RNA-seq Analysis 3:

Differential Expression

Pietà Schofield

Contents

Data Exploration

Sanity Checks

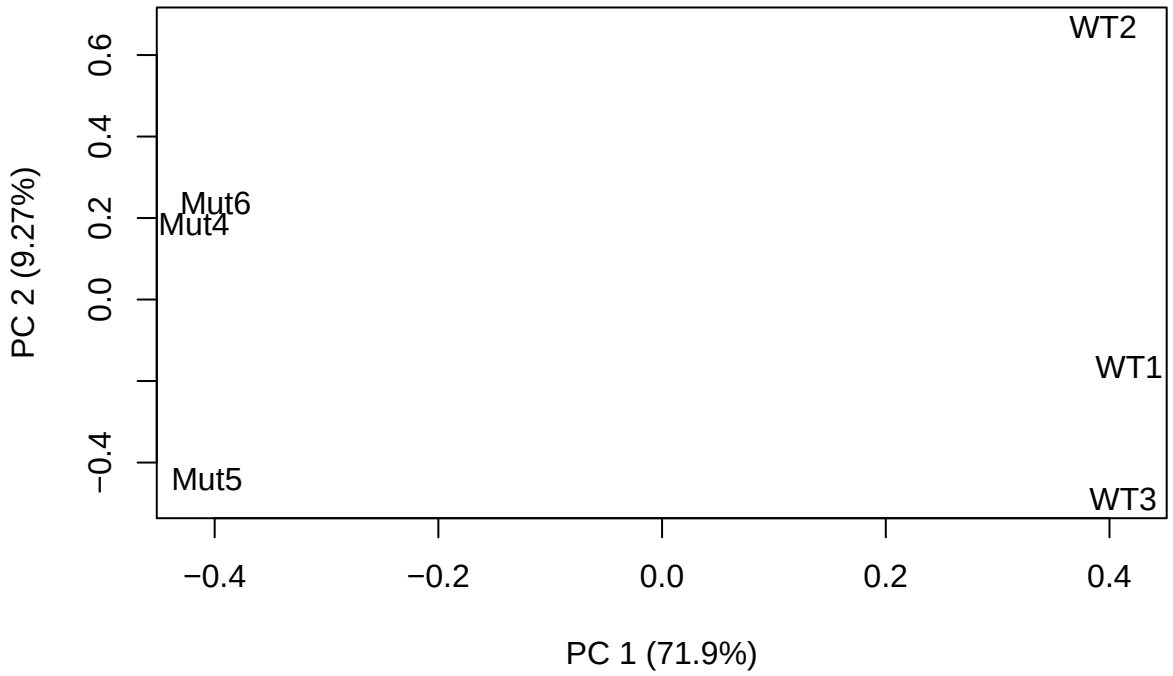
Once we have our feature counts object we can extract the matrix of counts and there are a few sanity checks we can perform to see if it looks like the experiment has work.

PCA and MDS

Principal Component Analysis (PCA) and Multi-Dimensional Scaling (MDS) are complexity reduction techniques and data exploration tools for multi-variate data, for example gene expression profiles where each gene represents a variable with an expression value. PCA attempts to find a transformation of the data that produces combinations of variable with maximum variation between samples. MDS is a broader term that encompasses PCA but include other techniques for reducing the dimension of multi-variate data.

Several of the Bioconductor packages for differential gene expression analysis include functions with names like `plotPCA()` and `plotMDS()` for visualising the relationships between samples. For example the package EDASeq (Risso, Schwartz, Sherlock, et al., 2011) provides a `plotPCA()` function

```
require(EDASeq)
plotPCA(countMatrix)
```



Exercise 3.1

Usine the EDASeq `plotPCA()` function to explore the counts matrix in the featureCounts object created earlier.

