```
library(readxl)
library(limma)
library(BiocGenerics)
library(pheatmap)
library(vegan)
library(rgl)
library(ape)
library(dendextend)
library(adegenet)
library(viridis)
library(edgeR)
library(gplots)
library(RColorBrewer)

setwd("~/Dropbox/Infection_Tagseq/")

#Setting up a list of my samples, variables...etc to put into a single
table.
data = read.table("final_matrix.txt", header=T, row.names=1, com='')
colnames(data) <-
c("GL10","GL10","GL11","GL11","GL12","GL12","GL13","GL13","GL14","GL14
","GL15","GL15","GL16","GL16","GL17","GL17","GL18","GL18","GL19","GL19
","GL1","GL1","GL21","GL21","GL23","GL23","GL2","GL2","GL3","GL3","GL5
","GL5","GL6","GL6","GL7","GL7","GL8","GL8","GL9","GL9","GV11","GV11",
"GV12","GV12","GV13","GV13","GV14","GV14","GV15","GV15","GV16","GV16",
"GV17","GV17","GV18","GV18","GV1","GV1","GV21","GV21","GV22","GV22","G
V24","GV24","GV25","GV25","GV2","GV2","GV3","GV3","GV4","GV4","GV5","G
V5","GV7","GV7","GV8","GV8","GV9","GV9","LL10","LL10","LL11","LL11","L
L12","LL12","LL13","LL13","LL14","LL14","LL15","LL15","LL16","LL16","L
L17","LL17","LL18","LL18","LL19","LL19","LL1","LL1","LL20","LL20","LL2
","LL2","LL3","LL3","LL4","LL4","LL5","LL5","LL6","LL6","LL7","LL7","L
L8","LL8","LL9","LL9","LV10","LV10","LV11","LV11","LV12","LV12","LV13"
,"LV13","LV14","LV14","LV15","LV15","LV16","LV16","LV17","LV17","LV18"
,"LV18","LV19","LV19","LV1","LV1","LV20","LV20","LV21","LV21","LV22","
LV22","LV24","LV24","LV25","LV25","LV5","LV5","LV6","LV6","LV7","LV7",
"LV9","LV9")
data <- as.data.frame(data)
data2 <- sumTechReps(data)

#Now make a new dataframe from the dataset so that you do not have to
reload the original data later if needed
rnaseqMatrix = as.matrix(data2)
head(rnaseqMatrix)

#bring in the metadata
Oyster_Data_condensed <-
read_excel("Oyster_Dermo_and_Weight_Data.xlsx")
Oyster_Data_condensed <- as.data.frame(Oyster_Data_condensed)
head(Oyster_Data_condensed)
Samples_to_keep <- Oyster_Data_condensed[,c(1)]
```

```
head(Samples_to_keep)

Samples_with_data <- subset(rnaseqMatrix, select=Samples_to_keep)
dim(Samples_with_data)
#[1] 38838    40
head(Samples_with_data)
#Now make this into a Differential Gene Expresion List
head(Oyster_Data_condensed)
Group <- factor(paste(Oyster_Data_condensed$`Infection_Group (low
infection A, mild infection B, high infection C)`))
cbind(Oyster_Data_condensed,Group=Group)

y <- DGEList(counts=Samples_with_data, group=Group,remove.zeros=TRUE)

keep <- rowSums(cpm(y) > 3 ) >= 20
table(keep)
y <- y[keep,, keep.lib.sizes=FALSE]
# 17,439 genes retained for DE analysis

#setting keep.lib.sizes to FALSE requires library sizes to be
recalculated after filtering
y <- calcNormFactors(y,method = "TMM")
logCPM <- cpm(y,log=T,prior.count = 2)

dds.pcoa=pcoa(vegdist(t(logCPM),method="euclidean")/1000)
scores=dds.pcoa$vectors
# Calculate the percent of variance explained by first two axes
percent <- dds.pcoa$values$Eigenvalues
cumulative_percent_variance <- (percent / sum(percent))*100
cumulative_percent_variance
color2 <- c("skyblue","goldenrod","coral","firebrick")
Infection <- as.factor(Oyster_Data_condensed$`Infection_Group (low
infection A, mild infection B, high infection C)`)

color2[Infection]
par(mfrow=c(1,1))
par(mar=c(5,5,5,5))
pch= c(19)
plot(scores[,1], scores[,2],cex=3,cex.axis=2,cex.lab = 2,type="p",pch=
19,col = color2[Infection],  xlab=paste("PC1, ",
round(cumulative_percent_variance[1], 2), "%"), ylab=paste("PC2, ",
round(cumulative_percent_variance[2], 2), "%"))
text(scores[,1],scores[,2], labels=Oyster_Data_condensed$Sample2,
cex=1, font=2)
ordibar(scores,as.factor(Oyster_Data_condensed$`Infection_Group (low
infection A, mild infection B, high infection C)`),label=T, cex=.
