edgeR Tutorial: Differential Expression in RNA-Seq Data

by Sean Ruddy
September 26, 2011

1 Introduction

1.1 Disclaimer

Use at your own risk. Some errors may exist and so it is imperative that you understand each line of code and what it's doing with respect to your experimental needs before copying and pasting line for line.

1.2 What is edgeR?

edgeR is a package written for [R] that performs differential tag/gene expression using count data under a negative binomial model. The methods used in edgeR do NOT support FPKM, RPKM or other types of data that are not counts.

You can obtain more information about edgeR from the bioconductor website `http://www.bioconductor.org/packages/2.8/bioc/html/edgeR.html` . Here you'll find a user's guide/vignette with detailed examples, a reference manual for all edgeR functions, and the [R] code for installing the package. A useful discussion on a forum about edgeR can be found at `http://seqanswers.com/forums/showthread.php?t=5591` .

1.3 Getting Help with edgeR

The bioconductor website provides access to a mailing list that all can subscribe to. Any questions concerning any bioconductor package can be sent to this mailing list. They are very responsive and very helpful; I have used it several times. To subscribe go to `https://stat.ethz.ch/mailman/listinfo/bioconductor` . For guides to posting questions go to `http://www.bioconductor.org/help/mailing-list/posting-guide/` .

If you just need help understanding an edgeR function or want to see an example of how it's used simply type **?functionname** into the [R] command line. This will bring up a text page with detailed information about the function.

1.4 Getting Help with [R]

Google your question! Write a short description and add on at the end "R-help". This will most likely bring up a thread in the R-help forum that deals with the same or similar problem. Almost every problem you could have can be found answered on this forum. You can also subscribe to the [R] mailing list by going to `https://stat.ethz.ch/mailman/listinfo/r-help`. A guide for posting questions can be found at `http://www.r-project.org/posting-guide.html`

Elizabeth Purdom has several [R] tutorials that can be found at `http://www.stat.berkeley.edu/~epurdom/`. In particular, see `http://www.stat.berkeley.edu/~epurdom/Saving/Saving.html` for using directories and saving objects and `http://www.stat.berkeley.edu/~epurdom/BrownLab/` for creating and manipulating data sets in [R].

1.5 Installing edgeR

With an internet connection, edgeR can be easily installed by running the following commands in [R]

```
source("http://www.bioconductor.org/biocLite.R")
biocLite("edgeR")
```

2 A Straightforward RNA-Seq Differential Expression Analysis using edgeR

2.1 Description

This is an RNA-Seq data set from an article by Li et al. [2008] titled, "Determination of tag density required for digital transcriptome analysis: Application to an androgen-sensitive prostate cancer model." The study compared hormone treated cancer cells to non-treated cancer cells. For the non-treated, there are 4 biological replicates. For the treated, there are 3 biological replicates. A full analysis of this data set can also be found in section 10 of the edgeR vignette.

2.2 Setting Up the Count Matrix

First, we load the egdeR package into [R] and set the directory to the folder where the data is contained. If using Windows "\\" is needed between directories; if using Unix "/" is sufficient.

```
library(edgeR)
setwd("C:\\Documents and Settings\\All Users\\Documents\\Shared Files\\School\\Seminars\\RNA-Seq Workshop\\Data")

# From the server
setwd("/global/courses/rnaseqworkshop/web/RNA_SEQ")
```

Reading in the data,

```
raw.data <- read.table( file = "pnas_expression.txt" , header = TRUE )
head( raw.data )
```

edgeR requires the the dataset to contain only the counts with the row names as the gene ids and the column names as the sample ids:

```
counts <- raw.data[ , -c(1,ncol(raw.data)) ]
 rownames( counts ) <- raw.data[ , 1 ] # gene names
 colnames( counts ) <- paste(c(rep("C_R",4),rep("T_R",3)),c(1:4,1:3),sep="") # sample names
head( counts )
```

