Statistical analysis of RNA sequencing data
Practical: Part II

Ernest Turro
University of Cambridge

14 Sep 2016

1 Introduction

In this practical we shall analyse RNA-seq data from a study of the *ps* splice factor in *Drosophila melanogaster* cell cultures [2]. The dataset consists of a treatment and a control group. The treatment group is composed of three cell cultures in which the *ps* splice factor has been knocked down (by approximately 60%). The remaining three cell cultures are untreated and serve as a control. We will investigate the effect of *ps* depletion on gene expression and relative isoform expression.

2 Preliminaries

The practical employs or refers to the following software:

- R (`http://www.r-project.org`)
- Bioconductor packages: Rsamtools, GenomicRanges, GenomicAlignments, biomaRt, DESeq (`http://bioconductor.org`)
- Integrative Genomics Browser
- SAMtools (`http://samtools.sf.net`)
- FastQC (`http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc`)
- FASTX toolkit (`http://hannonlab.cshl.edu/fastx_toolkit`)
- Bowtie1 aligner (`http://bowtie-bio.sf.net`)
- TopHat1 spliced aligner (`http://tophat.cbcb.umd.edu`)
- MMSEQ (`https://github.com/eturro/mmseq`)

To begin, log in to the shell as directed and run the following command:

```
cd ~/data/day3/practical
```

Unless stated otherwise, it is assumed throughout this practical that the working directory, both in the shell and within RStudio, is as shown above. Below is a brief summary of the contents of the directory:

all_reads/ The six sets of FASTQ files containing the sequenced reads.

ref/ Reference files: genome/transcriptome fastas and Bowtie indexes, gene annotations in GTF format and as an RData file.

th_out/ The TopHat output files. There is one merged BAM file per sample. SRX014458...SRX014460 are the IDs for the untreated group. SRX014461...SRX014463 are the IDs for the treated group.

bwt_out/ As above, but for transcriptome alignment using Bowtie.

mmseq_out/ MMSEQ output files containing isoform and gene level expression estimates.

3 Counting reads in R

We will now load the gene annotations for chromosome 2L into R, read in the BAM files and count the number of reads overlapping each gene.

Load the biomaRt library and use it to connect to Ensembl and download the annotations for *Drosophila*:

```
library(biomaRt)
ensembl <- useMart(biomart="ENSEMBL_MART_ENSEMBL",dataset="dmelanogaster_gene_ensembl", host="ensembl.org")
annot<-getBM(
  c("chromosome_name",
  "strand",
  "ensembl_gene_id",
  "ensembl_transcript_id",
  "ensembl_exon_id",
  "start_position", "end_position",
  "transcript_start",
  "transcript_end",
  "exon_chrom_start",
  "exon_chrom_end"),
 mart=ensembl,
 filters="chromosome_name",
 values=c("2L")
)
```

Now have a look at the first few entries of the `annot` object and interpret the meaning of all the fields in the table. What is the length of the longest gene? What is the name of the gene containing the shortest exon? How short is it? What are the implications of this with regards to the alignment?

```
head(annot)
```

```
  chromosome_name strand ensembl_gene_id ensembl_transcript_id ensembl_exon_id
1              2L      1     FBgn0031208           FBtr0300689   FBgn0031208:1
2              2L      1     FBgn0031208           FBtr0300689   FBgn0031208:3
3              2L      1     FBgn0031208           FBtr0300690   FBgn0031208:1
4              2L      1     FBgn0031208           FBtr0300690   FBgn0031208:2
5              2L      1     FBgn0031208           FBtr0300690   FBgn0031208:4
6              2L     -1     FBgn0002121           FBtr0078168       CG2671:13
  start_position end_position transcript_start transcript_end exon_chrom_start
1           7529         9484             7529           9484             7529
2           7529         9484             7529           9484             8193
3           7529         9484             7529           9484             7529
4           7529         9484             7529           9484             8193
5           7529         9484             7529           9484             8668
6           9836        21372             9836          21372            21066
  exon_chrom_end
1           8116
2           9484
3           8116
4           8589
5           9484
6          21372
```

Note: the annotation pertains to the *latest* release of Ensembl. The release used for the alignment is no longer available online, so please load a replacement `annot` object from the saved Rdata file: `load("ref/2L_biomart.Rdata")`.

