COMP90014 – Assignment 2

Michael McLellan - 665968
Due Date: 19 October 2014

Code available:
https://github.com/mmclellan/COMP90014-Assign2-MichaelMcLellan

Introduction

This project involved carrying out a differential gene expression analysis on two different data sets of RNA-seq data and discuss and compare the results.

Task 1 - Differential gene expression analysis

For the following analyses the R package edgeR was used to find differential gene expressions between a male and female group with each group containing twenty replicants. Initial set up for the experiment is to load the data into R Studio and look at the data.

```
# Load in read in counts table.
alldata = read.table("assignment2data_fullcounts.tsv", header = TRUE, sep = "\t")
```

A good place to start for an analysis is too look at the data. In this analysis we look at the dimension of the data, showing how many genes and individuals we are working with. The output shows that there are 40 individuals and 44061 genes in the data set as we expected. We can see the data is also in the correct format for edgeR.

```
# Look at dimension, row and column names
dim(alldata)
```

[1] 44061 44

First step is to filter out genes with zeroes in both the male and female repliants. For this analysis it was decided to remove genes with zero counts if both the female and male samples have predominantly zeros. Using the apply function we filter out genes who's medians in both the male and female samples are zero. As a result 19443 genes were removed.

```
# Apply function to remove genes whos median is 0 for both male and female
# samples and view new dimension of filtered data set.
low_counts = apply(alldata[, 1:20], 1, median) == 0 & apply(alldata[, 21:40],
    1, median) == 0
filtered = alldata[!low_counts, ]

dim(filtered)

## [1] 24618    44
```

Next the data was split into two data frames. One containing the raw counts and the other containing the information for the gene information.

```r
# Split data into expression counts and gene information
expression = filtered[, 1:40]
geneInfo = filtered[, 41:44]
```

Now we start to use edgeR to perform a differential expression analysis on the data set. We are required to provide edgeR with the information telling which of the data belongs to each group. The standard way to do this is to use a design matrix, but as this is a simple two case comparison we only need to provide edgeR with a vector. In the 'ismale' vector, 0's represent the columns belonging to the female group while 1's being to the male group. Next we package all the counts into a list object called DGEList for edgeR. The DGEList function converts the count matrix into an edgeR object. The counts and information can still be accessed with in the DGE object.

```r
# Vector distinguishing which group the twe belong too.
ismale = c(rep(0, 20), rep(1, 20))
ismale
# Creating the DGE object
dge = DGEList(expression, group = ismale)
```

To take into account variation as a result of the different experiments from the replicates, edgeR's built in normalisation is used to scale each samples counts. By default a TMM (trimmed mean) normalisation is used.

```r
# Normalising the data.
dge = calcNormFactors(dge)
```

A good step in an analysis is to visualise the data. We can use a multi-dimensional scaling (MDS) plot to visualise the level of similarity of the two cases in the data. The basic version of a MDS plot is known as Principle coordinates analysis. It can clearly be seen that there is clustering between the male and female samples.

```r
plotMDS(dge, col = c(rep("red", 20), rep("blue", 20)))
```

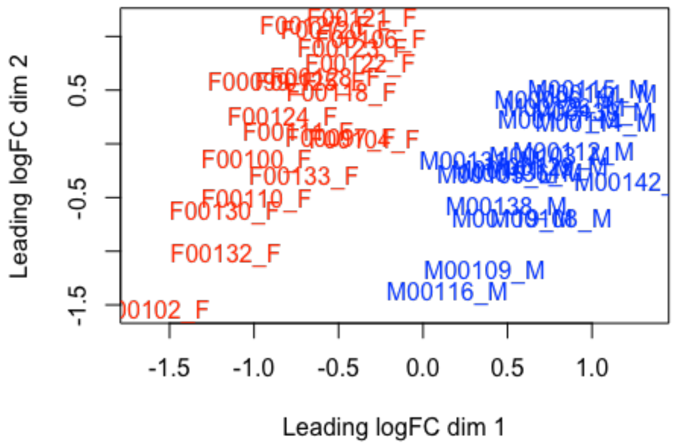

Figure 1: The multidimensional scaling plot of the high coverage RNA- data set. Shows the similarity between the individuals of the data set.

Next we model the variance of the data. Using edgeR we fit the negative binomial model to find the common dispersion for the data, so each gene has the same value for the dispersion. Next the per-gene estimates of variance are found using the `estimateTagewiseDisp()` function. The estimates are computed using an empirical Bayes method on weighted conditional maximum likelihood. The model uses the common dispersion as a prior when calculating the negative binomial dispersion preventing over fitting to the data. The BCV plot (biological coefficient of variance) plots the biological coefficient of variation (BCV) against gene abundance showing the common and tag wise BCV estimates.

```
# Estimate the common and tag wise dispersion, and view the relationship in
# a BCV plot
dge = estimateCommonDisp(dge)
dge$common.dispersion

## [1] 0.1242

dge = estimateTagwiseDisp(dge)
plotBCV(dge)
```

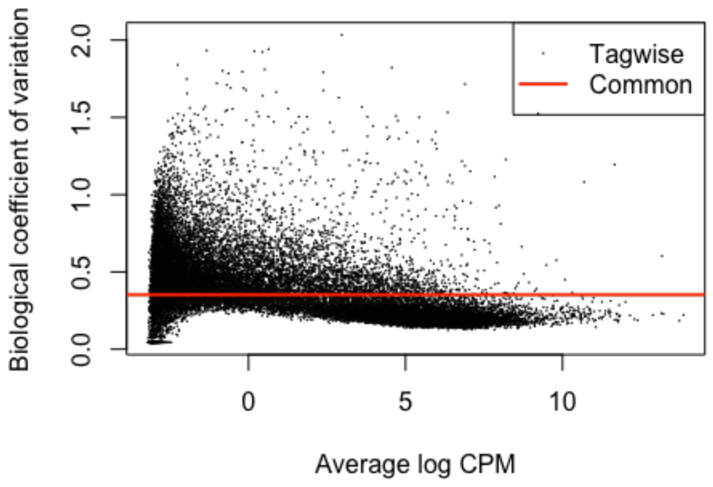

Figure 2: The BCV plot shows the common dispersion as the red line and the tag wise estimate as the black dots

