Commands for using edgeR to identify group-specific sequences and displaying the taxonomic classifications of the group-specific sequences
Brazelton Lab, November 2015

References:

Schloss, P. D. et al. (2009). Introducing mothur: Open Source, Platform-independent, Community-supported Software for Describing and Comparing Microbial Communities. *Appl. Environ. Microbiol.*, AEM.01541–09. doi:10.1128/AEM.01541-09

McMurdie, P. J., & Holmes, S. (2013). phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data. *PloS One*, *8*(4), e61217. doi:10.1371/journal.pone.0061217

McMurdie, P. J., & Holmes, S. (2014). Waste not, want not: why rarefying microbiome data is inadmissible. *PLoS Computational Biology*, *10*(4), e1003531. doi:10.1371/journal.pcbi.1003531

Robinson, MD, McCarthy, DJ, Smyth, GK (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26, 139–140.

Starting materials:

- count table from mothur
- taxonomy table from mothur
- sample info file with exact same sample names as count table

* Check that all files have unix linebreaks.

Edit the taxonomy table from mothur with this script to make it more readable downstream
taxonomy2tsv.py [filename].taxonomy
==> [filename].taxonomy.tsv

Before going further, you might want to use the 'screen' command so that you can recover your work in progress after losing connection to the server:

```
screen
```

Now launch R and create the phyloseq objects

```
srun --x11=first --pty R
library(phyloseq)
otus = read.table("[filename].count_table", header=TRUE, row.names=1)
otus = otu_table(otus, taxa_are_rows=TRUE)
otus = subset(otus, select=-c(1:1))     # to delete the "total" column from the mothur count_table
tax = read.table("[filename].taxonomy.tsv", sep='\t', header=FALSE, row.names=1)
tax = tax_table(as.matrix(tax))
sam = read.csv("[filename]-sample-info.csv")
sam = sample_data(sam)
row.names(sam) = sample_names(otus)
merged = merge_phyloseq(otus,tax,sam)
```

```
merged
```
[copy screen output of 'merged' into your notebook]

 # Output 1: bar chart of whole-sample taxonomic classifications. “V6” corresponds to family level. “V5” is order, “V4” is class, etc.

```
merged_props = transform_sample_counts(merged, function(x) 100 * x/sum(x))
merged_props_glomV6 = tax_glom(merged_props, "V6")
merged_props_glomV6
library(ggplot2)
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2",
"#D55E00", "#CC79A7")
scale = length(get_taxa_unique(merged, 'V6'))
image_width = 8 * scale
image_height = 40 * scale
png(filename = 'predeseq-barplot-glomV6.png', width = image_width, height =
image_height, units = 'px', pointsize = 18)
plot_bar(merged_props_glomV6, fill='V6') + coord_flip() + ylab('Percent Sequences') +
scale_fill_manual(values=colorRampPalette(cbPalette)(scale)) +
theme(legend.position='bottom') + guides(fill=guide_legend(keywidth=1, keyheight=1,
ncol=4, label.position='right', title.position='top', title.hjust=0.5))
dev.off()
```

Next we will use independent filtering to remove low-variance sequences, which will improve our statistical power later. The threshold can be adjusted for different kinds of data.

```
varianceThreshold = 1e-5 #adjust this number for your dataset
keep_seqs = names(which(apply(otu_table(merged_props), 1, var) > varianceThreshold))
merged_pruned5 = prune_taxa(keep_seqs, merged)
merged_pruned5
```
[copy screen output to your notebook]

