7659 HW6

Guannan Shen

October 30, 2018

Contents


```
## set up workspace
library(knitr)
library(tidyverse)
library(edgeR)
library(cqn)
library(DESeq)
options(stringsAsFactors = F)
options(dplyr.width = Inf)
getwd()
```

```
## [1] "/home/guanshim/Documents/Stats/CIDA_OMICs/7659Stats_Genetics/HW6"
```

```
## not in function
"%nin%" <- Negate("%in%")

# ######## clean memory ##################### rm(list =
# ls()) gc()
```

1 HW6

1.1 1. Next Generation Sequencing: Differential Expression

1.1.1 (a) Calculate the RPKM for each gene in montgomery.subset using your own code. And show top genes by t-test.

```
## montgomery data from cqn
data(montgomery.subset)
## GC and gene length of montgomery
data(uCovar)
## total reads vector of length 10 containing the number of
## mapped reads for each sample
data(sizeFactors.subset)

dim(montgomery.subset)
```

```
## [1] 23552    10
```

```
dim(uCovar)
```

```
## [1] 23552     2
```

```
head(montgomery.subset)
```

```
##                 NA06985 NA06994 NA07037 NA10847 NA11920 NA11918 NA11931
## ENSG00000000419      69      54      67      70     146      80     300
## ENSG00000000457      53      37      27      41      59      18      89
## ENSG00000000460      12      25      33      22      97      46     132
## ENSG00000000938     168     270     140     103     231      20     296
## ENSG00000000971       0       0       0       0       0       0       0
## ENSG00000001036     100      74      61      60     137     112     197
##                 NA12003 NA12006 NA12287
## ENSG00000000419      84      99     126
## ENSG00000000457      32      46      62
## ENSG00000000460      53     209     119
## ENSG00000000938     524     462      55
## ENSG00000000971       0       0       1
## ENSG00000001036     153     201     187
```

```
head(uCovar)
```

```
##                 length gccontent
## ENSG00000000419   1207 0.3976802
## ENSG00000000457   2861 0.4606781
## ENSG00000000460   4912 0.4338355
## ENSG00000000938   3524 0.5749149
## ENSG00000000971   8214 0.3613343
## ENSG00000001036   2590 0.4312741
```

```
#### number of genes
n1 <- nrow(uCovar)
### number of samples
n2 <- length(montgomery.subset)

## the two groups are the first 1-5 subjects and then the
```

```
## second five subjects 6-10.
rpkm <- function(counts, length, total) {
    ## The RPKM
    RPKM = counts/(length/1000 * total/1e+06)
}
## the matrix to store
mont.rpkm <- matrix(NA, nrow = n1, ncol = n2)

## calculate the rpkm for whole dataset
for (i in 1:n2) {
    total <- sizeFactors.subset[i]
    for (j in 1:n1) {
        counts <- montgomery.subset[j, i]
        length <- uCovar[j, 1]
        mont.rpkm[j, i] <- rpkm(counts, length, total)
    }
}
head(mont.rpkm)
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 18.3967821 18.727499 17.980343 20.327935 25.370755 39.704334
## [2,]  5.9615344  5.413488  3.056866  5.023061  4.325357  3.768859
## [3,]  0.7861817  2.130468  2.176136  1.569881  4.141915  5.609894
## [4,] 15.3416979 32.071640 12.868342 10.244808 13.748771  3.399768
## [5,]  0.0000000  0.000000  0.000000  0.000000  0.000000  0.000000
## [6,] 12.4251111 11.959837  7.628875  8.119962 11.094522 25.904395
##           [,7]      [,8]      [,9]      [,10]
## [1,] 44.973098 16.497572 11.697838 25.1523734
## [2,]  5.628737  2.651430  2.293072  5.2214308
## [3,]  4.862442  2.557794  6.068280  5.8371963
## [4,] 15.198287 35.248725 18.697509  3.7604731
## [5,]  0.000000  0.000000  0.000000  0.0293333
## [6,] 13.762752 14.003600 11.068123 17.3963260
```

```
## the normalized data is mont.rpkm # carry out individual
## t-tests welch t-test

indi <- lapply(1:n1, function(row) {
    test = t.test(mont.rpkm[row, 1:5], mont.rpkm[row, 6:10],
        alternative = "two.sided")
    test.sum = c(rownames(uCovar)[row], test$p.value, test$statistic)
    test.sum
})
indi_t <- data.frame(matrix(unlist(indi), ncol = 3, byrow = TRUE))
colnames(indi_t) <- c("genenames", "pvalue", "tstatistic")
indi_t <- indi_t %>% mutate(pvalue = as.numeric(pvalue), tstatistic = as.numeric(tstatistic))
## top10 genes
kable(head(indi_t[order(abs(indi_t$pvalue), decreasing = F),
    ], 10), caption = "Top10 Genes by gene-specific t-test")
```