By plotting the mean-variance relationship we are able to see how well the data fits to the estimate dispersion parameters. The grey dots are the raw variance of the counts, blue dots are the tag wise dispersion and the solid blue line is the variance using the common dispersion. The straight line is the mean variance relationship of a Poisson distribution. The plot models the mean variance relationship of the estimated dispersion.

```
plotMeanVar(dge, show.raw.vars = TRUE, show.tagwise.vars = TRUE, show.binned.commo
n.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 Scal
e)")
```

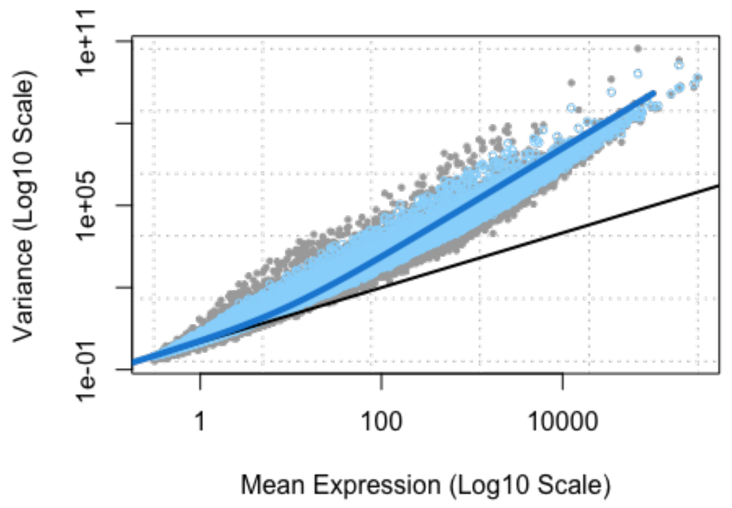

Figure 3: The grey dots are the raw variance of the counts, blue dots are the tag wise dispersion estimates and the solid blue line is common dispersion. The straight line is the mean variance relationship of a Poisson distribution

To test for DGE we perform pair-wise tests between the male and female groups. The exact test looks for statistically significant differences between the means of the negative binomial distribution fitted to the data. For this test the null hypothesis is that there is no differential expression, with Type 1 error being we declare there is differential expression when there is not and type 2 errors being we fail to detect the differential expression.

The results are stored in an edgeR data class 'DGEExact' with each row containing the gene with its logFC value, logCPM and P Value from the test. For this test the null hypothesis is that there is no differential expression. The exact test is looking for

```
# Perform an exact test on the data and view the structure of the results
results = exactTest(dge)
str(results)
```

We can view the top ten hits from the results using the `topTags()`. We can then refer back to the data frame containing the information for each gene and find the information for each differential expressed gene.

```
# Top ten differentially expressed genes show with gene information
topGI = merge(geneInfo[rownames(topTen), ], topTags(results, n = 10), by = 0,
    sort = FALSE)
del = c("Row.names", "start_position", "FDR")  #Columns not needed in output
topGI[, !(colnames(topGI) %in% del), drop = FALSE]

##    gene_name chromosome_name   gene_biotype  logFC logCPM PValue
## 1     RPS4Y1               Y protein_coding 10.357  6.617      0
## 2    TXLNG2P               Y     pseudogene 10.155  5.356      0
## 3      KDM5D               Y protein_coding 10.005  5.107      0
## 4      DDX3Y               Y protein_coding  9.992  6.432      0
## 5        UTY               Y protein_coding  9.950  4.217      0
## 6       XIST               X        lincRNA -9.856  8.121      0
## 7      USP9Y               Y protein_coding  9.840  4.644      0
## 8     EIF1AY               Y protein_coding  9.098  5.284      0
## 9        ZFY               Y protein_coding  9.023  3.206      0
## 10    TTTY15               Y        lincRNA  9.019  2.866      0
```

To view the bottom ten results we can get the full list from the `topTags()` command and then use the `tail()` function in R to select the bottom ten rows of the data frame. For this the bottom ten results were interpreted to be the the bottom of the full list of genes.

```
# Bottom 10 genes from the data set after being sorted by the topTags
# function.
low = topTags(results, n = nrow(results$table))$table
low_GI = merge(geneInfo[rownames(low), ], tail(low, n = 10), by = 0, sort = FALSE)
low_GI[, !(colnames(low_GI) %in% del), drop = FALSE]
```

```
##       gene_name chromosome_name   gene_biotype      logFC logCPM PValue
## 1       PLEKHF2               8 protein_coding -2.143e-04  6.015      1
## 2   RP13-578N3.3              4      antisense -1.204e-04 -2.989      1
## 3        AMOTL1              11 protein_coding  8.016e-05  4.085      1
## 4       NDUFAF3               3 protein_coding  5.758e-05  5.511      1
## 5         OXSR1               3 protein_coding -4.336e-05  7.236      1
## 6         PPP4C              16 protein_coding -4.206e-05  6.693      1
## 7          EIF5              14 protein_coding -3.152e-05  7.959      1
## 8       FAM131B               7 protein_coding -3.136e-05 -3.066      1
## 9         GSKIP              14 protein_coding -1.843e-05  4.998      1
## 10        CENPB              20 protein_coding -5.929e-06  5.805      1
```

The `decideTestsDGE()` function will implement multiple testing procedures to determine whether a series of differential expression statistics are up, down or not significant. The functions determines if each log-fold change in the results matrix should be considered significantly different from zero. The summary output shows the number of genes whose fold change is up or down.

```
de = decideTestsDGE(results)
summary(de)
```

```
##    [,1]
## -1    61
## 0  24512
## 1     45
```

For visualisation a heatmap is a common way to view expression data. A heatmap is a two-dimensional grid with different colours representing expression values. For this case we will use the top twenty differential expressed genes in the heatmap. The heatmap shows clear clusters of differential expressed genes.

```
# Produce heat map
topgenes = rownames(topTags(results, n = 20))
log.dge = cpm(dge, prior.count = 2, log = TRUE)
topgene.log.expression = log.dge[topgenes, ]
redtogreen = colorRampPalette(c("red", "yellow", "green"))(n = 299)
heatmap(topgene.log.expression, col = redtogreen)
```