Now that we have loaded the annotation, let us load in the first BAM file and have a look at the resulting object.

```
library(GenomicAlignments)
alnRanges <- readGAlignments("th_out/untreated2/accepted_hits.bam")
head(alnRanges)
```

```
GappedAlignments with 6 alignments and 0 elementMetadata values:
      seqnames strand       cigar    qwidth     start       end     width
         <Rle>  <Rle> <character> <integer> <integer> <integer> <integer>
  [1]       2L      +         37M        37      7599      7635        37
  [2]       2L      +         37M        37      7608      7644        37
  [3]       2L      +         37M        37      7611      7647        37
  [4]       2L      +         37M        37      7612      7648        37
  [5]       2L      +         37M        37      7614      7650        37
  [6]       2L      +         37M        37      7624      7660        37
          ngap
     <integer>
  [1]         0
  [2]         0
  [3]         0
  [4]         0
  [5]         0
  [6]         0
  ---
  seqlengths:
       2L
   5000000
```

```
length(alnRanges)
```

```
[1] 5215987
```

Notice the CIGAR string field, which contains information on whether a read mapped contiguously (37M: 37 matches) or contains a splice junction (N operation). How many of the 5,215,987 reads aligned with a splice junction? (hint: try the `grepl` command).

Now we create a `GRanges` object containing the ranges of each exon. Since the protocol used in this experiment is unstranded, we remove strand information from this object:

```
exonRanges <- GRanges(seqnames = Rle(annot$chromosome_name),
  ranges = IRanges( start=annot$exon_chrom_start,
  end=annot$exon_chrom_end),
  strand = Rle( annot$strand ),
  exon=annot$ensembl_exon_id,
  gene=annot$ensembl_gene_id )
strand(exonRanges) <- "*"
head(exonRanges)
```

```
GRanges with 6 ranges and 2 elementMetadata values:
      seqnames         ranges strand |          exon        gene
         <Rle>      <IRanges>  <Rle> |   <character> <character>
  [1]       2L [ 7529,  8116]      * | FBgn0031208:1 FBgn0031208
  [2]       2L [ 8193,  9484]      * | FBgn0031208:3 FBgn0031208
  [3]       2L [ 7529,  8116]      * | FBgn0031208:1 FBgn0031208
  [4]       2L [ 8193,  8589]      * | FBgn0031208:2 FBgn0031208
  [5]       2L [ 8668,  9484]      * | FBgn0031208:4 FBgn0031208
  [6]       2L [21066, 21372]      * |     CG2671:13 FBgn0002121
  ---
  seqlengths:
   2L
   NA
```

Next we count the number of alignments per exon and set the name of the counts to the gene name:

```
exonCounts <- countOverlaps( exonRanges, alnRanges )
names( exonCounts ) <- elementMetadata(exonRanges)$gene
```

Take a look at the distribution of read counts per exon:

```
hist(log10(exonCounts), breaks=100)
```

Next we split the `exonCounts` by gene name so we can group the exon-level counts into gene-level counts. The `split` function in R is extremely useful: it takes a vector or data frame and divides it into a list according to a factor of interest. Here we divide the `exonCounts` by gene name before summing the counts across exons of the same gene:

```
splitCounts <- split(exonCounts, names(exonCounts) )
geneCounts <- sapply( splitCounts, function(x) sum(x) )
```

Which gene has the highest number of counts in this sample?