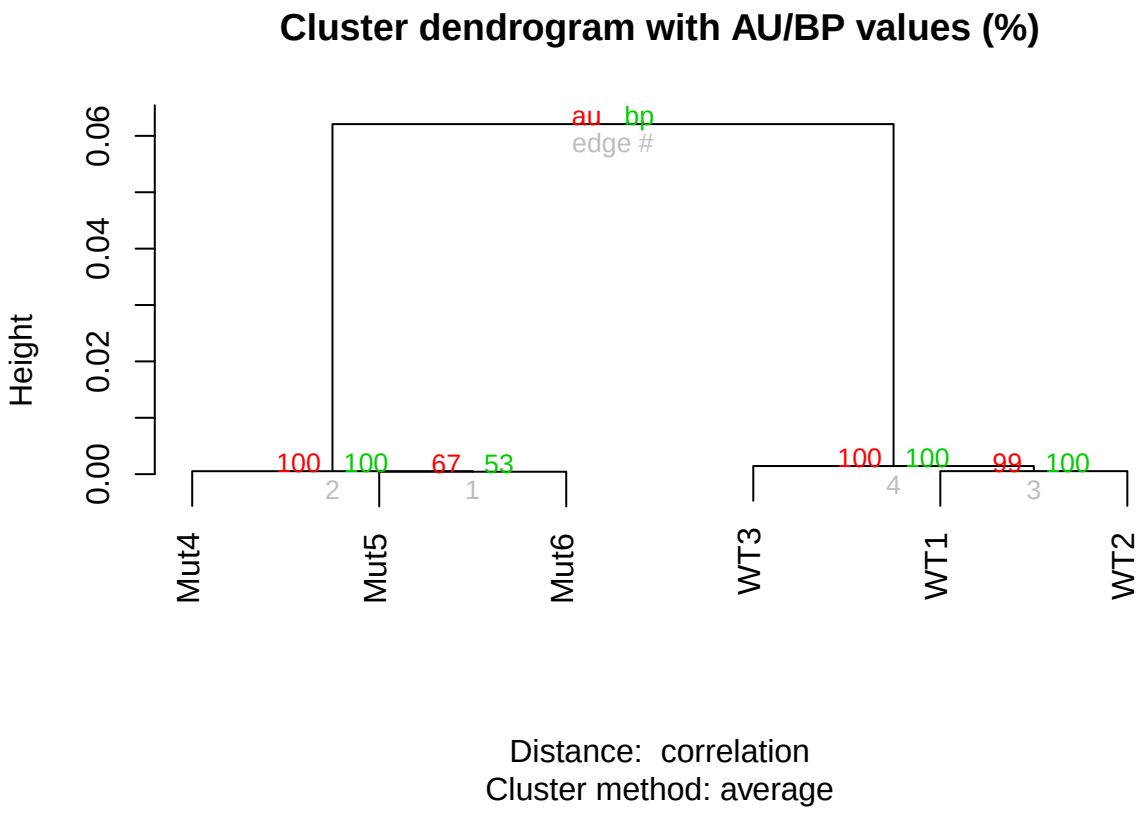
Clustering

Another common data exploration technique used is clustering analysis. In this a distance is calculated between samples based on some combination of variable and a distance matrix generated. A clustering algorithm is them run over the distance matrix that aggregates (groups) samples based on their distance. There are several clustering methods and several clustering packages available in R of particular use in this context are the heirarchical clustering functions `hclust()` available as a base R function and `pvclust()` available in the pvclust package (Suzuki and Shimodaira, 2006) The advantage with the pvclust() function is it uses bootstrapping to provide a level of confidence that the observed clusters are robust to slight changes in the samples.

```
require(pvclust)
pvc <- pvclust(countMatrix)
```

```
Bootstrap (r = 0.5)... Done.
Bootstrap (r = 0.6)... Done.
Bootstrap (r = 0.7)... Done.
Bootstrap (r = 0.8)... Done.
Bootstrap (r = 0.9)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.1)... Done.
Bootstrap (r = 1.2)... Done.
Bootstrap (r = 1.3)... Done.
Bootstrap (r = 1.4)... Done.
```

