RNA-Seq Analysis

William Walters

RNA-Seq Analysis

Task 1: DGE Analysis on Original Dataset

The following R code performs Differential Gene Expression (DGE) analysis on the full dataset.

```
devtools::install_github('NMF', 'renozao', 'devel')
```

```
# Task 1:

suppressMessages(library(edgeR))

## Warning:  package ’limma’ was built under R version 3.4.2

suppressMessages(library(xtable))
suppressMessages(library(NMF))

# Load the count data and check size
fullCountData = read.table("fullcounts.tsv", header=TRUE, sep="\t")
dim(fullCountData)

## [1] 44061    34

# remove genes with very low count profiles
lowCounts = apply(fullCountData[,1:15], 1, median)==0 &
  apply(fullCountData[,16:30], 1 ,median)==0
countsFiltered = fullCountData[!lowCounts,]

# sanity check the filtering worked
dim(countsFiltered)[1] < dim(fullCountData)[1]

## [1] TRUE

dim(countsFiltered)

## [1] 24329    34

# separate count data from extra data
expressionData = countsFiltered[,1:30]
geneInfo = countsFiltered[,31:34]

# Tell edgeR about Male and Female groups
isMale = c( rep(0,15), rep(1,15) )
# prepare data for edgeR to use, comparison between males and females
```

```
dge = DGEList(expressionData, group=isMale)
# perform TMM (trimmed mean) normalisation
# assumption that most genes are not differentially expressed
dge = calcNormFactors(dge)

cols = c(rep("red",15), rep("blue",15))
plotMDS(dge, col=cols, main = "MDS plot of full lymphoblastoid cell data")
```

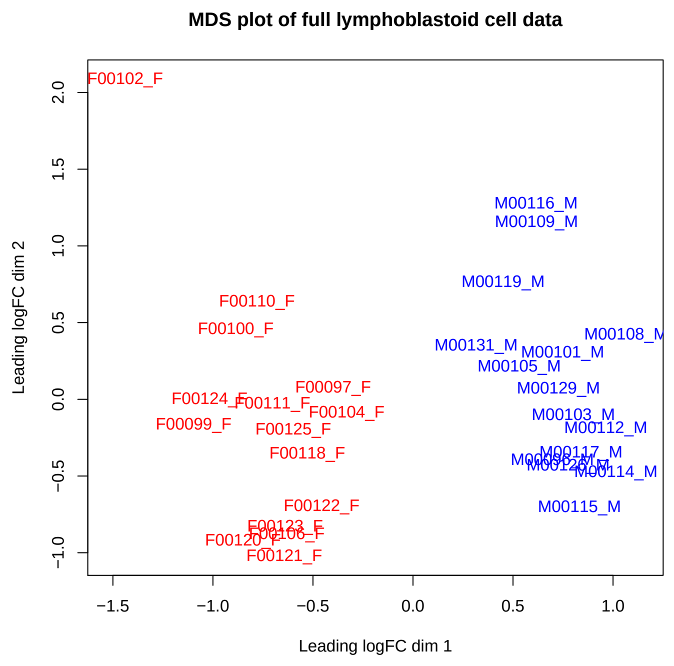

Figure 1 MDS Plot on the full lymphoblastoid cell dataset.

```
# model randomness in RNA-seq counts.
dge = estimateCommonDisp(dge)
# per-gene estimates of the variance
dge = estimateTagwiseDisp(dge)
# examine differential expression
results = exactTest(dge)

# give the top genes sorted by p-value
topGenes = topTags(results, n=20)
chromosome = geneInfo[rownames(topTags(results, n=20)), 2]
name = geneInfo[rownames(topTags(results, n=20)), 1]
expressionTable1 = data.frame(topGenes, chromosome, name)
```

```
print(xtable(expressionTable1[1:10, c(1,2,3,5,6)],
             caption = "Expression table for the top 10 differentially expressed genes
             on the full lymphoblastoid cell dataset.",
             display = c("s","f","f","e","s","s"), digits = 4))
```

	logFC	logCPM	PValue	chromosome	name
ENSG00000131002	10.2518	5.3389	0.0000e+00	Y	TXLNG2P
ENSG00000129824	10.2460	6.6312	0.0000e+00	Y	RPS4Y1
ENSG00000012817	10.0724	5.0934	0.0000e+00	Y	KDM5D
ENSG00000067048	9.9740	6.4486	0.0000e+00	Y	DDX3Y
ENSG00000229807	-9.7704	8.1540	0.0000e+00	X	XIST
ENSG00000183878	9.6835	4.2552	0.0000e+00	Y	UTY
ENSG00000114374	9.5365	4.6219	0.0000e+00	Y	USP9Y
ENSG00000198692	9.0576	5.2932	0.0000e+00	Y	EIF1AY
ENSG00000067646	8.9534	3.1856	6.1029e-276	Y	ZFY
ENSG00000233864	9.0712	2.8371	2.3682e-258	Y	TTTY15

Table 1: Expression table for the top 10 differentially expressed genes on the full lymphoblastoid cell dataset.

```r
# Find top expressed genes to plot in a heatmap
topGeneNames = rownames(topTags(results, n=15))
logDGE = cpm(dge, prior.count=2, log=TRUE)
topGenesLogExpression = logDGE[topGeneNames,]
geneNames = expressionTable1[rownames(topGenesLogExpression),6]
row.names(topGenesLogExpression) = geneNames

# Removing heteroscedascity from count data
groupMatrix <- model.matrix(~isMale)
v = voom(dge, groupMatrix, plot=FALSE)
```

```r
# Construct a Heatmap of most differentially expressed genes
ann_col = list(Groups = c("Female","Female","Female","Female","Female","Female",
                          "Female","Female","Female","Female","Female","Female",
                          "Female","Female","Female","Male","Male","Male","Male",
                          "Male","Male","Male","Male","Male","Male","Male", "Male",
                          "Male","Male","Male"))
aheatmap(v$E[topGeneNames,], annCol = ann_col, width = 16,
         labRow = geneNames)
```