Table 1: Top10 Genes by gene-specific t-test

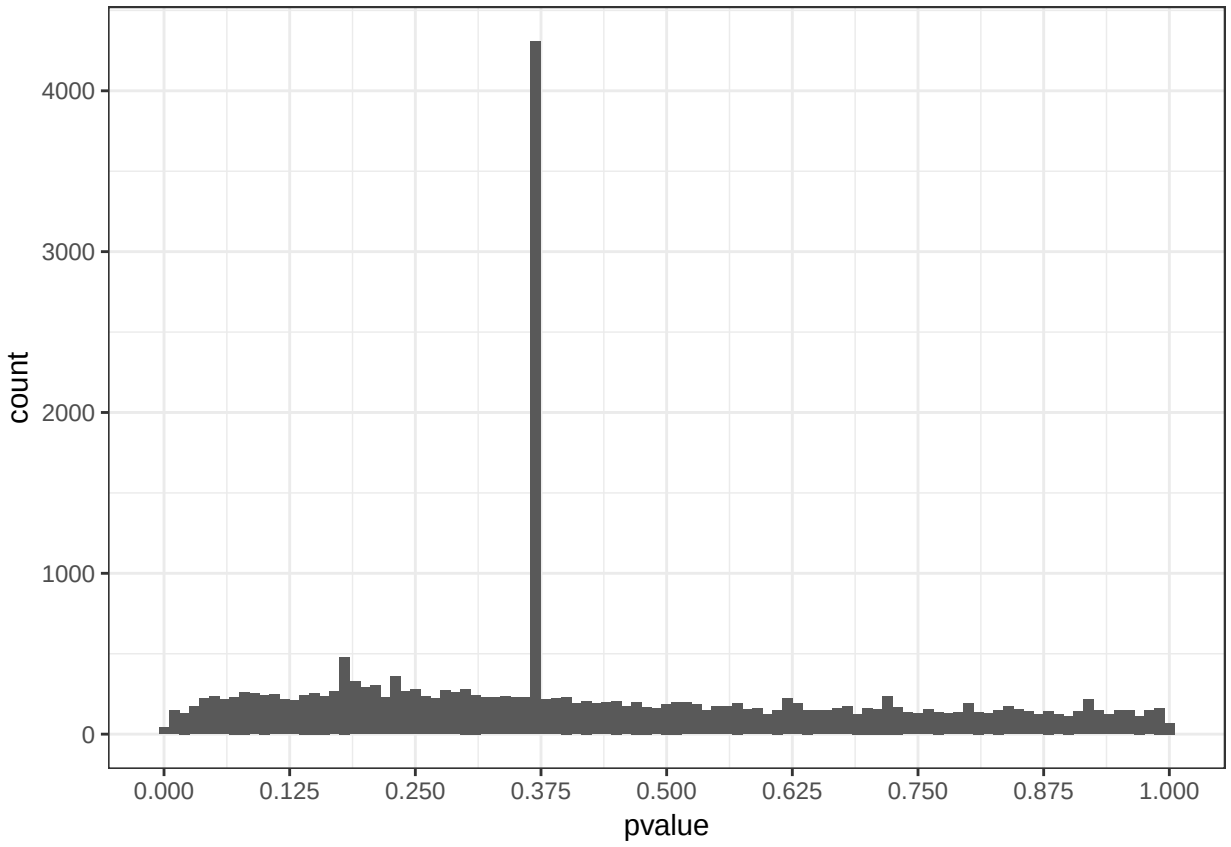
	genenames	pvalue	tstatistic
2774	ENSG00000105723	0.0000622	8.926705

	genenames	pvalue	tstatistic
5597	ENSG00000131116	0.0001396	6.824888
17998	ENSG00000228109	0.0001532	6.839553
9450	ENSG00000162889	0.0002489	6.295599
19134	ENSG00000232818	0.0003185	6.630320
6220	ENSG00000135677	0.0003786	-5.925278
18832	ENSG00000231615	0.0003813	-6.615887
6048	ENSG00000134686	0.0004748	5.840015
8648	ENSG00000155906	0.0006043	-5.721094
3126	ENSG00000108599	0.0006564	-5.395974

1.1.2 (b) It is good practice to plot the histogram of p-values. What shape would be expected? Plot the histogram of p-values from part a). What do you see? Extra Credit: What explains the odd pattern that you find?

I would expect to see a spike at small p-value region, in other words, at the far left end of the x-axis, and a uniform distribution of p-value elsewhere. In terms of this dataset, there is a big spike located around p-value = 0.375 and an approximately uniform distribution elsewhere. In my opinion, this odd pattern occured might because the RPKM normalization method did not work well here, influenced by highly expressed genes in some samples. Thus, the normalized data had a lot genes have similar values, for instance, a lot of zeros and ones, lead to a p-value spike at 0.375.

```
ggplot(indi_t, aes(pvalue)) + geom_histogram(binwidth = 0.01,
    bins = 30) + scale_x_continuous(breaks = c(0, 0.125, 0.25,
    0.375, 0.5, 0.625, 0.75, 0.875, 1)) + theme_bw()
```