75,col = "grey30")
ordiellipse(scores,as.factor(Oyster_Data_condensed$`Infection_Group
(low infection A, mild infection B, high infection C)`),label=T,
cex=3,col = "grey30")
```

```
adonis2(t(logCPM)~Oyster_Data_condensed$`count/
g`,data=Oyster_Data_condensed,method="euclidean",permutations =
1000000)

#Permutation test for adonis under reduced model
#Terms added sequentially (first to last)
#Permutation: free
#Number of permutations: 1e+06
#
#adonis2(formula = t(logCPM) ~ Oyster_Data_condensed$`count/g`, data =
Oyster_Data_condensed, permutations = 1e+06, method = "euclidean")
#Df SumOfSqs      R2      F  Pr(>F)
#Oyster_Data_condensed$`count/g`  1    12263 0.03272 1.2856 0.09098 .
#Residual                        38   362461 0.96728
#Total                           39   374723 1.00000
#---
#  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

CV_Gene_Annotations <- read.delim("CV_Gene_Annotations.txt")
row.names(CV_Gene_Annotations) <- CV_Gene_Annotations$CV_gene
CV_Gene_Annotations2 <-
merge(CV_Gene_Annotations,y$counts,by="row.names")
head(CV_Gene_Annotations2)
CV_Gene_MWU_ref <- CV_Gene_Annotations2[,c(1,12)]
head(CV_Gene_MWU_ref)
#write.table(CV_Gene_MWU_ref,"Infection_GO/
CV_Gene_MWU_ref.tab",sep="\t",quote=F,row.names=F)

design <- model.matrix(~0 + Group)
colnames(design) <- levels(Group)
y <- estimateDisp(y, design, robust=TRUE)
fit <- glmFit(y, design, robust=TRUE)

setwd("~/Dropbox/Infection_Tagseq/Dermo_edgeR/")
#save(data2,y,design,fit,Oyster_Data_condensed,CV_Gene_Annotations,Gro
up,file = "Dermo_dataInput.RData")
#ln = load("Dermo_dataInput.RData")
colnames(fit)
con.1 <- makeContrasts(D_A = D - A,levels=design)

con.2 <- makeContrasts(C_A = C - A,levels=design)

con.3 <- makeContrasts(B_A = B - A,levels=design)


res_1 <- glmLRT(fit, contrast=con.1)

is.de_1 <- decideTestsDGE(res_1, adjust.method="BH", p.value=0.05)
summary(is.de_1)
```

```
#       -1*A 1*D
#Down                                8
#NotSig                          17400
#Up                                 31

logCPM <- cpm(y, prior.count=1, log=TRUE)


o <- order(res_1$table$PValue)
logCPM_res_1 <- logCPM[o[1:100],]
colnames(logCPM_res_1)
logCPM_res_1 <- logCPM_res_1[,c(21:40)]
#scale each row (each gene) to have mean zero and standard deviation
one:
dim(logCPM_res_1)
logCPM_res1 <- t(scale(t(logCPM_res_1)))
col.pan <- colorpanel(256, "blue", "white", "red")
##We can have the top bar annotated with colors that are assocaited
with the levels in Group
##Here LC samples are coded  with dark blue for GI and dark green for
LUMCON
##And VB samples are coded with light blue for GI and light green for
LUMCON
t <-heatmap.2(logCPM_res1, col=col.pan, Rowv=TRUE, scale="row",
              trace="none", dendrogram="both", labRow = FALSE,
cexCol=1,
              main = "ANODEV Population Sig Genes (n=353)")

res_1_GO_MWU <- res_1$table
res_1_GO_MWU$negPval <- -log(res_1_GO_MWU$PValue)
res_1_GO_MWU$direction <- ifelse(res_1_GO_MWU$logFC >=0,1,-1)
