`

bio-101

Marker Genes (16S and ITS) Protocol for Plant Microbiome Analyses

Geoffrey Zahn*

Biology Department, Utah Valley University, Orem, UT, USA
*For correspondence: gzahn@uvu.edu

Abstract

Many research questions in plant science depend on surveys of the plant microbiome. When the questions depend on answering "who is there" rather than "what are they doing," the most cost-effective method for interrogating microbiomes is often via targeted meta-amplicon surveys. There are numerous platforms for processing and analyzing meta-amplicon data sets, but here we will look at a flexible, reproducible, and fully customizable pipeline in the R environment. This approach has the benefit of being able to process and analyze your data in the same setting, without moving back and forth between standalone platforms, and further benefits from being completely flexible in terms of analyses and visualizations you can produce, without being limited to pre-selected tools available in point-and-click analysis engines, such as QIIME, Galaxy, or MG-RAST.

Keywords: Exact sequence variants, Meta-amplicon, Environmental DNA, Fungal, Bacterial, Microbiome

Background

It is increasingly common to find oneself needing to take a snapshot of microbial community structure in a given environment. From tracking key microbial members of the human gut, to comparing pathogen effects on crop-associated mutualistic bacteria, or testing the effect of probiotic applications in preventing disease in endangered plants, this is a common and powerful approach for gathering data (Busby *et al*., 2016; Zahn and Amend, 2017; Darcy *et al*., 2020). Whether your question is centered on the microbes or the host, you may find yourself with cost-effective, confusing, and potentially useful meta-amplicon data.

Typically, these data are obtained by sequencing either 16S (for bacteria) or ITS (for fungi) amplicons, which have been tagged with sample-specific barcodes. Hundreds of samples can be simultaneously sequenced on a single Illumina flowcell in this manner, making this a useful method for addressing the question "who is there?", and require high replication for statistical inferences. A sequencing center will typically return demultiplexed data, where each sample is in its own file. It's at this point a researcher suddenly realizes they don't know what to do with all of these files.

Here, I describe a workflow that (with minor adaptations) can be used for bacteria or fungi. The supplementary materials contain the exact code scripts, with all the documentation and lots of comment lines, but here we describe exactly each step, as we move from raw sequence files to hypothesis testing. Many of the tools and algorithms used in this workflow are quite complicated, so I have made an effort to keep explanations simple and at a surface level. More detailed explanations of each tool are best left to the associated publications.

The workflow presented here utilizes an "amplicon sequence variant" (ASV) instead of an operational taxonomic unit (OTU) approach. OTUs rely on clustering similar sequences, typically with random seeds, and a pre-defined set of similarity thresholds. ASVs take error-corrected sequences as the *de facto* unit of observation. The advantages of ASVs include the potential to uncover diversity that could otherwise be masked by lumping sequences together, and enabling fully reproducible results (Tipton *et al*., 2022). Rather than each taxonomic unit being denoted by a pseudo-random cloud of sequences, an ASV is an exact variant that can be compared across multiple studies.

The standard approach for any meta-amplicon workflow can be broken into two main tasks: 1) turning raw data into a sequence abundance table, and 2) using that table (with associated metadata) to generate and test hypotheses about microbial communities. These tasks have very different pitfalls to be aware of. The first task is much more formulaic, with only a few parameters that typically need to be tuned for each new study. The second task is more open-ended, but carries all the usual caveats that come with data exploration and hypothesis testing. This protocol will focus on the first task of generating a sequence abundance table from raw meta-amplicon reads, and will then give an example workflow that explores alpha- and beta-diversity, and looks for differentially abundant taxa between sample groups. This example will serve as a useful demonstration of how to go about exploring the derived dataset that this workflow generates, but is specific to the example data presented here.

