Introduction to RNASeq

Malay (malay@uab.edu)

November 16, 2018

Contents

1 Introduction

RNASeq is a very vast topic and tons of papers have been and are being written on the topic. The following is just an overview.

Originally the idea was proposed by Mortazavi *et al.* (2008). Although several modifciation of the original idea have been developed, the basics did not change. In this handout will use the latest in the RNASeq methodology through the use of software called RSEM (Li and Dewey, 2011).

2 Normalization of RNASeq data

People have proposed several methods of normalization of RNASeq data. For a comparison see Dillies *et al.* (2013).

3 Datasets

Every diffential expression measurment should have biological replicates. For demonstration, we will use only 1 replicate for two biological conditions. But in real life, this should never be used. We will use two small datasets from Illumina Body Map project. These are samples prpared from adrenal gland and brain and only from chromosome 19. You can download the datasets here:

https://github.com/cb2edu/CB2-101-BioComp/raw/2018/10-RNASeq/data/rnaseq_data.tar.gz

Unzip the file.

4 Required software

1. Install on Linux the following packages: `openssl` `openssl-devel` `curl` `curl-devel`
2. It was previously fairly complicated to install bioconductor packages. But if you're using R version 3.5 or above, a package from CRAN can make the process easier. Run in R console:

   install.packages("BiocManager")

Use `BiocManager::install()` function to install R packages.

5 STAR

STAR is a modern fast aligner for RNASeq data to reference genome.

```
wget https://github.com/alexdobin/STAR/archive/2.5.1b.tar.gz
tar -xvzf 2.5.1b.tar.gz
cd STAR-2.5.1b
make
```

Put the software in your path

```
cd Linux_x86_64_static/
export PATH=$PATH:`pwd`
```

Prepare the referene genome:

```
mkdir hs
STAR --runThreadN 8 --genomeDir hs --runMode genomeGenerate \
   --genomeFastaFiles chr19.fa --sjdbGTFfile human_chr19.gtf
```

Now create the alignment. There is a special option for STAR to create a “transcriptome alignment” that could be fed directly to RSEM.

```
STAR --runThreadN 8 --genomeDir hs --readFilesIn adrenal_R1.fq \
    adrenal_R2.fq --quantMode TranscriptomeSAM
```

6 RSEM

RSEM is a cutting-edge RNASeq analysis package that is an end-to-end solution for differential expression, and simplifies the whole process. It also intriduces a new more robust unit of RNASeq measurement called TPM.

6.1 Installing RSEM

```
wget http://deweylab.biostat.wisc.edu/rsem/src/rsem-1.2.19.tar.gz
tar -xvzf rsem-1.2.19.tar.gz
cd rsem-1.2.19/
make
export PATH=$PATH:`pwd`

# Install ebseq
module load R/R-3.1.2
make ebseq
cd EBSeq/
export PATH=$PATH:`pwd`
```

6.2 Prepare reference

```
rsem-prepare-reference --gtf human_chr19.gtf chr19.fa rsem/chr19
```

6.3 Calculate expression directly from STAR output

```
rsem-calculate-expression --no-bam-output --paired-end \
  --bam Aligned.toTranscriptome.out.bam rsem/chr19 adrenal
```

6.4 Simpler way to estimating expression

```
rsem-prepare-reference --gtf human_chr19.gtf --star --star-path \
  ../STAR-2.5.1b/bin/Linux_x86_64_static -p 8 chr19.fa hs/chr19
rsem-calculate-expression  --paired-end --star --star-path \
  ../STAR-2.5.1b/bin/Linux_x86_64_static/ -p 8 adrenal_R1.fq \
  adrenal_R2.fq hs/chr19 adrenal_rsem
```

```
rsem-calculate-expression  --paired-end --star --star-path \
  ../STAR-2.5.1b/bin/Linux_x86_64_static/ -p 8 brain_R1.fq brain_R2.fq \
  hs/chr19 brain_rsem
```

