BIOS 7659 Homework 6

Tim Vigers

01 November 2020

data("montgomery.subset")  
data("uCovar")

# 1. Next Generation Sequencing: Differential Expression

## a) Calculate RPKM and perform a t test

RPKM stands for reads per kilobases per million reads and is calculated by

lg = uCovar$length / 1000  
t = sum(unlist(montgomery.subset))  
rpkm = montgomery.subset/(lg\*t)

Do a standard t test between the groups for each gene:

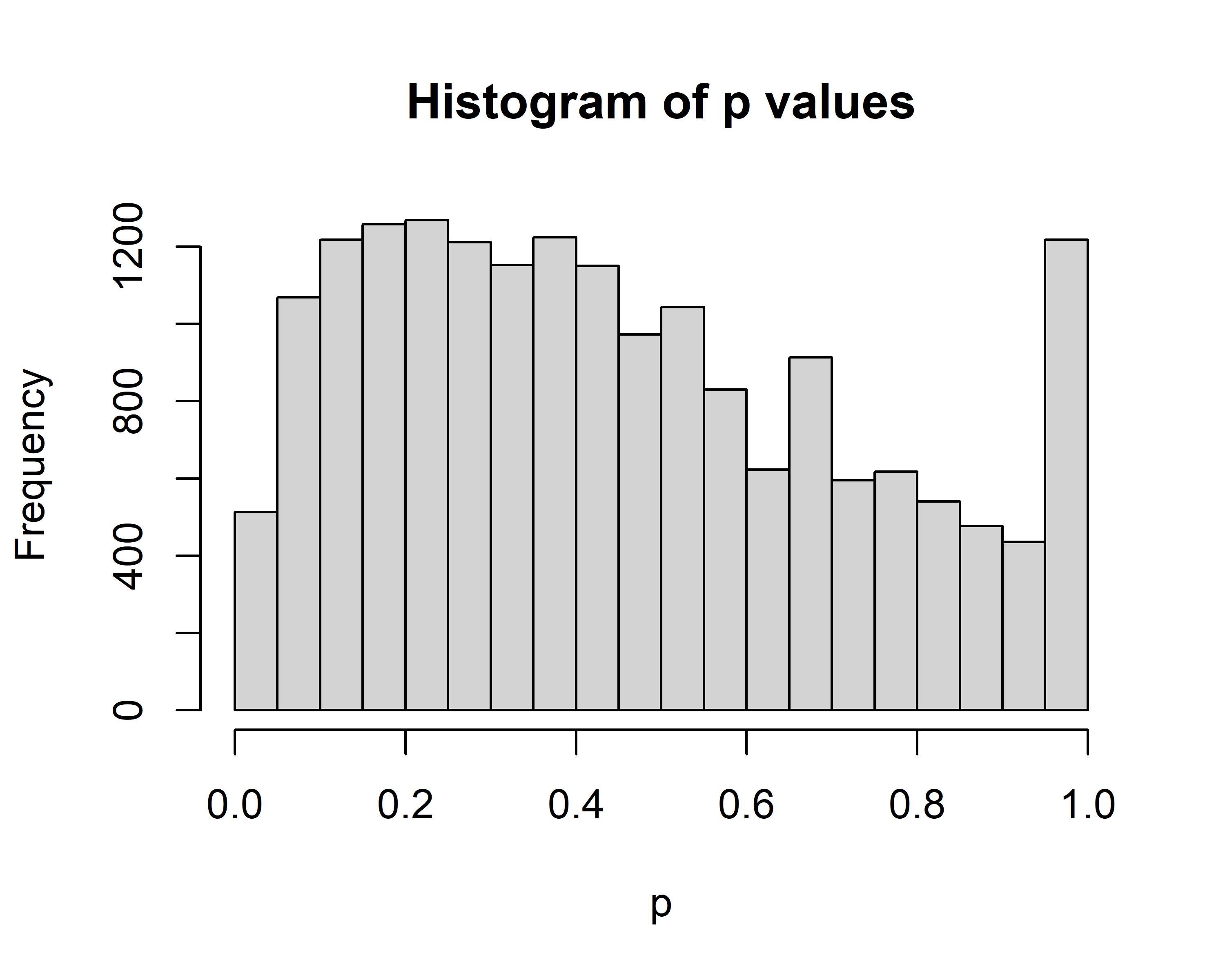
# T test for each row  
t\_tests = apply(rpkm,1,function(x){  
 group1 = as.numeric(x[1:5])  
 group2 = as.numeric(x[6:10])  
 if(var(group1)==0 | var(group2)==0){ # Skip those with constant values   
 return(c(NA,NA))  
 } else {  
 t <- t.test(group1,group2)  
 return(c(t$statistic,t$p.value))  
 }  
})  
# Format and print  
t\_tests = as.data.frame(t(t\_tests))  
t\_tests = t\_tests %>% rownames\_to\_column() %>%   
 set\_names(c("Gene","T","p")) %>% arrange(desc(abs(T)))   
  
t\_tests %>% head(10) %>% flextable(.) %>%   
 set\_caption("Top 10 Genes by t-statistic") %>% autofit(.)

