BIOS 7659 Homework 6

Tim Vigers

02 November 2020

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

# 1. 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(.)

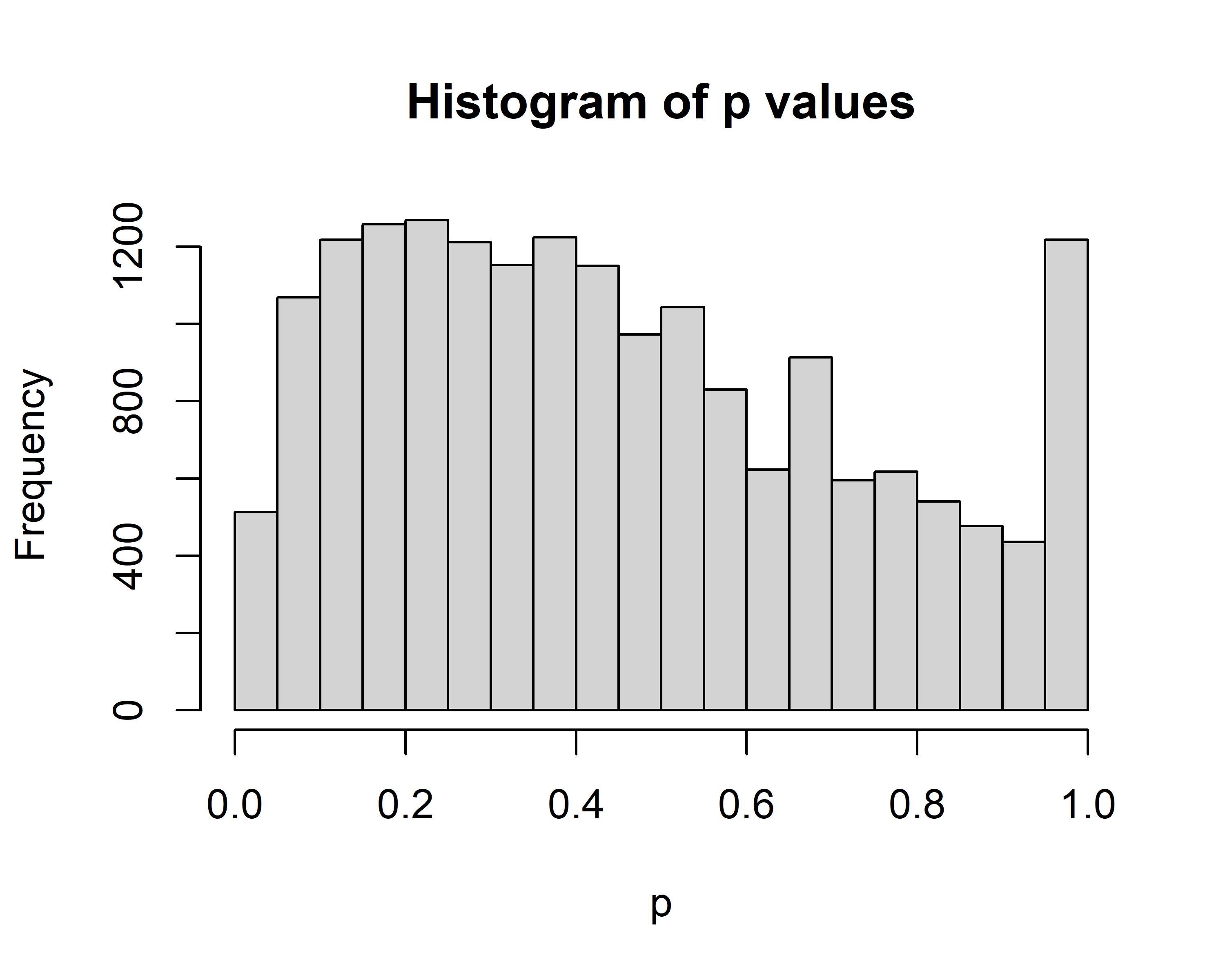
Top 10 Genes by t-statistic

| 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

# THIS IS PROBABLY SOMETHING TO DO WITH RPKM - LOOK INTO IT

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 = rep(c(1,2),each=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

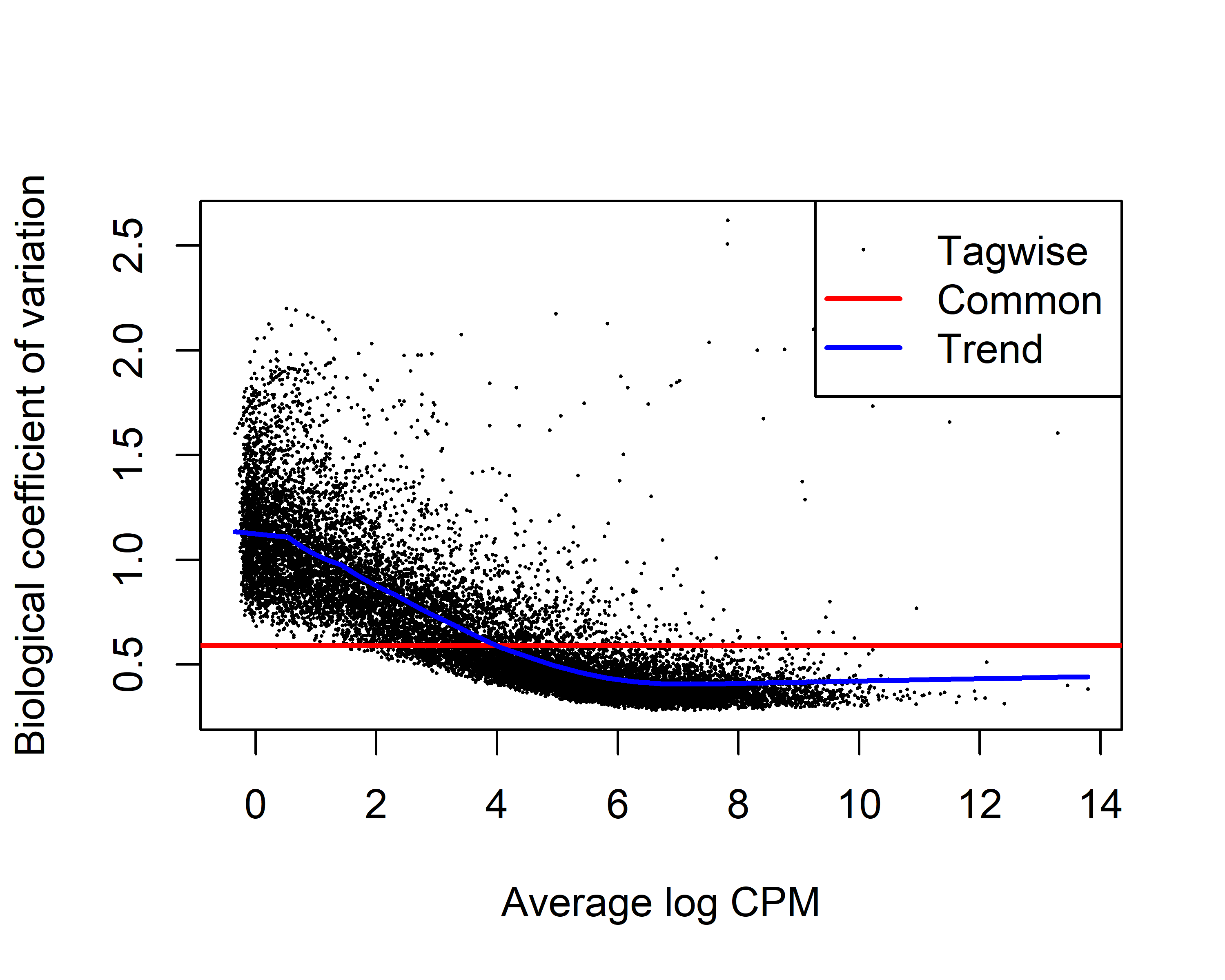
filtered\_dge = estimateDisp(filtered\_dge)