Now repeat as above for the remaining 5 samples. This may take a minute or two to run:

```
sams=file.path("th_out", c(paste("untreated",3:4,sep=""), paste("treated",1:3,sep="")))
for(s in sams) {
  cat("Reading in", file.path(s,"accepted_hits.bam"), "\n")
  alnRanges <- readGAlignments(file.path(s,"accepted_hits.bam") )

  exonRanges <- GRanges(seqnames = Rle(annot$chromosome_name),
    ranges = IRanges( start=annot$exon_chrom_start,
    end=annot$exon_chrom_end),
    strand = Rle( annot$strand ),
    exon=annot$ensembl_exon_id,
    gene=annot$ensembl_gene_id )

  strand(exonRanges) <- "*"

  exonCounts <- countOverlaps( exonRanges, alnRanges )
  names( exonCounts ) <- elementMetadata(exonRanges)$gene
  splitCounts <- split(exonCounts, names(exonCounts) )

  geneCounts <- cbind(geneCounts, sapply( splitCounts, function(x) sum(x) ) )
}

colnames(geneCounts) <- c("untreated2",basename(sams))
head(geneCounts)
```

You should now have a matrix of gene counts with genes along the rows and samples along the columns.

4 Differential expression

Let us start by creating a CountDataSet object containing the count data and the condition contrast:

```
library(DESeq)
cds = newCountDataSet(geneCounts, c(rep("untreated", 3), rep("treated", 3)) )
cds
head(counts(cds))
pData(cds)
```

```
          sizeFactor condition
SRX014458         NA untreated
SRX014459         NA untreated
SRX014460         NA untreated
SRX014461         NA   treated
SRX014462         NA   treated
SRX014463         NA   treated
```

Notice how the pData slot `sizeFactor`, which corresponds to the sample-level normalisation factor, is set to NA for all samples. Let us calculate the appropriate normalisation factors with the DESeq method:

```
cds <- estimateSizeFactors(cds)
pData(cds)
```

```
          sizeFactor condition
SRX014458  1.2733205 untreated
SRX014459  0.8327400 untreated
SRX014460  0.8933315 untreated
SRX014461  1.1883723   treated
SRX014462  0.8535191   treated
SRX014463  1.0669885   treated
```

Recall from the lecture how DESeq calculates and applies the normalisation factors:

- For each gene g in sample i, calculate deviation of $\log r_{ig}$ from the mean $\log r_{ig}$ over all libraries: $d_{ig} = \log r_{ig} - \frac{1}{I}\sum \log r_{ig}$.
- Calculate median over all genes: $\log S^{(i)} = \text{median}(d_{ig})$
- Adjust $\hat{\mu}_{ig}$ by a factor of $S^{(i)}$ for all genes g

Are you able to calculate the size factor for the first sample without using the DESeq function? (hint: the sum of the counts on the log scale can be obtained with `rowSums(log(counts(cds)))` and the median of a vector containing non-finite values can be obtained with `median(vec[is.finite(vec)])`).

The next step is to estimate the dispersion values for each gene and fitting a curve through the mean of the normalised counts vs. the dispersion estimates. By "normalised" we mean that the counts have been adjusted by the size factors. You can obtain them by setting the `normalized` option to `TRUE` in the `counts` accessor:

```
counts(cds, normalized=TRUE)[1:3,]
```

```
            SRX014458  SRX014459  SRX014460 SRX014461  SRX014462  SRX014463
FBgn0000018  605.5035   583.6155   611.1953  626.9079   601.0411   549.2093
FBgn0000052 8737.7846 11447.7508 11738.0834 8403.9321 11002.6833 10417.1692
FBgn0000053 3650.2987  4429.9541  4549.2629 4491.8583  5972.9187  5796.6884
```

Can you reproduce the table above without using `normalized=TRUE` ?

