Homework 6 - BIOS 7649

Dominic Adducci

Question 1: RNA-Seq: Differential Expression

```
##  [1] "C57BL_6J" "C57BL_6J" "C57BL_6J" "C57BL_6J" "C57BL_6J" "C57BL_6J"
##  [7] "C57BL_6J" "C57BL_6J" "C57BL_6J" "C57BL_6J" "DBA_2J"   "DBA_2J"
## [13] "DBA_2J"   "DBA_2J"   "DBA_2J"   "DBA_2J"   "DBA_2J"   "DBA_2J"
## [19] "DBA_2J"   "DBA_2J"   "DBA_2J"
```

```
##  [1] C57BL/6J C57BL/6J C57BL/6J C57BL/6J C57BL/6J C57BL/6J C57BL/6J C57BL/6J
##  [9] C57BL/6J C57BL/6J DBA/2J   DBA/2J   DBA/2J   DBA/2J   DBA/2J   DBA/2J
## [17] DBA/2J   DBA/2J   DBA/2J   DBA/2J   DBA/2J
## Levels: C57BL/6J DBA/2J
```

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

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

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

Part A

Create a new data frame with genes that have at least 10 counts (summed across samples). How many genes are kept? Create the data object for DESeq2 (use **DESeqDataSetFromMatrix()**) and the data object for edgeR (use **DGEList()**).

After creating a new data frame with genes that have at least 10 counts there are 11870 genes remaining.

Part B

Calculate the DESeq2 size factors (use **estimateSizeFactors()** and **sizeFactors()**). Calculate the edgeR size factors using the "TMM" method (use **calcNormFactors()**). What are size factors? How do the two sets of size factors compare?

Part C

Test for differences between the two strains using DESeq2 (use **nbinomWaldTest()**) and edgeR (use **glmFit()** and **glmLRT()**). Note that the two methods do not return the same amount of details for the results. Using adjusted p-values with the Benjamini-Hochberg method (Note: check what the functions provide or

if you need to do this yourself), how many genes are found in each method to be differentially expressed? What is the overlap between the methods? Check the results for one example gene that is significant in one method but not the other. Compare the methods based on the estimate of the fold change and p-value for the example. What do you conclude about the differences between the two methods?

The adjusted p-value from the DESeq2 method is calculated using the Benjamini-Hochberg method.