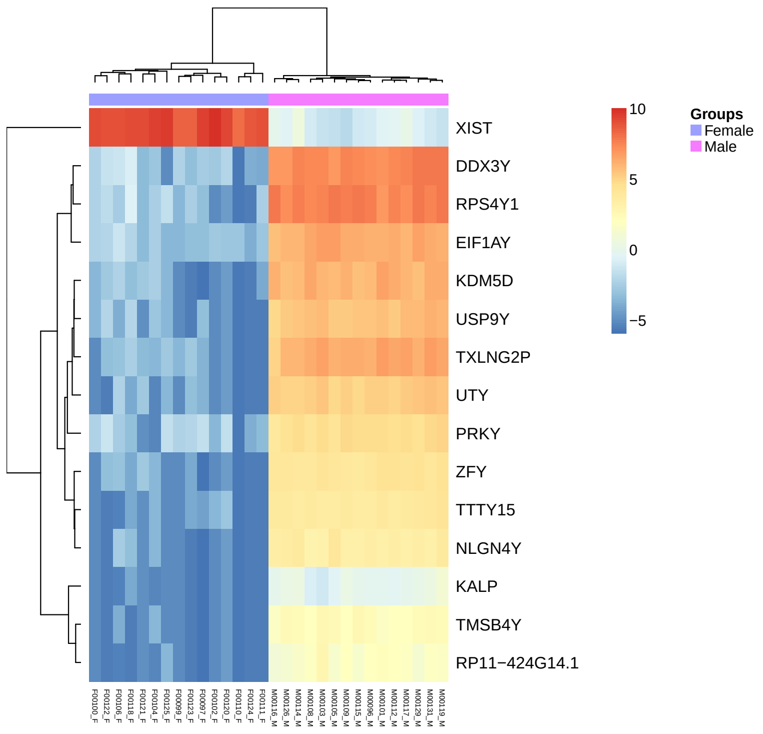

Figure 2 Heatmap of gene expression for the top 15 differentially expressed genes (determined by P-value) on the full lymphoblastoid cell dataset.

```
# plot a histogram of uncorrected p-values
hist(results$table$PValue, breaks=150, xlab = "P-values",
     main = "Gene Frequency vs. Uncorrected p-values")
```

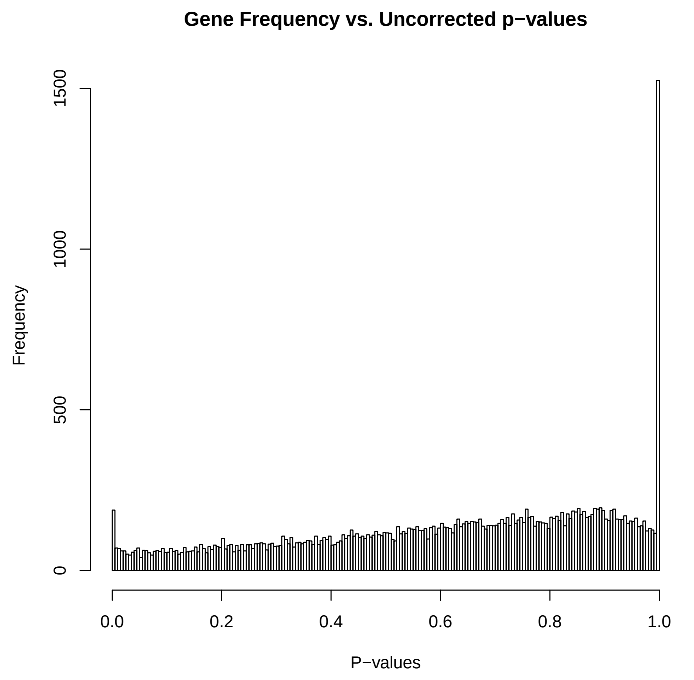

Figure 3 Histogram of uncorrected P-values on the full lymphoblastoid cell dataset.

a)

```
# Task 1: a)
# count number of uncorrected significant genes
numSigGenes = sum(results$table$PValue < 0.05)
numSigGenes

## [1] 737

propSigGenes = numSigGenes / dim(countsFiltered)[1]
propSigGenes = round(propSigGenes, digits = 4)
propSigGenes

## [1] 0.0303
```

Using the full dataset, 737 genes were found to have uncorrected P-values below 0.05. Of all the genes tested, this proportion was 0.0303.

b)

```
# Task 1: b)
# create a histogram on B-H adjusted p-values
qValues = p.adjust(results$table$PValue, method="BH")
hist(qValues, breaks=100, ylim=c(0,500), xlab = "B-H Corrected P-values",
     main = "Gene Frequency vs. B-H Corrected p-values")
```

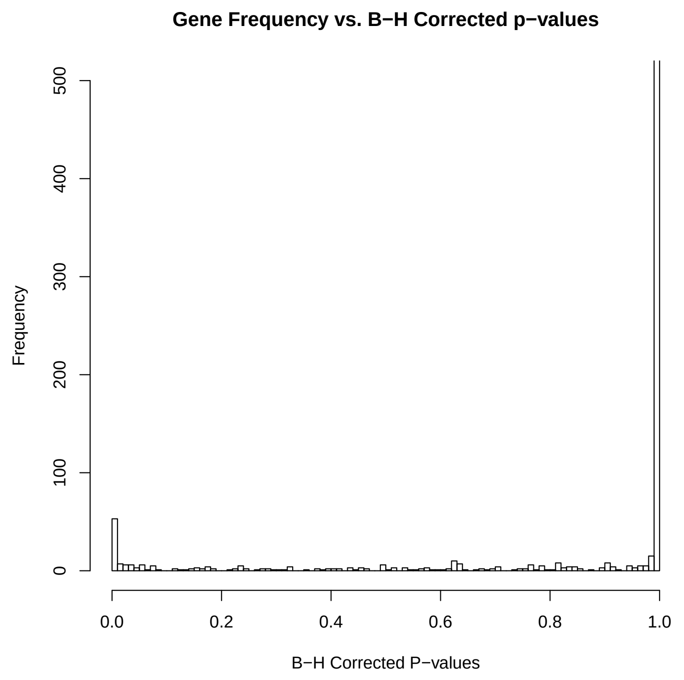

Figure 4 Histogram of Benjamini-Hochberg corrected P-values on the full lymphoblastoid cell dataset.

```
# count number of B-H corrected significant genes
numBHSigGenes = sum(qValues < 0.05)
numBHSigGenes

## [1] 75

propBHSigGenes = numBHSigGenes / dim(countsFiltered)[1]
propBHSigGenes = round(propBHSigGenes, digits = 4)
propBHSigGenes

## [1] 0.0031
```