Software and Data sets

Software

1. cutadapt (Martin, M., 2011; version 2.10; https://cutadapt.readthedocs.io/en/stable/changes.html#v2-1-2019-03-15)
2. ITSxpress (Rivers *et al*., 2018; version 1.0; https://github.com/usda-ars-gbru/itsxpress)
3. R (R Core Team, 2017; version 3.6.3; https://www.R-project.org/)
4. tidyverse R package (Wickham *et al*., 2019; version 1.3.0; https://www.tidyverse.org/)
5. DADA2 R package (Callahan *et al*., 2016; version 1.14.1; https://benjjneb.github.io/dada2/)
6. decontam R package (Davis *et al*., 2018; version 1.6.0; https://benjjneb.github.io/decontam/)
7. phyloseq R package (McMurdie and Holmes, 2013; version 1.30.0; https://joey711.github.io/phyloseq/)
8. Biostrings R package (Pagès *et al*., 2021; version 2.54.0; http://www.bioconductor.org/packages/release/bioc/html/Biostrings.html)
9. phangorn R package (Schliep, 2011; version 2.2.5; https://CRAN.R-project.org/package=phangorn)

10. msa R package (Bodenhofer *et al.*, 2015; version 1.18.0; https://bioconductor.org/packages/release/bioc/html/msa.html)
11. ShortRead R package (Morgan *et al.*, 2009; version 1.44.3; https://bioconductor.org/packages/release/bioc/html/ShortRead.html)
12. corncob R package (Martin, B. D. *et al.*, 2020; version 0.2.0; https://CRAN.R-project.org/package=corncob)
13. vegan R package (Okansen *et al.*, 2016; version 2.6.0; https://CRAN.R-project.org/package=vegan)
14. patchwork R package (Pedersen, 2020; version 1.0.1; https://CRAN.R-project.org/package=patchwork)

Input data

1. Input data are the demultimlexed fastq reads from sequenced bacterial 16S amplicons. There should be two files for each experimental sample that was sequenced; a file for forward reads, and a file for reverse reads. This workflow is written for the commonly used Illumina sequencing platform, but IonTorrent sequences can be used with some slight modification (noted in the workflow).
2. Example data consists of custom subsets of host-associated bacterial 16S amplicons taken from 46 seagrass samples (including PCR negatives). Files have been truncated and modified from their original form to allow for faster computational times, while still showing significant differences in community composition between sampling groups. They no longer represent any real study system, but are meant to be used as a teaching tool. Raw and intermediate data are provided in the GitHub repository associated with this publication.

Procedure

Case study

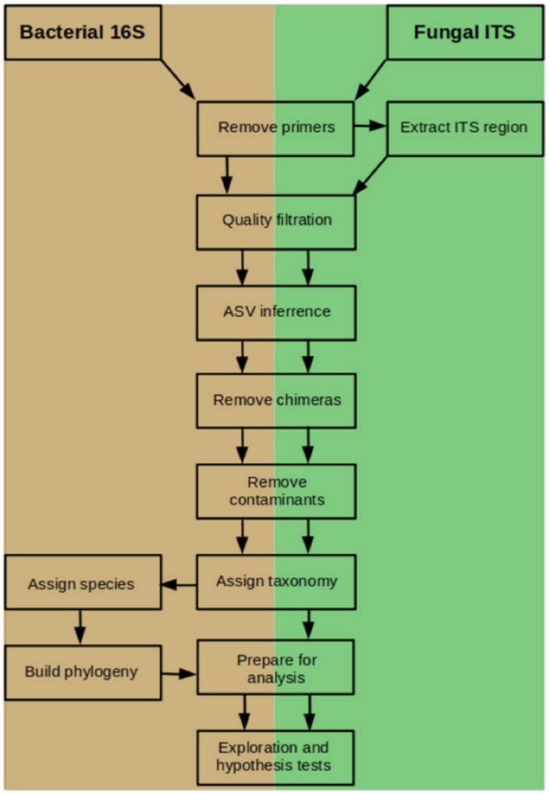

Figure 1. Workflow overview.
Steps unique to either bacterial or fungal amplicons are indicated.