1.1.3 (c) How many genes have at least 10 counts across subjects (i.e., total sum across the gene 10)? Create a new data frame with only those genes.

```
## number of genes have at least 10 counts across subjects
n10 <- sum(rowSums(montgomery.subset) >= 10)
## filtered data
mont.filtered <- montgomery.subset[rowSums(montgomery.subset) >=
    10, ]
## create an edgeR object
group.dge <- rep(c(1, 2), each = 5)
mont.dge <- DGEList(counts = mont.filtered, group = group.dge)
## the library size after filtering
mont.dge$samples$lib.size
```

```
##  [1] 2817342 2168185 2782270 2551303 4259582 1461731 5013427 3623044
##  [9] 6332824 3786172
```

```
colSums(mont.filtered)
```

```
## NA06985 NA06994 NA07037 NA10847 NA11920 NA11918 NA11931 NA12003 NA12006
## 2817342 2168185 2782270 2551303 4259582 1461731 5013427 3623044 6332824
## NA12287
## 3786172
```

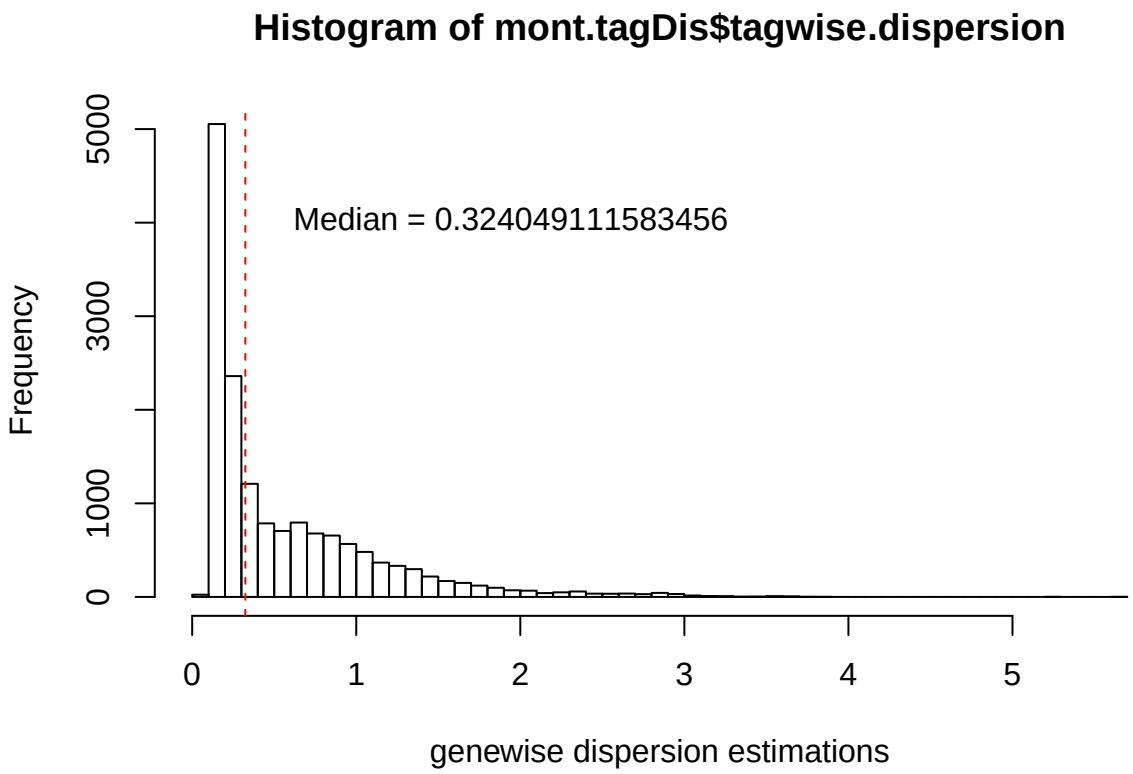
There are 15620 genes have at least 10 counts across subjects. The DGEList includes the lib.size as colSums by default.

1.1.4 (d) Using the estimateCommonDisp() function, what is the common dispersion estimate? Using the estimateTagwiseDisp() function, plot a histogram of the dispersion estimate for each gene.

```
## the common dispersion estimate
mont.comDis <- estimateCommonDisp(mont.dge)
mont.comDis$common.dispersion
```

```
## [1] 0.3496063
```

```
## dispersion estimate for each gene
mont.tagDis <- estimateTagwiseDisp(mont.comDis)
hist(mont.tagDis$tagwise.dispersion, xlab = "genewise dispersion estimations",
    breaks = 50)
abline(v = median(mont.tagDis$tagwise.dispersion), col = "red",
    lwd = 1, lty = 2)
text(0.5, 4000, paste("Median = ", median(mont.tagDis$tagwise.dispersion),
    sep = ""), pos = 4)
```



The common dispersion estimated is 0.3496063. According to the User's guide, edgeR uses the quantile-adjusted conditional maximum likelihood (qCML) method for experiments with single factor. This method performs best in the situation of many small samples with a common dispersion. The common dispersion is slightly larger than the median of the gene level dispersion estimates. This indicates the majority of genes have a dispersion close to the common dispersion. However, there still exists a long tail at the right end.