Using the full dataset, 75 genes were found to have Benjamini-Hochberg corrected P-values below 0.05. Of all the genes tested, this proportion was 0.0031.

c)

```
# Task 1: c)
# find significant genes on the Y chromsome
numYGenes = sum(geneInfo[, 2] == "Y")
numYGenes

## [1] 34
```

```r
Y = geneInfo$chromosome_name=="Y"
YGeneNames = rownames(geneInfo)[Y]

qValues = p.adjust(results$table$PValue, method="BH")
results$table = cbind(results$table, qValues)
colnames(results$table) = c("logFC", "logCPM", "PValue", "QValue")
YGeneResults = results$table[YGeneNames, ]
YSigGeneResults = YGeneResults[which(YGeneResults$QValue < 0.05), ]

numYBHSigGenes = nrow(YSigGeneResults)
numYBHSigGenes

## [1] 24

propYBHSigGenes = numYBHSigGenes / numYGenes
propYBHSigGenes = round(propYBHSigGenes, digits = 4)
propYBHSigGenes

## [1] 0.7059
```

Using the full dataset, 24 Y chromosome genes were found to have Benjamini-Hochberg corrected P-values below 0.05. Of all the Y chromosome genes tested, this proportion was 0.7059.

d)

```r
# Task 1: d)
# find the logFC for the XIST gene
XIST = geneInfo$gene_name=="XIST"
rownameXIST = rownames(geneInfo)[XIST]
logFC_XIST = round(results[rownameXIST,]$table$logFC, digits = 4)
logFC_XIST

## [1] -9.7704
```

Using the full dataset, the log fold-change for the gene XIST is: -9.7704

Task 2: DGE Analysis on Subsampled Dataset

The following R code performs Differential Gene Expression (DGE) analysis on the subsampled (low coverage) dataset.

```r
# Task 2:

# Load the count data and check size
subCountData = read.table("subsample.tsv", header=TRUE, sep="\t")
dim(subCountData)

## [1] 44061    34

# remove genes with very low count profiles
```

```
lowCounts = apply(subCountData[,1:15], 1, median)==0 &
  apply(subCountData[,16:30], 1 ,median)==0
countsFiltered = subCountData[!lowCounts,]

# sanity check the filtering worked
dim(countsFiltered)[1] < dim(fullCountData)[1]

## [1] TRUE

dim(countsFiltered)

## [1] 14449    34

# separate count data from extra data
expressionData = countsFiltered[,1:30]
geneInfo = countsFiltered[,31:34]

# Tell edgeR about Male and Female groups
isMale = c( rep(0,15), rep(1,15) )
# prepare data for edgeR to use, comparison between males and females
dge = DGEList(expressionData, group=isMale)
# perform TMM (trimmed mean) normalisation
# assumption that most genes are not differentially expressed
dge = calcNormFactors(dge)

cols = c(rep("red",15), rep("blue",15))
plotMDS(dge, col=cols, main = "MDS plot of subsampled lymphoblastoid cell data")
```

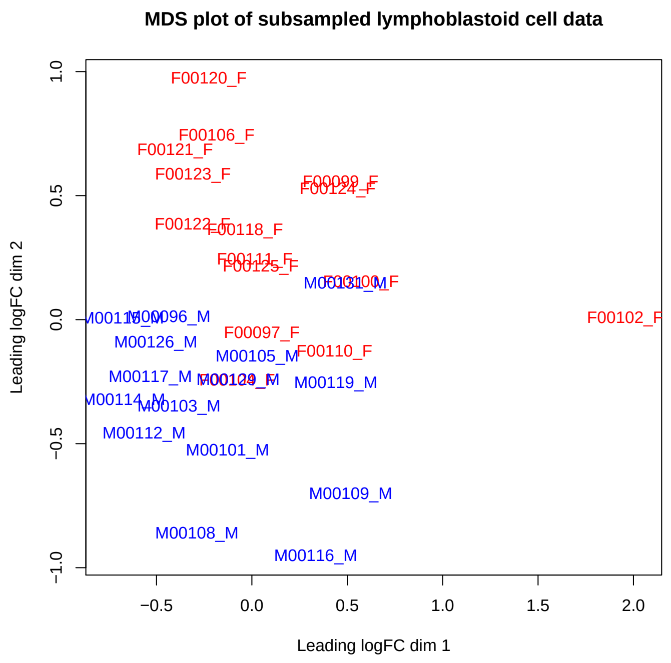

Figure 5 MDS Plot on the subsampled lymphoblastoid cell dataset.

```r
# model randomness in RNA-seq counts.
dge = estimateCommonDisp(dge)
# per-gene estimates of the variance
dge = estimateTagwiseDisp(dge)
# examine differential expression
results = exactTest(dge)

# give the top genes sorted by p-value
topGenes = topTags(results, n=20)
chromosome = geneInfo[rownames(topTags(results, n=20)), 2]
name = geneInfo[rownames(topTags(results, n=20)), 1]
expressionTable2 = data.frame(topGenes, chromosome, name)
```

```r
print(xtable(expressionTable2[1:10, c(1,2,3,5,6)],
             caption = "Expression table for the top 10 differentially expressed genes
             on the subsampled lymphoblastoid cell dataset.",
             display = c("s","f","f","e","s","s"), digits = 4))
```

