Jul 03, 2025

16S/18S Amplicon Sequence Processing and Microbial Community Analysis

DOI
https://dx.doi.org/10.17504/protocols.io.14egnrbwzl5d/v1

Margaret Rettig[1], Christopher Ward[1]

[1]Bowling Green State University

MicrobialEcology@Ward...

Margaret Rettig
Bowling Green State University

DOI: https://dx.doi.org/10.17504/protocols.io.14egnrbwzl5d/v1

Protocol Citation: Margaret Rettig, Christopher Ward 2025. 16S/18S Amplicon Sequence Processing and Microbial Community Analysis. **protocols.io** https://dx.doi.org/10.17504/protocols.io.14egnrbwzl5d/v1

Protocol status: Working
We use this protocol and it's working

Created: July 01, 2025

Last Modified: July 03, 2025

Protocol Integer ID: 221427

Keywords: amplicon sequence processing, microbial community analysis adapted tutorial for the processing, end fastq file, microbial community analysis adapted tutorial

Abstract

Adapted tutorial for the processing and initial analysis of 16S/18S illumina-sequenced paired-end fastq files.

R Commands

1 Formatting guidelines:

Copy and paste the script below into a R notebook or a Markdown file, delineating chunks with ```{r} to open and ``` to close each chunk. When you run lines from a chunk, the results will show up at the bottom of the chunk, so you may find that smaller chunks work well for your purposes.

Bolded text is text that you will need to refine based on your particular sequencing data, computer setup, etc.

Library loading and workspace setup

2 Set up workspace -- load required packages and libraries

These steps download and load necessary packages, you can skip this step if you already have these installed.

```
install.packages(c("ggplot2", "plyr", "dplyr", "vegan", "scales",
"grid", "reshape2", "RColorBrewer", "devtools", "BiocManager",
"rmarkdown")

library(BiocManager)

BiocManager::install(c("dada2", "phyloseq"))
```

The Bioconductor Manger install command may have to be run separately.

Sequence Processing

3 Load the necessary packages. It is easiest to do this all together at the beginning. The packages loaded here will support this workflow, but additional analysis may need additional packages.

```
library(ggplot2)
library(dplyr)
library(vegan)
library(reshape2)
library(RColorBrewer)
library(rmarkdown)

library(devtools)
library(dada2)
library(phyloseq)
theme_set(theme_bw())
```

Note

If sequencing is performed in multiple runs, it will have different quality and error profiles. Sequence processing should be run separately and can be merged at the end of at the end of the sequence processing steps.

The sequence files should be placed in different fastq_files folders to keep them separated.

4 Locate the folder where your sequence read files are and set that as your path.

```
path <- "/replace-with-your-path-to-fastq-file/fastq_files" #set
path to the directory containing the 16S/18S fastq files
list.files(path) # readout of files in path directory

# Forward and reverse fastq filenames should have format:
SAMPLENAME_R1_001.fastq and SAMPLENAME_R2_001.fastq.gz (if other
suffix, update the patterns below)

fnFs <- sort(list.files(path, pattern="_R1_001.fastq.gz",
full.names = TRUE))
fnRs <- sort(list.files(path, pattern="_R2_001.fastq.gz",
full.names = TRUE))

# Extract sample names, assuming filenames have format:
SAMPLENAME_XXX.fastq.gz
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
```

A few notes on copying the path for the file directory:

- On a Windows computer, copying the file path will contain backslashes (\) instead of frontslashes (/). These backslashes should be changed to front slashes before running the code.
- The easiest way to copy the file path on a Mac is to open the Terminal application (in applications), and drag and drop the file into the Terminal. This will return the file path that you can then copy and paste.

5 Examine your sequences

```
plotQualityProfile(fnFs[1:2]) #add observations about sequence
quality here
plotQualityProfile(fnRs[1:2]) #add observations here
```

6 Trim primers and low-quality base calls, and filter out reads with low overall quality.

```
# Filtered files to go into new subdirectory
filtFs <- file.path(path, "filtered", paste0(sample.names,
"_F_filt.fastq"))  # to write filtered F reads into new file with
provided suffix
filtRs <- file.path(path, "filtered", paste0(sample.names,
"_R_filt.fastq"))  # to write filtered R reads into new file with
provided suffix

# Standard filtering parameters: maxN=0 (DADA2 requirement),
truncQ=2, rm.phix=TRUE and maxEE=2
# Additional argument for trimming 22-nt primer sequences from
start of sequences
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs,
              trimLeft = c(17,21), truncLen=c(275,225),
              maxN=0, maxEE=2, truncQ=2, rm.phix=TRUE,
              compress=TRUE, multithread=TRUE)
head(out)   # removes ~XX% reads
```

The trimLeft argument will need to be changed to trim the specific primers used in the sequencing run. The truncLen argument may need to be adjusted to allow for differences in sequence quality based on the quality profile plots generated above.

