Marina Proximity 16S Amplicon Analysis Notebook

Cassie Ettinger

August 13, 2018

Some background

During the summer of 2013, Sofie Voerman collected leaf, root, water and sediment samples for microbiome analysis from 0.25 m2 quadrats (n = 3) located near the marina (2 m from the marina wall), at an intermediate distance halfway into the patch (20 m from the marina wall), and at the farthest side of the patch from the harbor (40 m from the marina wall and 2 m from the outer edge of the patch) in Bodega Bay, CA. Bacterial and archaeal 16S rRNA genes were amplified from DNA extracted from these samples using 515F and 806R primers. Sequencing was performed by the DNA Technologies Core facility in the Genome Center (http://dnatech.genomecenter.ucdavis.edu/) on an Illumina MiSeq (250 bp paired-end reads). The resulting fastq files have been deposited at GenBank under the accession no. PRJNA350006. This notebook describes the analysis of this 16S dataset.

Loading packages and setting up the analysis

First, load in the packages that will be used and note their versions

```
library(phyloseq)
library(ggplot2)
library(rmarkdown)
library(dada2)
library(decontam)
library(phangorn)
library(DECIPHER)
library(vegan)
library(dplyr)
library(RColorBrewer)
library(reshape)
library(coin)
library(FSA)

#get versions for the packages, that I am using
#packageVersion("dada2") #1.4.0
#packageVersion("phyloseq") #1.20.0
#packageVersion("decontam") # 0.20.0
#packageVersion("phangorn") #2.4.0
#packageVersion("DECIPHER") #2.6.0
#packageVersion("ggplot2") #2.2.1
#packageVersion("dplyr") #0.7.6
#packageVersion("vegan") #2.5.2
#packageVersion("RColorBrewer") #1.1.2
#packageVersion("reshape") #0.8.7
#packageVersion("coin") #1.2.2
#packageVersion("FSA") #0.8.20
```

Going to set the "seed", this ensures any randomization always happens the same way if this analysis needs to be re-run

```
set.seed(5311)
```

Using Dada2 to create an amplicon sequence variant (ASV) table

Setting the path for the fastq files

```
#location of data
raw_data <- "/Users/Cassie/Box Sync/Seagrass/HarborData/RawFastq/"
list.files(raw_data)
```

```
##  [1] "SG104_BB041_1.fastq.gz" "SG104_BB041_2.fastq.gz"
##  [3] "SG104_BB042_1.fastq.gz" "SG104_BB042_2.fastq.gz"
##  [5] "SG104_BB043_1.fastq.gz" "SG104_BB043_2.fastq.gz"
##  [7] "SG104_BB044_1.fastq.gz" "SG104_BB044_2.fastq.gz"
##  [9] "SG104_BB045_1.fastq.gz" "SG104_BB045_2.fastq.gz"
## [11] "SG104_BB046_1.fastq.gz" "SG104_BB046_2.fastq.gz"
## [13] "SG104_BB047_1.fastq.gz" "SG104_BB047_2.fastq.gz"
## [15] "SG104_BB048_1.fastq.gz" "SG104_BB048_2.fastq.gz"
## [17] "SG104_BB049_1.fastq.gz" "SG104_BB049_2.fastq.gz"
## [19] "SG104_BB050_1.fastq.gz" "SG104_BB050_2.fastq.gz"
## [21] "SG104_BB051_1.fastq.gz" "SG104_BB051_2.fastq.gz"
## [23] "SG104_BB052_1.fastq.gz" "SG104_BB052_2.fastq.gz"
## [25] "SG104_BB053_1.fastq.gz" "SG104_BB053_2.fastq.gz"
## [27] "SG104_BB054_1.fastq.gz" "SG104_BB054_2.fastq.gz"
## [29] "SG104_BB055_1.fastq.gz" "SG104_BB055_2.fastq.gz"
## [31] "SG104_BB056_1.fastq.gz" "SG104_BB056_2.fastq.gz"
## [33] "SG104_BB057_1.fastq.gz" "SG104_BB057_2.fastq.gz"
## [35] "SG104_BB058_1.fastq.gz" "SG104_BB058_2.fastq.gz"
## [37] "SG104_BB059_1.fastq.gz" "SG104_BB059_2.fastq.gz"
## [39] "SG104_BB060_1.fastq.gz" "SG104_BB060_2.fastq.gz"
## [41] "SG104_BB061_1.fastq.gz" "SG104_BB061_2.fastq.gz"
## [43] "SG104_BB062_1.fastq.gz" "SG104_BB062_2.fastq.gz"
## [45] "SG104_BB063_1.fastq.gz" "SG104_BB063_2.fastq.gz"
## [47] "SG104_BB064_1.fastq.gz" "SG104_BB064_2.fastq.gz"
## [49] "SG104_BB065_1.fastq.gz" "SG104_BB065_2.fastq.gz"
## [51] "SG104_BB066_1.fastq.gz" "SG104_BB066_2.fastq.gz"
## [53] "SG104_BB067_1.fastq.gz" "SG104_BB067_2.fastq.gz"
## [55] "SG104_BB068_1.fastq.gz" "SG104_BB068_2.fastq.gz"
## [57] "SG104_BB069_1.fastq.gz" "SG104_BB069_2.fastq.gz"
## [59] "SG104_BB070_1.fastq.gz" "SG104_BB070_2.fastq.gz"
## [61] "SG104_BB071_1.fastq.gz" "SG104_BB071_2.fastq.gz"
## [63] "SG104_BB072_1.fastq.gz" "SG104_BB072_2.fastq.gz"
## [65] "SG104_BB073_1.fastq.gz" "SG104_BB073_2.fastq.gz"
## [67] "SG104_BBNC_1.fastq.gz"  "SG104_BBNC_2.fastq.gz"
```

