Grey Patch Disease in Coral and 16S Analysis of Bacteria

Susritha Kopparapu, Isabel Novick, Corinne Vietorisz

3/9/2021

*The following script is modified from this* [*tutorial*](https://benjjneb.github.io/dada2/tutorial.html) *and the edits by Carly D. Kenkel & Alizah Ali & Nicola Kriefall & Sarah Davies*

## Introduction

Coral reefs are increasingly threatened by climate change, thus the need to identify the diversity of the microbial communities of coral before they die off is becoming more urgent in order to understand the roles their microbial communities play in coral health. Microbial communities may affect host coral fitness, and coral disease may influence the integrity of their microbiome, exacerbating the negative effects. We sought to examine how coral microbial communities change after outbreaks of grey-patch disease.

Our data is from Sweet et al. (2019), “Compositional homogeneity in the pathobiome of a new, slow-spreading coral disease.” We identified and examined the bacterial communities on healthy and diseased corals using 16S sequencing and DNA barcoding. A reanalysis of the Sweet et al. (2019) data was performed using the dada2 pipeline in R followed by the assigning of microbial taxonomy with the Silva 16S database. We found that diseased colonies have a higher abundance of cyanobacteria than healthy colonies, in accordance with the findings of Sweet et al (2019). We additionally found that healthy colonies had higher abundances of gammaproteobacteria compared to diseased colonies.

## Setup

#### R Version

R version 4.0.2 was used for this analysis.

#### Packages

The following packages were used to clean, analyze, and visualize data. Packages were installed with [Bioconductor](https://bioconductor.org/biocLite.R) version 3.12 using the vegan package.

library(dada2); #Version 1.18.0  
library(ShortRead); #Version 1.48.0  
library(ggplot2); #Version 3.3.3  
library(phyloseq); #Version 1.34.0

#### Data Retrieval

[Sweet et. al 2019](https://microbiomejournal.biomedcentral.com/articles/10.1186/s40168-019-0759-6) collected samples from healthy and diseased *Porites lobata*, as well as the water column of the Luminao reef flat in Guam (in 2016). Their data that was reanalyzed here was retrieved from [NCBI](https://www.ncbi.nlm.nih.gov/sra?LinkName=bioproject_sra_all&from_uid=448821).

Samples diseased 13, healthy 11, healthy 4, diseased 14, and water column 2 were selected. All files in the directory were put in the variable fns.

path <- "/projectnb/bi594/skoppara/Assignment1"  
fns <- list.files(path)  
fns

## Trimming and Filtering Data

We filtered through all the files to find only the fastq files. We then sorted them so that the reads are in the same order.

fastqs <- fns[grepl(".fastq$", fns)]  
fastqs <- sort(fastqs)   
fastqs

## [1] "SRR6946098\_1.fastq" "SRR6946099\_1.fastq" "SRR6946100\_1.fastq"  
## [4] "SRR6946101\_1.fastq" "SRR6946115\_1.fastq"

The file names are split and stored in another variable as sample names without the .fastq file extension.

sample.names <- sapply(strsplit(fastqs, ".fastq"), `[`, 1)   
fnFs <- file.path(path, fastqs)

## Sequence Verification

*code adapted from* [*Nicola Kriefall*](https://github.com/Nicfall/moorea_holobiont/blob/master/mr_16S/mr16s.R)

The forward and reverse primers as defined from the Sweet et. al paper are shown below.

FWD <- "TATGGTAATTGTCTCCTACTTRRSGCAGCAG"   
REV <- "AGTCAGTCAGCCGGACTACNVGGGTWTCTAAT"

The below function takes in a character vector primer and creates all the orientations of that sequence. The vector is converted to a DNA String object using the Biostrings package. The DNA string object can be used to evaluate the complement of the sequences. All orients are converted back to character vectors upon return.

allOrients <- function(primer) {  
 require(Biostrings)  
 dna <- DNAString(primer)   
 orients <- c(Forward = dna, Complement = complement(dna), Reverse = reverse(dna),   
 RevComp = reverseComplement(dna))  
 return(sapply(orients, toString))   
}

The forward and reverse orients are as follows.