Some quick summaries:

```
dim( counts )
colSums( counts ) # Library Sizes
colSums( counts ) / 1e06 # Library Sizes in millions of reads
table( rowSums( counts ) )[ 1:30 ] # Number of genes with low counts
```

2.3 Building the edgeR Object

DGEList() is the function that coverts the count matrix into an edgeR object. First, we create a group variable that tells edgeR with samples belong to which group and supply that to DGEList in addition to the count matrix. We can then see the elements that the object contains by using the names() function. These elements can be accessed using the $ symbol:

```
group <- c(rep("C", 4) , rep("T", 3))
cds <- DGEList( counts , group = group )
 names( cds )

head(cds$counts) # original count matrix
cds$samples # contains a summary of your samples
sum( cds$all.zeros ) # How many genes have 0 counts across all samples

cds # or type the name of the object
```

We need to filter out low count reads since it would be impossible to detect differential expression. The method used in the edgeR vignette is to keep only those genes that have at least 1 read per million in at least 3 samples. Once this filtering is done, we can calculate the normalization factors which correct for the different compositions of the samples. The effective library sizes are then the product of the actual library sizes and these factors.

```
cds <- cds[rowSums(1e+06 * cds$counts/expandAsMatrix(cds$samples$lib.size, dim(cds)) > 1) >= 3, ]
dim( cds )

cds <- calcNormFactors( cds )
cds$samples

# effective library sizes
cds$samples$lib.size * cds$samples$norm.factors
```

2.4 Multi-Dimensional Scaling Plot

An MDS plot measures the similarity of the samples and projects this measure into 2-dimensions. When creating plots in R, you can also save plots as pdf's, png's, etc. (see *?pdf*).

```
# To view the plot immediately
plotMDS.dge( cds , main = "MDS Plot for Count Data", labels = colnames( cds$counts ) )

# Output plot as a pdf
pdf( "MDS_plot_1_ex1.pdf" , width = 7 , height = 7 ) # in inches
 plotMDS.dge( cds , main = "MDS Plot for Count Data", labels = colnames( cds$counts ) )
dev.off() # this tells [R] to close and stop writing to the pdf.
```

2.5 Estimating Dispersions

The first dispersion type to calculate is the common dispersion. In the common dispersion setting, each gene gets assigned the same dispersion estimate. The output of the estimation will include the estimate as well as some other elements added to the edgeR object, cds.

```
cds <- estimateCommonDisp( cds )
 names( cds )

# The estimate
cds$common.dispersion
```

To understand what this value means, recall the parameterization for the variance of the negative binomial is $\nu(\mu) = \mu + \mu^2 \cdot \phi$. For poisson it's $\nu(\mu) = \mu$. The implied standard deviations are the square-roots of the variances. Now, suppose a gene had an average count of 200. Then the sd's under the two models would be,

```
sqrt( 200 ) # poisson sd
sqrt( 200 + 200^2 * cds$common.dispersion ) # negative binomial sd
sqrt( 200 + 200^2 * cds$common.dispersion ) / sqrt( 200 ) # NB sd is over 2 times larger
```

Once the common dispersion is estimated we can estimate the tagwise dispersions. In this scenario, each gene will get its own unique dispersion estimate, but the common dispersion is still used in the calculation. The tagwise dispersions are squeezed toward the common value. The amount of squeezing is governed by the paramter *prior.n*. The higher prior.n, the closer the estimates will be to the common dispersion. The recommended value is the nearest integer to $50/(\#samples - \#groups)$. For this data set that's $50/(7-2) = 10$.

```
# Default Setting
cds <- estimateTagwiseDisp( cds , prior.n = 10 )
 names( cds )
 summary( cds$tagwise.dispersion )

# More shrinkage/sqeezing toward the common
cds <- estimateTagwiseDisp( cds , prior.n = 25 )
 summary( cds$tagwise.dispersion ) # not much changed, but the ends got squeezed in quite a bit.

# The recommended setting for this data set is the default of 10. Let's stick with that.
cds <- estimateTagwiseDisp( cds , prior.n = 10 )
```