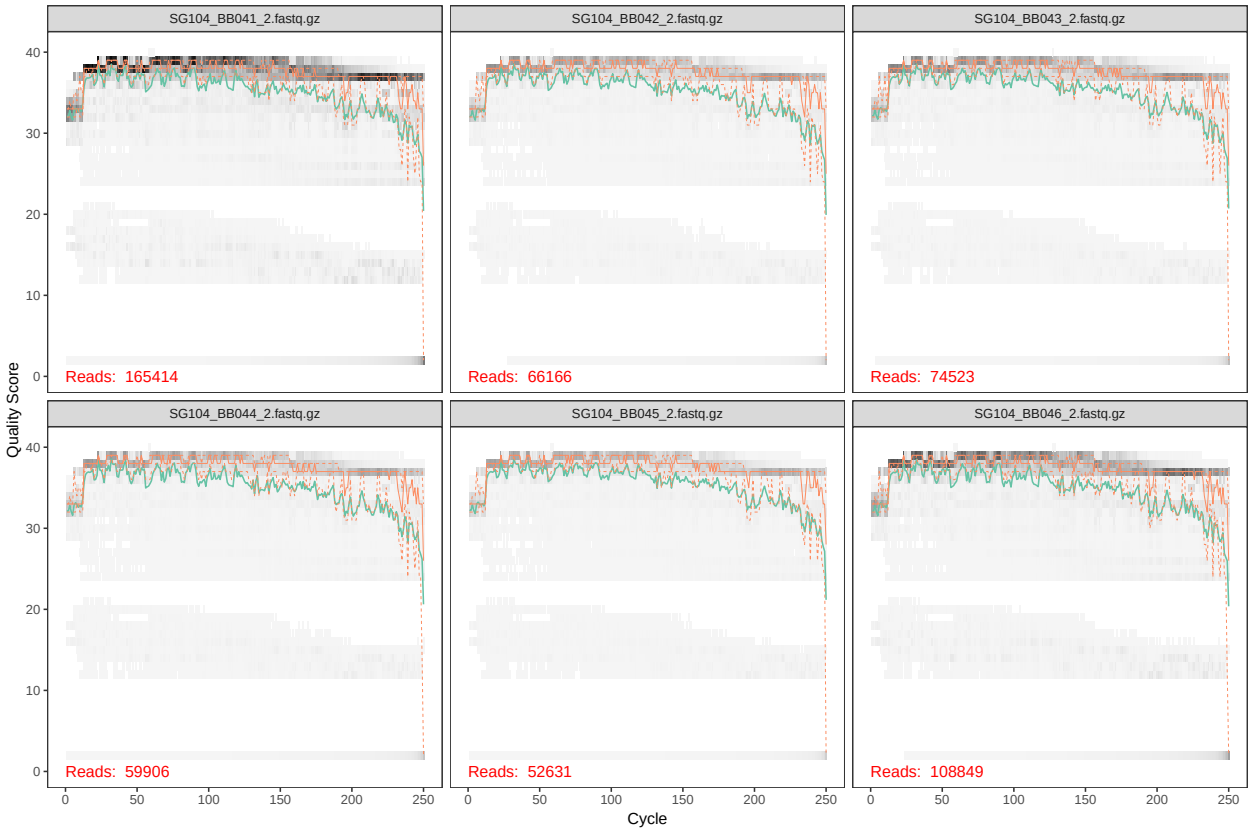
```
#Sort and get sample names
fnFs <- sort(list.files(raw_data, pattern="_1.fastq.gz"))
fnRs <- sort(list.files(raw_data, pattern="_2.fastq.gz"))
sample.names <- sapply(strsplit(fnFs, "_"), `[`, 2)

#specify full paths to the data
fnFs <- file.path(raw_data, fnFs)
fnRs <- file.path(raw_data, fnRs)
```

Quality Inspection

Looking at our data to try to see at what point the read quality drops below Q20

```
#Inspecting quality of data
plotQualityProfile(fnFs[1:6]) #fwd reads for first 6 samples
```


```
plotQualityProfile(fnRs[1:6]) #reverse reads for first 6 samples
```



Filter and Trim

```
# File parsing
#specify where to save filtered data that we will generate and what to name the files
pathF <- "/Users/Cassie/Box Sync/Seagrass/HarborData/Dada2/FWD/"
pathR <- "/Users/Cassie/Box Sync/Seagrass/HarborData/Dada2/REV/"
filtpathF <- file.path(pathF, "filtered")
filtpathR <- file.path(pathR, "filtered")
fastqFs <- sort(list.files(pathF, pattern="fastq"))
fastqRs <- sort(list.files(pathR, pattern="fastq"))

#checking to make sure we have matching files
if(length(fastqFs) != length(fastqRs)) stop("Forward and reverse files do not match.")

#Filtering:
#chose 200 bp trim forward and 250 bp trim rev arbitrarily based on error profiles
#I think read 1 = is really the reverse read, and read 2 = fwd read
#hence why read 1 is worse quality than read 2
#max n = 0, bc dada2 can't handle N's
#truncQ = truncate when quality score falls below 10, this might be too high
out <- filterAndTrim(fwd=file.path(pathF, fastqFs), filt=file.path(filtpathF, fastqFs),
                     rev=file.path(pathR, fastqRs), filt.rev=file.path(filtpathR, fastqRs),
                     truncLen=c(200,250), maxEE=2, truncQ=2, maxN=0, rm.phix=TRUE,
                     compress=TRUE, verbose=TRUE, multithread=TRUE)
```

Sample Inference

```
# File parsing
# new paths to filtered files
filtpathF <- "/Users/Cassie/Box Sync/Seagrass/HarborData/Dada2/FWD/filtered/"
filtpathR <- "/Users/Cassie/Box Sync/Seagrass/HarborData/Dada2/REV/filtered/"
filtFs <- list.files(filtpathF, pattern="fastq", full.names = TRUE)
filtRs <- list.files(filtpathR, pattern="fastq", full.names = TRUE)

# filtered file names
sample.names <- sapply(strsplit(basename(filtFs), "_"), `[`, 2)
sample.namesR <- sapply(strsplit(basename(filtRs), "_"), `[`, 2)

#check if have corresponding files
if(!identical(sample.names, sample.namesR)) stop("Forward and reverse files do not match.")
names(filtFs) <- sample.names
names(filtRs) <- sample.names

# Learn forward error rates
errF <- learnErrors(filtFs, nread=1e6, multithread=TRUE)
```

```
## Warning in learnErrors(filtFs, nread = 1e+06, multithread = TRUE): The
## nreads parameter is DEPRECATED. Please update your code with the nbases
## parameter.
```

```
## 215633400 total bases in 1078167 reads from 27 samples will be used for learning the error rates.
## Initializing error rates to maximum possible estimate.
## selfConsist step 1 ..........................
##    selfConsist step 2
##    selfConsist step 3
##    selfConsist step 4
##    selfConsist step 5
##    selfConsist step 6
##    selfConsist step 7
## Convergence after  7  rounds.
```

```
# Learn reverse error rates
errR <- learnErrors(filtRs, nread=1e6, multithread=TRUE)
```