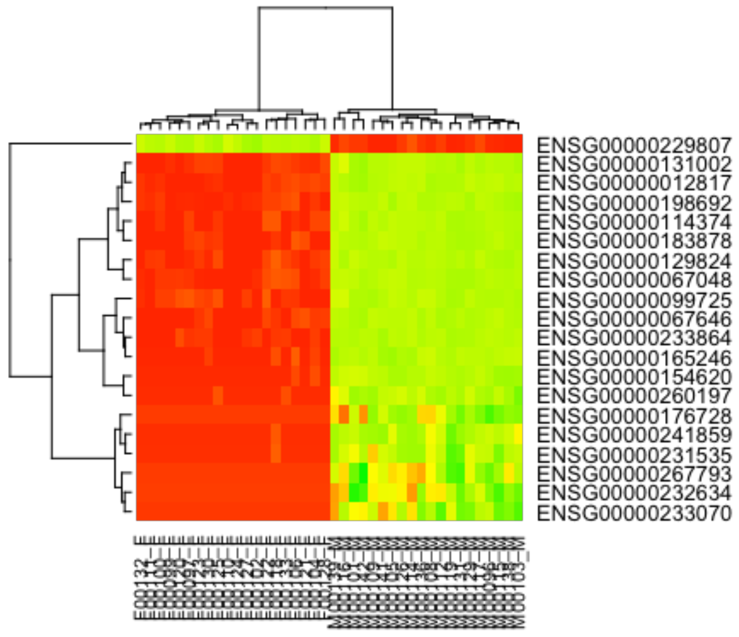

Figure 4: Heat map for the top 20 genes with differential expression

A second common plot is a histogram plot showing the distribution of P-values. We are expecting to see the distribution to be largely uniform.

```
hist(results$table$PValue, breaks = 150)
```

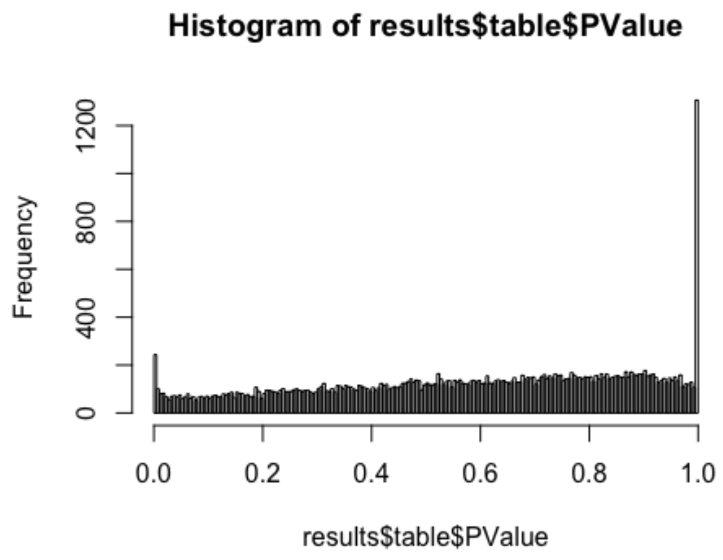

Figure 5: Histogram of the P values from the results table

Another useful plot is a smear plot that shows the relationship between concentration and fold change across the genes. The DE genes are coloured red while non-DE genes are coloured black. The orange dots represent genes with zero counts. The blue lines represent two times fold change in expression which can be viewed as biologically significant.

```
# Smear plot
resultsTbl = topTags(results, n = nrow(results$table))$table
de.genes = rownames(resultsTbl)[resultsTbl$PValue <= 0.05]
plotSmear(results, de.tags = de.genes)
abline(h = c(-2, 2), col = "blue")
```

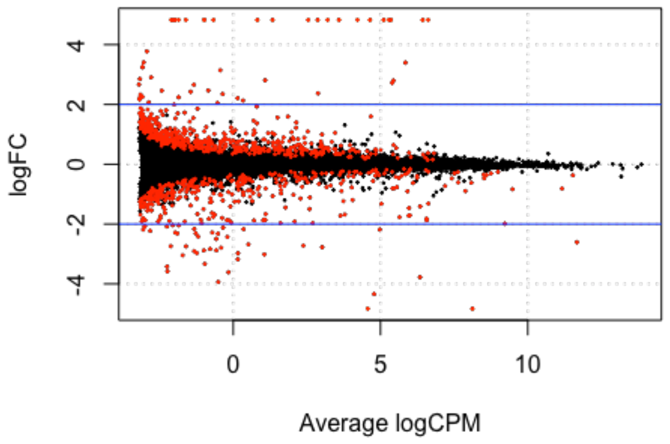

Figure 6: Smear plot showing the fold changes for genes

Task 1 Numerical Questions

1. Number of genes with p-values below 0.05, What proportion of the genes tested?

```
nrow(results$table[results$table$PValue < 0.05, ])

## [1] 912

(nrow(results$table[results$table$PValue < 0.05, ])/nrow(results$table))

## [1] 0.03705
```

2. Out of all genes on Y chromosome, how many have a P-value below 0.05?

```
significant = results$table[results$table$PValue < 0.05, ]
significantGI = geneInfo[rownames(significant), ]
significantY = geneInfo[geneInfo$chromosome_name == "Y", ]
nrow(significantY)

## [1] 36
```

3. Out of the 100 genes with the lowest P-value, how many are from the X chromosome?

```
top100 = topTags(results, n = 100)
significantX = geneInfo[rownames(top100), ]
nrow(significantX[significantX$chromosome_name == "X", ])

## [1] 22
```

4. What is the log-fold change for the gene XIST?

```
xist = geneInfo[geneInfo$gene_name == "XIST", ]
xistResult = results$table[rownames(results$table) == rownames(xist), ]
xistResult$logFC

## [1] -9.856
```

Task 2 - DGE analysis of subsampled dataset

For the second task a differential expression analysis on a sub sample of the data set. The sub sample represents a lower coverage RNA-seq experiment, with all the same genes and replicants but with fewer counts. The same workflow was followed as task one.