FWD.orients <- allOrients(FWD)  
REV.orients <- allOrients(REV)  
FWD.orients

## Forward Complement   
## "TATGGTAATTGTCTCCTACTTRRSGCAGCAG" "ATACCATTAACAGAGGATGAAYYSCGTCGTC"   
## Reverse RevComp   
## "GACGACGSRRTTCATCCTCTGTTAATGGTAT" "CTGCTGCSYYAAGTAGGAGACAATTACCATA"

REV.orients

## Forward Complement   
## "AGTCAGTCAGCCGGACTACNVGGGTWTCTAAT" "TCAGTCAGTCGGCCTGATGNBCCCAWAGATTA"   
## Reverse RevComp   
## "TAATCTWTGGGVNCATCAGGCCGACTGACTGA" "ATTAGAWACCCBNGTAGTCCGGCTGACTGACT"

fnFs.filtN <- file.path(path, "filtN", basename(fnFs))  
filterAndTrim(fnFs, fnFs.filtN, maxN = 0, multithread = TRUE)

The below function takes in a primer sequence and a read. It counts the number of reads in which the primer is found and returns this value.

primerHits <- function(primer, fn) {  
 nhits <- vcountPattern(primer, sread(readFastq(fn)), fixed = FALSE)  
 return(sum(nhits > 0))  
}  
primerHits

The below tables show that the primers were removed from the sequences already. Therefore, no left trim is necessary when filtering the actual sequences.

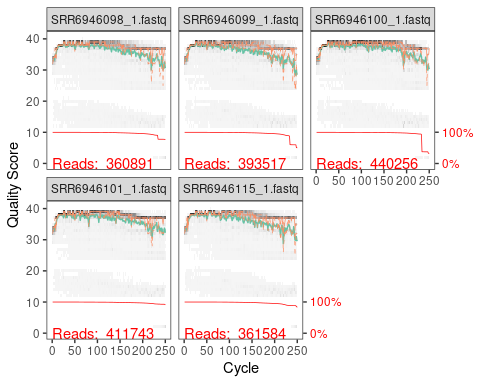
rbind(FWD.ForwardReads = sapply(FWD.orients, primerHits, fn = fnFs.filtN[[3]]),   
 FWD.ReverseReads = sapply(FWD.orients, primerHits, fn = fnFs.filtN[[3]]),   
 REV.ForwardReads = sapply(REV.orients, primerHits, fn = fnFs.filtN[[3]]),   
 REV.ReverseReads = sapply(REV.orients, primerHits, fn = fnFs.filtN[[3]]))

## Forward Complement Reverse RevComp  
## FWD.ForwardReads 0 0 0 0  
## FWD.ReverseReads 0 0 0 0  
## REV.ForwardReads 0 0 0 0  
## REV.ReverseReads 0 0 0 0

#### Visualize Raw Data

The quality plot below shows that the quality stays above or at 20 for the whole length of the sequence, however, around 200 the quality does start to drop to 20. For this reason, the truncation length was chosen to be 200.

plotQualityProfile(fnFs[c(1,2,3,4,5,6,7,8,9)])



**Figure 1.** Quality plots of reads from all samples.

A directory is made for the filtered files and filenames using the following code.

filt\_path <- file.path(path, "trimmed")  
if(!file\_test("-d", filt\_path)) dir.create(filt\_path)  
filtFs <- file.path(filt\_path, paste0(sample.names, "\_F\_filt.fastq.gz"))

The truncation length was defined to 200. No Ns are allowed and expected errors allowed was 1. All other settings were set to the defaults.

out <- filterAndTrim(fnFs, filtFs, truncLen= 200,  
 maxN=0,   
 maxEE=1,   
 truncQ=2,   
 rm.phix=TRUE,   
 compress=TRUE,   
 multithread=FALSE)

## Error Rates

To estimate error rates, the code below was used. The maximum number of cycles was increased to 30 to allow convergence. Convergence was found after 4 rounds.

setDadaOpt(MAX\_CONSIST=30)  
errF <- learnErrors(filtFs, multithread=TRUE)

## Dereplicate Reads and Infer Sequence Variants

In order to increase DADA2’s accuracy and avoid extraneous comparisons, the following code was used for dereplication and to create the consensus quality profile.

derepFs <- derepFastq(filtFs, verbose=TRUE)  
names(derepFs) <- sample.names

To infer sequence variants, we used a band size of 16 for our 16S data. 16 is the default band size and is typically used for 16S data.

setDadaOpt(BAND\_SIZE=16)  
dadaFs <- dada(derepFs, err=errF, multithread=TRUE)

The below shows how many real sequence variants are present.

dadaFs[[1]]  
dadaFs[[5]]

A sequence table was constructed below.

seqtab <- makeSequenceTable(dadaFs)

## Remove Chimeras

To further increase the accuracy of the analysis and provide a sequence that is representative of what exists in nature, the following code was used to remove chimeras. 1 chimera was identified out of 117 input sequences.

