Author: Bernadette Johnson
Created: Oct 2020
Last Modified: March 2022

Differential Expression Analysis using RSEM with EBSeq or edgeR

About this Protocol

Here I will show you how to install and run RSEM, EBSeq, and edgeR. This protocol is for users who have assembled transcriptome data and are interested in a differential expression analysis between samples. The reason why I am focusing on EBSeq and edgeR is because they are the more popular tools for differential expression analysis. For a general suggestion, edgeR is preferred when you have a smaller sample size, and EBSeq tends to use a more conservative approach when testing differentially expressed genes. However, the more time you spend getting to know each program, the better you will be able to compare the quality of results, and thus make decisions on which one works best with your study.

Table of Contents

Before you start **1**
Installing RSEM and Bowtie **2**
Installing R, EBSeq, and edgeR **4**
Running RSEM **6**
Running EBSeq (Two condition comparison) **8**
Running EBSeq (Multiple condition comparison) **9**
Running edgeR (Two condition comparison) **12**
Running edgeR (Multiple condition comparison) **15**

Before you start:

Computer requirements and recommendations:

- A Linux system, subsystem, or a Linux virtual box
- Additional hard drive, other than your Local Disk (C:) drive, with ~1 TB of free space
- Available memory RAM of at least 30 GB, but preferably more
- Available CPU of at least 6, but preferable more

Total programs downloaded for this protocol:

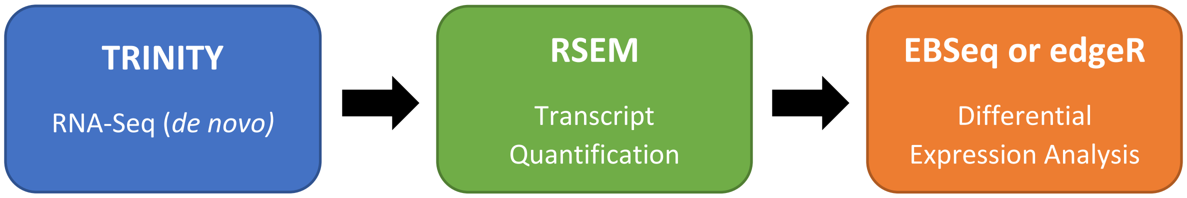
R, RSEM, Bowtie, Bioconductor, EBSeq (also needs blockmodeling and digest), edgeR

Information for working example:

Context: For this project, I am interested in comparing the testes and ovaries transcriptomes of the same species of fish. I assembled the transcriptome using Trinity, so my main transcriptome file is *Trinity.fasta*.

System: I will be using a Linux subsystem installed on Windows 10. I have an additional hard drive with 5 TB of free space, 128 GB of RAM and 12 CPU logical processors that I will be using to run this demonstration. In my home directory, I have a folder named 'shared' which I will work in for most of the process and examples.

General Pipeline:



Please visit and learn more about each of the programs:

RSEM (https://deweylab.github.io/RSEM/)
EBSeq (https://bioconductor.org/packages/release/bioc/html/EBSeq.html)
edgeR (https://www.bioconductor.org/packages/release/bioc/html/edgeR.html)

If you run into an issue following this protocol, or have general comments, please feel free to contact me at john0533@vandals.uidaho.edu.

Let's get started

1. Installing RSEM and Bowtie

RSEM

1. From the RSEM website (https://deweylab.github.io/RSEM/), download the latest version of the source code (for me this is RSEM v1.3.1).

2. Change to the directory, or folder, that you saved the RSEM zipped file to. Unzip the file using gunzip.

3. To compile RSEM, move into the unzipped file and in the command line type:
   ```
   > sudo make
   ```

4. To install RSEM, run
   ```
   > sudo make install
   ```

5. Next, we will add RSEM to your PATH, so you can run any RSEM programs from any directory. Change directories to your home directory, and open your .bashrc script.
   ```
   > nano .bashrc
   ```

6. Without modifying pre-existing parts of the file, append the follow to the very end:
   ```
   #RSEM
   export PATH=$PATH:~/shared/RSEM-1.3.1/
   ```
   *The path to your RSEM folder might be different then what is stated here. My RSEM folder is located in another folder titled "shared".