The sub sample data set was loaded, genes with zero counts removed as well as splitting the data into a data frame of counts and data frame of gene information. For this data set a total of 29458 genes were removed.

```r
# Read in data set
sub.Data = read.table("assignment2data_subsampled_665968.tsv", header = TRUE,
    sep = "\t")
dim(sub.Data)
```

```
## [1] 44061    44
```

```r
# remove genes with 0 counts in male and female
sub.low.counts = apply(sub.Data[, 1:20], 1, median) == 0 & apply(sub.Data[,
    21:40], 1, median) == 0
sub.filtered = sub.Data[!sub.low.counts, ]

# split data
sub.expression = sub.filtered[, 1:40]
sub.geneInfo = sub.filtered[, 41:44]
```

The DGE object was created for edgeR and normalised. The MDS plot still shows separation between the male and female groups.

```r
# create dge object, using same group vector from task 1
sub.dge = DGEList(sub.expression, group = ismale)
# normalise
sub.dge = calcNormFactors(sub.dge)
# plot MDS plot
plotMDS(sub.dge, col = c(rep("red", 20), rep("blue", 20)))
```

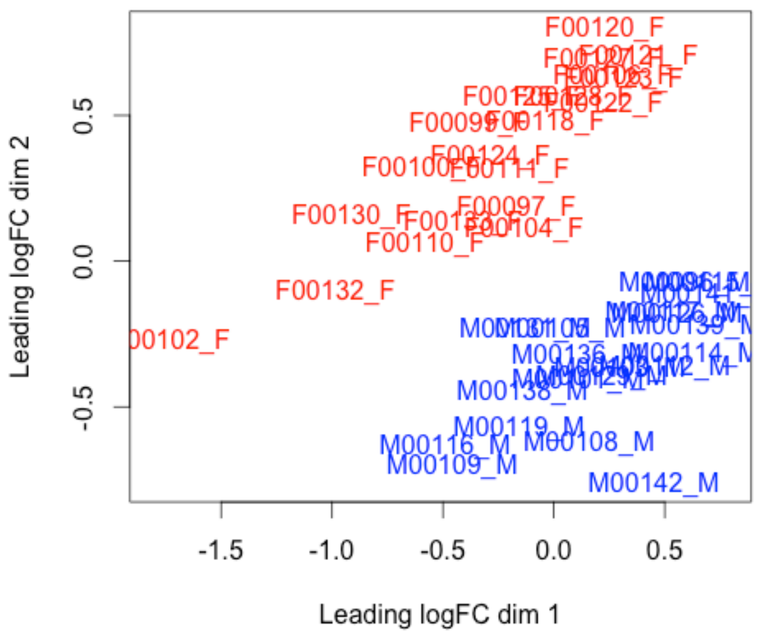

Figure 7: MDS plot for lower depth data set. Shows clear clustering of male and female groups

The common and tag wise dispersion are estimated for the data set and the are viewed using the `plotBCV()` function.

```
# fit models
sub.dge = estimateCommonDisp(sub.dge)
sub.dge = estimateTagwiseDisp(sub.dge)
plotBCV(sub.dge)
```

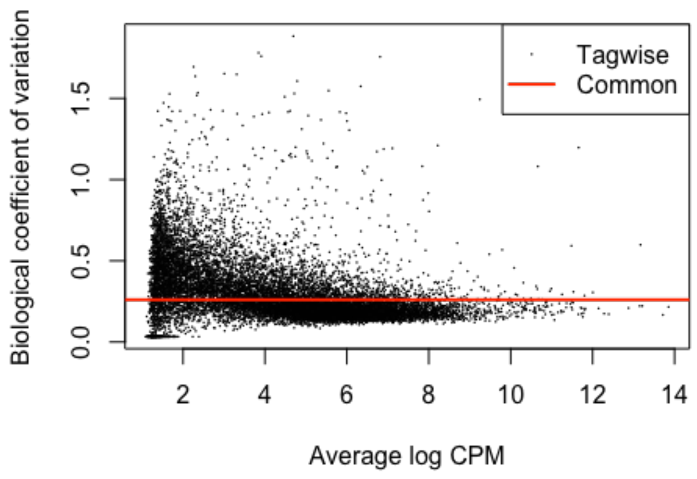

Figure 8: BCV plot for the lower depth data set

To test for DGE in the lower coverage data set, the exact test was used to look for statistically significant differences between the means of the negative binomial distribution fitted to the data.

```
sub.results = exactTest(sub.dge)
```

The `plotMeanVar()` function was used to visualise how the data is fitted to the models.

```
# plot mean variance plot to show how data fits to the model
plotMeanVar(sub.dge, show.raw.vars = TRUE, show.tagwise.vars = TRUE, show.binned.c
ommon.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 Scal
e)")
```

Using the same methods as Task 1 we are able to find the top ten differential expressed genes as well as the bottom 10 genes. Using the `decideTestsDGE()` function again we can view the number of genes that have significantly different fold changes.

```
# Sub top ten hits
sub.topGI = merge(sub.geneInfo[rownames(sub.topTen), ], topTags(sub.results,
    n = 10), by = 0, sort = FALSE)
sub.topGI[, !(colnames(sub.topGI) %in% del), drop = FALSE]

##    gene_name chromosome_name   gene_biotype  logFC logCPM     PValue
## 1      DDX3Y               Y protein_coding 10.008  6.441  0.000e+00
## 2     RPS4Y1               Y protein_coding  9.090  6.643  0.000e+00
## 3       XIST               X        lincRNA -9.800  8.096 3.281e-293
## 4      EIF1AY               Y protein_coding  7.748  5.346 3.970e-212
## 5       KDM5D               Y protein_coding  7.574  5.181 3.376e-191
## 6       USP9Y               Y protein_coding  8.811  4.817 4.047e-175
## 7     TXLNG2P               Y     pseudogene  9.472  5.440 1.426e-165
## 8         UTY               Y protein_coding  7.858  4.406 1.802e-129
## 9        PRKY               Y     pseudogene  6.207  3.758 3.586e-104
## 10        ZFY               Y protein_coding  6.519  3.560  2.014e-88
```

```
# Sub bottom 10 hits
sub.low = topTags(sub.results, n = nrow(sub.results$table))$table
sub.low.GI = merge(sub.geneInfo[rownames(sub.low), ], tail(sub.low, n = 10),
    by = 0, sort = FALSE)
sub.low.GI[, !(colnames(sub.low.GI) %in% del), drop = FALSE]
```

```
##    gene_name chromosome_name   gene_biotype      logFC logCPM PValue
## 1      CPNE5               6 protein_coding -3.566e-04  5.026      1
## 2      LIMK2              22 protein_coding  3.518e-04  5.594      1
## 3       BBC3              19 protein_coding  3.085e-04  4.003      1
## 4       ALG1              16 protein_coding -3.084e-04  4.099      1
## 5      SPRY4               5 protein_coding -2.478e-04  3.614      1
## 6       COCH              14 protein_coding  2.467e-04  3.516      1
## 7     SFMBT2              10 protein_coding  2.005e-04  6.274      1
## 8    MTERFD1               8 protein_coding -1.165e-04  5.081      1
## 9     GOLT1B              12 protein_coding -4.951e-05  5.661      1
## 10     UBE2O              17 protein_coding  4.757e-05  6.239      1
```

```
# Summary of up/down regulaton genes, tagwise results
sub.de = decideTestsDGE(sub.results)
summary(sub.de)
```

```
##    [,1]
## -1    26
## 0  14557
## 1     20
```

```
sub.detags = rownames(sub.dge)[as.logical(sub.de)]
```