For additional help: Check out the **filterAndTrim** in dada2 documentation to see what each argument is doing and what other available arguments could be added. Removing arguments from the above command will revert them to default setting. Using a single value for an argument will apply it to both F and R; using c(X,Y) allows you to use different values for F and R.

Note

Primers:
The Earth Microbiome Project primer set has a 19nt-long forward primer and a 20-nt long reverse primer, while Klindworth primer set has a 17nt-long forward primer and a 21-nt long reverse primer. You want to trim the primers from the left side of forward and reverse reads. Leaving primers may result in sequences as low-quality or chimeric in later steps and removed, altering the composition of the resulting communities.

7 Learn error rates for DADA2 error model on all sequence reads before dereplicating libraries.

```
errF <- learnErrors(filtFs, multithread=TRUE) # 100548862 total bases in 439078 reads from 51 samples will be used for learning the error rates. (replace with information specific to your data set)
errR <- learnErrors(filtRs, multithread=TRUE) # 100548862 total bases in 439078 reads from 51 samples will be used for learning the error rates. (replace with information specific to your data set)

plotErrors(errF, nominalQ=TRUE)  # observations of error frequencies in F reads
plotErrors(errR, nominalQ=TRUE)  # observations of error frequencies in R reads

dada2:::checkConvergence(errF)
dada2:::checkConvergence(errR)
```

Note

Error correction:
The biggest difference between DADA2 and earlier OTU clustering techniques is that DADA2 predicts sequencing errors from the Illumina reads and corrects for them, rather than using a 97-99% sequence similarity to gloss over errors.

8 Dereplicate libraries to speed up next steps

```
derepFs <- derepFastq(filtFs, verbose=FALSE)
derepRs <- derepFastq(filtRs, verbose=FALSE)
names(derepFs) <- sample.names  # name F derep objects by the
sample names
names(derepRs) <- sample.names  # name R derep objects by the
sample names
```

9 Sequence inference with DADA2 error correction model

```
dadaFs <- dada(derepFs, err=errF, multithread=TRUE, verbose=FALSE)
dadaFs[[1]]  # 216 sequence variants were inferred from 5581 input
unique sequences. (replace with information specific to your data)
dadaRs <- dada(derepRs, err=errR, multithread=TRUE, verbose=FALSE)
dadaRs[[1]]  # 216 sequence variants were inferred from 5902 input
unique sequences
```

10 Merge paired-end reads

```
mergers <- mergePairs(dadaFs, derepFs, dadaRs, derepRs,
verbose=TRUE)
```

Note

When analyzing 18S data, merging the forward and reverse sequences may not be possible. If this step is run and the console output indicates that few merged pairs are being generated, there is likely not enough sequence overlap to generate merged reads.

Most reads should merge sucessfully. If not, the upstream parameters may need to be altered to ensure that overlap between reads was not trimmed away.

If reads cannot be merged, continue using the forward reads in place of the merged reads.

11 Construct sequence table

```
seqtab <- makeSequenceTable(mergers) #if reads could not be
merged, replace mergers with the dadaFs object
dim(seqtab) # 100 7682 # Inspect distribution of sequence lengths
table(nchar(getSequences(seqtab)))  # spread from 230-337 nts
long, most between 248-254 (will change for different data sets)
(expected values vary based on target organisms, sequencing
method, and amplified region, but check to make sure the values
make sense for your data)

# Optional: remove sequences that are shorter/longer than expected
amplicon from sequence table
seqtab2 <- seqtab[,nchar(colnames(seqtab)) %in% seq(244,257)]  #
2nd seq table removing short and long seqs
```

Sequences that are much longer or shorter than expected may be due to non-specific priming. If this occurs in your data set, it is best to remove it to verify you are getting amplicons of the targeted length.