6.5 Differential expression

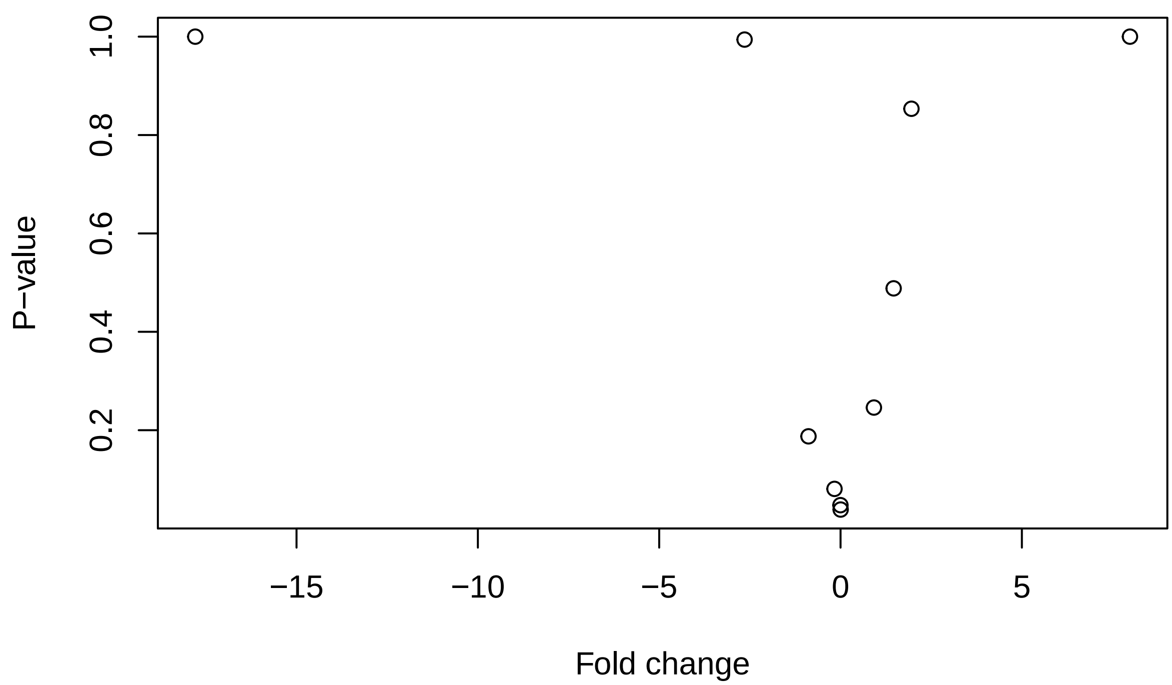
```
rsem-generate-data-matrix adrenal_chr19.genes.results human_chr19.genes.results \
  >diff-brain-adrenal.txt
rsem-run-ebseq diff-brain-adrenal.txt 1,1 expression.results.txt
rsem-control-fdr expression.results.txt 0.05 expression_final.txt
```

And we have our differentially expressed genes.

6.6 Volcano plot

Volcano plot is a good way to show the differentially expressed genes. For that we need the p-value for the differentially expressed genes and the the fold change. Given by "PPEE" and "RealFC" values.

```
data<-read.table("expression.results.txt")
plot(log2(data$RealFC),data$PPDE,xlab="Fold change",ylab="P-value")
```



7 EDGER

For EDGER we need a count table data for mutiple sample. The supplied `pnas_expression.txt` is a sample file derived from the paper here: https://www.ncbi.nlm.nih.gov/pubmed/19088194.

```
raw.data <- read.table("../data/pnas_expression.txt",header=T)
head(raw.data)
counts <- raw.data[ , -c(1,ncol(raw.data))]
rownames(counts) <- raw.data$ensembl_ID
colnames(counts)  <- paste(c(rep("C_R",4),rep("T_R",3)),c(1:4,1:3),sep="")

library(edgeR)
group <- c(rep("C", 4) , rep("T", 3))
cds <- DGEList( counts , group = group )
cds <- calcNormFactors(cds)
design <- model.matrix(~group)
y <- estimateDisp(cds, design)
fit <- glmQLFit(y,design)
qlf <- glmQLFTest(fit,coef=2)
topTags(qlf)
```

8 DESEQ2

```
suppressPackageStartupMessages(library(DESeq2))
counts <- read.table("../data/pnas_expression.txt",header = T)
row.names(counts) <- counts$ensembl_ID
counts <- as.matrix(counts[,-c(1,ncol(counts))])
counts <- counts[rowSums(counts) != 0,]
coldata <- data.frame(condition=c(rep("C",4), rep("T",3)))
row.names(coldata) <- colnames(counts)
#coldata <- as.matrix(coldata)
dds <- DESeqDataSetFromMatrix(countData = counts, colData = coldata, design = ~ condition)
dds <- estimateSizeFactors(dds)
```

