T3_RNAseq

Francisco Martínez Picó

5/23/2020

Explicación del experimento

Estos datos provienen de un estudio realizado sobre el nematodo *Caenorhabditis elegans*. Este organismo mide aproximadamente 1 mm de longitud y vive en ambientes templados. *C. elegans* es utilizado ampliamente como modelo experimental en Biología. Además, fue el primer organismo multicelular cuyo genoma pudo secuenciarse al completo. Esto sucedió en 1998.

Más concretamente, en este estudio se trata de **analizar el efecto de la dieta de *C. elegans* sobre su metabolismo y crecimiento**. Por tanto, se compara la expresión génica en **tres diferentes grupos** establecidos, contando cada uno de ellos con tres réplicas. Los tres grupos definidos consisten en *C. elegans* crecido en un cultivo con:

1: **E. coli OP50** 2: **Chryseobacterium sp. CHNTR56** 3: **Comamonas sp. 12022**

La relevancia de este estudio subyace en que aporta más información sobre *C. elegans* como organismo modelo. Sin embargo, aún no se ha publicado ningún artículo con estos datos. Los datos fueron depositados en el NCBI por Orcun Hacariz (*Institute of Parasitology*, *McGill University*) el 16 de enero de 2020.

Los datos utilizados en esta viñeta se encuentran en **BioProject código PRJNA601724** y en **GEO código GSE143794**. Los links son los siguientes:

BioProject: https://www.ncbi.nlm.nih.gov/bioproject/601724

GEO: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE143794

Se pueden obtener las runs del mismo en el siguiente link:

SRA run selector: https://www.ncbi.nlm.nih.gov/Traces/study/?acc=PRJNA601724&o=acc_s%3Aa

Cargando paquetes

Antes de empezar, cargamos los paquetes necesarios:

- *Biobase*: este paquete para trabajar con Bioconductor contiene estructuras estandarizadas de datos para representar información genómica.
- *SummarizedExperiment*: paquete utilizado para trabajar con el objeto RangedSummarizedExperiment, que contendrá toda la información de nuestro experimento de RNAseq.
- *Rsamtools*: utilizado para obtener la matriz de conteos (objeto RangedSummarizedExperiment) a partir de los archivos .bam.
- *GenomicFeatures*: contiene una serie de herramientas y métodos para construir y manicular anotaciones de tránscritos.
- *GenomicAlignments*: utilizado para manipular los alineamientos de RNAseq.
- *edgeR*: lo utilizaremos para llevar a cabo el análisis de la expresión diferencial (e, implícitamente, llevar a cabo la normalización de los datos).
- *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.
- *org.Ce.eg.db*: para la anotación de los genes y hallar la correspondencia con el id de entrez

- *AnnotationDbi*: para hacer la anotación de los genes.

Dataset (obtención de los datos)

Todo el proceso de obteción y procesado de datos se realizó sobre el ordenador del master, **masterbio**.

Los archivos .sra que contienen las runs utilizadas son las siguientes:

- SRR10902994 —> *C. elegans* crecido en un cultivo con **E. coli OP50**
- SRR10902995 —> *C. elegans* crecido en un cultivo con **E. coli OP50**
- SRR10902996 —> *C. elegans* crecido en un cultivo con **E. coli OP50**
- SRR10902997 —> *C. elegans* crecido en un cultivo con **Chryseobacterium sp. CHNTR56**
- SRR10902998 —> *C. elegans* crecido en un cultivo con **Chryseobacterium sp. CHNTR56**
- SRR10902999 —> *C. elegans* crecido en un cultivo con **Chryseobacterium sp. CHNTR56**
- SRR10903000 —> *C. elegans* crecido en un cultivo con **Comamonas sp. 12022**
- SRR10903001 —> *C. elegans* crecido en un cultivo con **Comamonas sp. 12022**
- SRR10903002 —> *C. elegans* crecido en un cultivo con **Comamonas sp. 12022**