12 Remove chimeras

When using DADA2 to generate ASVs rather than other methods that use OTUs, chimeric sequences are identified if they can be exactly reconstructed by combining a left-segment and a right-segment from two more abundant “parent” sequences.

```
seqtab.nochim <- removeBimeraDenovo(seqtab, method="consensus",
multithread=TRUE, verbose=TRUE)
dim(seqtab.nochim) # 100 7626
sum(seqtab.nochim)/sum(seqtab) # 0.9971214

Optional step if you elminated longer/shorter sequences in the
previous step
seqtab2.nochim <- removeBimeraDenovo(seqtab2, method="consensus",
multithread=TRUE, verbose=TRUE)
dim(seqtab2.nochim) # 100 7611 -- not too different from seqtab
sum(seqtab2.nochim)/sum(seqtab2) # 0.9971212
```

Most of your reads should remain after the chimera removal, but a majority of sequence variants may be removed. If a large number of reads were removed as chimeric, then upstream processing parameters may need to be reevaluated. The likely cause of this is the failure to completely remove primer sequences, especially ones containing ambiguous nucleotides not fully removed in the trimming step.

13 Track reads through the pipeline

This step will track how many reads from each library are removed in the quality control steps. A majority of the raw reads should be kept, and no one step, especially after filtering, should result in a very large drop in read counts.

```
getN <- function(x) sum(getUniques(x))
track <- cbind(out, sapply(dadaFs, getN), sapply(dadaRs, getN),
sapply(mergers, getN), rowSums(seqtab.nochim))
# If processing a single sample, remove the sapply calls: e.g.
replace sapply(dadaFs, getN) with getN(dadaFs)
colnames(track) <- c("input", "filtered", "denoisedF",
"denoisedR", "merged", "nonchim")
rownames(track) <- sample.names
head(track)
```

Note

Sanity check:
No step after filtering should result in the loss of a majority of the reads.
If a majority of reads failed to merge, the truncLen parameter in the trimming and filtering step should be reevaluated to make sure that the trimmed reads still encompass the amplicon overlap. This may not be possible for 18S data, if that is the case use the forward reads only.

If the majority of reads are removed as chimeric, there are likely still primer sequences hanging around, you may need to ensure that all primer sequences were removed, as the ambiguous nucleotides in primers interfere with the chimera detection.

14 Assign taxonomy to sequences

For this, you may want to use multiple reference databases. The microbial taxa of some sample types or environments may have better representation or resolution in one database than others; this might be known in which case you can choose a priori which database to use, if not you can run multiple taxonomy assignments and compare the results. Some classifiers have supplementary species assignment databases -- approach with caution as representation of species references may be spotty for non-human-associated taxa.

Databases for taxonomy assignment using the assignTaxonomy function in DADA2 will need to be specifically formatted for use with this function. Links to download several

appropriate classifiers are located here, but additional classifiers for various purposes are available online.

Note

Depending on the size of your data and the database you are using, this step can require a large amount of memory and take a long time to run. You may want to set up your computer plugged in overnight to run this step, particularly if you are using the PR2 database for 18S data. Computers with less memory will likely struggle with PR2 and 18S data, my personal recommendation is a computer with at least 16 GB of memory, if not more.

Taxonomy assignment for 16S data using RDP and SILVA, optional species assignment step:

```
RDP assignment:
taxa.rdp <- assignTaxonomy(seqtab.nochim,
"/Users/cward/Documents/seqproc/tax/rdp_train_set_16.fa.gz",
multithread=TRUE) #replace the file path with the path to your
downloaded training set
taxa.rdp.print <- taxa.rdp # Removing sequence rownames for
display only
rownames(taxa.rdp.print) <- NULL
head(taxa.rdp.print)

Optional addition of species assignment:
taxa.rdp <- addSpecies(taxa.rdp,
"/Users/ward56/Documents/seqproc/tax/rdp_species_assignment_16.fa.
gz")

SILVA assignment:
taxa.slv <- assignTaxonomy(seqtab.nochim,
"/Users/cward/Documents/seqproc/tax/silva_nr_v132_train_set.fa.gz"
, multithread=TRUE) # replace file path with the path to your
downloaded training set
taxa.slv.print <- taxa.slv # Removing sequence rownames for
display only
rownames(taxa.slv.print) <- NULL
head(taxa.slv.print)
```