The heat map shows still clear differences in the expressions between the groups but blocks are less clear and well defined.

```
# heatmap
sub.topgenes = rownames(topTags(sub.results, n = 20))
sub.log.dge = cpm(sub.dge, prior.count = 2, log = TRUE)
sub.topgene.log.expression = sub.log.dge[sub.topgenes, ]
heatmap(sub.topgene.log.expression, col = redtogreen)
```

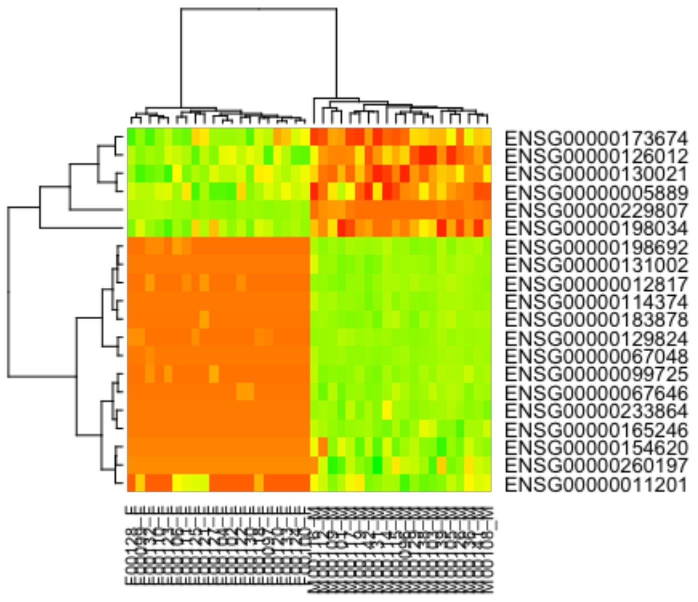

Figure 9: Heatmap for the lower coverage data set.

Histogram showing the distribution of the P values for the sub sampled data set.

```r
# histogram
hist(sub.results$table$PValue, breaks = 150)
```

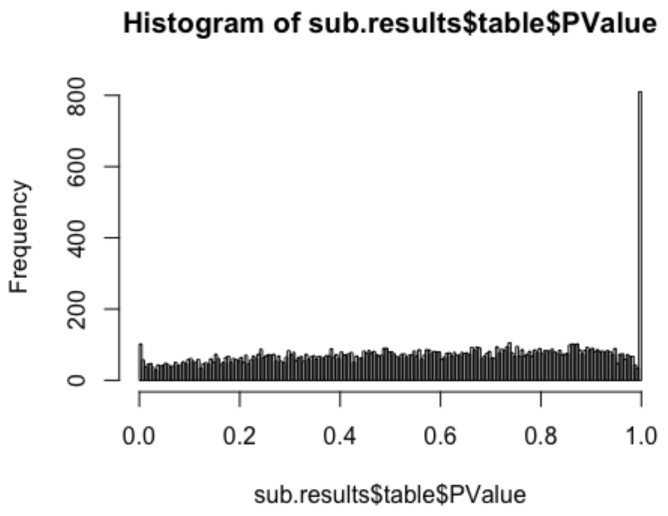

Figure 10: Histogram of P Values of lower depth data set

Smear plot showing the fold changes for the genes in the sub sampled data set.

```r
sub.resultsTbl = topTags(sub.results, n = nrow(sub.results$table))$table
sub.de.genes = rownames(sub.resultsTbl)[sub.resultsTbl$PValue <= 0.05]
plotSmear(sub.results, de.tags = sub.de.genes)
abline(h = c(-2, 2), col = "blue")
```

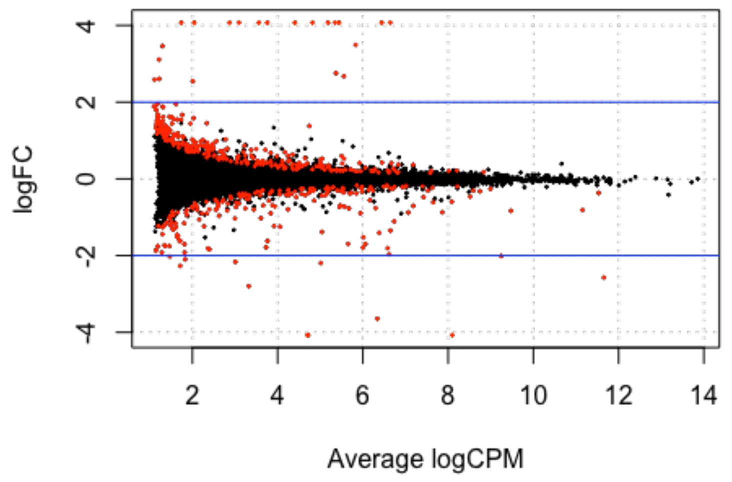

Figure 11: Smear plot showing fold changes for lower depth data set

Task 2: Numerical Questions

1. Number of genes with p-values below 0.05, What proportion of the genes tested?

```
nrow(sub.results$table[sub.results$table$PValue < 0.05, ])

## [1] 478

(nrow(sub.results$table[sub.results$table$PValue < 0.05, ])/nrow(sub.results$table
))

## [1] 0.03273
```

