DIFFERENTIAL GENE EXPRESSION ANALYSIS

Module 5: Differential Gene Expression Analysis

DESEQ2 WORKFLOW USING R

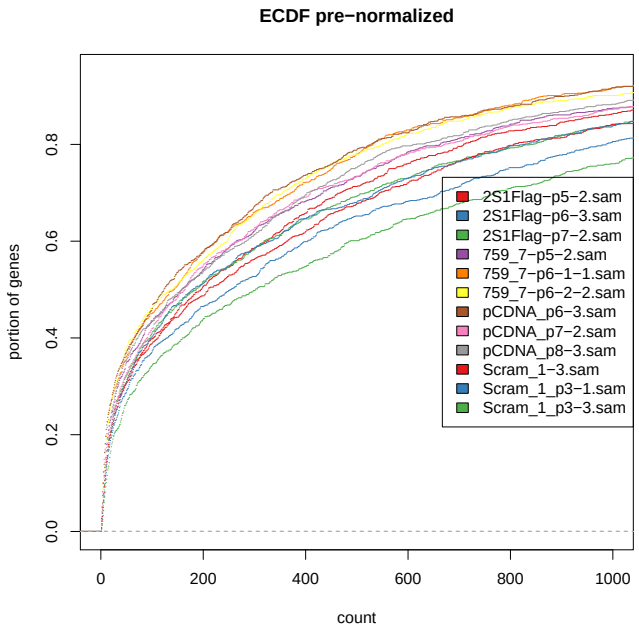
```
cd /home/$USER/DGE_Virtual/DESEQ2
```

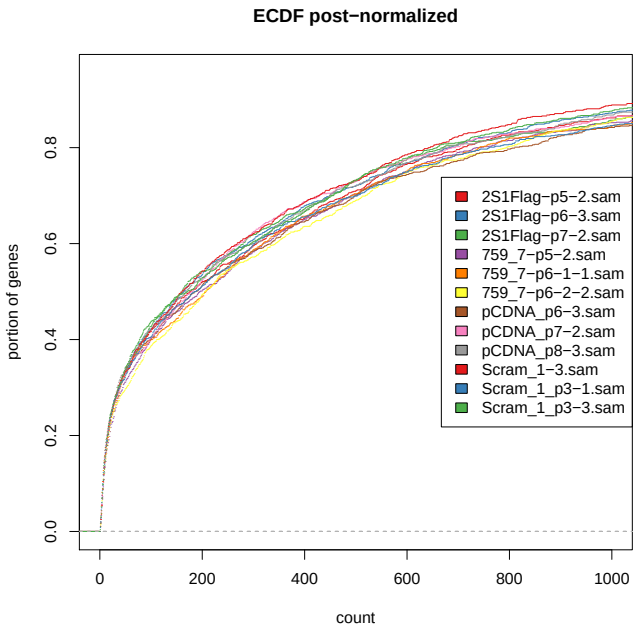
This is your working directory; all files will be written to this folder