Databases appropriate for use with 16S (prokaryotic) data:
SILVA (version 138.2+) https://zenodo.org/records/14169026
RDP (RDP release 19+) https://zenodo.org/records/14168771 **also includes an assignSpecies training set

Taxonomy assignment for 18S data using PR2:

```
PR2 assignment:
taxa.pr2 <- assignTaxonomy(seqtab.nochim, "path/to-downloaded-pr2-database", multithread = TRUE, taxLevels =
c("Domain","Supergroup","Division","Subdivision",
"Class","Order","Family","Genus","Species"))
taxa.pr2.print <- taxa.pr2
rownames(taxa.pr2.print) <- NULL
head(taxa.pr2.print)
```

Databases appropriate for use with 18S (eukaryotic) data:
PR2 (version 5.0.0+) https://github.com/pr2database/pr2database/releases

- **Important note:** PR2 has different taxLevels than the dada2 default. When assigning taxonomy against PR2, use the following code in the assignTaxonomy function (for PR2 v5.0.0+):

```
assignTaxonomy(..., taxLevels =
c("Domain","Supergroup","Division","Subdivision",
"Class","Order","Family","Genus","Species"))
```

SILVA (v128 and v132, maintained separately from the bacterial SILVA database)
https://zenodo.org/records/1447330
Fungal SILVA (fungi specific beta version, fungi only)
https://zenodo.org/records/15044434 **the standard SILVA database no longer includes eukaryotes, so it must be the fungi specific database if using for fungal taxonomic assignment

Note

If reads do not appear to be correctly assigned (lots of NAs at high taxonomic levels), try the reverse complement orientation by adding assignTaxonomy(..., tryRC = TRUE).

Port into Phyloseq

15 Read into phyloseq
This step is going to import the tables generated by DADA2 into a single data object and add metadata to this object.

You may want to make the object names more specific to your data set

```
OTU = otu_table(seqtab.nochim, taxa_are_rows = FALSE)
TAX = tax_table(taxa.rdp)
OTUTAX = phyloseq(OTU, TAX)
MAP =
import_qiime_sample_data("/Users/cward/Documents/Projects/EPA_Ohio
Lakes16S/lakes_map2.txt") #replace with the path to your sample
data file

po = merge_phyloseq(OTUTAX, MAP)
po  # verify size and format of new phyloseq object
```

Note

If sequencing was performed in multiple runs and needed to be processed separately, you should do so after creating separate phyloseq objects for each of the sequencing runs. The merge_phyloseq(object1, object2) function should do this easily.

16 Track ASV sequences

When creating the phyloseq object at the end of the DADA2 workflow, it is useful to add a DNAStringSet object (to be accessed by the refseq function) to keep track of the ASV sequences. This code will retain the unique sequence identifier and actual sequence while also making printing the taxonomy table easier.

```
sequences <- Biostrings::DNAStringSet(taxa_names(po))  # ASV names
are actual sequence, very cumbersome!
names(sequences) <- taxa_names(po)
po <- merge_phyloseq(po, sequences)
po
taxa_names(po) <- paste0("ASV", seq(ntaxa(po)))  # rename ASVs to
more convenient names while retaining corresponding unique
sequence identifier
```

Community analysis - alpha diversity

17 Since several measures of alpha diversity are particularly susceptible to variation in sampling effort (i.e. sequence depth), we must standardize the sampling effort across all samples before calculating alpha diversity. The conventional approach used by

community ecologists is rarefaction. Some biostatisticians, including the developers of phyloseq and DADA2, do not embrace this practice.