This entire pipeline takes place in the R software environment due to its advantages in flexibility and customization over popular point-and-click pipelines, such as QIIME (Bolyen *et al.*, 2019), Galaxy (Afgan *et al.*, 2018), or MG-RAST (Meyer *et al*., 2008). Some other environments are used as well, but they are accessed via R, for convenience. The analysis, including all data, code, and output, exists as an R-Project within the RStudio IDE. This ensures that all file paths are relative, and the project is portable and self-contained (Bryan, 2017). Keeping a project's data and code in a self-contained directory means that it can be run on any computer with the appropriate software, without modification of the file paths. The pipeline will vary slightly depending on whether you are analyzing bacterial or fungal amplicons. We will work through the pipeline using bacterial 16S sequences, and make note as to where the pipeline differs for fungi. All R scripts noted below are present in the GitHub repository, and are also available as supplementary files. Sections of code that can be adapted for your own data are noted in the code comments. Figure 1 shows an overview of the following steps for bacterial and fungal amplicons. Estimated run time for this example data set on a typical laptop computer is approximately 20–30 min.

A. Remove primers (bacteria and fungi)

The first step is to remove any PCR primers that may still be present on your reads. Often, sequencing centers remove these by default, but it's a good idea to check. The R script "00_Remove_Primers.R" walks you through this process. In short, you provide the PCR primers that were used for generating the amplicons, and then use the *cutadapt* tool to search for their presence in all possible orientations. They are then removed, and we can proceed with quality filtration.

This process first removes all reads with ambiguous (N) bases, and then removes primer sequences and adaptors from remaining reads. The newly modified reads are stored in a separate directory (in this case ./data/filtN/). When working with bacterial 16S amplicons, this step can generally be accomplished by simply trimming the first N bases from both forward and reverse reads. However, using *cutadapt* is a more generalized approach that also accounts for mis-oriented reads, does not require you to know ahead of time if your primers are present or not, and is thus reccomended. Input files are the raw amplicon read fastq files. Output files have adaptors trimmed off.

B. Extract ITS region (fungi only)

This step only applies to fungal amplicons, but if you're working with fungi it's important that we extract the ITS region of interest at this point. The problem is that, while ITS1 or ITS2 are good at predicting fungal taxonomy, the primers used to amplify these regions sit inside highly conserved flanking regions (Gardes and Bruns, 1993). Since ITS lengths can vary so greatly (Bengtsson-Palme *et al*., 2013), keeping these conserved regions in our sequences can greatly bias downstream taxonomic assignment algorithms. A solution exists though, in the itsxpress software (Rivers *et al*., 2018), which can identify and trim these conserved regions from the ends of our ITS reads. This is typically run via the command line, but we can do it from within R. The supplementary file "00_Trim_ITS_Region.R" gives a thorough walkthrough example of how to extract the ITS1 region. Input files are the (fungal) fastq files without adaptors from previous step. Output files are (fungal) reads with flanking 18S/5.8S regions removed.

1. Quality filtration (bacteria and fungi)

Once you are sure your primers aren't present in your reads (or flanking conserved regions if analyzing fungal ITS data), it's time to do some quality control work, to filter out ambiguous and low-quality reads, and to trim reads where quality begins to suffer. The R script "01_Process_Raw_16S_Reads.R" goes through this step by step, but it's worth our time to explain what's going on. The workhorse software package for this and subsequent steps is DADA2 (Callahan *et al*., 2016).

We first parse our files generated by *cutadapt* (and *itsxpress*, in the case of fungal amplicons). The only filtration done on these so far was to remove ambiguous reads and any remaining primer sequences. The file parsing needs to account for separate forward and reverse reads, and it does so by matching patterns in the file names. In our example, data forward reads are denoted by "...pass_1", and reverse reads are denoted by "...pass_2" in their names. This can vary between sequencing centers, and your reads may have different patterns, such as "R1" and "R2", or "F" and "R." Be sure to modify the example code to match your forward and reverse reads accordingly, where noted.