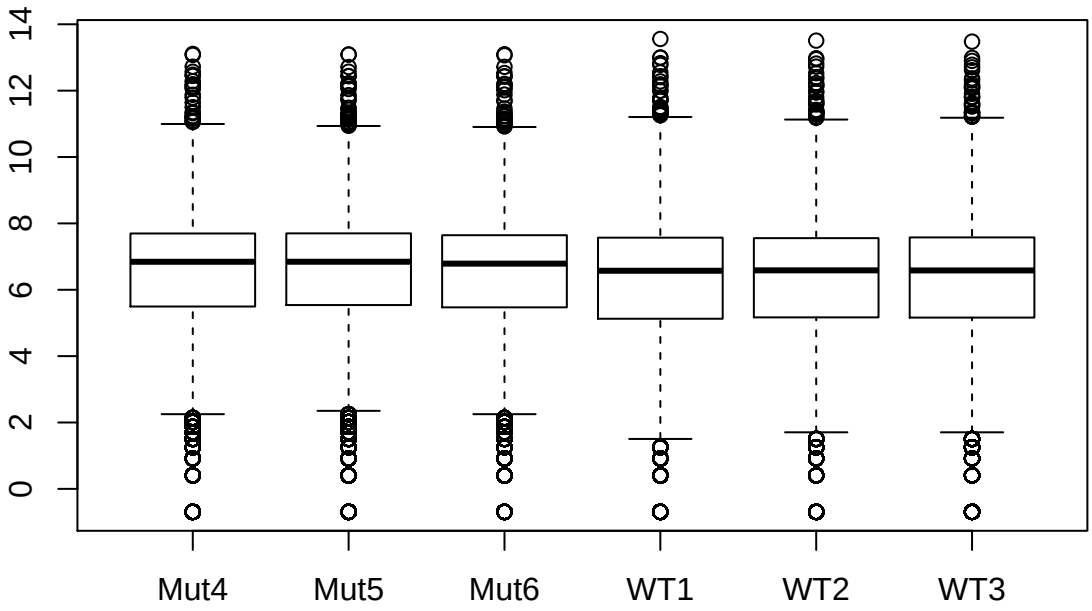
```
plot(pvc)
```



Sample Distributions

The average is a measure of a central tendency in data distribution. There a several familiar averages, the mean, the mode and the median for example. Another way of exploring the samples is to plot box plots of the sample distributions. Again several packages provide convenience functions for box plots of sample distributions. The most obvious is the `boxplot()` function that is available in both the R base package `graphics` and also the bioconductor infrastructure package `BiocGenerics`.

```
boxplot(log(countMatrix+0.5))
```



variations of this are to plot the distribution of the log of the count instead.

The EDASeq package also provides the function `plotRLE()` where RLE stands for Relative Log Expression. The RLE is calculated by first calculating the median expression value of the distribution of all counts and then for each count calculating the expression relative to this average.

Exercise 3.2

Use the `pvclust()` function and the EDASeq function plotRLE() to explore the counts matrix in the featureCounts object created earlier.

Normalisation

There are always sources of unwanted variation in data that are not due to the biological question of interest. The perpose of what is often called normalisation is the removal of unwanted variation between samples so that the variation due to the biological question of interest is not masked by unwanted variation.

Library Size

The most obvious source of variation in NGS data is the difference in the total number of sequences produced for a sample by the sequencer. This is generally called the library size and correcting for library size is an important step.

One way of normalising for library size is to divide the counts for each gene by the total number of reads and multiply by a fixed large number, often choosen as 1M, or sometimes an average for example mean or median of the library sizes between samples. Values normalised for library size are sometimes referred to as Counts Per Million (CPM).

Gene Length

Another source of variation between genes is the length of the gene. A longer gene transcript has a greater chance of having fragments detected by sequencing because there will be more fragments to detect. This also means that when looking for differential expression a gene that has more fragments is likely to have higher chance of a significant result. As with library size one method is to divide the counts per gene by the gene length in base pairs and multiply by a fixed large number often 1K

When counts have been normalised by library size and then gene length they are often referred to as Fragments (or Reads) Per Thousand Bases Per Million Fragments (or Reads) written as FPKM or RPKM. Another slight variation on this approach is to normalise by gene length first and then normalise by the new library size of the normalised values this produces a value called Transcripts Per Million (TPM). There is an explanation of this you can watch over and over here

GC content

Due to the way sequencing technology works it has been noted that genes with higher GC content are more likely to be significantly differentially expressed. The EDASeq packages has functions for GC content normalisation if you supply the sequence for the genes.

RNA composition

Due to the limitation on sequence resources if a highly expressed gene in one condition is not expressed in another condition there will be more sequencing resources to devote to reads from other genes. This can give the impression that the other genes are more highly expressed in the second condition when it is purely due to just having more sequencing resouces available for them. The `edgeR` packages default normalisation routine Trimmed Mean of M-values (TMM) (Robinson and Oshlack, 2010) performs an adjustment that adjusts for RNA composition based on the assumption that most genes are not changing. (See the section on MA plots below for an explanation of M value).

