Grape Differential Expression Analysis with EdgeR

Scott Teresi

Purpose:

Produce a matrix of differentially expressed genes for the grape RNA data. I will be using the package edgeR for this task.

Accessibility and Help:

The following guide, source code, and other components of the grape expression analysis pipeline can be found at Grape Expression Analysis Github page. This page includes general information, the keys of the samples, and the list of comparisons desired.

Installation and Loading of Library:

Using this link as reference, please install **edgeR** with:

```r
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("edgeR")
```

We also need tidyverse and dplyr, which can be installed with:

```r
install.packages("tidyverse")
install.packages("dplyr")
```

And then load the libraries with:

```r
library(edgeR)
library(tidyverse)
library(dplyr)
```

Analysis of the 1st Chunk of Data:

Our first chunk of data comes from the **20190809_mRNASeq_PE150** group and the subsequent **809.tsv** output from HtSeq. First we will load the data, then we have to use a lot of *regex* and subsetting to appropriately identify the samples.

```
# Change this to your own path
setwd('/home/scott/Documents/Uni/Research/Projects/Grape_RNA')

# Import data
Seq_809 = read.csv('809_Seq.tsv', sep='\t',
           stringsAsFactors = FALSE,
           header = TRUE)

Seq_802 = read.csv('802_Seq.tsv', sep='\t',
           stringsAsFactors = FALSE,
           header = TRUE)

Seq_724 = read.csv('724_Seq.tsv', sep='\t',
           stringsAsFactors = FALSE,
           header = TRUE)

# Merge that data
my_data = merge(Seq_809, Seq_802, by = 'gene_id')
my_data = merge(my_data, Seq_724, by = 'gene_id')
rm(Seq_724, Seq_802, Seq_809)

# Remove those extraneous rows from HTSeq
my_data = tail(my_data, -5)

# Set the index
rownames(my_data) = my_data$gene_id

# Get rid of gene name column because it is now the index
my_data = subset(my_data, select = -c(gene_id))

# Remove the rows that are completely 0s, uninformative rows
my_data = my_data[apply(my_data, 1, function(x) {
  !all(x == 0)}), ]
```