Bowtie

7. Installing Bowtie
   ```
   > sudo apt install bowtie
   ```

Alternative: Installing Bowtie 2

8. If you instead want to install Bowtie 2 to run RSEM, navigate to Bowtie's website and download the latest binary version (https://sourceforge.net/projects/bowtie-bio/files/bowtie2/). For me that is *bowtie2-2.3.5.1*-linux-x86_64.

9. Unzip this downloaded folder, and make sure it is somewhere easily accessible.

10. Next, we will add Bowtie2 to your PATH. Change directories to your home directory, and open your .bashrc script.

```
> nano .bashrc
```

11. Without modifying pre-existing parts of the file, append the follow to the very end:

```
#Bowtie2
export PATH=$PATH:~/shared/bowtie2-2.3.5.1-linux-x86_64/
```

2. Installing R, EBSeq, and edgeR

For installing EBSeq and edgeR, you have two options. Installing on RStudio or in your Linux subsystem. I generally prefer to run these programs through my Linux subsystem because I am more comfortable with this option. If you want to use any built-in functions for generating heat maps and plots, as well as any G.O. analysis, such as those available in edge R, then I recommend installing on RStudio.

Installing on RStudio: This is very easy and provides the user with a more GUI feel. To accomplish installation on RStudio, follow the steps below:

1. Correctly install RStudio. In short, you should install R before installing RStudio. There are many online tutorials you might need to look up if you need further assistance.

2. To install EBSeq:

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
> BiocManager::install("EBSeq")
> library(EBSEq) #You will need to load this library every time
you reopen R and want to use EBSeq.
```

3. To install edgeR:

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
> BiocManager::install("edgeR")
> library(edgeR) #You will need to load this library every time
you reopen R and want to use edgeR.
```

Installing on your Linux subsystem: We will start by installing R (version 3.6.1 "Action of the Toes"), and Bioconductor (3.9), which are needed to install EBSeq (version 1.24.0) and edgeR (version 3.26.8).

1. To install the correct version of R, first check what version of Ubuntu you have.

```
> lsb_release -a
```

2. My version is Ubuntu 18.04 (*bionic*). Making note of your version, adjust the following commands as necessary:

```
> sudo apt install apt-transport-https software-properties-common
> sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys
E298A3A825C0D65DFD57CBB651716619E084DAB9
> sudo add-apt-repository 'deb https://cloud.r-
project.org/bin/linux/ubuntu bionic-cran35/'
> sudo apt update
> sudo apt install r-base
> R --version
```

   * If you instead followed instructions from the R website and ran `sudo apt-get install r-base` in your command line, you will have most likely installed R version 3.4.4. This is not the version we want! Run this to remove this version of R:

```
> sudo apt-get remove r-base-core
```