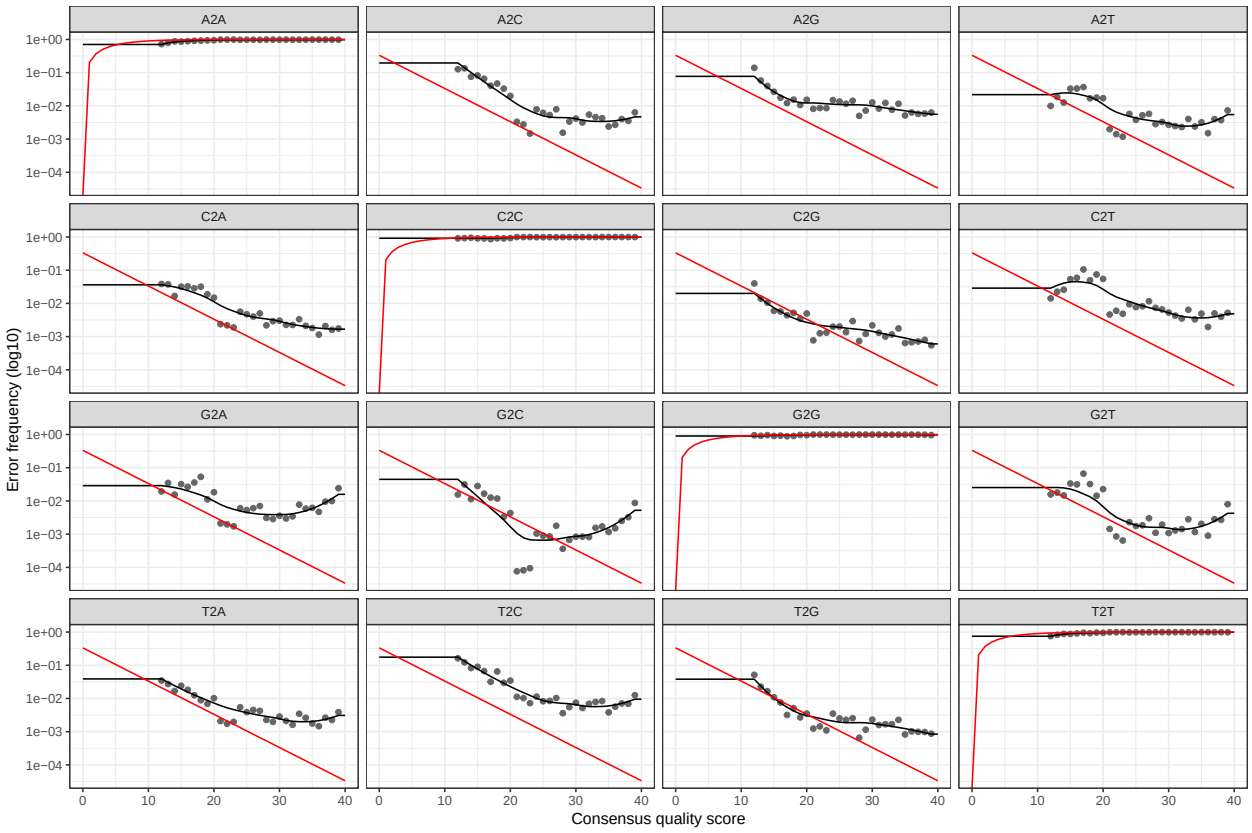
```
## Warning in learnErrors(filtRs, nread = 1e+06, multithread = TRUE): The
## nreads parameter is DEPRECATED. Please update your code with the nbases
## parameter.
```

```
## 269541750 total bases in 1078167 reads from 27 samples will be used for learning the error rates.
## Initializing error rates to maximum possible estimate.
## selfConsist step 1 ..........................
##    selfConsist step 2
##    selfConsist step 3
##    selfConsist step 4
##    selfConsist step 5
##    selfConsist step 6
##    selfConsist step 7
##    selfConsist step 8
##    selfConsist step 9
##    selfConsist step 10
```

```
## Self-consistency loop terminated before convergence.
```

```
#graph error rate estimates
plotErrors(errF, nominalQ=TRUE)
```

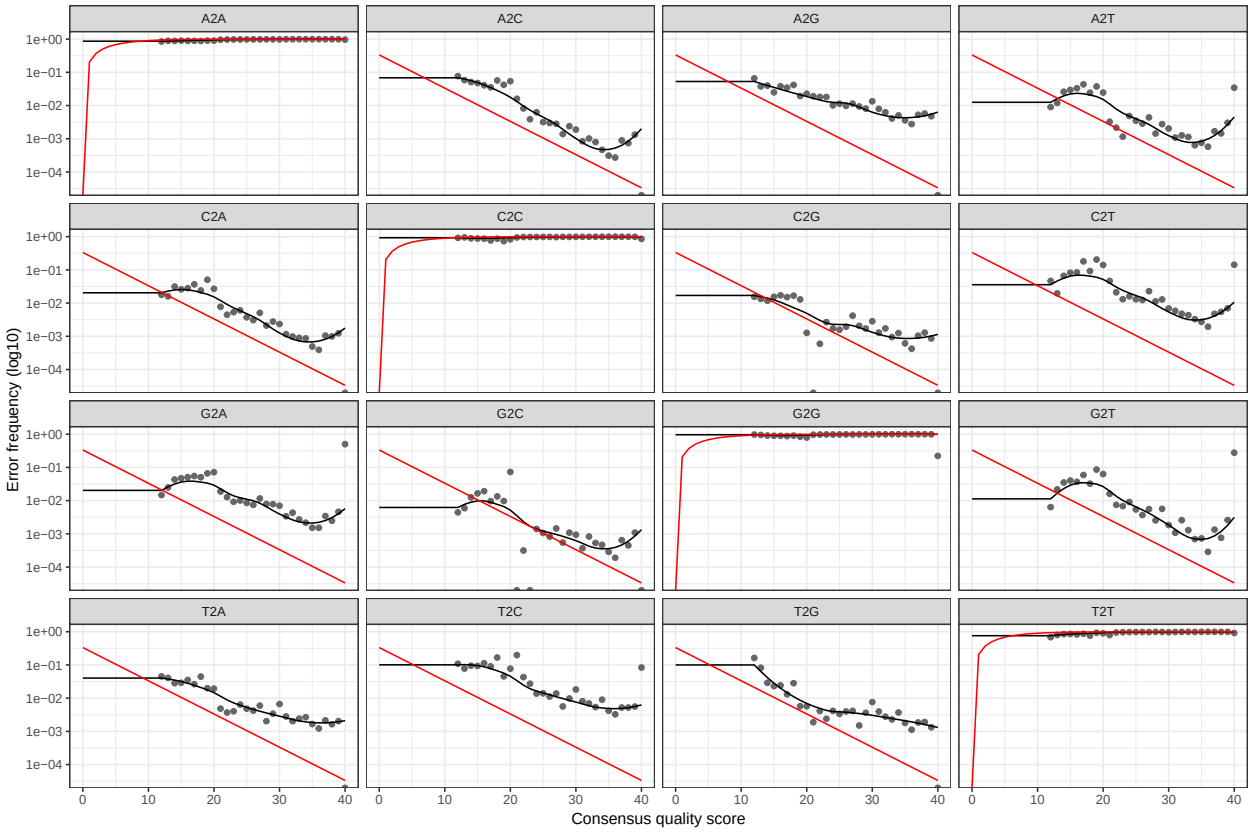
```
## Warning: Transformation introduced infinite values in continuous y-axis
```




```
plotErrors(errR, nominalQ=TRUE)
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```



Sample inference and merging of paired-end reads

