```
Effects of temperature on EPEC expression
==========================================================

Introduction
------------

This analysis describes differential expression analysis of an RNA-Seq data s
23 degrees C (C23EPEC) and 37 degrees C (T37EPEC)

The samples were from the laboratory of Christine White-Ziegler. You will be
instructions to create your own RStudio analysis with your own data.


###Questions to answer in this data analysis
###
* How similar were the samples within or across groups?
* How many genes were expressed in EPEC23 and in EPEC37?
* How many genes were differentially expressed due to the treatment, i.e., th
What were they?

Analysis
--------

To answer these questions, we'll import the edgeR package from the Bioconduct
perform differential expression analysis using the edgeR "classic" methods. A
manual and vignettes.

Note that this code assumes you've downloaded and installed edgeR. You also n
edgeR and limma, and then the library, instead of just running the chunk belo
enter, each line individually.  For some reason, the library does not install
You have to try to install the library, wait for it to get stuck on installin
limma install line, then the library again.

```{r message=FALSE}
source("http://bioconductor.org/biocLite.R")
biocLite("edgeR")
library(edgeR)
biocLite("limma")
library(edgeR)
```

The edgeR user's guide is worth a look! Open it:

```{r}
edgeRUsersGuide()
```
```

```
```

Notice the guide contains many case studies illustrating different applicatio

See also

* http://www.bioconductor.org/packages/release/bioc/html/edgeR.html

###About input data files
###
We used Rockhopper to align the sample reads onto the EPEC genome, and count
overlapping annotated genes, requesting "Verbose Output" in the Rockhopper pa
the raw reads. The resulting data file for the sample data used for this wor
~/RockhopperResults/EPEC_forRstudio/NC_011601_transcripts_forRstudio.txt.
We re-formatted this data from the 'transcripts.txt' file, produced by Rockho
Excel, sorting all data by 'Synonym', deleting all rows with no synonym, sort
'Translation Start' column, and copying and pasting the 'Synonym' column and
columns to a new workbook. Now close the original transcripts file and DO NO
was all to get rid of rows with the same name and keep in the same order as t
will use downstream, BUT we don't want to change the original file so we do n
Alternatively, you can duplicate the original so you don't risk messing it up
created file, we shortened the Raw Counts headings to <10 characters and inse
spaces (Easiest if you start all control names with C and all treatment names
this as a new tab delimited text .txt file with the same name but adding '_fo
this file in TextWrangler. Under 'View' go to 'Text Display' -> 'Show Invisib
and any tabs (depicted as triangles) that come before the first sample name.
this new file to perform differential expression analysis.

####Read the data-put your new results file of interest into your folder yo w
####otherwise, you will need to include the entire path name in the chunk bel
####~/Rockhopper_Results/EPEC23vsEPEC37/NC_011601_transcripts_forRstudio.txt
####filename is)
####
```{r}
d=read.delim('3newto2oldandnew_011601_transcripts_forRstudio.txt')
```

View the first few rows of data:

```{r}
head(d)
```

Note that the data are nicely formatted, with gene names as row names and sam
```
```

names.

####Make a DGEList object
####
We'll use the DGEList function to create a DGEList object to contain the data

The DGEList function needs our table of counts (d) and a vector indicating wh
belongs to. We also include an optional named argument (remove.zeros) that el
zero counts. We eliminate genes with zero counts since it makes no sense to t
differential expression if they were not expressed.

To learn more about the DGEList() function, type `help(DGEList)`

First, make a vector that indicates the group affiliation for each sample - C
treatment.

```{r}
#define a new character vector using the "c" function
group=c('C','C','C','C','T','T','T')
```

Then use the table of counts and the group affiliation vector to build the DG

```{r}
dge=DGEList(d,group=group,remove.zeros=TRUE)
```

Take a moment to look at the DGEList object:

```{r}
mode(dge) # find out what type it is
dge # type it in the console to see what it is
```

It's a list and it has two members: the original data frame that has been tur
matrix and also an object called 'samples' that is also a list. Note that whe
DGEList object, it used the groups object to assign group membership to each
calculated the library sizes.

###Normalization
###
Some libraries had more reads than others and so counts per gene are not dire
have to normalize. This is obvious and easy to understand. However, there is
normalization that we need to take into account, which is that a treatment (l

increase in the expression of a subset of genes, thus "consuming" counts that come from less highly expressed genes and making those genes appear down-regu they were not.

For a deeper discussion, see [Robinson and Oshlack (2010) A scaling normaliza differential expression analysis of RNA-Seq data](http://genomebiology.com/20

Note that when we normalize in edgeR, the software does not change the counts object. Intead, it calculates *normalization factors* that will be used later

```{r}
dge=calcNormFactors(dge)
```

Note that when we apply the calcNormFactors function to the dge object, it bo the dge object while also changing it. Specifically, it added new information case, it updated the normalization factors that were stored in the samples co

Look at the normalization factors - notice that the norm.factors now are bigg depending on the read depth of the corresponding sample libraries:

```{r}
dge$samples
```

###Visualizing differences between samples
###
We expect that 23 degree samples will more closely resemble other 23 degree s samples, and vice versa. If they don't, this can indicate flaws in the experi protocol. In general, the greater the separation between experimental groups, of differentially expressed genes.