	logFC	logCPM	PValue	chromosome	name
ENSG00000129824	9.5282	6.7022	1.7779e-274	Y	RPS4Y1
ENSG00000229807	-9.7300	8.1441	1.7609e-240	X	XIST
ENSG00000067048	9.3342	6.5116	1.8322e-226	Y	DDX3Y
ENSG00000012817	9.0003	5.1600	1.9622e-168	Y	KDM5D
ENSG00000198692	8.1057	5.3362	1.2134e-155	Y	EIF1AY
ENSG00000183878	7.5430	4.4107	2.7976e-123	Y	UTY
ENSG00000131002	8.1391	5.3695	4.2956e-122	Y	TXLNG2P
ENSG00000114374	7.9061	4.7397	4.3812e-97	Y	USP9Y
ENSG00000099725	6.1670	3.8906	5.5034e-69	Y	PRKY
ENSG00000067646	7.2735	3.6256	4.1768e-59	Y	ZFY

Table 2: Expression table for the top 10 differentially expressed genes on the subsampled lymphoblastoid cell dataset.

```r
# Find top expressed genes to plot in a heatmap
topGeneNames = rownames(topTags(results, n=15))
logDGE = cpm(dge, prior.count=2, log=TRUE)
topGenesLogExpression = logDGE[topGeneNames,]
geneNames = expressionTable2[rownames(topGenesLogExpression),6]
row.names(topGenesLogExpression) = geneNames

# Removing heteroscedascity from count data
groupMatrix <- model.matrix(~isMale)
v = voom(dge, groupMatrix, plot=FALSE)
```

```r
# Construct a Heatmap of most differentially expressed genes
aheatmap(v$E[topGeneNames,], annCol = ann_col, width = 16,
         labRow = geneNames)
```

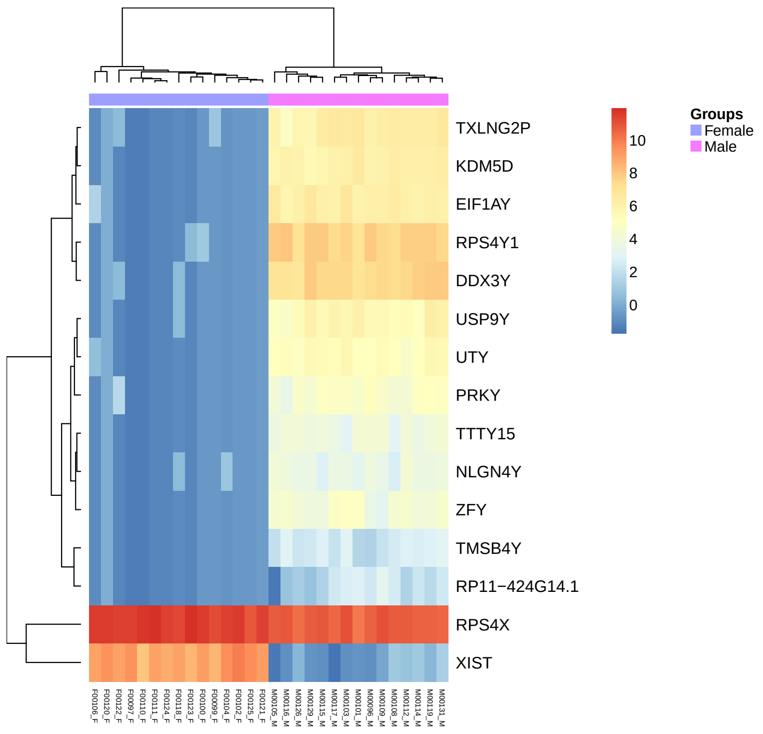

Figure 6 Heatmap of gene expression for the top 15 differentially expressed genes (determined by P-value) on the subsampled lymphoblastoid cell dataset

```r
# plot a histogram of uncorrected p-values
hist(results$table$PValue, breaks=150, xlab = "P-values",
     main = "Gene Frequency vs. Uncorrected p-values")
```

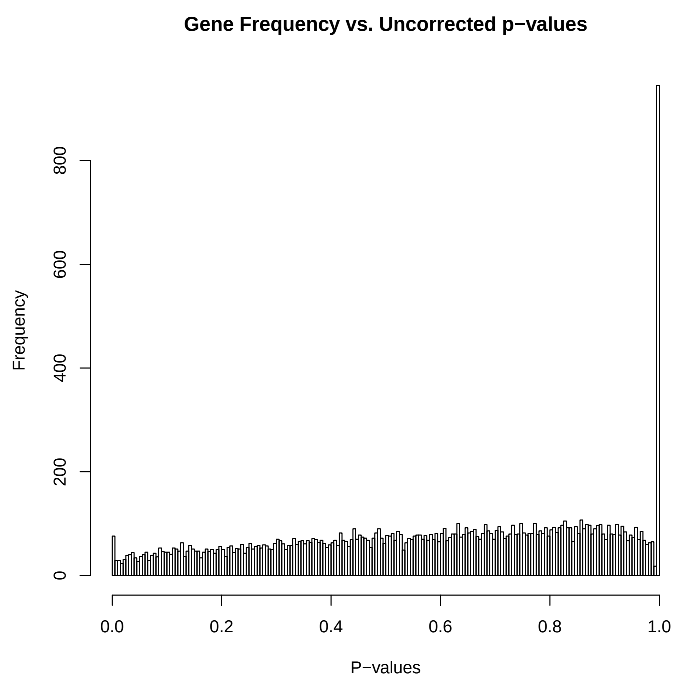

Figure 7 Histogram of uncorrected P-values on the subsampled lymphoblastoid cell dataset.

a)

```
# Task 2: a)
# count number of uncorrected significant genes
numSigGenes = sum(results$table$PValue < 0.05)
numSigGenes

## [1] 372

propSigGenes = numSigGenes / dim(countsFiltered)[1]
propSigGenes = round(propSigGenes, digits = 4)
propSigGenes

## [1] 0.0257
```