```
mergers <- vector("list", length(sample.names))
names(mergers) <- sample.names
for(sam in sample.names) {
  cat("Processing:", sam, "\n")
  derepF <- derepFastq(filtFs[[sam]])
  ddF <- dada(derepF, err=errF, multithread=TRUE)
  derepR <- derepFastq(filtRs[[sam]])
  ddR <- dada(derepR, err=errR, multithread=TRUE)
  merger <- mergePairs(ddF, derepF, ddR, derepR)
  mergers[[sam]] <- merger
}
```

Making the ASV table

```
# Construct sequence table and remove chimeras
seqtab <- makeSequenceTable(mergers)

#saveRDS(seqtab, "Harbor_seqtab.rds")
#write.csv(seqtab,  "Harbor_seqtab.csv")
```

```
seqtab <- readRDS( "Harbor_seqtab.rds")
dim(seqtab)
```

```
## [1]   34 7672
```

Removing chimeras

```
# Remove chimeras
seqtab2 <- removeBimeraDenovo(seqtab, method="consensus", multithread=TRUE)

#saveRDS(seqtab2, "Harbor_seqtab_nochimeras.rds")
```

```
seqtab2 <- readRDS( "Harbor_seqtab_nochimeras.rds")

dim(seqtab2)
```

```
## [1]   34 7577
```

```
# Inspect distribution of sequence lengths
table(nchar(getSequences(seqtab2)))
```

```
##
##  250  251  252  253  254  255  256  257  258  262  269  270  276  285  289
##   13   12  175 6296  915  103   42    5    3    1    1    1    1    2    1
##  292  301  414  422
##    1    1    2    2
```

```
#percent seqs passed
sum(seqtab2)/sum(seqtab)
```

```
## [1] 0.9875985
```

Assign Taxonomy

```
#silva v132
tax_v132 <- assignTaxonomy(seqtab2,
                           "/Users/Cassie/Box Sync/Databases/silva/silva132/silva_nr_v132_train_set.fa.

tax_v132_sp <- addSpecies(tax_v132,
                          "/Users/Cassie/Box Sync/Databases/silva/silva132/silva_species_assignment_v13

#saveRDS(tax_v132, "Harbor_tax_silva132.rds")
#saveRDS(tax_v132_sp, "Harbor_tax_silva132_w_sp.rds")
```

Build Phylogeny

```
#get DNA sequences from chimera-removed ASV table
seqs <- getSequences(seqtab2)
names(seqs) <- seqs # This propagates to the tip labels of the tree

#Align DNA sequences
alignment <- AlignSeqs(DNAStringSet(seqs), anchor=NA,verbose=FALSE)

#Turn alignment into a matrix
```

```
phangAlign <- phyDat(as(alignment, "matrix"), type="DNA")

#calculate maximum liklihood values for alignment
dm <- dist.ml(phangAlign)

#build neighbor joining tree
#a GTR+G+I (Generalized time-reversible with Gamma rate variation)
#maximum likelihood tree using the neighbor-joining tree
treeNJ <- NJ(dm) # Note, tip order != sequence order
fit = pml(treeNJ, data=phangAlign)
fitGTR <- update(fit, k=4, inv=0.2)
fitGTR <- optim.pml(fitGTR, model="GTR", optInv=TRUE,
                    optGamma=TRUE, rearrangement = "stochastic", control = pml.control(trace = 0))

#saveRDS(fitGTR, "Harbor_seqtab2_tree.rds")

#Now to root the tree
fitGTR <- readRDS("Harbor_seqtab2_tree.rds")

#using most abundant archaea as outgroup
outgroupseq <- "CCGGTTCGTGCCCCTAGGCTTCGTCCCTGACCGTCGGATCCGTTCCAGTGTGGCGCCTTCGCAACTGGTGGTCCTCTAAGGATGACAC

phyloseq.tree <- root(fitGTR$tree, outgroup = outgroupseq, resolve.root = TRUE)

#saveRDS(phyloseq.tree, "Harbor_seqtab2_tree_archaea_rooted.rds")
```

Identify Contaminants

```
## Load into phyloseq

mapping <- read.csv("SG_Harbor_Proximity_metadata.csv")

row.names(mapping) <- mapping$X.SampleID

otu_table = otu_table(seqtab2, taxa_are_rows=FALSE)
mapping_file = sample_data(mapping)
taxa_table = tax_table(tax)
ps <- phyloseq(otu_table,mapping_file, taxa_table)
```

Using decontam to deal with our NC

We do not have input DNA concentration, so we are using decontom's "prevalence" method. In this method, the prevalence (presence/absence across samples) of each sequence feature in true positive samples is compared to the prevalence in negative controls to identify contaminants.