RUNNING DESEQ2 USING R

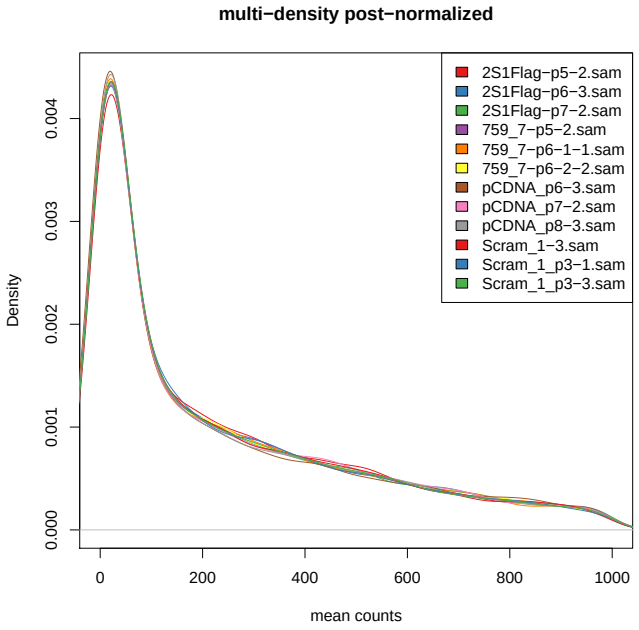
```
# THERE IS AN ENVIRONMENT TITLED "deseq2-r" CREATED FOR DESEQ2 R PACKAGE

source activate deseq2-r

# INVOKE R

R

##################### YOU WILL ENTER THE R ENVIRONMENT ########################

#Load the DESeq2 library into R

>library('DESeq2')

# Set working directory

>setwd('/home/$USER/DGE_Virtual/DESEQ2/')

# Make a variable called directory and specify the entire path to your counts files

>sampleCounts <- read.table("DESEQ2_matrix.tsv", header = TRUE, row.names = 1, check.names=FALSE)

# Create a dataframe, using the first row as column names and the first column as row names

>dim(sampleCounts)

# Check the dimensions of the newly created sampleTable dataframe object
# Specify the sample conditions for sampleTable
# In this example, we have 4 different sample conditions

>sampleCondition<- as.factor(c("2S1_Flag","2S1_Flag", "2S1_Flag", "759_7", "759_7",
"759_7", "pCDNA", "pCDNA", "pCDNA", "Scram", "Scram", "Scram"))

# Notice that we use the "sampleCondition" variable created previously.

>sampleTable <- data.frame(condition=sampleCondition)

# View contents of sampleTable

>sampleTable

# Make a data table from the count file using the sampleTable information.
```

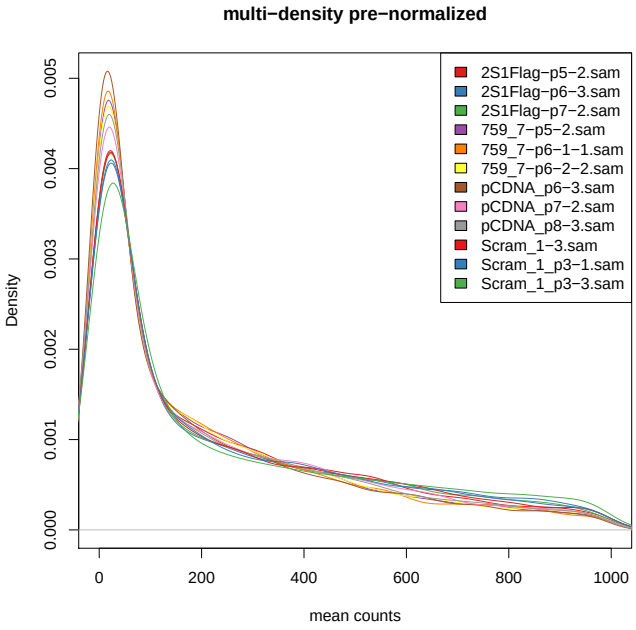
```
>DESeqDataset <- DESeqDataSetFromMatrix(countData=sampleCounts, colData=sampleTable, design=~condition)

# OPTIONAL STEP. Count Genes with non-zero in all samples.
# This command will count all the rows that have value > 0.

>GeneCounts <- counts(DESeqDataset)
>idx.nz <- apply(GeneCounts,1,function(x){all(x>0)})
>sum(idx.nz)

# Estimate Size factor
>DESeqSF <- estimateSizeFactors(DESeqDataset)
>sizeFactors(DESeqSF)

#Load gplots library

>library("gplots")

#Load library geneplotter

>library ("geneplotter")

# Plot densities of counts for the different samples to assess their distribution
# Note: When you plot pdf on command line, first name the file.
# Then type the command to generate the pdf and then finally save/write the pdf in your working directory.

# empirical cumulative density function not normalized

>pdf ("count_density_prenorm.pdf")
>multiecdf(counts(DESeqSF, normalized=F)[idx.nz,],main="ECDF pre-normalized",
xlab="count", ylab="portion of genes", xlim=c(0,1000))
>dev.off()

# empirical cumulative density function, normalized

>pdf ("count_density_postnorm.pdf")
>multiecdf(counts(DESeqSF, normalized=T)[idx.nz,],main="ECDF post-normalized",
xlab="count", ylab="portion of genes", xlim=c(0,1000))
>dev.off()
```