Top 10 Genes by t-statistic

| Gene | T | p |
| --- | --- | --- |
| ENSG00000245208 | 5.498052 | 0.0005766918 |
| ENSG00000228109 | 5.258951 | 0.0011172910 |
| ENSG00000158985 | -5.146613 | 0.0053740327 |
| ENSG00000175274 | 4.826005 | 0.0023624029 |
| ENSG00000154710 | 4.810702 | 0.0021405930 |
| ENSG00000197747 | 4.714045 | 0.0018518080 |
| ENSG00000170802 | -4.686417 | 0.0028152695 |
| ENSG00000125629 | -4.624472 | 0.0035333403 |
| ENSG00000185947 | -4.575696 | 0.0085521476 |
| ENSG00000005187 | -4.495972 | 0.0026741415 |

## b) Plot the histogram of p-values

hist(t\_tests$p,main = "Histogram of p values",xlab = "p")



Normally we would expect a uniform distribution of p values, but this distribution appears to have a peak at around 0.3 and another at p = 1. My guess is that this is because we have only filtered genes with all 0 counts, but kept other genes with counts so low that they are effectively 0.

## c) Filter genes by total counts

# DOESN’T DGELIST AUTOMATICALLY INCLUDE lib.size

with the total reads per subject?

# Remove genes with low counts  
filtered = montgomery.subset[rowSums(montgomery.subset)>=10,]  
# Create edgeR object  
filtered\_dge = DGEList(filtered,group = c(rep(1,5),rep(2,5)))

## d) Calculate TMM normalization factors

norm = calcNormFactors(filtered\_dge)  
autofit(flextable(norm$samples))

| group | lib.size | norm.factors |
| --- | --- | --- |
| 1 | 2817342 | 0.9420009 |
| 1 | 2168185 | 1.0077134 |
| 1 | 2782270 | 0.9975042 |
| 1 | 2551303 | 0.9790312 |
| 1 | 4259582 | 0.9025138 |
| 2 | 1461731 | 1.1474002 |
| 2 | 5013427 | 1.0833630 |
| 2 | 3623044 | 0.9307130 |
| 2 | 6332824 | 0.9784469 |
| 2 | 3786172 | 1.0558556 |

The effective library sizes are generally similar to the column sums because the normalization factors are fairly close to 1. A normalization factor > 1 increases the library size, which is similar to downscaling the counts (and vice versa for factors < 1). So, the effective library size for sample 6 is increased by about 115%. Conversely, the effective library size for sample 5 is decreased by approximately 90%, which suggests that there are a small number of high-count sequences that need to be adjusted for.