Let's calculate the dispersion and plot the scatterplot of the normalised mean vs. the estimated dispersions with an overlay of the fit.

```
cds <- estimateDispersions( cds )
plot( rowMeans( counts( cds, normalized=TRUE ) ), fitInfo(cds)$perGeneDispEsts, log="xy" )
xg <- 10^seq( -.5, 5, length.out=300 )
lines( xg, fitInfo(cds)$dispFun( xg ), col="red" )
```

The function `fitInfo(cds)$dispFun` simply returns $\alpha_0 + \alpha_1/x$, where x is the input normalised counts. In our case, the fitted coefficients are:

```
fitInfo(cds)$dispFun
```

```
function (q)
coefs[1] + coefs[2]/q
<environment: 0x13d62f740>
attr(,"coefficients")
asymptDisp  extraPois
0.01562284 2.87826709
attr(,"fitType")
[1] "parametric"
```

Finally we can perform the negative binomial test to obtain p-values under the null of no differential expression between the treated and untreated groups:

```
res  <- nbinomTest(cds, "untreated","treated")
head(res)
```

```
           id     baseMean    baseMeanA   baseMeanB foldChange log2FoldChange
1 FBgn0000018   596.245425   600.104766  592.386085  0.9871378    -0.01867664
2 FBgn0000052 10291.233902 10641.206273 9941.261531  0.9342232    -0.09816086
3 FBgn0000053  4815.163527  4209.838569 5420.488486  1.2875763     0.36465796
4 FBgn0000055    30.229172    41.829550   18.628793  0.4453501    -1.16698826
5 FBgn0000056    15.235224    20.877262    9.593186  0.4595040    -1.12185058
6 FBgn0000061     2.727951     2.928028    2.527874  0.8633367    -0.21200470
       pval      padj
1 0.91657303 1.0000000
2 0.64215907 1.0000000
3 0.02554497 0.2865028
4 0.23318301 0.9159927
5 0.21713177 0.8988226
6 1.00000000 1.0000000
```

Plot a histogram of the p-values—is there an enrichment of low p-values? The `padj` label is is a misnomer: a gene has a `padj` value of x if it is significant at a false discovery rate (FDR) of x% while a gene has a `pval` value of x if it is significant at a false positive rate (FPR) of x% (the standard definition of a p-value). How many genes are significant at a FDR of 10%? How many genes are significant at a FPR of 5%? The family-wise error rate (FWER) is the probability of a single false positive among all your tests and can be controlled with the Bonferroni correction. How many genes are significant at a FWER of 5%? (hint: look up the `p.adjust` function).

Now plot the log global mean vs. the log fold change and colour in the significant genes in red:

```
plot(res$baseMean, res$log2FoldChange, log="x", col=ifelse(res$padj < 0.1, "red","black"))
```

Notice how the threshold on the log fold change appears to be stricter for lowly expressed genes. Do you have any intuition as to why this is? (hint: what does this plot look like under the null?).

5 Transriptome alignment and isoform-level estimates

The reads were also aligned to the transcriptome. cDNA sequences for chromosome 2L of *Drosophila melanogaster* were downloaded from Ensembl (`http://ensembl.org`) and saved in the `ref` directory. A Bowtie index was built for them with the `bowtie-build` utility. The sets of FASTQ files were aligned against the reference transcript sequences with the `bowtie` program. We set `-a --best --strata` to keep only the best alignments. We used `-S` to output to SAM format. We also used the `--fullref` option to make sure the full FASTA headers were saved in the output. This is important because the headers contain information about which gene each transcript belongs to. We set the maximum insert size to 400 with `-X 400`. Again, we used `-p 12` to spawn 12 threads.

We only kept pairs that were properly aligned and sorted the alignments by read name using `samtools`.

Here is an example of a Bowtie command:

```
cd ~/data/day3/practical/all_reads/GSM461178_untreated3
bowtie -a --best --strata -S --fullref -p 12 -X 400 ../../../ref/2L_cdna \
  -1 SRR031714_1.fastq,SRR031715_1.fastq -2 SRR031714_2.fastq,SRR031715_2.fastq | \
  samtools view -F 0xC -bS - | samtools sort -n - ../../../bwt_out/SRX014459_namesorted
```

The BAM files are inside the `bwt_out` directory.