Now we will clean our data. We have previously imported the data and performed minimal filtering, we will now add labels to the data.

```
#-----------------------------------
# Add two empty rows at the end of the data frame to be filled with the
  # experimental factors that we later plug in.
```

```r
my_data[(nrow(my_data) + 1):(nrow(my_data) + 3), ] = NA

# Loop through the columns and assign experimental factors based on the
# sample names, filling the last two rows
# grepl returns a logical vector
columns = colnames(my_data)
for (i in 1:ncol(my_data)) {


#-----------------------------------
# RULES FOR: 802 and 724 groupings
#-----------------------------------
  if (grepl("W_C_*", colnames(my_data)[i])) {
    #my_data$Transformed = 'Water_Control'
    #my_data$Transformed = 'Water_Control'
    my_data[(nrow(my_data) - 2),i] <- "Water_Control"
    my_data[(nrow(my_data) - 1), i] <- "Water"

    # GA and CK
  } else if (grepl("GA3_CK_C_*", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "GA_CK_Control"
    my_data[(nrow(my_data) - 1), i] <- "GA_CK"
  } else if (grepl("GA3_[^CK_C]", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "GA_Treatment"
    my_data[(nrow(my_data) - 1), i] <- "GA"

    # AUX
  } else if (grepl("AUX_C_", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "AUX_Control"
    my_data[(nrow(my_data) - 1), i] <- "AUX"
  } else if (grepl("AUX_[^C]", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "AUX_Treatment"
    my_data[(nrow(my_data) - 1), i] <- "AUX"

    # CK
  } else if (grepl("^CK_", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "CK_Treatment"
    my_data[(nrow(my_data) - 1), i] <- "CK"
  }


#-----------------------------------
# RULES FOR: 809 grouping
#-----------------------------------
```

```
    # SB
    else if (grepl("SB\\.C[^K](.*?)_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Untransformed"
    my_data[(nrow(my_data) - 1), i] <- "Control_Region"
    my_data[(nrow(my_data)), i] <- "No_Gall"
  } else if (grepl("SB\\.CK(.*?)G", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Untransformed"
    my_data[(nrow(my_data) - 1), i] <- "Control_Region"
    my_data[(nrow(my_data)), i] <- "Gall"

    # DPR
  } else if (grepl("DPR(.*?)G_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Untransformed"
    my_data[(nrow(my_data) - 1), i] <- "Empty_Vector"
    my_data[(nrow(my_data)), i] <- "Gall"
  } else if (grepl("DPR(.*?)[^G]_S*", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Untransformed"
    my_data[(nrow(my_data) - 1), i] <- "Empty_Vector"
    my_data[(nrow(my_data)), i] <- "No_Gall"

    # WUS Reg 1
  } else if (grepl("X1(.*?)G_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Transformed"
    my_data[(nrow(my_data) - 1), i] <- "WUS_Reg_1"
    my_data[(nrow(my_data)), i] <- "Gall"
  } else if (grepl("X1(.*?)[^G]_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Transformed"
    my_data[(nrow(my_data) - 1), i] <- "WUS_Reg_1"
    my_data[(nrow(my_data)), i] <- "No_Gall"

    # WUS Reg 2
  } else if (grepl("X2(.*?)G_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Transformed"
    my_data[(nrow(my_data) - 1), i] <- "WUS_Reg_2"
    my_data[(nrow(my_data)), i] <- "Gall"
  } else if (grepl("X2(.*?)[^G]_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Transformed"
    my_data[(nrow(my_data) - 1), i] <- "WUS_Reg_2"
    my_data[(nrow(my_data)), i] <- "No_Gall"

    # LFY Reg 1
  } else if (grepl("X3(.*?)G_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Transformed"
    my_data[(nrow(my_data) - 1), i] <- "LFY_Reg_1"
    my_data[(nrow(my_data)), i] <- "Gall"
  } else if (grepl("X3(.*?)[^G]_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Transformed"
    my_data[(nrow(my_data) - 1), i] <- "LFY_Reg_1"
    my_data[(nrow(my_data)), i] <- "No_Gall"

    # LFY Reg 2
  } else if (grepl("X4(.*?)G_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Transformed"
    my_data[(nrow(my_data) - 1), i] <- "LFY_Reg_2"
    my_data[(nrow(my_data)), i] <- "Gall"
  } else if (grepl("X4(.*?)[^G]_S", colnames(my_data)[i])) {
    my_data[(nrow(my_data) - 2),i] <- "Transformed"
    my_data[(nrow(my_data) - 1), i] <- "LFY_Reg_2"
    my_data[(nrow(my_data)), i] <- "No_Gall"
  }
}


# Update rows with "metadata" on each sample's identity
row.names(my_data)[(nrow(my_data) - 2) : (nrow(my_data))] = c("Transformation", "Knockou
Complete_Set = my_data
rm(my_data)
```

