Introduction to RNASeq

Malay (malay@uab.edu)

November 18, 2016

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:

http://cmb.path.uab.edu/training/docs/CB2-201-2015/rnaseq_data.tar.gz

Unzip the file.

4 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
```

5 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.

5.1 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`
```

5.2 Prepare reference

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

5.3 Calculate expression directly from STAR output

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

5.4 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
```

5.5 Differential expression

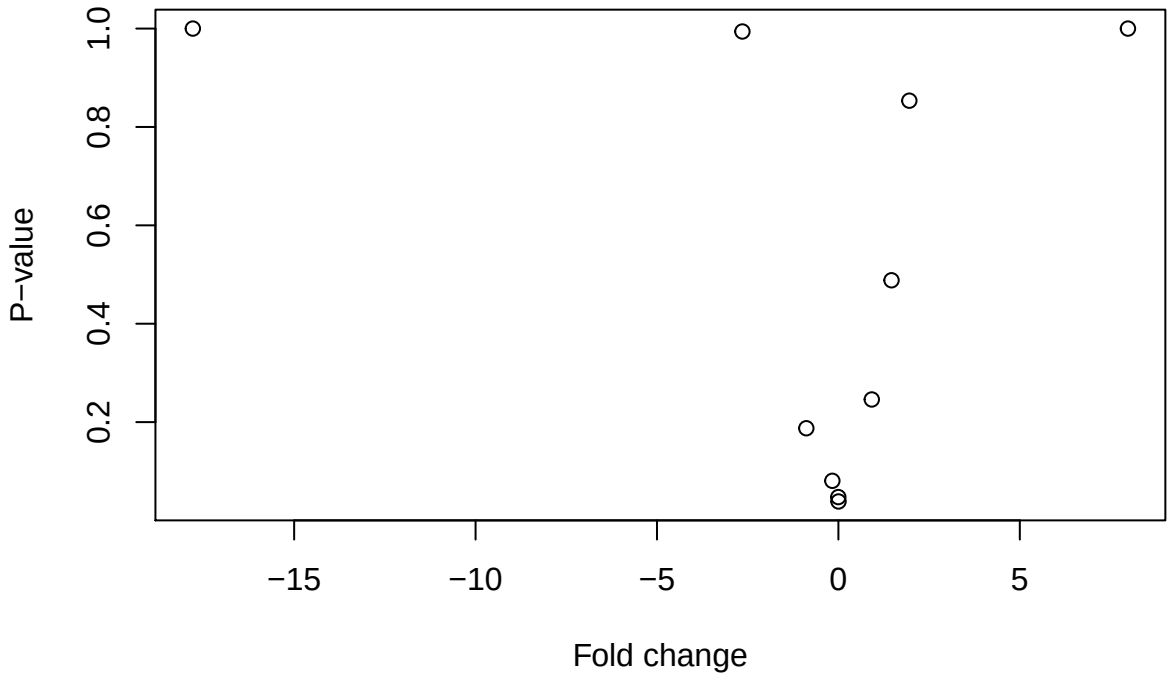
```
rsem-generate-data-matrix adrenal_chr19.genes.results human_chr19.genes.results \
  >diff-brain-adrenal.txt
rsem-run-ebseq diff-brain-adrenal.txt 1,1 expression.results.txt
rsem-control-fdr expression.results.txt 0.05 expression_final.txt
```

And we have our differentially expressed genes.

5.6 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")
```



