Differential Gene Expression Analysis

EdgeR & DESeq2 Comparison

Lisa Hoeg, 1151916

BINF6210 A5

1. Introduction

For this assignment, I am choosing option 7, which explores differential gene expression. There are many software packages and pipelines, such as voom-limma, baySeq, and Cufflinks, as have been extensively compared each showing their own strengths and best circumstances for use (Li et al. 2020). Even more packages are being developed to specialize the statistics and algorithms to single-cell RNA-seq data's unique considerations like increased zero read counts, but at this stage of development, analytic tools developed for bulk RNA-seq data such as DESeq2 and EdgeR perform comparably well (Wang et al. 2019). The abundance of available choices can be quite overwhelming to a new user. Based on a summary of fourteen studies comparing normalization methods, the five most recent included both edgeR and DESeq2 in the analysis, and the two packages consistently performed best under specific circumstances, or comparably to others (Li et al. 2020). As they seem consistently popular and well performing, I will be comparing EdgeR and DESeq, both of which had tutorials listed in the helpul links section of the assignment details (Chen et al. 2016; Love et al. 2015).

The main steps in differential gene expression analysis, in their simplest terms, are processing sequencing data into a count matrix, normalization of library sizes, and analysis of normalized counts. Each step in the process relies on certain statistical assumptions for the best way to process the data. I will begin analysis with the count matrix already constructed from an experiment uploaded to Gene Expression Omnibus (GEO). For normalization methods, EdgeR uses "Trimmed Mean of M-values" (TMM), and DESeq2 uses "Relative Log Expression" (RLE) (Maza 2016). Both edgeR and DESeq2 use an negative binomial distribution for the modeling of gene expression, but they diverge again at their test for differential expression, with EdgeR using the Exact test, and DESeq2 using a Wald test (De Paepe, 2014-2015; Tang et al. 2015).

To contribute to the ongoing community discussion of the relative merits of different gene expression packages, I will use the default statistical settings within each package when options exist, but otherwise keep filtering and processing of the data within each package whenever possible. The end goal is to produce an assessment of which package's statistical method, with specific reference to which results are most liberal or conservative in it's estimation of differentially expressed genes, and which package produces the most similar results to the original author's published analysis.

2. Description of Data Set

The data set for this assignment was downloaded from an original research project by Stilling et al. exploring which genes are differentially by social interaction tasks in conventionally (CON) raised mice, germ-free (GF) mice, and ex-GF mice that were colonized after weaning (2018). The research was interested in how the microbiome can impact gene expression and the associated metabolic pathways. There are six groups in the original study, naive (P) and social interaction (SI) mice for each of CON, GF, and ex-GF, totaling 40 samples. Experimental data is available through NCBI's Gene Expression Omnibus (GEO) using accession

number GSE114702, at the following location:
https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE114702

Count data was downloaded from GEO on 11/24/2020, and the results matrix was downloaded on 11/27/2020. For my study I will subset the data to a manageable size, and only use 4 biological replicates each of CON-SI and GF-SI mice.