18 Rarefy to equal sequence count per library

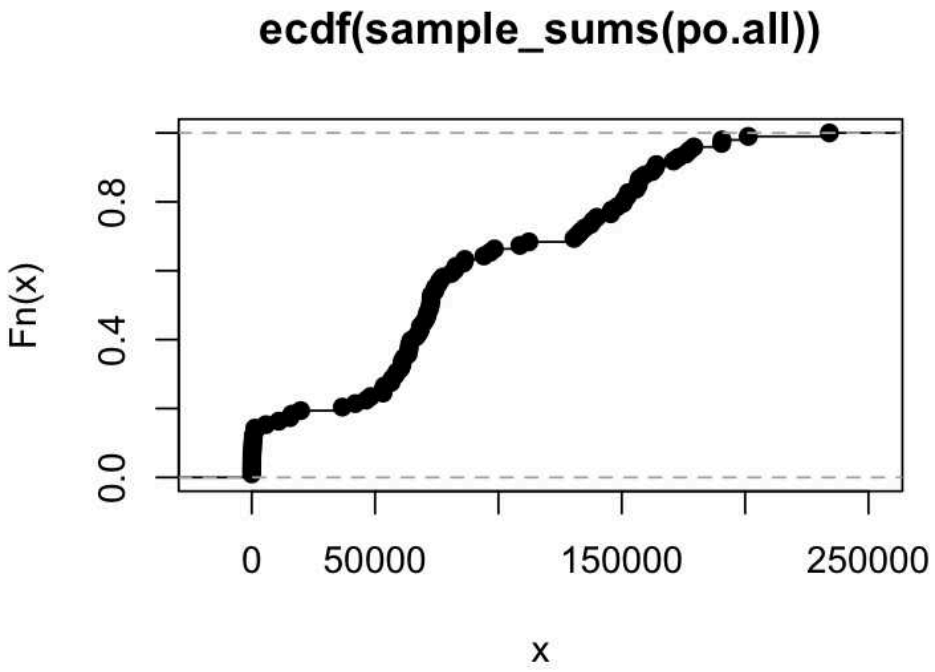
```
min(sample_sums(po)) #lowest sample sequence= 4002 -- change to value specific to your data
max(sample_sums(po)) #highest sample sequence=4002 -- change to value specific to your data
plot(ecdf(sample_sums(po)))  #plot distributions of sequencing efforts. Ticks up around 5500-6000 -- change to values specific to your data

po_r = rarefy_even_depth(po, sample.size = 4000, rngseed = 50)
#rarefy all samples to 4k, 0 samples removed and 262 ASVs no longer present -- depth to rarefy samples to will vary depending on the data, update this to correspond to your data

nsamples(po_r)  #100 samples remaining
sample_sums(po_r) #confirm rarefaction to 4k reads/library
```

Note

Rarefaction depth:
If you have high quality libraries, it is acceptable to rarefy to the depth of your lowest sample sequence.

To pick a depth to rarefy to, look at the plot generated by the code below. On this plot, you will likely want to rarefy to the value where sequencing efforts begin to uptick. In the plot below for example, you would want to pick somewhere around 50k as your rarefaction depth.

You will also want to record the value you use for rngseed = ... when rarefying for reproducibility. This should be included in the methods section of a publication.



Plot of sampling efforts for use in deciding rarefaction depth.

19 Compare alpha diversity

Now that all libraries have the same sequence depth, you may more accurately compare alpha diversity. Examine the plot_richness documentation for options for other arguments.

```
plot_richness(po_r, measures=c("Observed", "Shannon", "Simpson",
"InvSimpson"))
plot_richness(po_r, measures=c("Observed", "Shannon", "Simpson",
"InvSimpson")) + geom_boxplot()

# Compare measures between rarefied and unrarefied
plot_richness(po, "SiteName", measures=c("Observed","Shannon"),
color="Zone") + geom_boxplot() # change bolded variables to match
your sample data
plot_richness(po_r, "SiteName", measures=c("Observed","Shannon"),
color="Zone") + geom_boxplot()
# include observations about how rarefaction affects alpha
diversity here
```

Community analysis - taxonomic composition

20 Below are a few ways to look at the taxonomic composition of our microbial community at various taxonomic levels. There are many other variations that you can explore once you get a grasp of the general syntax.

As usual, change bolded values to reflect the names of your data objects and your sample data.

Note

Considerations for 18S data:
If you assigned the taxonomy of your ASVs using the PR2 database, the taxonomic ranks of your data will be different. They should fall into the order of Domain, Supergroup, Division, Subdivision, Class, Order, Family, Genus, Species, so be aware of this when attempting to plot the data at certain taxonomic levels.