3. Now we can start R.

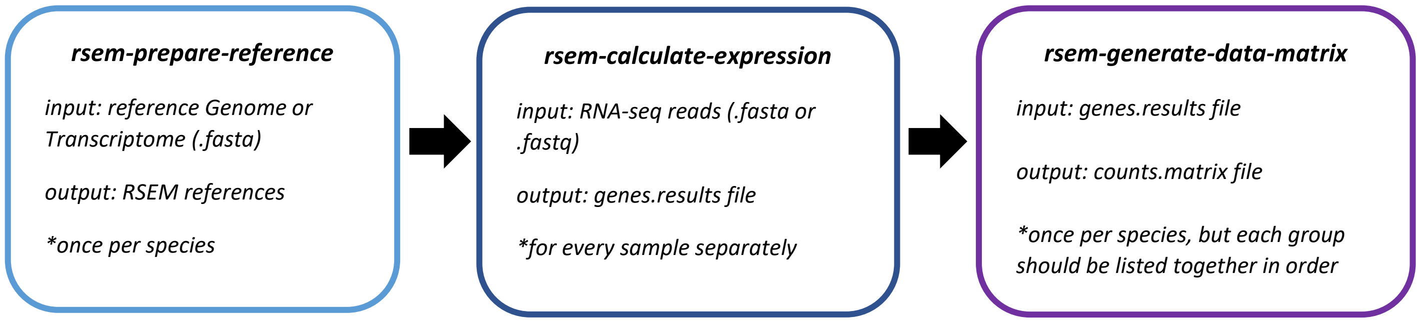
```
> R
```

4. Now we will be running commands in R (**all commands in this font are for R**).

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
install.packages("BiocManager")
> BiocManager::install("EBSeq")
> BiocManager::install("edgeR")
> install.packages('blockmodeling')
> install.packages('digest')
```

5. Once installed, restart your terminal window by closing and reopening.

3. Running RSEM


rsem-prepare-reference
input: reference Genome or Transcriptome (.fasta)
output: RSEM references
*once per species
rsem-calculate-expression
input: RNA-seq reads (.fasta or .fastq)
output: genes.results file
*for every sample separately
rsem-generate-data-matrix
input: genes.results file
output: counts.matrix file
*once per species, but each group should be listed together in order

We will use RSEM (with the above pipeline) for differential expression analysis. We are not going to pre-process or filter any data from Trinity beforehand. The data will go through RSEM as is, and filter data through the statistical analysis portion (end) of the process. You will need 5-15 GB memory. Your *.fasta* file should be a resulting assembled reference transcriptome from Trinity, and your read files should be trimmed, paired reads from Trimmomatic. We will map each pair of read files, one pair at a time, against the reference transcriptome. It will be best to perform the following steps from within a folder, that contains your reference transcriptome (*.fasta*) and your read files. Information on this command can be found (http://deweylab.biostat.wisc.edu/rsem/rsem-prepare-reference.html).

1. We will start by preparing the reference sequences to run RSEM. This will also build indices for bowtie.

   ```
   > rsem-prepare-reference --bowtie Trinity.fasta TrinityRef1
   ```

   ```
   *This will build indices for bowtie. If you instead want bowtie2,
   then use the --bowtie2 option.
   ```

2. Now we will calculate the expression levels using the reference we prepared and the paired read files for each sample, separately.

   ```
   > rsem-calculate-expression -p 10 --paired-end
   paired_read_1_forward.fastq paired_read_1_reverse.fastq
   TrinityRef1 Male1
   ```

```
*-p option is the number of threads to use, or how many times the
computer should subdivide the overall process and will depend on
your computer’s memory capabilities.
*You will need to run this line for each sample. For us, this
would be run six times for Male 1, Male 2, Male 3, Female 1,
Female 2, and Female 3. All six will use the same Trinity
Reference file, TrinityRef1.
```

3. Now we will calculate the expression levels using the reference we prepared and the paired read files. Samples should be listed by group, so for example, if you are interested in differential expression between male and female tissue types, have all of one group listed first, then the other. Here I list all the males first, then all the females.

```
> rsem-generate-data-matrix Male1.genes.results
Male2.genes.results Male3.genes.results Female1.genes.results
Female2.genes.results Female3.genes.results >
RSEM_digi.counts.matrix
```

```
*The important output file that will be generated will be
RSEM_digi.counts.matrix which will be used for differential
analysis. Additionally, a set of files with the extension
genes.results will be generated for each sample. These are also
important for
```

4. From here, you may run EBSeq through RSEM or if you would like to use EBSeq independently (or if you want to use edgeR), skip this step and proceed to the next page.

   Using this command, we specify that we want to run the on-board EBSeq function through RSEM. We also specify the number of samples within each of our groups. From step 3, we should have our males as the first three samples, followed by three female samples. In the last part of the command we specify our output file. The output file can be read through Notepad++ and copied into an excel spreadsheet.

```
> rsem-run-ebseq RSEM_digi.counts.matrix 3,3
malevsfemale_ebseq_results
```

4. Running EBSeq (Two condition comparison)

1. Now we can start R.

```
> R
```

* This will start R. Now we will be running commands in R (**all commands in this font are for R**). Make sure you are in the directory that contains your "`RSEM_digi.counts.matrix`" file.

2. Load EBSeq package. This will automatically load other required packages.

```
> library(EBSeq)
```

3. Create a factor descriptor that explains what your 'treatment' groups are, in the order you placed them. In the example above we have three males and three females (first example).

```
> Conditions=as.factor(c("M","M","M","F","F","F"))
```

4. Now double-check that your conditions specified are accurate. Also check to make sure they levels are correctly specified.

```
> Conditions
```

5. We are going to load your counts matrix into R, into a data matrix named 'GenMat'. This step will not work unless you are in the directory where your digi.counts matrix is. The MedianNorm function is a median-by-ratio normalization function.

