```
# Script created by Mark Christie, contact at Redpath.Christie@gmail.com
# Script created in version R 3.0.1
# This script:  Performs differetial gene expression using the edgeR package
# Usage notes:  Use line by line, adjusts for differences between "tanks"
using paired design
#=============================================================================
=========================================#
# Set working directory, import packages, source functions, initialize global
variables

setwd("C:/Dropbox/InPrep/RNASeq/working/DifferentialExpression_MainEffects")
library(edgeR)


#=============================================================================
=========================================#
dat.all <- read.csv("genecounts.csv", sep = '\t', header=TRUE, row.names=1)
sam <- cbind(1:length(colnames(dat.all)), colnames(dat.all))  # use this to
determine whch fish to use
samples <- cbind(sam, substr(sam[,2],16,20), substr(sam[,2],22,22),
substr(sam[,2],24,24))
samples
#samples <- samples[which(samples[, 4] == "F"), ]             # isolate by
sex of offspring
samples <- samples[which(samples[, 3] != "HFxWM"), ]          # isolate by
matrix type
samples <- samples[which(samples[, 3] != "WFxHM"), ]          # isolate by
matrix type
#samp1 <- samples[which(samples[, 5] == "K"), ]               # isolate by
specific matrix
#samp2 <- samples[which(samples[, 5] == "M"), ]
#samp3 <- samples[which(samples[, 5] == "O"), ]
#samp4 <- samples[which(samples[, 5] == "P"), ]
#samp5 <- samples[which(samples[, 5] == "V"), ]
#samples = rbind(samp1, samp2, samp3, samp4, samp5)

dat <- dat.all[, as.numeric(as.character(samples[, 1]))]       # index
desired individuals
dat.names <- colnames(dat)                                     # check to
make sure desired individuals were obtained
dat.names
group <- substr(dat.names, 16, 20)
table(group)

y <- DGEList(counts=dat,group=group)                           # Begin
formatting for edgeR
dim(y)                                                         # total number
of contigs and sample
y$samples                                                      # total number
of reads per sample
levels(y$samples$group)                                        # check that
all groups are represented

filter <- rowSums(cpm(y)> 10) >= 69                            # keep genes
with at least x count per million reads in at least y samples
y.filtered <- y[filter,]
dim(y.filtered)
```

```
y.normalized <- calcNormFactors(y.filtered)                    # normalize

barplot(y.normalized$samples$lib.size*1e-6, ylab = "library size (millions)")
#creates barplot of filtered read #
plotMDS(y.normalized, labels=group)
plotMDS(y.normalized, labels=dat.names, top = 100)             # MDS plot:
label options group for group, dat.names for individual sample names

# begin GLm approach
treatment <- group
litter <- substr(dat.names, 24, 24)
litter <- factor(litter)
treatment <- factor(treatment)
#treatment <- relevel(treatment, ref="WFxWM")

#design <- model.matrix(~litter+treatment+litter:treatment, data =
y.filtered$samples)  # various types of design matrices, interactions test
for differences in slope
#design <- model.matrix(~treatment+litter+treatment:litter, data =
y.filtered$samples)
#design <- model.matrix(~treatment+treatment:litter, data =
y.filtered$samples)
#design <- model.matrix(~treatment+litter, data = y.filtered$samples)
#design <- model.matrix(~treatment, data = y.filtered$samples)

design <- model.matrix(~litter+treatment)
rownames(design) <- colnames(y.normalized)

# estimate the dispersion
y.processed <- estimateGLMCommonDisp(y.normalized, design, verbose = TRUE)  #
estimate the overall dispersion
y.processed <- estimateGLMTrendedDisp(y.processed, design)                  #
estimate gene-wise dispersion rates
y.processed <- estimateGLMTagwiseDisp(y.processed, design)
plotBCV(y.processed)

fit <- glmFit(y.processed, design)

#lrt1 <- glmLRT(fit, coef = "treatmentWFxWM")   # ww vs hh
#lrt1 <- glmLRT(fit, coef = 2)   # ww vs hh
lrt1 <- glmLRT(fit)


top <- topTags(lrt1)
top
cpm(y)[rownames(top), ]


summary(de <- decideTestsDGE(lrt1))
#summary(de <- decideTestsDGE(lrt2))

detags <- rownames(lrt1)[as.logical(de)]
plotSmear(lrt1, de.tags=detags)
abline(h = c(-2, 2), col = "blue")
```

```
top <- topTags(lrt1, n = 1000)
top <- lrt1$table                # get all genes for pathway analysis
write.table(top,file="Allgenes.txt",col.names=TRUE, row.names=TRUE,
sep="\t",append=FALSE)
write.table(top,file="Topgenes_consolidated_mosthits.txt",col.names=TRUE,
row.names=TRUE, sep="\t",append=FALSE)
```