Question 2: RNA-Seq: Remove Unwanted Variation

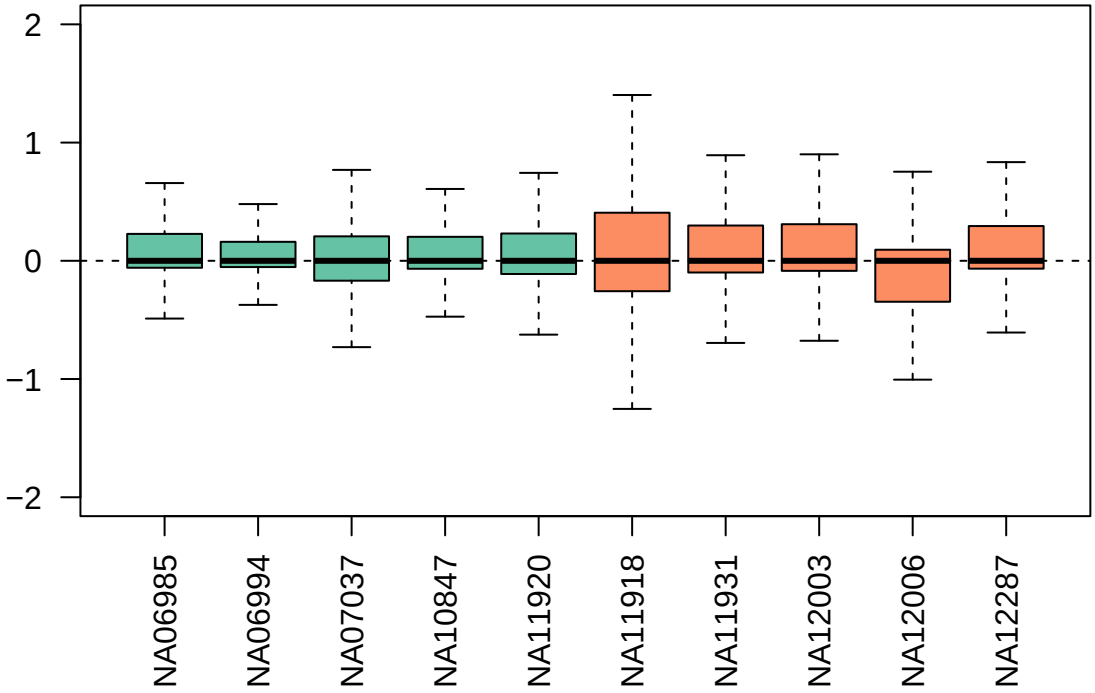
Part A

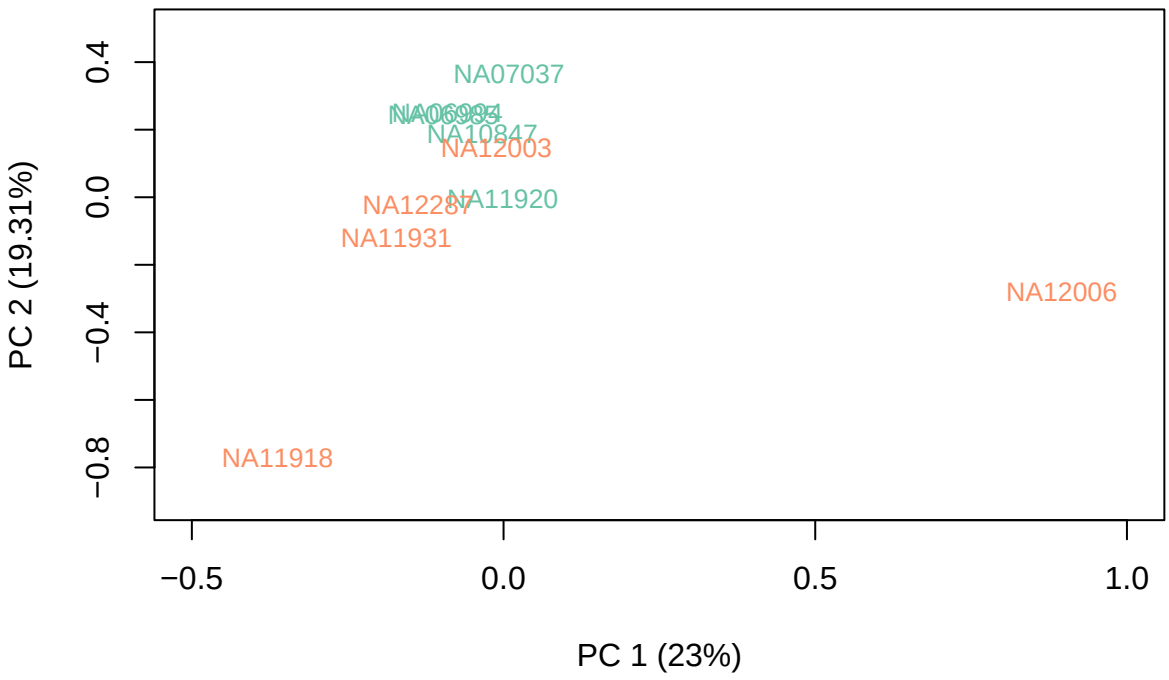
Use the Montgomery data set from HW5. For this problem, the two groups are the first five subjects and the second group are the second five subjects. Test for differential expression between the groups using a standard likelihood ratio test (**glmFit()** and **glmLRT()**). See Section 2.11.3 of the **edgeR** user's manual. How many genes have FDR adjusted p-values < 0.05?

After filtering the genes for those with an FDR corrected p-value <0.05 there are 6 genes which are differentially expressed between the two groups.