```
#first tell it which samples are the NC
sample_data(ps)$is.neg <- sample_data(ps)$X.SampleID == "BBNC"

#threshold=0.5, which will identify as contaminants
#all sequences thare are more prevalent in negative controls
#than in positive samples
```

```
contamdf.prev05 <- isContaminant(ps, method="prevalence", neg="is.neg", threshold=0.5)
table(contamdf.prev05$contaminant)

#which ASV is contaminant
head(which(contamdf.prev05$contaminant))
#seq 193, 508, 4963

#now we have 3 contaminants!
```

Looking at the number of times these taxa were observed in negative controls vs. real samples:

```
#Make phyloseq object of presence-absence in negative controls
ps.neg <- prune_samples(sample_data(ps)$is.neg == "TRUE", ps)
ps.neg.presence <- transform_sample_counts(ps.neg, function(abund) 1*(abund>0))

#Make phyloseq object of presence-absence in true positive samples
ps.pos <- prune_samples(sample_data(ps)$is.neg == "FALSE", ps)
ps.pos.presence <- transform_sample_counts(ps.pos, function(abund) 1*(abund>0))

#Make data.frame of prevalence in positive and negative samples
#using prev threshold = 0.5
df.pres <- data.frame(prevalence.pos=taxa_sums(ps.pos.presence),
                      prevalence.neg=taxa_sums(ps.neg.presence),
                      contam.prev=contamdf.prev05$contaminant)
ggplot(data=df.pres, aes(x=prevalence.neg, y=prevalence.pos, color=contam.prev)) + geom_point()
```

Get sequences that are contaminants

```
#Seq 193
contamdf.prev05[193,]
#CCTGTTTGCTCCCCACGCTTTCGAGCCTCAACGTCAGTTACAGTCCAGTAAGCCGCCTTCGCCACCGGTGTTCCTCCTAATATCTACGCATTTCACCGCTAC

#Seq 508
contamdf.prev05[508,]
#CCTGTTTGCTCCCCACGCTTTCGCACCTCAGCGTCAATACCAGTCCAGTGAGCCGCCTTCGCCACTGGTGTTCTTCCGAATATCTACGAATTTCACCTCTAC

#Seq 4963
contamdf.prev05[4963,]
#CCTGTTTGCTCCCCACGCTTTCGAGCCTCAGCGTCAGTTGCCGTCCAGTAAGCCGCCTCCGCCACCGGTGTTCTTCCTGATATCTGCGCATTTCACCGCTAC
```

We will want to remove these three ASVs from further analyses

```
#Make a list of contaminants
contaminants <- c("CCTGTTTGCTCCCCACGCTTTCGAGCCTCAACGTCAGTTACAGTCCAGTAAGCCGCCTTCGCCACCGGTGTTCCTCCTAATATC
                  "CCTGTTTGCTCCCCACGCTTTCGCACCTCAGCGTCAATACCAGTCCAGTGAGCCGCCTTCGCCACTGGTGTTCTTCCGAATATC
                  "CCTGTTTGCTCCCCACGCTTTCGAGCCTCAGCGTCAGTTGCCGTCCAGTAAGCCGCCTCCGCCACCGGTGTTCTTCCTGATATC
```

Analyze data in phyloseq

```
## Load into phyloseq
```

```
seqtab2 <- readRDS( "Harbor_seqtab_nochimeras.rds")
tax <- readRDS("Harbor_tax_silva132_w_sp.rds")
mapping <- read.csv("SG_Harbor_Proximity_metadata.csv")
phanghorn_tree <- readRDS("Harbor_seqtab2_tree_archaea_rooted.rds")

row.names(mapping) <- mapping$X.SampleID

otu_table = otu_table(seqtab2, taxa_are_rows=FALSE)
mapping_file = sample_data(mapping)
taxa_table = tax_table(tax)
tree = phy_tree(phanghorn_tree)
ps <- phyloseq(otu_table,mapping_file, taxa_table, tree)
```

Remove contaminants

```
#get all taxa names
allTaxa = taxa_names(ps)

#returns list of all taxa, except contaminants

allTaxa <- allTaxa[!(allTaxa %in% contaminants)]

#keeps taxa in allTaxa and gets rid of anthing else
ps_filt = prune_taxa(allTaxa, ps)
```

Remove Chloroplasts and Mitochondria

```
#removing chloroplasts, mitochondria FOR SILVA 132 - silva 128 has chloro as a Class
ps_filt_noChloro <- subset_taxa(ps_filt, Order !="o__Chloroplast")
ps_filt_noChloro_noMito <- subset_taxa(ps_filt_noChloro, Family != "f__Mitochondria")
```

Rarefaction to 6,000 reads per sample

```
ps.rare = rarefy_even_depth(ps_filt_noChloro_noMito, sample.size = 6000,
                            replace = FALSE, rngseed = 5311)
```

```
## `set.seed(5311)` was used to initialize repeatable random subsampling.

## Please record this for your records so others can reproduce.

## Try `set.seed(5311); .Random.seed` for the full vector

## ...

## 2 samples removedbecause they contained fewer reads than `sample.size`.

## Up to first five removed samples are:

## BB058BBNC

## ...
```

```
## 4510TUs were removed because they are no longer
## present in any sample after random subsampling