2.6 Mean-Variance Plot

Now that we've estimated the dispersion parameters we can see how well they fit the data by plotting the mean-variance relationship. Four things are shown in the plot: the raw variances of the counts (grey dots), the variances using the tagwise dispersions (light blue dots), the variances using the common dispersion (solid blue line), and the $variance = mean$ a.k.a. poisson variance (solid black line). The plot function outputs the variances which will be stored in the data set meanVarPlot.

```
meanVarPlot <- plotMeanVar( cds , show.raw.vars=TRUE ,
                                  show.tagwise.vars=TRUE ,
                                  show.binned.common.disp.vars=FALSE ,
                                  show.ave.raw.vars=FALSE ,
                                  dispersion.method = "qcml" , NBline = TRUE ,
                                  nbins = 100 ,
                                  pch = 16 ,
                                  xlab ="Mean Expression (Log10 Scale)" ,
                                  ylab = "Variance (Log10 Scale)" ,
                                  main = "Mean-Variance Plot" )
```

2.7 Testing

The function exactTest() performs pair-wise tests for differential expression between two groups. The important parameters are *common.disp* which if true uses the common dispersion estimate for testing and if false uses the tagwise dispersion estimates; and *pair* which indicates which two groups should be compared. The output of exactTest() is a list of elements, one of which is a table of the results. The third line of code uses the dispersion parameter to set a dispersion of zero, ie. the poisson model to which we can compare the negative binomial results.

```
de.cmn <- exactTest( cds , common.disp = TRUE , pair = c( "C" , "T" ) )
de.tgw <- exactTest( cds , common.disp = FALSE , pair = c( "C" , "T" ) )
de.poi <- exactTest( cds , dispersion = 1e-06  , pair = c( "C" , "T" ) )

names( de.tgw )
de.tgw$comparison # which groups have been compared
head( de.tgw$table ) # results table in order of your count matrix.
head( cds$counts )
```

The table from exactTest() does not contain p-values adjusted for multiple testing. The function *topTags()* takes the output from exactTest(), adjusts the raw p-values using the False Discovery Rate (FDR) correction, and returns the top differentially expressed genes. The output is similar to that of exactTest() but with a column of adjusted p-values and sorted by increasing p-value. The *sort.by* argument allows you to sort the table by p-value, concentration or fold-change if desired. We can also use topTags() to return the original counts of the top differentially expressed genes. By setting the n parameter to the total number of genes, we can save the entire topTags() results table.

```
# Top tags for tagwise analysis
options( digits = 3 ) # print only 3 digits
topTags( de.tgw , n = 20 , sort.by = "p.value" ) # top 20 DE genes

# Back to count matrix for tagwise analysis
cds$counts[ rownames( topTags( de.tgw , n = 15 )$table ) , ]


# Sort tagwise results by Fold-Change instead of p-value
resultsByFC.tgw <- topTags( de.tgw , n = nrow( de.tgw$table ) , sort.by = "logFC" )$table
 head( resultsByFC.tgw )


# Store full topTags results table
resultsTbl.cmn <- topTags( de.cmn , n = nrow( de.cmn$table ) )$table
resultsTbl.tgw <- topTags( de.tgw , n = nrow( de.tgw$table ) )$table
resultsTbl.poi <- topTags( de.poi , n = nrow( de.poi$table ) )$table
 head( resultsTbl.tgw )
```

Now that we have the full results table with the adjusted p-values we can compare those p-values to the significance level and see how many differentially expressed genes we find. Here we use a significance level of 0.05. In addition, we can use the function *decideTestsDGE()* to determine how many DE genes are up or down regulated compared to control.

```
# Names/IDs of DE genes
de.genes.cmn <- rownames( resultsTbl.cmn )[ resultsTbl.cmn$adj.P.Val <= 0.05 ]
de.genes.tgw <- rownames( resultsTbl.tgw )[ resultsTbl.tgw$adj.P.Val <= 0.05 ]
de.genes.poi <- rownames( resultsTbl.poi )[ resultsTbl.poi$adj.P.Val <= 0.05 ]

# Amount significant
length( de.genes.cmn )
length( de.genes.tgw )
length( de.genes.poi )

# Percentage of total genes
length( de.genes.cmn ) / nrow( resultsTbl.cmn ) * 100
length( de.genes.tgw ) / nrow( resultsTbl.tgw ) * 100
length( de.genes.poi ) / nrow( resultsTbl.poi ) * 100

# Up/Down regulated summary for tagwise results
summary( decideTestsDGE( de.tgw , p.value = 0.05 ) ) # the adjusted p-values are used here
```