Using the subsampled (low coverage) dataset, 372 genes were found to have uncorrected P-values below 0.05. Of all the genes tested, this proportion was 0.0257.

b)

```
# Task 2: b)
# create a histogram on B-H adjusted p-values
qValues = p.adjust(results$table$PValue, method="BH")
hist(qValues, breaks=100, ylim=c(0,500), xlab = "B-H Corrected P-values",
     main = "Gene Frequency vs. B-H Corrected p-values")
```

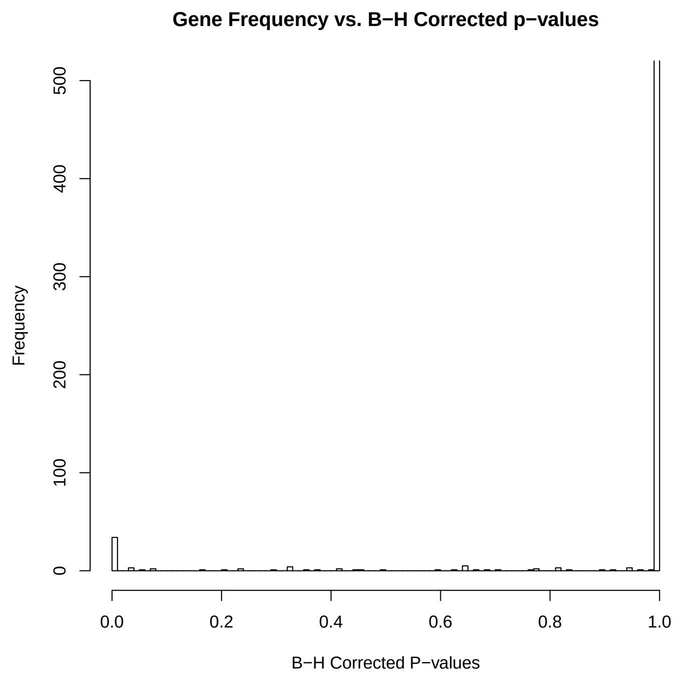

Figure 8 Histogram of Benjamini-Hochberg corrected P-values on the subsampled lymphoblastoid cell dataset.

```
# count number of B-H corrected significant genes
numBHSigGenes = sum(qValues < 0.05)
numBHSigGenes

## [1] 37

propBHSigGenes = numBHSigGenes / dim(countsFiltered)[1]
propBHSigGenes = round(propBHSigGenes, digits = 4)
propBHSigGenes

## [1] 0.0026
```

Using the subsampled (low coverage) dataset, 37 genes were found to have Benjamini-Hochberg corrected P-values below 0.05. Of all the genes tested, this proportion was 0.0026.

c)

```
# Task 2: c)
# find significant genes on the Y chromsome
numYGenes = sum(geneInfo[, 2] == "Y")
numYGenes

## [1] 20
```

```
Y = geneInfo$chromosome_name=="Y"
YGeneNames = rownames(geneInfo)[Y]

qValues = p.adjust(results$table$PValue, method="BH")
results$table = cbind(results$table, qValues)
colnames(results$table) = c("logFC", "logCPM", "PValue", "QValue")
YGeneResults = results$table[YGeneNames, ]
YSigGeneResults = YGeneResults[which(YGeneResults$QValue < 0.05), ]

numYBHSigGenes = nrow(YSigGeneResults)
numYBHSigGenes

## [1] 15

propYBHSigGenes = numYBHSigGenes / numYGenes
propYBHSigGenes = round(propYBHSigGenes, digits = 4)
propYBHSigGenes

## [1] 0.75
```

Using the subsampled (low coverage) dataset, 15 Y chromosome genes were found to have Benjamini-Hochberg corrected P-values below 0.05. Of all the Y chromosome genes tested, this proportion was 0.75.

d)

```
# Task 2: d)
# find the logFC for the XIST gene
XIST = geneInfo$gene_name=="XIST"
rownameXIST = rownames(geneInfo)[XIST]
logFC_XIST = round(results[rownameXIST,]$table$logFC, digits = 4)
logFC_XIST

## [1] -9.73
```

Using the subsampled dataset (lower coverage), the log fold-change for the gene XIST is: -9.73

Task 3: DGE Analysis on Subsampled, Randomised Dataset

The following R code performs Differential Gene Expression (DGE) analysis on the subsampled (low coverage) dataset with randomised sample groups. Note that since the sample groups were randomised, the reported gene numbers, P-values and log Fold-Change from the subsections can change between sampling runs.

```
# Task 3:

# group split for the randomised DGE analysis
# randomgroup is the vector used to assign samples to groups
randomgroup = sample(c(0,1), 30, replace=TRUE)
randomgroup

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

```
# Load the count data and check size
subCountData = read.table("subsample.tsv", header=TRUE, sep="\t")
dim(subCountData)

## [1] 44061    34

# remove genes with very low count profiles
lowCounts = apply(subCountData[,1:15], 1, median)==0 &
  apply(subCountData[,16:30], 1 ,median)==0
countsFiltered = subCountData[!lowCounts,]

# sanity check the filtering worked
dim(countsFiltered)[1] < dim(fullCountData)[1]

## [1] TRUE

dim(countsFiltered)