En primer lugar, se descargaron los 9 archivos .sra con el comando *wget*.

```
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra58/SRR/010647/SRR10902994
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra57/SRR/010647/SRR10902995
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra55/SRR/010647/SRR10902996
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra56/SRR/010647/SRR10902997
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra39/SRR/010647/SRR10902998
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra36/SRR/010647/SRR10902999
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra55/SRR/010647/SRR10903000
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra36/SRR/010647/SRR10903001
wget https://sra-download.ncbi.nlm.nih.gov/traces/sra19/SRR/010647/SRR10903002
```

Seguidamente, se obtuvieron los archivos FASTQ utilizando el comando *fastq-dump* (nuestras runs .sra cuentan con 1 read por archivo).

```
fastq-dump SRR10902994.sra
fastq-dump SRR10902995.sra
fastq-dump SRR10902996.sra
fastq-dump SRR10902997.sra
fastq-dump SRR10902998.sra
fastq-dump SRR10902999.sra
fastq-dump SRR10903000.sra
fastq-dump SRR10903001.sra
fastq-dump SRR10903002.sra
```

Después se descargó el genoma de referencia de *C. elegans*.

```
wget http://igenomes.illumina.com.s3-website-us-east-1.amazonaws.com/Caenorhabditis_elegans/Ensembl/WBc

gzip -d Caenorhabditis_elegans_Ensembl_WBcel235.tar.gz
tar xvf SCaenorhabditis_elegans_Ensembl_WBcel235.tar
```

Se realizaron los alineamientos de los FASTQ sobre el mismo utilizando *bowtie2 -x -U archivo.fastq -S archivo.sam*.

```
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10902994.fastq -S
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10902995.fastq -S
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10902996.fastq -S
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10902997.fastq -S
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10902998.fastq -S
```

```
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10902999.fastq -S
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10903000.fastq -S
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10903001.fastq -S
bowtie2 -x /Caenorhabditis_elegans/Ensembl/WBcel235/Sequence/Bowtie2Index/genome -U SRR10903002.fastq -S
```

A continuación, se pasaron estos archivos .sam a .bam con el comando *samtools view -bS archivo | samtools sort -o archivo.bam.*

```
samtools view -bS SRR10902994.sam | samtools sort -o SRR10902994
samtools view -bS SRR10902995.sam | samtools sort -o SRR10902995
samtools view -bS SRR10902996.sam | samtools sort -o SRR10902996
samtools view -bS SRR10902997.sam | samtools sort -o SRR10902997
samtools view -bS SRR10902998.sam | samtools sort -o SRR10902998
samtools view -bS SRR10902999.sam | samtools sort -o SRR10902999
samtools view -bS SRR10903000.sam | samtools sort -o SRR10903000
samtools view -bS SRR10903001.sam | samtools sort -o SRR10903001
samtools view -bS SRR10903002.sam | samtools sort -o SRR10903002
```

Se creo un archivo con todos los nombres de los archivos .bam utilizando:

```
ls *.bam > BamFiles.txt
```

El contenido de BamFiles.txt era el siguiente:

SRR10902994.bam SRR10902995.bam SRR10902996.bam SRR10902997.bam SRR10902998.bam SRR10902999.bam SRR10903000.bam SRR10903001.bam SRR10903002.bam

Y, finalmente, para crear la matriz de conteos, se utilizó Rsamtools. El código utilizado es el siguiente.

```
library(Rsamtools)
library(GenomicFeatures)
library(GenomicAlignments)
gtfFile = "/Caenorhabditis_elegans/Ensembl/WBcel235/Annotation/Genes/genes.gtf"
txdb = makeTxDbFromGFF(gtfFile, format="gtf")
genes = exonsBy(txdb, by="gene")
dirActualData =  paste(getwd(),"/",sep="")
sampleTableSingle = read.table("BamFiles.txt")
fls = paste(dirActualData,sampleTableSingle[,1],sep="")
bamLst = BamFileList(fls, index=character(),yieldSize=100000,obeyQname=TRUE)
PRJNA601724 = summarizeOverlaps(features = genes,read=bamLst,
                                mode="Union",
                                singleEnd=TRUE,
                                ignore.strand=TRUE,
                                fragments=FALSE)
Run = c("SRR10902994","SRR10902995","SRR10902996","SRR10902997","SRR10902998","SRR10902999",
        "SRR10903000","SRR10903001","SRR10903002")
Treatment = c(1,1,1,2,2,2,3,3,3)
Treatment = factor(Treatment,levels=1:3,labels=c("E.coli","Chryseobacterium","Comamonas"))
colData(PRJNA601724) = DataFrame(Treatment)
colnames(PRJNA601724) = Run
save(PRJNA601724,file="PRJNA601724.rda")
```