We can also can also compare the analyses to see how many DE gene are in common.

```
sum( de.genes.tgw %in% de.genes.cmn ) / length( de.genes.tgw ) * 100 # Tagwise to Common
sum( de.genes.cmn %in% de.genes.tgw ) / length( de.genes.cmn ) * 100 # Common to Tagwise
sum( de.genes.tgw %in% de.genes.poi ) / length( de.genes.tgw ) * 100 # Tagwise to Poisson

# Percent shared out of top 10, 100 & 1000 between tagwise and common
sum( de.genes.tgw[1:10] %in% de.genes.cmn[1:10] ) / 10 * 100
sum( de.genes.tgw[1:100] %in% de.genes.cmn[1:100] )
sum( de.genes.tgw[1:1000] %in% de.genes.cmn[1:1000] ) / 1000 * 100

# Percent shared out of top 10, 100 & 1000 between tagwise and poisson
sum( de.genes.tgw[1:10] %in% de.genes.poi[1:10] ) / 10 * 100
sum( de.genes.tgw[1:100] %in% de.genes.poi[1:100] )
sum( de.genes.tgw[1:1000] %in% de.genes.poi[1:1000] ) / 1000 * 100
```

2.8 Visualizing Results

The first thing we'll look at is the spread of expression levels for the top DE genes. Below, we plot a histogram of log concentrations for the top 100 genes for each analysis.

```
par( mfrow=c(3 ,1) )
 hist( resultsTbl.poi[de.genes.poi[1:100],"logConc"] , breaks=10 , xlab="Log Concentration" ,
       col="red" , xlim=c(-18,-6) , ylim=c(0,0.4) , freq=FALSE , main="Poisson: Top 100" )

 hist( resultsTbl.cmn[de.genes.cmn[1:100],"logConc"] , breaks=25 , xlab="Log Concentration" ,
       col="green" , xlim=c(-18,-6) , ylim=c(0,0.4) , freq=FALSE , main="Common: Top 100" )

 hist( resultsTbl.tgw[de.genes.tgw[1:100],"logConc"] , breaks=25 , xlab="Log Concentration" ,
       col="blue" , xlim=c(-18,-6) , ylim=c(0,0.4) , freq=FALSE , main="Tagwise: Top 100" )
par( mfrow=c(1,1) )
```

The next plot is an MA plot that shows the relationship between concentration and fold-change across the genes. The differentially expressed genes are colored red and the non-differentially expressed are colored black. The orange dots represent genes in which the counts were zero in all samples of one of the groups. Here, we just look at the tagwise and poisson analyses since the common analyses in this case is very similar. The blue line is added at a log-FC of 2 to represent a level for biological significance.

```
par( mfrow=c(2,1) )
 plotSmear( cds , de.tags=de.genes.poi , main="Poisson" ,
            pair = c("C","T") ,
            cex = .35 ,
            xlab="Log Concentration" , ylab="Log Fold-Change" )
  abline( h = c(-2, 2) , col = "dodgerblue" )

 plotSmear( cds , de.tags=de.genes.tgw , main="Tagwise" ,
            pair = c("C","T") ,
            cex = .35 ,
            xlab="Log Concentration" , ylab="Log Fold-Change" )
  abline( h=c(-2,2) , col="dodgerblue" )
par( mfrow=c(1,1) )
```