2. Out of all genes on Y chromosome, how many have a P-value below 0.05?

```
sub.sig = sub.results$table[sub.results$table$PValue < 0.05, ]
sub.sigGI = sub.geneInfo[rownames(sub.sig), ]
sub.sigY = sub.sigGI[sub.sigGI$chromosome_name == "Y", ]
nrow(sub.sigY)

## [1] 17
```

3. Out of the 100 genes with the lowest P-value, how many are from the X chromosome?

```
sub.top100 = topTags(sub.results, n = 100)
sub.sigX = sub.geneInfo[rownames(sub.top100), ]
nrow(sub.sigX[sub.sigX$chromosome_name == "X", ])

## [1] 26
```

4. What is the log-fold change for the gene XIST?

```
subxist = sub.geneInfo[sub.geneInfo$gene_name == "XIST", ]
sub.xistResult = sub.results$table[rownames(sub.results$table) == rownames(xist),
    ]
round(sub.xistResult$logFC, digits = 4)

## [1] -9.8
```

Task 3: Randomised sub sample set

The third task is to perform a sanity check and to be sure that our analysis does not contain any obvious errors or bias. As there is no reason to suspect significant differential expression between two randomly assigned groups we group individuals at random as opposed to their gender. Here we should see no differential expression between the two groups if our analysis is correct.

To assign the groups randomly we use R to create a vector or random 0's and 1's. The set seed function ensures that the results here are reproducable.

```
# create vector of random 0 and 1
seed = 1234
set.seed(seed)
randomgroup = sample(c(0, 1), 40, replace = TRUE)

randomgroup

##  [1] 0 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0
## [36] 1 0 0 1 1
```

The same analysis is performed as Task 2 on the data. The only difference is that we are now using the randomly assigned groups.

```
# create dge object
rand.dge = DGEList(sub.expression, group = randomgroup)
# normalise
rand.dge = calcNormFactors(rand.dge)
```

As there is still similarity between the male and females there are still clustered together in the MDS plot.

```
# plot MDS plot
plotMDS(rand.dge, col = c(rep("red", 20), rep("blue", 20)))
```

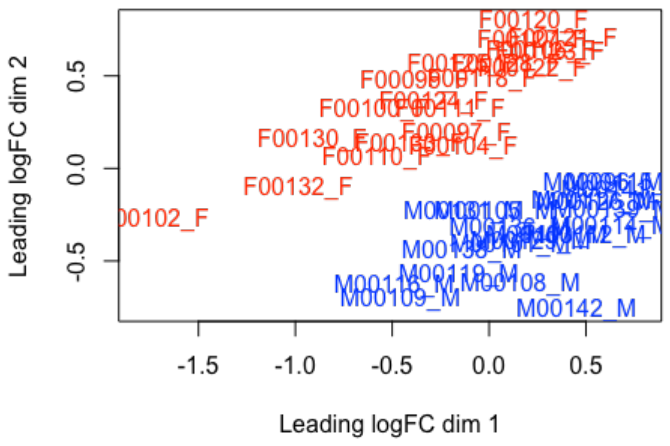

Figure 12: MDS plot for random assigned group

```
# fit models
rand.dge = estimateCommonDisp(rand.dge)
rand.dge = estimateTagwiseDisp(rand.dge)
plotBCV(rand.dge)
```

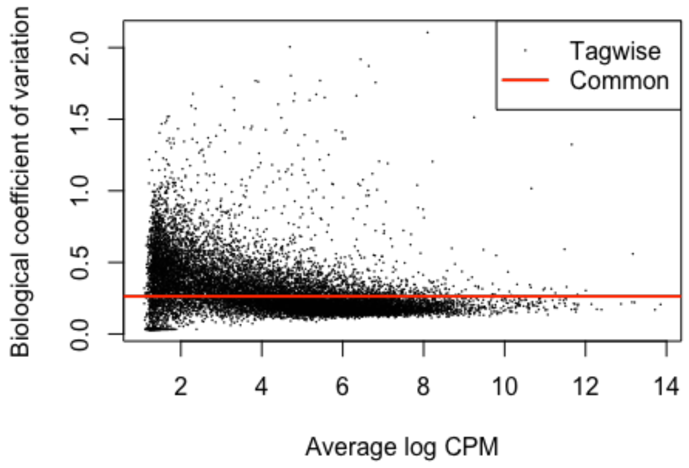

Figure 13: BCV plot for the randomly assigned groups

```
rand.results = exactTest(rand.dge)

# plot mean variance plot to show how data fits to the model
plotMeanVar(rand.dge, 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 Scal
e)")
```

