Differential Expression Analysis using *edgeR* and *DESeq2*

Bernard Pereira & Oscar Rueda

November 26, 2015

Contents

1 Introduction

In this tutorial, we will be using *edgeR*[2] and *DESeq2*[3] to analyse some RNA-seq data taken from [1].

You can find out more about *edgeR* and *DESeq2* from the:

- EdgeR paper [2]
- DESeq paper [3]
- Bioconductor website, which hosts vignettes for both packages

While you carry out the exercises, try to focus on what each function is doing in terms of the overall RNA-seq workflow, and look for similarities and differences between the *edgeR* and *DESeq2* implementations.

1.1 The Data

We will be using data downloaded from GEO (GSE29968)[1]. Here is the authors' summary of the data from the GEO website (`http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE29968`):

We performed the integrative transcriptome analysis of human esophageal squamous cell carcinoma (ESCC) using Illumina high-throughput sequencing. A total of 187 million 38bp sequencing reads were generated containing 7 billion bases for three pairs of matched patient-derived ESCC clinical specimens and their adjacent non-tumorous tissues. By investigating the digital gene expression profiling, we found 1425 genes significantly differentially expressed and detected more than 9000 SNPs across all six samples. We also identified protein tyrosine kinase 6 (PTK6) as a novel tumor suppressor gene, which is critical in ESCC development.

In this practical, we will use the counts generated with *Rsubread* to find genes that are differentially expressed between the tumours and normal samples.

2 edgeR Workflow

The *edgeR* workflow consists of:

1. **Data Input** - *edgeR* requires a matrix of raw read counts. Note that, like many other Bioconductor packages, *edgeR* uses a container object (DGEList) to store and process the data.
2. **Filtering** - Genes with low counts across all samples are unlikely to appear on the final list of differentially expressed genes, but represent additional hypothesis tests that will need to be taken into account when performing multiple testing corrections. Consequently, these genes can be removed before we carry out any further analyses.
3. **Normalisation** - We need to ensure that read counts across samples are actually comparable before carrying out any differential expression analyses. The trimmed mean of M-values (TMM) method *edgeR* uses is based on the assumption that most genes are not differentially expressed, and uses a correction factor based on the weighted mean of the log ratios of test/reference (this should be close to 1).
4. **Model specification** -We need to construct our model based on the experimental setup we have. This will be described using a design matrix, through which we can specify the parameters we wish to consider.
5. **Variance estimation** - *edgeR* fits the read counts to a negative binomial model, which can be thought of as being an overdispersed Poisson distribution. The package uses an empirical Bayes method to estimate the dispersion parameters for the genes.
6. **Differential expression testing** - Once the dispersion parameters have been estimated, we can proceed to test for differential expression. When we use the generalised linear models (GLM) framework, we can use contrasts to specify the questions of interest.

2.1 Reading in the Data

We first need to load the required library and data required for this practical. You may use the file you generated previously, or the set of read counts in Day2/Counts.RData.

Note that the genes in this file are identified by their Entrez gene ids.

```r
library(edgeR)
load("Day2/Counts.RData")       #Load data
Counts <- tmp$counts
colnames(Counts) <- c("16N", "16T", "18N", "18T", "19N", "19T") #Rename the columns

#Explore the counts matrix
dim(Counts)
head(Counts)
```

2.2 Creating a DGEList object

We will now create a DGEList object to hold our read counts. This object is a container for the counts themsleves, and also for all the associated metadata - these include, for example, sample names, gene names and normalisation factors once these are computed. The DGEList is an example of the custom task-specific structures that are frequently used in Bioconductor to make analyses easier.

```
dgList <- DGEList(counts=Counts, genes=rownames(Counts))
```

```
dgList
dgList$samples
head(dgList$counts)      #Many rows!
head(dgList$genes)       #Likewise!
```

2.3 Filtering

There are approximately 26000 genes in this dataset. However, many of them will not be expressed, or will not be represented by enough reads to contribute to the analysis. Removing these genes means that we have ultimately have fewer tests to perform, thereby reducing the problems associated with multiple testing. The *edgeR* vignette states that from a "biological point of view, a gene must be expressed at some minimal level before it is likely to be translated into a protein or to be biologically important".