We can also see the difference in log concentrations between the DE genes under the negative binomial model and under the poisson model in the MA plot. Below, we plot only the top 500 DE genes and color the rest black. We can see that within the first 500 DE genes, the tagwise analysis puts more density toward the lower concentration values compared to the poisson.

```
par( mfrow = c(2,1) )
 plotSmear( cds , de.tags=de.genes.poi[1:500] , main="Poisson" ,
            pair=c("C","T") ,
            cex=.5 ,
            xlab="Log Concentration" , ylab="Log Fold-Change" )
  abline( h=c(-2,2) , col="dodgerblue" )

 plotSmear( cds , de.tags=de.genes.tgw[1:500] , main="Tagwise" ,
            pair=c("C","T") ,
            cex = .5 ,
            xlab="Log Concentration" , ylab="Log Fold-Change" )
  abline( h=c(-2,2) , col="dodgerblue" )
par( mfrow=c(1,1) )
```

2.9 Outputting Results

The analysis is essentially complete and now we want to output a table or a csv file containing all the results. We'll pull together the concentrations, fold-changes, p-values, adjusted p-values, up/down regulated variable, dispersions, and the count matrix. Here's how that's done:

```
# Change column names to be specific to the analysis, logConc and logFC are the same in both.
colnames( resultsTbl.cmn ) <- c( "logConc" , "logFC" , "pVal.Cmn" , "adj.pVal.Cmn" )
colnames( resultsTbl.tgw ) <- c( "logConc" , "logFC" , "pVal.Tgw" , "adj.pVal.Tgw" )

# Below provides the info to re-order the count matrix to be in line with the order of the results.
wh.rows.tgw <- match( rownames( resultsTbl.tgw ) , rownames( cds$counts ) )
wh.rows.cmn <- match( rownames( resultsTbl.cmn ) , rownames( cds$counts ) )
 head( wh.rows.tgw )

# Tagwise Results
combResults.tgw <- cbind( resultsTbl.tgw ,
                          "Tgw.Disp" = cds$tagwise.dispersion[ wh.rows.tgw ] ,
                          "UpDown.Tgw" = decideTestsDGE( de.tgw , p.value = 0.05 )[ wh.rows.tgw ] ,
                          cds$counts[ wh.rows.tgw , ] )
head( combResults.tgw )

# Common Results
combResults.cmn <- cbind( resultsTbl.cmn ,
                          "Cmn.Disp" = cds$common.dispersion ,
                          "UpDown.Cmn" = decideTestsDGE( de.cmn , p.value = 0.05 )[ wh.rows.cmn ] ,
                          cds$counts[ wh.rows.cmn , ] )
head( combResults.cmn )
```

Combining both common and tagwise results together:

```
wh.rows <- match( rownames( combResults.cmn ) , rownames( combResults.tgw ) )

combResults.all <- cbind( combResults.cmn[,1:4] ,
                          combResults.tgw[wh.rows,3:4] ,
                          "Cmn.Disp" = combResults.cmn[,5],
                          "Tgw.Disp" = combResults.tgw[wh.rows,5],
                          "UpDown.Cmn" = combResults.cmn[,6],
                          "UpDown.Tgw" = combResults.tgw[wh.rows,6],
                          combResults.cmn[,7:ncol(combResults.cmn)] )
head( combResults.all )

# Ouput csv tables of results
write.table( combResults.tgw , file = "combResults_tgw_ex1.csv" , sep = "," , row.names = TRUE )
write.table( combResults.cmn , file = "combResults_cmn_ex1.csv" , sep = "," , row.names = TRUE )
write.table( combResults.all , file = "combResults_all_ex1.csv" , sep = "," , row.names = TRUE )
```

3 References

- *Robinson MD, McCarthy DJ and Smyth GK* (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics 26, 139-140*
- *Robinson MD and Smyth GK* (2007). Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics 23, 2881-2887*
- *Robinson MD and Smyth GK* (2008). Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics, 9, 321-332*
- *Benjamini, Y., and Hochberg, Y.* (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B, 57, 289300.*