Get the size factor estimate

```
#get the sizefactors
sizeFactors(dds)
```

```
##     lane1     lane2     lane3     lane4     lane5     lane6     lane8
## 0.7912131 0.9433354 1.1884939 1.2307145 1.4099376 1.4224762 0.5141056
```

To get the count normalized count table:

```
head(counts(dds, normalized=T))
```

```
##                     lane1     lane2      lane3      lane4     lane5
## ENSG00000124208 604.13562 656.18234 528.399861 604.526907 342.56834
```

```
## ENSG00000182463  34.12482  21.20137  22.717828  21.125940  34.04406
## ENSG00000124201 227.49877 231.09491 246.530508 223.447445 264.55071
## ENSG00000124205   0.00000   0.00000   4.207005   4.062681   0.00000
## ENSG00000124207  96.05504  84.80547  71.519089  78.816008  56.74010
## ENSG00000125835 166.83243 212.01368 168.280211 185.258246 198.59034
##                     lane6     lane8
## ENSG00000124208 503.34760 466.83017
## ENSG00000182463  38.66497  46.68302
## ENSG00000124201 211.60283 171.17106
## ENSG00000124205   0.00000   0.00000
## ENSG00000124207  56.94295  71.96965
## ENSG00000125835 143.41188 101.14654
```

Now we can run the differential expression analysis.

```
dds <- DESeq(dds)
```

```
## using pre-existing size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
results <- results(dds)
results
```

```
## log2 fold change (MLE): condition T vs C
## Wald test p-value: condition T vs C
## DataFrame with 21877 rows and 6 columns
##                          baseMean      log2FoldChange             lfcSE
##                         <numeric>           <numeric>         <numeric>
## ENSG00000124208  529.427260872946  -0.455515880019994 0.139864048241602
## ENSG00000182463  31.2231423174851    0.67450988112742 0.338751310529578
## ENSG00000124201  225.128032833452 -0.0816388444834328 0.156690575767686
## ENSG00000124205  1.18138372707495   -3.53219672310413  2.48011766800347
## ENSG00000124207  73.8354725014805  -0.443039144864501  0.22394686815464
## ...                           ...                 ...               ...
## ENSG00000218597  7.59661107144179  -0.761121820575701 0.676044725980966
## ENSG00000217348  17.9021532077252   0.108422724490082 0.472039639668887
## ENSG00000217342 0.116076595034914  -0.704419543712666  3.81694647031143
## ENSG00000216298 0.180554578664798  -0.704419543712666  3.81694647031143
## ENSG00000183878  137.611821535415   0.482068242918588 0.204800611849386
##                               stat              pvalue                padj
##                          <numeric>           <numeric>           <numeric>
## ENSG00000124208  -3.25684752977501 0.00112656928625465 0.00778617584929813
## ENSG00000182463   1.99116537755365  0.0464627088392363   0.158175972106834
```

```
## ENSG00000124201 -0.521019493887577    0.60235319156063    0.80120731973773
## ENSG00000124205  -1.42420529827022   0.154387051623564                  NA
## ENSG00000124207  -1.97832257497154  0.0478923279674894   0.161990055805166
## ...                            ...                 ...                 ...
## ENSG00000218597  -1.12584536396062   0.260230978843914   0.503303116935888
## ENSG00000217348  0.229689872160176   0.818332765368809   0.923967332544385
## ENSG00000217342 -0.184550543003866   0.853581580430102                  NA
## ENSG00000216298 -0.184550543003866   0.853581580430102                  NA
## ENSG00000183878   2.35384181016563  0.0185805163793376  0.0783501590624973
```

Sort results based on p-value.