## ...
```

BBNC drops out when you rarefy to 6,000, but after removing contaminants, it had 0 reads. BB58 drops out, after removing contaminants + mitochondria + chloroplast, it only had 1339 reads.

Calculate RA per sample

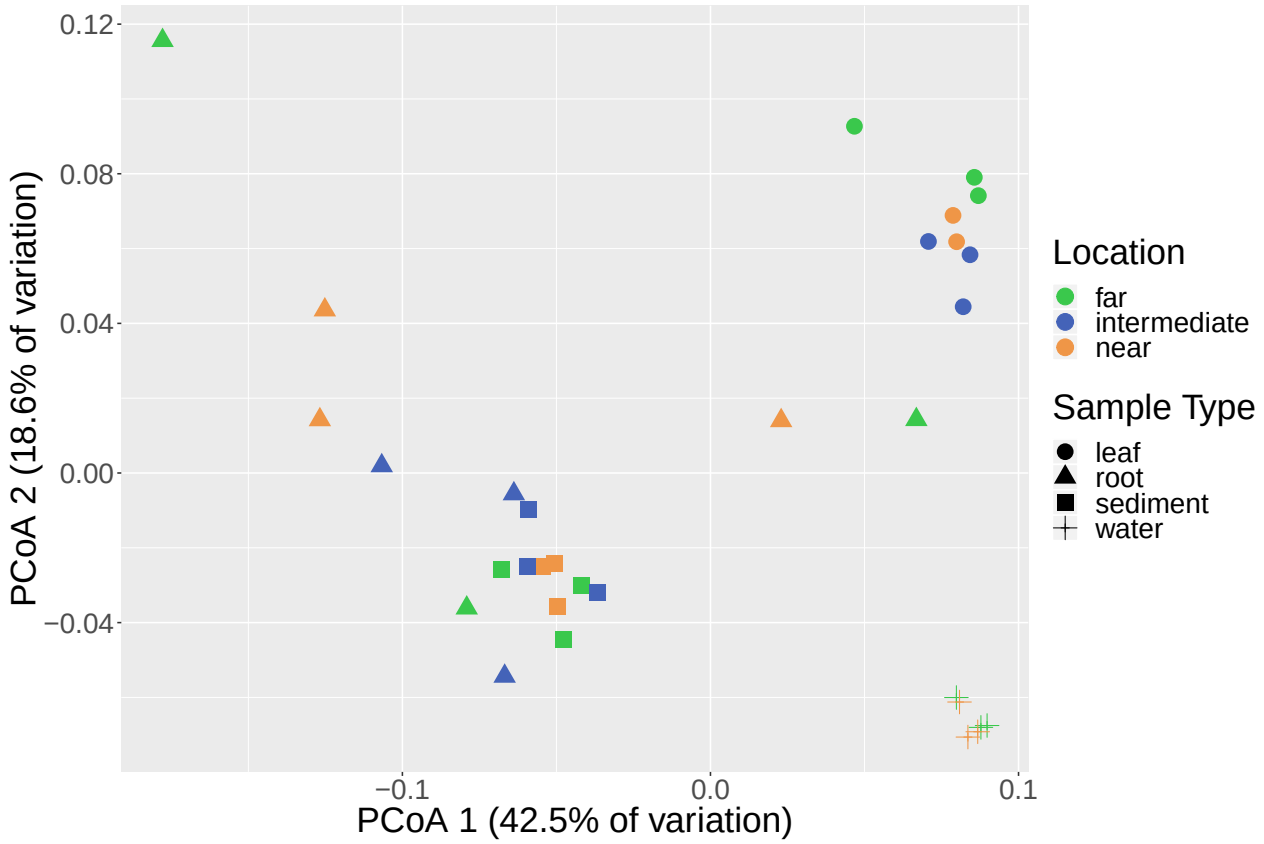
```
ps.rare.RA = transform_sample_counts(ps.rare, function(x) (x/6000))
```

Ordinate

```
ps.rare.ord <- ordinate(ps.rare, "PCoA", "wunifrac")
```

Figure 1

```
#PCoA plot
p = plot_ordination(ps.rare, ps.rare.ord, color="Location", shape="Substrate")
p = p + xlab("PCoA 1 (42.5% of variation)") + ylab("PCoA 2 (18.6% of variation)") +
                      geom_point(size = 5) + theme(text = element_text(size=24))
p + scale_color_manual(name="Location", labels = c("far", "intermediate", "near"),
                       values=c("#3AC84C", "#4463B8", "#EF9647"))+labs(shape="Sample Type")
```

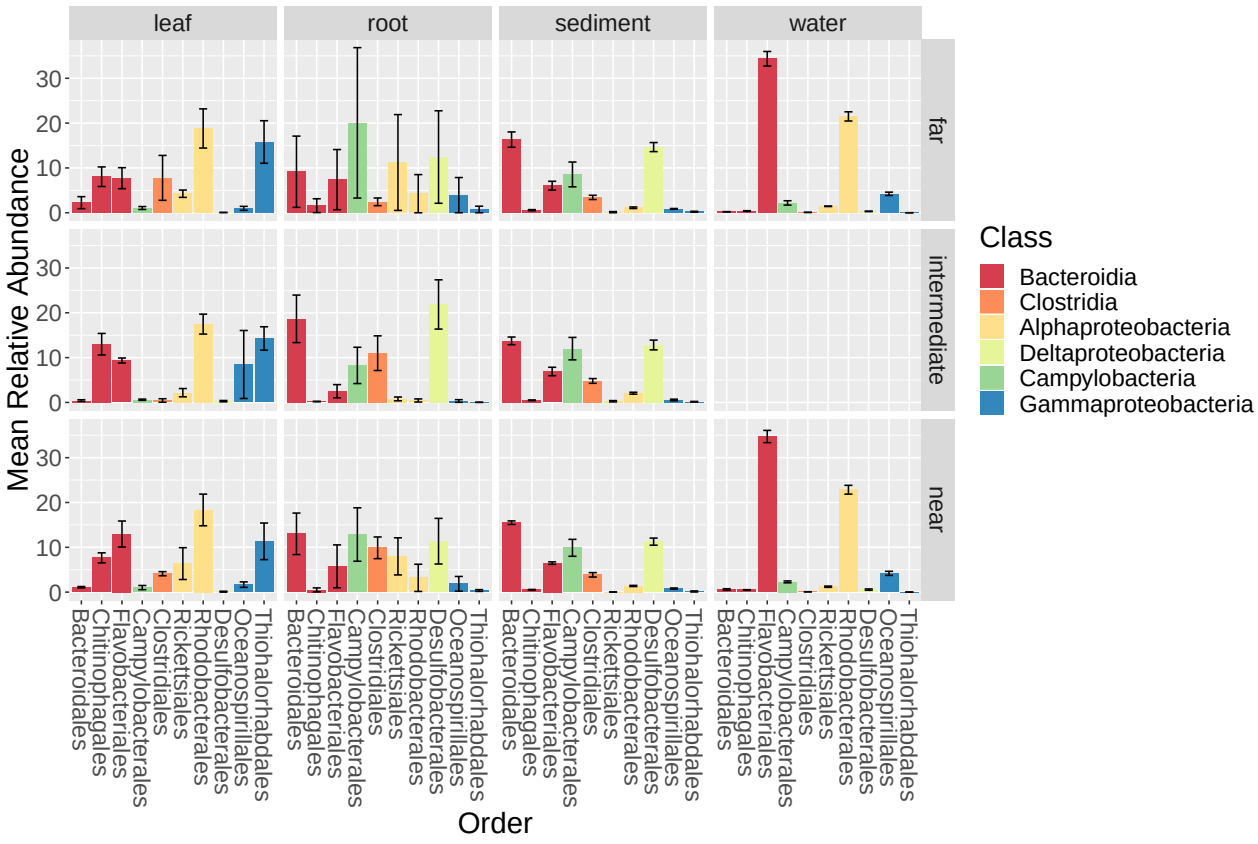


Ordination Stats