seqtab.nochim <- removeBimeraDenovo(seqtab, method="consensus", multithread=TRUE, verbose=TRUE)  
dim(seqtab.nochim)

The ratio of normal reads to sequence variants was 0.9998201, as there was only 1 chimera.

sum(seqtab.nochim)/sum(seqtab)

Then, we generate a .csv file of the data without sequence variants.

write.csv(seqtab,file="16s\_seqtab.csv")  
write.csv(seqtab.nochim,file="16s\_nochim.csv")

## Track Read Stats

Lastly, we use the following code to track read stats, and save it into a final .csv file. This file contains the cumulative filtered, trimmed, denoised, and read data.

getN <- function(x) sum(getUniques(x))  
track <- cbind(out, sapply(dadaFs, getN), sapply(dadaFs, getN), rowSums(seqtab), rowSums(seqtab.nochim))  
colnames(track) <- c("input", "filtered", "denoised", "merged", "tabled", "nonchim")  
rownames(track) <- sample.names  
head(track)  
tail(track)  
  
write.csv(track,file="ReadFilterStats\_AllData\_final.csv",row.names=TRUE,quote=FALSE)

## Assign Taxonomy

To assign each sequence its taxa, we compare the sequences to a reference database containing 16S sequences and corresponding taxonimic information for each sequence. We chose the Silva database because it contains a comprehensive library of 16S sequences and is specifically formatted for Dada2.

*Citation:* Michael R. McLaren. (2020). Silva SSU taxonomic training data formatted for DADA2 (Silva version 138) (Version 2) [Data set]. Zenodo. <http://doi.org/10.5281/zenodo.3986799>

First, we classify our sequences according to our reference database with the assignTaxonomy function. This assigns taxonomic classifications to each sequence in each sample up to the Genus level.

taxa <- assignTaxonomy(seqtab.nochim, "/projectnb/bi594/skoppara/Assignment1/silva\_nr99\_v138\_train\_set.fa", minBoot = 50, multithread=TRUE, tryRC=TRUE)

Then, we add species-level annotation to the taxonomic table with the addSpecies function.

taxa <- addSpecies(taxa, "/projectnb/bi594/skoppara/Assignment1/silva\_species\_assignment\_v138.fa.gz")

We next generate a .csv file to save the taxonomic table.

write.csv(taxa, file="taxa.csv",row.name=TRUE,quote=FALSE)  
unname(head(taxa, 30))  
unname(taxa)

## Visualize Data

Let’s read in our sequence table, associated taxa list, and associated metadata.

seqtab.nochim <- readRDS("final\_seqtab\_nochim.rds")  
taxa <- readRDS("final\_taxa\_blastCorrected.rds")  
samdf<-read.csv("variabletable.csv")  
rownames(samdf) <- samdf$sample

Here, we make a phyloseq object with the OTU table, sample data, and taxonomy table.

ps <- phyloseq(otu\_table(seqtab.nochim, taxa\_are\_rows=FALSE),   
 sample\_data(samdf),   
 tax\_table(taxa))  
ps

## phyloseq-class experiment-level object  
## otu\_table() OTU Table: [ 1215 taxa and 5 samples ]  
## sample\_data() Sample Data: [ 5 samples by 3 sample variables ]  
## tax\_table() Taxonomy Table: [ 1215 taxa by 7 taxonomic ranks ]

We replace the sequences with shorter names.

ids<-taxa\_names(ps)  
ids <- paste0("sq",seq(1, length(colnames(seqtab.nochim))))  
colnames(seqtab.nochim) <- ids

We create a phyloseq object with just the top 90 taxa.

top90 <- names(sort(taxa\_sums(ps), decreasing=TRUE))[1:90]  
ps.top90 <- transform\_sample\_counts(ps, function(OTU) OTU/sum(OTU))  
ps.top90 <- prune\_taxa(top90, ps.top90)  
ps.top90

