COMP90014 - Assignment2

Michael McLellan - 665968

Due Date; 17 October 2014

Task 1 - Differential gene expression analysis

For the following analyses the R package edgeR was used to find differentially expressed genes between a male and female group with each group containing twenty replicants. Initial set up for the experiment is to load the data into RStudio and look at the data. The data is an RNA-Seq data set containing only the counts with the row names as the gene ids and the column names as sample ids.

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

A good place to start for an analyses is too look at the ddata. 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. 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 majoritively zeros. Using the apply function we filter out genes whos 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 dimensio
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.

```
#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. The ismale vector, 0's represent the columns belonging to the female group while 1's beong to the male group. Next we package all the counts into a list object called DGEList for edgeR. The DGEList function converst the count matrix into an edgeR object. The counts and information can still be accessed with in the DGE object.

```
#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.

```
#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 groups.

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

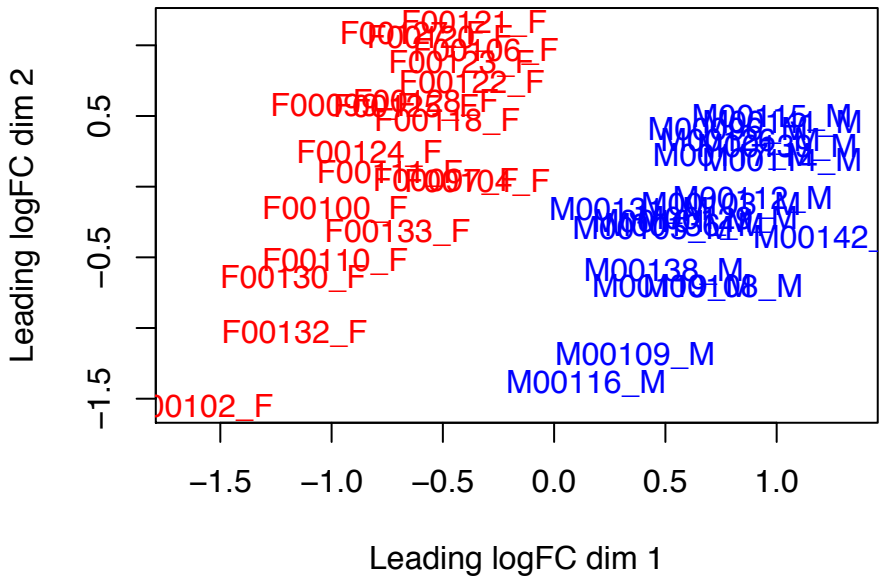

Figure 1: The multidimensional scaling plot is a way to visualise the similarity between the individuals in the data set. We can clearly see the male and females clustering into seperate groups.

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 binomal dispersions preventing overfitting to the data. The BCV plot (biological coefficient of variance) plots the biological coefficient of variation (BCV) against gene abundance showing the common the and tagwise 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)
```

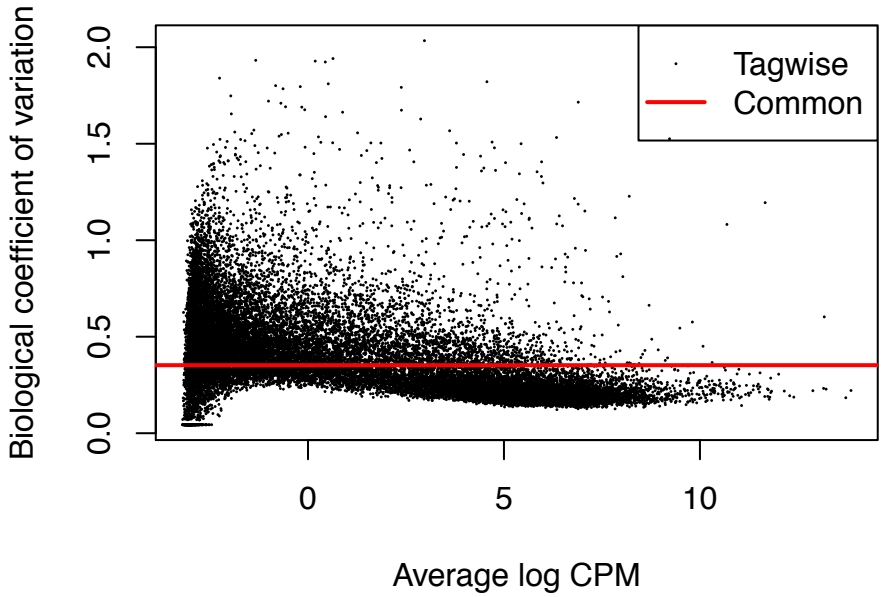

Figure 2: The BCV plot shows the common dispersion as the red line and the tagwise estimates as the black dots.