Part B

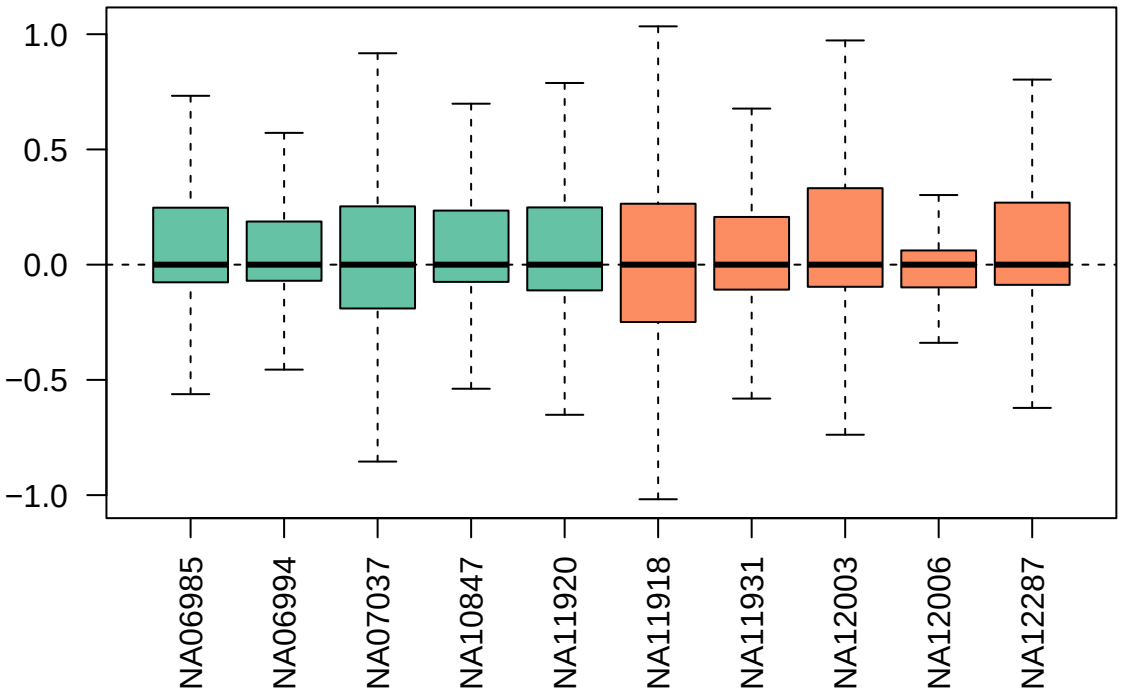
We will use **RUVSeq** to fit **edgeR** models that adjust for unwanted variation. Following section 2.1 of the **RUVSeq** documentation, create an expression set and perform upper-quantile normalization using the **betweenLaneNormalization** function. Plot the relative log expression of the upper-quantile normalized counts and create a PCA plot colored by group. Are there any issues you note in these plots that may benefit from additional normalization?

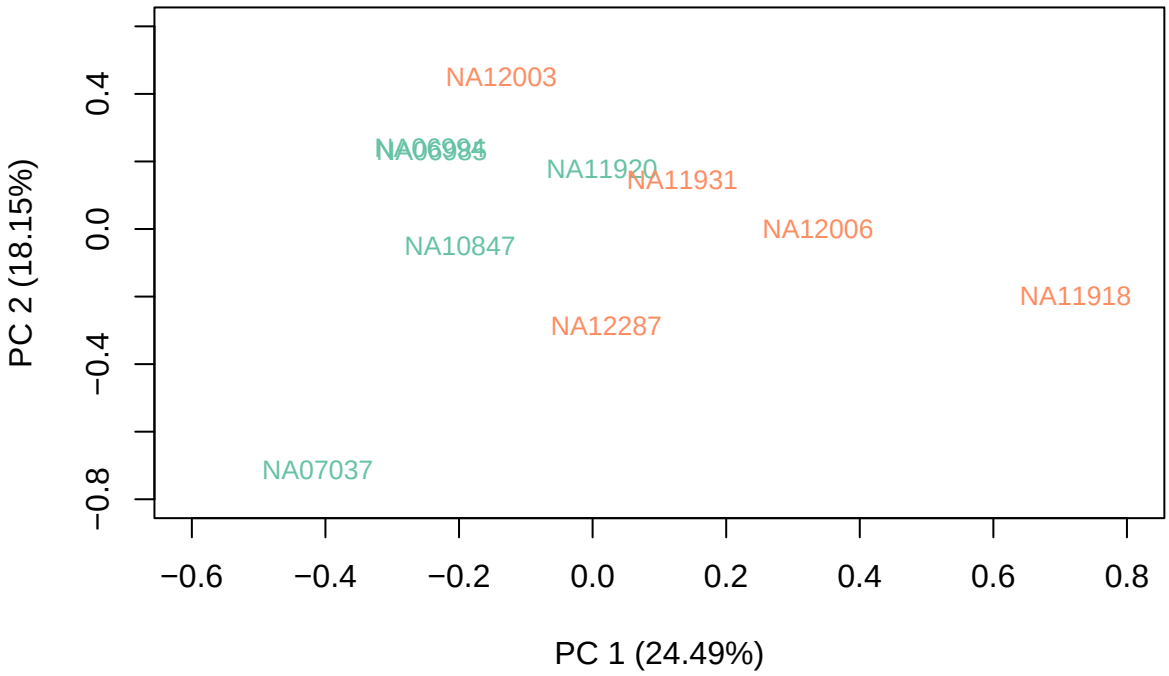




Part C

Perform RUVg using negative empirical control genes. To define a set of control genes, take the 10,000 genes with the largest likelihood ratio test p-values from part (a). Follow the steps in Section 2.2 and 2.3 of the **RUVSeq** manual, using upper-quartile normalization and k=1. Create boxplots of RLE and a PCA of the normalized counts. Comment on these plots. How many genes had FDR adjusted p-values < 0.05 after controlling for 1 factor of unwanted variation.