Two types of plots are useful in this step - MDS plots and hierarchical clust

####MDS Plot
####
A multi-dimensional scaling (MDS) plot can show similarity between samples.

From the edgeR guide:

>>>The function plotMDS draws a multi-dimensional scaling plot of the RNA sam
>>>correspond to leading log-fold-changes between each pair of RNA samples. T
>>>log-fold-change is the average (root-mean-square) of the largest absolute
>>>between each pair of samples. This plot can be viewed as a type of unsuper
>>>

Create the plot:

```{r}
#color for controls
cn.color='blue'
#color for treatments
tr.color='brown'
#define a title for the plot
main='MDS Plot for Count Data'
#par(las=1) # makes y axis labels horizontal not vertical
colors=c(rep(cn.color,4),rep(tr.color,3))
plotMDS(dge,main=main,labels=colnames(dge$counts),
        col=colors,las=1)
```

The second dimension does a good job of separating samples but often is not
their similarities or differences if there is a low N. Let's see what Hierar
do.

###Hierarchical clustering
###
You can use hierarchical clustering to view the relationships between samples
calculates a distance value between each sample using measurements from all t
then clustered according to how similar they are with respect to that distanc

Note we need to use normalized "counts per million" instead of the raw count
that we have to transpose the matrix so that columns become rows and rows bec
words, to do the clustering, the entities that should be clustered need to be

```{r}
normalized.counts=cpm(dge)
transposed=t(normalized.counts) # transposes the counts matrix
distance=dist(transposed) # calculates distance
clusters=hclust(distance) # does hierarchical clustering
plot(clusters) # plots the clusters as a dendrogram
```

Looks like the samples grown at 37 degrees clustered together and away from s
degrees, which formed their own cluster as expected. This is a useful plot f
or something inherently wrong with the experiment overall. We will continue a
studies, if you see different groups clustering together or a sample off on i
to eliminate samples (or verify their labelling throughout the experiment) an

###Differential expression analysis
###

Now let's use functions in edgeR to identify genes whose expression changed d
change. For our analysis, we'll treat all the 23 and 37 degree samples as rep
variables as well we would ignore this for the first pass and treat all sampl
condition as a replicate. This corresponds to what the edgeR manual calls th

An alternative approach would be to use edgeR's generalized linear models met
differential expression analysis that takes batch effects (batches collected
into account. The Arabidopsis study in the User's Guide gives a nice example
this Markdown, however, we'll stick with the classic approach for demonstrati

####Estimating dispersion
####
To start, we need to estimate dispersion, which reflects the degree to which
depends on expression level. We need this to model gene expression and test w
has changed due to the treatment.

The details of how this works are explained in the edgeR user's guide. In a n
is that we use the negative binomial model to estimate a dispersion parameter
edgeR calls "the degree of inter-library variability" for that particular gen

Using edgeR to calculate common and tagwise dispersions:

```{r}
dge=estimateCommonDisp(dge)
dge=estimateTagwiseDisp(dge)
```

Now you can look again at the dge object - notice there are now new component
common and tagwise dispersions.

####Differential expression
####
Now that we have estimates for dispersion for every gene, we can use the exac
each gene individually using the exact negative binomial test.
```{r}
dex=exactTest(dge,pair=c("C","T"),dispersion="tagwise")
#design = model.matrix(~group)
#dge=estimateGLMCommonDisp(dge,design)
#dge=estimateGLMTrendedDisp(dge,design)
#dge=estimateGLMTagwiseDisp(dge,design)
#fit=glmFit(dge,design)
#dex=glmLRT(fit,coef=2)
```

Note This is a "classic" edgeR analysis according to the User's Guide. To t

linear modeling) edgeR analysis, comment the first line and uncomment the rem

The exactTest function returns a new DGEExact test object that we can pass to
Now we have a new object (dex) that captures the results from testing for dif
genes across the 23 degree and 37 degree samples. Type it into the console to
Notice that it has logFC, logCPM (counts per million), and PValue for each ge

####Multiple Hypothesis testing correction
####
Even if the data were totally random, around 5% of the p values would be 0.05
would be 0.10 or less, and so on. This is because p values reflect the probab
given result by chance. So even if our data were random, we would expect to o
positives if we use a p value threshold of 0.05 to determine significance. Th
multiple hypothesis testing problem which afflicts testing highly parallel, h
correct for this, we can calculate a false discovery rate (FDR) value for eac
values as inputs.

```{r}
#use Benjamini-Hochberg method of calculated false discovery
#rate for each gene
fdrvalues=p.adjust(dex$table$PValue, method='BH')
dex$table$fdr=fdrvalues
```

####Picking an FDR cutoff
####
For subsequent steps, we'll need to select an FDR cutoff for determination of
expression. This is somewhat arbitrary, but unavoidable for some analyses. Fo
Ontology enrichment analysis using GOSeq, we have to designate some genes as
DE.