1.1.5 (e) Fit the negative binomial model (see Section 2.9.2 and 3.2 in the User's Guide) and test for differential expression using the common dispersion estimate and report the final results for the top 10 genes. How do the results change between the two approaches?

```
## with common dispersion
et.com <- exactTest(mont.comDis)
topTags(et.com)
```

```
## Comparison of groups:  2-1
##                      logFC    logCPM       PValue          FDR
## ENSG00000211642 -11.029374  6.504852 4.829699e-30 7.543989e-26
## ENSG00000211660 -10.678678  6.162008 1.501495e-28 1.172668e-24
## ENSG00000211890  -7.595319 10.228001 2.836330e-25 1.476783e-21
## ENSG00000211937  -7.122491  6.050909 3.167991e-22 1.237101e-18
## ENSG00000211638   7.650254  5.057996 2.330341e-20 7.279985e-17
## ENSG00000243063  -7.258785  5.828126 4.241787e-19 1.104278e-15
## ENSG00000211651   7.344596  3.878056 4.634741e-17 1.034209e-13
## ENSG00000238649   6.652606  4.320887 7.136667e-17 1.393434e-13
## ENSG00000211938   5.225670  7.022711 2.747992e-15 4.769294e-12
## ENSG00000253701   6.352322  3.786882 5.015365e-15 7.255669e-12
```

```
## get genes names
com_10 <- rownames(topTags(et.com))
com_10
```

```
##  [1] "ENSG00000211642" "ENSG00000211660" "ENSG00000211890"
##  [4] "ENSG00000211937" "ENSG00000211638" "ENSG00000243063"
##  [7] "ENSG00000211651" "ENSG00000238649" "ENSG00000211938"
## [10] "ENSG00000253701"
```

```
## with genewise dispersion
et.tag <- exactTest(mont.tagDis)
topTags(et.tag)
```

```
## Comparison of groups:  2-1
##                      logFC   logCPM       PValue          FDR
## ENSG00000253701   6.325635 3.786882 3.126976e-08 0.0004884336
## ENSG00000211892  -6.054972 2.662697 1.147079e-06 0.0089586866
## ENSG00000211642 -10.985263 6.504852 4.070063e-06 0.0211914617
## ENSG00000134184  -6.453823 1.042168 5.645983e-06 0.0220475621
## ENSG00000239223   3.390171 3.261859 7.946801e-06 0.0248258076
## ENSG00000148411  -2.289071 5.087784 1.466857e-05 0.0353180574
## ENSG00000211660 -10.627837 6.162008 1.582755e-05 0.0353180574
## ENSG00000180611   2.149954 5.272109 5.072124e-05 0.0986756939
## ENSG00000152952   4.965225 3.370829 6.122781e-05 0.0986756939
## ENSG00000189337  -3.945926 4.040531 6.343649e-05 0.0986756939
```

```
tag_10 <- rownames(topTags(et.tag))
tag_10
```

```
##  [1] "ENSG00000253701" "ENSG00000211892" "ENSG00000211642"
##  [4] "ENSG00000134184" "ENSG00000239223" "ENSG00000148411"
##  [7] "ENSG00000211660" "ENSG00000180611" "ENSG00000152952"
## [10] "ENSG00000189337"
```

```
## inner joint
top_top <- com_10[com_10 %in% tag_10]
top_top
```

```
## [1] "ENSG00000211642" "ENSG00000211660" "ENSG00000253701"
```

The exact test is based on the qCML methods. With common dispersion approach, only one parameter was estimated. Thus, the top10 genes have overall much smaller p-values and adjusted p-values (FDR). Based on FDR, only 7 genes in the genewise approach have adjusted p values less than 0.05. This approach estimated more parameters, has less power. Nevertheless, 3 genes in Top10 list were shared by two approaches: ENSG00000211642, ENSG00000211660, ENSG00000253701.