## e) Use the estimateDisp() function to calculate the common, trended and tagwise dispersions

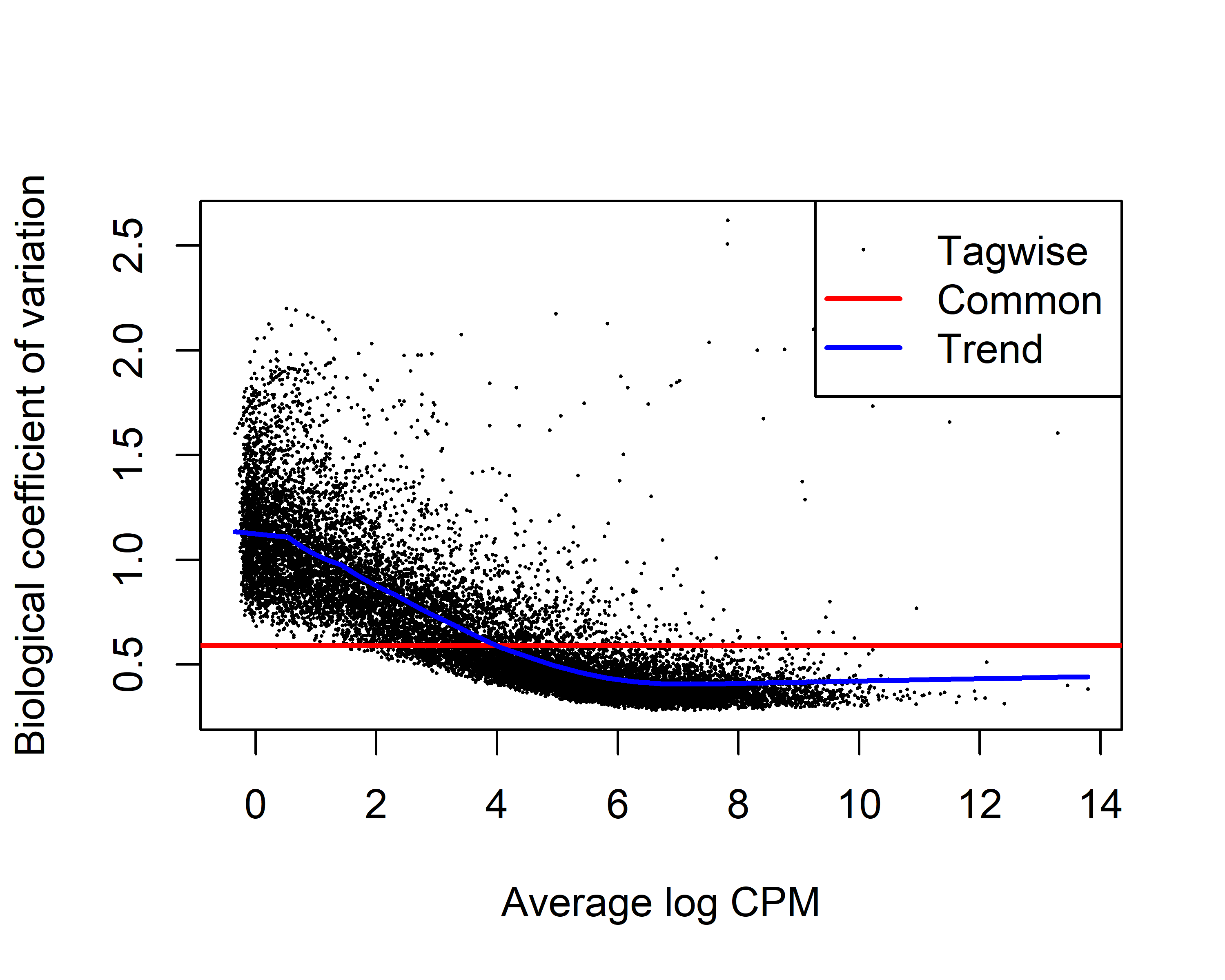
filtered\_dge = estimateDisp(filtered\_dge)

## Design matrix not provided. Switch to the classic mode.

The common dispersion estimate is approximately 0.349. Plot the tagwise dispersion estimate for each gene vs. the average log counts per million:

# IS IT OKAY TO USE THE BUILT IN FUNCTION THAT PLOTS ON SQUARE ROOT SCALE?

plotBCV(filtered\_dge)



The common dispersion estimate only seems to work well for a narrow range of log CPM around 4. It appears to underestimate dispersion for the lower count genes and overestimate the higher average count genes.

## f) Fit the negative binomial model

### Using the common dispersion estimate

et <- exactTest(filtered\_dge,dispersion = "common")  
top\_common <- topTags(et) %>% as.data.frame(.) %>%   
 rownames\_to\_column(.,var = "Gene")  
autofit(flextable(top\_common))

| Gene | logFC | logCPM | PValue | FDR |
| --- | --- | --- | --- | --- |
| ENSG00000211642 | -11.029384 | 6.504848 | 4.646432e-30 | 7.257727e-26 |
| ENSG00000211660 | -10.678692 | 6.162012 | 1.447959e-28 | 1.130856e-24 |
| ENSG00000211890 | -7.595350 | 10.228002 | 2.730213e-25 | 1.421531e-21 |
| ENSG00000211937 | -7.122400 | 6.050910 | 3.073339e-22 | 1.200139e-18 |
| ENSG00000211638 | 7.650287 | 5.058009 | 2.270402e-20 | 7.092736e-17 |
| ENSG00000243063 | -7.259087 | 5.828130 | 4.131889e-19 | 1.075668e-15 |
| ENSG00000211651 | 7.344597 | 3.878077 | 4.544282e-17 | 1.014024e-13 |
| ENSG00000238649 | 6.652599 | 4.320843 | 6.989085e-17 | 1.364619e-13 |
| ENSG00000211938 | 5.225677 | 7.022704 | 2.688483e-15 | 4.666012e-12 |
| ENSG00000253701 | 6.352338 | 3.786868 | 4.927015e-15 | 7.153687e-12 |