Here, we retain only those genes that are represented at least 1cpm reads in at least two samples (cpm=counts per million).

```
countsPerMillion <- cpm(dgList)
summary(countsPerMillion)
#'summary' is a useful function for exploring numeric data; eg. summary(1:100)
```

```
countCheck <- countsPerMillion > 1
head(countCheck)

keep <- which(rowSums(countCheck) >= 2)
dgList <- dgList[keep,] #How many genes do we have left?
summary(cpm(dgList))     #Compare this to the original summary
```

2.4 Normalisation

As we have discussed, it is important to normalise RNA-seq data. *edgeR* implements the TMM method. As the vignette describes, *edgeR* is mostly interested in correcting for sample-specific effects (most importantly, for library size and library composition). There is less attention paid to factors such as gene length and GC content, as one can assume these have the same effects across all samples.

Note that *edgeR* does not actually transform the read counts themselves. The normalisation factors are used as correction factors that *edgeR* uses when building the statistical model.

```
dgList$samples #Look at the 'norm.factors' column
head(dgList$counts)

?calcNormFactors         #How does the function work?
dgList <- calcNormFactors(dgList, method="TMM")

dgList$samples #Look at the 'norm.factors' column
head(dgList$counts) #Counts have not changed
```

2.5 Data Exploration

We can examine inter-sample relationships by producing a plot based on mutlidimensional scaling.

```
plotMDS(dgList)
```

2.6 Setting up the Model

We are now ready to set up the model! We first need to specify our design matrix, which describes the setup of the experiment.

```
sampleType<- rep("N", ncol(dgList))      #N=normal; T=tumour
sampleType[grep("T", colnames(dgList))] <- "T"
#'grep' is a string matching function.

sampleReplicate <- paste("S", rep(1:3, each=2), sep="")
```

```
designMat <- model.matrix(~sampleReplicate + sampleType)
designMat #What is used as the intercept?  What parameters are we interested in?
```

2.7 Estimating Dispersions

As disucssed, we need to estimate the dispersion parameter for our negative binomial model. Some strategies include:

- Using a common estimate across all genes
- Fitting an estimate based on the mean-variance trend across the dataset, such that genes similar abundances have similar variance estimates (trended dispersion).
- Computing a genewise dispersion (tagwise dispersion)

It is somewhat naive to assume that genes share a common dispersion parameter, or that they share the same mean-variance relationships. However, as there are only a few replicates, it is difficult to estimate the dispersion accurately for each gene. *edgeR*, we use an empirical Bayes method to 'shrink' the genewise dispersion estimates towards the common dispersion (tagwise dispersion) as a method to 'share' information between genes.

Note that either the common or trended dispersion needs to be estimated before we can estimate the tagwise dispersion.

```
dgList <- estimateGLMCommonDisp(dgList, design=designMat)
dgList <- estimateGLMTrendedDisp(dgList, design=designMat)
dgList <- estimateGLMTagwiseDisp(dgList, design=designMat)
```

We can plot the estimates and see how they differ. The biological coefficient of variation (BCV) is the square root of the dispersion parameter in the negative binomial model.

```
plotBCV(dgList)
```

2.8 Differential Expression

We can now find our differentially expressed genes. After fitting the model, we can use the topTags() function to explore the results, and set theresholds to identify subsets of differentially expressed genes.

```
fit <- glmFit(dgList, designMat)
lrt <- glmLRT(fit, coef=4)

#The glmLRT function also (see ?glmLRT - especially the description for 'coef' and 'contrast')
#accepts a contrasts vector.  Can you work out  and use the contrast to use
```

```
#to obtain the same result
#(ie. differentially expressed genes between tumour and normal samples)?
?glmLRT
```

```
edgeR_result <- topTags(lrt)
?topTags
resultToSave <- topTags(lrt,n=15000)$table

save(resultToSave, file='Day2/edgeR_Result.RData')      #We will need this later
```

Finally, we can plot the log-fold changes of all the genes, and the highlight those that are differentially expressed.

```
?decideTests
deGenes <- decideTestsDGE(lrt, p=0.001) #Can play around with these options
deGenes <- rownames(lrt)[as.logical(deGenes)]
plotSmear(lrt, de.tags=deGenes)
abline(h=c(-1, 1), col=2)
```

3 DESeq2 Workflow

We will now perform the same analysis using *DESeq2*. This will hopefully both reinforce the fundamental principles of differential expression analysis that most packages follow, and also provide an idea of the different approaches used to implement these. The workflow is as follows:

1. **Data Input** - *DESeq2* also has its own structures for storing and processing data.
2. **Filtering** - Like *edgeR*, *DESeq2* also filters genes that are unlikely to be significantly differentially expressed. However, *DESeq2* attempts to remove these genes using thresholds derived from the data itself
3. **Normalisation** - *DESeq2* also performs normalisation under the assumption that most genes are not differentially expressed, and that the median ratio of the read counts in a sample to the read counts in a reference should be around 1.
4. **Model specification** -In *DESeq2*, the model is specified when the data object is constructed.
5. **Variance estimation** - *DESeq2* follows a similar procedure to *edgeR*, even if the precise details differ. After estimating a per-gene and trended dispersion, the package uses an empirical Bayes method to shrink the per-gene estimates towards the trended estimate.
6. **Differential expression testing** - Differential expression testing is performed with a Wald test. We can switch off independent filtering at this stage.