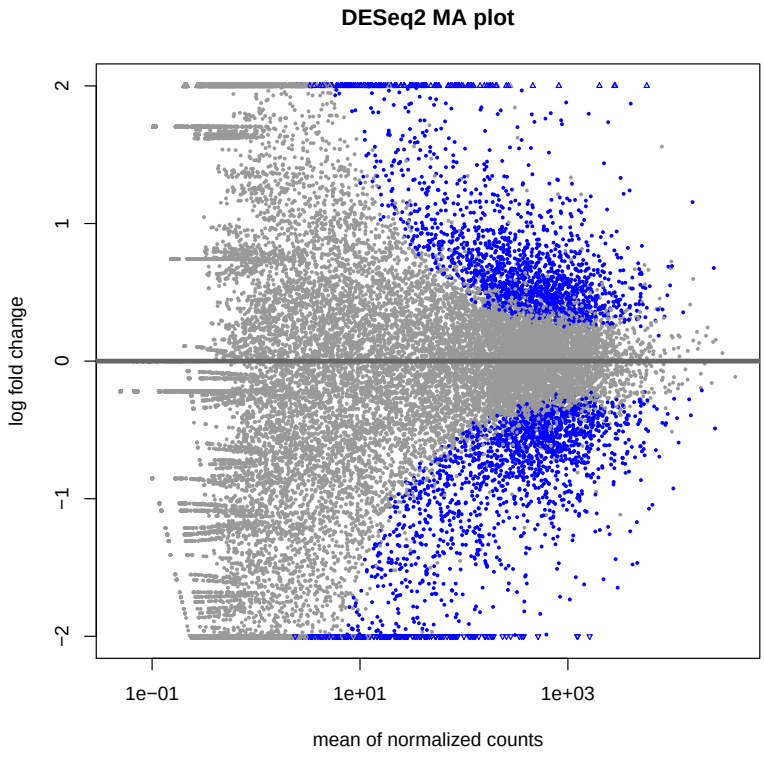
```
# multidensity plot, not normalized

>pdf ("multidensity_prenorm.pdf")
>multidensity(counts(DESeqSF, normalized=F)[idx.nz,],xlab="mean counts", xlim=c(0,1000))
>dev.off()

# multidensity plot, normalized

>pdf ("multidensity_postnorm.pdf")
>multidensity(counts(DESeqSF, normalized=T)[idx.nz,],xlab="mean counts", xlim=c(0,1000))
>dev.off()
```