## 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).

# ESSENTIALLY BECAUSE THE VARIABILITY (DISPERSION) ISN’T REALLY ACCOUNTED FOR CORRECTLY WITH COMMON

## 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 |

# Most of these top genes have…

### 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 |

# These are more evenly spread…

### Get Ensembl information for the top genes

#### Based on common dispersion

ensembl = useEnsembl(biomart="ensembl", dataset="hsapiens\_gene\_ensembl")  
en = getBM(attributes=c('ensembl\_gene\_id','description'),   
 filters ='ensembl\_gene\_id', values = top\_common\_counts$Gene,   
 mart = ensembl)  
set\_table\_properties(flextable(en),  
 width = .75, layout = "autofit")

| ensembl\_gene\_id | description |
| --- | --- |
| ENSG00000211638 | immunoglobulin lambda variable 8-61 [Source:HGNC Symbol;Acc:HGNC:5931] |
| ENSG00000211642 | immunoglobulin lambda variable 10-54 [Source:HGNC Symbol;Acc:HGNC:5884] |
| ENSG00000211651 | immunoglobulin lambda variable 1-44 [Source:HGNC Symbol;Acc:HGNC:5879] |
| ENSG00000211660 | immunoglobulin lambda variable 2-23 [Source:HGNC Symbol;Acc:HGNC:5890] |
| ENSG00000211890 | immunoglobulin heavy constant alpha 2 (A2m marker) [Source:HGNC Symbol;Acc:HGNC:5479] |
| ENSG00000211937 | immunoglobulin heavy variable 2-5 [Source:HGNC Symbol;Acc:HGNC:5576] |
| ENSG00000211938 | immunoglobulin heavy variable 3-7 [Source:HGNC Symbol;Acc:HGNC:5620] |
| ENSG00000238649 | small nucleolar RNA, C/D box 42A [Source:HGNC Symbol;Acc:HGNC:10180] |
| ENSG00000243063 | immunoglobulin kappa variable 3-7 (non-functional) [Source:HGNC Symbol;Acc:HGNC:5821] |

#### Based on tagwise dispersion

en = getBM(attributes=c('ensembl\_gene\_id','description'),   
 filters ='ensembl\_gene\_id', values = top\_tagwise\_counts$Gene,   
 mart = ensembl)  
set\_table\_properties(flextable(en),  
 width = .75, layout = "autofit")

| ensembl\_gene\_id | description |
| --- | --- |
| ENSG00000023445 | baculoviral IAP repeat containing 3 [Source:HGNC Symbol;Acc:HGNC:591] |
| ENSG00000134184 | glutathione S-transferase mu 1 [Source:HGNC Symbol;Acc:HGNC:4632] |
| ENSG00000148411 | NACC family member 2 [Source:HGNC Symbol;Acc:HGNC:23846] |
| ENSG00000180611 | Mab-21 domain containing 2 [Source:HGNC Symbol;Acc:HGNC:30438] |
| ENSG00000189337 | kazrin, periplakin interacting protein [Source:HGNC Symbol;Acc:HGNC:29173] |
| ENSG00000211642 | immunoglobulin lambda variable 10-54 [Source:HGNC Symbol;Acc:HGNC:5884] |
| ENSG00000211660 | immunoglobulin lambda variable 2-23 [Source:HGNC Symbol;Acc:HGNC:5890] |
| ENSG00000211892 | immunoglobulin heavy constant gamma 4 (G4m marker) [Source:HGNC Symbol;Acc:HGNC:5528] |
| ENSG00000239223 | ribosomal protein L34 pseudogene 31 [Source:HGNC Symbol;Acc:HGNC:36899] |

# 2. Remove Unwanted Variation.

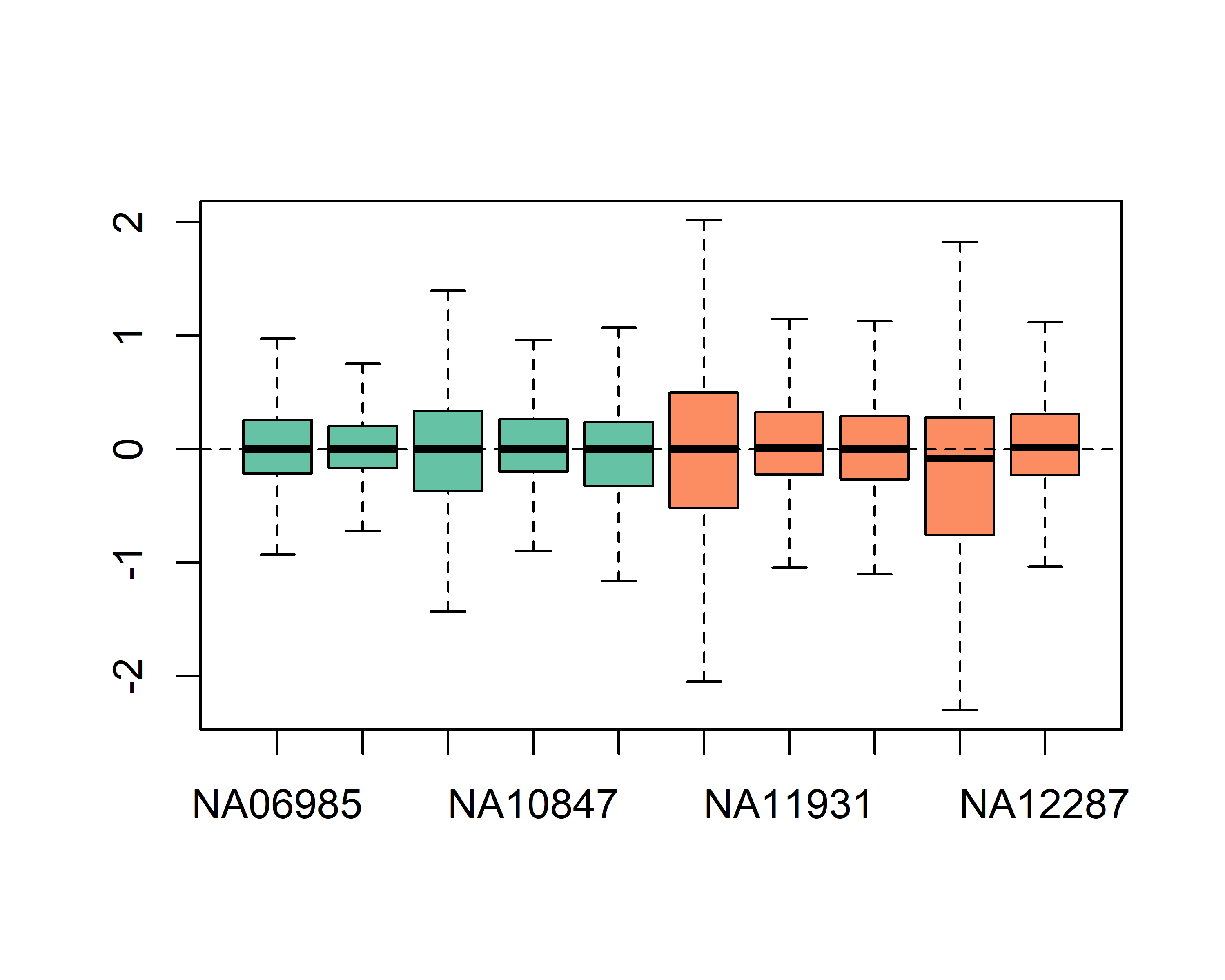
## a) Create a design matrix

# Fit the GLM   
group = rep(c(1,2),each=5)  
design = model.matrix(~group)  
filtered\_dge = DGEList(filtered)  
filtered\_dge = calcNormFactors(filtered\_dge,"upperquartile")  
filtered\_dge = estimateDisp(filtered\_dge,design = design)  
glm\_fit\_a = glmFit(filtered\_dge,  
 dispersion = filtered\_dge$common.dispersion)  
glm\_lrt\_a = glmLRT(glm\_fit\_a)$table  
glm\_lrt\_a$PValue\_FDR = p.adjust(glm\_lrt\_a$PValue,"fdr")