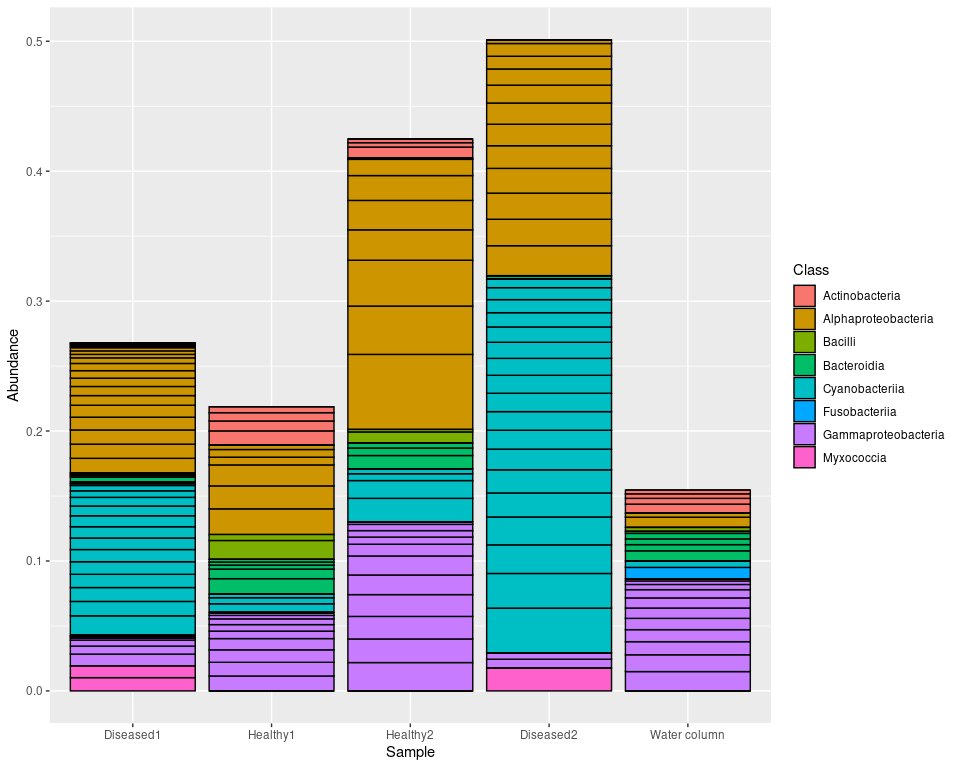
## phyloseq-class experiment-level object  
## otu\_table() OTU Table: [ 90 taxa and 5 samples ]  
## sample\_data() Sample Data: [ 5 samples by 3 sample variables ]  
## tax\_table() Taxonomy Table: [ 90 taxa by 7 taxonomic ranks ]

Now, we turn the phyloseq object containing our top 90 taxa into a datatable that will allow us to create graphics in ggplot. With this datatable, we make a .csv file for the phyloseq data. This contains the abundances of each individual in each sample by Class.

psz <- psmelt(ps.top90)  
write.csv(psz, file="Phyloseqoutputfinal.csv")

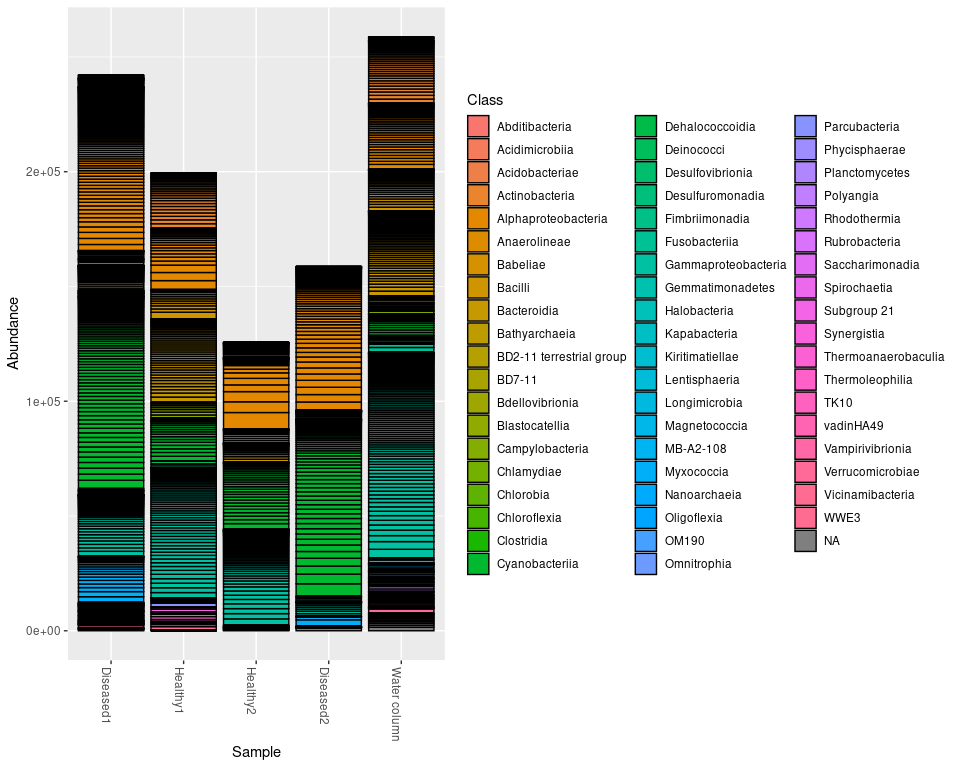
Now, we make bar plots representing the abundances of the 90 most abundant taxa by sample, separated by Class.

