Homework 6 - BIOS 7649

Dominic Adducci

All analysis were performed in R version 4.3.1

Question 1: RNA-Seq: Differential Expression

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()**).

The Bottomly data set was filtered so that only genes with at least 10 counts summed across all samples were kept. Objects for DESeq2 and edgeR were created for further analysis. The strain names needed to be adjusted, as the "/" symbol causes problems in R. 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?

Size factors take into account how many reads are in each library. This allows for expression levels to be calculated relative to the number of sequences read for each sample. In DESeq2 the estimateSizeFactor function uses the default option 'ratio', which uses the standard median ratio method. From the help file the size factor is the median ratio o the sample over a pseudosample, where the pseudosample for each gene is the geometric mean of all samples. From the edgeR package the normLibSizes function calculates size factors using the default method of 'TMM'. This method works by using a trimmed mean of M-values between each pair of samples. The product of the original library size and the scaling factor is called the effective library size (or the normalized library size), and replaces the original library size for downstream analysis. Due to the different methods for calculation the size factors between DESeq2 and edgeR are noticeably different. For example, the first C57BL_6J sample has a size factor from DESeq2 of 0.6439291, while in edgeR this same sample has a size factor of 0.9858019. Additionally, the size factors from DESeq2 range from around 0.64 to around 1.59, while in edgeR the size factors range between around 0.97 to around 1.03.

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, while for edgeR the Benjamini-Hochberg adjustedment needed to be calculated manually. From DESeq2 there are 784 genes which are significant, while from edgeR there are 793 genes which are significant. Between both methods there are 703 genes that are the same.

Table 1: Example Gene ENSMUSG00000025217

	Fold Change	BH p-value
DESeq2	-0.267	5.035e-06
edgeR	-0.240	0.0006816

Table 1 shows an example of a gene that is significant DESeq2 and not edgeR. For this gene the fold change is similar, but the adjusted p-value is significantly different. Because the Benjamini-Hochberg method is based on p-value ranks it seems that slight differences between the two methods can adjust the rank of genes even if the log change values are similar.