Transformed_v_Untransformed

Now that we have the complete set of labeled data, we will begin performing pairwise comparisons on groups. This makes it necessary to perform some subsetting for each grouping. First we will work on the Transformed vs. Untransformed.

```
# Switch to output folder
setwd('/home/scott/Documents/Uni/Research/Projects/Grape_RNA/Output')
#-----------------------------------
# Transformed vs. Untransformed
# WUS1 and WUS2 vs SB-C
WUS_Transformed_V_Untransformed = function(Complete_Set) {
  # Subset the Data
  "X2(.*?)[^G]_S"
  WUS_Reg1_No_Gall =  select(Complete_Set, matches("X1(.*?)[^G]_S"))
  WUS_Reg2_No_Gall =  select(Complete_Set, matches("X2(.*?)[^G]_S"))
  SB_C = select(Complete_Set, matches("SB\\.C[^K](.*?)_S"))
  WUS_Comp_List = list(WUS_Reg1_No_Gall, WUS_Reg2_No_Gall, SB_C)
  # Return list of subsetted data
```

```r
  return(WUS_Comp_List)
}
WUS_Transformed_V_Untransformed = WUS_Transformed_V_Untransformed(Complete_Set)
WUS_Reg1_No_Gall = WUS_Transformed_V_Untransformed[[1]]
WUS_Reg2_No_Gall = WUS_Transformed_V_Untransformed[[2]] #DNE due to lack of data input
SB_C = WUS_Transformed_V_Untransformed[[3]]

# Specify how to merge
Counts = merge(WUS_Reg1_No_Gall, WUS_Reg2_No_Gall, by = 'row.names')
# Give the 3rd data set a Row.names column
SB_C$Row.names = rownames(SB_C)
# Merge the 3rd into the already combined two
Counts = merge(Counts, SB_C, by = 'Row.names')
rm(SB_C, WUS_Reg1_No_Gall, WUS_Reg2_No_Gall)

# Make grouping for treatment groups.
Counts = Counts[-(1:2),]
rownames(Counts) <- Counts$Row.names
Counts = subset(Counts, select = -c(Row.names))
my_grouping = c(tail(Counts, 1)) # transformed vs untransformed grouping declared

Counts = head(Counts, -1) # drop Transformed designation

Counts = as.matrix.data.frame(Counts)
x = rownames(Counts)
# We need the gene names for later
Gene_Row_Key = data.frame("Num"=rownames(as.data.frame(x)),"Gene"=x)
# Write the table if you want.
#write.table(Gene_Row_Key, file = 'Trans_v_Untrans_WUS_SBC_Gene_Row_Key.tsv', sep='\t'
rm(x)

Counts = apply(Counts, 2, as.numeric)
D = DGEList(Counts, group = unlist(my_grouping))
D = calcNormFactors(D)
D_Samples = D$samples

D = estimateCommonDisp(D)
D = estimateTagwiseDisp(D)
Fish_Exact = exactTest(D)
topTags = topTags(Fish_Exact)

# P = 0.05
simplified_DGE = decideTestsDGE(Fish_Exact, p=0.05, adjust="BH")
```