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 dispersions 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 dispersions. The relatipnship is not linear but is probably quadratic.

```
plotMeanVar(dge, show.raw.vars=TRUE, show.tagwise.vars=TRUE, show.binned.common.disp.vars=FALSE, show.a
```

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.The results are stored in an edgeR data class 'DGEExact' with each row containing the gene with its logFC value, logCPM and PValue from the test.

```
#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 tags from the results, ie the top ten differentially expressed genes between the two groups. We can then refer back to the data frame containing the information for each gene and find the information for each differentially 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]
```

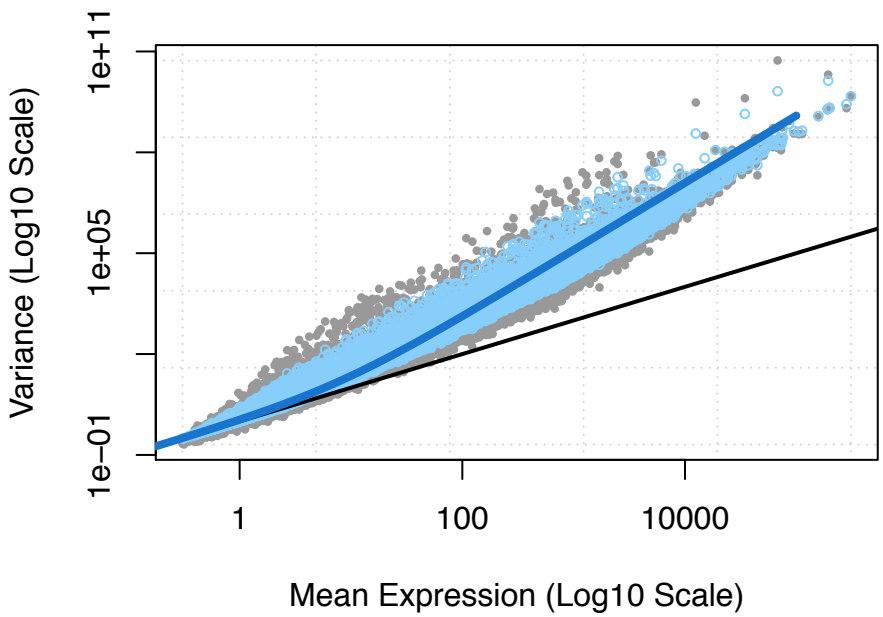

Figure 3: The grey dots are the raw variance of the counts, blue dots are the tag wise dispersions and the solid blue line is the variance using the common dispersion.

```
##    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 interpretted to be the the bottom of the list of genes after being sorted.

```
#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 wehter 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 differentially expressed genes in the heatmap. The heatmap shows clear areas of different levels of expression.

```
#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)
```

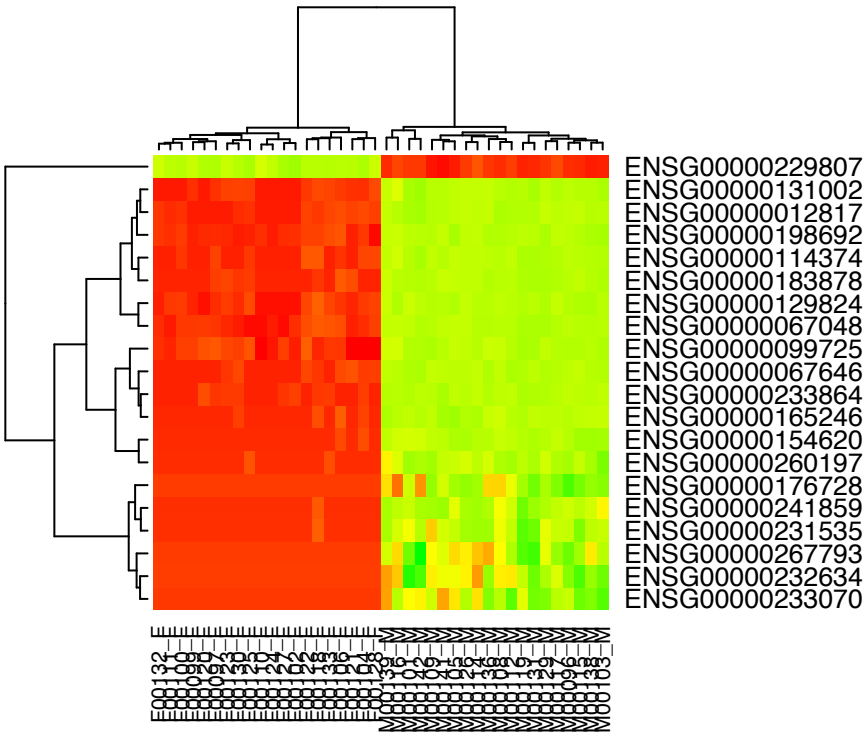

Figure 4: Heat map for the top 20 differentially expressed genes

A second common plot is a histogram plot showing the distribution of P-values.