## [1] 14449    34

# separate count data from extra data
expressionData = countsFiltered[,1:30]
geneInfo = countsFiltered[,31:34]

# prepare data for edgeR to use, comparison between males and females
dge = DGEList(expressionData, group=randomgroup)
# perform TMM (trimmed mean) normalisation
# assumption that most genes are not differentially expressed
dge = calcNormFactors(dge)

cols = rep("", 30)
for (i in seq(1,30,1)){
  if (randomgroup[i] == 0)
    cols[i] = "red"
  else
    cols[i] = "blue"
}
plotMDS(dge, col=cols, main = "MDS plot of randomised, subsampled lymphoblastoid cell data")
```

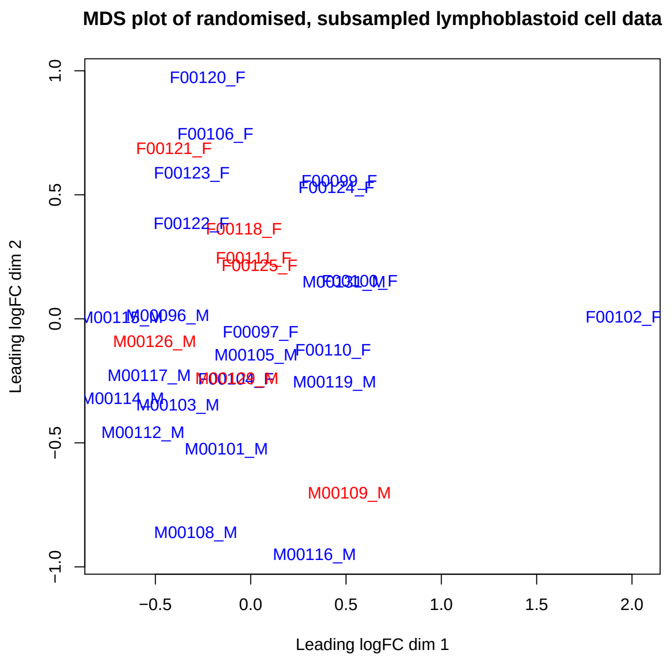

Figure 9 MDS Plot on the subsampled, randomised lymphoblastoid cell dataset.

```
# model randomness in RNA-seq counts.
dge = estimateCommonDisp(dge)
# per-gene estimates of the variance
dge = estimateTagwiseDisp(dge)
# examine differential expression
results = exactTest(dge)

# give the top genes sorted by p-value
topGenes = topTags(results, n=20)
chromosome = geneInfo[rownames(topTags(results, n=20)), 2]
name = geneInfo[rownames(topTags(results, n=20)), 1]
expressionTable3 = data.frame(topGenes, chromosome, name)
```

```
print(xtable(expressionTable3[1:10, c(1,2,3,5,6)],
             caption = "Expression table for the top 10 differentially expressed genes
             on the subsampled, randomised lymphoblastoid cell dataset.",
             display = c("s","f","f","e","s","s"), digits = 4))
```

```
# Find top expressed genes to plot in a heatmap
topGeneNames = rownames(topTags(results, n=15))
logDGE = cpm(dge, prior.count=2, log=TRUE)
topGenesLogExpression = logDGE[topGeneNames,]
geneNames = expressionTable3[rownames(topGenesLogExpression),6]
```

	logFC	logCPM	PValue	chromosome	name
ENSG00000162817	-2.1846	2.2345	2.2876e-07	1	C1orf115
ENSG00000211942	-2.8220	4.1749	6.0291e-05	14	IGHV3-13
ENSG00000075340	-1.5220	2.3490	7.9807e-05	2	ADD2
ENSG00000073861	-0.9826	4.5824	3.4962e-04	17	TBX21
ENSG00000117791	-1.4670	2.3939	5.2334e-04	1	MARC2
ENSG00000163823	-0.7911	4.6271	5.5274e-04	3	CCR1
ENSG00000242779	-1.8937	1.8201	6.6272e-04	19	ZNF702P
ENSG00000198794	-1.3036	3.2196	1.3520e-03	15	SCAMP5
ENSG00000232527	3.3435	1.5977	1.5379e-03	1	RP11-14N7.2
ENSG00000169398	-0.7262	4.0368	1.6130e-03	8	PTK2

Table 3: Expression table for the top 10 differentially expressed genes on the subsampled, randomised lymphoblastoid cell dataset.

```r
row.names(topGenesLogExpression) = geneNames

# Remove heteroscedascity from count data
groupMatrix <- model.matrix(~randomgroup)
v = voom(dge, groupMatrix, plot=FALSE)
```

```r
# Construct a Heatmap of most differentially expressed genes
aheatmap(v$E[topGeneNames,], annCol = ann_col, width = 16,
         labRow = geneNames)
```

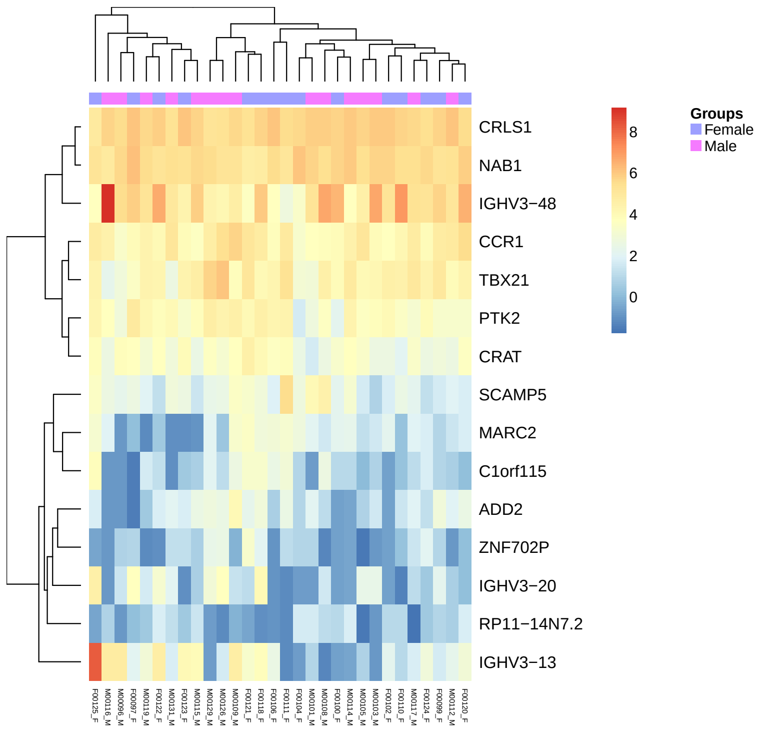

Figure 10 Heatmap of gene expression for the top 15 differentially expressed genes (determined by P-value) on the subsampled, randomised lymphoblastoid cell dataset