1.1.6 (f) For the top 10 genes based on the common dispersion, extract the raw counts (counts are contained in the counts value in the DGEList you created). What counts do you observe across the subjects for these genes? Using Ensembl what type of genes are in the top list?

```
# human RNA-seq data from lymphoblastoid cells raw counts of
# common
mont.dge$counts[rownames(mont.dge$counts) %in% com_10, ]
```

```
##                 NA06985 NA06994 NA07037 NA10847 NA11920 NA11918 NA11931
```

```
## ENSG00000211638       0       0       0       4       0       0       0
## ENSG00000211642       0    1831       0     155       1       0       0
## ENSG00000211651       0       0       0       1       1       1     682
## ENSG00000211660       0       0       0      11    2994       0       0
## ENSG00000211890       0       2       0    3216   45407      53       9
## ENSG00000211937       0       0       0     630    1727       0       0
## ENSG00000211938       2       0       0      85       0    1791       0
## ENSG00000238649       1       0       2       2       0     294       0
## ENSG00000243063       0       0       0       1    2356       9       0
## ENSG00000253701       0       0       0       4       0      77       8
##                 NA12003 NA12006 NA12287
## ENSG00000211638      82    1889       1
## ENSG00000211642       0       0       1
## ENSG00000211651       0       0       2
## ENSG00000211660       0       0       1
## ENSG00000211890      29      86      15
## ENSG00000211937       0      23       0
## ENSG00000211938      14     243       8
## ENSG00000238649       0       0       1
## ENSG00000243063       0       0       0
## ENSG00000253701     134      33     138
```

```
## raw counts of tag
mont.dge$counts[rownames(mont.dge$counts) %in% tag_10, ]
```

```
##                 NA06985 NA06994 NA07037 NA10847 NA11920 NA11918 NA11931
## ENSG00000134184      13       9       3      12       5       0       0
## ENSG00000148411     232      96     260     102      75      31      28
## ENSG00000152952       1       1       2       4       0     116       6
## ENSG00000180611       6      16      76      48      60      88     350
## ENSG00000189337     248     108      23       9       4       2      18
## ENSG00000211642       0    1831       0     155       1       0       0
## ENSG00000211660       0       0       0      11    2994       0       0
## ENSG00000211892       0      19      84      14      55       0       0
## ENSG00000239223       7       4       4       5       1      59      52
## ENSG00000253701       0       0       0       4       0      77       8
##                 NA12003 NA12006 NA12287
## ENSG00000134184       0       0       0
## ENSG00000148411      23     100      33
## ENSG00000152952       1      33      55
## ENSG00000180611     131     277     381
## ENSG00000189337      11       3       4
## ENSG00000211642       0       0       1
## ENSG00000211660       0       0       1
## ENSG00000211892       0       2       1
## ENSG00000239223      42     114      15
## ENSG00000253701     134      33     138
```

The raw counts for top10 genes by the common dispersion approach was shown above. Some genes with extremely high expression in some samples were selected by this method. Even though the high expression level are not consistent across samples within the group. It seems like the p-value was decided by only one sample in the group, while others are just zeros. Because this method does not take gene level variation into account. Usually, high expression genes tend to have high variations. In general, by looking at the raw counts, this approach is defective. By the Ensembl, most of the top10 genes are Immunoglobulin components. It is not surprising that this is RNA-Seq data from human lymphoblastoid cells. The top10 genes may not be

interesting to the investigators.

The raw counts by genewise approach are more reasonable. Top10 genes here tend to have more consistent raw counts within the group. And the difference between groups is on the group level, not caused by one or two samples from a group. Compared with the previous one, this method makes more sense. By the Ensembl, top10 genes are much more diverse. There are Glutathione S-transferase, Immunoglobulin, Mab-21 domain containing 2, NACC family member 2, Procollagen-lysine etc. This top10 gene list may be more interesting and informative to the investigators.

1.2 2. Next Generation Sequencing: Method Comparisons (access data from Recount.)

1.2.1 (a) Create a new data frame with genes that have at least 10 counts (summed across samples). How many genes are kept?

```
load(url("http://bowtie-bio.sourceforge.net/recount/ExpressionSets/bottomly_eset.RData"))
library(Biobase)
phenoData(bottomly.eset)  #gives information about the table
```

```
## An object of class 'AnnotatedDataFrame'
##   sampleNames: SRX033480 SRX033488 ... SRX033494 (21 total)
##   varLabels: sample.id num.tech.reps ... lane.number (5 total)
##   varMetadata: labelDescription
```

```
## phenoData(bottomly.eset)@data #outputs the table
phenoData(bottomly.eset)$strain  #gives mouse strain variable as vector
```

```
##  [1] C57BL/6J C57BL/6J C57BL/6J C57BL/6J C57BL/6J C57BL/6J C57BL/6J
##  [8] C57BL/6J C57BL/6J C57BL/6J DBA/2J   DBA/2J   DBA/2J   DBA/2J
## [15] DBA/2J   DBA/2J   DBA/2J   DBA/2J   DBA/2J   DBA/2J   DBA/2J
## Levels: C57BL/6J DBA/2J
```

```
featureNames(bottomly.eset)[1:10]  # gives first 10 genes in count table
```

```
##  [1] "ENSMUSG00000000001" "ENSMUSG00000000003" "ENSMUSG00000000028"
##  [4] "ENSMUSG00000000031" "ENSMUSG00000000037" "ENSMUSG00000000049"
##  [7] "ENSMUSG00000000056" "ENSMUSG00000000058" "ENSMUSG00000000078"
## [10] "ENSMUSG00000000085"
```

```
bottomly.count.table <- exprs(bottomly.eset)  #creates count table
dim(bottomly.count.table)  #36536x21
```

```
## [1] 36536    21
```

```
head(row.names(bottomly.count.table))  #names of genes
```

```
## [1] "ENSMUSG00000000001" "ENSMUSG00000000003" "ENSMUSG00000000028"
## [4] "ENSMUSG00000000031" "ENSMUSG00000000037" "ENSMUSG00000000049"
```

```
## genes that have at least 10 counts
n10.bot <- sum(rowSums(bottomly.count.table) >= 10)
## filtered dataset
bottom.filter <- bottomly.count.table[rowSums(bottomly.count.table) >=
    10, ]
dim(bottom.filter)
```

```
## [1] 11870    21
```

```
## DESeq dataset
condition.bot <- phenoData(bottomly.eset)$strain
cds.bot <- newCountDataSet(bottom.filter, condition.bot)

## edgeR
bot.dge <- DGEList(counts = bottom.filter, group = condition.bot)
```