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

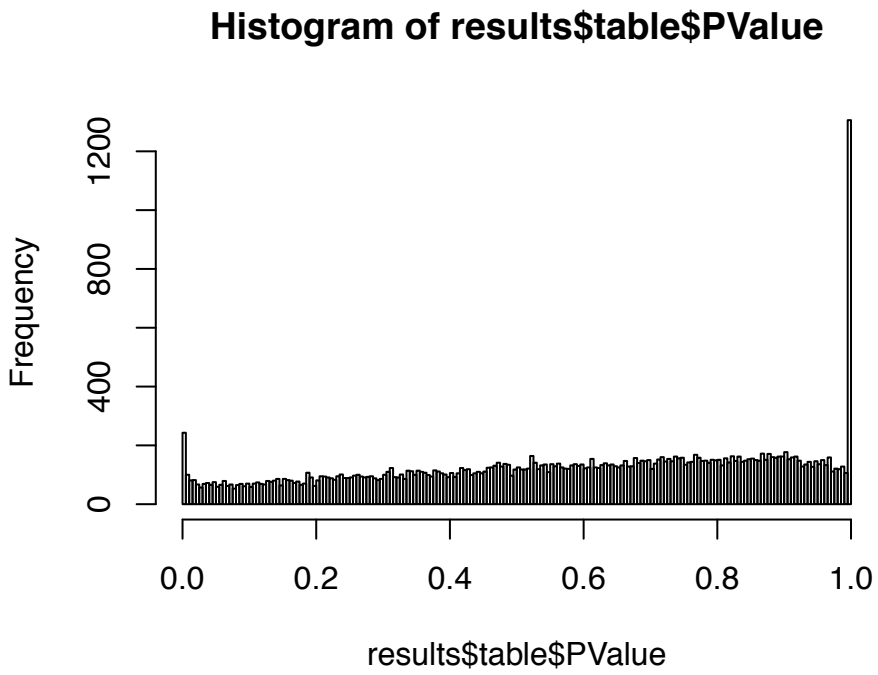

Figure 5: Histogram of Pvalues from the results table

Another useful plot is a smear plot that shows the relationship between concentraton 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")
```

Task 1 Numerical Questons

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?

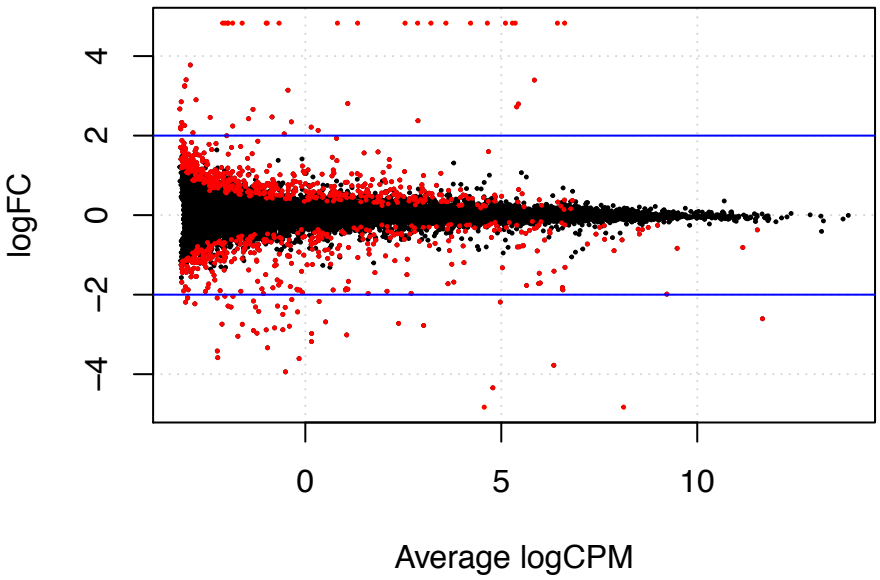

Figure 6: Smear plot showing the fold changes for genes.

```
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 analalysis on a subsample of the data set. The sub sample contained the same number of replicants and genes but had different number of counts. The same workflow was followed as task one.

The subsample 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.

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

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

```
#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 seperation between the male and female groups.

```
#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) ))
```

The common and tagwise dispersions are estimated for the data set and the are viewed using the `plotBCV()` functon. An exact test was used to find

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

To test for DGE in the sumsampled 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.common.disp.vars=FALSE, sh
```