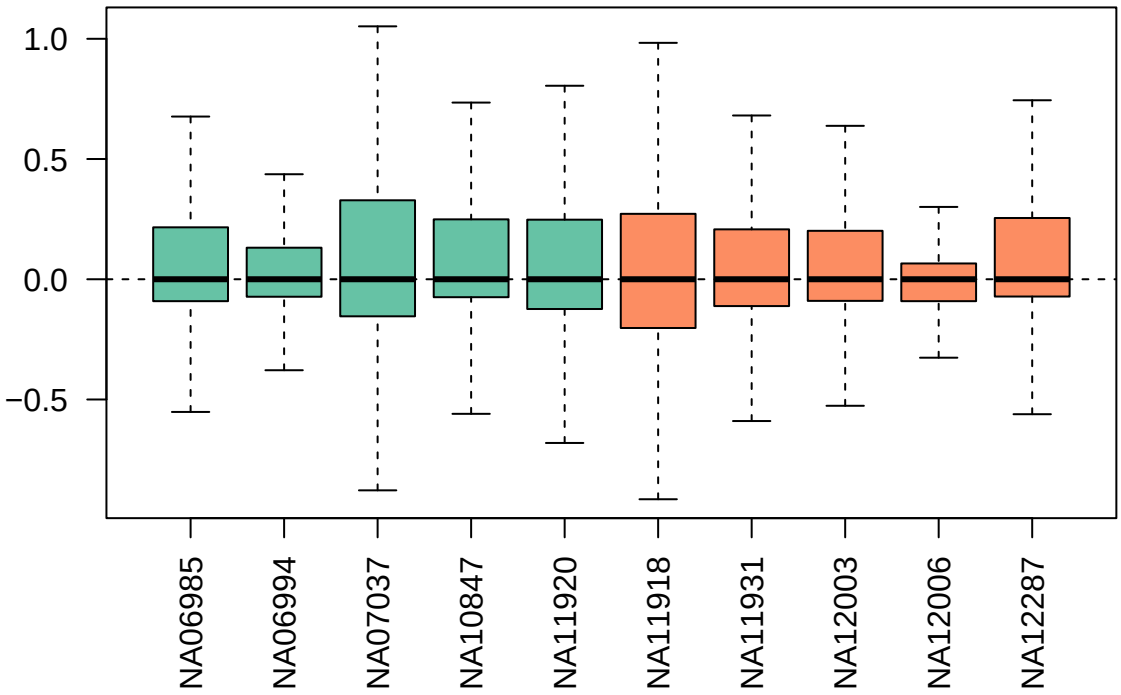

1.0
0.5
0.0
−0.5
−1.0
NA06985
NA06994
NA07037
NA10847
NA11920
NA11918
NA11931
NA12003
NA12006
NA12287

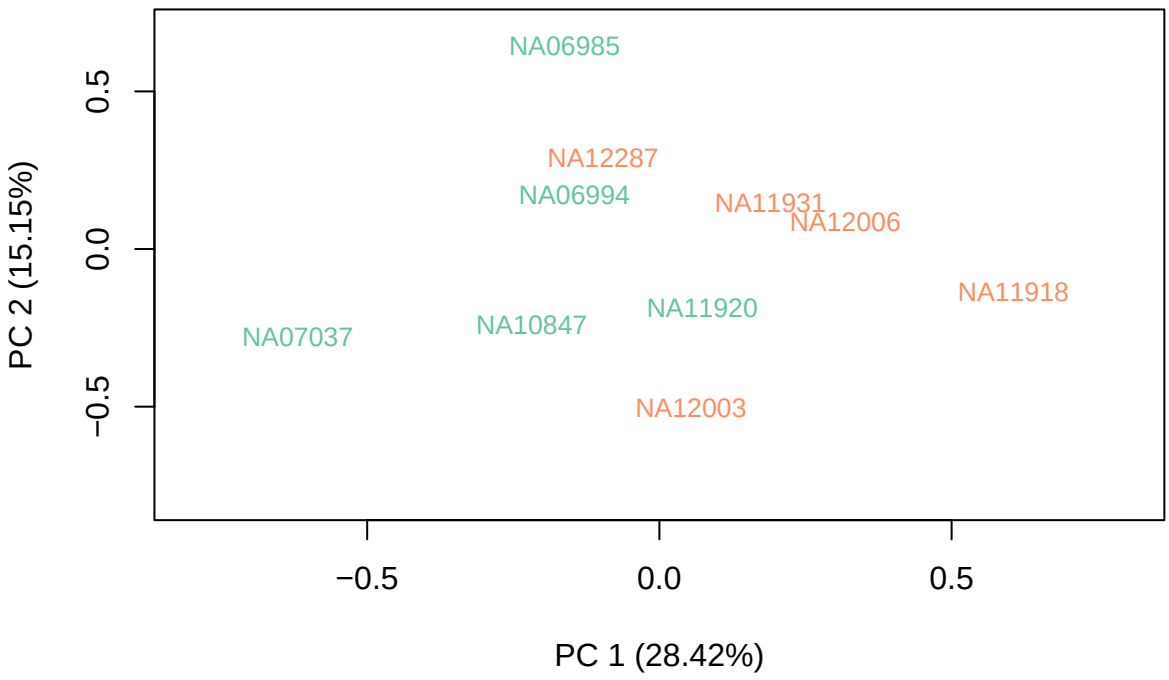


After controlling for 1 factor of unwanted variation there are 1322 genes with an FDR corrected p-value $<$ 0.05.

Part D

Repeat part(c) above using k=2. How many genes had FDR adjusted p-values $<$0.05 after controlling for 2 factors of unwanted variation?