res_1_GO_MWU$signPval <- res_1_GO_MWU$direction * res_1_GO_MWU$negPval
res_1_GO_MWU$gene <- row.names(res_1_GO_MWU)
colnames(res_1_GO_MWU)
res_1_GO_MWU <- res_1_GO_MWU[,c(8,7)]
#write.csv(res_1_GO_MWU,"Infection_GO/
Calcasieu_for_GOMWU_DA.csv",quote=F,row.names=F)
res_1_sig <- topTags(res_1,n=Inf,adjust.method = "BH",p.value = 0.05)
res_1_sig <- res_1_sig$table
rest_1_sig_annot <-
merge(res_1_sig,CV_Gene_Annotations,by="row.names")


res_2 <- glmLRT(fit, contrast=con.2)

is.de_2 <- decideTestsDGE(res_2, adjust.method="BH", p.value=0.05)
summary(is.de_2)
#       -1*A 1*C
#Down                                0
#NotSig                          17438
```

```
#Up                                   1
logCPM <- cpm(y, prior.count=1, log=TRUE)


o <- order(res_2$table$PValue)
logCPM_res2 <- logCPM[o[1:200],]
logCPM_res2 <- logCPM_res2[,c(1:20)]

#scale each row (each gene) to have mean zero and standard deviation
one:
logCPM_res2 <- t(scale(t(logCPM_res2)))
col.pan <- colorpanel(256, "blue", "white", "red")
##We can have the top bar annotated with colors that are assocaited
with the levels in Group
##Here LC samples are coded  with dark blue for GI and dark green for
LUMCON
##And VB samples are coded with light blue for GI and light green for
LUMCON
t <-heatmap.2(logCPM_res2, col=col.pan, Rowv=TRUE, scale="row",
              trace="none", dendrogram="both", labRow = FALSE,
cexCol=1,
              main = "ANODEV Population Sig Genes (n=353)")

res_2_GO_MWU <- res_2$table
res_2_GO_MWU$negPval <- -log(res_2_GO_MWU$PValue)
res_2_GO_MWU$direction <- ifelse(res_2_GO_MWU$logFC >=0,1,-1)
res_2_GO_MWU$signPval <- res_2_GO_MWU$direction * res_2_GO_MWU$negPval
res_2_GO_MWU$gene <- row.names(res_2_GO_MWU)

res_2_GO_MWU <- res_2_GO_MWU[,c(8,7)]
head(res_2_GO_MWU)
#write.csv(res_2_GO_MWU,"Infection_GO/
Vermillion_for_GOMWU_DA.csv",quote=F,row.names=F)

res_3 <- glmLRT(fit, contrast=con.3)

is.de_3 <- decideTestsDGE(res_3, adjust.method="BH", p.value=0.05)
summary(is.de_3)
#-1*A 1*B
#Down                                  1
#NotSig                            17437
#Up                                    1



res_3_sig <- topTags(res_3,n=Inf,adjust.method = "BH",p.value = 0.05)
res_3_sig <- res_3_sig$table
rest_3_sig_annot <-
merge(res_3_sig,CV_Gene_Annotations,by="row.names")
```

```
res_3_GO_MWU <- res_3$table
res_3_GO_MWU$negPval <- -log(res_3_GO_MWU$PValue)
res_3_GO_MWU$direction <- ifelse(res_3_GO_MWU$logFC >=0,1,-1)
res_3_GO_MWU$signPval <- res_3_GO_MWU$direction * res_3_GO_MWU$negPval
res_3_GO_MWU$gene <- row.names(res_3_GO_MWU)

res_3_GO_MWU <- res_3_GO_MWU[,c(8,7)]
head(res_3_GO_MWU)
#write.csv(res_3_GO_MWU,"Infection_for_GOMWU_DA.csv",quote=F,row.names
=F)


EdgeR_results <- cbind(is.de_1,is.de_2,is.de_3,is.de_4)
EdgeR_results <- as.data.frame(EdgeR_results)
colnames(EdgeR_results) <- c("DA","CA","BA","DB")
EdgeR_results$DE <- (abs(EdgeR_results$DA) +
                       abs(EdgeR_results$CA) +
                       abs(EdgeR_results$BA))