Using the same methods as Task 1 we are able to find the to ten differential expressed genes as well as the bottom 10 genes. Using the `decideTestsDGE()` function again we can view the number of genes edgeR 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]
```

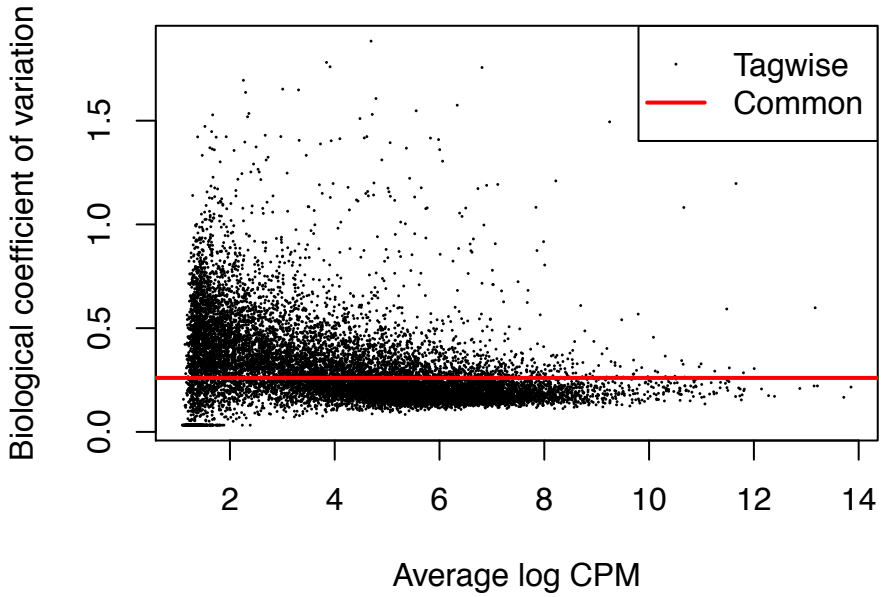

Figure 7: BCV plot for the sub samlped data.

```
##    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
merge(sub.geneInfo[rownames(sub.low),], tail(sub.low, n=10), by=0, sort=FALSE)
```

```
##          Row.names gene_name chromosome_name start_position   gene_biotype
## 1  ENSG00000124772     CPNE5               6       36708552 protein_coding
## 2  ENSG00000182541     LIMK2              22       31608225 protein_coding
## 3  ENSG00000105327      BBC3              19       47724081 protein_coding
## 4  ENSG00000033011      ALG1              16        5083703 protein_coding
## 5  ENSG00000187678     SPRY4               5      141689992 protein_coding
## 6  ENSG00000100473      COCH              14       31343720 protein_coding
## 7  ENSG00000198879    SFMBT2              10        7200586 protein_coding
## 8  ENSG00000156469   MTERFD1               8       97251626 protein_coding
## 9  ENSG00000111711    GOLT1B              12       21654715 protein_coding
## 10 ENSG00000175931     UBE2O              17       74385532 protein_coding
##          logFC logCPM PValue FDR
## 1  -3.566e-04  5.026      1   1
## 2   3.518e-04  5.594      1   1
## 3   3.085e-04  4.003      1   1
## 4  -3.084e-04  4.099      1   1
```

```
## 5  -2.478e-04  3.614      1   1
## 6   2.467e-04  3.516      1   1
## 7   2.005e-04  6.274      1   1
## 8  -1.165e-04  5.081      1   1
## 9  -4.951e-05  5.661      1   1
## 10  4.757e-05  6.239      1   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)
```

Histogram showing the distribution of the Pvalues for the subsampled dataset.

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

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

```
#smear plot - shows the relationship between concentraiton and the fold-change across the genes. Differ
#Blue line is at log-FC of 2, level of biological significance.
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")
```

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
```

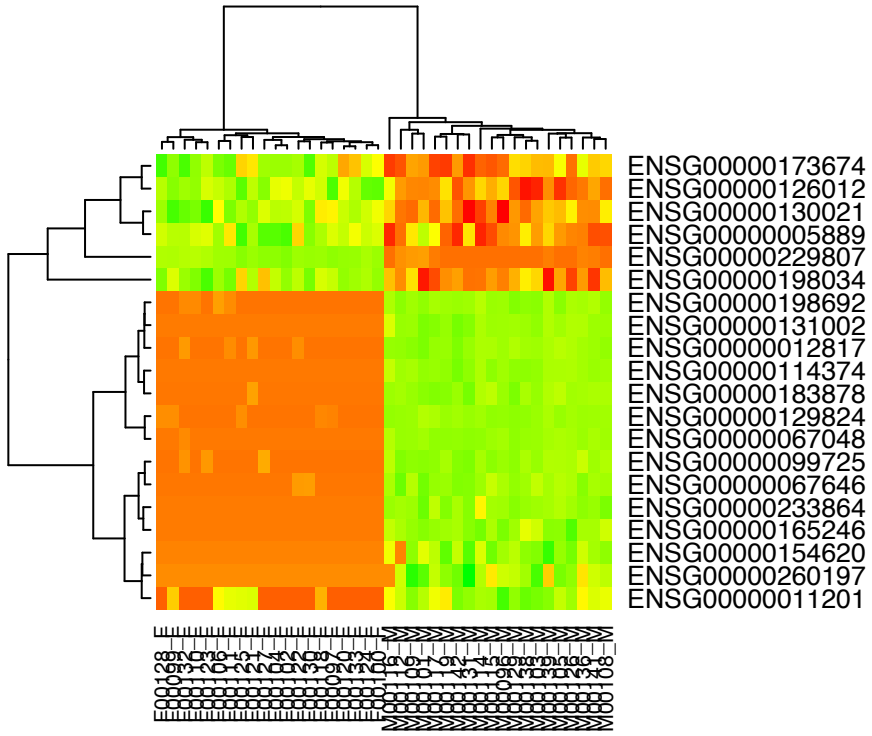

Figure 8: Heatmap for the subsampled data.

