# 16S rRNA gene amplicon community analysis protocol

This protocol is modified based on the [Mothur MiSeq Standard Operating Procedure](https://www.mothur.org/wiki/MiSeq_SOP).

Please note that this is a **DRAFT** protocol and not yet ready for production use. Discussion questions are listed as **bold** bullet points.

1. Collect all gzipped FASTQ files into a single directory (your "project directory") - one pair of files (forward/reverse reads) for each sample.

* **Do we need to worry about the standard Illumina method for dividing a sequencing run based on multiplex barcoding, or are we happy with what comes off the machine? Do we need to deal with sequences "leaking" in between libraries (samples)?**

1. Create a text file in the same folder as your .fastq.gz files to indicate to mothur which file matches which sample. The file should be called basename.files where basename is a short descriptive name for the dataset you're working with. The file should have one line per sample, with the following format (3-sample example):

* Sample1 Sample1\_S20\_L001\_R1\_001.fastq.gz Sample1\_S20\_L001\_R2\_001.fastq.gz  
  Sample2 Sample2\_S19\_L001\_R1\_001.fastq.gz Sample2\_S19\_L001\_R2\_001.fastq.gz  
  Sample3 Sample3\_S21\_L001\_R1\_001.fastq.gz Sample3\_S21\_L001\_R2\_001.fastq.gz

1. Open mothur and merge your forward and reverse reads and set the input/output directory (your "project directory") with the following command. The number of processors you choose will depend on your computer - if you don't know how many processors your computer has, then just set it to 1. Note that you will need to specify the drive and use backslashes for your directory paths on Windows (i.e. C:\_dir), the forward slashes shown below are for Mac or Linux.

* **What about sequences with poor quality ends - are they going to merge OK? Does removing sequences with ambiguities take care of this problem, or are there going to be some sequencing errors leftover? Or is 0 ambiguities too strict - are we throwing away too much data?**
* make.contigs(file=basename.files, input=/path/to/project\_directory, output=/path/to/input/project\_directory, processors=8)

1. Remove reads that are too short or suspiciously long:

* screen.seqs(fasta=basename.trim.contigs.fasta, group=basename.contigs.groups, maxambig=0, minlength=400, maxlength=500)

1. Find the unique sequences only to save time:

* unique.seqs(fasta=basename.trim.contigs.good.fasta)

1. So that mothur can keep track of the unique sequences across different samples, we need to run this command:

* count.seqs(name=basename.trim.contigs.good.names,group=basename.contigs.good.groups)

1. Align the sequences to a version of the silva database trimmed according to the primers you used for your sequencing.

* **Does our Silva database need to be pre-trimmed according to our primers? Or is this step unnecessary? We would probably need to include a database prep protocol to accompany this document.**
* **There is some mention of the "standard-issue" silva mothur database lacking many archaeal groups due to the lower quality of their sequences in the Silva database. Should this worry us if we're looking into archaea?**
* align.seqs(fasta=basename.trim.contigs.good.unique.fasta, reference=/path/to/database/silva.nr\_v123.pcr.align)

1. Now run summary sequences on the aligned sequences to see how well they aligned.

* summary.seqs(fasta=basename.trim.contigs.good.unique.align, count=basename.trim.contigs.good.count\_table)

1. Take a look at the summary sequences - the output will look something like this:

* Start End NBases Ambigs Polymer NumSeqs  
  Minimum: 0 0 0 0 1 1  
  2.5%-tile: 6428 23440 64 0 4 3521  
  25%-tile: 6428 23440 403 0 4 35210  
  Median: 6428 23440 407 0 5 70420  
  75%-tile: 6428 23440 426 0 6 105630  
  97.5%-tile: 13859 23440 428 0 6 137319  
  Maximum: 26102 26103 447 0 12 140839  
  Mean: 7047.93 23349.2 395.411 0 4.89188  
  # of unique seqs: 99392  
  total # of seqs: 140839

1. The vast majority of sequences align from position 6428 to 23440, but a small minority of sequences do not. Use the following command to remove those sequences that are poorly aligned to the reference database. We also use this step to remove sequences with a string of identical bases (homopolymers) longer than 8 bases using the maxhomop command.

* screen.seqs(fasta=basename.trim.contigs.good.unique.align, count=basename.trim.contigs.good.count\_table, summary=basename.trim.contigs.good.unique.summary, start=6428, end=23440, maxhomop=8)

1. Now filter the sequences to remove overhangs outside of the aligned region:

* **Is this the only and best way to remove primers, bits of adapter etc.?**
* filter.seqs(fasta=basename.trim.contigs.good.unique.good.align, vertical=T, trump=.)