3. Code Section 1 – Data Acquisition, Exploration, Filtering, and Quality Control

Preliminary Set-up & Data Acquisition

```
# All libraries were loaded at the beginning of the script for efficient organization.
# install.packages('stringr')
library(stringr)
# BiocManager::install('edgeR')
library(edgeR)
# BiocManager::install('DESeq2')
library(DESeq2)
# install.packages('ggplot2')
library(ggplot2)
# install.packages('gridExtra')
library(gridExtra)
# install.packages('readxl')
library(readxl)
# BiocManager::install('vidger')
library(vidger)
# install.packages('pheatmap')
library(pheatmap)
```

```
# A variable, 'dir' is used to define the working directory User can specify directory
# here for downloading and saving files, or leave empty (dir = '')
dir <- "/Users/lisa/Documents/Bioinformatics/6210/A5/scripts_data/a5_data/"
```

```
# Download count file with these two commands Count_URL <-
# paste('http://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE114702', 'format=file',
#'file=GSE114702_HTSeq_gene_counts.txt.gz', sep='&')
# Let user specify save name for file:
count_file <- "GSE114702_HTSeq_gene_counts.txt.gz"
# download.file(Count_URL, paste0(dir, count_file))

# Read in file; first column is the gene identifiers, which will be used for row names
countdata_mice <- read.delim(paste0(dir, count_file), row.names = 1)
rm(count_file)

# Check data loaded as expected:
class(countdata_mice)
dim(countdata_mice)
countdata_mice[1:5, c(1, 6, 18, 30)]
```

```
## [1] "data.frame"
## [1] 43629    40
##                    CON.P1 CON.S11 exGF.P2 GF.S11
```

```
## ENSMUSG00000000001    189     763     355   1052
## ENSMUSG00000000003      0       0       0      0
## ENSMUSG00000000028      5      23       6     25
## ENSMUSG00000000031      5      28      10     20
## ENSMUSG00000000037     11      15       6     26
```

```
# Download results spreadsheet Results_URL <-
# paste('http://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE114702', 'format=file',
# 'file=GSE114702_Processed_data_file_2_-_DESeq2_results.xlsx', sep='&') Let user specify
# save name for file:
res_file <- "GSE114702_DESeq2_results.xlsx"
# download.file(Results_URL, paste0(dir, res_file)

# If unsure which sheet, can check list: sheets <- excel_sheets(paste0(dir, res_file))
# sheets
res_OG <- read_excel(paste0(dir, res_file), sheet = "CONsi-GFsi", na = "NA")
# There are a few warnings reading in the spreadsheet that show Excel formatted a number
# into a date. Comparing the cells with warnings in spreadsheet to the dataframe in R,
# the cells belong to the '% GC content' variable, and seem to have an expected value R's
# dataframe. Since this analysis does not use % GC content, I will ignore it, but if it
# was part of the analysis, I might consider replacing those cells with NAs to be safe.

res_OG <- as.data.frame(res_OG)
rownames(res_OG) <- res_OG$`Ensembl Gene ID`

# Check results
class(res_OG)
res_OG[1:6, c(9, 12, 13)]
```

```
## [1] "data.frame"
##                     log2FoldChange       pvalue         padj
## ENSMUSG00000099980     1.81994890 6.582956e-41 9.498548e-37
## ENSMUSG00000000003             NA           NA           NA
## ENSMUSG00000056536     0.59611802 8.673109e-18 6.257215e-14
## ENSMUSG00000021098    -0.93604847 1.590558e-12 7.650053e-09
## ENSMUSG00000069117    -0.57894157 4.743351e-12 1.451312e-08
## ENSMUSG00000000049     0.01704621 8.289833e-01           NA
```

```
# In order to efficiently sample the biological replicates of the groups I wish to work
# with, I will first define a function to sample a single group at a time, and a
# companion function to sample all groups at once.
replicate_sample <- function(samples, group, size = 4) {
    gp_index <- grep(paste0("^", group), samples)
    sample_index <- sample(gp_index, size)
    return(sample_index)
}
```

```
# With the function for sampling the biological replicates defined, I will extract 4
# biological replicates for the social interaction group mice of the CON and GF types
# (CON.S & GF.S)
sample_names <- colnames(countdata_mice)
# Setting seed for reproducibility
set.seed(6210)
```

```r
# Applying function for 2 groups of interest
col_S_CON_GF <- lapply(c("CON.S", "GF.S"), FUN = function(x) replicate_sample(sample_names,
    x))
col_S_CON_GF <- unlist(col_S_CON_GF)
# Subsetting data
counts_S_CON_GF <- countdata_mice[, col_S_CON_GF]

# Check
colnames(counts_S_CON_GF)
# Remove variable no longer needed
rm(col_S_CON_GF, sample_names)
```

```
## [1] "CON.S12" "CON.S10" "CON.S11" "CON.S6"  "GF.S1"   "GF.S5"   "GF.S10"
## [8] "GF.S9"
```

Create & Explore Data Objects for EdgeR and DESeq2 Packages

```r
# DESeq2 requires a dataframe of column information for creating the DESeq Data Set
group_data <- str_remove(colnames(counts_S_CON_GF), pattern = ".S[0-9]+")
group_data <- factor(group_data)
coldata_mice <- data.frame(Group = group_data)

# Check
table(group_data)
class(coldata_mice)
```

```
## group_data
## CON  GF
##   4   4
## [1] "data.frame"
```

```r
# The count matrix and corresponding data is put together for EdgeR package
ER_Counts <- DGEList(counts_S_CON_GF, group = group_data)

# A few simple checks
class(ER_Counts)
# Shows group & library sizes
head(ER_Counts$samples, 4)
# Check the count matrix
head(ER_Counts$counts, 4)
```

```
## [1] "DGEList"
## attr(,"package")
## [1] "edgeR"
##         group lib.size norm.factors
## CON.S12   CON 17465387            1
## CON.S10   CON 10959458            1
## CON.S11   CON 15157737            1
## CON.S6    CON 13612582            1
##                    CON.S12 CON.S10 CON.S11 CON.S6 GF.S1 GF.S5 GF.S10 GF.S9
## ENSMUSG00000000001     871     607     763    724   325   829    709   872
## ENSMUSG00000000003       0       0       0      0     0     0      0     0
```

```
## ENSMUSG00000000028      34      20      23      14      5    13      16    19
## ENSMUSG00000000031      26      12      28      12      5    11      19    13
```

```
# In EdgeR, a design matrix is also used for some functions and can be created here. The
# design is specified simply in this case as ~0+group_data, so that samples are simply
# marked as part of a group or not.
ER_des.m <- model.matrix(~0 + group_data)
colnames(ER_des.m) <- levels(group_data)
# Check; transpose to take less space
t(ER_des.m)
```

```
##     1 2 3 4 5 6 7 8
## CON 1 1 1 1 0 0 0 0
## GF  0 0 0 0 1 1 1 1
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$group_data
## [1] "contr.treatment"
```

```
# The count matrix and corresponding data is put together for the DESeq2 object. The same
# design as used for EdgeR is specified.
DS_Counts <- DESeqDataSetFromMatrix(countData = counts_S_CON_GF, colData = coldata_mice,
    design = ~0 + Group)

# A few simple checks
class(DS_Counts)
# Shows library sizes
colSums(counts(DS_Counts))
# Check the count matrix
head(assay(DS_Counts), 4)
```

```
## [1] "DESeqDataSet"
## attr(,"package")
## [1] "DESeq2"
##  CON.S12  CON.S10  CON.S11   CON.S6    GF.S1    GF.S5   GF.S10    GF.S9
## 17465387 10959458 15157737 13612582  7291810 15937529 12979407 15103884
##                    CON.S12 CON.S10 CON.S11 CON.S6 GF.S1 GF.S5 GF.S10 GF.S9
## ENSMUSG00000000001     871     607     763    724   325   829    709   872
## ENSMUSG00000000003       0       0       0      0     0     0      0     0
## ENSMUSG00000000028      34      20      23     14     5    13     16    19
## ENSMUSG00000000031      26      12      28     12     5    11     19    13
```