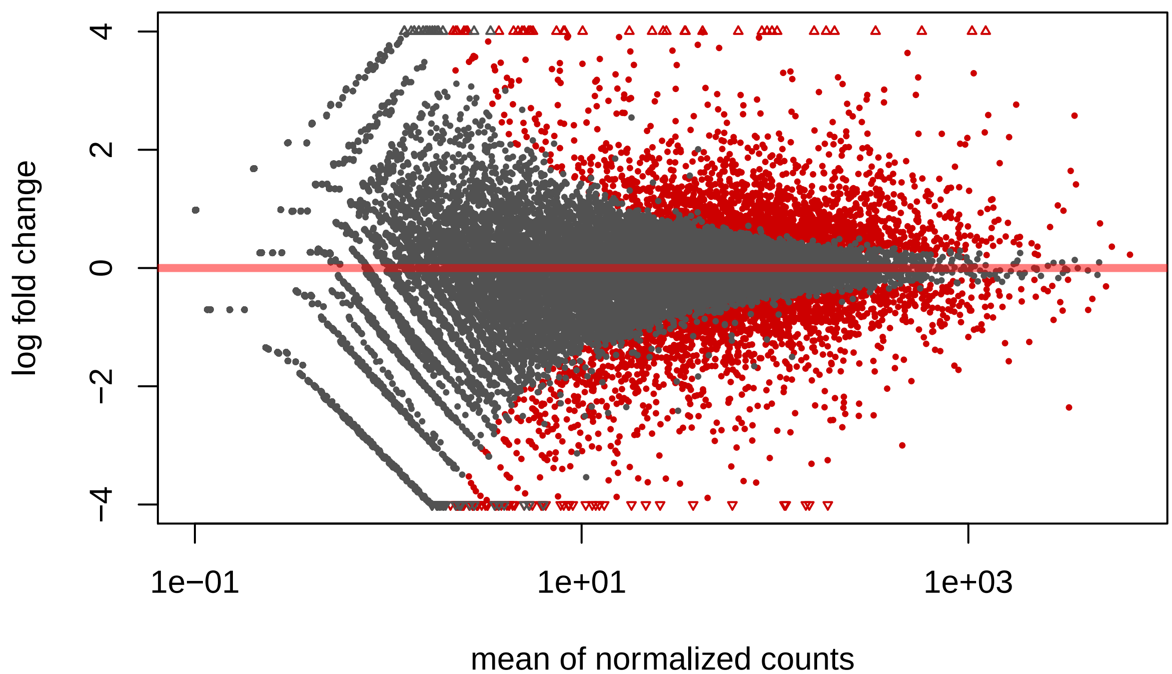
```r
results<- results[order(results$padj),]
results
```

```
## log2 fold change (MLE): condition T vs C
## Wald test p-value: condition T vs C
## DataFrame with 21877 rows and 6 columns
##                          baseMean     log2FoldChange              lfcSE
##                         <numeric>          <numeric>          <numeric>
## ENSG00000115648  3541.83366415174   2.57711098861239 0.0605106044187003
## ENSG00000096060  1229.68447449314   4.98303553728308  0.102418010273941
## ENSG00000151503  1044.92132174665   5.79746873080713  0.128241540075143
## ENSG00000162772  1065.71178318354    3.2941498194609 0.0874343644555919
## ENSG00000166451  574.459907549637   4.66552425498009  0.141831358540006
## ...                           ...                ...                ...
## ENSG00000129864 0.331992920462719  -1.58627675956136   3.71181451287262
## ENSG00000217720 0.180554578664798 -0.704419543712666   3.81694647031143
## ENSG00000218917 0.221521753656518  0.260916396756257   3.81611432442547
## ENSG00000217342 0.116076595034914 -0.704419543712666   3.81694647031143
## ENSG00000216298 0.180554578664798 -0.704419543712666   3.81694647031143
##                              stat               pvalue
##                         <numeric>            <numeric>
## ENSG00000115648   42.589410787904                    0
## ENSG00000096060  48.6538990940633                    0
## ENSG00000151503  45.2074166249883                    0
## ENSG00000162772   37.675687814189                    0
## ENSG00000166451   32.894871084973 2.60222034624108e-237
## ...                           ...                  ...
## ENSG00000129864  -0.42735884405326     0.66911797902377
## ENSG00000217720 -0.184550543003866    0.853581580430102
## ENSG00000218917 0.0683722694276302    0.945489295950407
## ENSG00000217342 -0.184550543003866    0.853581580430102
## ENSG00000216298 -0.184550543003866    0.853581580430102
##                                padj
##                           <numeric>
## ENSG00000115648                   0
## ENSG00000096060                   0
## ENSG00000151503                   0
```

```
## ENSG00000162772                        0
## ENSG00000166451 7.63283271959435e-234
## ...                                  ...
## ENSG00000129864                       NA
## ENSG00000217720                       NA
## ENSG00000218917                       NA
## ENSG00000217342                       NA
## ENSG00000216298                       NA
```