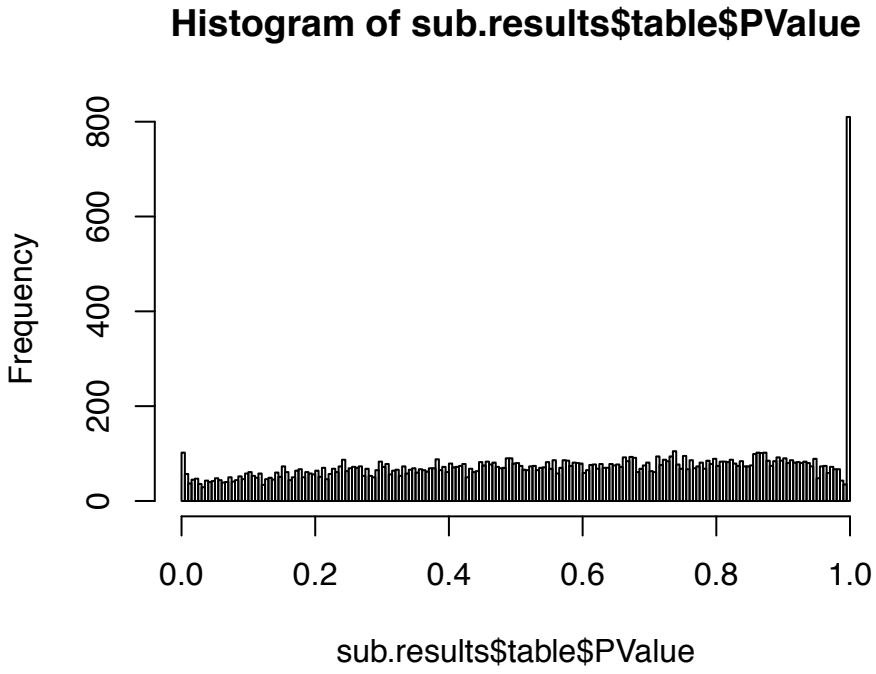

Figure 9: Histogram of Pvalues of sub sampled data set.

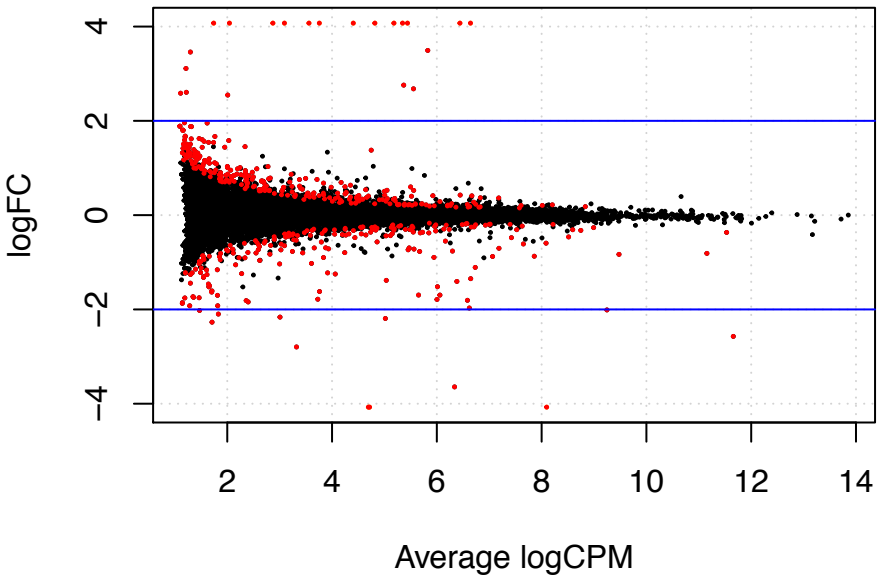

Figure 10: Sub sampled smear plot

```
(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.

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 we get the same randomisation each time the analysis is carried out.

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