Filtering and Quality Control

```
# The EdgeR has a function designed for filtering according to user-set parameters, which
# returns a logical vector that can be passed to subset the count data object.  The
# default parameters for are min.count=10 (minimum numeric count for at least some
# samples), min.total.count = 15, and min.prop=0.7, with values being calculated as
# counts per million (CPM)
ER_filt_data_def <- filterByExpr(ER_Counts, ER_des.m)
table(ER_filt_data_def)
```

```
## ER_filt_data_def
## FALSE  TRUE
## 28734 14895
```

```
# The DESeq2 tutorial filters out genes that have 1 or fewer reads total. The authors
# describe this only as a pre-filtering step to eliminate empty and nearly empty rows,
# with further filtering occurring downstream.
DS_filt_data_def <- rowSums(counts(DS_Counts)) > 1
table(DS_filt_data_def)
```

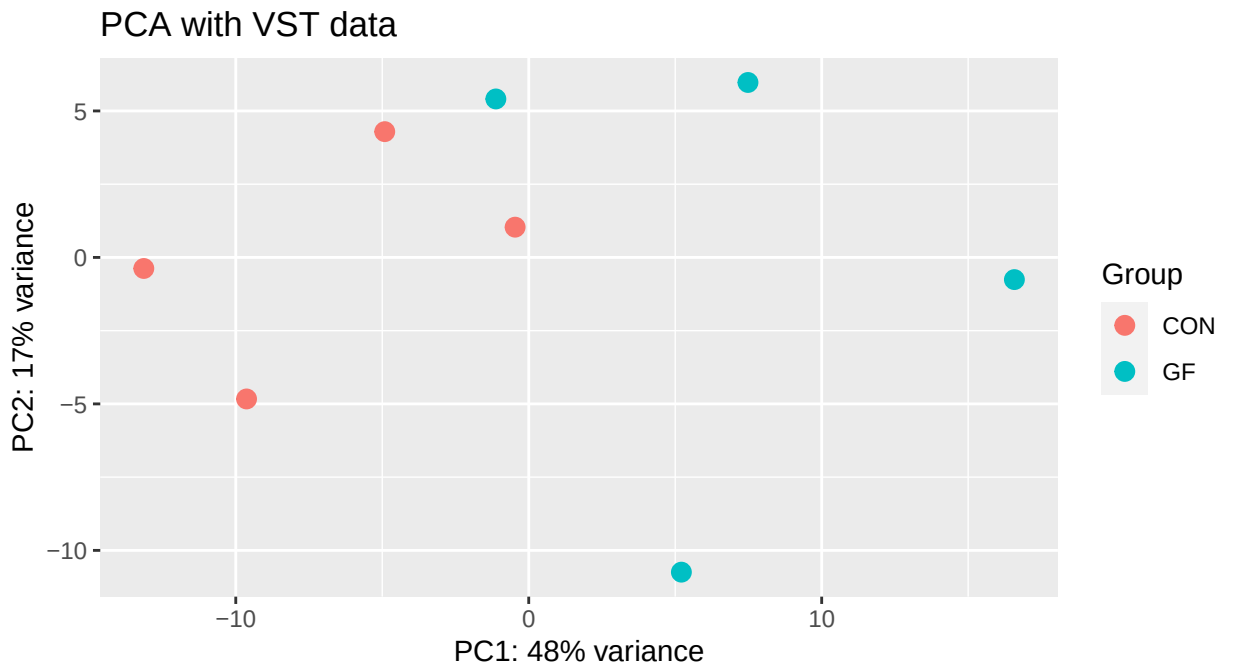
```
## DS_filt_data_def
## FALSE  TRUE
## 20023 23606
```

```
# The original paper indicated that they used DESeq2 with default options for their
# analysis, but did not specify if that included the pre-filtering. We can use the number
# of genes that have 'NA' in the p-value column as a guess for which genes were removed
# prior to analysis
data.frame(`NA` = sum(is.na(res_OG$pvalue)), DATA = sum(!is.na(res_OG$pvalue)))
```

```
##     NA.  DATA
## 1 15590 28039
```

```
# Since the number of remaining genes in the original paper, 28039, is similar to the
# amount using the recommended filter of the DESeq2 package, 23606, and quite different
# from the EdgeR filtering at this stage (14895 remaining), I will use the filtering
# recommeded in the DESeq2 tutorial.  EdgeR has an internal library sizes variable, so we
# use the argument keep.lib.sizes = FALSE to reevaluate the library sizes after
# eliminating some genes.  DESeq2 uses the sum of count data, and does not need the
# argument.
ER_Counts <- ER_Counts[DS_filt_data_def, , keep.lib.sizes = FALSE]
DS_Counts <- DS_Counts[DS_filt_data_def, ]
```

```
# The quality of data can be checked by comparing the relative distances of samples with
# a Principal Component Analysis (PCA) plot, which visually shows the higher-order
# distances between samples using transformed data. Variance stabilizing transformation
# (VST) will be used, as recommended by Love et al.  Since only the DESeq2 package
# provides a function for plotting the PCA, and the counts are identical at this point, I
# will proceed with the count matrix in the DDS object
vst_Counts <- vst(DS_Counts, blind = FALSE)
# Assign PCA data to an object to plot with ggplot
pcaData <- plotPCA(vst_Counts, intgroup = "Group")
# Calculate variances of top 2 components for labeling axes
ggplot(pcaData$data, aes(x = PC1, y = PC2, col = Group)) + geom_point(size = 3) + xlab(pcaData$labels$x
    ylab(pcaData$labels$y) + coord_fixed() + ggtitle("PCA with VST data")
```




```
# This plot shows that there is a definite pattern to the distribution between the GF and
# CON mice sampled, but the space between the two groups is not too different than the
# space within each group.
```