There are 11870 genes kept.

1.2.2 (b) Calculate the size factors

```
## size factor for DESeq
cds <- estimateSizeFactors(cds.bot)
sizeFactors(cds)
```

```
## SRX033480 SRX033488 SRX033481 SRX033489 SRX033482 SRX033490 SRX033483
## 0.6439291 1.3453539 0.5784839 1.4295303 0.6355123 1.5239501 0.7933382
## SRX033476 SRX033478 SRX033479 SRX033472 SRX033473 SRX033474 SRX033475
## 1.1271894 1.0772279 0.8984474 0.8886335 1.0255149 0.7987292 0.7795619
## SRX033491 SRX033484 SRX033492 SRX033485 SRX033493 SRX033486 SRX033494
## 1.6161933 0.9881892 1.5720164 0.7557824 1.5922159 0.8264069 1.4715139
```

```
## size factor for edgeR TMM
y <- calcNormFactors(bot.dge)
y$samples$norm.factors
```

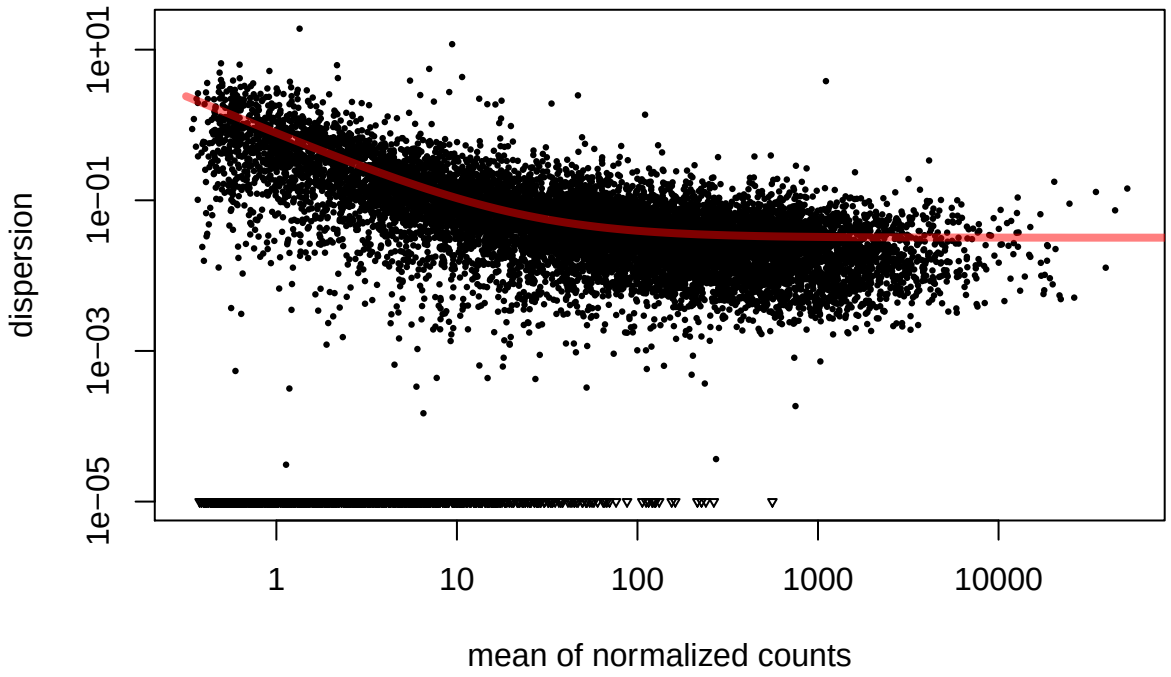
```
##  [1] 0.9858019 0.9789546 1.0094511 0.9965089 0.9880781 0.9840943 0.9942930
##  [8] 0.9971411 1.0023364 0.9963740 1.0193662 0.9924446 0.9975642 1.0124273
## [15] 1.0133338 1.0370408 1.0070477 1.0135210 0.9960492 0.9831050 0.9969959
```

Size factors are used for normalization. Size factors are the effective library sizes. For instance, in terms of two group comparison scenario, if the library was sequenced twice as deeply as the other one, lead to the counts of non-differentially expressed genes in one sample are, on average, twice as high as in another. The size factor in group 1 is the twice of the group 2.