```
randomgroup
```

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

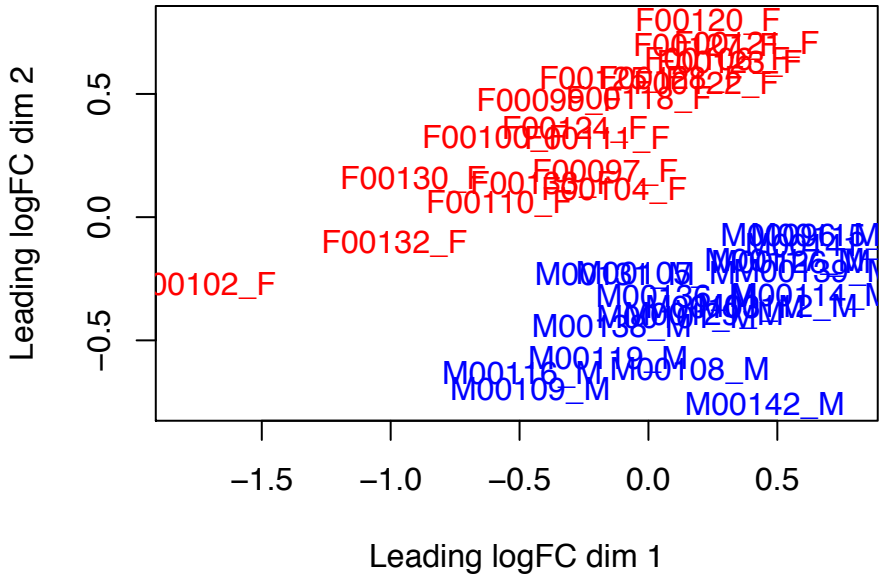

Figure 11: MDS plot for the randomly assigned group.

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

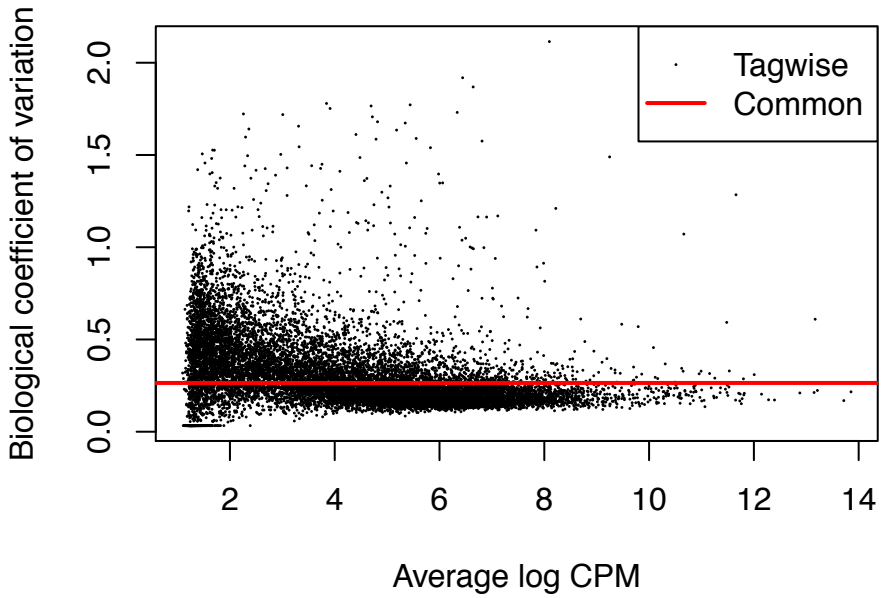

Figure 12: BCV plot for the randomly assigned groups.

```
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, sh
```

As the groups were assigned at random there are still some differentially expressed genes but not as many as found in Task 2.

```
#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  IGLV2-18             22    IG_V_gene 5.276  4.694 3.811e-09
```

```
#bottom 10 hits
rand.low = topTags(rand.results, n=nrow(rand.results$table))$table
merge(sub.geneInfo[rownames(rand.low),], tail(rand.low, n=10), by=0, sort=FALSE)
```

```
##          Row.names   gene_name chromosome_name start_position
## 1  ENSG00000156711      MAPK13               6       36095586
## 2  ENSG00000155111       CDK19               6      110931181
## 3  ENSG00000054392        HHAT               1      210501596
## 4  ENSG00000235162    C12orf75              12      105629068
```

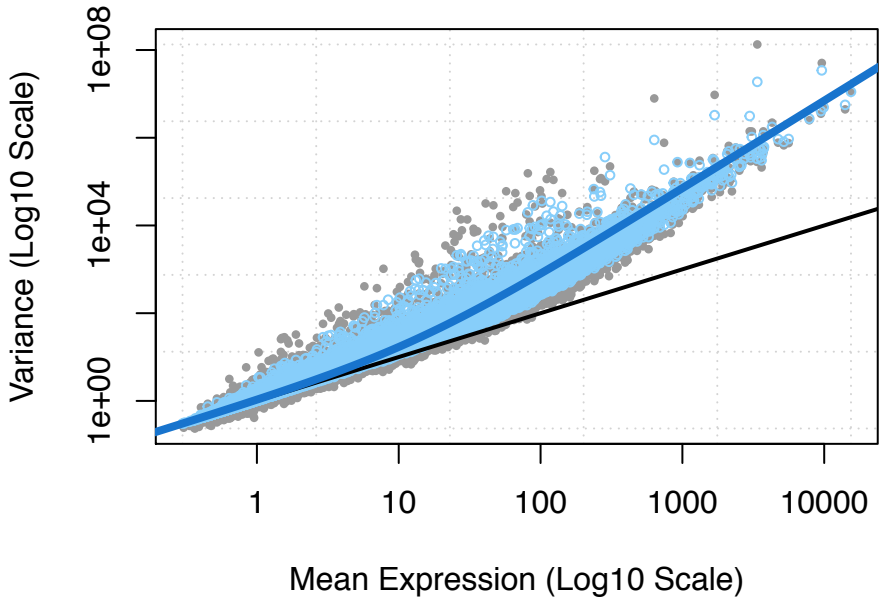

Figure 13: Mean variance plot for the randomly assigned groups.