The *plotQualityProfile*() function from the DADA2 package takes a look inside these sequence files, and allows you to make an informed judgement about how to filter and trim your raw reads. For example, in Figure 2 we are looking at the separate quality profiles of the forward and reverse reads of two samples. The X-axis shows read length, and the Y-axis shows summary information of quality scores for every read in the file. The green line is the mean, the orange line is the median, and the dashed orange lines are the 25th and 75th quantiles. It's up to you to determine where good cut-off points might be for the forward and reverse reads. Keep in mind that, to merge the reads later on, they will need to have sufficient overlap (typically >50 bases).

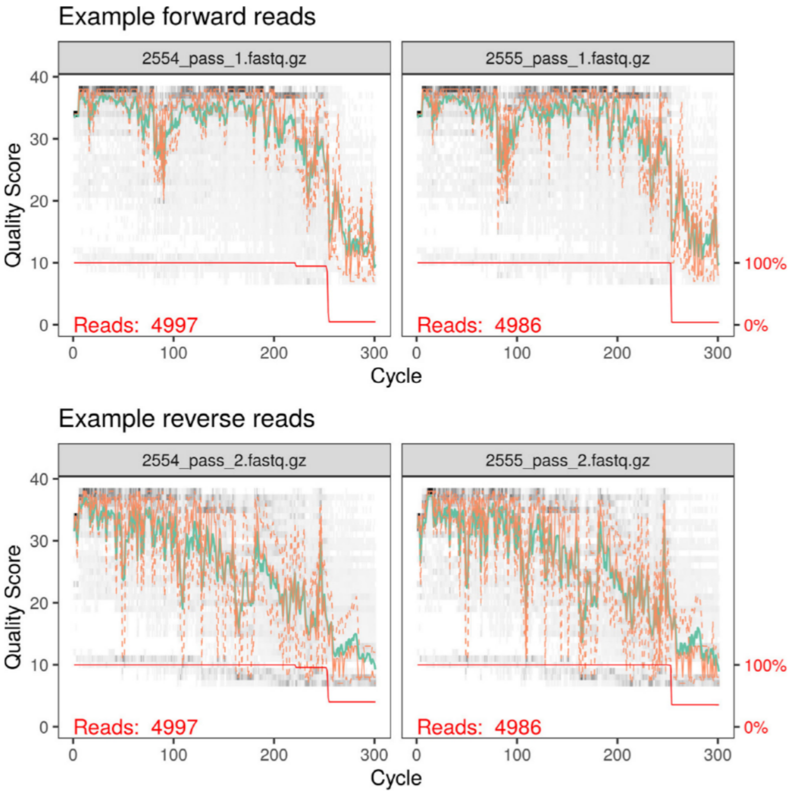

Figure 2. Quality profile plots for two samples.
Top row shows forward reads and bottom row shows reverse reads. Lines show summary stats along the length of reads: green is mean, orange is median, and dashed are 25th and 75th quantiles.

In our case (and this is typical), the forward reads are generally of better quality than our reverse reads (Figure 2). There is no "right value" for this next step, but a common practice is to truncate reads where the mean quality score dips below 25. In this example, that roughly lies at base position 250 for our forward reads, and at base position 200 for our reverse reads. These values vary between data sets, so make your best judgement. When in doubt, it's best to be conservative.

If you are working with fungal ITS reads, it is typical that reverse reads aren't of sufficient quality to merge with forward reads. Especially with variable ITS lengths, it is somewhat common practice to forego using reverse reads, and to process just the forward reads output by *itsxpress* (Darcy *et al*., 2020; Tipton *et al*., 2019, 2022).

The trimming and filtration step will discard some reads, but how many depends on how stringently you set the parameters. If you notice that the majority of your reads are being discarded, you might try relaxing the *maxEE* parameter slightly. Our reads had rather low quality scores, so it's not surprising that roughly half of them were rejected by our filtration step (Table 1). Figure 3 shows a comparison of quality profiles for a single sample before and after our filtration step. You can clearly see that the reads have been truncated at 250 nucleotides, and that mean quality scores are higher along the length of the reads. Input files for bacteria are fastq files stripped of adaptors. Input files for fungi are fastq files stripped of adaptors and flaking 18S/5.8S regions. Output files are fastq files filtered and trimmed to meet set quality thresholds.