Next we want to run edgeR, but first the phyloseq-to-edgeR function must be defined because the phyloseq package does not include it. The code for the function is in the edgeR documentation: http://joey711.github.io/phyloseq-extensions/edgeR.html.
Copy and paste the below into R.

```
phyloseq_to_edgeR = function(physeq, group, method="RLE", ...){
  require("edgeR")
  require("phyloseq")
  # Enforce orientation.
  if( !taxa_are_rows(physeq) ){ physeq <- t(physeq) }
  x = as(otu_table(physeq), "matrix")
  # Add one to protect against overflow, log(0) issues.
  x = x + 1
  # Check `group` argument
  if( identical(all.equal(length(group), 1), TRUE) & nsamples(physeq) > 1 ){
    # Assume that group was a sample variable name (must be categorical)
```

```
    group = get_variable(physeq, group)
  }
  # Define gene annotations (`genes`) as tax_table
  taxonomy = tax_table(physeq, errorIfNULL=FALSE)
  if( !is.null(taxonomy) ){
    taxonomy = data.frame(as(taxonomy, "matrix"))
  }
  # Now turn into a DGEList
  y = DGEList(counts=x, group=group, genes=taxonomy, remove.zeros = TRUE, ...)
  # Calculate the normalization factors
  z = calcNormFactors(y, method=method)
  # Check for division by zero inside `calcNormFactors`
  if( !all(is.finite(z$samples$norm.factors)) ){
    stop("Something wrong with edgeR::calcNormFactors on this data,
         non-finite $norm.factors, consider changing `method` argument")
  }
  # Estimate dispersions
  return(estimateTagwiseDisp(estimateCommonDisp(z)))
}
```

Now we can run edgeR:

```
dge = phyloseq_to_edgeR(merged_pruned[number of your variance threshold above],
group="[your-group-category]")
et = exactTest(dge, pair=c('[your-group1]','[your-group2]'))
```

first group will be reported as negative differential abundance. second group will be reported as positive differential abundance.

 # Output 2: table of edgeR results:

```
tt = topTags(et, n=nrow(dge$table), adjust.method="BH", sort.by="PValue")
res = tt@.Data[[1]]
write.csv(res,file='edgeR-results-withfiltering.csv')
```

logFC = $\log_2$ fold change, logCPM = $\log_2$ counts per million, FDR = false discovery rate

 # Output 3: plot differential abundance of sequences, marking those in red which have better than 5% FDR:

```
de = decideTestsDGE(et)
detags = rownames(dge)[as.logical(de)]
png(filename = "smear-withfiltering.png", width = 480, height = 480, units = "px",
pointsize = 18)
plotSmear(et, de.tags=detags)
dev.off()
```

Stop. The three outputs above constitute the actual statistical results of the analysis. Spend some time with them and decide whether you want to proceed with the rest of the workflow, which is just a re-visualization of these results in a possibly more aesthetically-pleasing manner.

Next we want to display the taxonomic breakdown of the sequences enriched in each group.
First: select significantly up- or down-"regulated" taxa:
```
sig = res[res$'FDR' < 0.05, ]
other = res[res$'FDR' > 0.05, ]
pos = sig[sig$'logFC' > 0, ]
neg = sig[sig$'logFC' < 0, ]
```

Add count table information to the selected results for future reference in the output files
```
pos_tax_counts = cbind(as(pos, "data.frame"), as(otu_table(merged)[rownames(pos), ],
"matrix"))
neg_tax_counts = cbind(as(neg, "data.frame"), as(otu_table(merged)[rownames(neg), ],
"matrix"))
other_tax_counts = cbind(as(other, "data.frame"),
as(otu_table(merged)[rownames(other), ], "matrix"))
```

 # Output 4: tables of edgeR results including taxonomy and count table information:
```
write.csv(pos_tax_counts,file='edgeR-pos-results.csv')
write.csv(neg_tax_counts,file='edgeR-neg-results.csv')
write.csv(other_tax_counts,file='edgeR-other-results.csv')
```

Make phyloseq objects with OTU counts and taxonomy for manipulation in R later
```
otus_pos = otus[row.names(pos), ]
otus_neg = otus[row.names(neg), ]
otus_other = otus[row.names(other), ]
physeq_pos = merge_phyloseq(otus_pos,tax)
physeq_neg = merge_phyloseq(otus_neg,tax)
physeq_other = merge_phyloseq(otus_other,tax)
```

Rename sample names and merge into a single phyloseq object
```
sample_names(physeq_pos) = paste("pos-", sample_names(physeq_pos), sep="")
sample_names(physeq_neg) = paste("neg-", sample_names(physeq_neg), sep="")
sample_names(physeq_other) = paste("other-", sample_names(physeq_other), sep="")
physeq_all = merge_phyloseq(physeq_pos, physeq_neg, physeq_other)
sample_names(physeq_all)
```