Library Normalization

```
# Library size normalization is where the two packages diverge significantly, each using
# a different statistical method In EdgeR, the default normalization is done by TMM, and
# the factors are added into the samples feature of the DGEList object
ER_Counts_N <- calcNormFactors(ER_Counts)
head(ER_Counts_N$samples, 4)
```

```
##         group lib.size norm.factors
## CON.S12   CON 17465046     1.031010
## CON.S10   CON 10959251     1.017135
## CON.S11   CON 15157428     1.037407
## CON.S6    CON 13612239     1.031932
```

```
# In DESeq2, the default normalization is done as part of the internal process of the
# DESeq command that determines which genes are differentially expressed. We can still
# calculate the size factors separately using the 'estimateSizeFactorsForMatrix' command,
# or add to the DESeqDataSet object using the estimateSizeFactors' command.

# DESeq2 uses RLE to estimate size factors
DS_Counts_Nf <- estimateSizeFactorsForMatrix(counts(DS_Counts))
head(DS_Counts_Nf, 4)
```

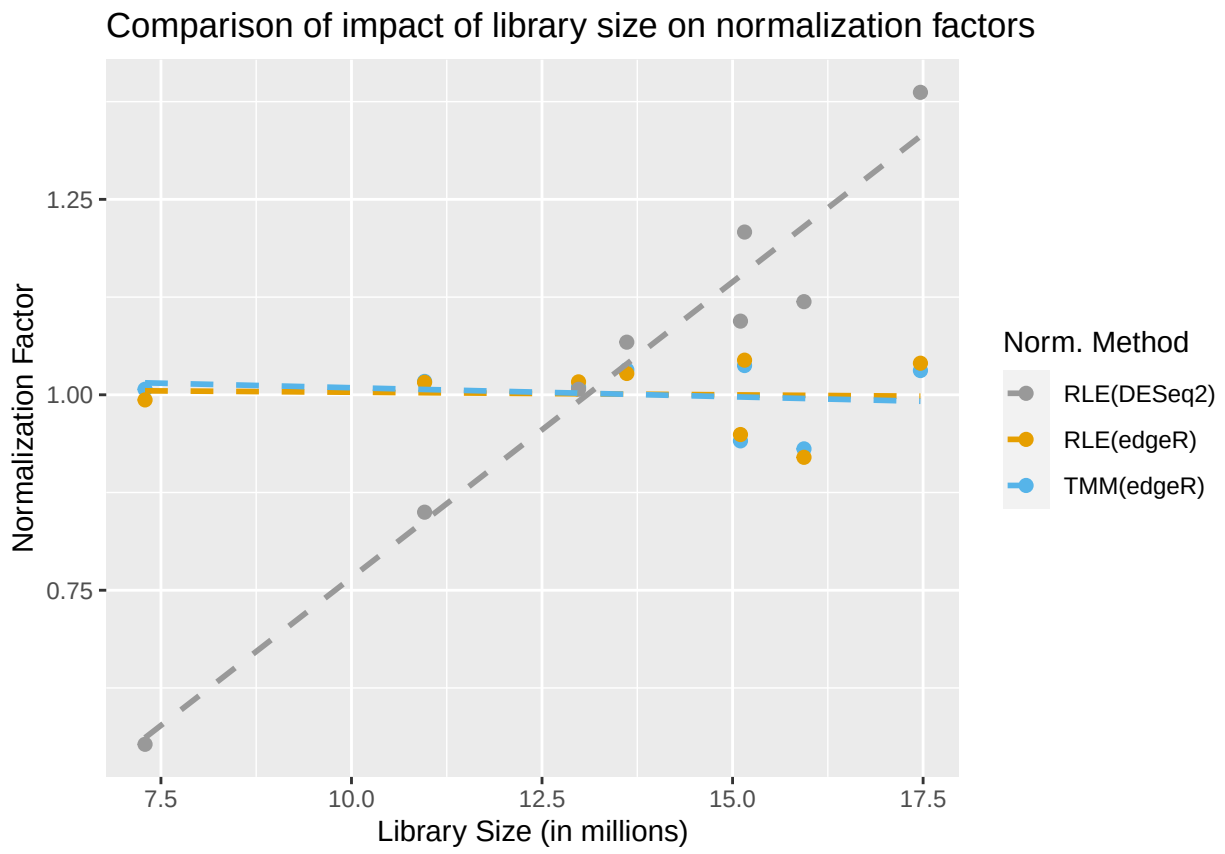
```
##   CON.S12   CON.S10   CON.S11    CON.S6
## 1.3869103 0.8498422 1.2081855 1.0673189
```

```
# EdgeR has an option to specify RLE as the statistical method, but the process is
# different, so the results will not be the same. I will calculate here for comparison
ER_Counts_RLE <- calcNormFactors(ER_Counts, method = "RLE")
head(ER_Counts_RLE$samples, 4)
```

```
##         group lib.size norm.factors
## CON.S12   CON 17465046     1.040399
## CON.S10   CON 10959251     1.015965
## CON.S11   CON 15157428     1.044310
## CON.S6    CON 13612239     1.027273
```