Question 2: RNA-Seq: Remove Unwanted Variation

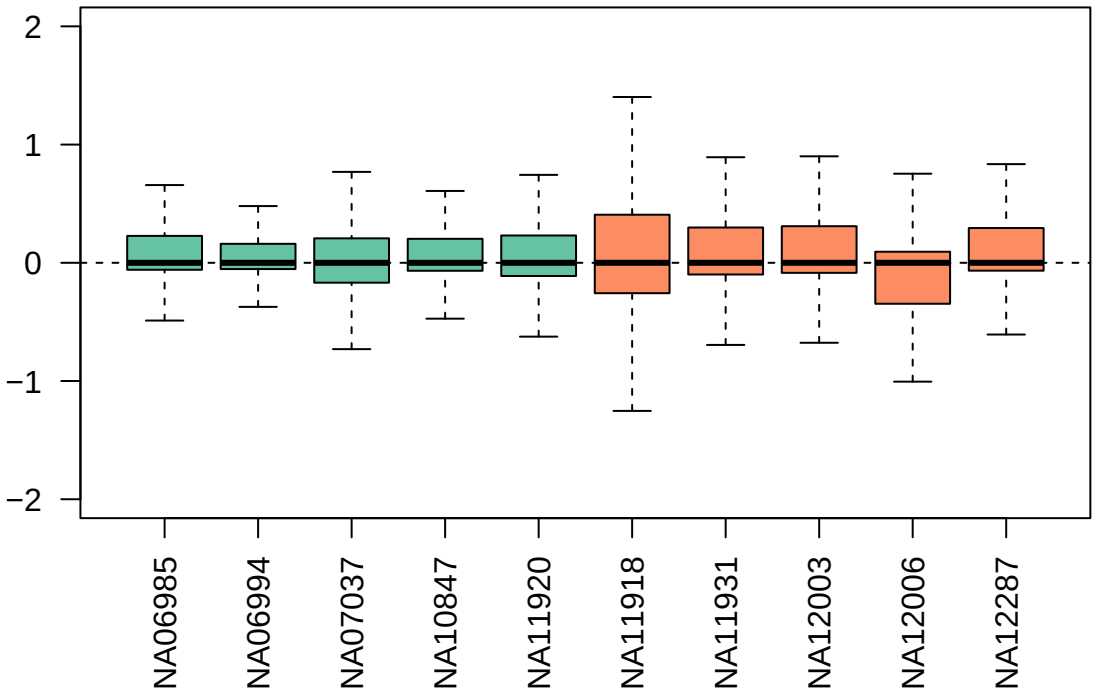
Part A

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?

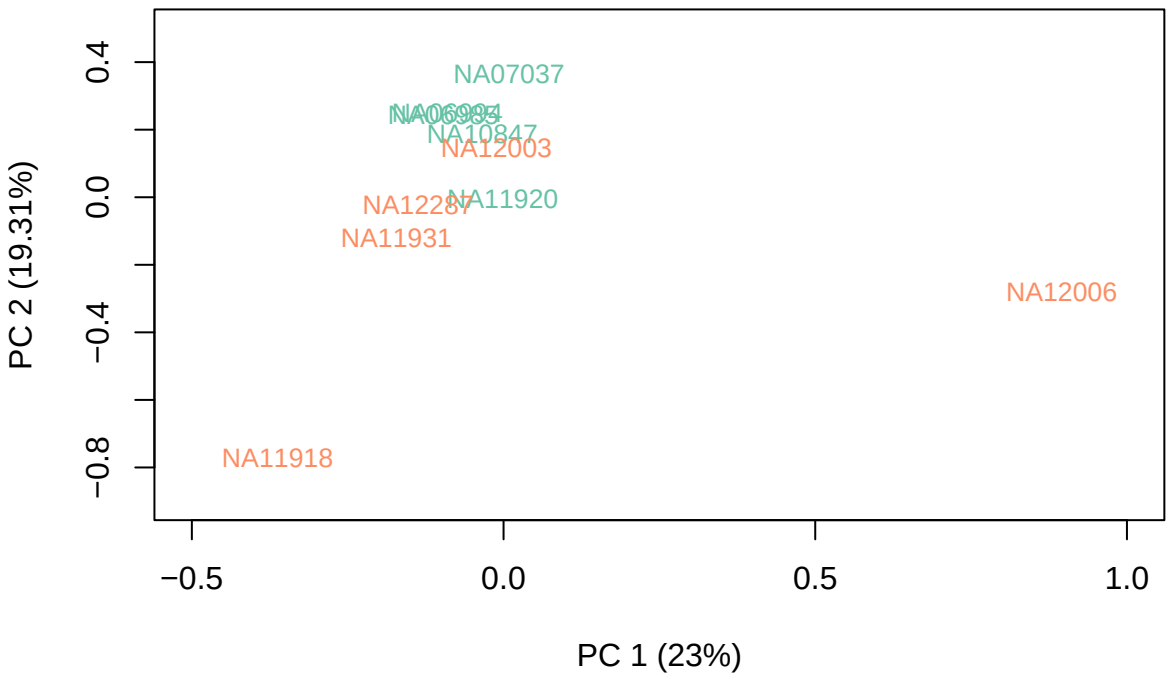
After filtering the genes for those with an FDR corrected p-value <0.05 there are 6 genes which are differentially expressed between the two groups. Details regarding calculation can be found in the code appendix.

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?



The relative log expression plot (RLE) shows that even after normalization there is significant differences in the variance between samples. These differences are most significant for the second group (orange).