```r
# plot a histogram of uncorrected p-values
hist(results$table$PValue, breaks=150, xlab = "P-values",
     main = "Gene Frequency vs. Uncorrected p-values")
```

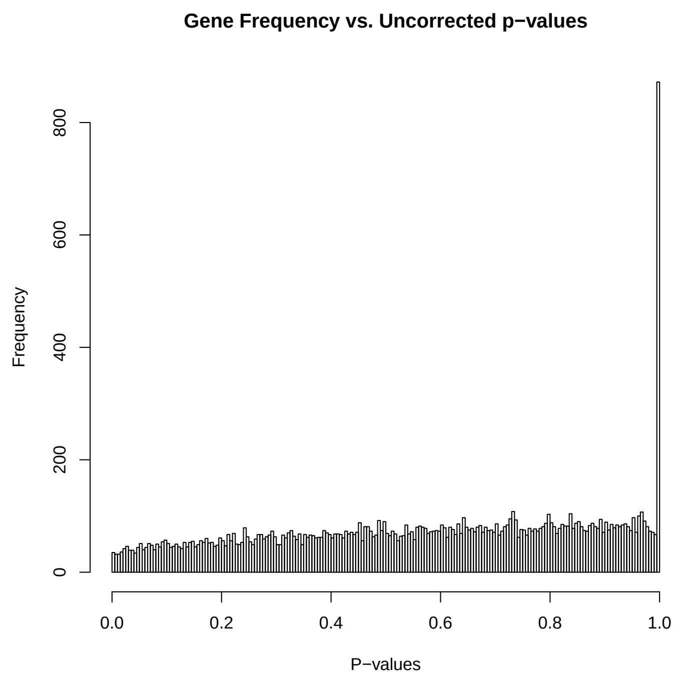

Figure 11 Histogram of uncorrected P-values on the subsampled, randomised lymphoblastoid cell dataset.

a)

```r
# Task 3: a)
# count number of uncorrected significant genes
numSigGenes = sum(results$table$PValue < 0.05)
numSigGenes

## [1] 379

propSigGenes = numSigGenes / dim(countsFiltered)[1]
propSigGenes = round(propSigGenes, digits = 4)
propSigGenes

## [1] 0.0262
```

Using the subsampled (low coverage) dataset with randomised sample groups, 379 genes were found to have uncorrected P-values below 0.05. Of all the genes tested, this proportion was 0.0262.

b)

```
# Task 3: b)
# create a histogram on B-H adjusted p-values
qValues = p.adjust(results$table$PValue, method="BH")
hist(qValues, breaks=100, ylim=c(0,100), xlab = "B-H Corrected P-values",
     main = "Gene Frequency vs. B-H Corrected p-values")
```

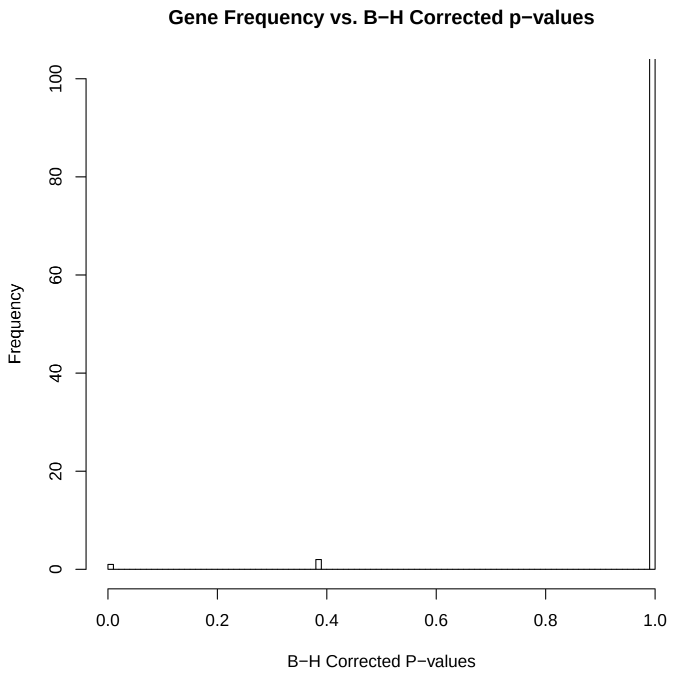

Figure 12 Histogram of Benjamini-Hochberg corrected P-values on the subsampled, randomised lymphoblastoid cell dataset.

```
# count number of B-H corrected significant genes
numBHSigGenes = sum(qValues < 0.05)
numBHSigGenes

## [1] 1

propBHSigGenes = numBHSigGenes / dim(countsFiltered)[1]
propBHSigGenes = round(propBHSigGenes, digits = 4)
propBHSigGenes

## [1] 1e-04
```

Using the subsampled (low coverage) dataset with randomised sample groups, 1 genes were found to have Benjamini-Hochberg corrected P-values below 0.05. Of all the genes tested, this proportion was 10^{-4}.

c)

```r
# Task 3: c)
# find significant genes on the Y chromsome
numYGenes = sum(geneInfo[, 2] == "Y")
numYGenes

## [1] 20

Y = geneInfo$chromosome_name=="Y"
YGeneNames = rownames(geneInfo)[Y]
YGeneResults = results[YGeneNames, ]

YQValues = p.adjust(YGeneResults$table$PValue, method="BH")
numYBHSigGenes = sum(YQValues < 0.05)
numYBHSigGenes

## [1] 0

propYBHSigGenes = numYBHSigGenes / numYGenes
propYBHSigGenes = round(propYBHSigGenes, digits = 4)
propYBHSigGenes

## [1] 0
```

Using the subsampled (low coverage) dataset with randomised sample groups, 0 Y chromosome genes were found to have Benjamini-Hochberg corrected P-values below 0.05. Of all the Y chromosome genes tested, this proportion was 0.

d)