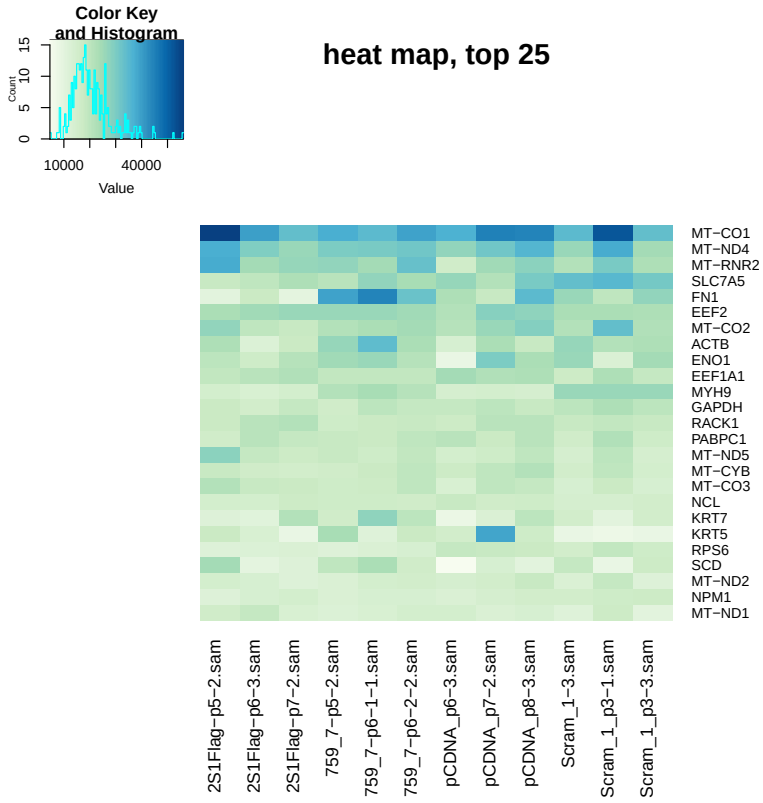

```
# Designate reference (this can be your control)

>colData(DESeqSF)$condition <- relevel(DESeqSF$condition, ref="Scram")

# Run DESeq2, which will do the following:
# estimating size factors
# estimating dispersions
# gene-wise dispersion estimates
# mean-dispersion relationship
# final dispersion estimates
# fitting model
# and testing

>dds <-DESeq(DESeqSF)

# Differential Expression Analysis for comparing the 759_7 to the Scram control sample.
# Inspect the counts files to confirm the direction differential expression.

>759_7_from_Scram <- results(dds, contrast=c("condition","759_7", "Scram"))

#generate results file and add normalized read counts to the results file

>resdata.1 <- merge(as.data.frame(759_7_from_Scram),
as.data.frame(counts(dds, normalized=TRUE)),by="row.names", sort=FALSE)
```

```
>write.csv(resdata.1, file="759_7_from_Scram.csv")

# Make MA plots of the DE results and print or save to a pdf file. The default DESeq2 p-value
# threshold is 0.1
# Change the level of significance cutoff:

>pdf("MA_759_from_Scram.pdf")
>plotMA(759_7_from_Scram, alpha = 0.05, ylim=c(-2,2),main="DESeq2 MA plot")
>dev.off()
```

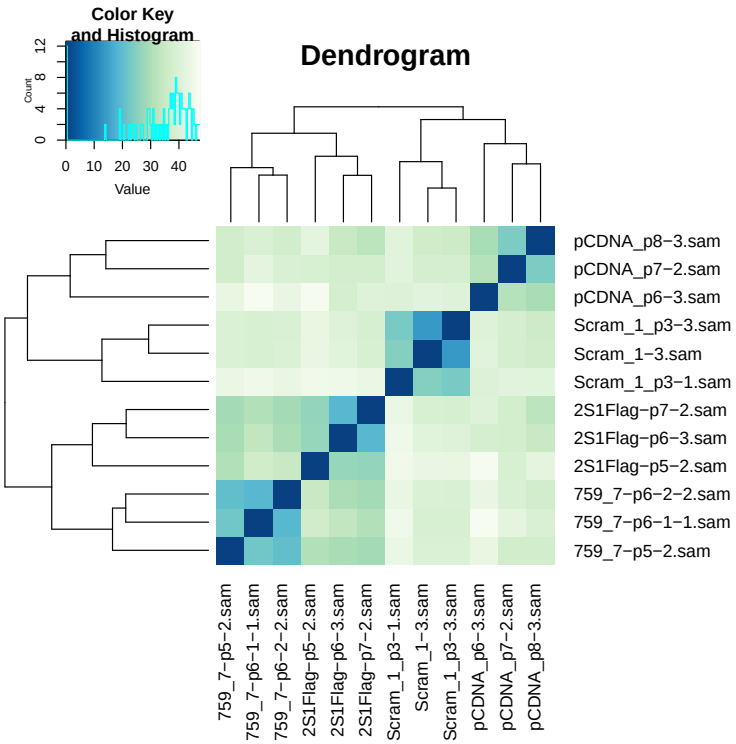



```
# The rlogTransformation function transforms the count data to the log2 scale in a way
# which minimizes differences between samples for rows with small counts, and which
# normalizes with respect to library size. The transformation is useful
# when checking for outliers.

>rld <-rlogTransformation(dds,blind=TRUE)
>assay(rld)

# This function calculates a variance stabilizing transformation (VST) from the fitted
# dispersion-mean relation(s) and then transforms the count data (normalized by division
# by the size factors or normalization factors), yielding a matrix of values which are
# now approximately homoskedastic (having constant variance along the range of mean values).
# The transformation also normalizes with respect to library size.

>vst <- varianceStabilizingTransformation(dds, blind=FALSE)
>assay(vst)

#Load library RColorBrewer

>library("RColorBrewer")
```