Como resultado final obtenemos un objeto **SummarizedExperiment::RangedSummarizedExperiment** que contendrá los datos y los metadatos con los que trabajaremos.

Primera aproximación a los datos

Observamos algunas características de nuestros datos. Contamos con 46.748 filas (o genes) y 9 muestras (recordar que eran tres grupos experimentales con 3 réplicas cada uno).

```
data('PRJNA601724', package = 'franciscomartinez')
se = PRJNA601724
# dim(se)
# assay(se) # Matriz de conteos
# colData(se) # Datos fenotípicos
# rownames(se) # Nombre de los genes
# rowRanges(se) # Información sobre la secuenciación (sobre las reads)
# rowData(se)
```

Filtrado independiente

Para comenzar, hacemos el sumatorio de los conteos por genes (filas). Como era de esperar, **hay genes no tiene ninguna lectura alineada en ninguna de las muestras**. Pero no sólo eso, además, **el primer cuartil es 0**. Esto quiere decir que **al menos la cuarta parte de las lecturas son 0**. Como nuestro objetivo es hacer un análisis de la expresión diferencial, deberemos eliminar estos datos, ya que no nos aportan información de ningún tipo (probablemente, son consecuencia del proceso de alineamiento).

```
x = apply(assay(se),1,sum)
summary(x)
sum(x == 0) # 23.200 genes no tienen ninguna lectura.
se0 = se[which(x > 0),] # Se quitan aquellas que en el total de las 9 muestras tienen 0 reads
dim(se[which(x > 0),]) # Nos quedamos con 23.548 genes
se0
```

Nos quedamos con 23.548 genes. Una vez hecho esto, ya podemos comenzar el proceso de normalización.

Normalización o preprocesado

Los problemas o factores no biológicos que aportan ruido y que tratamos de corregir con la normalización de los datos son:

1. *Profundidad de secuenciación*: tendremos algunos de los genes cuyo conteo en algunas de las muestras será 0, mientras que otros genes contarán con un valor de conteo muy grande. Es decir, **tenemos unos conteos con unas variabilidades muy grandes**. Esto produce un problema a la hora de comparar los datos.

Comprobamos la profundidad de lectura entre las muestras:

```
y = apply(assay(se),2,sum)
summary(y) # Profundidad de secuenciación es similar entre muestras.
```

2. *Composición del RNA*
3. *Contenido GC*
4. *Longitud del gen*: como en el proceso de secuencicación se trocean los mRNAs, si un gen es más grande se producirán más trozos del mismo y por tanto su conteo final será mayor.

Para llevar a cabo esta normalización, podríamos haber utilizado el procedimiento **TMM**. Sin embargo, los métodos utilizados para el análisis de expresión diferencial **edgeR** y **DEseq2** ya incorporan de forma nativa un proceso de homogeneización o normalización. Por lo tanto, como para el análisis de la expresión diferencial utilizaremos **edgeR**, también utilizaremos su proceso de normalización integrado.