par(mfrow=c(1,2))
vennDiagram(EdgeR_results[,1:3],
circle.col=c("salmon","turquoise"),include=c("up","down"),main =
"Shared DEG's",)
vennDiagram(EdgeR_results[,1:3],
circle.col=c("salmon","turquoise"),include=c("both"),main = "Shared
DEG's",)

Is.DE <- EdgeR_results$DE > 0
logCPM <- cpm(y,log=T,prior.count=2)
logCPM_DE <- logCPM[Is.DE,]
logCPM_DE <- merge(logCPM_DE,CV_Gene_Annotations,by = "row.names")
row.names(logCPM_DE) <- logCPM_DE$Row.names
logCPM_DE$Row.names = NULL
colnames(logCPM_DE)
logCPM_DE2 <- t(scale(t(logCPM_DE[,c(1:40)])))
dim(logCPM_DE2)
colnames(logCPM_DE2) <- Oyster_Data_condensed$Sample2

colors <- c("skyblue","dodgerblue","coral","red")
Oyster_Data_condensed$`Infection_Group (low infection A, mild
infection B, high infection C)`
colors[as.factor(Oyster_Data_condensed$`Infection_Group (low infection
A, mild infection B, high infection C)`)]
col.pan <- colorpanel(256, "blue", "white", "red")
t <-heatmap.2(logCPM_DE2,density.info = c("density"),denscol=
"black" , col=col.pan, Rowv=TRUE, scale="row",ColSideColors =
colors[as.factor(Oyster_Data_condensed$`Infection_Group (low infection
A, mild infection B, high infection C)`)],
              trace="none", dendrogram="both",cexRow = 1.25, labRow =
logCPM_DE$Protein.Name, cexCol=1.25,margins=c(8,42), labCol =
```

```
Oyster_Data_condensed$`Infection_Group (low infection A, mild
infection B, high infection C)`,
              main = "Differential gene expression (n=47)")


res_1_nonsig <- topTags(res_3,n=Inf,adjust.method = "BH")
res_2_nonsig <- topTags(res_3,n=Inf,adjust.method = "BH")
res_3_nonsig <- topTags(res_3,n=Inf,adjust.method = "BH")
res_4_nonsig <- topTags(res_3,n=Inf,adjust.method = "BH")

EdgeR_results2 <-
cbind(res_1_nonsig$table,res_2_nonsig$table,res_3_nonsig$table,res_4_n
onsig$table)
colnames(EdgeR_results2) <-
c("DA_logFC","logCPM","LR","DA_PValue","DA_FDR",

"CA_logFC","CA_logCPM","CA_LR","CA_PValue","CA_FDR",

"BA_logFC","BA_logCPM","BA_LR","BA_PValue","BA_FDR",

"DB_logFC","DB_logCPM","DB_LR","DB_PValue","DB_FDR")


head(EdgeR_results2)
EdgeR_results3 <-
EdgeR_results2[,c(2,3,1,4,5,6,9,10,11,14,15,16,19,20)]
colnames(EdgeR_results3)
colnames(EdgeR_results) <-
c("DA_sig","CA_sig","BA_sig","DB_sig","All_DEGs")
EdgeR_results4 <- cbind(EdgeR_results3,EdgeR_results)
head(EdgeR_results4)


#Combine methylation and expression

dim(Infection_nonsig_gene_reduced2)
#8383
EdgeR_results4$ID <- row.names(EdgeR_results4)
dim(EdgeR_results4)
#17439
Expression_and_methylation <-
merge(Infection_nonsig_annotated2,EdgeR_results4,by="ID")
dim(Expression_and_methylation)
#5405
table(Expression_and_methylation$All_DEGs)
EandM_sig <-
Expression_and_methylation[c(Expression_and_methylation$All_DEGs >0),]
colnames(CV_Gene_Annotations2)[1] <- c("ID")
EandM_sig2 <- merge(EandM_sig,CV_Gene_Annotations2,by="ID")
```

```
getwd()
setwd("/Users/Kevin_Work_Mac/Dropbox/Infection_Tagseq/Infection_GO//")
# Edit these to match your data file names:


#MWU test from https://github.com/z0on/GO_MWU