p <- ggplot(psz, aes(x = Sample, y=Abundance, fill=Class))  
p + geom\_bar(stat="identity", colour="black") +  
 scale\_x\_discrete(labels=c('Diseased1', 'Healthy1', "Healthy2", "Diseased2", "Water column"))

 Note that the diseased samples have more cyanobacteria than the healthy samples, and the healthy samples have more gammaproteobacteria than the diseased samples. Here, we have also included a sample from the water column as a reference to see what taxa are already present in the water.

To plot counts rather than abundances, we create bar plots of the counts of all the individuals found in each Class by sample.

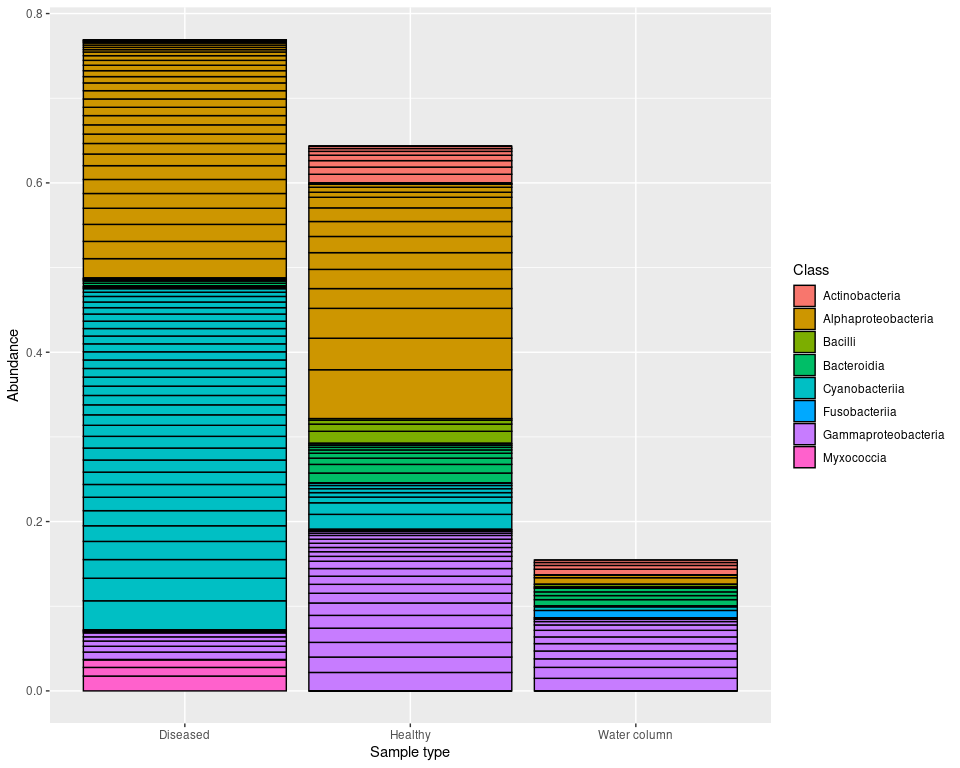
p2<- plot\_bar(ps, x = "Sample", fill= "Class")  
p2<- p2 + scale\_x\_discrete(labels=c('Diseased1', 'Healthy1', "Healthy2", "Diseased2", "Water column"))  
p2



**Figure 3.** Counts of all bacterial taxa by sample

To more clearly see how bacterial communities vary between healthy and diseased corals, we combine the healthy and diseased samples to plot taxa abundances by sample type.

p3 <- ggplot(psz, aes(x = type, y=Abundance, fill=Class))  
p3 + geom\_bar(stat="identity", colour="black") +  
 labs(x="Sample type") +  
 scale\_x\_discrete(labels=c('Diseased', 'Healthy', "Water column"))



**Figure 4.** Bacterial taxa differ between healthy corals, diseased corals, and the water column.

## Conclusion

Bacterial communities differed considerably between individual samples, with much greater bacterial abundance and diversity in coral-associated samples than the water column. We found that diseased colonies have a higher abundance of cyanobacteria than healthy colonies, in accordance with the findings of Sweet et al (2019). We additionally found that healthy colonies had higher abundances of gammaproteobacteria compared to diseased colonies. Overall, these results show that coral-associated bacterial communities differ between healthy and diseased corals.