Table 1. Example output from filtration step (first 3 files shown)

	reads.in	reads.out
2554_pass_1.fastq.gz	5000	2143
2555_pass_1.fastq.gz	5000	2245
2557_pass_1.fastq.gz	5000	2056

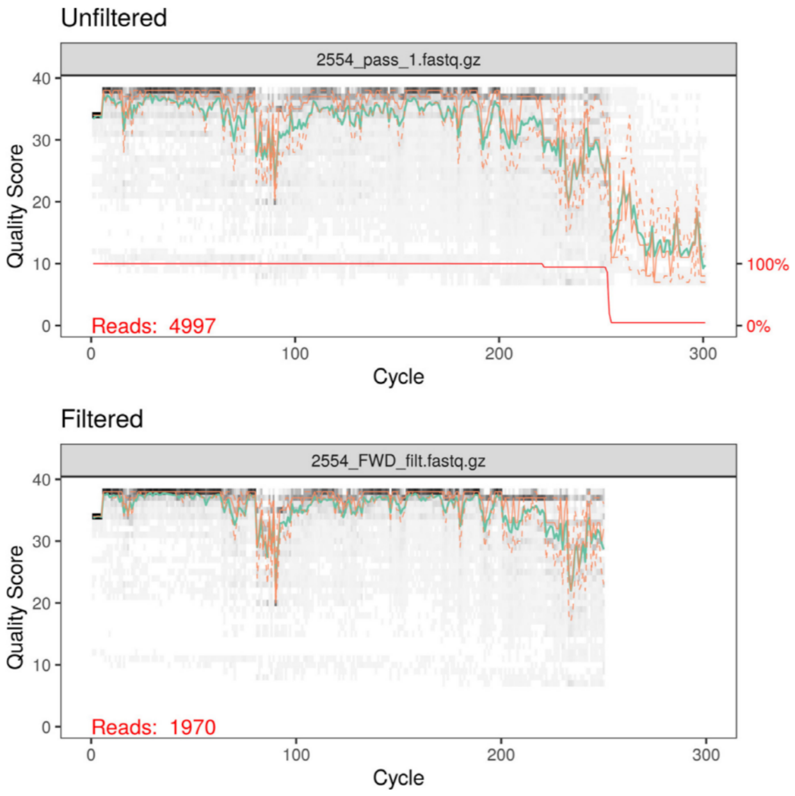

Figure 3. Comparison of quality profiles for a single sample pre- and post-filtration.

2. ASV inferrence (bacteria and fungi)

Now that we have a set of quality-controlled sequence files, we're ready to perform a "denoising" step, that can predict and correct likely sequencing errors. This is an important step when analyzing exact "amplicon sequence variants" (ASVs), as opposed to "operational taxonomic units" (OTUs). There is always a bit of "noise" in sequence data, representing sequencing errors. OTUs were, in part, conceived of as a way to deal with this constant threat of sequencing noise. By clustering highly similar sequences together, it accounts for small amounts of variation. However, much of that variation is real and informative (Callahan *et al*., 2017; Tipton *et al*., 2022). Methods now exist to discover and correct sequencing errors, allowing us to use ASVs. The advantages include finer taxonomic resolution, the potential to discover patterns that might be hidden by artificially clustering sequences together, and most importantly, the ability to easily combine data from multiple studies. Since each ASV is an actual sequence, rather than an undefined cloud of sequences (as with OTUs), they are directly comparable outside of a given study. This facilitates reproducibility and re-use, such as in meta-analyses.