```r
# Task 3: d)
# find the logFC for the XIST gene
XIST = geneInfo$gene_name=="XIST"
rownameXIST = rownames(geneInfo)[XIST]
logFC_XIST = round(results[rownameXIST,]$table$logFC, digits = 4)
logFC_XIST

## [1] -0.3382
```

Using the subsampled dataset (lower coverage) with randomised sample groups, the log fold-change for the gene XIST is: -0.3382

Task 4: Differential Gene Expression Discussion

To ensure proper comparison between the full (task 1) and subsampled (task 2) datasets and the subsampled (task 2) and subsampled, randomised (task 3) datasets, the same preprocessing and filtering methods were applied before differential gene expression analysis was undertaken. The standard preprocessing of RNA-seq data was performed: filtering low count genes and performing trimmed mean normalisation. The datasets were also adjusted for heteroscedasity, as often with RNA-seq data, the variance of counts is mean-dependent[1].

a) The Full and Subsampled Datasets

Beginning with an analysis of the full and subsampled count datasets, the MDS plots are important to visualise how the male and female groups cluster. Comparing, figures 1 and 5 both groups are linearly separable by the first two dimensions alone. However, figure 5 does show some overlap between the male and female groups. This is expected with lower count RNA-seq data as we have less power to separate groups. In both figures 1 and 5, the replicate $'F00102\_F'$ doesn't cluster well with the female group and is likely a poor sample that could be safely removed.
Looking at tables 1 and 2, both analyses produced the same genes as significant, most of which were from chromosome Y. This was expected, since the groups, in both analyses, were split by sex, the most differentially expressed genes are on the X and Y chromosomes. The XIST gene is the only gene not from the Y chromosome that was significantly expressed in both analyses. Since this gene is only transcribed in the presence of multiple X chromosomes to inactivate the other X chromosome. With this, XIST is the only significant gene with a negative logFC as it is not expressed in the male group but is highly expressed in the female group (1d and 2d). Characteristic of lower read depth, the logFC in the subsampled dataset genes is mostly lower than the full count dataset.

Comparing the results from 1c and 2c, the proportion of significant differentially Y genes (after multiple testing correction) has decreased. Whilst these genes should still be differentially expressed between the sexes, the decrease in counts has led to a decrease in number of genes with which the statistical tests can confidently claim differential expression. Moreover, there is a decrease in significant differentially expressed genes -after multiple testing- in general (results 1b and 2b). It is known that increasing the read depth increases the number of false positive genes reported[2] and so performing a subsampled analysis on the same group data can control for false positives. This is a particularly good remedy for false positive genes that were not detected by Benjamini-Hochberg (B-H) multiple testing correction.

b) The Subsampled and Subsampled, Randomised Datasets

Beginning again with an inspection of the MDS plots for the subssampled (task 2) and subsampled, randomised (task 3) datasets figures 5 and 9 are drastically different. As before, figure 5 shows a clear linear separability in the first two dimensions, where figure 9 displays complete group overlap, (and likely would not be separable if higher dimensions were included). Leading from this, the analysis from task 3 acts as a negative control.
To gain confidence in the differentially expressed genes reported in table 2 it is important that they also don't arise in table 3 with the same logFC. Table 3 indicates that there are no genes reported on either chromosome X or Y and so there is no overlap between table 2. Further, the logFC values for these genes in table 3 is significantly lower than it is in tables 1 and 2. This indicates that these genes reported may not actually be differentially expressed. To analyse this further, 3b and 3c report on the significant genes and Y chromosome genes after B-H correction. 1 genes are found to be significant in 3b compared to 37 in 2b and from 3c no Y genes were significantly differentially expressed between groups compared to 16 in 2c. This is desirable from a negative control and is a result of male replicates in both groups decided by the *'randomgroup'* vector. Figures 2, 6 and 10 are heatmaps for tasks 1, 2 and 3 respectively. Unlike figures 2 and 6, figure 10 has lost the 'hot/cold' symmetry between groups and hierarchically clusters between group replicates before clustering within group replicates. Similarly to the expression of Y genes the XIST gene logFC has decreased substantially between 2d and 3d as a result of splitting female replicates between groups.

Conclusion

In sum, using a subsampled analysis alongside full and subsampled, randomised analyses we can successfully remove more false positives, missed by multiple testing correction and add a negative control. The combination of these three approaches creates a more robust analysis where there is a higher confidence that the differentially expressed genes reported in tasks 1 and 2, but not task 3, are truly differentially expressed.

References

1. Law CW., Chen Y., Shi W., et al. Voom: Precision Weights Unlock Linear Model Analysis Tools for RNA-seq Read Counts. Genome Biology, 15(2), R29, (2014).
2. Tarazona S., Garcia-Alcalde F., Dopazo J., et al. Differential Expression in RNA-seq: A matter of Depth. Genome Research, 21(12): 2213-2223, (2011).

```
sessionInfo()
```

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: OS X El Capitan 10.11.3
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## 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] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] RColorBrewer_1.1-2  NMF_0.20.6          Biobase_2.36.2
##  [4] BiocGenerics_0.22.1 cluster_2.0.6       rngtools_1.2.4
##  [7] pkgmaker_0.26.9     registry_0.3        xtable_1.8-2
## [10] edgeR_3.18.1        limma_3.32.10       knitr_1.17
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.14      magrittr_1.5      munsell_0.4.3
##  [4] doParallel_1.0.11 colorspace_1.3-2  gridBase_0.4-7
##  [7] lattice_0.20-35   rlang_0.1.4       foreach_1.4.3
## [10] plyr_1.8.4        stringr_1.2.0     highr_0.6
## [13] tools_3.4.1       grid_3.4.1        gtable_0.2.0
## [16] withr_2.1.0       iterators_1.0.8   lazyeval_0.2.1
## [19] digest_0.6.12     tibble_1.3.4      reshape2_1.4.2
## [22] ggplot2_2.2.1     codetools_0.2-15  evaluate_0.10.1
## [25] stringi_1.1.6     compiler_3.4.1    scales_0.5.0
## [28] locfit_1.5-9.1
```