You probably want to separate the taxonomy of sequences that occurred in the "positive" samples from those in the "negative" samples. In other words, the percentage of sequences with a given taxonomy in the "positive" group should only reflect the counts of sequences in "positive" samples, and vice versa. Therefore, make a list of which samples to keep in the final bar charts, labeling each sample as either "pos" or "neg". **For example (replace names with your actual names)**:
```
keep =
c('pos-Serp_Lig_GOR1AB_2013','pos-Serp_Lig_GOR1CDEF_2013','neg-Serp_Lig_GORupA_2013','
neg-Serp_Lig_GORupA_2013','neg-Serp_Lig_GORupB_2013','neg-Serp_LIG_Gor_upA_Jul12_Bv4v5
','neg-Serp_LIG_Gor_upB_Jul12_Bv4v5')
```

```
physeq_all = prune_samples(keep, physeq_all)
```

#The next set of commands will merge the counts for the positive, negative, and other categories of samples. Edit the first command so that it lists the correct number of pos, neg, and other. Check sample_names(physeq_all) to be sure:

```
sample_names(physeq_all)
types = rep(c("pos", "neg"), times=c(2, 4))
sampledata = sample_data(data.frame(Type = types, size = nsamples(physeq_all), replace
= TRUE, row.names = sample_names(physeq_all), stringsAsFactors = FALSE))
sampledata # check that assignments are correct
physeq_all_merged = merge_phyloseq(physeq_all,sampledata)
physeq_all_merged = merge_samples(physeq_all_merged, 'Type')
```

#Ignore warning message about NAs introduced by coercion

```
physeq_all_merged_props = transform_sample_counts(physeq_all_merged, function(x) 100 *
x/sum(x))
```

 # Output 5: Bar charts of taxonomy enriched in each group of samples

#Family-level plot:

```
physeq_all_merged_props_glom6 = tax_glom(physeq_all_merged_props, 'V6')

library(ggplot2)
scale = length(get_taxa_unique(physeq_all_merged_props_glom6, 'V6'))
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2",
"#D55E00", "#CC79A7")
png(filename = "edgeR-barplot-glom6.png", width = 3000, height = 3000, units = "px",
pointsize = 18)
plot_bar(physeq_all_merged_props_glom6, fill="V6") + coord_flip() + ylab("Percent
Sequences") + scale_fill_manual(values=colorRampPalette(cbPalette)(scale)) +
theme(legend.position='bottom') + guides(fill=guide_legend(keywidth=1, keyheight=1,
ncol=4, label.position='right', title.position='top', title.hjust=0.5))
dev.off()
```

 # Output 6: Tables of results corresponding to bar charts above

#Write csv file containing otu counts and taxonomy to aid in interpretation of the bar plot. The t() transposes the otu_table so that species are rows, as in the tax table.

```
final_tax_otu_props_glom6 = cbind(as(t(otu_table(physeq_all_merged_props_glom6)),
"matrix"), as(tax_table(physeq_all_merged_props_glom6), "matrix"))
write.csv(final_tax_otu_props_glom6,file='final_tax_otu_props_glom6.csv')
```

Alternative code with TMM normalization method and without adding one to correct for zero counts:

```
phyloseq_to_edgeR = function(physeq, group, method="TMM", ...){
  require("edgeR")
```

```
  require("phyloseq")
  # Enforce orientation.
  if( !taxa_are_rows(physeq) ){ physeq <- t(physeq) }
  x = as(otu_table(physeq), "matrix")
  # Add one to protect against overflow, log(0) issues.
  # x = x + 1
  # Check `group` argument
  if( identical(all.equal(length(group), 1), TRUE) & nsamples(physeq) > 1 ){
          # Assume that group was a sample variable name (must be categorical)
          group = get_variable(physeq, group)
  }
  # Define gene annotations (`genes`) as tax_table
  taxonomy = tax_table(physeq, errorIfNULL=FALSE)
  if( !is.null(taxonomy) ){
          taxonomy = data.frame(as(taxonomy, "matrix"))
  }
  # Now turn into a DGEList
  y = DGEList(counts=x, group=group, genes=taxonomy, ...)
  # Calculate the normalization factors
  z = calcNormFactors(y, method=method)
  # Check for division by zero inside `calcNormFactors`
  # if( !all(is.finite(z$samples$norm.factors)) ){
   # stop("Something wrong with edgeR::calcNormFactors on this data,
   #      non-finite $norm.factors, consider changing `method` argument")
  #}
  # Estimate dispersions
  return(estimateTagwiseDisp(estimateCommonDisp(z)))
}
```