Take Home

The various differential expression packages available in Bioconductor vary in the way they normalise of correct samples and genes for unwanted variation and the field is one of active research without a consensus at the moment to the best approach. The appropriate approach may also depend on the question being addressed. Packages often have several normalisation approaches available. One technique is to perform several different normalisations and look for changes that are robust to normalisation differences. But you should be aware this is a conservative approach and may lead to false negative result, however the alternative of cherry picking the normalisation technique that gives the most positive results should also be avoided.

Be aware also some normalisations may not be appropriate for a particular differential expression tool. Some tools such as limma (Ritchie, Phipson, Wu, et al., 2015) are designed to apply the adjustment to the count data first and then model the differential expression and calculate significance while others such as edgeR (Robinson and Oshlack, 2010; McCarthy, Chen, and Smyth, 2012; Robinson and Smyth, 2007; Robinson and Smyth, 2008; Zhou, Lindsay, and Robinson, 2014) and DESeq2 (Love, Huber, and Anders, 2014) are designed to use the unadjusted counts and apply the adjustment to the model.

MAplots

The MAplot is a form of Bland–Altman plot where a measure of the difference between a variable in two conditions if plotted against mean value of the variable in the two conditions. In general

$$M = log(\frac{V_1}{V_2}) = log(V_1) - log(V_2)$$

and

$$A = log(\frac{V_1 + V_2}{2})$$

Because this is such a standard plot again many differential expression packages provide convenience fuctions for producing them.

Diferential Expression

Once we have explored our data and hopefully removed as much unwanted variance as we can through normalisation we can then calculate the difference between the expression counts in the conditions and calculate a level of confidence about the result. This will normally be in the form of a p.value or p.value adjusted for multiple comparisons by some method.

In this workshop we will use the package edgeR (Robinson and Oshlack, 2010; McCarthy, Chen, and Smyth, 2012; Robinson and Smyth, 2007; Robinson and Smyth, 2008; Zhou, Lindsay, and Robinson, 2014) the process is generally very similar between packages, though they vary slightly in the data structures they uses as well as the normalisation steps and models they use.

```
require(edgeR)

# convert the count matrix into a format edgeR wants
# In this case what is called a DGEList object

# create a factor that says what each sample is here because the first character of the sample name
# distinquishes the sample type lets use that
cond <- as.factor(substr(colnames(countMatrix),1,1))

# create the DGEList object
dge <- DGEList(countMatrix,group= cond)
dge
```

```
An object of class "DGEList"
$counts
          Mut4 Mut5 Mut6  WT1  WT2  WT3
YDL248W   2549 2767 2471 2447 2663 2540
YDL247W-A   17   22   23   37   34   38
YDL247W    548  621  512  396  365  381
YDL246C    417  459  415  271  316  261
YDL245C    410  413  370  329  333  273
7121 more rows ...

$samples
     group lib.size norm.factors
Mut4     M 20959721            1
Mut5     M 20944139            1
Mut6     M 20383770            1
WT1      W 22462140            1
```

```
WT2      W 21967338            1
WT3      W 22154938            1
```

EdgeR provides several normalisation options through the `calcNormFactors()` the default option is to use one called Trimmed Mean of M-values (TMM) (Robinson and Oshlack, 2010)

```
dge <- calcNormFactors(dge, method="TMM")
dge
```

```
An object of class "DGEList"
$counts
          Mut4 Mut5 Mut6  WT1  WT2  WT3
YDL248W   2549 2767 2471 2447 2663 2540
YDL247W-A   17   22   23   37   34   38
YDL247W    548  621  512  396  365  381
YDL246C    417  459  415  271  316  261
YDL245C    410  413  370  329  333  273
7121 more rows ...

$samples
     group lib.size norm.factors
Mut4     M 20959721    1.1527364
Mut5     M 20944139    1.1534680
Mut6     M 20383770    1.1321977
WT1      W 22462140    0.8579882
WT2      W 21967338    0.8783654
WT3      W 22154938    0.8814256
```