Análisis de la expresión diferencial

Para llevar a cabo el **análisis de la expresión diferencial** utilizaremos el método propuesto en la siguiente bibliografía (el modelo probabilístico que subyace es una binomial negativa):

1. Robinson, M. D. & Smyth, G. K. Small-sample estimation of negative binomial dispersion, with applications to SAGE data Biostatistics, 2008, 9, 321-332
2. Robinson, M. D. & Smyth, G. K. Moderated statistical tests for assessing differences in tag abundance Bioinformatics, 2007, 23, 2881-2887

Para ello, utilizarenos el **paquete edgeR**. Como tenemos 3 condiciones, haremos las comparaciones 2 a 2.

```
# 1. E.coli - Chryseobacterium:

sel1 = colData(PRJNA601724)[,"Treatment"] == "E.coli" |
  colData(PRJNA601724)[,"Treatment"] == "Chryseobacterium"

sel1 = which(sel1)

dge1 = DGEList(counts = assay(PRJNA601724)[,sel1],
               group = colData(PRJNA601724)[sel1,"Treatment"])

dge1c =  estimateCommonDisp(dge1) # Asumimos dispersión común para todos los genes
dge1c$common.dispersion # 0.1519314

dge1t = estimateTagwiseDisp(dge1c) # Asumimos una dispersión distinta para cada gen.
dge1t$tagwise.dispersion # Para cada gen

dge1cET = exactTest(dge1c)
dge1tET = exactTest(dge1t)

topTags(dge1cET)
topTags(dge1tET) # Nos centraremos en este.

out1 = topTags(dge1tET, n = nrow(assay(se0)))

p.values1 = out1$table[,"PValue"]
p.values1.BH = p.adjust(p.values1, "BH")

wormID = rownames(out1) # Cogemos los ID de los genes y creamos el link a la URL.

final1 = data.frame(GENE = wormID, P.VALUE = out1[,'PValue'], p.BH = p.values1.BH)

# Para añadir más información de anotación de los genes:
genesInfo = AnnotationDbi::select(org.Ce.eg.db, keys = wormID, columns = c("ENTREZID", "SYMBOL"), keyty

genesInfo

# Nos quedamos con la primera coincidencia de WormBase:
posiciones = match(unique(genesInfo[,1]),genesInfo[,1]) # Para WORMBASE
genesInfo = genesInfo[posiciones,]

final1 = data.frame(WORMBASEID = getURL_WBC(genesInfo[,1]), ENTREZID =getURL_entrez(genesInfo[,2]), SYM
                    P.VALUE = out1[,'PValue'], p.BH = p.values1.BH)
columns = c('WormBaseID', 'EntrezID', 'Symbol', 'p.Value', 'p.BH')
```

```r
colnames(final1) = columns

final1
```

```r
# 2. E.coli - Comamonas:

sel2 = colData(PRJNA601724)[,"Treatment"] == "E.coli" |
  colData(PRJNA601724)[,"Treatment"] == "Comamonas"

sel2 = which(sel2)

dge2 = DGEList(counts = assay(PRJNA601724)[,sel2],
               group = colData(PRJNA601724)[sel2,"Treatment"])

dge2c =  estimateCommonDisp(dge2) # Asumimos dispersión común para todos los genes
dge2c$common.dispersion # 0.03970863

dge2t = estimateTagwiseDisp(dge2c) # Asumimos una dispersión distinta para cada gen.
dge2t$tagwise.dispersion # Para cada gen

dge2cET = exactTest(dge2c)
dge2tET = exactTest(dge2t)

topTags(dge2cET)
topTags(dge2tET) # Nos centraremos en este.

out2 = topTags(dge2tET, n = nrow(assay(se0)))

p.values2 = out2$table[,"PValue"]
p.values2.BH = p.adjust(p.values2, "BH")

wormID = rownames(out2) # Cogemos los ID de los genes y creamos el link a la URL.

final2 = data.frame(GENE = wormID, P.VALUE = out2[,'PValue'], p.BH = p.values2.BH)

# Para añadir más información de anotación de los genes:
genesInfo = AnnotationDbi::select(org.Ce.eg.db, keys = wormID, columns = c("ENTREZID", "SYMBOL"), keyty

genesInfo

# Nos quedamos con la primera coincidencia de WormBase:
posiciones = match(unique(genesInfo[,1]),genesInfo[,1]) # Para WORMBASE
genesInfo = genesInfo[posiciones,]

final2 = data.frame(WORMBASEID = getURL_WBC(genesInfo[,1]), ENTREZID =getURL_entrez(genesInfo[,2]), SYM
                    P.VALUE = out2[,'PValue'], p.BH = p.values2.BH)
columns = c('WormBaseID', 'EntrezID', 'Symbol', 'p.Value', 'p.BH')
colnames(final2) = columns

final2
```

```r
# 3. Chryseobacterium - Comamonas:

sel3 = colData(PRJNA601724)[,"Treatment"] == "Chryseobacterium" |
```

```
  colData(PRJNA601724)[,"Treatment"] == "Comamonas"

sel3 = which(sel3)

dge3 = DGEList(counts = assay(PRJNA601724)[,sel3],
            group = colData(PRJNA601724)[sel3,"Treatment"])

dge3c =  estimateCommonDisp(dge3) # Asumimos dispersión común para todos los genes
dge3c$common.dispersion # 0.1519314

dge3t = estimateTagwiseDisp(dge3c) # Asumimos una dispersión distinta para cada gen.
dge3t$tagwise.dispersion # Para cada gen

dge3cET = exactTest(dge3c)
dge3tET = exactTest(dge3t)

topTags(dge3cET)
topTags(dge3tET) # Nos centraremos en este.

out3 = topTags(dge3tET, n = nrow(assay(se0)))

p.values3 = out3$table[,"PValue"]
p.values3.BH = p.adjust(p.values3, "BH")

wormID = rownames(out3) # Cogemos los ID de los genes y creamos el link a la URL.

final3 = data.frame(GENE = wormID, P.VALUE = out3[,'PValue'], p.BH = p.values3.BH)

# Para añadir más información de anotación de los genes:
genesInfo = AnnotationDbi::select(org.Ce.eg.db, keys = wormID, columns = c("ENTREZID", "SYMBOL"), keyty

genesInfo

# Nos quedamos con la primera coincidencia de WormBase:
posiciones = match(unique(genesInfo[,1]),genesInfo[,1]) # Para WORMBASE
genesInfo = genesInfo[posiciones,]

final3 = data.frame(WORMBASEID = getURL_WBC(genesInfo[,1]), ENTREZID =getURL_entrez(genesInfo[,2]), SYM
                    P.VALUE = out3[,'PValue'], p.BH = p.values3.BH)
columns = c('WormBaseID', 'EntrezID', 'Symbol', 'p.Value', 'p.BH')
colnames(final3) = columns

final3
```