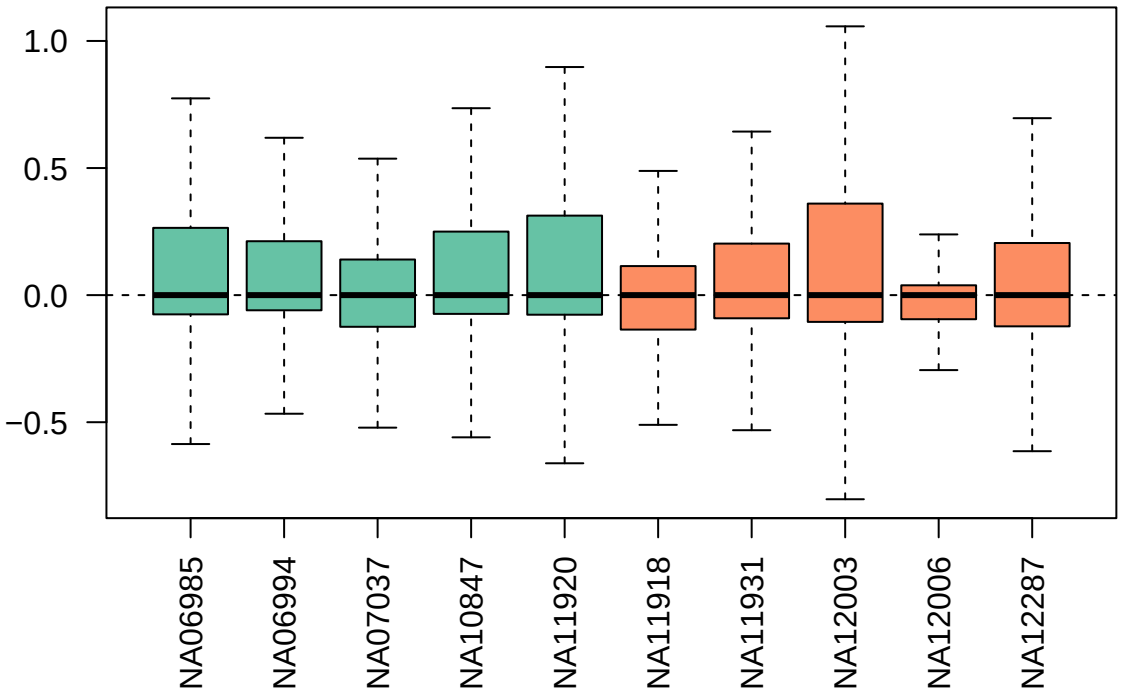

1.0
0.5
0.0
−0.5
NA06985
NA06994
NA07037
NA10847
NA11920
NA11918
NA11931
NA12003
NA12006
NA12287

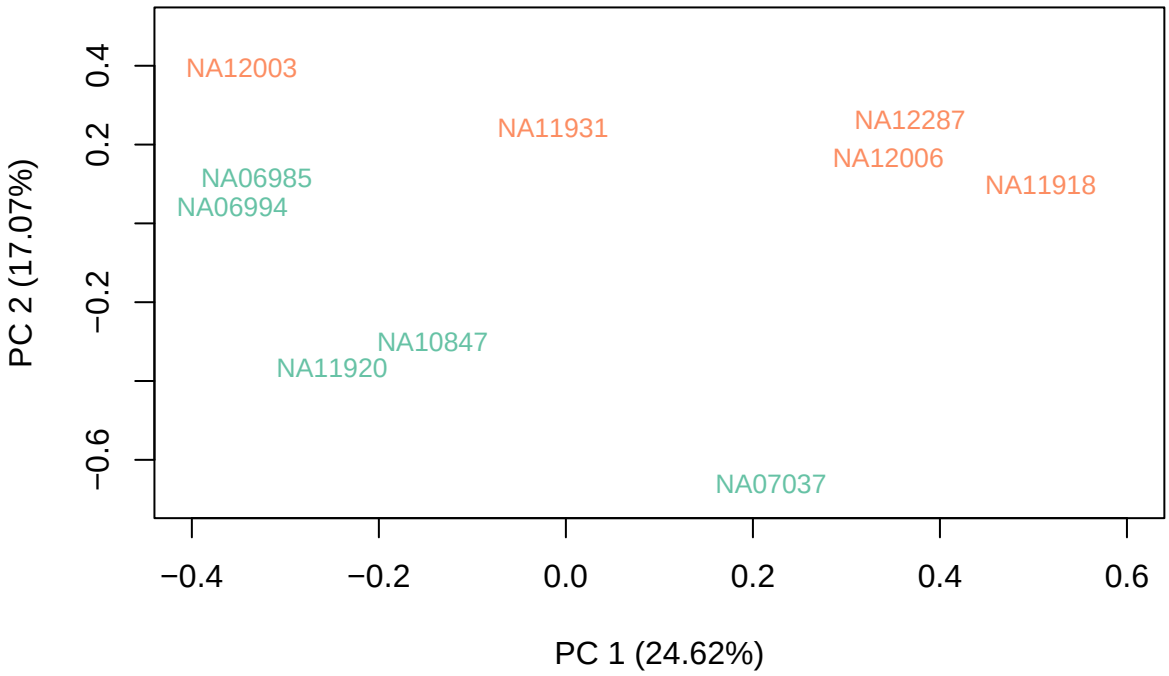


After controlling for 1 factor of unwanted variation there are 279 genes with an FDR corrected p-value $<$ 0.05.

PART E

Repeat part(d) using the RUVr method with k=2, following Section 4 of the **RUVSeq** manual. How many genes had FDR adjusted p-values $<$ 0.05 after controlling for 2 factors of unwatned variation using RUVr?


1.0
0.5
0.0
−0.5
NA06985
NA06994
NA07037
NA10847
NA11920
NA11918
NA11931
NA12003
NA12006
NA12287



After applying the RUVr method there are 279 genes with an FDR adjusted p-value < 0.05.