Examining the community at the phylum level:

```
lakes_phylum <- po %>% tax_glom(taxrank = "Phylum") %>%
transform_sample_counts(function(x) {x/sum(x)} ) %>% psmelt() %>%
arrange(Kingdom)  # first viewing the community at the phylum
level
ggplot(lakes_phylum, aes(x = SiteName, y = Abundance, fill =
Phylum)) +
  geom_bar(stat = "identity", position = "fill") +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  guides(fill = guide_legend(keywidth = 1, keyheight = 1)) +
  ylab("Relative Abundance")
```

Examining the community at the phylum level with an abundance filter (1% relative abundance) and faceting the plot by zone (or another variable)

```
lakes_phylum2 <- po_lakes %>% tax_glom(taxrank = "Phylum") %>%
transform_sample_counts(function(x) {x/sum(x)} ) %>% psmelt() %>%
filter(Abundance > 0.01) %>% arrange(Kingdom)  # adding a minimum
abundance threshold of 1% in any sample (other thresholds
possible)
ggplot(lakes_phylum2, aes(x = SiteName, y = Abundance, fill =
Phylum)) +
  facet_grid(.~Zone, scale="free") +
  geom_bar(stat = "identity") +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  guides(fill = guide_legend(keywidth = 1, keyheight = 1)) +
  ylab("Relative Abundance")  # added auto-scaling of both axes
and faceting by phylum and sample type
```

Examine the community at the order level

```
lakes_order <- po %>% tax_glom(taxrank = "Order") %>%
transform_sample_counts(function(x) {x/sum(x)} ) %>% psmelt() %>%
arrange(Phylum)  # now viewing the community at the order level
ggplot(lakes_order, aes(x = SiteName, y = Abundance, fill =
Order)) +
  geom_bar(stat = "identity", position = "fill") +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  guides(fill = guide_legend(keywidth = 1, keyheight = 1)) +
  ylab("Relative Abundance")
```

Examine community at the genus level and facet by phylum

```
lakes_genus <- po %>% tax_glom(taxrank = "Genus") %>%
transform_sample_counts(function(x) {x/sum(x)} ) %>% psmelt() %>%
filter(Abundance > 0.01) %>% arrange(Phylum)  # genus level,
adding a minimum abundance filter since there are so many genera
to display
ggplot(lakes_genus, aes(x = SiteName, y = Abundance, fill =
Genus)) +
  facet_grid(Phylum~., scale="free_x") +
  geom_bar(stat = "identity") +
  theme(axis.title.x = element_blank()) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  guides(fill = guide_legend(keywidth = 1, keyheight = 1)) +
  ylab("Relative Abundance")  # added clustering genera by their
phyla and auto-scaling of x-axis
```

Note

By default, the tax_glom function removes NA values, meaning the relative abundances of unassigned taxa are not included in these plots. If you wish to include all unassigned taxa grouped under a category NA, you can alter the argument to be tax_glom(..., NArm = FALSE).

Additional ways of including unassigned taxa exist, including altering the taxa table to list all unassigned taxa as "Unclassified". More intensive manipulation can change all NAs to a format similar to Last-assigned-taxa_X, but this is a more complicated process.

21 Subset dataset by taxa

You can also subset your data set by taxa using the subset_taxa function if you want to explore a specific group. To explore a specific sample type you can use the subset_samples function.

An example:

```
lakes_cyano = subset_taxa(po, Phylum=="Cyanobacteria/Chloroplast")
cyano_family <- lakes_cyano %>% tax_glom(taxrank = "Family") %>%
transform_sample_counts(function(x) {x/sum(x)} ) %>% psmelt() %>%
filter(Abundance > 0.01) %>% arrange(Phylum)
```

Community analysis - Beta diversity

22 Beta diversity refers to the differences between communities. There are numerous ways to calculate and visualize beta diversity -- this workflow includes just a few examples.

23 Ordination: NMDS and PCoA

This **website** set up by the **Laboratory for Innovative Biodiversity Research And Analysis (LIBRA)** at Oklahoma State is a particularly useful source of information on ordination approaches for ecological communities. It's important to remember that these methods are relatively easy to perform and get results, but there's a lot under the hood and a deep dive into the details is essential for those using them regularly.

Both of these examples use bray-curtis dissimilarity, which is a common method for evaluating the similarity of samples/communities/clusters. However, a number of other methods do exist that may be more fitting for your data, so it is important to consider what you are attempting to communicate with these plots.