6 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)
```

7 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       stat
##                    <numeric>      <numeric> <numeric>  <numeric>
## ENSG00000124208  529.427261    -0.45551872 0.1398108 -3.2581080
## ENSG00000182463   31.223142     0.67452218 0.3389832  1.9898397
## ENSG00000124201  225.128033    -0.08164487 0.1567135 -0.5209816
## ENSG00000124205    1.181384    -3.53222385 2.4790556 -1.4248264
## ENSG00000124207   73.835473    -0.44301600 0.2241038 -1.9768337
## ...                     ...            ...       ...        ...
## ENSG00000218597   7.5966111     -0.7611109 0.6765486 -1.1249907
## ENSG00000217348  17.9021532      0.1084396 0.4722588  0.2296190
## ENSG00000217342   0.1160766     -0.7044195 3.8169465 -0.1845505
## ENSG00000216298   0.1805546     -0.7044195 3.8169465 -0.1845505
## ENSG00000183878 137.6118215      0.4820682 0.2047983  2.3538678
##                       pvalue        padj
##                    <numeric>   <numeric>
## ENSG00000124208 0.001121577 0.007766312
## ENSG00000182463 0.046608601 0.158635820
## ENSG00000124201 0.602379559 0.801253970
## ENSG00000124205 0.154207387          NA
## ENSG00000124207 0.048060426 0.162350279
## ...                     ...         ...
## ENSG00000218597  0.26059298  0.50400325
## ENSG00000217348  0.81838781  0.92404925
## ENSG00000217342  0.85358158          NA
## ENSG00000216298  0.85358158          NA
## ENSG00000183878  0.01857922  0.07838977
```

Sort results based on p-value.

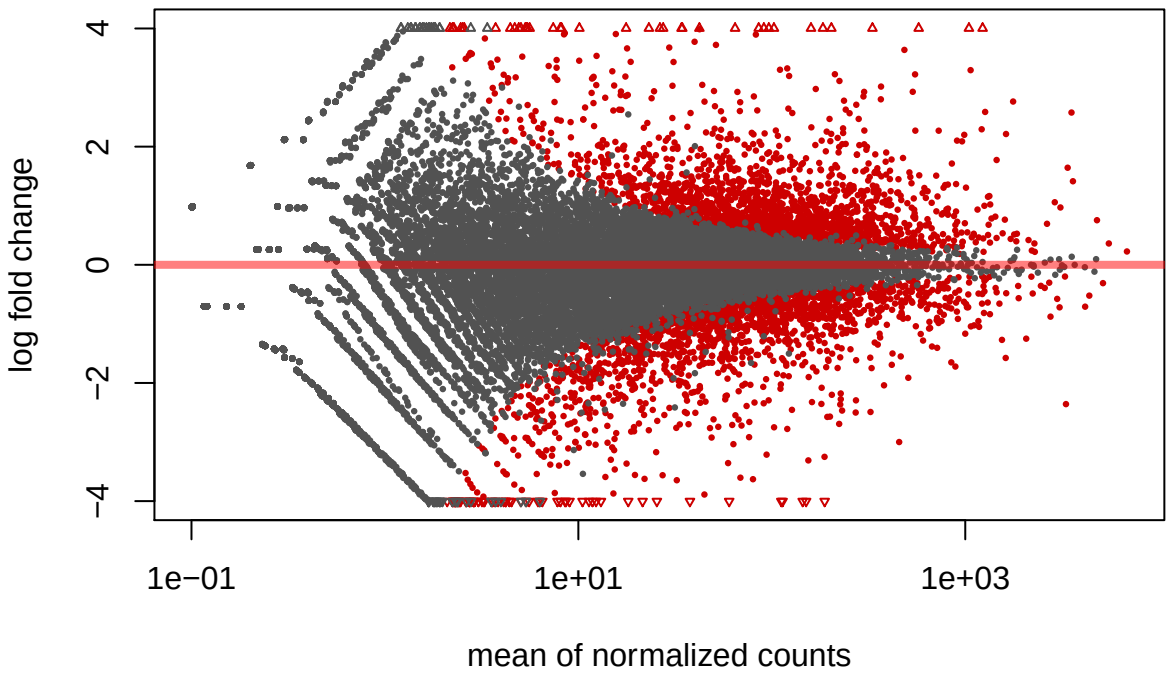
```
results<- results[order(results$padj),]
results
```

```
## 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       stat
```

```
##                 <numeric>      <numeric>  <numeric>   <numeric>
## ENSG00000115648 3541.8337       2.577108 0.06053668    42.57101
## ENSG00000096060 1229.6845       4.983040 0.10245352    48.63708
## ENSG00000151503 1044.9213       5.797471 0.12828712    45.19138
## ENSG00000162772 1065.7118       3.294151 0.08747217    37.65942
## ENSG00000166451  574.4599       4.665525 0.14184300    32.89218
## ...                   ...            ...        ...         ...
## ENSG00000129864 0.3319929     -1.5862848   3.712179 -0.42731901
## ENSG00000217720 0.1805546     -0.7044195   3.816946 -0.18455054
## ENSG00000218917 0.2215218      0.2609164   3.816114  0.06837227
## ENSG00000217342 0.1160766     -0.7044195   3.816946 -0.18455054
## ENSG00000216298 0.1805546     -0.7044195   3.816946 -0.18455054
##                       pvalue          padj
##                    <numeric>     <numeric>
## ENSG00000115648  0.000000e+00  0.000000e+00
## ENSG00000096060  0.000000e+00  0.000000e+00
## ENSG00000151503  0.000000e+00  0.000000e+00
## ENSG00000162772  0.000000e+00  0.000000e+00
## ENSG00000166451 2.843651e-237 8.340997e-234
## ...                       ...           ...
## ENSG00000129864     0.6691470            NA
## ENSG00000217720     0.8535816            NA
## ENSG00000218917     0.9454893            NA
## ENSG00000217342     0.8535816            NA
## ENSG00000216298     0.8535816            NA
```