### Using the tag-wise dispersion estimates

et <- exactTest(filtered\_dge,dispersion = "tagwise")  
top\_tagwise <- topTags(et) %>% as.data.frame(.) %>%   
 rownames\_to\_column(.,var = "Gene")  
autofit(flextable(top\_tagwise))

| Gene | logFC | logCPM | PValue | FDR |
| --- | --- | --- | --- | --- |
| ENSG00000253701 | 6.321934 | 3.786868 | 1.225411e-07 | 0.001914093 |
| ENSG00000211892 | -6.059085 | 2.662691 | 2.531525e-06 | 0.019771212 |
| ENSG00000134184 | -6.453312 | 1.042133 | 4.336917e-06 | 0.022580879 |
| ENSG00000239223 | 3.390230 | 3.261841 | 7.399928e-06 | 0.028896719 |
| ENSG00000148411 | -2.288891 | 5.087781 | 1.593862e-05 | 0.049792261 |
| ENSG00000211642 | -10.981435 | 6.504848 | 3.127496e-05 | 0.081419147 |
| ENSG00000180611 | 2.150017 | 5.272110 | 6.616724e-05 | 0.147647480 |
| ENSG00000211660 | -10.624115 | 6.162012 | 9.998158e-05 | 0.178008287 |
| ENSG00000189337 | -3.946439 | 4.040523 | 1.112568e-04 | 0.178008287 |
| ENSG00000023445 | 1.276689 | 7.313150 | 1.152851e-04 | 0.178008287 |

There are only 3 overlapping genes in the top 10 table for the two methods.Also, the top genes as determined by the common dispersion estimate approach appear to be driven more by fold change than those that are most significant using the tagwise method, because the table is essentially in decreasing order of logFC (with a couple of minor exceptions).

## g) Extract the raw counts

### For the top 10 genes based on the common dispersion

top\_common\_counts = filtered\_dge$counts[top\_common$Gene,] %>%  
 as.data.frame(.) %>% rownames\_to\_column(var = "Gene")  
set\_table\_properties(flextable(top\_common\_counts),  
 width = .5, layout = "autofit")

| Gene | NA06985 | NA06994 | NA07037 | NA10847 | NA11920 | NA11918 | NA11931 | NA12003 | NA12006 | NA12287 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ENSG00000211642 | 0 | 1831 | 0 | 155 | 1 | 0 | 0 | 0 | 0 | 1 |
| ENSG00000211660 | 0 | 0 | 0 | 11 | 2994 | 0 | 0 | 0 | 0 | 1 |
| ENSG00000211890 | 0 | 2 | 0 | 3216 | 45407 | 53 | 9 | 29 | 86 | 15 |
| ENSG00000211937 | 0 | 0 | 0 | 630 | 1727 | 0 | 0 | 0 | 23 | 0 |
| ENSG00000211638 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 82 | 1889 | 1 |
| ENSG00000243063 | 0 | 0 | 0 | 1 | 2356 | 9 | 0 | 0 | 0 | 0 |
| ENSG00000211651 | 0 | 0 | 0 | 1 | 1 | 1 | 682 | 0 | 0 | 2 |
| ENSG00000238649 | 1 | 0 | 2 | 2 | 0 | 294 | 0 | 0 | 0 | 1 |
| ENSG00000211938 | 2 | 0 | 0 | 85 | 0 | 1791 | 0 | 14 | 243 | 8 |
| ENSG00000253701 | 0 | 0 | 0 | 4 | 0 | 77 | 8 | 134 | 33 | 138 |

### For the top 10 genes based on tagwise dispersion

top\_tagwise\_counts = filtered\_dge$counts[top\_tagwise$Gene,] %>%  
 as.data.frame(.) %>% rownames\_to\_column(var = "Gene")  
set\_table\_properties(flextable(top\_tagwise\_counts),  
 width = .5, layout = "autofit")

| Gene | NA06985 | NA06994 | NA07037 | NA10847 | NA11920 | NA11918 | NA11931 | NA12003 | NA12006 | NA12287 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ENSG00000253701 | 0 | 0 | 0 | 4 | 0 | 77 | 8 | 134 | 33 | 138 |
| ENSG00000211892 | 0 | 19 | 84 | 14 | 55 | 0 | 0 | 0 | 2 | 1 |
| ENSG00000134184 | 13 | 9 | 3 | 12 | 5 | 0 | 0 | 0 | 0 | 0 |
| ENSG00000239223 | 7 | 4 | 4 | 5 | 1 | 59 | 52 | 42 | 114 | 15 |
| ENSG00000148411 | 232 | 96 | 260 | 102 | 75 | 31 | 28 | 23 | 100 | 33 |
| ENSG00000211642 | 0 | 1831 | 0 | 155 | 1 | 0 | 0 | 0 | 0 | 1 |
| ENSG00000180611 | 6 | 16 | 76 | 48 | 60 | 88 | 350 | 131 | 277 | 381 |
| ENSG00000211660 | 0 | 0 | 0 | 11 | 2994 | 0 | 0 | 0 | 0 | 1 |
| ENSG00000189337 | 248 | 108 | 23 | 9 | 4 | 2 | 18 | 11 | 3 | 4 |
| ENSG00000023445 | 390 | 228 | 206 | 178 | 321 | 444 | 1397 | 785 | 921 | 673 |