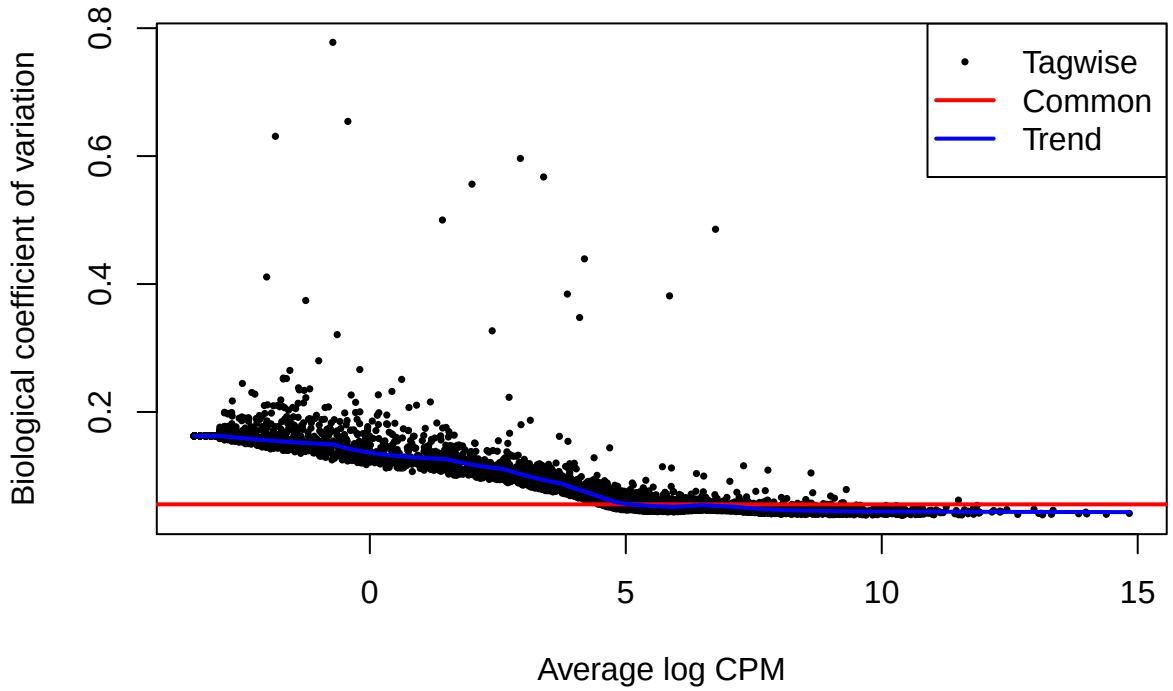
Once the normalisation has been done you need to create a model that describes the experimental design. Here we can use the `model.matrix()` function and the group factor to define our model

```
design <- model.matrix( ~dge$sample$group )
design
```

```
  (Intercept) dge$sample$groupW
1           1                 0
2           1                 0
3           1                 0
4           1                 1
5           1                 1
6           1                 1
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$`dge$sample$group`
[1] "contr.treatment"
```

Then we can call the edgeR commands to fit the model and find the top genes. The first task is to use the `estimateDisp()` function to estimate the dispersions that edgeR uses for its modelling. The common dispersion is the estimated dispersion over all the data, the trended dispersion is the dispersion for genes with similar expression and the tagwise dispersion is the individual gene dispersion. Thes can be visualised with the `plotBCV()` function

```
dge <- estimateDisp(dge, design)
plotBCV(dge, cex = 0.5)
```



We can then use the edgeR Generalised Linear Modelling fit `glmFit()` and calculate significance with the Log Ratio Test `glmLRT()` function. The `topTags()` function is then used to recover the results of the modelling.

```
fit <- glmFit(dge,design)
lrt <- glmLRT(fit,coef=2)
topTags(lrt)
```