NMDS:

```
lakes_nmds <- ordinate(physeq = po, method = "NMDS", distance =
"bray")
stressplot(lakes_nmds)
plot_ordination(po, ordination = lakes_nmds, color = "Zone", shape
= "Zone") + geom_point(size = 3)
```

PCoA:

```
lakes_pcoa <- ordinate(physeq = po, method = "PCoA", distance =
"bray")
plot_ordination(po, ordination = lakes_pcoa) + geom_point(alpha =
0.7, size = 4)
```

Note

The statistical significance of clusters/differences visualized or notated by ordination plots can be tested by running a PERMANOVA using the adonis2 function from the vegan package. A brief example is included below, but your data will likely require additional considerations.

24 PERMANOVA and dispersion analysis

PERMANOVA is a statistical test that uses the distance matrix created in your ordination to analyze variance. This allows you to test your ordination for statistical significance.

PERMANOVA testing:

```
permanova_result <- adonis2(dist_matrix ~
metadata$treatment/test_variable, data = metadata, permutations =
999, method = "bray")
print(permanova_result)
```

Following running the PERMANOVA, it is helpful to test for the significance of dispersion, particularly if the results of the PERMANOVA are significant. This checks to make sure that the significance of the PERMANOVA is due to differences in the group compositions and not the spread of the data.

```
dispersion <- betadisper(dist_matrix, metadata$test-group-
variable)
anova(dispersion)
permutest(dispersion)
```

25 Hierarchical clustering

This can be used to visualize clusters of sites/another variable based on how similar they are to each other. This code uses Bray-Curtis dissimilarity to evaluate this, but other methods could also be used.

```
set.seed(1)
lakes_bray <- phyloseq::distance(po, method = "bray")
comm.bray.clust <- hclust(lakes_bray, method = "average") #
cluster communities using average-linkage algorithm
plot(comm.bray.clust, ylab = "Bray-Curtis dissimilarity") # plot
hierarchical cluster diagram
```

Additional Analyses

26 Export selected ASV sequences so they can be explored in other tools (e.g. BLAST, multiple sequence alignment)

```
lakes_chloroplast = subset_taxa(po, Order=="Chloroplast")
chloro_seqs <- taxa_names(lakes_chloroplast) # Store sequences
write(chloro_seqs,
"/Users/cward/Documents/Projects/EPA_OhioLakes16S/chloro_asvseqs.t
xt")
```

27 DESeq2 and volcano plots

```
library(EnhancedVolcano)
EnhancedVolcano(worm_deseq_phy,
    lab = rownames(worm_deseq_phy),
    x = 'log2FoldChange',
    y = 'padj',
    xlim=c(-5,5),
    ylim=c(0,6),
    pCutoff = 0.05,
    FCcutoff = 1)
```

28 ASV Presence/Absence Venn Diagrams

If you have particular treatments or different sample categories, you can visualize the ASVs that ubiquitous as well as those that are only found in certain treatments using a venn diagram. This method provides additional information if you are attempting to evaluate the differences in community structure, especially when using presence absence data.

Here is an example, however, this will likely need to be altered heavily for your data.

```
physeq_pa <- transform_sample_counts(fungi.winam.gulf.dominant,
function(x) ifelse(x>0, 1, 0))
otu_df <- as.data.frame(otu_table(physeq_pa))
sample_df <- as.data.frame(sample_data(physeq_pa))

asv_counts <- merge(otu_df, sample_df, by = 0)

otu_df <- as.data.frame(otu_table(fungi_physeq_pa))
otu_df <- t(otu_df)

# Get sample metadata
sample_df <- as.data.frame(sample_data(fungi_physeq_pa))

# Identify unique treatments
treatment_groups <- unique(sample_df$CyanoDominance)

# Create a list to store ASV presence for each treatment
asv_lists <- list()

for (treatment in treatment_groups) {
  samples_in_treatment <-
rownames(sample_df[sample_df$CyanoDominance == treatment, ])
  present_asvs <- rownames(otu_df)[rowSums(otu_df[,
samples_in_treatment, drop = FALSE]) > 0]
  asv_lists[[treatment]] <- present_asvs
}

#presence/absence venn diagram
ggvenn(asv_lists)
```

Protocol references

https://benjjneb.github.io/dada2/tutorial.html