Use the summary and decideTestsDGE commands to find out how many genes are up
using different Benjamini-Hockberg adjusted FDR cutoffs:

```{r}
summary(decideTestsDGE(dex,p=0.05))
summary(decideTestsDGE(dex,p=0.01))
summary(decideTestsDGE(dex,p=0.005))
```

Based on this, pick a cutoff:

```{r}
cutoff=0.01
```

```

###Get an overview of the DE genes in the data set
###
The plotSmear function illustrates the relationship between logFC and averag
the differentially expressed genes, which are highlighted in red. The blue l
are up or downregulated two fold.

```{r}
de = decideTestsDGE(dex, p = cutoff, adjust = "BH")
detags = rownames(dex)[as.logical(de)]
plotSmear(dex, de.tags = detags)
abline(h = c(-1, 1), col = "blue")
```

Not surprisingly, application of 37 degrees to EPEC results in differential
genes, hopefully many involved in virility and increased metabolism, and for
fold-change is greater than two.

Writing results files
=====================

For the next steps, we'll write files containing lists of differentially exp
then load into other programs like EcoCyc. Just run the following code so th
file. We are just using our .ptt file from our replicon folder. But ptt is a
it works for this exercise, you may find that many sequence feature files co
and many programs want to use that. A great little converter is BEDOPS. It
bed, that you can use here instead of the ptt file or use for further analys
need to install BEDOPS on your Mac (follow instructions on the BEDOPS site).
a script called 'convert2bed'. Once installed, on your terminal you cd into
say a gff file. Then type:

convert2bed -i gff < /Users/lbierwer/Desktop/RtestOnRockhopperResults/NC_011
/Users/lbierwer/Desktop/RtestOnRockhopperResults/NC_011601.bed

Here I told it the format I had, then my input file name with full path (on
drop the file), and what I wanted my output file name to be once converted.

```{r}
#get normalized counts per million
cpms=cpm(dge$counts)
#find out which columns have controls
cn=grep('C',colnames(cpms))
#find out which columns have treatments
tr=grep('T',colnames(cpms))
#calculate the average expression for control samples
```
```

```
ave.cn=apply(cpms[,cn],1,mean)
#calculate average expression for treatment samples
ave.tr=apply(cpms[,tr],1,mean)
#make a data frame
res=data.frame(synonym=row.names(dex$table),
               fdr=dex$table$fdr,
               logFC=dex$table$logFC,
               Cn=ave.cn,
               Tr=ave.tr)
#read gene information (originally from the replicon info we gave Rockhopper
#this is a "sequence feature table" file
annots_file='NC_011601.ptt'
#keep gene id and gene description column
annots=read.delim(annots_file,sep='\t',header=F)[,5:6]
#name the column
names(annots)=c('gene','synonym')
#combine gene expression and annotations
res=merge(res,annots,by.x='synonym',by.y='synonym')
#put the results in order of signficance (fdr)
res=res[order(res$fdr),]
#put columns in an order easy to browse
res=res[,c('fdr','logFC','Cn','Tr','gene','synonym')]
#write DE genes to a file:
out_file='results/3newto2oldandnew_011601_EPEC_DE.txt'
#select just the rows that made the cutoff
de=res$fdr<=cutoff
#write DE genes only
write.table(res[de,],file=out_file,row.names=F,sep='\t',quote=F)
#Make a file we can load into EcoCyc for pathways visualization
out_file='results/3newto2oldandnew_011601_forEcoCyc.txt'
write.table(res[de,c('synonym','logFC')],file=out_file,quote=FALSE,
            sep='\t',col.names=FALSE,row.names=FALSE)
#write all DE genes
out_file='results/3newto2oldandnew_011601_EPEC_AllDE.txt'
write.table(res,file=out_file,row.names=F,sep='\t',quote=F)
```

Summarizing the results

Check out your tables. Cool tip for viewing big spreadsheets in excel: open
furthest right header go to 'Window' -> 'Freeze Pane' Now as you scroll thr
put. Go to 'Window' -> 'Unfreeze Pane' to stop.

Conclusion

```
Of `r nrow(d)` annotated genes, there were `r nrow(dge)` genes with at least
Around `r round(nrow(dge)/nrow(d)*100,1)` % of EPEC37 was expressed in EPEC2
* At fdr of `r cutoff`, we observed `r sum(de)` differentially expressed gen

Limitations of the analysis
---------------------------

* The FDR cutoff we picked might not be the most appropropriate for your dat
* If the number of fragment counts per library is low. Power to detect diffe
also be low, leading to false negatives.
* The assumption about that collection time had no effect could be wrong. To
generalized linear modeling features (glm fucnctions in edgeR) to test for d
while conditioning on collection time.

Session information
-------------------

```{r}
sessionInfo()
```
```