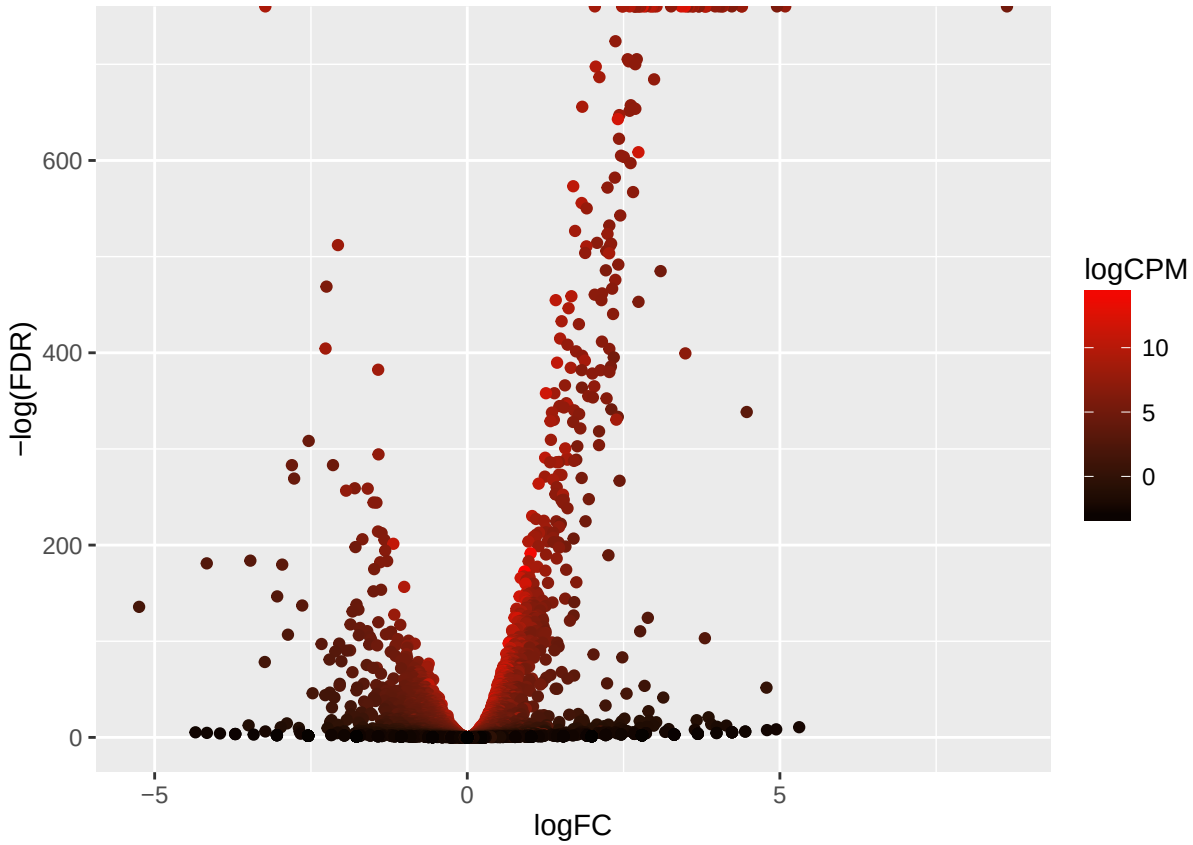
```
Coefficient:  dge$sample$groupW
             logFC    logCPM       LR PValue FDR
YOR290C   8.632233  5.111611 2992.133      0   0
YNR034W-A 5.085122  7.575412 4054.997      0   0
YHR136C   4.954126  5.815795 2576.039      0   0
YPR160W   4.393469 10.284363 5174.951      0   0
YOL052C-A 4.387435  7.746017 3443.786      0   0
YDR119W-A 4.231237  6.245736 1861.161      0   0
YML123C   4.088704  8.364057 3611.995      0   0
YBR054W   4.070385  7.593227 3512.584      0   0
YHR215W   4.028718  6.536383 2428.819      0   0
YIL121W   3.964813  7.353371 3170.554      0   0
```

Volcano plots

A standard plot for graphically examining the results of differential expression is the volcano plot that plots the significance against the fold change, often coloured by mean expression.

```
require(ggplot2)
tt <- topTags(lrt,n=10000)$table

ggplot(data=tt) + geom_point(aes(x=logFC,y=-log(FDR),color=logCPM)) +
  scale_colour_gradientn(colours=c("#000000" ,"#FF0000" ))
```



Exercise 3.3

Using the data in the saved featureCounts object /usr/local/share/BS32010/pschofield/Rdata/SCfcFull.Rdata perform a differential expression analysis with edgeR.