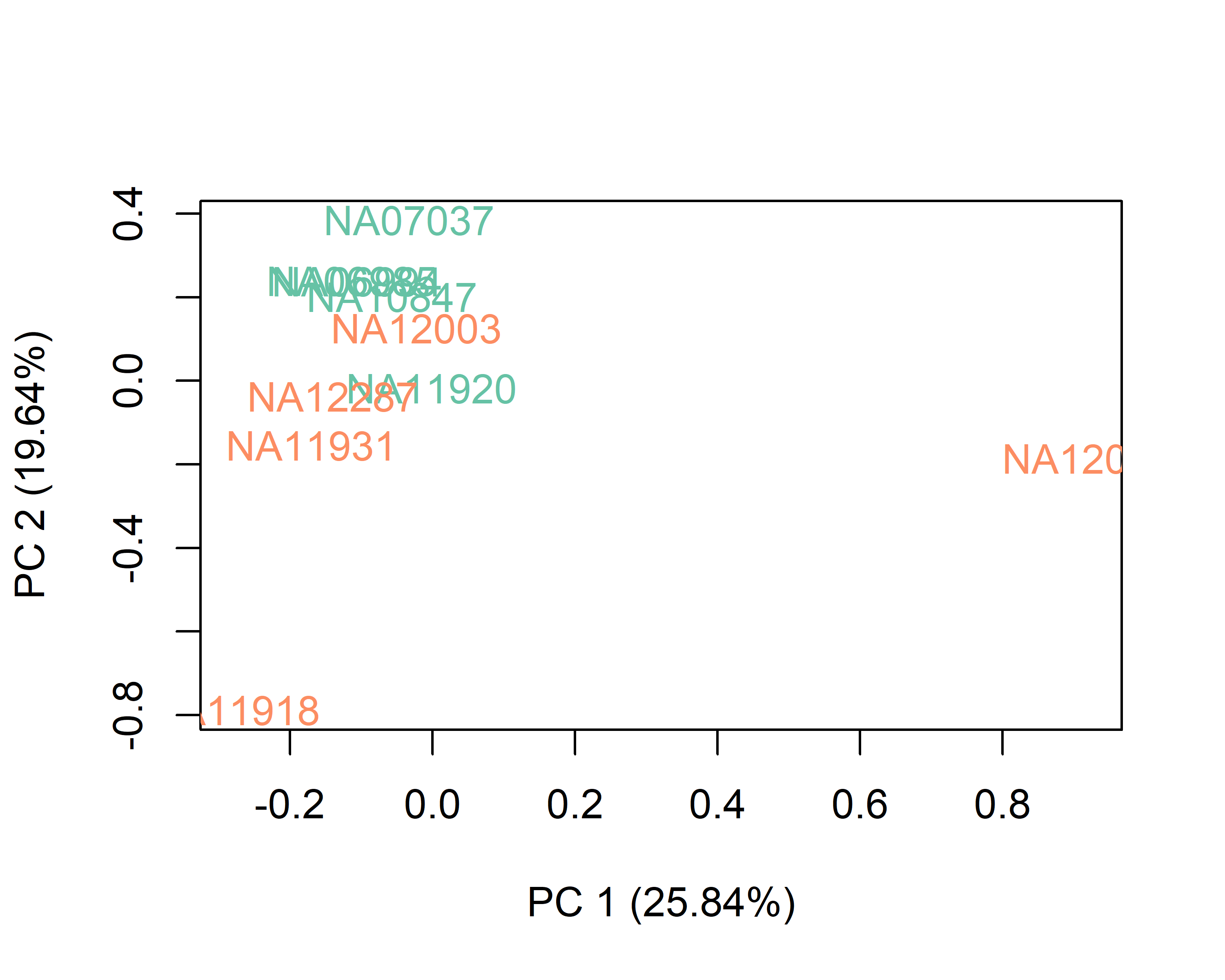
Using the likelihood ratio test, there are 565 genes with FDR-adjusted p values < 0.05.

## b) Fit edgeR models that adjust for unwanted variation

colors = brewer.pal(3, "Set2")  
set = newSeqExpressionSet(  
 as.matrix(filtered),  
 phenoData = data.frame(group,row.names=colnames(filtered)))  
set = betweenLaneNormalization(set, which="upper")  
plotRLE(set, outline=FALSE,col=colors[group])



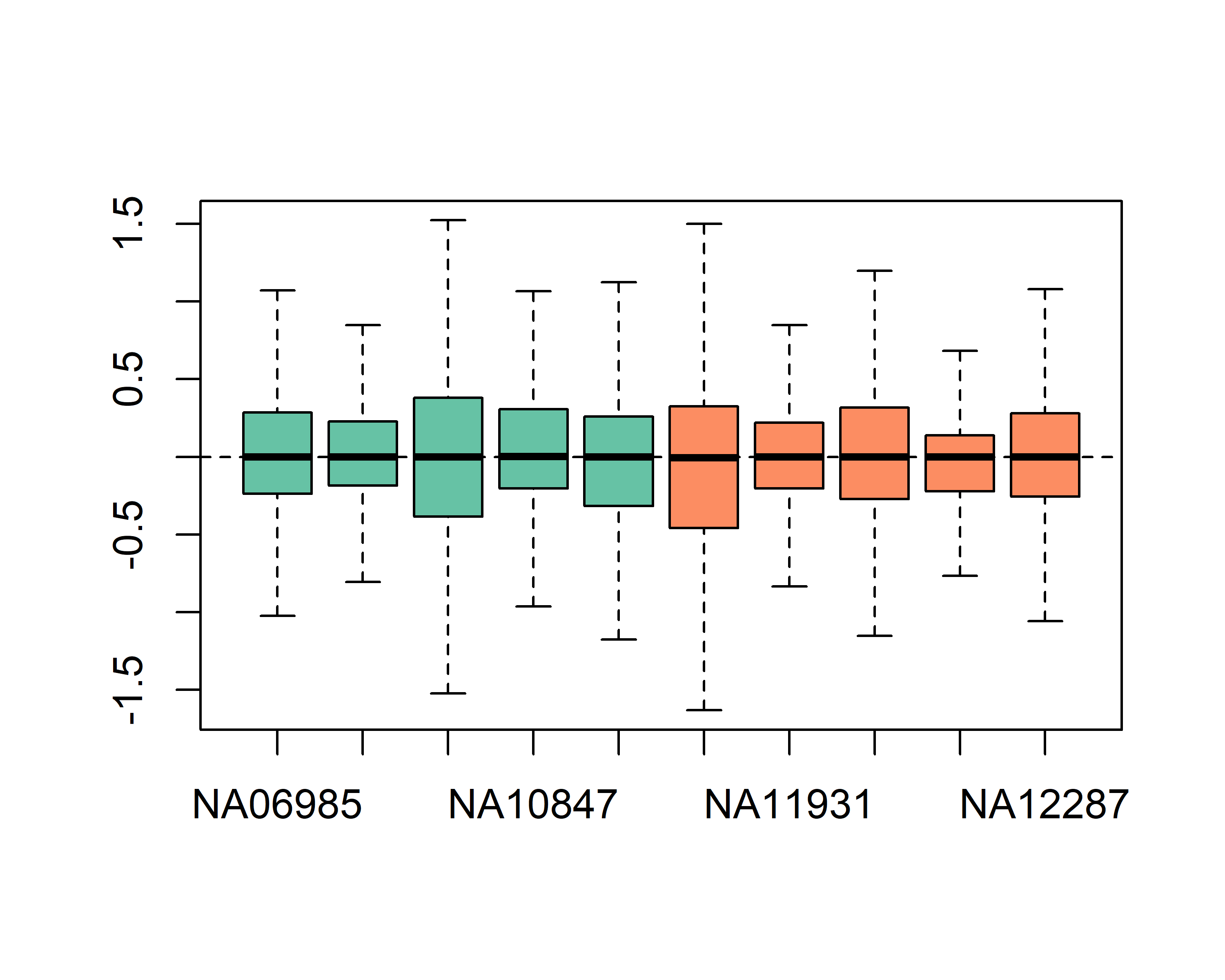
plotPCA(set,col=colors[group])