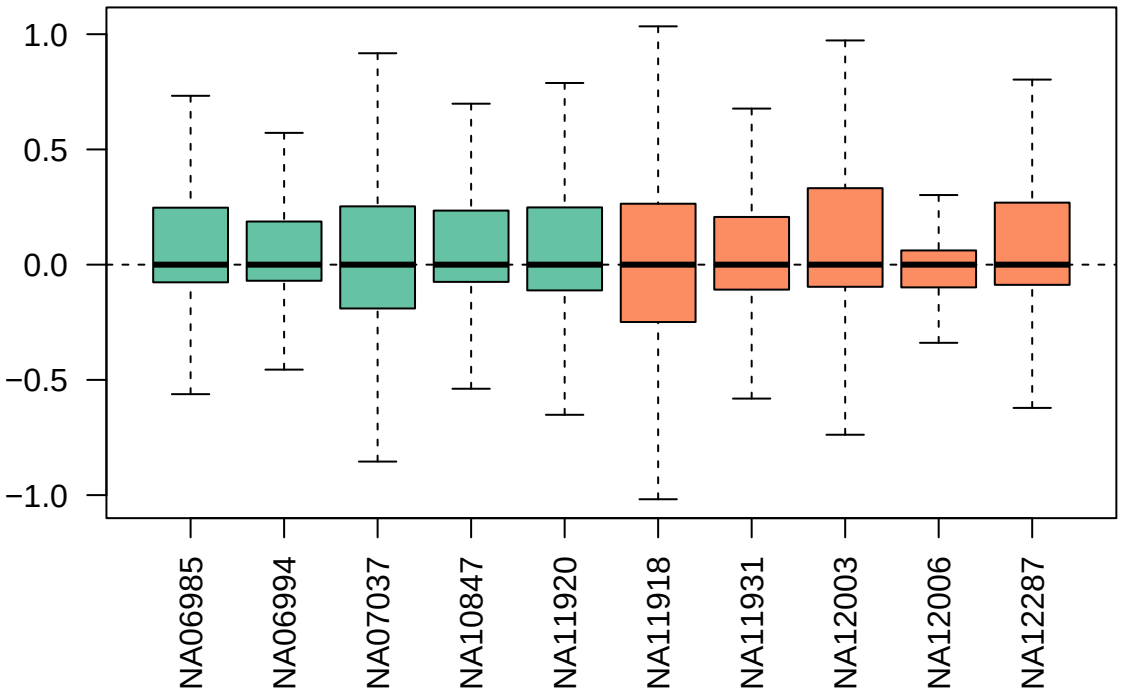


The plot above shows the PCA after normalization. Samples NA11918 and NA12006 are significantly different from the other samples, which is similar to the results from the RLE plot. Further normalization should be performed to reduce the variance in these samples.