# GO_MWU uses continuous measure of significance (such as fold-change
or -log(p-value) ) to identify GO categories that are significantly
enriches with either up- or down-regulated genes. The advantage - no
need to impose arbitrary significance cutoff.

# If the measure is binary (0 or 1) the script will perform a typical
"GO enrichment" analysis based Fisher's exact test: it will show GO
categories over-represented among the genes that have 1 as their
measure.

# On the plot, different fonts are used to indicate significance and
color indicates enrichment with either up (red) or down (blue)
regulated genes. No colors are shown for binary measure analysis.

# The tree on the plot is hierarchical clustering of GO categories
based on shared genes. Categories with no branch length between them
are subsets of each other.

# The fraction next to GO category name indicates the fracton of
"good" genes in it; "good" genes being the ones exceeding the
arbitrary absValue cutoff (option in gomwuPlot). For Fisher's based
test, specify absValue=0.5. This value does not affect statistics and
is used for plotting only.

# Stretch the plot manually to match tree to text

# Mikhail V. Matz, UT Austin, February 2015; matz@utexas.edu

##############################################################
# First, press command-D on mac or ctrl-shift-H in Rstudio and
navigate to the directory containing scripts and input files. Then
edit, mark and execute the following bits of code, one after another.


input="Infection_for_GOMWU_DA.csv" # two columns of comma-separated
values: gene id, continuous measure of significance. To perform
standard GO enrichment analysis based on Fisher's exact test, use
binary measure (0 or 1, i.e., either sgnificant or not).
goAnnotations="CV_Gene_MWU_ref.tab" # two-column, tab-delimited, one
line per gene, multiple GO terms separated by semicolon. If you have
multiple lines per gene, use nrify_GOtable.pl prior to running this
script.
goDatabase="go.obo" # download from http://www.geneontology.org/
```

```
GO.downloads.ontology.shtml
goDivision="BP" # either MF, or BP, or CC
source("gomwu.functions.R")


# Calculating stats. It might take ~3 min for MF and BP. Do not rerun
it if you just want to replot the data with different cutoffs, go
straight to gomwuPlot. If you change any of the numeric values below,
delete the files that were generated in previos runs first.
gomwuStats(input, goDatabase, goAnnotations, goDivision,
           perlPath="perl", # replace with full path to perl
executable if it is not in your system's PATH already
           largest=0.1,  # a GO category will not be considered if it
contains more than this fraction of the total number of genes
           smallest=3,   # a GO category should contain at least this
many genes to be considered
           clusterCutHeight=0.25, # threshold for merging similar
(gene-sharing) terms. See README for details.