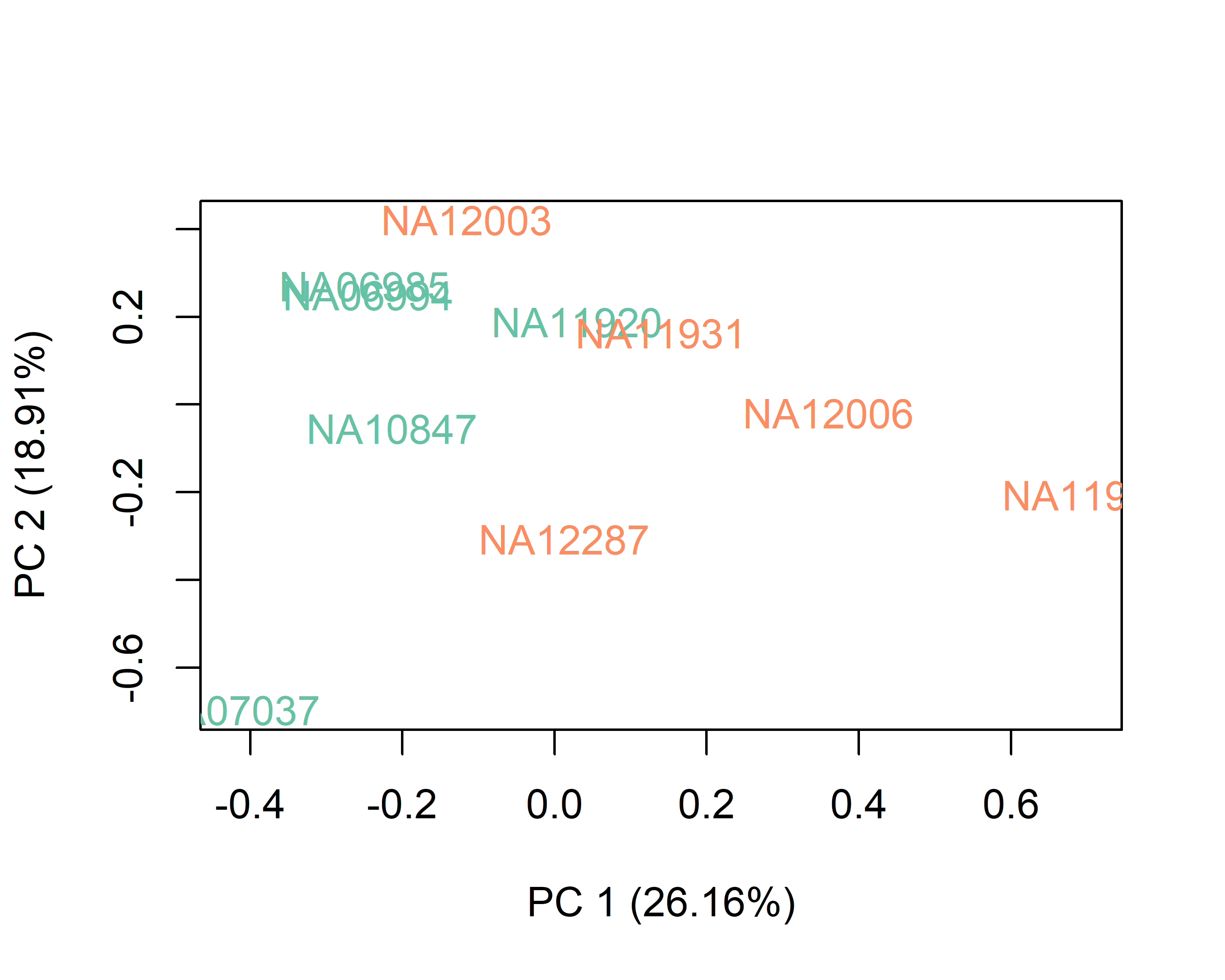
Two of the samples in group 2 (NA11918 and NA12006) appear to be very different from the rest of the samples. In the boxplot these samples have a much wider range (more variability) than the others. Also, in the PCA plot there is one cluster with samples from both groups, and the two outliers from group 2 are clearly different from everything else. The first PC appears to be driven by the difference between NA12006 and the other samples, whereas PC2 seems to be driven by NA11918. Between-lane normalization does not appear to be sufficient for these data.

## c) Perform RUVg using negative empirical control genes

# take the 10,000 genes with the largest likelihood ratio test   
# p-values from part a)  
neg\_controls =   
 rownames(head(glm\_lrt\_a[order(glm\_lrt\_a$PValue,  
 decreasing = T),],10000))  
set1 = RUVg(set,neg\_controls,k=1)  
plotRLE(set1, outline=FALSE,col=colors[group])



plotPCA(set1,col=colors[group])



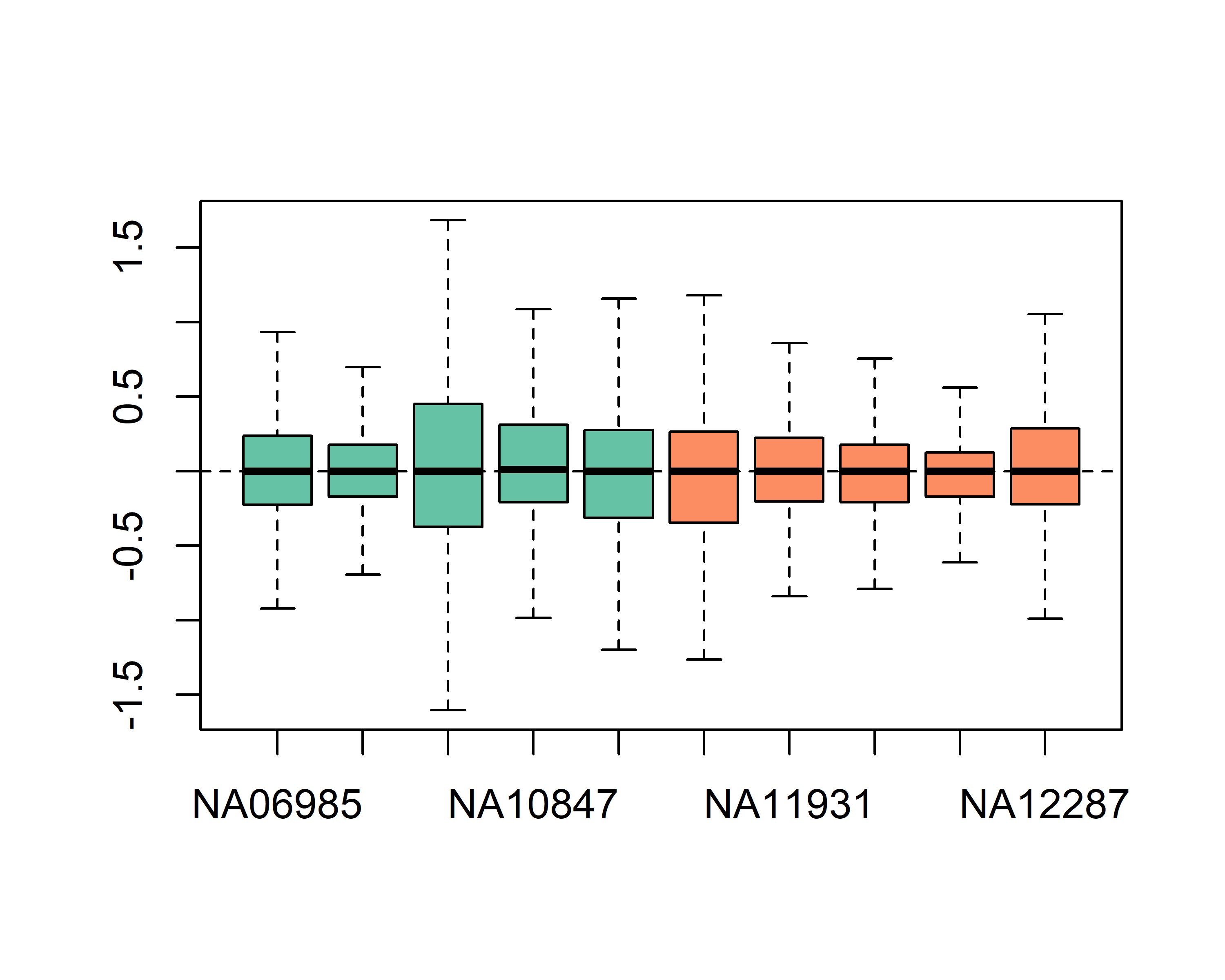
These plots are definitely an improvement on the previous ones. There still appear to be two samples with more variability than the others (NA07037 and NA11918), but in both the boxplot and PCA plot all of the samples are closer to one another than before RUVg.