We then produced a list of mappings between reads and transcript sets using the `bam2hits` program in the MMSEQ package [3]. These mappings were then fed to the `mmseq` program to produce tables of expression estimates at the isoform level. Aggregate estimates over isoforms of the same gene were also produced. These tables are in the `mmseq_out` folder and are called `*.mmseq` and `*.gene.mmseq` for isoform and gene-level estimates respectively.

6 Isoform-level analysis

The `mmseq2counts.R` R function reads in the MMSEQ estimates, combines them and unnormalises them to (estimated) count equivalents. Read in the MMSEQ files containing isoform and gene level estimates:

```
setwd("mmseq_out")
tab_iso <- mmseq2counts()
tab_gene <- mmseq2counts(dir(pattern=".*gene.mmseq"))
head(tab_iso)
```

```
               SRX014458    SRX014459    SRX014460    SRX014461    SRX014462
FBtr0005088 9.621853e+03 2.254283e+03 2.229896e+03 9.427421e+03 2.192797e+03
FBtr0005673 2.144494e-01 9.033614e-04 8.085199e-04 4.585436e+01 4.373476e-04
FBtr0005674 4.593497e-03 5.724035e-04 6.868712e-04 3.348064e-02 2.209242e-04
FBtr0006151 1.873065e-03 3.162023e-04 4.537447e-04 9.549578e-04 0.000000e+00
FBtr0077377 1.150837e+01 4.529747e+00 6.177427e-01 1.257522e+01 6.171641e-01
FBtr0077378 4.577007e+00 0.000000e+00 0.000000e+00 2.513351e+00 1.560714e+00
               SRX014463
FBtr0005088 2.373606e+03
FBtr0005673 2.950444e-01
FBtr0005674 1.055402e-02
FBtr0006151 1.137599e-04
FBtr0077377 4.497688e+00
FBtr0077378 0.000000e+00
```

```
head(tab_gene)
```

```
            SRX014458.gene SRX014459.gene SRX014460.gene SRX014461.gene
FBgn0000018    821.2365617   2.263759e+02    236.0301298    809.0954520
FBgn0000052   3967.4363495   1.517246e+03   1555.5077560   3685.8030291
FBgn0000053   3306.1248728   1.169517e+03   1213.3348390   3960.9798652
FBgn0000055      7.6726818   4.744607e-02      3.4092459      4.1906191
```

```
FBgn0000056      0.5693853   6.426933e-02      0.1066947      0.3356476
FBgn0000061      2.6221067   6.451597e-01      1.6041405      4.4879443
            SRX014462.gene SRX014463.gene
FBgn0000018    224.0171085   2.798952e+02
FBgn0000052   1378.8453342   1.789369e+03
FBgn0000053   1400.3271860   1.934607e+03
FBgn0000055      0.0000000   5.851156e-01
FBgn0000056      0.0000000   1.767749e-02
FBgn0000061      0.6391335   6.320105e-01
```

Let us now use the gene-level count estimates obtained by MMSEQ by summing over isoforms to run a differential expression analysis and compare them to the previous results that used counts from genome alignment:

```
cds_gene = newCountDataSet(round(tab_gene), c(rep("untreated",times=3), rep("treated",times=3)) )
cds_gene <- estimateDispersions(estimateSizeFactors( cds_gene ) )
res_gene  <- nbinomTest(cds_gene, "untreated","treated")
```

How many of the genes are significant at a FPR of 5%? Of these, how many were also significant with the gene counts approach used earlier?