The *DADA2* R package gives us a set of tools for inferring these ASVs. First, we build a mathematical profile of error rates along the length of our reads. In basic terms, we can think of this error profile as being a mathematical representation of our quality profile plots (Figure 3). The *learnErrors()* function is a machine-learning tool that builds an error model based on your data. Figure 4 shows this error model. What you're looking for is a decent fit between observed and estimated errors, and a decline in error as quality scores increase. Thus, it doesn't need to be a perfect fit. If your model looks reasonable (as ours does in Figure 4), you are ready to use that error model to correct sequence errors. A poorly fit error model (indicated by large discrepancies between observed points and the model fit line) may be a result of binned quality scores, or a lack of 'diversity' in quality scores. In this case, be sure you are using the raw sequence read files for creating the error profile, not reads that have been pre-processed in any way.

The *dada()* function performs denoising based on the estimated error model. In essence, it is performing statistical tests that determine whether a given sequence was observed too many times to have been caused by sequencing errors in currently inferred sequences (Rosen *et al*., 2012). The *dada()* function is run separately on forward and reverse reads, and the output is a set of sequence variants that were inferred from the "noisy" input sequences. From this point, we are ready to merge our forward and reverse reads and construct a sequence table. These tasks are accomplished with the *mergePairs()* and *makeSequenceTable()* functions, respectively. Input files are quality trimmed/filtered fastq files.

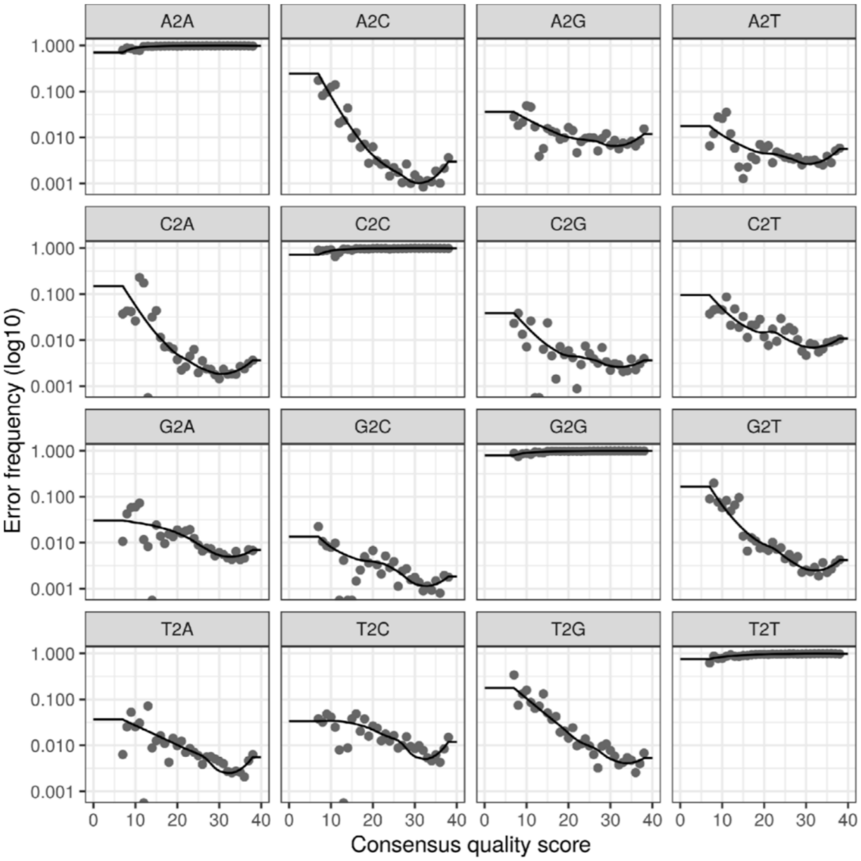

Figure 4. Estimated error rate plot.

Black dots show observed error frequencies at a given quality score (X-axis); Black lines show the estimated error model for each putative error type. Panels represent modeled substitutions; "A2C" is a model of the frequency that an A nucleotide in the real sequence shows up as a C nucleotide in the sequence read, for example.