design = model.matrix(~group + W\_1, data=pData(set1))  
y = DGEList(counts=counts(set1), group=group)  
y = calcNormFactors(y, method="upperquartile")  
y = estimateGLMCommonDisp(y, design)  
y = estimateGLMTagwiseDisp(y, design)  
glm\_fit\_c = glmFit(y, design)  
glm\_lrt\_c = glmLRT(glm\_fit\_c, coef=2)$table  
glm\_lrt\_c$PValue\_FDR = p.adjust(glm\_lrt\_c$PValue,"fdr")

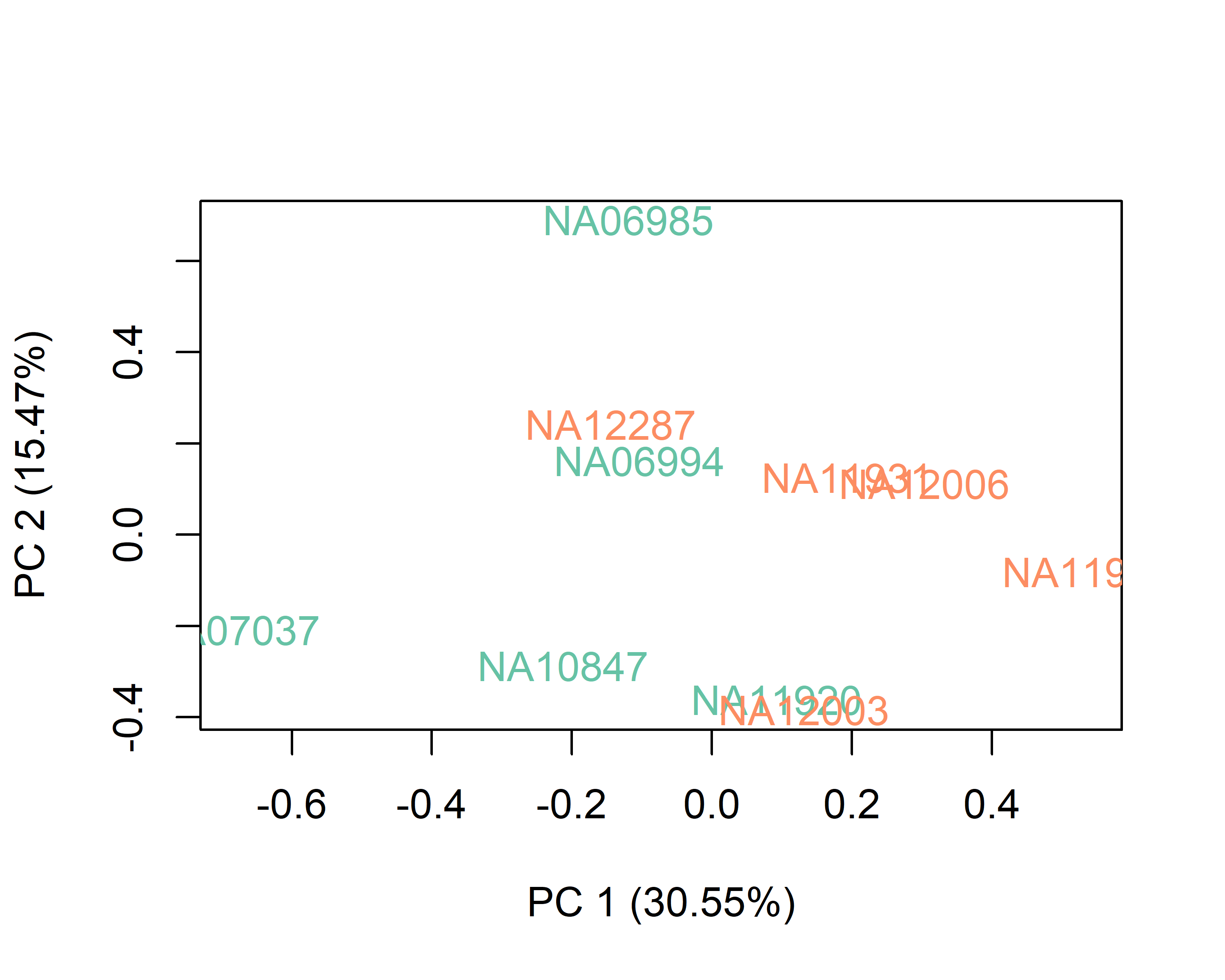
After controlling for unwanted variation, there are 157 genes with FDR adjusted p values < 0.05.

## d) Repeat part c) using k=2

set2 = RUVg(set,neg\_controls,k=2)  
plotRLE(set2, outline=FALSE,col=colors[group])



plotPCA(set2,col=colors[group])



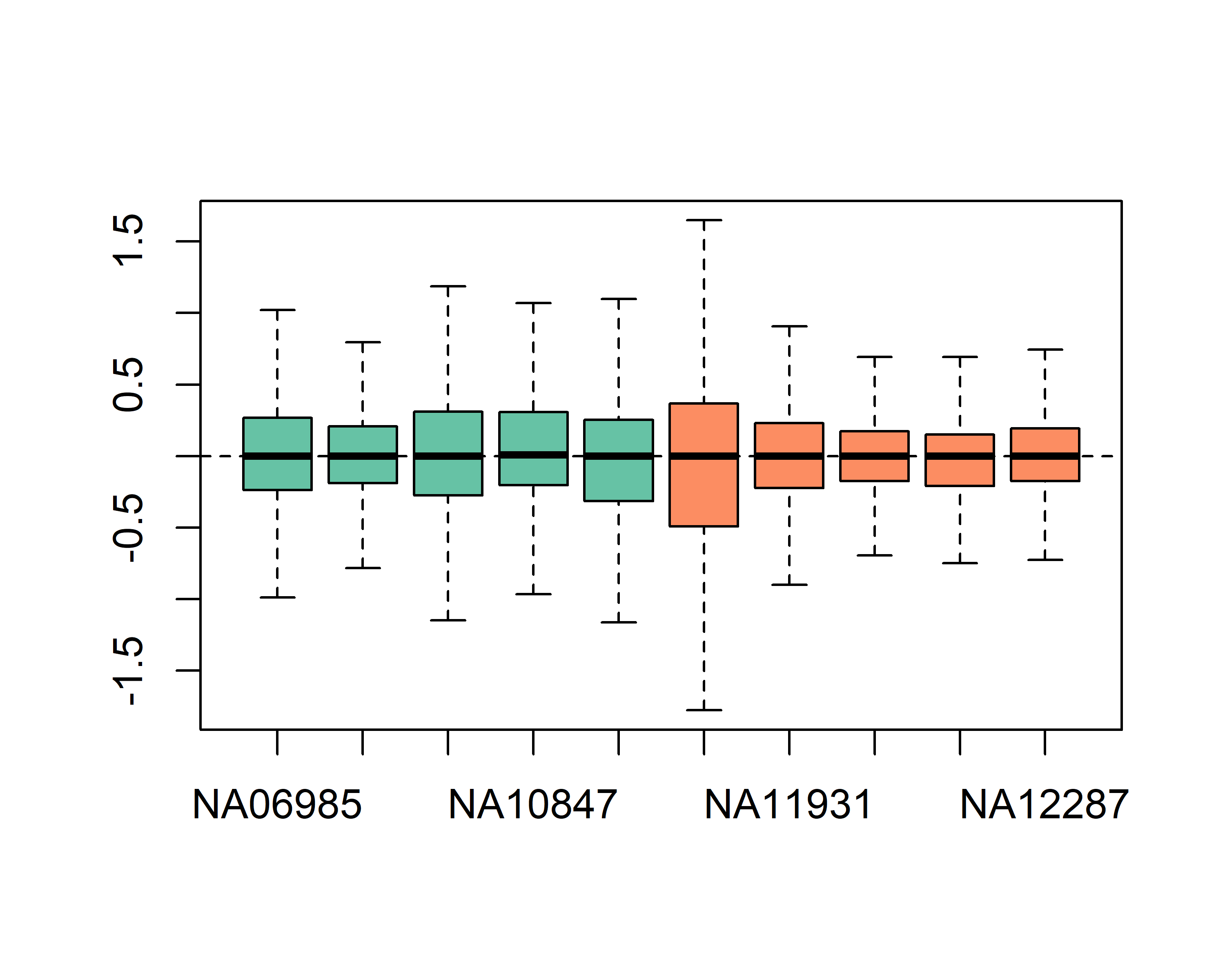
Increasing the number of factors of unwanted variation appears to improve these plots somewhat, although now sample NA07037 looks like a bit of an outlier (particularly in the boxplot). Overall though, the distribution of the samples in the boxplot appears to be more similar than in previous steps, and separation in the PCA plot doesn’t appear to be driven as much by single samples.