Data Visualization

```
# To compare different normalization methods make dataframe of library size and
# normalization factors to pass to ggplot
df_norm_factors <- rbind(ER_Counts_N$samples, ER_Counts_RLE$samples, data.frame(group = rep("RLE(DESeq2)
    8), lib.size = colSums(counts(DS_Counts)), norm.factors = DS_Counts_Nf))
df_norm_factors$group <- unfactor(df_norm_factors$group)
df_norm_factors$group[1:16] <- c(rep("TMM(edgeR)", 8), rep("RLE(edgeR)", 8))

ggplot(df_norm_factors, aes(x = lib.size/(10^6), y = norm.factors, col = group)) + geom_point(size = 2)
    scale_color_manual(values = c("#999999", "#E69F00", "#56B4E9")) + labs(x = "Library Size (in millio
    y = "Normalization Factor", title = "Comparison of impact of library size on normalization factors"
    guides(col = guide_legend("Norm. Method"), group = FALSE) + geom_smooth(method = lm,
    se = FALSE, fullrange = TRUE, linetype = "dashed")
```




```
# This plot demonstrates that the internal statistics of normalization in EdgeR do not
# depend on the size of the library, but DESeq2 does. The two methods within EdgeR are
# much more similar than the same method (RLE) in the two packages.
```

4. Main Software Tools Description

The default normalization methods of the EdgeR and DESeq2 packages differ in both statistical model, and design on its application. The main difference in process is that DESeq2 generates a correction factor that can be applied to the raw counts because it already accounts for the library size, but EdgeR calculates a correction factor that will be applied to the library size during the differential expression analysis, but does not include the library size in the calculation of the correction factor (De Paepe 2014-2015; Maza 2016; Varet et al.: S1Appendix 2016) . This can be seen in the plot above where the linear regression line for the different statistical normalization methods applied in EdgeR are nearly horizontal, but the correction factor increases with increasing library size for DESeq2. A thorough exploration of the steps and corresponding mathematics for each method can be found in Maza (2016), and an in-depth discussion of how the normalization methods differ can be found in the Supplementary Appendix file for Varet et al. (2016). The normalization method can be highly influential on the final results of the differential expression (DE) analysis (Maza 2016), but the statistical model for each package at the DE stage is also different for each package. The underlying principle to each package is the same – a using a negative binomial distribution and the empirical Bayes method—but the application of different tests for differential expression (exact test in EdgeR vs. Walt test in DESeq2) can impact the results (Maza 2016; Wang 2019; De Paepe 2014-2015; Li 2020).