```
> GenMat<-data.matrix(read.table(file="RSEM_digi.counts.matrix"))
> Sizes=MedianNorm(GenMat)
```

6. Now we will run EBSeq on our data matrix with the conditions specified which indicates which samples belong to which group for comparison. If you have only two conditions, then use '**EBTest**', when you have more then two conditions, use 'EBMultiTest'. The '**maxround**' option is the number of iterations the test should be run.

```
> EBOut=EBTest(Data=GenMat, Conditions=Conditions, sizeFactors=Sizes, maxround=5)
```

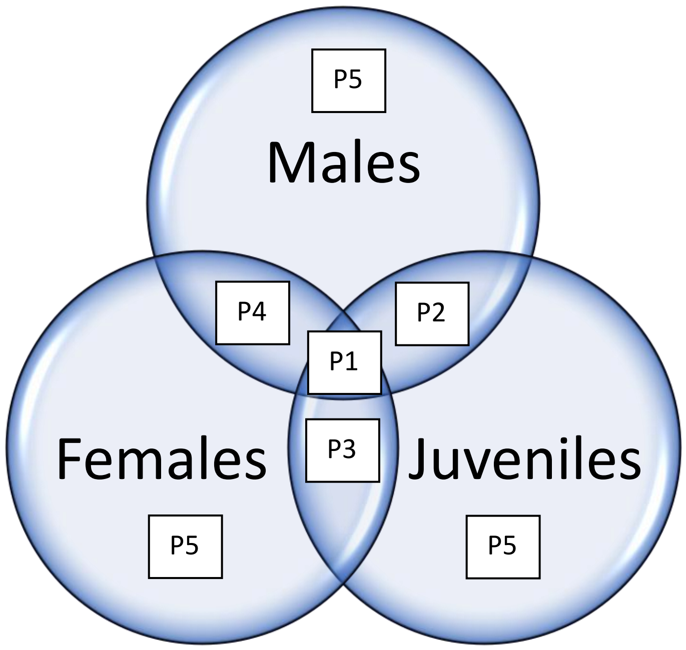
7. These next steps are to move these results into a .csv file, which can then be read in OpenOffice or Excel.

```
> PP=GetPPMat(EBOut)
> write.csv(PP,file="DiffExpressionPosteriorProbs.csv")
```

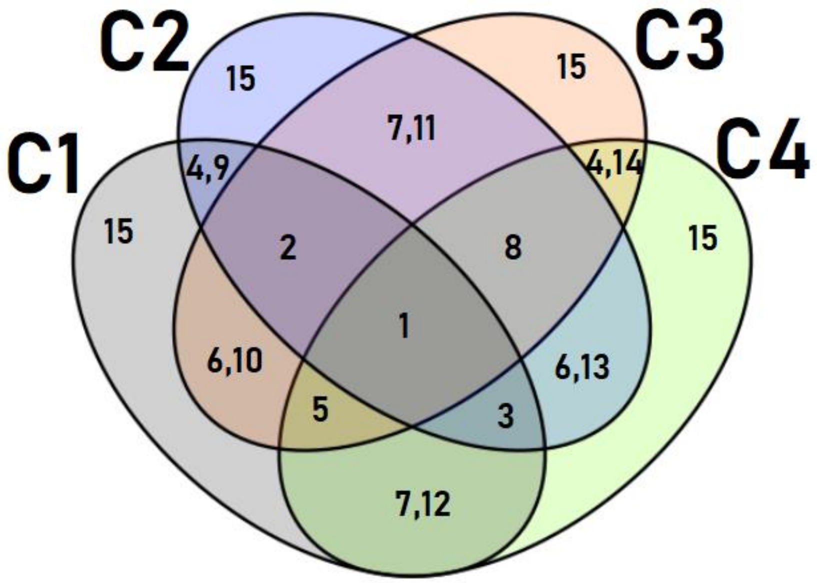
5. Running EBSeq (Multiple condition comparison)

1. If you have more than two conditions, for example, if we had added data on juveniles to our female and male comparison study, we now have multiple comparisons (or 5 patterns). For this type of comparison EBSeq will provide a posterior probability of each gene being in a particular pattern. The probability value is from a Bayesian approach where the priors are all equal- meaning each pattern is equally likely to occur. Here is an illustration that might help clarify:

	J	M	F
Pattern1	1	1	1
Pattern2	1	1	2
Pattern3	1	2	1
Pattern4	1	2	2
Pattern5	1	2	3



In this figure, Pattern 1 is where all conditions share the same expression level, Pattern 2 is where the expression level is shared only between Juveniles and Males but not Females, Pattern 3 is where the expression level is shared by Juveniles and Females but not Males, Pattern 4 is where the expression level is shared by Males and Females but not Juveniles, and Pattern 5 is where the expression level is different for three conditions. *This venn diagram is not an entirely accurate depiction* however, as when you increase the number of conditions the interpretation can become a bit trickier. Here is an example for a 4-condition test, where a venn diagram is not helpful.



	C1	C2	C3	C4
Pattern1	1	1	1	1
Pattern2	1	1	1	2
Pattern3	1	1	2	1
Pattern4	1	1	2	2
Pattern5	1	2	1	1
Pattern6	1	2	1	2
Pattern7	1	2	2	1
Pattern8	1	2	2	2
Pattern9	1	1	2	3
Pattern10	1	2	1	3
Pattern11	1	2	2	3
Pattern12	1	2	3	1
Pattern13	1	2	3	2
Pattern14	1	2	3	3
Pattern15	1	2	3	4

In this figure, patterns such as Pattern 4 indicate that conditions 1 and 2 have a similar expression level and conditions 3 and 4 have a similar expression, but they are different from each other. This is a different interpretation than Pattern 9, where conditions 1 and 2 have a similar expression that is different from expression in condition 3 and 4. Here, conditions 3 and 4 do not have a similar expression to each other. This becomes hard to interpret, so make sure you consider your experimental design beforehand and determine if you need to run two pairwise comparisons or a multi-comparison.

2. We have a similar process as before once we start R.