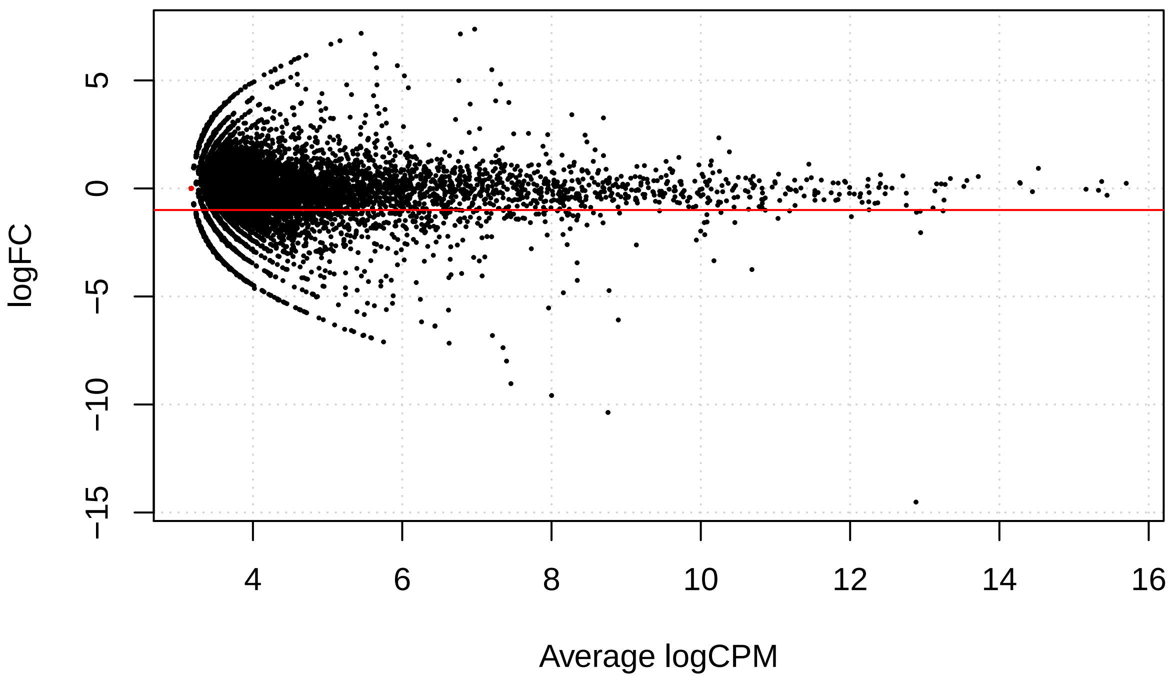
```
# Write summary table
write.table(summary(simplified_DGE), file = 'Trans_v_Untrans_WUS_SBC_Summary.txt', quote

# Write direction of differntial expression table
simplified_DGE_frame = data.frame(simplified_DGE)
colnames(simplified_DGE_frame) = 'Direction_Differentially_Regulated'
row.names(simplified_DGE_frame) = Gene_Row_Key$Gene
Trans_v_Untrans_Wus_SBC_Direction = as_tibble(rownames_to_column(simplified_DGE_frame, v
write_tsv(Trans_v_Untrans_Wus_SBC_Direction, 'Trans_v_Untrans_WUS_SBC_Direction.tsv')

# Write full output
row.names(Fish_Exact) = Gene_Row_Key$Gene
Fish_tbl = as_tibble(rownames_to_column(Fish_Exact$table, var = 'Gene_Name'))
write_tsv(Fish_tbl, 'Trans_v_Untrans_WUS_SBC_Full.tsv')

# Clear workspace
rm(Gene_Row_Key, D, D_Samples, Fish_Exact, my_grouping, Counts, topTags, WUS_Transformed
```