```
## 5  ENSG00000262468 RP11-95P2.1              16        4295826
## 6  ENSG00000196562       SULF2              20       46285092
## 7  ENSG00000184792       OSBP2              22       31089769
## 8  ENSG00000187257      RSBN1L               7       77325760
## 9  ENSG00000152213       ARL11              13       50202435
## 10 ENSG00000105722         ERF              19       42751724
##      gene_biotype      logFC logCPM PValue FDR
## 1  protein_coding -4.641e-04  3.458      1   1
## 2  protein_coding  3.442e-04  5.767      1   1
## 3  protein_coding -3.137e-04  3.120      1   1
## 4  protein_coding  2.643e-04  7.100      1   1
## 5         lincRNA  2.237e-04  2.132      1   1
## 6  protein_coding  2.141e-04  3.275      1   1
## 7  protein_coding -1.605e-04  2.177      1   1
## 8  protein_coding  8.564e-05  5.063      1   1
## 9  protein_coding -5.253e-05  3.698      1   1
## 10 protein_coding -2.140e-06  4.956      1   1
```

```
#bottom 10 differentially expressed hits.
rand.low.de = rand.low[rand.low$PValue < 0.05,]
merge(sub.geneInfo[rownames(rand.low),], tail(rand.low.de, n=10), by=0, sort=FALSE)
```

```
##          Row.names    gene_name chromosome_name start_position
## 1  ENSG00000181798    LINC00471               2      232373137
## 2  ENSG00000139351        SYCP3              12      102122426
## 3  ENSG00000261542 RP11-16E18.3               8       64128539
## 4  ENSG00000081386       ZNF510               9       99518147
## 5  ENSG00000004961         HCCS               X       11129421
## 6  ENSG00000231125   AF129075.5              21       30430394
## 7  ENSG00000120833        SOCS2              12       93963590
## 8  ENSG00000222037        IGLC6              22       23261707
```

```
## 9  ENSG00000167861         HID1              17       72946838
## 10 ENSG00000137547       MRPL15               8       55047770
##            gene_biotype   logFC logCPM  PValue FDR
## 1  processed_transcript -0.6262  1.827 0.04956   1
## 2        protein_coding  1.1539  1.260 0.04964   1
## 3               lincRNA  0.8722  1.331 0.04965   1
## 4        protein_coding  0.2693  3.930 0.04967   1
## 5        protein_coding -0.2275  5.110 0.04968   1
## 6        sense_intronic  1.1769  1.228 0.04972   1
## 7        protein_coding  0.2911  4.577 0.04975   1
## 8       IG_C_pseudogene  1.3337  2.674 0.04981   1
## 9        protein_coding  0.6256  2.312 0.04982   1
## 10       protein_coding -0.2533  5.952 0.04986   1
```

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

```
##    [,1]
## -1    0
## 0 14601
## 1     2
```

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

Visualisation

As expected there is no real clustering between the differentially 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)
```

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

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

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

```
#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")
```