5. Code Section 2 – Main Analysis

Differential Gene Expression

```
# EdgeR First constructs a contrast matrix. This step seems most relevant when there are
# multiple possible contrasts in the complete data set, but as I have only retained naive
# mice for CON and GF types, it is basically just reformatting the existing design
ER.CONvsGF <- makeContrasts(CON - GF, levels = ER_des.m)
class(ER.CONvsGF)
```

```
## [1] "matrix" "array"
```

```
ER.CONvsGF
```

```
##       Contrasts
## Levels CON - GF
##    CON         1
##    GF         -1
```

```
# Estimates dispersion by maximizing negative binomial likelihood
ER_Disp <- estimateDisp(ER_Counts_N, ER_des.m, robust = TRUE)

# EdgeR uses the dispersion to estimate the GLM for each gene
ER_fit <- glmQLFit(ER_Disp, ER_des.m, robust = TRUE)

# The results are then calculated using the contrast matrix and the GLM fit
res_ER <- glmQLFTest(ER_fit, contrast = ER.CONvsGF)

class(res_ER)
```

```
## [1] "DGELRT"
## attr(,"package")
## [1] "edgeR"
```

```
# The top tags extracts the 10 genes listed by order of lowest p-value/FDR (False
# Discovery Rate) and provides some useful statistics
head(topTags(res_ER), 4)
```

```
## Coefficient:  1*CON -1*GF
##                         logFC    logCPM         F       PValue         FDR
## ENSMUSG00000099980 -6.0304188 0.5825342 102.92542 2.029108e-08 0.000425745
## ENSMUSG00000093483  3.4030827 2.8058075 123.71567 3.607091e-08 0.000425745
## ENSMUSG00000052684 -0.8733963 6.1380529  76.53747 1.568760e-07 0.001234405
## ENSMUSG00000021098  3.2411723 0.8014206  47.61730 3.368870e-06 0.019881387
```

```
# Identify the top 10 genes by p-value for comparison with the DESeq method
top_p_ER <- row.names(topTags(res_ER))
```

```
# DESeq2 The first step does the estimation of size factors, dispersion estimate, and GLM
# fitting all internally, and thus uses the non-normalized dataset as the input.
DS.CONvsGF <- DESeq(DS_Counts)
```

```r
# The results function extracts a results table from the DESeq object
res_DS <- results(DS.CONvsGF)
class(res_DS)
colnames(res_DS)

# Identify the top 10 genes by adjusted p-value, which is comparable to EdgeR's FDR for
# comparison with the EdgeR method
top_p_DS <- row.names(head(res_DS[order(res_DS$padj), ], 10))
```

```
## [1] "DESeqResults"
## attr(,"package")
## [1] "DESeq2"
## [1] "baseMean"       "log2FoldChange" "lfcSE"          "stat"
## [5] "pvalue"         "padj"
```

```r
# Check if same genes had most significant p-values in both packages:
top_p_DS_ER <- intersect(top_p_DS, top_p_ER)
length(top_p_DS_ER)
```

```
## [1] 6
```

```r
# Compare to results from original paper,
top_p_OG <- row.names(head(res_OG[order(res_OG$padj), ], 10))

top_p_OG_DS <- intersect(top_p_OG, top_p_DS)
length(top_p_OG_DS)
```

```
## [1] 4
```

```r
top_p_OG_ER <- intersect(top_p_OG, top_p_ER)
length(top_p_OG_ER)
```

```
## [1] 5
```

```r
top_p_all <- intersect(intersect(top_p_DS, top_p_ER), top_p_OG)
length(top_p_all)
```

```
## [1] 4
```

```r
# The function used to determine differential gene expression in EdgeR allows the p.value
# cut-off to be changed. I will use 0.1, as that is the value indicated in the original
# research by Stilling et al.
ER_p10 <- decideTestsDGE(res_ER, p.value = 0.1)
class(ER_p10)
```

```
## [1] "TestResults"
## attr(,"package")
## [1] "limma"
```

```r
summary(ER_p10)
```

```
##        1*CON -1*GF
## Down             5
## NotSig       23595
## Up               6
```

```r
# So few! Only 11

# DESeq2 uses the subset function to apply cutoff values to the p.value or other
# statistics.
DS_p10 <- subset(res_DS, padj < 0.1)
summary(DS_p10)
```

```
##
## out of 313 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 205, 65%
## LFC < 0 (down)     : 108, 35%
## outliers [1]       : 0, 0%
## low counts [2]     : 0, 0%
## (mean count < 4)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

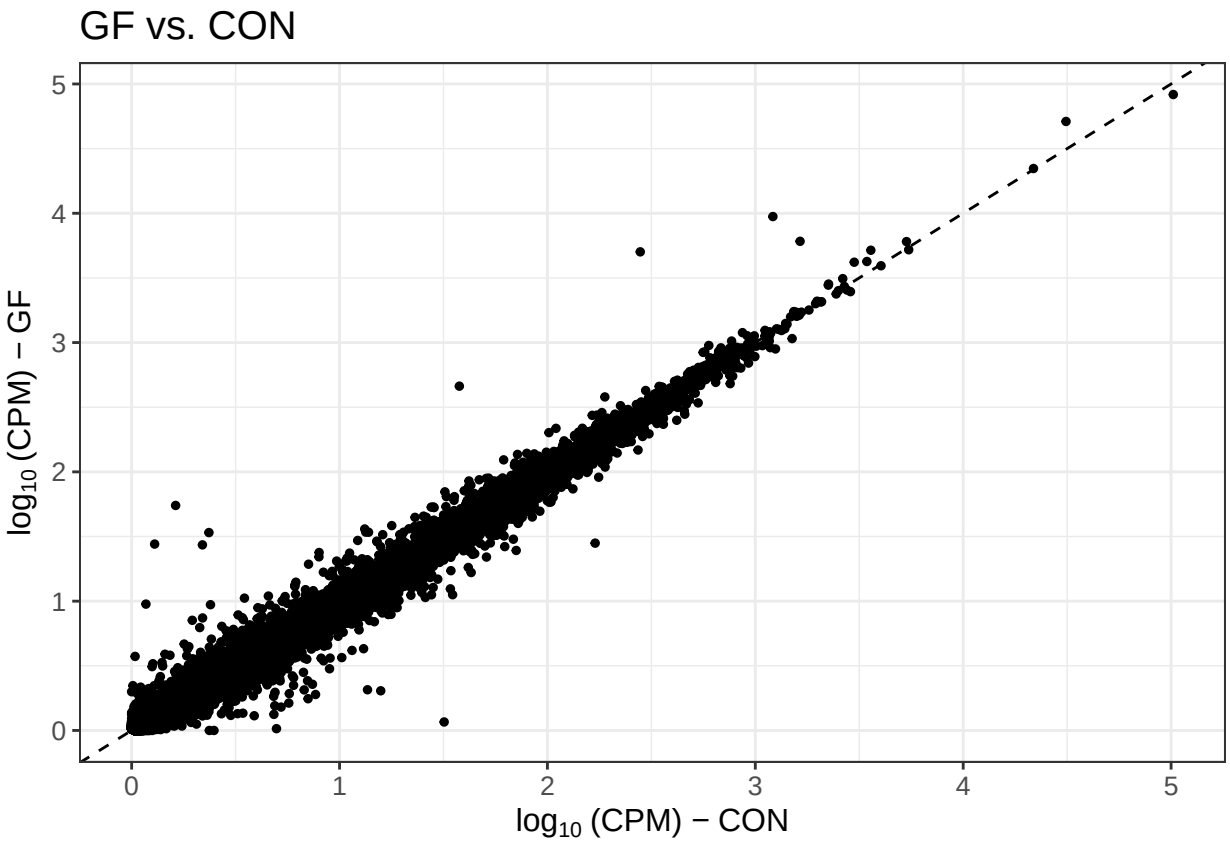
```r
# Far more genes were determined to be differentially expressed, total 313