1. There is some redundancy across the sequences now following trimming, so we're going to run unique.seqs again.

* unique.seqs(fasta=basename.trim.contigs.good.unique.good.filter.fasta, count=basename.trim.contigs.good.good.count\_table)

1. Now to pre-cluster the sequences to remove sequencing error - this is thought to remove sequencing error (up to 1 bp variation per 100 bp). Since our amplicons are around 450bp long, we'll pre-cluster based on up to 4 differences.

* **Is this pre-clustering and unique.seqs command (step 12) really necessary? If our OTU-making is fast enough further down the track, does this contribute in any way rather than reducing the number of sequences and thus speeding things up? Could these steps be removed without any change to the overall outcome?**
* pre.cluster(fasta=basename.trim.contigs.good.unique.good.filter.unique.fasta, count=basename.trim.contigs.good.unique.good.filter.count\_table, diffs=4)

1. Now to detect and remove chimeric sequences using uchime.

* **Is uchime the best tool for this? Should we consider using vsearch or another tool?**\*
* chimera.uchime(fasta=basename.trim.contigs.good.unique.good.filter.unique.precluster.fasta, count=basename.trim.contigs.good.unique.good.filter.unique.precluster.count\_table, dereplicate=t)  
  remove.seqs(fasta=basename.trim.contigs.good.unique.good.filter.unique.precluster.fasta, accnos=basename.trim.contigs.good.unique.good.filter.unique.precluster.denovo.uchime.accnos)

1. Now the sequences are ready for classification.

* **The standard mothur protocol carries out classification before OTU clustering, then classifies the OTUs based on the classified pre-clustered sequences. Why? If it's the OTUs we're interested in, why not classify the OTUs after they're OTUs?**
* **This is the default mothur classification algorithm (wang). Is this what we want to use?**
* classify.seqs(fasta=basename.trim.contigs.good.unique.good.filter.unique.precluster.pick.fasta, count=basename.trim.contigs.good.unique.good.filter.unique.precluster.denovo.uchime.pick.count\_table, reference=/path/to/database/silva.nr\_v123.pcr.align, taxonomy=/path/to/database/silva.nr\_v123.pcr.align)

1. Now to cluster the sequences into OTUs using the vsearch agc (abundance-based greedy clustering) algorithm.

* **Is this the clustering algorithm we should be using? We need to consider both speed (performance) and accuracy. What about usearch?**
* cluster(fasta=basename.trim.contigs.good.unique.good.filter.unique.precluster.pick.fasta, count=basename.trim.contigs.good.unique.good.filter.unique.precluster.denovo.uchime.pick.count\_table, method=agc)

1. Determine the number of times each OTU occurs in each sample using this command.

* make.shared(list=basename.trim.contigs.good.unique.good.filter.unique.precluster.pick.agc.unique\_list.list, count=basename.trim.contigs.good.unique.good.filter.unique.precluster.denovo.uchime.pick.count\_table, label=0.03)

1. Classify OTUs (based on the classification of the preclustered sequences) like so:

* classify.otu(list=basename.trim.contigs.good.unique.good.filter.unique.precluster.pick.agc.unique\_list.list, count=basename.trim.contigs.good.unique.good.filter.unique.precluster.denovo.uchime.pick.count\_table, taxonomy=basename.trim.contigs.good.unique.good.filter.unique.precluster.pick.pcr.wang.taxonomy, label=0.03)

1. From this point you will switch from using mothur to using R. You can quit mothur using the quit() command, or leave it open if you think you might use it again. Open R and import your data using the phyloseq package with the following commands:

* setwd("/path/to/project\_directory")  
    
  library("phyloseq")  
  library("ggplot2")  
    
  moth\_shared <- "basename.trim.contigs.good.unique.good.filter.unique.precluster.pick.agc.unique\_list.shared"  
  moth\_tax <- "basename.trim.contigs.good.unique.good.filter.unique.precluster.pick.agc.unique\_list.0.03.cons.taxonomy"  
  basename.phyloseq = import\_mothur(mothur\_shared\_file=moth\_shared, mothur\_constaxonomy\_file=moth\_tax)

1. Plot alpha diversity estimates for all your samples:

* **Would rarefaction make these values more or less accurate?**
* plot\_richness(basename.phyloseq)

1. Plot OTU abundance coloured by phylum-level classification:

* plot\_bar(basename.phyloseq, fill="Rank2")

1. Plot a basic NMDS (PCA) of your samples.

* **This is without any transformation or normalisation of the data. Should we be normalizing the data, for example using DESeq2, before making this kind of plot?**
* basename.phyloseq.ord <- ordinate(basename.phyloseq, "NMDS", "bray")  
  plot\_ordination(basename.phyloseq, basename.phyloseq.ord, type="sample")

1. Phyloseq (and R, for that matter) contains many other powerful analysis and plotting tools for your data. See the [Phyloseq Homepage](https://joey711.github.io/phyloseq/index.html) for many good examples.