Transformed_v_Untransformed_V2

Now we will work on the other section of untransformed data

```
# Switch to output folder
setwd('/home/scott/Documents/Uni/Research/Projects/Grape_RNA/Output')
#-----------------------------------
# Transformed vs. Untransformed
# WUS1 and WUS2 vs SB-C
LFY_Transformed_V_Untransformed_No_Gall = function(Complete_Set) {
  # Subset the Data
  LFY_Reg1_No_Gall =  select(Complete_Set, matches("X3(.*?)[^G]_S"))
  LFY_Reg2_No_Gall =  select(Complete_Set, matches("X4(.*?)[^G]_S"))
  SB_C = select(Complete_Set, matches("SB\\.C[^K](.*?)_S"))
  LFY_Comp_List = list(LFY_Reg1_No_Gall, LFY_Reg2_No_Gall, SB_C)
  # Return list of subsetted data
  return(LFY_Comp_List)
}
LFY_Transformed_V_Untransformed_No_Gall = LFY_Transformed_V_Untransformed_No_Gall(Compl
LFY_Reg1_No_Gall = LFY_Transformed_V_Untransformed_No_Gall[[1]]
LFY_Reg2_No_Gall = LFY_Transformed_V_Untransformed_No_Gall[[2]]
SB_C = LFY_Transformed_V_Untransformed_No_Gall[[3]]

#Counts = merge.data.frame(SB_C, LFY_Reg1_No_Gall, by=0) # merge the relevant data
#Counts = merge(LFY_Reg1_No_Gall, LFY_Reg2_No_Gall)
```

```r
# Specify how to merge
Counts = merge(LFY_Reg1_No_Gall, LFY_Reg2_No_Gall, by = 'row.names')
# Give the 3rd data set a Row.names column
SB_C$Row.names = rownames(SB_C)
# Merge the 3rd into the already combined two
Counts = merge(Counts, SB_C, by = 'Row.names')
rm(SB_C, LFY_Reg1_No_Gall, LFY_Reg2_No_Gall)

# Make grouping for treatment groups.
rownames(Counts) <- Counts$Row.names
Counts = subset(Counts, select = -c(Row.names))
my_grouping = c(tail(Counts, 1)) # transformed vs untransformed grouping declared)
Counts = Counts[-(1:2),]
Counts = head(Counts, -1) # drop Transformed designation

Counts = as.matrix.data.frame(Counts)
x = rownames(Counts)
# We need the gene names for later
Gene_Row_Key = data.frame("Num"=rownames(as.data.frame(x)),"Gene"=x)
# Write the table if you want.
#write.table(Gene_Row_Key, file = 'Trans_v_Untrans_LFY_SBC_Gene_Row_Key.tsv', sep='\t'
rm(x)

Counts = apply(Counts, 2, as.numeric)
D = DGEList(Counts, group = unlist(my_grouping))
D = calcNormFactors(D)
D_Samples = D$samples

D = estimateCommonDisp(D)
D = estimateTagwiseDisp(D)
Fish_Exact = exactTest(D)
topTags = topTags(Fish_Exact)

# P = 0.05
simplified_DGE = decideTestsDGE(Fish_Exact, p=0.05, adjust="BH")
my_test = rownames(Fish_Exact)[as.logical(simplified_DGE)]
plotSmear(Fish_Exact, de.tags=simplified_DGE)
abline(h=c(-1,10), col=2)
```



```
# Write summary table
write.table(summary(simplified_DGE), file = 'Trans_v_Untrans_LFY_SBC_Summary.txt', quote

# Write direction of differntial expression table
simplified_DGE_frame = data.frame(simplified_DGE)
colnames(simplified_DGE_frame) = 'Direction_Differentially_Regulated'
row.names(simplified_DGE_frame) = Gene_Row_Key$Gene
Trans_v_Untrans_Wus_SBC_Direction = as_tibble(rownames_to_column(simplified_DGE_frame, v
write_tsv(Trans_v_Untrans_Wus_SBC_Direction, 'Trans_v_Untrans_LFY_SBC_Direction.tsv')

# Write full output
row.names(Fish_Exact) = Gene_Row_Key$Gene
Fish_tbl = as_tibble(rownames_to_column(Fish_Exact$table, var = 'Gene_Name'))
write_tsv(Fish_tbl, 'Trans_v_Untrans_LFY_SBC_Full.tsv')

# Clear workspace
rm(Gene_Row_Key, D, D_Samples, Fish_Exact, my_grouping, Counts, topTags, LFY_Transformed
```