CODE

```
library(DESeq2)
library(edgeR)
library(Biobase)
library(RUVSeq)
library(cqn)
library(tidyverse)
library(RColorBrewer)


### START QUESTION 1 CODE ###

# Loading in data
load(url("https://bowtie-bio.sourceforge.net/recount/ExpressionSets/bottomly_eset.RData"))

# Formatting data for R
gsub("/","_",phenoData(bottomly.eset)$strain)

phenoData(bottomly.eset)@data[,'strain']

featureNames(bottomly.eset)[1:10]
bottomly.count.table <- exprs(bottomly.eset)
dim(bottomly.count.table)
head(row.names(bottomly.count.table))


## START QUESTION 1 PART A CODE ##

# Filtering data so that only genes with at 10 counts are kept.
bottomly_filter <- as.data.frame(bottomly.count.table) %>%
  mutate(row_sums = rowSums(across(where(is.numeric)))) %>%
  filter(row_sums >= 10) %>%
  select(-row_sums)

# Checking how many genes (rows) were kept.
filter_gene_count <- nrow(bottomly_filter)

# Making objects for DESeq2 and edgeR

# DESeq2:
# Extracting strain information.
bottomly_strain <- gsub("/","_",phenoData(bottomly.eset)$strain)

colnames(bottomly_filter) <- bottomly_strain

DESeq2_obj <- DESeqDataSetFromMatrix(as.matrix(bottomly_filter),
                                     data.frame(bottomly_strain),
                                     ~ bottomly_strain)

# edgeR:
edgeR_obj <- DGEList(counts = bottomly_filter,group = factor(bottomly_strain))
```

```
## FINISH QUESTION 1 PART A CODE ##


## START QUESTION 1 PART B CODE ##

# Calculating size factors using estimateSizeFactors from DESeq2.
# Default values were used, with the exception of quieting output.
# Results are extracted using the sizeFactors function.
DESeq2_sf_obj <- estimateSizeFactors(DESeq2_obj, quiet = TRUE)

DESeq2_sf <- sizeFactors(DESeq2_sf_obj)

# Calculating size factors using the normLibSizes function (the function
# formerly known as calcNormFactors) from edgeR. "TMM" method selected.
edgeR_sf <- normLibSizes(edgeR_obj,method = "TMM")


## START QUESTION 1 PART C CODE ##

# Testing using DESeq2. First need to estimate dispersion, and then use the
# Negative Binomial Wald Test. Results are extracted using the 'results()'
# function and converted to a data frame.
DESeq2_disp <- estimateDispersions(DESeq2_sf_obj)
DESeq2_diff_test <- nbinomWaldTest(DESeq2_disp, quiet = TRUE)
DESeq2_results <- data.frame(results(DESeq2_diff_test))

# Order results by adjusted p-value, and selecting adjusted p-values that
# are less than 0.01.
DESeq2_results_order <- DESeq2_results[order(DESeq2_results$padj),]
DESeq2_BH_corr <- DESeq2_results_order %>% filter(padj < 0.01)

# Testing using edgeR. First need to make the design matrix and estimate the
# dispersion. Fit object will be iiput into the LRT function.
edgeR_design <- model.matrix(~factor(bottomly_strain))
edgeR_disp <- estimateDisp(edgeR_obj,edgeR_design)
edgeR_glmFit <- glmFit(edgeR_disp,edgeR_design)
edgeR_glmLRT <- glmLRT(edgeR_glmFit)

# Extracting table of results.
edgeR_results <- edgeR_glmLRT$table

# Applying Benjamini-Hochberg correction. Will need to order the results by
# p-values, add a column for ranks, and add a column for the critical values.
edgeR_results_order <- edgeR_results[order(edgeR_results$PValue),]