8.1 MA-plot

```
plotMA(results)
```



8.2 Principal component analysis

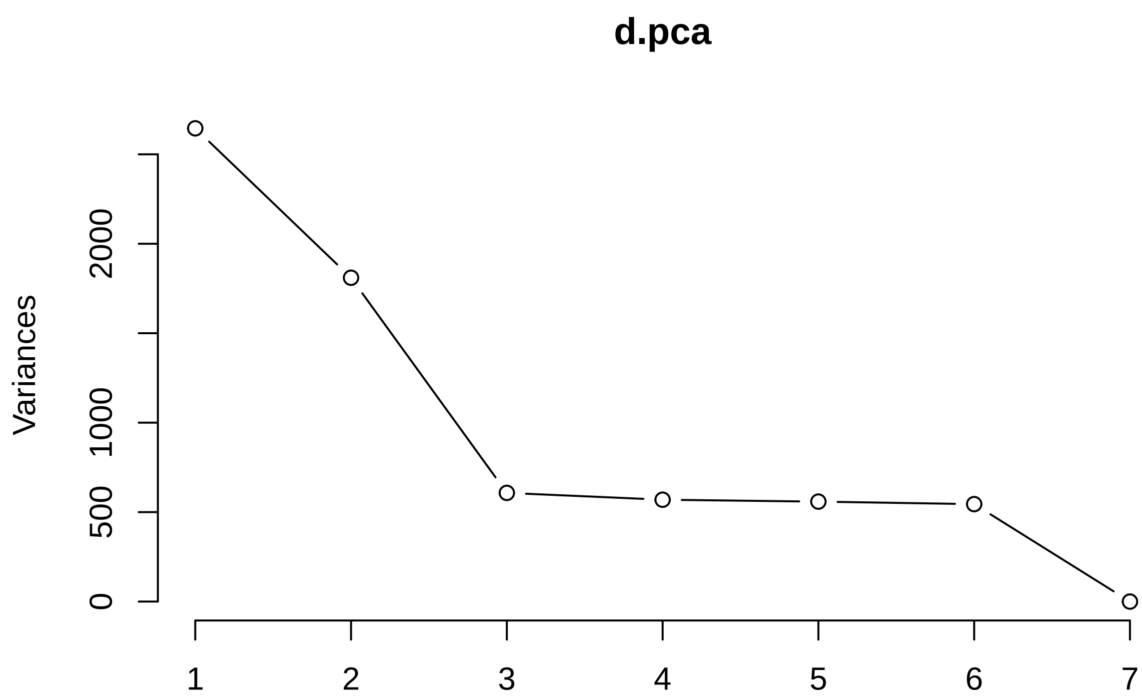
```
d <- read.table("../data/pnas_expression.txt",header=T)
rownames(d) <- d$ensembl_ID
d <- d[,-c(1,9)]
d <- d+1
log.d <- log(d)
```

```r
#We would like to cluster the samples
# Samples are rows
log.d.t <- t(log.d)

d.pca <- prcomp(log.d.t)

#head(print(d.pca))

#scree plot
plot(d.pca,type="l")
```