# Original data
sum(res_OG$padj < 0.1, na.rm = TRUE)
```

```
## [1] 531
```

```r
# 531 met the criteria of the original study.
```

Visualizations

```r
# vidger is a package designed for plots that help interpret gene expression data. The
# Scatterplot shows the counts per million (CPM) of a pairwise comparison of groups. The
# closer to the diagonal line, the more similar the counts in the groups, the further
# from the line, the more different.  The package can be used with a number of popular DE
# packages, and the raw counts are normalized to CPM in the function, so the raw counts
# can be used.
vsScatterPlot(x = "CON", y = "GF", data = ER_Counts, type = "edger")
```



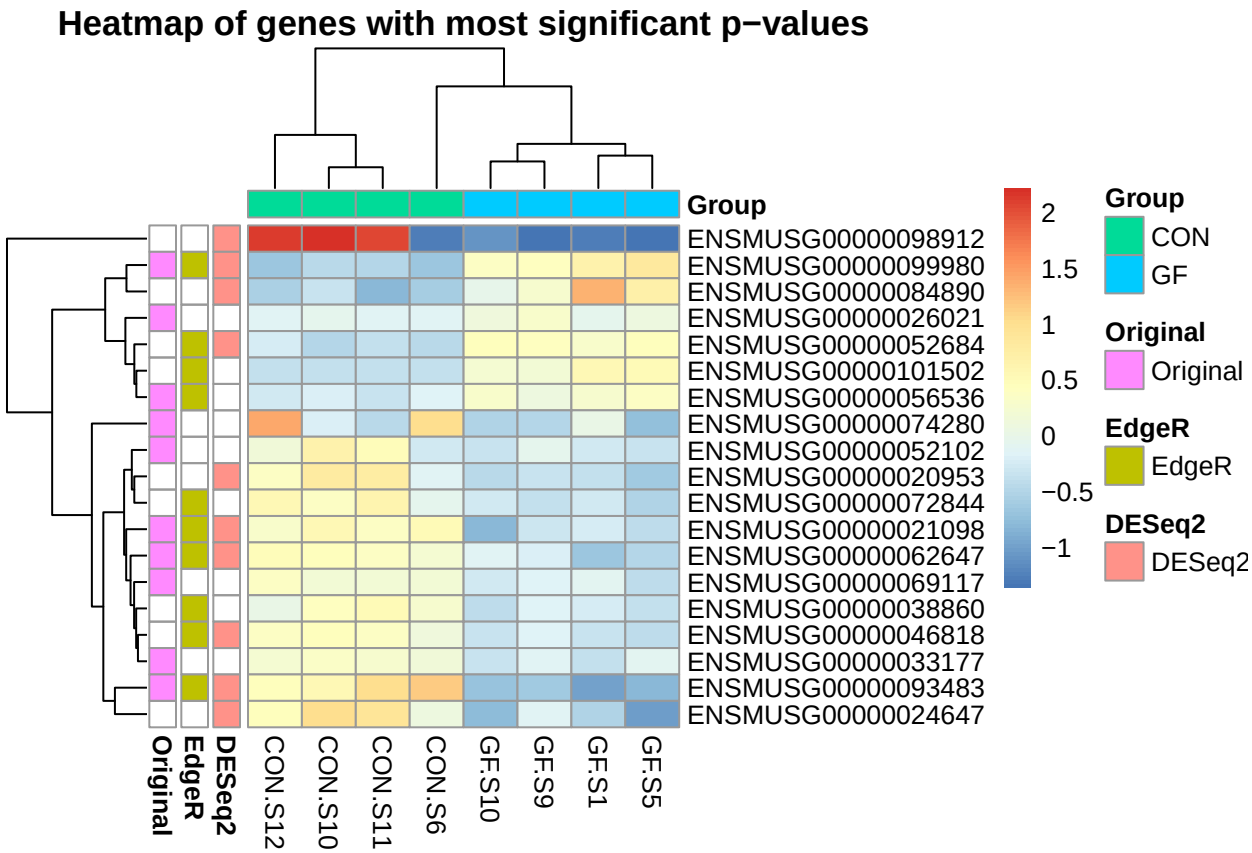

```
# Most genes fall close to the line, with only a few genes showing large differences
# between the groups.
```

```
# To make heatmap I'll subset the combined total of the top 10 most significant p value
# genes from each package and the original Sampling the count matrix that with VS
# transformation & difference from gene's mean count
top_p_full <- unique(c(top_p_DS, top_p_ER, top_p_OG))
vst_top_Counts <- assay(vst_Counts)[top_p_full, ]
vst_top_Counts <- vst_top_Counts - rowMeans(vst_top_Counts)

# Make dataframe for column annotations
group_anno <- data.frame(Group = unfactor(group_data))
rownames(group_anno) <- colnames(vst_top_Counts)

# Make dataframe for row annotations
df_top_p <- data.frame(DESeq2 = rep("DESeq2", 19), EdgeR = rep("EdgeR", 19), Original = rep("Original",
    19))
rownames(df_top_p) <- top_p_full
df_top_p[!(rownames(df_top_p) %in% top_p_DS), "DESeq2"] <- NA
df_top_p[!(rownames(df_top_p) %in% top_p_ER), "EdgeR"] <- NA
df_top_p[!(rownames(df_top_p) %in% top_p_OG), "Original"] <- NA

# Check
head(df_top_p, 4)
```

```
##                     DESeq2 EdgeR Original
```

```
## ENSMUSG00000093483 DESeq2 EdgeR Original
## ENSMUSG00000052684 DESeq2 EdgeR     <NA>
## ENSMUSG00000021098 DESeq2 EdgeR Original
## ENSMUSG00000099980 DESeq2 EdgeR Original
```

```
# Heatmap
pheatmap(vst_top_Counts, main = "Heatmap of genes with most significant p-values", annotation_col = grou
    annotation_row = df_top_p)
```