Llegados a este punto ya hemos aplicado el test edgeR. Por lo cual, se han obtenido los genes con expresión diferencial significativa entre las comparaciones dos a dos entre los diferentes grupos. Podemos pasar a crear los informes html utilizando la herramienta ReportingTools.

```
# 1. E.coli - Chryseobacterium:
foutput1 = 'PRJNA601724_Ecoli_Chryseo.report'
htmlRep1 = HTMLReport(shortName = foutput1, title = foutput1, reportDirectory = './reports')
publish(final1,htmlRep1)
finish(htmlRep1)
```

```
# 2. E.coli - Comamonas:
foutput2 = 'PRJNA601724_Ecoli_Comam.report'
htmlRep2 = HTMLReport(shortName = foutput2, title = foutput2, reportDirectory = './reports')
publish(final2,htmlRep2)
finish(htmlRep2)

# 3. Chryseobacterium - Comamonas:
foutput3 = 'PRJNA601724_Chryseo_Comam.report'
htmlRep3 = HTMLReport(shortName = foutput3, title = foutput3, reportDirectory = './reports')
publish(final3,htmlRep3)
finish(htmlRep3)
```

De esta forma se ha conseguido elaborar **tres informes** en los que se muestran los genes que se expresan diferencialmente como resultado de las *comparaciones de los grupos dos a dos.*