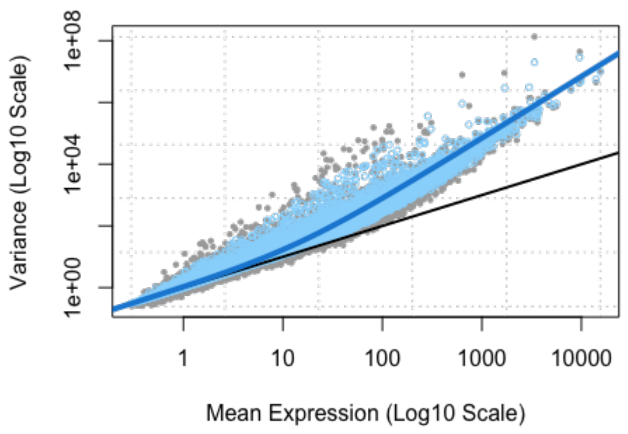

Figure 14: Mean variance plot for randomly assigned group

As the groups were assigned at random there are still some differential expressed genes.

```
# get top ten hits
rand.topGI = merge(sub.geneInfo[rownames(rand.topTen), ], topTags(rand.results,
    n = 10), by = 0, sort = FALSE)
rand.topGI[, !(colnames(rand.topGI) %in% del), drop = FALSE]

##    gene_name chromosome_name   gene_biotype   logFC logCPM    PValue
## 1    IGHV6-1              14      IG_V_gene -3.6094  5.827 1.352e-06
## 2   IGKV2-30               2      IG_V_gene -2.5934  5.017 6.891e-06
## 3   IGKV2-24               2      IG_V_gene -3.0829  5.368 7.342e-06
## 4   IGKV1-12               2      IG_V_gene  2.1819  4.630 7.892e-06
## 5   IGLV2-18              22      IG_V_gene -4.5077  4.694 5.196e-05
## 6      HYOU1              11 protein_coding -0.4391  8.015 9.444e-05
## 7   IGHV3-73              14      IG_V_gene -2.4655  3.933 1.328e-04
## 8       AARS              16 protein_coding -0.3665  8.074 1.381e-04
## 9     SPRYD3              12 protein_coding -0.3746  5.742 1.707e-04
## 10    CCRN4L               4 protein_coding -0.6480  3.531 1.962e-04

# bottom 10 hits
rand.low = topTags(rand.results, n = nrow(rand.results$table))$table
rand.lowGI = merge(sub.geneInfo[rownames(rand.low), ], tail(rand.low, n = 10),
    by = 0, sort = FALSE)
rand.lowGI[, !(colnames(rand.lowGI) %in% del), drop = FALSE]

##    gene_name chromosome_name   gene_biotype      logFC logCPM PValue
## 1      BCMO1              16 protein_coding  2.818e-04  2.134      1
## 2     LRRIQ3               1 protein_coding -2.620e-04  2.395      1
## 3     FCHSD1               5 protein_coding  2.581e-04  4.537      1
## 4      FBXW9              19 protein_coding  2.406e-04  3.063      1
## 5      SPG11              15 protein_coding -2.184e-04  6.760      1
## 6    C7orf43                  7 protein_coding -1.933e-04  3.769      1
## 7      TOR2A                  9 protein_coding  1.112e-04  3.550      1
## 8       RBM7                 11 protein_coding -8.416e-05  4.591      1
## 9      WDR73                 15 protein_coding -7.019e-05  4.255      1
## 10     TRMT5                 14 protein_coding  1.827e-05  4.888      1
```

```
# Summary of up/down regulaton genes, tagwise results
rand.de = decideTestsDGE(rand.results)
summary(rand.de)
```

```
##    [,1]
## -1     3
## 0  14599
## 1      1
```

```
rand.detags = rownames(rand.dge)[as.logical(rand.de)]
```

Visualisation

As expected there is no real clustering between the differential expressed genes. The heat map looks to be uniform and random.

```
# heatmap
rand.topgenes = rownames(topTags(rand.results, n = 20))
rand.log.dge = cpm(rand.dge, prior.count = 2, log = TRUE)
rand.topgene.log.expression <- rand.log.dge[rand.topgenes, ]
heatmap(rand.topgene.log.expression)
```

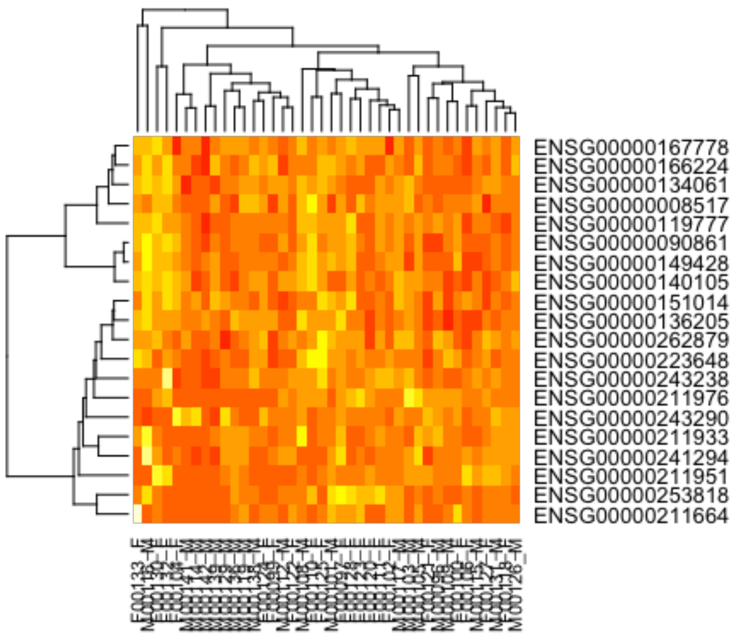

Figure 15: Heat map for the randomly assigned group

There are still peaks in the histogram but there are viewer. Over all the histogram is smaller and looks more uniform.

```r
# histogram
hist(rand.results$table$PValue, breaks = 150)
```

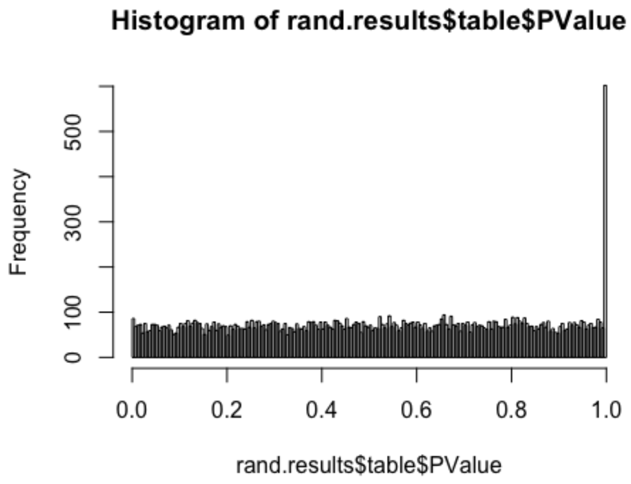

Figure 16: Histogram for the randomly assigned groups

The smear plot also shows few genes that are differential expressed with large fold changes. The summary from before also only shows three differential expressed genes with significant fold changes.

```r
# smear plot - shows the relationship between concentraiton and the
# fold-change across the genes.
rand.resultsTbl = topTags(rand.results, n = nrow(rand.results$table))$table
rand.de.genes = rownames(rand.resultsTbl)[rand.resultsTbl$PValue <= 0.05]
plotSmear(sub.results, de.tags = rand.de.genes)
abline(h = c(-2, 2), col = "blue")
```