Now repeat as above for the isoform-level estimates:

```
cds_iso = newCountDataSet(round(tab_iso), c(rep("untreated",times=3), rep("treated",times=3)) )
cds_iso <- estimateDispersions(estimateSizeFactors( cds_iso ) )
res_iso  <- nbinomTest(cds_iso, "untreated","treated")
```

How many of the isoforms are significant at a FPR of 5%?

Finally, let us investigate whether some isoforms appear to be differentially expressed while the overall gene does not. To achieve this, we first need to obtain mappings from isoforms to the genes they belong to and add them to our isoform table:

```
t2g <- unique(annot[,c("ensembl_transcript_id","ensembl_gene_id")])
res_iso <- cbind(res_iso, gene_id = t2g[match(res_iso$id, t2g[,1]),2])
res_iso$gene_id <- as.vector(res_iso$gene_id)
head(res_iso)
```

```
           id    baseMean    baseMeanA   baseMeanB foldChange log2FoldChange
1 FBtr0005088 3651.156058 3727.2553316 3575.056784  0.9591661     -0.0601475
2 FBtr0005673    3.410750    0.0000000    6.821500        Inf            Inf
3 FBtr0005674    0.000000    0.0000000    0.000000        NaN            NaN
4 FBtr0006151    0.000000    0.0000000    0.000000        NaN            NaN
5 FBtr0077377    4.482797    4.8669930    4.098602  0.8421219     -0.2478990
6 FBtr0077378    1.140104    0.7240645    1.556143  2.1491779      1.1037849
       pval padj     gene_id
1 0.8438824    1 FBgn0260439
2 0.4190353    1 FBgn0002887
3        NA   NA FBgn0002887
4        NA   NA FBgn0000056
5 0.9901268    1 FBgn0031620
6 0.9891312    1 FBgn0031621
```

Now, that we have mappings from isoform names to gene names, the list of genes for the significant isoforms can be obtained:

```
sig_iso <- res_iso[res_iso$pval < 0.05 & !is.na(res_iso$pval),]$gene_id
```

How many isoforms which are declared differentially expressed belong to genes which are declared non-differentially expressed? (hint: try the `intersect` function).

Perhaps a good way to visualise this and end the practical is to produce a scatterplot of the log fold change for genes vs. isoforms within those genes. First we need to create a vector containing gene-level estimates which are repeated as many times as it has isoforms:

```
x <- unlist(sapply(1:nrow(res_gene),
  function(i) {
      rep(res_gene[i,]$log2FoldChange, sum(res_iso$gene_id==res_gene[i,]$id))
    }
  )
)
```

Then we create a second vector containing the isoform-level estimates corresponding to the genes in the first vector:

```
y <- unlist(sapply(1:nrow(res_gene),
    function(i) {
      res_iso[res_iso$gene_id==res_gene[i,]$id,]$log2FoldChange
    }
  )
)
```

Finally, we plot x vs. y.

```
plot(x,y, xlab="gene log2FC", ylab="isoform log2FC")
abline(0,1,col=2)

smoothScatter(x,y, xlab="gene log2FC", ylab="isoform log2FC") # smoothed density version
abline(0,1,col=2)
```

7 Acknowledgement

Parts of this practical were inspired by the DESeq [1] vignette and an EMBL practical produced by Ângela Gonçalves and Gabriella Rustici from the European Bioinformatics Institute.

References

[1] Anders, S. and Huber, W. 2010. Differential expression analysis for sequence count data. *Genome Biol*, 11(10):R106.

[2] Brooks, A. N., Yang, L., Duff, M. O., Hansen, K. D., Park, J. W., Dudoit, S., Brenner, S. E., and Graveley, B. R. 2011. Conservation of an RNA regulatory map between Drosophila and mammals. *Genome Res*, 21(2):193–202.

[3] Turro, E., Su, S.-Y., Goncalves, A., Coin, L. J. M., Richardson, S., and Lewin, A. 2011. Haplotype and isoform specific expression estimation using multi-mapping RNA-seq reads. *Genome Biol*, 12(2):R13.