Now we are going to work on comparing SB-C within itself

```
# Switch to output folder
setwd('/home/scott/Documents/Uni/Research/Projects/Grape_RNA/Output')
#-----------------------------------
# SB_C Within Self
# SB-C-x vs SB-CK-x-G
```

```
SBC_Comparisons = function(Complete_Set) {
  # Subset the Data
  SB_CKG = select(Complete_Set, matches("SB\\.CK(.*?)G"))
  SB_C = select(Complete_Set, matches("SB\\.C[^K](.*?)_S"))
  SBC_Comp_List = list(SB_CKG, SB_C)
  # Return list of subsetted data
  return(SBC_Comp_List)
}
SBC_Comparisons = SBC_Comparisons(Complete_Set)
SBCKG_Unt_With_Gall = SBC_Comparisons[[1]]
SBC_Unt_No_Gall = SBC_Comparisons[[2]]

Counts = merge.data.frame(SBCKG_Unt_With_Gall, SBC_Unt_No_Gall, by=0) # merge relevant
rm(SBC_Unt_No_Gall, SBCKG_Unt_With_Gall)

# Make grouping for treatment groups.
rownames(Counts) <- Counts$Row.names
Counts = subset(Counts, select = -c(Row.names))
my_grouping = c(head(Counts, 1)) # Gall grouping declared for later
Counts = Counts[-(1:2),] # drop first two rows
Counts = head(Counts, -1) # drop Transformed designation

Counts = as.matrix.data.frame(Counts)
x = rownames(Counts)
Gene_Row_Key = data.frame("Num"=rownames(as.data.frame(x)),"Gene"=x)
# Decoder to match back the gene names to the numbers
write.table(Gene_Row_Key, file = 'Gene_Row_Key_SBC.tsv', sep='\t', quote=FALSE, row.name
rm(x)

Counts = apply(Counts, 2, as.numeric)
D = DGEList(Counts, group = unlist(my_grouping))
D = calcNormFactors(D)
D_Samples = D$samples
D = estimateCommonDisp(D)
D = estimateTagwiseDisp(D)
Fish_Exact = exactTest(D)
topTags = topTags(Fish_Exact)


# P = 0.05
simplified_DGE = decideTestsDGE(Fish_Exact, p=0.05, adjust="BH")

# Write summary table
write.table(summary(simplified_DGE), file = 'Trans_v_Untrans_LFY_SBC_Summary.txt', quote
# Write direction of differntial expression table
simplified_DGE_frame = data.frame(simplified_DGE)
colnames(simplified_DGE_frame) = 'Direction_Differentially_Regulated'
row.names(simplified_DGE_frame) = Gene_Row_Key$Gene
Trans_v_Untrans_Wus_SBC_Direction = as_tibble(rownames_to_column(simplified_DGE_frame, v
write_tsv(Trans_v_Untrans_Wus_SBC_Direction, 'Trans_v_Untrans_LFY_SBC_Direction.tsv')

# Write full output
row.names(Fish_Exact) = Gene_Row_Key$Gene
Fish_tbl = as_tibble(rownames_to_column(Fish_Exact$table, var = 'Gene_Name'))
write_tsv(Fish_tbl, 'Trans_v_Untrans_LFY_SBC_Full.tsv')







# Clear workspace
rm(Gene_Row_Key, D, D_Samples, Fish_Exact, my_grouping, Counts, topTags, SBC_Comparisons
```

Below we are going to examine the DPR grouping. These are the empty vector controls with and without galls.

NOTE: this comparison cannot be performed as there is no DPR no gall (DPR-x) sample. I will verify later.

```
# Switch to output folder
setwd('/home/scott/Documents/Uni/Research/Projects/Grape_RNA/Output')
#-----------------------------------
# DPR Within Self
# DPR-x vs DPR-x-G
DPR_Comparisons = function(Complete_Set) {
  # Subset the Data
  DPR_EmptVec_With_Gall = select(Complete_Set, matches("DPR(.*?)G_S"))
  DPR_EmptVec_No_Gall= select(Complete_Set, matches("DPR(.*?)[^G]_S*"))
  DPR_Comp_List = list(DPR_EmptVec_With_Gall, DPR_EmptVec_No_Gall)
  # Return list of subsetted data
  return(DPR_Comp_List)
}

DPR_Comparisons = DPR_Comparisons(Complete_Set)
DPR_EmptVec_With_Gall = DPR_Comparisons[[1]]
DPR_EmptVec_No_Gall = DPR_Comparisons[[2]]
```

```
Counts = merge.data.frame(DPR_EmptVec_With_Gall, DPR_EmptVec_No_Gall, by=0) # merge rel
rm(DPR_EmptVec_With_Gall, DPR_EmptVec_No_Gall)

# Make grouping for treatment groups.
rownames(Counts) <- Counts$Row.names
Counts = subset(Counts, select = -c(Row.names))
my_grouping = c(head(Counts, 1)) # Gall grouping declared for later
Counts = Counts[-(1:2),] # drop first two rows (Gall and Vector/Region)
Counts = head(Counts, -1) # drop Transformed designation

Counts = as.matrix.data.frame(Counts)
x = rownames(Counts)
Gene_Row_Key = data.frame("Num"=rownames(as.data.frame(x)),"Gene"=x)
# Decoder to match back the gene names to the numbers
write.table(Gene_Row_Key, file = 'Gene_Row_Key_DPR.tsv', sep='\t', quote=FALSE, row.name
rm(x)

Counts = apply(Counts, 2, as.numeric)
D = DGEList(Counts, group = unlist(my_grouping))
D = calcNormFactors(D)
D_Samples = D$samples
D = estimateCommonDisp(D)
D = estimateTagwiseDisp(D)
Fish_Exact = exactTest(D)
topTags = topTags(Fish_Exact)

# Write to file
write.table(Fish_Exact$table, file="DPR_Output.txt",quote=FALSE,sep="\t",row.names=TRUE)
# Clear workspace
rm(Gene_Row_Key, D, D_Samples, Fish_Exact, my_grouping, Counts, topTags, DPR_Comparisons
```