```
DistWU = phyloseq::distance(ps.rare, method = "wunifrac", type="samples")

adonis(DistWU ~ Location, as(sample_data(ps.rare), "data.frame"), permutations = 9999)
#Df SumsOfSqs   MeanSqs F.Model      R2 Pr(>F)
#Location   2   0.01850 0.0092485 0.56059 0.03722 0.8676
#Residuals 29   0.47844 0.0164978         0.96278
#Total     31   0.49693                   1.00000

adonis(DistWU ~ Substrate, as(sample_data(ps.rare), "data.frame"), permutations = 9999)
#Df SumsOfSqs  MeanSqs F.Model      R2 Pr(>F)
#Substrate  3   0.30340 0.101135  14.632 0.61055  1e-04 ***
#Residuals 28   0.19353 0.006912         0.38945
#Total     31   0.49693                  1.00000

adonis(DistWU ~ Location*Substrate, as(sample_data(ps.rare), "data.frame"), permutations = 9999)
#Df SumsOfSqs  MeanSqs F.Model      R2 Pr(>F)
#Location            2   0.01850 0.009249  1.2442 0.03722 0.2509
#Substrate           3   0.29606 0.098688 13.2763 0.59578 0.0001 ***
#Location:Substrate  5   0.02627 0.005255  0.7069 0.05287 0.8449
#Residuals          21   0.15610 0.007433         0.31413
#Total              31   0.49693                  1.00000
```

Taxonomic Bar Plots

```
#define standard error function
se <- function(x) sqrt(var(x)/length(x))

#collapse ASVs at order level
AvgRA_o = tax_glom(ps.rare.RA, taxrank="Order", NArm = FALSE)

#filter out taxa that don't vary > 0.2 %
AvgRA99_O = filter_taxa(AvgRA_o, function(x) var(x) > .002, TRUE)

df_o <- psmelt(AvgRA99_O)

#group and calculate mean, sd and se for different taxonomic levels
grouped_o <- group_by(df_o, Location, Substrate, Order, Class, Phylum)
avgs_o <- summarise(grouped_o, mean=100*mean(Abundance), sd=100*sd(Abundance), se=100*se(Abundance))

#change the key
locations <- c("water" = "water", "leaf" = "leaf", "root" = "root",
               "sediment" = "sediment", "far" = "far", "inter" = "intermediate", "near"="near")

#changing the order
avgs_o$Order <-factor(avgs_o$Order,
                      levels = c("Bacteroidales", "Chitinophagales", "Flavobacteriales",
                                 "Campylobacterales", "Clostridiales","Rickettsiales", "Rhodobacterales
                                 "Thiohalorhabdales"))
avgs_o$Class <-factor(avgs_o$Class,
                      levels = c("Bacteroidia","Clostridia", "Alphaproteobacteria",
                           "Deltaproteobacteria","Campylobacteria","Gammaproteobacteria"))
avgs_o$Phylum <- factor(avgs_o$Phylum,
                        levels = c("Bacteroidetes","Epsilonbacteraeota",
                                   "Firmicutes", "Proteobacteria"))

#create plot
p = ggplot(avgs_o, aes(x=Order, y= (mean), fill=Class)) +
                        geom_bar(stat="identity", position=position_dodge()) +
                        geom_errorbar(aes(ymin=(mean-se), ymax=(mean+se)),
                                      width=.4, position=position_dodge(.9))
p = p + facet_grid(Location~Substrate, labeller = as_labeller(locations)) +
                        theme(axis.text.x = element_text(angle = -90, hjust = 0, vjust=.5)) +
                        ylab("Mean Relative Abundance") + theme(text = element_text(size=20))
p + scale_fill_manual(values = colorRampPalette(brewer.pal(6, "Spectral"))(6)) +
                      scale_color_manual(values=colorRampPalette(brewer.pal(6,"Spectral"))(6))
```



Statistics - Taxonomy

Location

```
avgs_o$TP <- as.factor(avgs_o$Location)

avgs_o_tp <- melt(data.frame(avgs_o), id.vars=c("Order", "Location"),
                  measure.vars=c("mean"))

kruskal_test(mean ~ Location, distribution=approximate(B=9999), data=avgs_o)

for (i in 1:length(levels(df_o$Order))) {
  new_df <-  subset(df_o, df_o$Order == levels(df_o$Order)[i])
  new_df$LOC <- as.factor(new_df$Location)
  print(new_df$Order[1])
  print(kruskal_test(Abundance ~ LOC, distribution=approximate(B=9999), data=new_df))
  print( dunnTest(Abundance ~ LOC, data=new_df, method ="bonferroni"))}


#start
chisq = NULL
pvals = NULL
listofcats_sig = NULL
DataSet = df_o
taxa = unique(df_o$Order)
```

```
for (cat in taxa) {
  new_df <- subset(DataSet, DataSet$Order == cat)
  new_df$LOC <- as.factor(new_df$Location)
  kw = kruskal.test(Abundance ~ LOC, data=new_df)
  chisq = c(kw$statistic, chisq)
  pvals = c(kw$p.value, pvals)

  if (kw$p.value <= 0.05) {
    listofcats_sig = c(cat, listofcats_sig)

  }
}

pvals.bonf = p.adjust(pvals, method="bonferroni")
df_taxa = data.frame(rev(taxa), chisq, pvals, pvals.bonf)
write.table(df_taxa, 'Location_rare6000_Mean_Order_KW_002var.txt', sep="\t")

#No significant taxa
```

Sample Type

```
avgs_o$ST <- as.factor(avgs_o$Substrate)

avgs_o_st <- melt(data.frame(avgs_o), id.vars=c("Order", "Substrate"),
                  measure.vars=c("mean"))

kruskal_test(mean ~ ST, distribution=approximate(B=9999), data=avgs_o)

for (i in 1:length(levels(df_o$Order))) {
  new_df <-  subset(df_o, df_o$Order == levels(df_o$Order)[i])
  new_df$ST <- as.factor(new_df$Substrate)
  print(new_df$Order[1])
  print(kruskal_test(Abundance ~ ST, distribution=approximate(B=9999), data=new_df))
  print( dunnTest(Abundance ~ ST, data=new_df, method ="bonferroni"))}