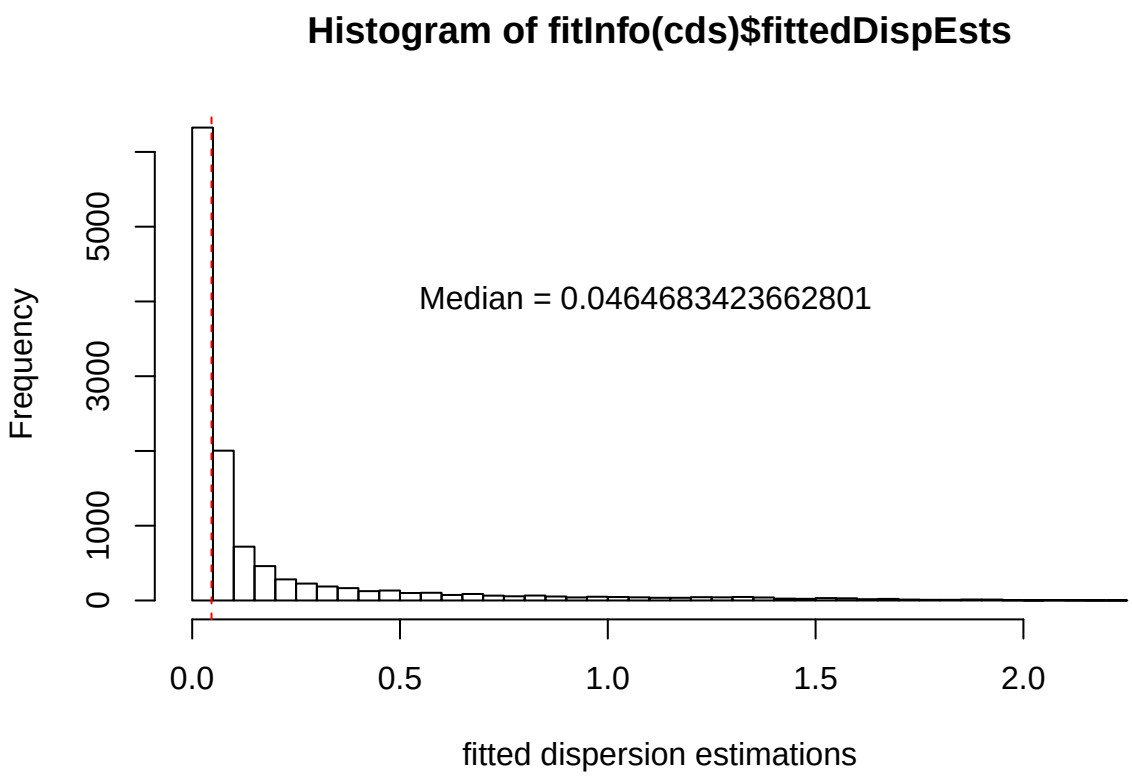
The size factors calculated by DESeq range from 0.578 to 1.616, however, size factors by the 2nd method are basically distributed around 1. Based on the TMM method, there is no sizable/obvious difference of library sizes among samples. However, since those samples labeled by different experiment number, size factors calculated by DESeq are more reasonable.

1.2.3 (c) Calculate the DESeq dispersions using the "local" method

```
## dispersion by DESeq
cds <- estimateDispersions(cds)
plotDispEsts(cds)
```