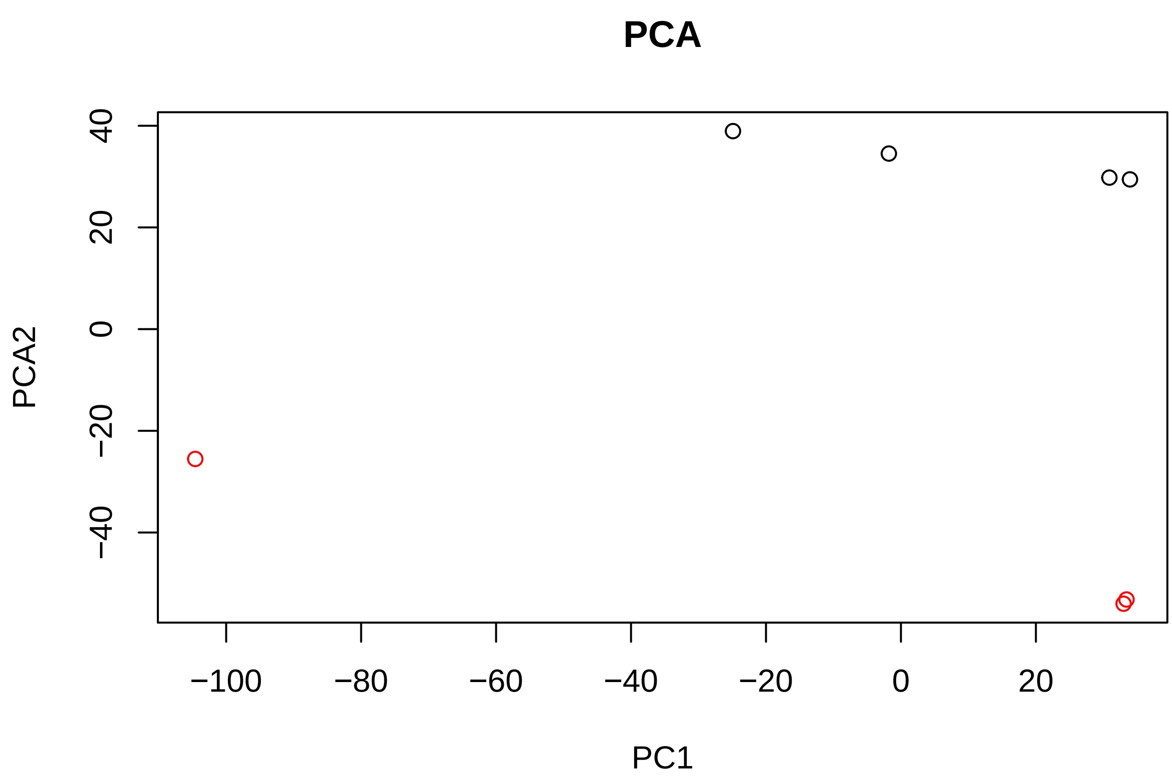

```r
summary(d.pca)
```

```
## Importance of components:
##                            PC1     PC2      PC3      PC4      PC5      PC6
## Standard deviation     51.4324 42.5448 24.64697 23.86156 23.63660 23.33782
## Proportion of Variance  0.3927  0.2687  0.09019  0.08453  0.08295  0.08086
## Cumulative Proportion   0.3927  0.6615  0.75166  0.83619  0.91914  1.00000
##                            PC7
## Standard deviation     3.538e-13
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

```
# Eigenvalues
d.pca$sdev^2
```

```
## [1] 2.645288e+03 1.810059e+03 6.074733e+02 5.693739e+02 5.586890e+02
## [6] 5.446539e+02 1.251782e-25
```

```
plot(d.pca$x[,1], d.pca$x[,2],col=coldata$condition, main="PCA",xlab="PC1",ylab="PCA2")
```



9 Gene set enrichment analysis

Consider a bag full of marbles (total N) containing K green marbles and $(N-K)$ red marbles. If we draw a sample of n from this bag, the probablity of getting exactly k green marble is given by *Hypergeometric distribution*.

$$Prob(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

This is exactly the scenario of finding a pathway or gene-ontology hits with differentially expressed gene sets. In this case:

$$N = \text{total number of genes in the genome}$$
$$n = \text{number of differentially expressed genes (degs) with significant p-value}$$
$$K = \text{Number of genes in a pathway or ontology}$$
$$k = \text{overlap between degs and that pathway}$$

To calculate this probability in `R` we need to calcualate the cumulative distribution of $(overlap >= k)$.

```
phyper(q=k, m=K, n=N-K, k=n, lower.tail = FALSE)
```

Bibliography

Dillies,M.-A. *et al.* (2013) A comprehensive evaluation of normalization methods for illumina high-throughput RNA sequencing data analysis. *Brief Bioinform*, **14**, 671–683.

Li,B. and Dewey,C.N. (2011) RSEM: Accurate transcript quantification from RNA-seq data with or without a reference genome. *BMC Bioinformatics*, **12**, 323.

Mortazavi,A. *et al.* (2008) Mapping and quantifying mammalian transcriptomes by RNA-seq. *Nat. Methods*, **5**, 621–628.