design = model.matrix(~group + W\_1 + W\_2, data=pData(set2))  
y = DGEList(counts=counts(set2), group=group)  
y = calcNormFactors(y, method="upperquartile")  
y = estimateGLMCommonDisp(y, design)  
y = estimateGLMTagwiseDisp(y, design)  
glm\_fit\_d = glmFit(y, design)  
glm\_lrt\_d = glmLRT(glm\_fit\_d, coef=2)$table  
glm\_lrt\_d$PValue\_FDR = p.adjust(glm\_lrt\_d$PValue,"fdr")

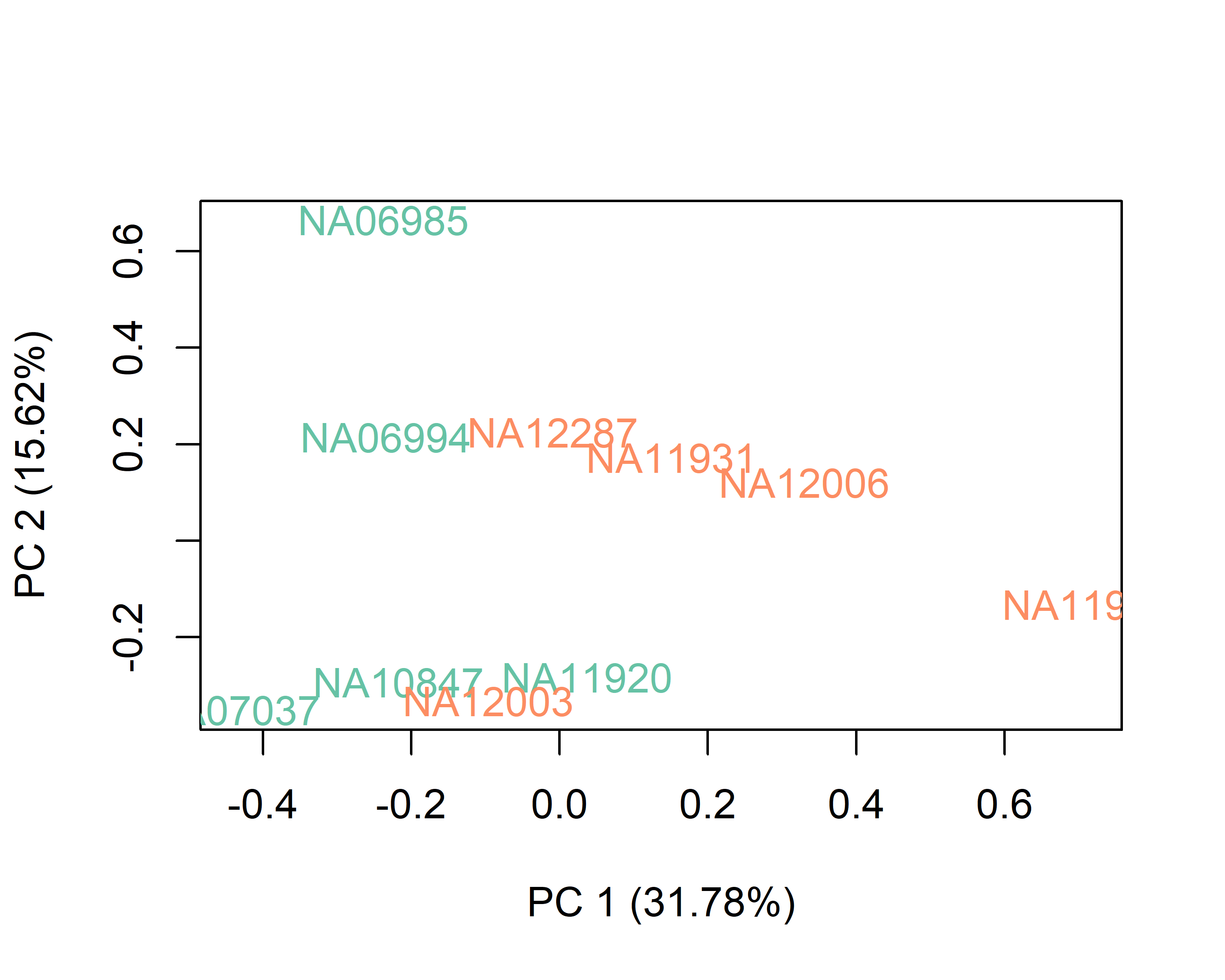
After controlling for two factors of unwanted variation, there are 230 genes with FDR adjusted p values < 0.05.

## e) Repeat part (d) using the RUVr method with k=2

# GLM residuals  
res = residuals(glm\_fit\_a, type="deviance")  
# RUVr  
set3 = RUVr(set,neg\_controls,k=2,res)  
plotRLE(set3,outline=FALSE,col=colors[group])



plotPCA(set3,col=colors[group])



Overall these plots look similar to using RUVg with k = 2, although in the boxplot it appears that NA11918 still has more variability than the other samples. NA11918 also seems to driving PC1, although this PCA plot is still a significant improvement over the original plot without any RUV methods applied.

design = model.matrix(~group + W\_1 + W\_2, data=pData(set3))  
y = DGEList(counts=counts(set3), group=group)  
y = calcNormFactors(y, method="upperquartile")  
y = estimateGLMCommonDisp(y, design)  
y = estimateGLMTagwiseDisp(y, design)  
glm\_fit\_e = glmFit(y, design)  
glm\_lrt\_e = glmLRT(glm\_fit\_e, coef=2)$table  
glm\_lrt\_e$PValue\_FDR = p.adjust(glm\_lrt\_e$PValue,"fdr")

After controlling for two factors of unwanted variation using RUVr, there are 274 genes with FDR adjusted p values < 0.05.