- Cluster the samples using pvclust.
- Try using a couple of alternative normalisations
  - Cluster the normalised gene counts, does this alter the clustering
  - Does this alter final list of significant genes
- What is the most highly differentially expressed gene
- What is the most highly up regulated gene

Bibliography

[1] M. I. Love, W. Huber and S. Anders. "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2". In: Genome Biology 15 (2014), p. 550.

[2] D. J. McCarthy, Y. Chen and G. K. Smyth. "Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation". In: Nucleic Acids Research 40 (2012).

[3] D. Risso, K. Schwartz, G. Sherlock, et al. "GC-content normalization for RNA-Seq data." In: BMC Bioinformatics 12 (2011), p. 480.

[4] M. E. Ritchie, B. Phipson, D. Wu, et al. "limma powers differential expression analyses for RNA-sequencing and microarray studies". In: Nucleic Acids Research 43 (2015).

[5] M. D. Robinson and A. Oshlack. "A scaling normalization method for differential expression analysis of RNA-seq data." In: Genome Biol 11.3 (2010), p. 25.

[6] M. D. Robinson and G. K. Smyth. "Moderated statistical tests for assessing differences in tag abundance". In: Bioinformatics 23 (2007).

[7] M. D. Robinson and G. K. Smyth. "Small-sample estimation of negative binomial dispersion, with applications to SAGE data". In: Biostatistics 9 (2008).

[8] R. Suzuki and H. Shimodaira. "Pvclust: an R package for assessing the uncertainty in hierarchical clustering". In: Bioinformatics 22 (2006), pp. 1540–1542.

[9] X. Zhou, H. Lindsay and M. D. Robinson. "Robustly detecting differential expression in RNA sequencing data using observation weights". In: Nucleic Acids Research 42 (2014).

Session Info

```
R version 3.2.3 (2015-12-10)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.11.3 (El Capitan)

locale:
[1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] ggplot2_2.0.0              edgeR_3.12.0
 [3] limma_3.26.8               pvclust_2.0-0
 [5] EDASeq_2.4.1               ShortRead_1.28.0
 [7] GenomicAlignments_1.6.3    SummarizedExperiment_1.0.2
 [9] Rsamtools_1.22.0           GenomicRanges_1.22.4
[11] GenomeInfoDb_1.6.3         Biostrings_2.38.4
[13] XVector_0.10.0             IRanges_2.4.8
[15] S4Vectors_0.8.11           BiocParallel_1.4.3
[17] Biobase_2.30.0             BiocGenerics_0.16.1
[19] knitr_1.12.3               RefManageR_0.10.6
[21] pietalib_0.1

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.3             locfit_1.5-9.1
 [3] lubridate_1.5.0         lattice_0.20-33
 [5] digest_0.6.9            plyr_1.8.3
 [7] futile.options_1.0.0    aroma.light_3.0.0
 [9] RSQLite_1.0.0           DESeq_1.22.1
```

```
[11] evaluate_0.8            zlibbioc_1.16.0
[13] GenomicFeatures_1.22.13 annotate_1.48.0
[15] R.utils_2.2.0           R.oo_1.20.0
[17] rmarkdown_0.9.5         labeling_0.3
[19] splines_3.2.3           geneplotter_1.48.0
[21] stringr_1.0.0           RCurl_1.95-4.7
[23] biomaRt_2.26.1          munsell_0.4.3
[25] rtracklayer_1.30.2      htmltools_0.3
[27] matrixStats_0.50.1      XML_3.98-1.3
[29] bitops_1.0-6            R.methodsS3_1.7.1
[31] grid_3.2.3              xtable_1.8-2
[33] gtable_0.2.0            DBI_0.3.1
[35] magrittr_1.5            formatR_1.2.1
[37] scales_0.4.0            bibtex_0.4.0
[39] stringi_1.0-1           hwriter_1.3.2
[41] genefilter_1.52.1       latticeExtra_0.6-28
[43] futile.logger_1.4.1     lambda.r_1.1.7
[45] RColorBrewer_1.1-2      tools_3.2.3
[47] RJSONIO_1.3-0           survival_2.38-3
[49] yaml_2.1.13             AnnotationDbi_1.32.3
[51] colorspace_1.2-6
```