edgeR_results_order$ranks <- seq.int(nrow(edgeR_results_order))
edgeR_results_order$critical_value <- (edgeR_results_order$rank/
                                       nrow(edgeR_results_order)) * 0.01

# Selecting out significant genes. All p-values above the highest p-value
# that is smaller than the critical value are also smaller than their
# corresponding critical values.
edgeR_BH_corr <- edgeR_results_order %>% filter(PValue < critical_value)
```

```
### START QUESTION 2 CODE ###

# Loading in Montgomery data set. Assuming that this data set is already
# filtered (no rows with 0 gene counts).
data("montgomery.subset")


### START QUESTION 2 PART A CODE ##

# Making group indicators for group 1 and group 2 from the
# Montgomery subset and making a design matrix.
mgd_group <- as.factor(rep(c("group1","group2"),each = 5))

mgd_design <- model.matrix(~ 1 + mgd_group, montgomery.subset)

# Making a DGE list for the Montgomery data, will normalize using
# the calcNormFactors function, and will estimate the dispersion using
# the estimateDisp function.
mgd_dgel <- DGEList(counts = montgomery.subset,
                    group = mgd_group)
mgd_dgel <- calcNormFactors(mgd_dgel,method = "upperquartile")
mgd_dgel <- estimateDisp(mgd_dgel,design = mgd_design)

# Performing LRT to detect differential expression.
mgd_glmFit <- glmFit(mgd_dgel,mgd_design)
mgd_glmLRT <- glmLRT(mgd_glmFit)

# Extracting results, and selecting FDR corrected p-values below 0.05.
mgd_results <- data.frame(topTags(mgd_glmLRT, n = Inf)) %>%
  filter(FDR < 0.05)

mgd_results_len <- nrow(mgd_results)

## FINISH QUESTION 2 PART A CODE ##


## START QUESTION 2 PART B CODE ##

# Creating an RUVSeq expression seq.
mgd_SeqSet <- newSeqExpressionSet(counts = as.matrix(montgomery.subset),
                                  PhenoData =
                                    data.frame(mgd_group,
                                               row.names =
                                               colnames(montgomery.subset)))

# Normalizing data.
mgd_SeqSet <- betweenLaneNormalization(mgd_SeqSet, which = "upper")

# Setting up colors for plots.
colors <- brewer.pal(3,"Set2")

# Plotting relative log expression.
```

```r
plotRLE(mgd_SeqSet, outline = FALSE, ylim = c(-2,2),
        col = colors[mgd_group],cex = 0.80,las = 2)


# Plotting PCA plot.
plotPCA(mgd_SeqSet, col = colors[mgd_group], cex = 0.80,
        ylim = c(-0.9,0.5), xlim = c(-0.5,1))


## START QUESTION 2 PART C CODE ##

# Selecting control genes using 10,000 genes with the highest p-values.
mgd_control <- data.frame(topTags(mgd_glmLRT, n = Inf)) %>%
  arrange(desc(PValue)) %>%
  head(n = 10000) %>%
  rownames()

# Performing RUVg
mgd_RUVg <- RUVg(mgd_SeqSet,mgd_control,k = 1)

# Plotting relative log expression.
plotRLE(mgd_RUVg,outline = FALSE, col = colors[mgd_group],
        cex = 0.80, las = 2)


plotPCA(mgd_RUVg,col = colors[mgd_group],cex = 0.80,
        ylim = c(-0.8,0.6),xlim = c(-0.6,.8))


# Making a design matrix for the RUVg data.
mgd_RUVg_design <- model.matrix(~ mgd_group + W_1, data = pData(mgd_RUVg))