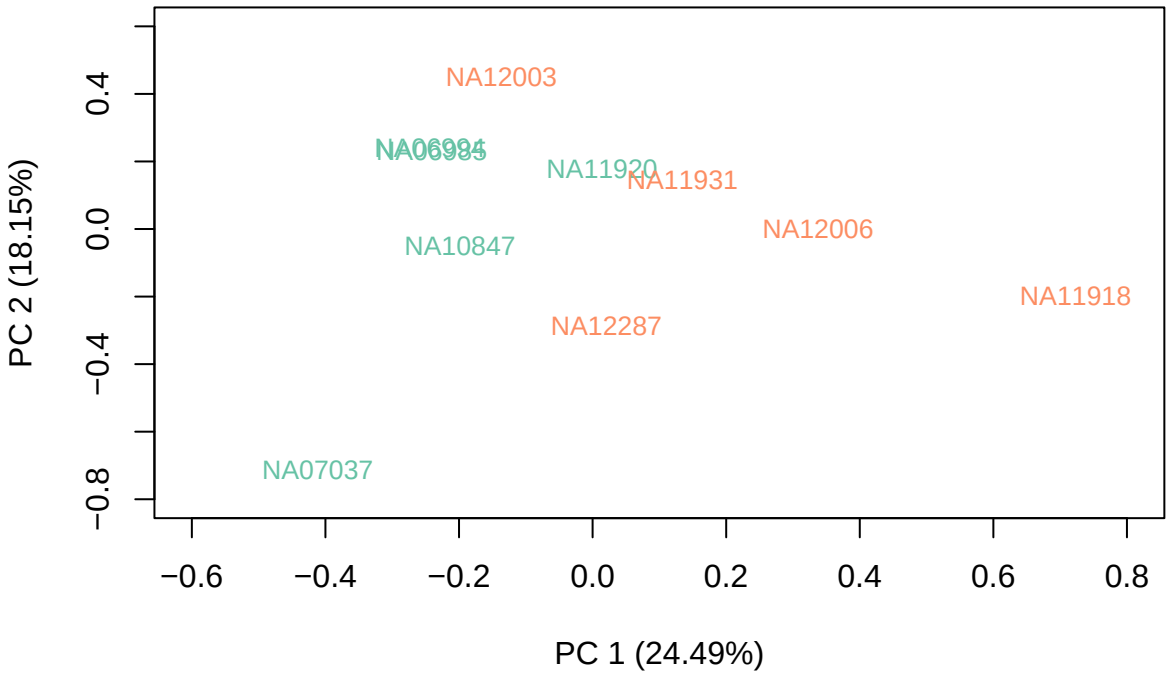
Part C

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.

After controlling for 1 factor of unwanted variation there are 65 genes with an FDR corrected p-value < 0.05. This is much larger than the original significant number of genes, which was 6.



The above RLE plot shows the results after RVUg using negative empirical control genes was performed. The variances for NA11918 and NA12003 are much closer to the other samples compared to the original plot.

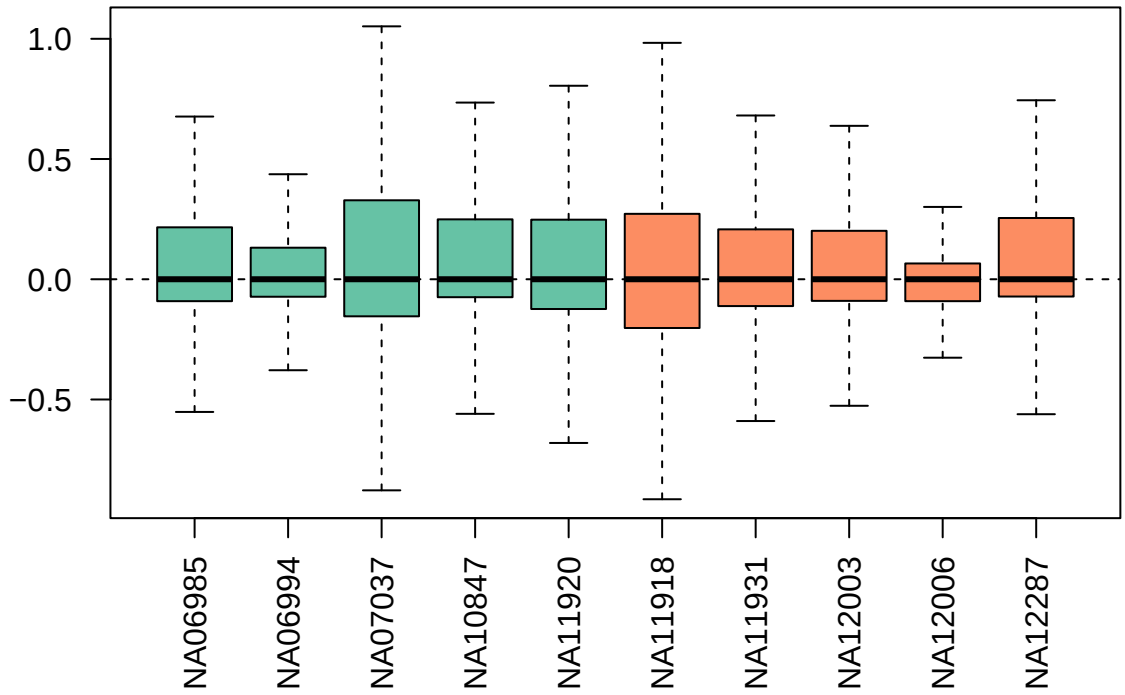


The plot above shows the PCA plot after RVUg using negative empirical control genes was performed. Samples NA11918 and NA12006 have been brought much closer to the other samples, although NA11918 is still noticeably separate from the other samples. Additionally, NA07037 is noticeably separate from the other samples as well.