           #     Alternative="g" # by default the MWU test is two-
tailed; specify "g" or "l" of you want to test for "greater" or "less"
instead.
           #     Module=TRUE,Alternative="g" # un-remark this if you
are analyzing a SIGNED WGCNA moduSle (values: 0 for not in module
genes, kME for in-module genes). In the call to gomwuPlot below,
specify absValue=0.001 (count number of "good genes" that fall into
the module)
           #     Module=TRUE # un-remark this if you are analyzing an
UNSIGNED WGCNA module
)
# do not continue if the printout shows that no GO terms pass 10% FDR.


# Plotting results
quartz()
results=gomwuPlot(input,goAnnotations,goDivision,
                  #      absValue=-log(0.05,10),  # genes with the
measure value exceeding this will be counted as "good genes". Specify
absValue=0.001 if you are doing Fisher's exact test for standard GO
enrichment or analyzing a WGCNA module (all non-zero genes = "good
genes").
                  absValue=1,
                  level1=0.1, # FDR threshold for plotting. Specify
level1=1 to plot all GO categories containing genes exceeding the
absValue.
                  level2=0.05, # FDR cutoff to print in regular (not
italic) font.
                  level3=0.01, # FDR cutoff to print in large bold
font.
                  txtsize=1.2,    # decrease to fit more on one page,
or increase (after rescaling the plot so the tree fits the text) for
```

```
better "word cloud" effect
                  treeHeight=0.5, # height of the hierarchical
clustering tree
                  #
colors=c("dodgerblue2","firebrick1","skyblue","lightcoral") # these
are default colors, un-remar and change if needed
)
# manually rescale the plot so the tree matches the text
# if there are too many categories displayed, try make it more
stringent with level1=0.05,level2=0.01,level3=0.001.

# text representation of results, with actual adjusted p-values

MWU_MF_Infection_DA <- read.csv("~/Dropbox/Infection_Tagseq/
Infection_GO/MWU_MF_Infection_for_GOMWU_DA.csv", sep="")
MWU_BP_Infection_DA <- read.csv("~/Dropbox/Infection_Tagseq/
Infection_GO/MWU_BP_Infection_for_GOMWU_DA.csv", sep="")
MWU_CC_Infection_DA <- read.csv("~/Dropbox/Infection_Tagseq/
Infection_GO/MWU_CC_Infection_for_GOMWU_DA.csv", sep="")

MWU_MF_Infection_DA_sig <-
MWU_MF_Infection_DA[(MWU_MF_Infection_DA$p.adj <= 0.05),]
MWU_BP_Infection_DA_sig <-
MWU_BP_Infection_DA[(MWU_BP_Infection_DA$p.adj <= 0.05),]
MWU_CC_Infection_DA_sig <-
MWU_CC_Infection_DA[(MWU_CC_Infection_DA$p.adj <= 0.05),]

colnames(MWU_MF_Infection_DA_sig) <-
c("MF_Infection.delta.rank","MF_Infection.pval","MF_Infection.level","
MF_Infection.nseqs","MF_Infection.term","name","MF_Infection.p.adj")
colnames(MWU_BP_Infection_DA_sig) <-
c("BP_Infection.delta.rank","BP_Infection.pval","BP_Infection.level","
BP_Infection.nseqs","BP_Infection.term","name","BP_Infection.p.adj")
colnames(MWU_CC_Infection_DA_sig) <-
c("CC_Infection.delta.rank","CC_Infection.pval","CC_Infection.level","
CC_Infection.nseqs","CC_Infection.term","name","CC_Infection.p.adj")
```