```
> library(EBSeq)
> Conditions=as.factor(c("M","M","M","F","F", "J", "J", "J", "J"))
> Conditions
> GenMat<- data.matrix(read.table(file= "fish.counts.matrix"))
> PosParti=GetPatterns(Conditions)
> PosParti  #These are your expression patterns.
> MultiSize=MedianNorm(GenMat)
> MultiOut=EBMultiTest(GenMat, NgVector=NULL, Conditions=Conditions, AllParti=PosParti,
sizeFactors=MultiSize, maxround=5)
```

3. We will produce two outputs from this process. The first file is a spreadsheet with all the posterior probabilities for each gene, for each pattern. The second file is a spreadsheet that gives the pattern with the highest posterior probability for each gene (best fit only).

```
> MultiPP=GetMultiPP(MultiOut)
> write.csv(MultiPP$PP, file= "MultiPostProbs.csv")
> write.csv(MultiPP$MAP, file= "MultiPostProbs_MAP.csv")
```

6. Running edgeR (Two condition comparison)

1. First, you will start by loading the resulting file from RSEM. edgeR is designed to work with *actual read counts*, not predicted transcript abundances. For this guide, the actual read counts for each sample can be found in the *RSEM_digi.counts.matrix* created previously. You might need to delete the gene ID column (then the row number will be interpreted as the read name).

   Here you can choose to pre-filter your data and impose certain cutoff values. For example, you might choose to only keep reads that have an actual count ≥ a value, for at least a certain number of samples. If you would like to pre-filter your reads, you can open the *RSEM_digi.counts.matrix* file in Notepad ++, copy and paste into an Excel sheet, sort and filter your reads, then save the resulting reads in a *data.csv* file. There is an additional filtering step you can do within edgeR, and for this reason, I typically do not pre-filter my data.

   ```
   > data<-read.csv("mydata.csv", header=T)       #load csv
   > rownames(data)=data$gene_id                  #rename rows to gene id
   >mydata<-data[,c(-1)]                          #remove first column that contained gene id
   ```

2. Now we will tell edgeR which samples belong to which experimental group. Then we will store the data in a list-based data object (DGEList). Reopen your *RSEM_digi.counts.matrix* file and make note of the order of samples for each column. For this example, we are working with three male "M" followed by three female "F" samples.

   ```
   > datagroup <- c ("M","M","M","F","F","F")

   > d<-DGEList(counts=mydata, group=factor(datagroup))
   ```

3. This step is an optional filtering step. Here we will filter out and remove reads that have a CPM value less than 2 in 3 or more samples. We will also check the library size before and after we filter to make sure nothing unexpected or dramatic occurs.

   ```
   > d #this shows the library size for each sample, or you can use dim(d) just to see the
   dimensions of everything together