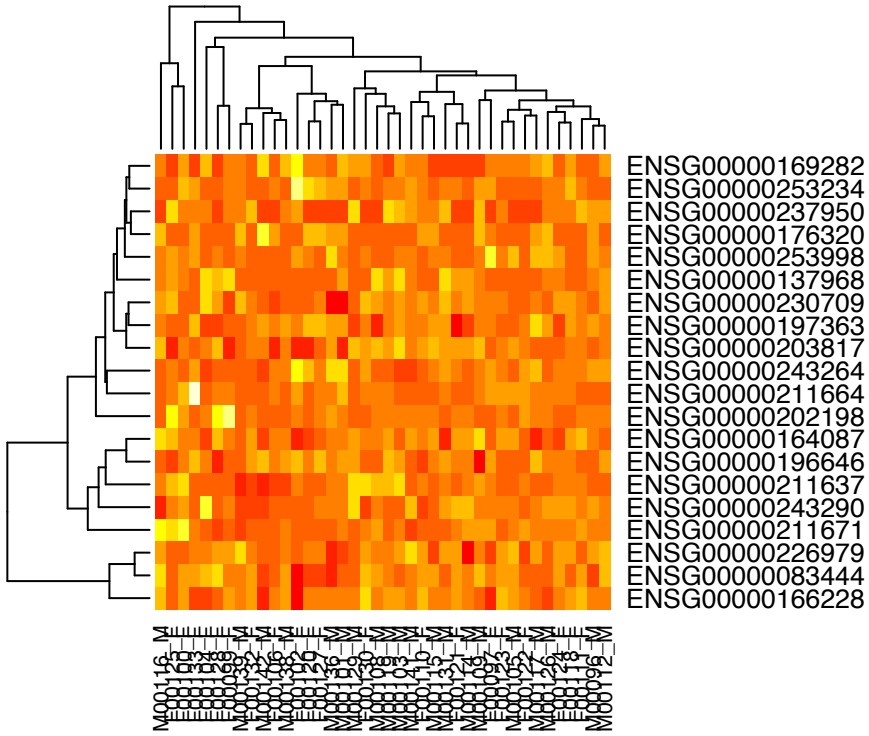

Figure 14: Heat map from the randomly assigned gropus

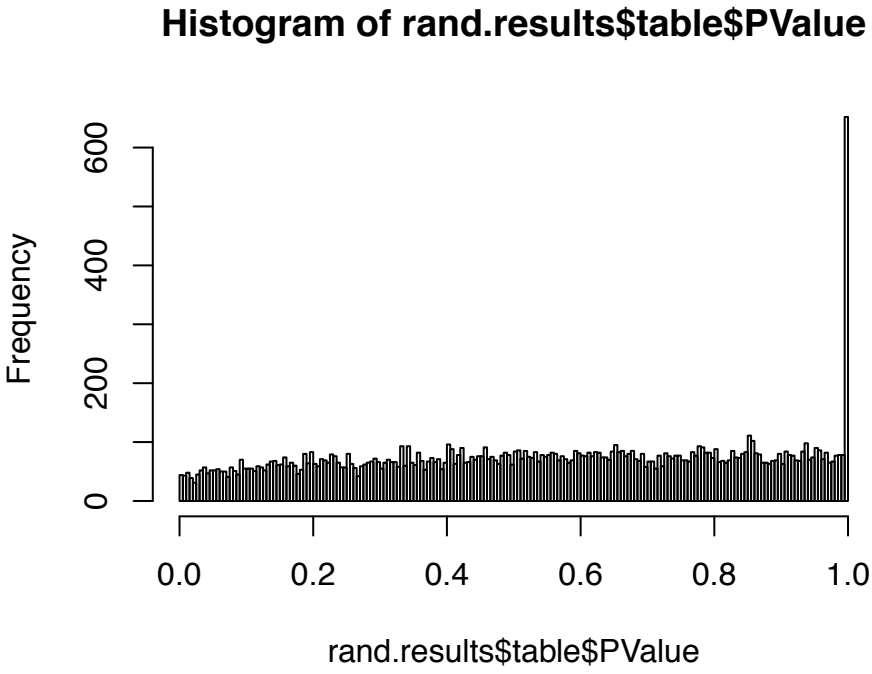

Figure 15: Histogram for the randomly assigned groups.

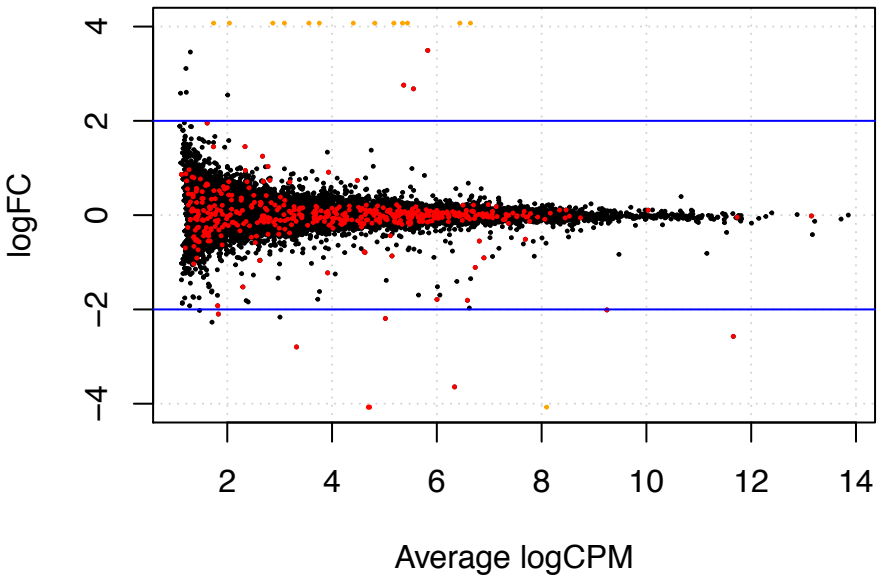

Figure 16: Smear plot for randomly assigned groups.

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] 458
```

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

```
## [1] 0.03136
```

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] 0
```

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$table)==rownames(xist), ]
rand.xist.Result$logFC
```

```
## [1] 0.6323
```

Task 4: Discussion

Comment on:

The differences between the analysis of the higher-depth RNA-seq data and the subsampled RNA-seq data. That is, compare your results from Task 1 to your results from Task 2.

The differences between the analysis of male vs female samples and the randomised analysis. That is, compare your results from Task 2 to your results from Task 3.

You should discuss any interesting differences you see in the results (which may include plots, tables and the answers to the numerical questions) and most importantly, the reasons for these differences. This discussion should be no more than 800 words. This word limit applies only to Task 4 and not to any other part of the assignment.

```
#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
```