#start
chisq = NULL
pvals = NULL
listofcats_sig = NULL
DataSet = df_o
taxa = unique(df_o$Order)

for (cat in taxa) {
  new_df <- subset(DataSet, DataSet$Order == cat)
  new_df$ST <- as.factor(new_df$Substrate)
  kw = kruskal.test(Abundance ~ ST, data=new_df)
  chisq = c(kw$statistic, chisq)
  pvals = c(kw$p.value, pvals)

  if (kw$p.value <= 0.05) {
```

```
    listofcats_sig = c(cat, listofcats_sig)

  }
}

pvals.bonf = p.adjust(pvals, method="bonferroni")
df_taxa = data.frame(rev(taxa), chisq, pvals, pvals.bonf)
write.table(df_taxa, 'SampleType_rare6000_Mean_Order_KW_002var.txt', sep="\t")


pvals.dunn = NULL
pvals.dunn.bonf = NULL
Zsc.dunn = NULL
comparison.dunn = NULL
cats.dunn = NULL

for (cat in listofcats_sig){
  new_df <- subset(DataSet, DataSet$Order == cat)
  new_df$ST <- as.factor(new_df$Substrate)
  dT = dunnTest(Abundance ~ ST, data = new_df, method = "bonferroni")

  for (i in 1:length(dT$res$Comparison)) {
    print(cat)

    pvals.dunn = c(dT$res$P.unadj[i], pvals.dunn)
    pvals.dunn.bonf = c(dT$res$P.adj[i], pvals.dunn.bonf)
    Zsc.dunn = c(dT$res$Z[i], Zsc.dunn)
    comparison.dunn = c(levels(dT$res$Comparison)[i], comparison.dunn)
    cats.dunn = c(cat, cats.dunn)

  }

}


df_taxa = data.frame(cats.dunn, comparison.dunn, Zsc.dunn, pvals.dunn, pvals.dunn.bonf)
write.table(df_taxa, 'SampleType_rare6000_Mean_Order_KW_002var_Dunn.txt', sep="\t")

#there are differences between sample types that are sig, but this is not surprising
```

Try subsetting by sample type (leaf, root, water or sediment) and then look for taxa that are significantly different across locations

```
Leaf <- subset_samples(AvgRA99_O, Substrate == "leaf")
Root <- subset_samples(AvgRA99_O, Substrate == "root")
Sediment <- subset_samples(AvgRA99_O, Substrate == "sediment")
Water <- subset_samples(AvgRA99_O, Substrate == "water")

df_l <- psmelt(Leaf)
df_r <- psmelt(Root)
df_s <- psmelt(Sediment)
df_w <- psmelt(Water)
```

```
#### LEAF x Location ####

#start
chisq = NULL
pvals = NULL
listofcats_sig = NULL
DataSet = df_l
taxa = unique(df_l$Order)

for (cat in taxa) {
  new_df <- subset(DataSet, DataSet$Order == cat)
  new_df$LOC <- as.factor(new_df$Location)
  kw = kruskal.test(Abundance ~ LOC, data=new_df)
  chisq = c(kw$statistic, chisq)
  pvals = c(kw$p.value, pvals)

  if (kw$p.value <= 0.05) {
    listofcats_sig = c(cat, listofcats_sig)

  }
}

pvals.bonf = p.adjust(pvals, method="bonferroni")
df_taxa = data.frame(rev(taxa), chisq, pvals, pvals.bonf)
write.table(df_taxa, 'LEAF_Location_rare6000_Mean_Order_KW_002var.txt', sep="\t")

#no sig difference in taxa on leaves x locations


#### ROOT x Location ####

#start
chisq = NULL
pvals = NULL
listofcats_sig = NULL
DataSet = df_r
taxa = unique(df_r$Order)

for (cat in taxa) {
  new_df <- subset(DataSet, DataSet$Order == cat)
  new_df$LOC <- as.factor(new_df$Location)
  kw = kruskal.test(Abundance ~ LOC, data=new_df)
  chisq = c(kw$statistic, chisq)
  pvals = c(kw$p.value, pvals)

  if (kw$p.value <= 0.05) {
    listofcats_sig = c(cat, listofcats_sig)

  }
}

pvals.bonf = p.adjust(pvals, method="bonferroni")
```

```
df_taxa = data.frame(rev(taxa), chisq, pvals, pvals.bonf)
write.table(df_taxa, 'ROOT_Location_rare6000_Mean_Order_KW_002var.txt', sep="\t")

#no sig difference in taxa on roots x locations


#### SEDIMENT x Location ####

#start
chisq = NULL
pvals = NULL
listofcats_sig = NULL
DataSet = df_s
taxa = unique(df_s$Order)

for (cat in taxa) {
  new_df <- subset(DataSet, DataSet$Order == cat)
  new_df$LOC <- as.factor(new_df$Location)
  kw = kruskal.test(Abundance ~ LOC, data=new_df)
  chisq = c(kw$statistic, chisq)
  pvals = c(kw$p.value, pvals)

  if (kw$p.value <= 0.05) {
    listofcats_sig = c(cat, listofcats_sig)

  }
}

pvals.bonf = p.adjust(pvals, method="bonferroni")
df_taxa = data.frame(rev(taxa), chisq, pvals, pvals.bonf)
write.table(df_taxa, 'SEDIMENT_Location_rare6000_Mean_Order_KW_002var.txt', sep="\t")

#no sig difference in taxa on sediment x locations


#### LEAF x Location ####

#start
chisq = NULL
pvals = NULL
listofcats_sig = NULL
DataSet = df_w
taxa = unique(df_w$Order)

for (cat in taxa) {
  new_df <- subset(DataSet, DataSet$Order == cat)
  new_df$LOC <- as.factor(new_df$Location)
  kw = kruskal.test(Abundance ~ LOC, data=new_df)
  chisq = c(kw$statistic, chisq)
  pvals = c(kw$p.value, pvals)

  if (kw$p.value <= 0.05) {
    listofcats_sig = c(cat, listofcats_sig)
```

```
  }
}

pvals.bonf = p.adjust(pvals, method="bonferroni")
df_taxa = data.frame(rev(taxa), chisq, pvals, pvals.bonf)
write.table(df_taxa, 'WATER_Location_rare6000_Mean_Order_KW_002var.txt', sep="\t")

listofcats_sig
#Bacteroidales
#1 sig difference here for Bacteroidales, BUT not significant after bonferroni correction
```