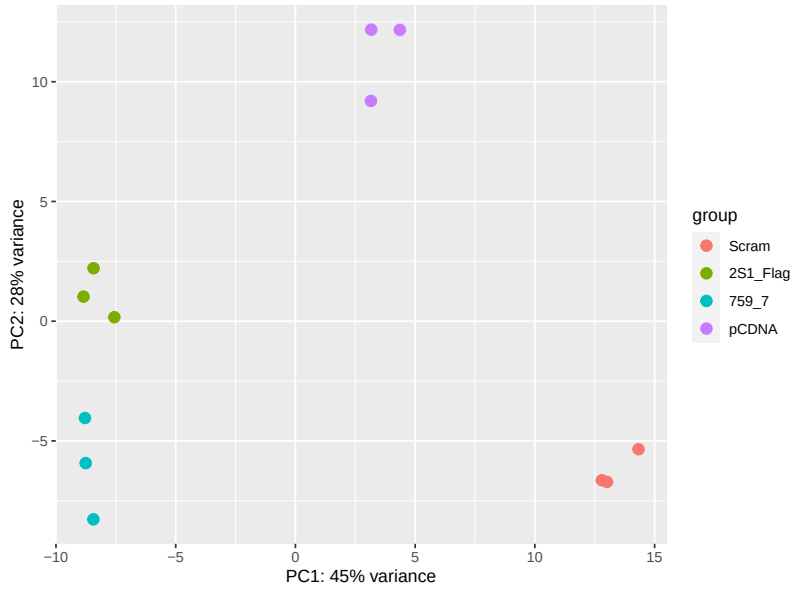
```
# Heatmaps for 25 most highly expressed genes (not necessarily the biggest fold change).
# Change the values in [1:25] to increase or decrease the number of genes in the heatmap

>pdf ("heatmap.top25.pdf")

>select <- order(rowMeans(counts(dds,normalized=TRUE)),decreasing=TRUE)[1:25]

>hmcol <- colorRampPalette(brewer.pal(9,"GnBu"))(100)

>heatmap.2(counts(dds,normalized=TRUE)[select,],main="heat map, top 25",
col=hmcol,Rowv=FALSE, Colv=FALSE, scale="none",
dendrogram="none",trace="none", margin=c(10,6))

>dev.off()
```




```
# Calculate the sample to sample distances using the rld transformed data to
# make a dendrogram to look at the clustering of the samples

>pdf("Dendrogram.pdf")
>distsRL <- dist(t(assay(rld)))
>mat <- as.matrix(distsRL)
>rownames(mat) <- colnames(mat) <- with(colData(dds), paste(colnames(DESeqDataset)))
>hc <- hclust(distsRL)
>heatmap.2(mat,Rowv=as.dendrogram(hc),main = "Dendrogram", symm=TRUE,trace="none", col=rev(hmcol),margin=c(13,13))
>dev.off()
```




```
# Perform a Principal Component Analysis and print the PCA plot from the rld transformed data.

>pdf("PCA.allsamples.pdf")
>print(plotPCA(rld,intgroup=c("condition")))
>dev.off()
```



```
#Save history
>savehistory(file= "tutorial_DGE.Rhistory")

#quit R
>q()
```