   > keep<-rowSums(cpm(d)>2)>=3

   > d<-d[keep,]

   >d$samples$lib.size <- colSums(d$counts)
   ```

```
> d$samples #now we are checking the library size after filtering, you can also use dim(d) again
```

4. Now we will normalize the data. This is only necessary for sample-specific effects (which we have in a comparison between males and females).

   ```
   > d<-calcNormFactors(d)
   > d$samples #this shows the normalization factors
   ```

5. Estimating dispersion using a quantile-adjusted conditional maximum likelihood (qCML). This method is recommended for experiments with a single factor (which means it is specific to the number of conditions).

   ```
   > d1 <- estimateCommonDisp(d, verbose=T)
   > d1 <- estimateTagwiseDisp(d1)
   > plotBCV(d1) #plots the tagwise biological coefficient of variation (square root of dispersions) against log2-CPM
   ```

6. Testing for differentially expressed genes. Here I use the exactTest() followed by the Benjamini-Hochberg false discovery rate with alpha set to 0.05. The exactTest is not the Fisher's exact test but has "strong parallels with Fisher's exact test" (via edgeR user guide). This is specific to the method used for estimating dispersion, if you estimated dispersion differentially then described in step 5, find the appropriate significance test for you.

   ```
   > et <- exactTest(d1)
   > de1 <- decideTestsDGE(et, adjust.method="BH", p.value=0.05)
   > summary(de1) #this describes the number of reads up-regulated in M, down-regulated in M (or up-regulated in F), and not significant (which usually will be most reads).
   ```

```
       M-F
Down     #
NotSig ####
Up       #
```

7. Saving the data. Here you have the option to save the top X number of reads you are interested in, or all of them.

```
> topTags(et, n=10) #see and save top 10

> result<-(topTags(et, n=10)$table)

> write.csv(result, file="topTags_results.csv")


> summary(de1) #add all the reads up, so total number=X

> result<-(topTags(et, n=X)$table)

> write.csv(result, file="topTags_results.csv") #save all of the reads
```

8. Here is an easy way to have an informative plot to visualize the data:

```
de1tags12 <- rownames(d1)[as.logical(de1)]

plotSmear(et, de.tags=de1tags12)

abline(h = c(-2, 2), col = "blue")
```

7. Running edgeR (Multiple condition comparison)

For a three (or more)-condition comparison, you will follow many of the same starting 4 steps for a two-condition comparison (with appropriate modifications) except for estimating dispersion and testing for differentially expressed genes. I highly recommend you plan your analysis after reading the user guide for edgeR, especially for more complex comparisons. This section will mostly be a condensed version, straight from their user guide. Changes to these two steps are described below:

5. Estimating dispersion using a quantile-adjusted conditional maximum likelihood (GLM). First, we create a design matrix.

   ```
   > design <- model.matrix(~ Sample + Treatment)

   > d1 <- estimateGLMCommonDisp(d, design)

   > d1 <- estimateGLMTrendedDisp(d1, design)

   > d1 <- estimateGLMTagwiseDisp(d1, design) #to estimate tagwise dispersions
   ```

6. Testing for differentially expressed genes using a QL model.

   ```
   > group <- factor(c(1,1,2,2,3,3))

   > design <- model.matrix(~group)

   > fit <- glmQLFit(y, design)
   ```

7. Comparing between groups.

   ```
   > qlf.2vs1 <- glmQLFTest(fit, coef=2) #compare group 2 vs 1

   > topTags(qlf.2vs1) #top 10 genes from comparison between groups 2 and 1

   > qlf.3vs1 <- glmQLFTest(fit, coef=3) #compare group 3 vs 1

   > qlf.3vs2 <- glmQLFTest(fit, contrast=c(0,-1,1)) #compare group 3 vs 2

   > qlf <- glmQLFTest(fit, coef=2:3) #to find genes different between any of the three groups

   > topTags(qlf)
   ```

8. Alternatively, you can perform a likelihood ratio test to test for differential expression

```
> fit <- glmFit(y, design)

> lrt.2vs1 <- glmLRT(fit, coef=2)

> topTags(lrt.2vs1)
```

9. To save the output.

```
> result<-topTags(x)

> write.csv(result, file="topTags_results.csv")
```