7.1 MA-plot

```
plotMA(results)
```



7.2 Principal component

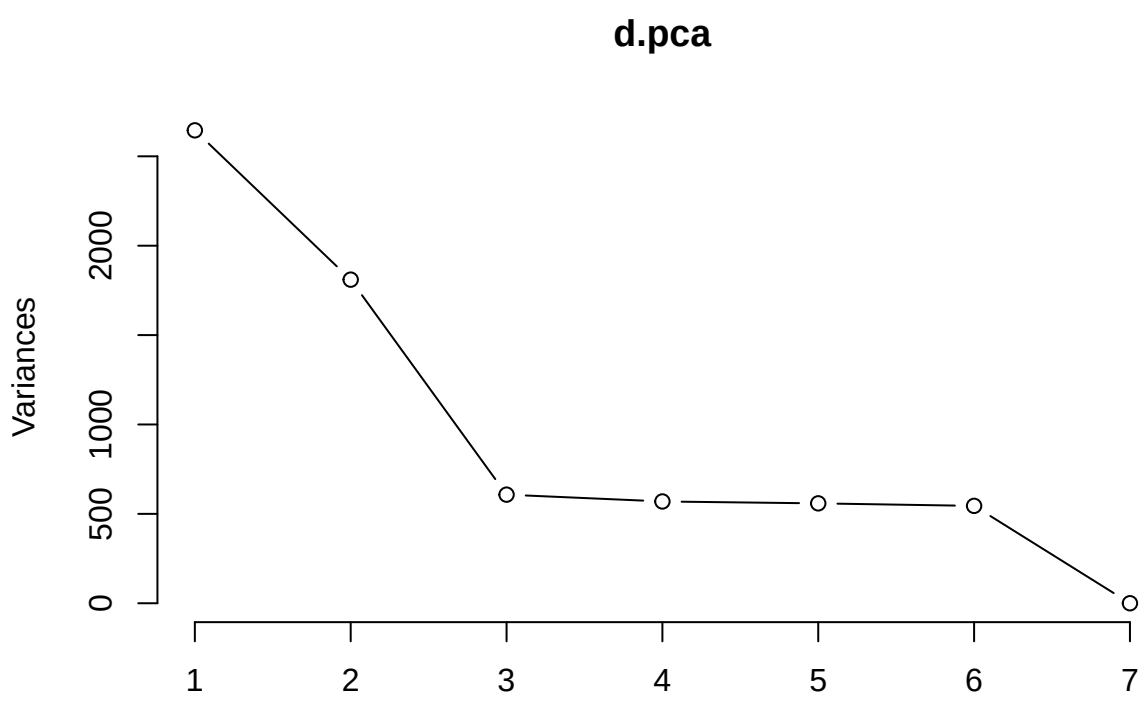
```
d <- read.table("../data/pnas_expression.txt",header=T)
rownames(d) <- d$ensembl_ID
d <- d[,-c(1,9)]
d <- d+1
log.d <- log(d)

#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")
```