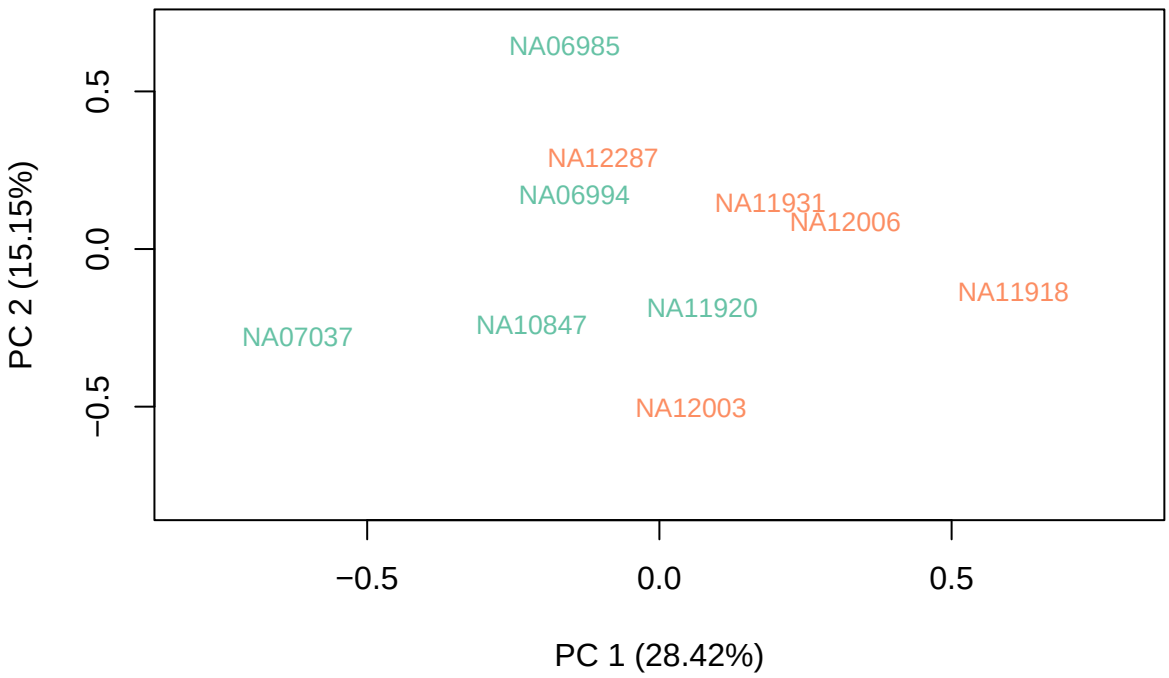
Part D

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?

After controlling for 2 factor of unwanted variation there are 36 genes with an FDR corrected p-value < 0.05. This is reduced from Part C, which had 65 significant genes.



The RLE plot above shows the results after controlling for 2 factors. This plot is not significantly different from the plot in part C.

