método de Benjamini-Hockberg, 88 de los genes tienen una expresión diferencial significativa.

Generando el primer informe

Para crear el informe, empezaremos por crear un data frame con la información correspondiente:

```
all_genes = new_tt # Todos los genes (significativos y no significativos).

significative_genes = new_tt[significativos.BH,] # Sólo los significativos con el p-valor ajustado con

ID = fData(gse69762)['PROBEID'] # Identificador de las sondas correspondientes a genes que pudimos anot

symbol = fData(gse69762)['SYMBOL'] # Símbolo del gen. Su nombre.

entrezID = fData(gse69762)['ENTREZID'] # EntrezID del gen.
entrezID_URL = apply(entrezID, 1, getURL_entrez) # Creamos la URL a partir del ID.

ensemblID = fData(gse69762)['ENSEMBL'] # EnsemblID del gen.
ensemblID_URL = apply(ensemblID, 1, getURL_ensembl) # Creamos la URL a partir del ID.

# Queremos coger SÓLO aquellos p.values correspondientes a genes que pudimos anotar correctamente. Mira
final_tt = data.frame(new_tt, rownames(new_tt)) # El PROBEID en 'new_tt' aparece como los nombres de la

positions = match(ID[,'PROBEID'], final_tt[,'rownames.new_tt.']) # Nos quedamos con las posiciones del

gene_pvalues = final_tt[positions,] # Montamos un data.frame con los p.values de únicamente los genes q
gene_pvalues

# Lo que funciona:
df1 = data.frame(probeid = ID, symbol = symbol, entrezID = entrezID_URL, ensembl = ensemblID_URL,
```

```
                p.value = gene_pvalues[,'p.value'], p.bonferroni = gene_pvalues[,'p.bonferroni'], p.BH =

columns = c('probeID', 'symbol', 'entrezID', 'ensembl', 'p.value', 'p.bonferroni', 'p.BH')
colnames(df1) = columns

df1
```

Una vez creado el data.frame, elaboramos el informe con la herramienta ReportingTools:

```
foutput = 'gse69762_ttest.report'
htmlRep1 = HTMLReport(shortName = foutput, title = foutput, reportDirectory = './reports')
publish(df1,htmlRep1)
finish(htmlRep1)
```

Test de la t moderado (limma)

Lo que se hace es ajustar un modelo lineal para cada fila de la matriz de expresión considerando que tenemos valores independientes para las distintas muestras

Consideraremos la variable que asigna a cada muestra si pertenece a una **paciente con enfermedad de Crohn (CD)** o a **un paciente sano (CONTROL)**.

```
pData(gse69762)[,'type']
```

Determinamos la matriz diseño o matriz de modelo:

```
dessign = model.matrix(~ pData(gse69762)[,'type']) # variable respuesta (se omite, es cada una de las f
colnames(dessign) = c('intercept', 'type') # 'intercept' sería ß0 (ordenada en el origen), mientras que
head(dessign)
```

Modelo lineal

Ajustamos los modelos lineales utilizando la función *limma::lmFit*. Con esta función, para cada una de las filas de nuestra matriz de expresión, se ajusta un modelo lineal con una sola variable predictora (es decir, utilizamos **regresión lineal simple**). De esta forma, la constante es distinta para cada uno de los modelos:

```
fit = limma::lmFit(gse69762,dessign)
head(fit$coefficients) # Se puede ver de nuevo ß0 ('intercept') y ß1 ('type').
```

Estimación de los errores estándar

A continuación, se estima el error estándar utilizando para esa estimación un modelo empírico bayesiano '*empirical-Bayes*'. **En esta estimación del error estándar se utiliza información de todos los genes**.

```
new_fit = limma::eBayes(fit)
```

Finalmente, vemos los resultados (ordenados por módulo de t). Las columnas a las que debemos prestar especial atención son t, *P.value* y *adj.P.Val*:

```
limma_out1 = limma::topTable(new_fit, coef = 2, adjust = 'bonferroni', number = nrow(exprs(gse69762)))

# Se visualizan los 10 primeros por defecto. Si queremos visualizar todos, debemos utilizar el parámetr

# Omitimos los genes que no hemos podido anotar:
limma_out1 = na.omit(limma_out1)
limma_out1
```

Y de nuevo, hacemos lo mismo pero utilizando el método de Benjamini-Hockberg:

```
limma_out2 = limma::topTable(new_fit, coef = 2, adjust = 'BH', number = nrow(exprs(gse69762))) # Conten

# Omitimos los genes que no hemos podido anotar:
limma_out2 = na.omit(limma_out2)
limma_out2
```

Debemos fijarnos en el p-valor ajustado (*adj.P.val*) por el método de Benjamini-Hockberg.

Generando el segundo informe

Construimos el data.frame con la información deseada:

```
entrezID = limma_out1['ENTREZID'] # Cogemos el EntrezID de los genes y creamos el link a la URL.
entrezID_URL = apply(entrezID, 1, getURL_entrez)

ensemblID = limma_out1['ENSEMBL'] #  Cogemos el código Ensembl de los genes y creamos el link a la URL.
ensemblID_URL = apply(ensemblID, 1, getURL_ensembl)

limma_gene_pvalues = data.frame(PROBEID = limma_out1['PROBEID'], SYMBOL = limma_out1['SYMBOL'],
                                ENTREZID = entrezID_URL, ENSEMBL = ensemblID_URL,
                                p.value = limma_out1['P.Value'], t = limma_out1['t'],
                                p.bonferroni = limma_out1['adj.P.Val'], p.BH = limma_out2['adj.P.Val'])

columns = c('probeID', 'symbol', 'entrezID', 'ensembl', 'p.value', 't', 'p.bonferroni', 'p.BH')
colnames(limma_gene_pvalues) = columns

limma_gene_pvalues
```

Una vez creado el data.frame, elaboramos el informe con la herramienta ReportingTools:

```
foutput = 'gse69762_limma.report'
htmlRep1 = HTMLReport(shortName = foutput, title = foutput, reportDirectory = './reports')
publish(df1,htmlRep1)
finish(htmlRep1)
```