```
# As expected, the CON mice and GF mice mostly cluster together, with outlier mouse
# CON.S6 being grouped with the GF mice. It is also interesting that the clustering of
# the genes shows no particular pattern to the package that identified them.
```

8. Results and Discussion

The EdgeR package found only 11 genes to be significant at a false discovery rate of 0.1, the same cut-off as was applied in the original research, in comparison to the 313 in DESeq2 and 513 in the original study. Since the published study used DESeq2 in their analysis, it was expected that they would be the most similar, but the wide differences between the three results is surprising. Of the 10 genes with the most significant p-value in each process, six were the same between EdgeR and DESeq2, and four were found in both packages and the results table provided by the original authors. This further demonstrates how different packages can produce large discrepancies with small changes in the methodologies. The heatmap of the most significantly different genes grouped the CON and GF mice each together with the exception of one outlier in the CON mice, which was as expected. The other interesting result of the heatmap was that the genes clustering

showed that the genes identified by each package were evenly distributed throughout the cluster, suggesting that statistical models used in each package are not producing a strong bias of a single type.
Although the comparison between packages for differential gene expression showed quite different results of which genes were considered differentially expressed, with EdgeR being much more conservative in its estimation, the scatterplot of normalized counts for the two groups shows that the divergence of most genes from the centre diagonal is minimal. This suggests that the statistical difference between which genes in my sample that would or would not be considered differentially expressed is quite sensitive (McDermaid et al. 2019). As such, the differences between the packages is not as surprising as was first believed.
My study subsampled the original data to only use 4 of the biological replicates each of the original study which had data for 8 mice in the CON-social condition, and 12 mice in the GF-social condition. The agreement between the packages studied and the original study might have been improved by including all of the same biological replicates that the study did. Interestingly, Li et al. (2020) found that either normalization method (TMM or RLE) performed better at differential expression analysis on their simulated data with small sample sizes by using EdgeR's Exact test than with DESeq's Wald test, which performed best at large samples sizes. Although identifying more genes in my analysis does not on its own make DESeq2 the better tool, but it's closer approximation to the original study supports that conclusion, which suggests that my study demonstrated the opposite results from Li et al.'s study (2020).

9. Acknowledgements

All work was done independently, referencing lecture material, vignettes, and online documentation as listed in the references below.

10. References

Academic References

De Paepe, K. (2014-2015). Comparison of methods for differential gene expression using RNA-seq data. Universiteit Gent: Master dissertation

Li, X, Cooper, NGF, O'Toole, TE, Rouchka, EC. (2020). Choice of library size normalization and statistical methods for differential gene expression analysis in balanced two-group comparisons for RNA-seq studies. *BMC Genomics*, 21:75. https://doi.org/10.1186/s12864-020-6502-7

Maza, E. (2016). In Papyro Comparison of TMM (edgeR), RLE (DESeq2), and MRN Normalization Methods for a Simple Two-Conditions-Without-Replicates RNA-Seq Experimental Design. *Frontiers in Genetics*, 7(n° 164). https://doi.org/10.3389/fgene.2016.00164

McDermaid, A, Monier, B, Zhao, J, Liu, B, Ma, Q. (2019). Interpretation of differential gene expression results of RNA-seq data: review and integration. *Briefings in Bioinformatics*, 20(6), 2044–2054. https://doi.org/10.1093/bib/bby067

Stilling, RM, Moloney, GM, Ryan, FJ, Hoban, AE, Bastiaanssem, TFS, Shanahan, F. (2018). Social interaction-induced activation of RNA splicing in the amygdala of microbiome-deficient mice. *eLife*, 7. https://doi.org/10.7554/elife.33070

Tang, M, Sun, J, Shimizu, K, Kadota, K. (2015). Evaluation of methods for differential expression analysis on multi-group RNA-seq count data. *BMC Bioinformatics*, 16:361. https://doi.org/10.1186/s12859-015-0794-7

Varet, H, Brillet-Guéguen, L, Coppée, J-Y, Dillies, M-A. (2016). SARTools: A DESeq2- and EdgeR-Based R Pipeline for Comprehensive Differential Analysis of RNA-Seq Data. *PloS One*, 11(6), e0157022. https://doi.org/10.1371/journal.pone.0157022

Wang, T, Li, B, Nelson, CE, Nabavi, S. (2019). Comparative analysis of differential gene expression analysis tools for single-cell RNA sequencing data. *BMC Bioinformatics*, 20:40. https://doi.org/10.1186/s12859-019-2599-6

Coding References

Chen, Y, Lun, ATL, Smyth, GK. (2016). From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline. *F1000 Research, 5*, 1438–1438. https://doi.org/10.12688/f1000research.8987.2

Love, MI, Anders, S, Kim, V, Huber, W. (2015). RNA-Seq workflow: gene-level exploratory analysis and differential expression. *F1000 Research, 4*, 1070. https://doi.org/10.12688/f1000research.7035.1

Pheatmap documentation (Greg Finak):
https://www.rdocumentation.org/packages/COMPASS/versions/1.10.2/topics/pheatmap

Knitr options (Yihui Xie):
https://yihui.org/knitr/options/

DESeq2 workflow tutorial (hbctraining):
https://hbctraining.github.io/DGE_workshop/lessons/04_DGE_DESeq2_analysis.html

vidger R package documentation (Brandon Monier):
https://www.bioconductor.org/packages/release/bioc/manuals/vidger/man/vidger.pdf