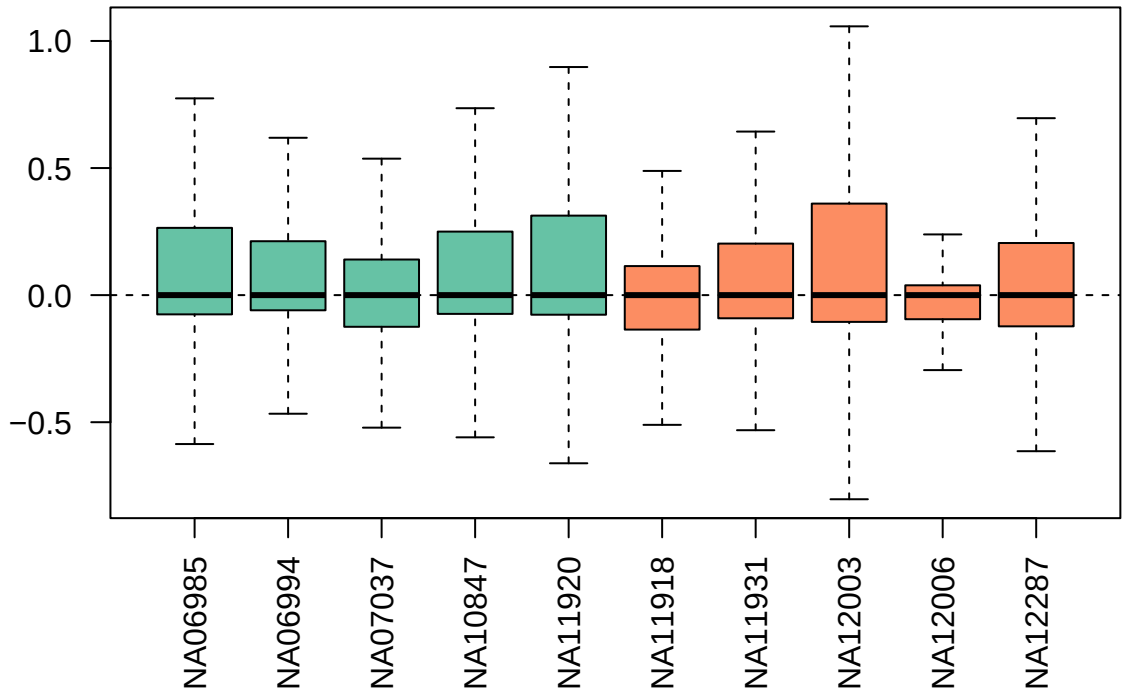


The PCA plot above shows the result after controlling for 2 factors of unwanted variation. This plot shows an improvement over Part D, where now the samples are much closer together.

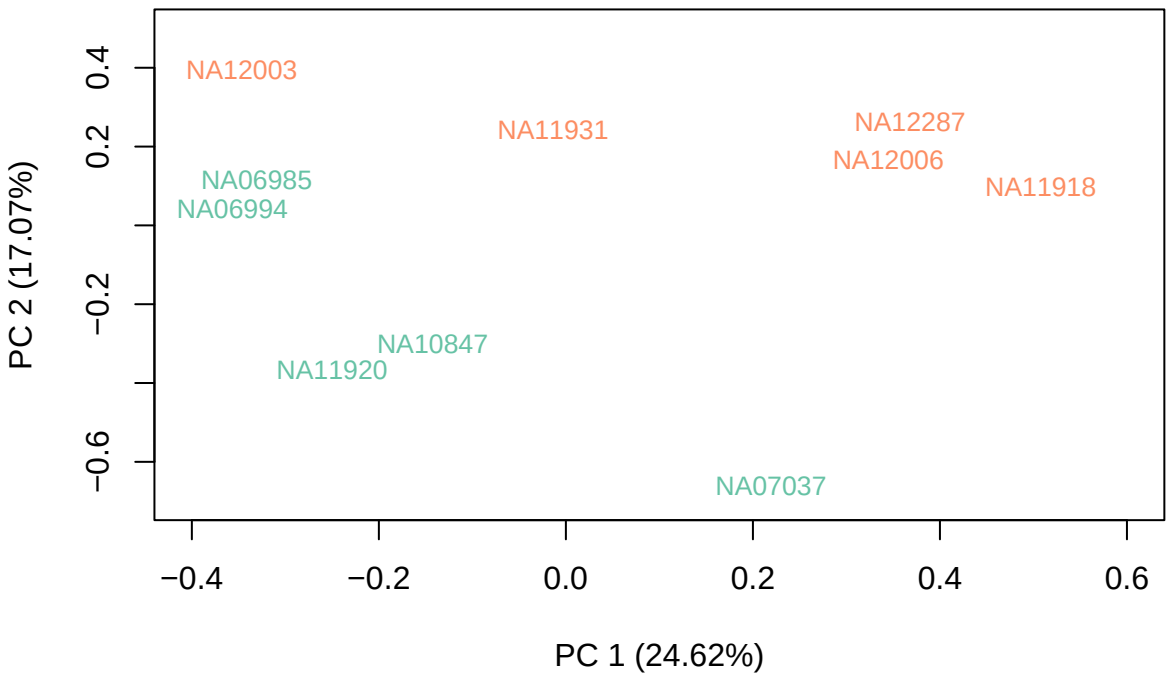
PART E

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 unwanted variation using RUVr?