#TODO ## Below we are going to examine the grouping. These are the empty vector controls with and without galls. NOTE: this comparison cannot be performed as there is no DPR no gall (DPR-x) sample. I will verify later.

```
# Switch to output folder
setwd('/home/scott/Documents/Uni/Research/Projects/Grape_RNA/Output')
#------------------------------------
# DPR Within Self
# DPR-x vs DPR-x-G
DPR_Comparisons = function(Complete_Set) {
  # Subset the Data
  DPR_EmptVec_With_Gall = select(Complete_Set, matches("DPR(.*?)G_S"))
  DPR_EmptVec_No_Gall= select(Complete_Set, matches("DPR(.*?)[^G]_S*"))
  DPR_Comp_List = list(DPR_EmptVec_With_Gall, DPR_EmptVec_No_Gall)
  # Return list of subsetted data
  return(DPR_Comp_List)
}

DPR_Comparisons = DPR_Comparisons(Complete_Set)
DPR_EmptVec_With_Gall = DPR_Comparisons[[1]]
DPR_EmptVec_No_Gall = DPR_Comparisons[[2]]

Counts = merge.data.frame(DPR_EmptVec_With_Gall, DPR_EmptVec_No_Gall, by=0) # merge rel
rm(DPR_EmptVec_With_Gall, DPR_EmptVec_No_Gall)

# Make grouping for treatment groups.
rownames(Counts) <- Counts$Row.names
Counts = subset(Counts, select = -c(Row.names))
my_grouping = c(head(Counts, 1)) # Gall grouping declared for later
Counts = Counts[-(1:2),] # drop first two rows (Gall and Vector/Region)
Counts = head(Counts, -1) # drop Transformed designation

Counts = as.matrix.data.frame(Counts)
x = rownames(Counts)
Gene_Row_Key = data.frame("Num"=rownames(as.data.frame(x)),"Gene"=x)
# Decoder to match back the gene names to the numbers
write.table(Gene_Row_Key, file = 'Gene_Row_Key_DPR.tsv', sep='\t', quote=FALSE, row.name
rm(x)

Counts = apply(Counts, 2, as.numeric)
D = DGEList(Counts, group = unlist(my_grouping))
D = calcNormFactors(D)
D_Samples = D$samples
D = estimateCommonDisp(D)
D = estimateTagwiseDisp(D)
Fish_Exact = exactTest(D)
topTags = topTags(Fish_Exact)

# Write to file
write.table(Fish_Exact$table, file="DPR_Output.txt",quote=FALSE,sep="\t",row.names=TRUE)
# Clear workspace
rm(Gene_Row_Key, D, D_Samples, Fish_Exact, my_grouping, Counts, topTags, DPR_Comparisons
```