## f) Concerns

My major concern about RUV methods in general is that they remove wanted variation in addition to unwanted. For example,

# LOOK AT THE PAPER FOR LIMITATIONS

# GENERALLY WE WANT THE LOWEST POSSIBLE K THAT HELPS

# 3. Method Comparisons

Load package and data:

library(DESeq2)  
load(url("http://bowtie-bio.sourceforge.net/recount/ExpressionSets/  
bottomly\_eset.RData"))  
bottomly.count.table = exprs(bottomly.eset)

## a) Create a new data frame with genes that have at least 10 counts

# Filter  
filtered = bottomly.count.table[rowSums(bottomly.count.table)>=10,]  
# edgeR object  
filtered\_dge = DGEList(filtered)  
# DESeq2 object  
pheno = factor(gsub("/","\_",phenoData(bottomly.eset)$strain))  
filtered\_dseq = DESeqDataSetFromMatrix(filtered, DataFrame(pheno),~pheno)

There are 11870 genes with at least 10 counts across all samples.

# KEEP AN EYE OUT FOR AUTOMATIC P VALUE ADJUSTMENT

## b) Calculate the DESeq2 size factors

filtered\_dseq = estimateSizeFactors(filtered\_dseq)  
filtered\_dge = calcNormFactors(filtered\_dge,method = "TMM")  
# Compare  
sizes = data.frame(sizeFactors(filtered\_dseq))  
sizes = cbind(sizes,filtered\_dge$samples$norm.factors) %>%  
 rownames\_to\_column()   
colnames(sizes) = c("Sample","DESeq Size Factor","TMM Norm Factor")  
autofit(flextable(sizes))

| Sample | DESeq Size Factor | TMM Norm Factor |
| --- | --- | --- |
| SRX033480 | 0.6439291 | 0.9858019 |
| SRX033488 | 1.3453539 | 0.9789546 |
| SRX033481 | 0.5784839 | 1.0094511 |
| SRX033489 | 1.4295303 | 0.9965089 |
| SRX033482 | 0.6355123 | 0.9880781 |
| SRX033490 | 1.5239501 | 0.9840943 |
| SRX033483 | 0.7933382 | 0.9942930 |
| SRX033476 | 1.1271894 | 0.9971411 |
| SRX033478 | 1.0772279 | 1.0023364 |
| SRX033479 | 0.8984474 | 0.9963740 |
| SRX033472 | 0.8886335 | 1.0193662 |
| SRX033473 | 1.0255149 | 0.9924446 |
| SRX033474 | 0.7987292 | 0.9975642 |
| SRX033475 | 0.7795619 | 1.0124273 |
| SRX033491 | 1.6161933 | 1.0133338 |
| SRX033484 | 0.9881892 | 1.0370408 |
| SRX033492 | 1.5720164 | 1.0070477 |
| SRX033485 | 0.7557824 | 1.0135210 |
| SRX033493 | 1.5922159 | 0.9960492 |
| SRX033486 | 0.8264069 | 0.9831050 |
| SRX033494 | 1.4715139 | 0.9969959 |

Size factors as calculated by DESeq2 are the median of the ratios of each sample over a psuedosample (the same as the RLE method in edgeR). The pseudosample is the geometric mean for each gene across all samples. The formula for the size factor of sample (with indexing gene) is:

The idea is to make samples which may have been sequenced at different depths more comparable. For this dataset, the size factors estimated by TMM tend to be close to 1, whereas those estimated by DESeq2 have a much larger range and are more variable.

## c) Calculate the DESeq2 dispersions

Histograms:

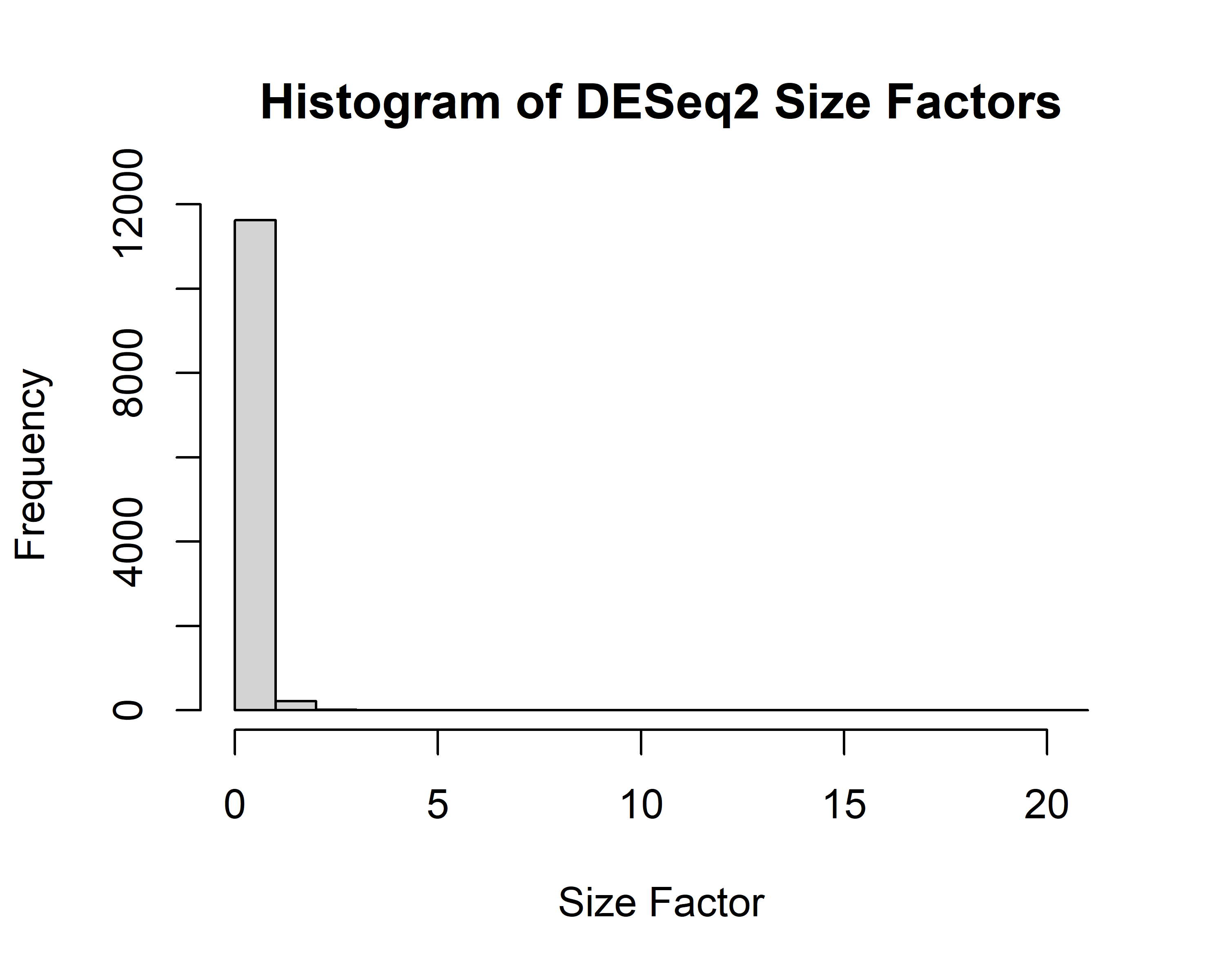
filtered\_dseq = estimateDispersions(filtered\_dseq,fitType = "local")