After applying the RUVr method there are 135 genes with an FDR adjusted p-value < 0.05. This method returns the largest number of differentially expressed genes by a large margin.



The RLE plot above shows the results after controlling for 2 factors of unwanted variation and applying the RUVr method. Most of the variances have been brought closer together compared to Part D, with the exception of NA12003, which shows a noticeable increase.



The PCA plot above shows the after controlling for 2 factors of unwanted variation and applying the RUVr method. All samples have been brought closer together compared to Part D.

Part F

What are your thoughts about using RUV methods on this dataset and the effect of changing k or different versions of RUV?

From my limited experience the RUV methods appear to normalize data adequately compared to simply using between lane normalization methods. The ability to manually control for a different number of factors of unwanted variation (k) gives more control for normalization. However, Part D compared to Part C showed that removing additional factors of variation alone does not necessarily increase the number of differentially expressed genes identified. The RUVr (residuals) method with two unwanted factors of variation removed vastly outperformed any of the other models. I do not have sufficient experience to determine if this method is always better however, and application should still be done carefully.

CODE

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


### START QUESTION 1 CODE ###

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

bottomly.count.table <- exprs(bottomly.eset)


## 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.
```

```r
# 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")

## FINISH QUESTION 1 PART B CODE ##


## 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)

# Determining overlap between both methods.
DESeq2_sig_genes <- rownames(DESeq2_BH_corr)
edgeR_sig_genes <- rownames(edgeR_BH_corr)

similar_genes <- sum(DESeq2_sig_genes %in% edgeR_sig_genes)
```

```r
dissimilar_genes <- setdiff(DESeq2_sig_genes,edgeR_sig_genes)
# Will select gene ENSMUSG00000025217, which is only significant in
# DESeq2 and not edgeR.

DESeq2_example <- DESeq2_results_order["ENSMUSG00000025217",c(2,6)]
edgeR_example <- edgeR_results_order["ENSMUSG00000025217",c(1,6)]

colnames(DESeq2_example) <- c("Fold Change","BH p-value")
colnames(edgeR_example) <- c("Fold Change","BH p-value")

example_df <- rbind(DESeq2_example,edgeR_example)

rownames(example_df) <- c("DESeq2","edgeR")

example_df <- example_df %>%
  mutate(`BH p-value` = format.pval(`BH p-value`,digits = 4))

example_table <- kbl(example_df,
                     caption = "Example Gene ENSMUSG00000025217",
                     booktabs = T, align = "lcc", digits = 3) %>%
  kable_styling(latex_options = "HOLD_position")


example_table

## FINISH QUESTION 1 PART C CODE ##

### FINISH QUESTION 1 CODE ##


### 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)
```

```r
# Performing LRT to detect differential expression.
mgd_glmFit <- glmFit(mgd_dgel,mgd_design)
mgd_glmLRT <- glmLRT(mgd_glmFit, coef = 2)

# 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.
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))

## FINISH QUESTION 2 PART B CODE ##


## 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)


# Making a design matrix for the RUVg data.
```

```r
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, coef = 2)

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

# 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))

## FINISH QUESTION 2 PART C CODE ##


## START QUESTION 2 PART D CODE ##

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


# 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, coef = 2)

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

## FINISH QUESTION 2 PART D CODE ##


# Plotting relative log expression.
plotRLE(mgd_RUVg2,outline = FALSE, col = colors[mgd_group],
        cex = 0.80, las = 2)
```

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

## 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)


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, coef = 2)

# Extracting results.
mgd_RUVr2_results <- data.frame(topTags(mgd_RUVr2_glmLRT,n = Inf)) %>%
  filter(FDR < 0.05)


# 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))

## FINISH QUESTION 2 PART E CODE ##
```

```
### FINISH QUESTION 2 CODE ###
```