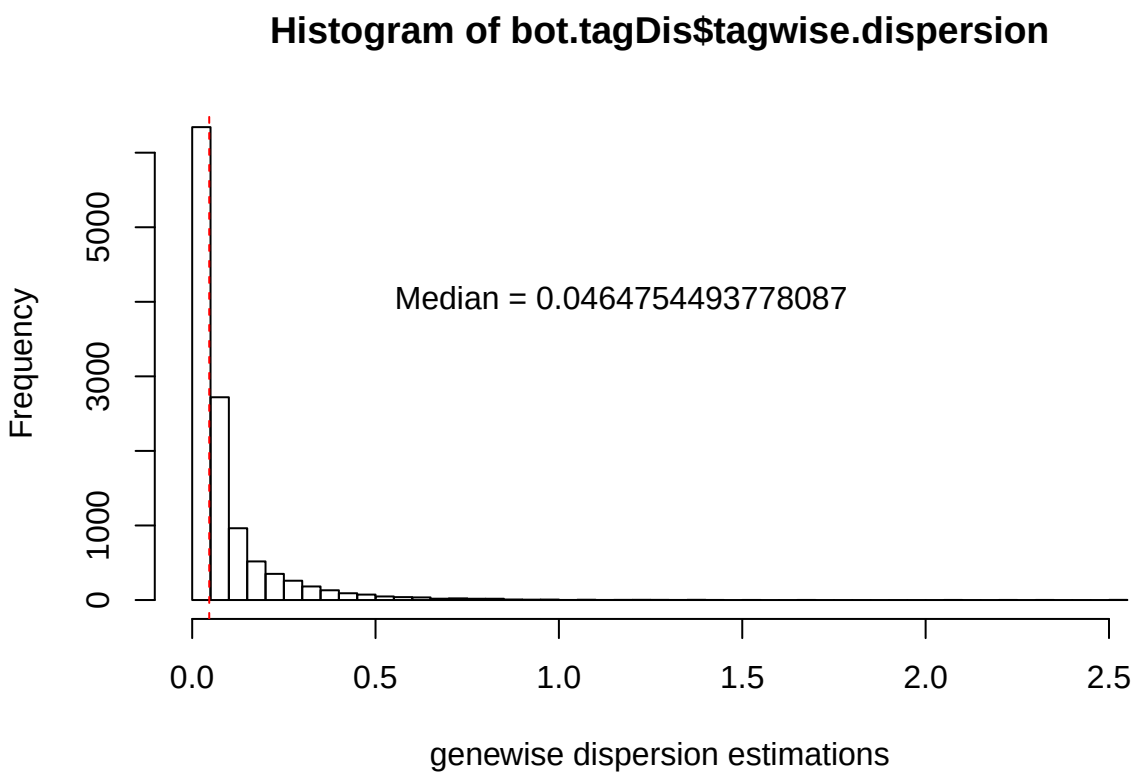



```
hist(fitInfo(cds)$fittedDispEsts, xlab = "fitted dispersion estimations",
    breaks = 50)
abline(v = median(fitInfo(cds)$fittedDispEsts), col = "red",
    lwd = 1, lty = 2)
text(0.5, 4000, paste("Median = ", median(fitInfo(cds)$fittedDispEsts),
    sep = ""), pos = 4)
```



```r
## edgeR ## the common dispersion estimate
bot.comDis <- estimateCommonDisp(bot.dge)

## dispersion estimate for each gene
bot.tagDis <- estimateTagwiseDisp(bot.comDis)
hist(bot.tagDis$tagwise.dispersion, xlab = "genewise dispersion estimations",
    breaks = 50)
abline(v = median(bot.tagDis$tagwise.dispersion), col = "red",
    lwd = 1, lty = 2)
text(0.5, 4000, paste("Median = ", median(bot.tagDis$tagwise.dispersion),
    sep = ""), pos = 4)
```



In general, two sets of dispersions are very close, except that the dispersion estimates from the DESeq are more obviously right skewed. The variance of y is modelled as $v = s\mu + \alpha s^2\mu^2$ in DESeq, where μ is the expected nomarlized count value. s is the size factor, α is the dispersion value. In comparison, in edgeR, the variance of y is modelled as $v = \mu + \phi\mu^2$. The difference is the size factor and the square root in the two equations. The dispersion by edgeR is like the square root tranformation of the one by DESeq, thus the former is less right skewed. In addition, the size factors in DESeq also impact the estimation of the dispersion to some extent.

1.2.4 (d) Test for differences between the two strains using DESeq

```r
## DE by DESeq
res <- nbinomTest(cds, "C57BL/6J", "DBA/2J")
res.p <- res %>% filter(padj <= 0.05)

## number of sig by DESeq
n.deseq <- nrow(res.p)
```

```
## edge R
et.bot.tag <- exactTest(bot.tagDis)
## calculate BH adjusted p
et.bot.tag$table$padj <- p.adjust(et.bot.tag$table$PValue, method = "BH")
###
bot.p <- et.bot.tag$table %>% filter(padj <= 0.05) %>% mutate(id = rownames(et.bot.tag$table)[et.bot.tag
    0.05]) %>% select(id, everything())
n.bot <- nrow(bot.p)

## overlap
sum(bot.p$id %in% res.p$id)
```

```
## [1] 604
```

```
## choose one not overlap
set.seed(1)
ran.gene <- sample(bot.p$id[bot.p$id %nin% res.p$id], 1)
ran.gene
```

```
## [1] "ENSMUSG00000025410"
```

```
## DESeq sample
res[res$id == ran.gene, c(1, 5:8)]
```

```
##                       id foldChange log2FoldChange       pval      padj
## 3187 ENSMUSG00000025410   1.188174      0.2487461 0.04854442 0.3538084
```

```
## edgeR
bot.p[bot.p$id == ran.gene, ]
```

```
##                      id     logFC   logCPM     PValue      padj
## 309 ENSMUSG00000025410 0.2763625 8.776548 0.00285056 0.0323357
```

604 genes are found to be differentially expressed by DESeq. 1265 genes are found to be differentially expressed by edgeR. The DESeq method is more stringent. The genes found by DESeq are all covered by the significant gene list of edgeR. Thus there are 604 overlapping genes.

I randomly sampled one gene: ENSMUSG00000025410. For this gene, the logFC after normalization is slightly larger by the edgeR method. Meanwhile, the p value is much smaller in edgeR. Thus, we know that the variation of counts for this gene estimated by edgeR is much smaller compared with DESeq. In this way, we have the smaller p-value by edgeR.