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

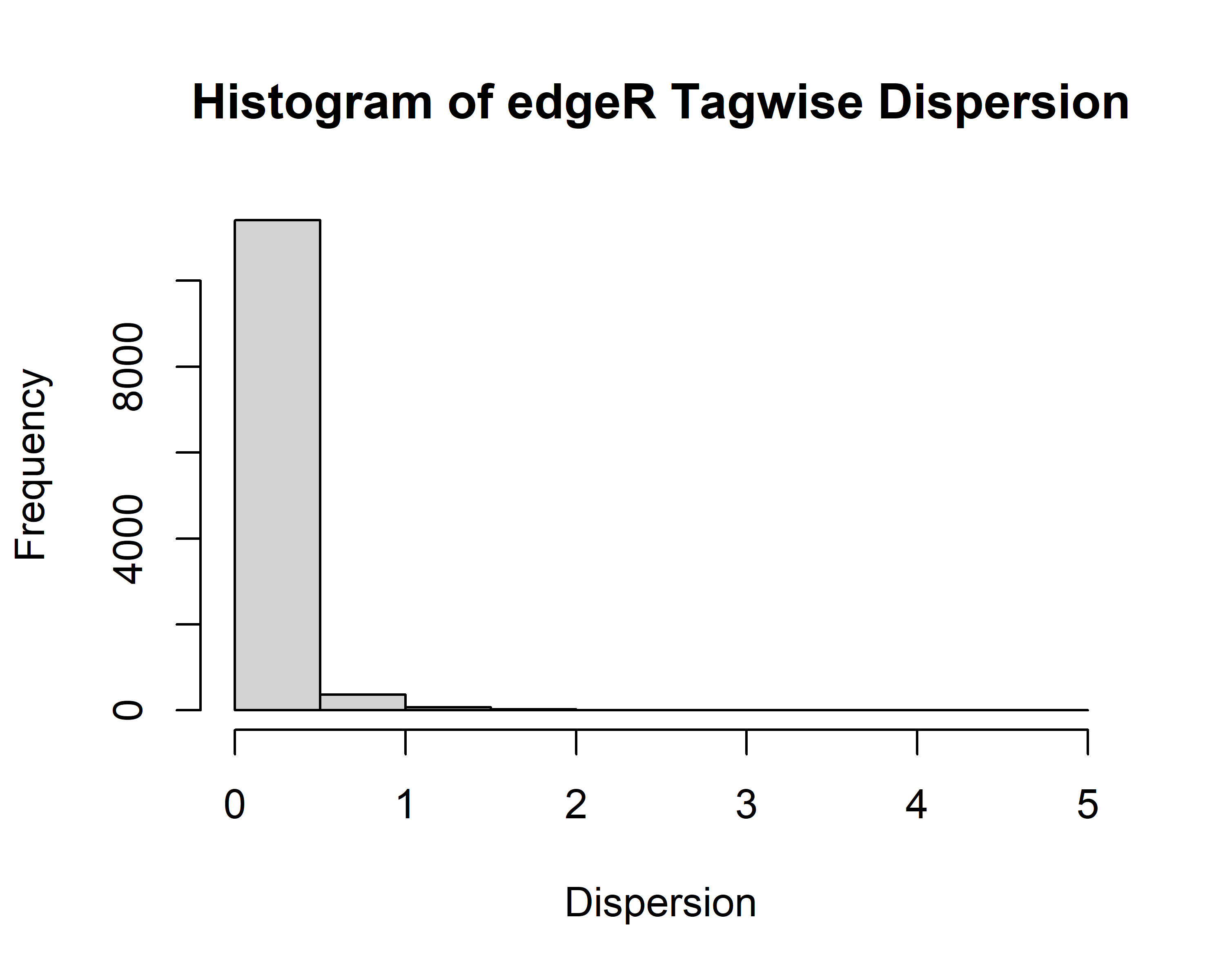
hist(dispersions(filtered\_dseq),xlab = "Size Factor",  
 main = "Histogram of DESeq2 Size Factors")



filtered\_dge = estimateDisp(filtered\_dge)

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

hist(filtered\_dge$tagwise.dispersion,xlab = "Dispersion",  
 main = "Histogram of edgeR Tagwise Dispersion")

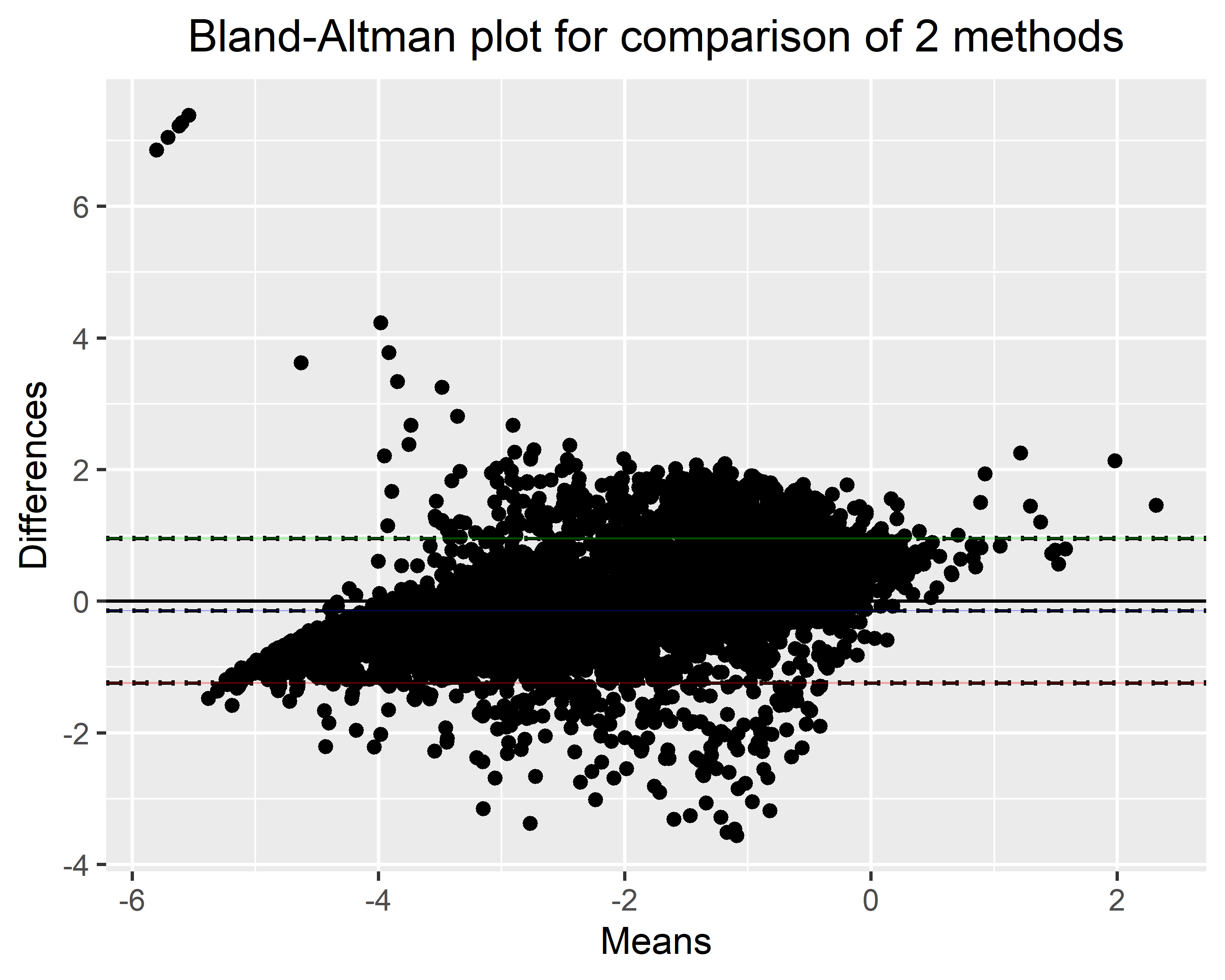


Bland-Altman plot:

blandr.draw(log(dispersions(filtered\_dseq)),  
 log(filtered\_dge$tagwise.dispersion))

## Warning: Use of `plot.data$x.axis` is discouraged. Use `x.axis` instead.

## Warning: Use of `plot.data$y.axis` is discouraged. Use `y.axis` instead.



## d) Test for differences between the two strains

filtered\_dseq = nbinomWaldTest(filtered\_dseq)  
res = results(filtered\_dseq)  
design = model.matrix(~pheno)  
glm\_fit = glmFit(filtered\_dge,design)  
glm\_lrt = glmLRT(glm\_fit)$table  
glm\_lrt$PValue\_BH = p.adjust(glm\_lrt$PValue,"BH")

The DESeq2 method finds 1404 genes with p values < 0.05 after Benjamini-Hochberg (BH) correction, and edgeR finds 696.