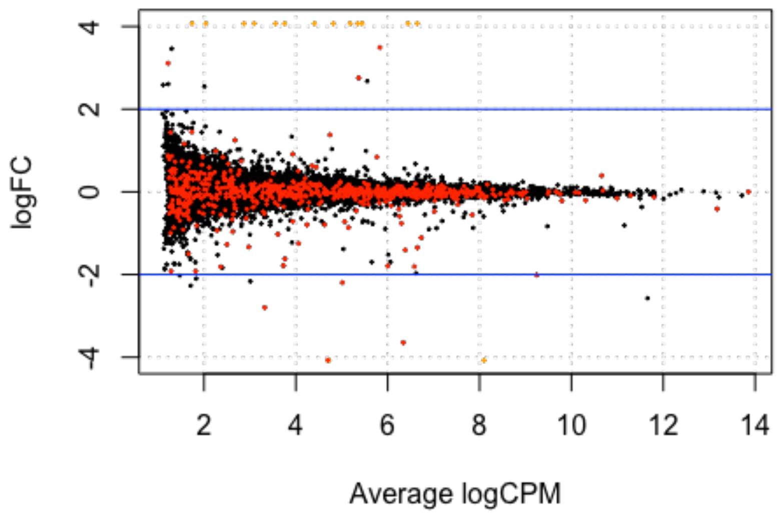

Figure 17: Smear plot for the randomly assigned groups

Task 3 Numerical Questons

1. Number of genes with p-values below 0.05, What proportion of the genes tested?

```
nrow(rand.results$table[rand.results$table$PValue < 0.05, ])

## [1] 689

(nrow(rand.results$table[rand.results$table$PValue < 0.05, ])/nrow(rand.results$ta
ble))

## [1] 0.04718
```

2. Out of all genes on Y chromosome, how many have a P-value below 0.05?

```
rand.sig = rand.results$table[rand.results$table$PValue < 0.05, ]
rand.sigGI = sub.geneInfo[rownames(rand.sig), ]
rand.sigY = rand.sigGI[rand.sigGI$chromosome_name == "Y", ]
nrow(rand.sigY)

## [1] 1
```

3. Out of the 100 genes with the lowest P-value, how many are from the X chromosome?

```
rand.top100 = topTags(sub.results, n = 100)
rand.sigX = sub.geneInfo[rownames(rand.top100), ]
nrow(rand.sigX[rand.sigX$chromosome_name == "X", ])

## [1] 26
```

4. What is the log-fold change for the gene XIST?

```
rand.xist = sub.geneInfo[sub.geneInfo$gene_name == "XIST", ]
rand.xist.Result = rand.results$table[rownames(rand.results$table) == rownames(xis
t),
    ]
rand.xist.Result$logFC
```

```
## [1] 0.6898
```

Task 4: Discussion

The purpose of this analysis was to find genes in our data set that were differential expressed between the two groups. As expected we found a number of differential expressed genes that are located on the Y chromosome.

Comparison of Task 1 and 2

The first task involved using high-depth RNA-seq data and the second involved the same groups but lower depth. The lower depth meant meant less statistical power for detecting differential expression. The first process of filtering counts we removed an extra 10,015 genes from the analysis, significantly reducing statistical power.

Comparing the two MDS plots for the higher and lower depth RNA-seq data we can clearly see in both the clustering between the male and female groups. So reduced depth does not effect the similarity between the samples.

The dispersion estimates for the two data sets are very similar. Looking at the BCV plots for the two data sets they appear to be very similar the major difference being the number of tag wise estimates (as more genes were removed). Looking at the raw value for the common dispersion they are also very close. The common dispersion for high depth was and for lower depth .

As expected the genes in the top ten differential expressed list for both data sets are slightly different. As the counts are different slightly different results are given but common genes are seen in both lists. The same applies to the lowest DE genes.

Looking at the `decideTestsDGE()` results we can see there are fewer genes that are significantly DE. The higher coverage with more power has found more genes both negatively and positively DE. The heat maps show clear differential expression between the positively and negatively expressed genes in the higher depth compared to the lower depth. This is expected as we have removed genes reducing the power of our experiment.

The comparison of histograms show the frequency of P value has reduced, this is again due to reduced number of genes in the lower coverage data set and the lower power of the analysis. Looking at the numerical analysis we are still detecting the same proportion of genes with P Value below 0.05. The log-fold change of the xist gene is also near the same value.

Comparison of Task 2 and Task 3

For Task 3 we performed some sanity checking by randomising the group assignment of the replicants. This in theory should show that there is no differential expression between the two groups, proving that our first analysis was working and detecting real differential expression. How ever this is random so we still do detect some genes with differential expression.

The MDS plot still shows clear similarity between the male and female samples despite being assigned to groups randomly. The random assignments does not change the similarity between the samples.

The comparison of heat maps clearly show the lack of differential expression between the randomly assigned groups. With no clear clustering of genes, everything appears uniform. With the output of the `decideTestsDGE()` from edgeR there are only three genes with differential expression found compared to 46 from task 2.The smear plot also only shows the few genes that are above or below the 2x fold change lines on the graph.

As expected we have fewer genes that have a P Value lower then 0.05 as we are not detecting many differential expressed genes. Looking at the difference in log-fold changes in the xist gene between the two tasks. The log-fold change in Task 2 is -9.8, where in Task 3 it is 0.6323 showing the lack of detection of DE in randomly assigned groups.

R session info

```
# Session info
sessionInfo()
```

```
## R version 3.1.1 (2014-07-10)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
##
## locale:
## [1] en_AU.UTF-8/en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.6    edgeR_3.6.8  limma_3.20.9
##
## loaded via a namespace (and not attached):
## [1] digest_0.6.4    evaluate_0.5.5  formatR_1.0     htmltools_0.2.6
## [5] rmarkdown_0.3.3 stringr_0.6.2   tools_3.1.1     yaml_2.1.13
```