3.1 Creating a DESeqDataSet

The data object used by *DESeq2* comprises the counts themselves, sample metadata, and a design formula. Like the design matrix specification in *edgeR*, the design formula essentially defines the experimental setup. In *DESeq2*, we specify the design at the beginning, although this can be changed later.

```
library(DESeq2)
load("Day2/Counts.RData")       #Load data
Counts <- tmp$counts
colnames(Counts) <- c("16N", "16T", "18N", "18T", "19N", "19T") #Rename the columns

Coldata <- data.frame(sampleReplicate=c("16", "16", "18", "18", "19", "19"),
sampleType=c("N", "T", "N", "T", "N", "T"))
rownames(Coldata) <- c("16N", "16T", "18N", "18T", "19N", "19T")

deSeqData <- DESeqDataSetFromMatrix(countData=Counts, colData=Coldata,
design= ~sampleReplicate + sampleType)
deSeqData
```

3.2 Filtering

DESeq2 distinguishes between two types of filtering: *pre-filtering* is used to remove genes with 'no or nearly no reads', and is performed to save memory and speed up computations, whereas *independent filtering* is used to reduce the number of hypothesis tests, and is applied by **default** (although it can be switched off).

```
nrow(counts(deSeqData))
summary(rowSums(counts(deSeqData)))

deSeqData <- deSeqData[rowSums(counts(deSeqData))>1,]

nrow(counts(deSeqData))
summary(rowSums(counts(deSeqData)))
```

DESeq2 uses the mean of the normalised counts to perform independent filtering (genes with mean normalised counts below a specific threshold are removed), and attempts to calculate an optimal threshold based on the numbers of adjusted *p*-values that fall above the significance threshold.

3.3 Normalisation

As we discussed, *DESeq2* calculates scaling factors for each sample by taking the median ratio of the read counts for each gene in a sample to the geometric mean of each gene across all other samples. Note that, as in *edgeR* the raw read counts are not directly transformed, although we can see how they change.

```
deSeqData <- estimateSizeFactors(deSeqData)
colData(deSeqData)

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

3.4 Estimating Dispersions

The process by which *DESeq2* obtains dispersion estimates is similar to that used by *edgeR*, in that both packages uses an empirical Bayes method to shrink per-gene dispersion estimates towards a trended estimate. The exact details, however, differ between the two packages.

Estimation of the dispersion parameter in *DESeq2* comprises three steps:

1. Estimate a dispersion value for each gene.
2. Fits a curve through the estimates (obtaining a curve of the mean-variance relationship.
3. Shrink the per-gene estimates to the trended estimates using an empirical Bayes method.

To understand this procedure better, we can visualise the results of these three steps using the function plotDispEsts. The per-gene estimates circled in blue are considered to be dispersion outliers.

```
deSeqData <- estimateDispersions(deSeqData)
mcols(deSeqData)

plotDispEsts(deSeqData)
```

3.5 Differential Expression

Finally, we can perform differential expression tests using our model specification and dispersion estimates.

```
deSeqData <- nbinomWaldTest(deSeqData)
res <- results(deSeqData)
#Lots of options for reporting results eg. significance threshold,
#multiple testing correction method etc
res     #What comparison is being made?
summary(res)

#As in edgeR, we can also use contrasts to specify our tests of interest.
#Can you work out the contrast vector that gives the same result?
?results
?resultsNames

plotMA(res, main="MA Plot")
```

References

[1] Ma, S. et al. (2012) *Identification of PTK6, via RNA Sequencing Analysis, as a Suppressor of Esophageal Squamous Cell Carcinoma*, Gastroenterology, 143 (3) 675-686.

[2] Robinson, MD.et al. (2010) *edgeR: a Bioconductor package for differential expression analysis of digital gene expression data*, Bioinformatics, 26 (1) 139-140.

[3] Love, MI.et al. (2014) *Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2*, Genome Biology, 15(12) 550.