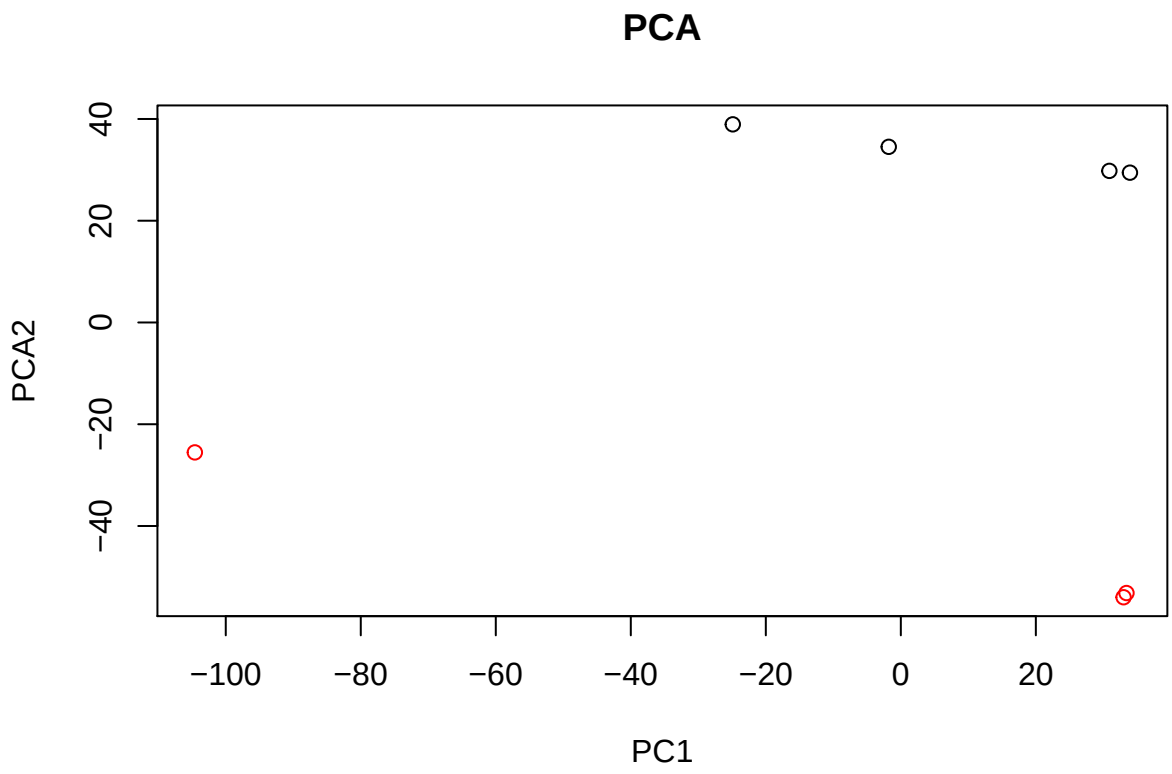

```
summary(d.pca)
```

```
## Importance of components%s:
##                             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=as.factor(coldata), main="PCA",xlab="PC1",ylab="PCA2")
```



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.