# Making a DGE list for the RUVg data.
mgd_RUVg_dgel <- DGEList(counts = counts(mgd_RUVg), group = mgd_group)
mgd_RUVg_dgel <- calcNormFactors(mgd_RUVg_dgel,method = "upperquartile")
mgd_RUVg_dgel <- estimateDisp(mgd_RUVg_dgel, design = mgd_RUVg_design)

# Performing LRT to detect differential expression.
mgd_RUVg_fit <- glmFit(mgd_RUVg_dgel,mgd_RUVg_design)
mgd_RUVg_LRT <- glmLRT(mgd_RUVg_fit)

mgd_RUVg_results <- data.frame(topTags(mgd_RUVg_LRT,n = Inf)) %>%
  filter(FDR < 0.05)

## FINISH QUESTION 2 PART C CODE ##


## FINISH QUESTION 2 PART D CODE ##

# Performing RUVg
mgd_RUVg2 <- RUVg(mgd_SeqSet,mgd_control,k = 2)

# Plotting relative log expression.
```

```r
plotRLE(mgd_RUVg2,outline = FALSE, col = colors[mgd_group],
        cex = 0.80, las = 2)



plotPCA(mgd_RUVg2,col = colors[mgd_group],cex = 0.80,
        ylim = c(-0.8,0.7),xlim = c(-0.8,.8))


# Making a design matrix for the RUVg data.
mgd_RUVg2_design <- model.matrix(~ mgd_group + W_1 + W_2 ,
                                 data = pData(mgd_RUVg2))

# Making a DGE list for the RUVg data.
mgd_RUVg2_dgel <- DGEList(counts = counts(mgd_RUVg2), group = mgd_group)
mgd_RUVg2_dgel <- calcNormFactors(mgd_RUVg2_dgel,method = "upperquartile")
mgd_RUVg2_dgel <- estimateDisp(mgd_RUVg2_dgel, design = mgd_RUVg2_design)

# Performing LRT to detect differential expression.
mgd_RUVg2_fit <- glmFit(mgd_RUVg2_dgel,mgd_RUVg2_design)
mgd_RUVg2_LRT <- glmLRT(mgd_RUVg2_fit)

mgd_RUVg2_results <- data.frame(topTags(mgd_RUVg2_LRT,n = Inf)) %>%
  filter(FDR < 0.05)

## FINISH QUESTION 2 PART D CODE ##


## START QUESTION 2 PART E CODE ##

# Setting up design matrix for RUVr method.
mgd_RUVr_design <- model.matrix(~ mgd_group, data = pData(mgd_SeqSet))

# Setting up DGE list for RUVr method.
mgd_RUVr_dgel <- DGEList(counts=counts(mgd_SeqSet),group = mgd_group)
mgd_RUVr_dgel <- calcNormFactors(mgd_RUVr_dgel,method = "upperquartile")
mgd_RUVr_dgel <- estimateDisp(mgd_RUVr_dgel,mgd_RUVr_design)
mgd_RUVr_fit <- glmFit(mgd_RUVr_dgel)
mgd_RUVr_res <- residuals(mgd_RUVr_fit,type = "deviance")

# Applying RUVr method.
mgd_RUVr <- RUVr(mgd_SeqSet,rownames(counts(mgd_SeqSet)),
                 k = 2, residuals = mgd_RUVr_res)

# Plotting relative log expression.
plotRLE(mgd_RUVr,outline = FALSE, col = colors[mgd_group],
        cex = 0.8, las = 2)


# Making PCA plot.
plotPCA(mgd_RUVr,col = colors[mgd_group],cex = 0.80,
        ylim = c(-0.7,0.5), xlim = c(-0.4,0.6))
```

```r
mgd_RUVr2_design <- model.matrix(~ mgd_group + W_1 + W_2,
                                 data = pData(mgd_RUVr))

# Making a second DGE list.
mgd_RUVr2_dgel <- DGEList(counts = counts(mgd_RUVr),
                          group = mgd_group)
mgd_RUVr2_dgel <- calcNormFactors(mgd_RUVr2_dgel,method = "upperquartile")
mgd_RUVr2_dgel <- estimateDisp(mgd_RUVr2_dgel,mgd_RUVr2_design)

mgd_RUVr2_glmFit <- glmFit(mgd_RUVr2_dgel,design = mgd_RUVr2_design)
mgd_RUVr2_glmLRT <- glmLRT(mgd_RUVr2_glmFit)

# Extracting results.
mgd_RUVr2_results <- data.frame(topTags(mgd_RUVg2_LRT,n = Inf)) %>%
  filter(FDR < 0.05)
```