3. The Remove chimeras (bacteria and fungi)

Chimeric sequences result from incomplete extension during PCR. Partial amplicons can serve as "false templates" for primers to bind to, resulting in reads that are composed of more than one true sequence. Chimera detection involves evaluating distinct matches between the left-side and right-side of reads and potential "parent" source reads. In practice, this detection and removal step can be conducted with the *removeBimeraDenovo()* function, as shown in the R script "01_Process_Raw_16S_Reads.R."

4. Remove contaminant sequences (bacteria and fungi)

While sadly not universally practiced, including and sequencing negative controls is a crucial step when performing meta-amplicon studies. Often, these control samples are blank DNA extractions that are then sequenced. This allows us to discover DNA sequences that were present in the lab environment that do not necessarily represent meaningful diversity in our real samples. The *decontam* package (Davis *et al.*, 2018) has been implemented to deal with these negative controls, using them to detect likely contaminant sequences. The way we will use this package is by giving the *isContaminant()* function our newly created sequence table, and a list of which samples are negative controls. Your sample metadata spreadsheet likely contains study-relevant information about each sequenced sample, but should also include a column denoting whether a sample is a negative control (Table 2 shows an example metadata sheet formatted correctly). Sequences with increased prevalence in control samples can then be identified and removed. Once potential contaminants are removed, we can then remove the negative control samples from further downstream steps since their job is done.

Table 2. Example metadata organization with column denoting whether a given sample is a negative control

SampleID	Island	Host_ID	Lat	Lon	Status	Control
CHPd62	Peros Banhos	H. ovalis	-5.42721	71.77779	healthy	FALSE
CHPd63	Peros Banhos	H. ovalis	-5.42721	71.77779	healthy	FALSE
CHPd65	Peros Banhos	H. ovalis	-5.42721	71.77779	impacted	FALSE
BLANK1	NA	NA	NA	NA	NA	TRUE
BLANK2	NA	NA	NA	NA	NA	TRUE

5. Assign taxonomy (bacteria and fungi)

Assigning taxonomy to DNA barcodes is a complicated subject. The success and accuracy of any assignments depend on the algorithms and databases that you use. In fact, with phylogeny-independent taxonomic assignments, the current limiting factors are the completeness and accuracy of the databases to which we match our sequences. With this in mind, it is critical that you carefully consider what methods to use. The RDP Naive Bayesian Classifier algorithm (Wang *et al.*, 2007) is one such algorithm that is widely used and has several advantages. First, it does not rely on sequence alignments, which makes it suitable for fungal ITS studies. Second, it is incredibly fast compared to algorithms such as BLAST. Third, it provides bootstrapped confidence values to all assignments, and allows matching to the "least common ancestor," reducing the incidence of false positive matches. Its performance for species-level taxonomic assignments of bacteria is lacking, and new information has shown that exact 100% matching is the only justifiable method for assigning species to bacterial ASVs (Edgar, 2018). A selection of properly formatted reference files for popular taxonomic databases are available at http://benjjneb.github.io/dada2/training.html. Here, we use two complementary methods to assign taxonomy to our reads: 1) the RDP Classifier using the RDP 16S Training Set database, implemented via the *assignTaxonomy()* function, and 2) exact 16S matching for species-level assignments, implemented via the *addSpecies()* function. The "Assign Taxonomy" section of "01_Process_Raw_16S_Reads.R" script covers these steps. These functions take our sequence table as input and return a data frame matching our ASVs to taxonomy at the deepest levels that could be assigned unambiguously.

If you are using fungal ITS reads, taxonomic assignment is essentially the same, but you will need to use a eukaryotic database such as the UNITE Eukaryotic Database (Abarenkov *et al.*, 2020), instead of a bacterial 16S database, and the *addSpecies()* function should be omitted.

The last step in this script is to construct a *phyloseq* object, in which to store your ASV table, along with your sample metadata, and taxonomic assignments. The *phyloseq* package (McMurdie and Holmes, 2013) provides a data structure that can combine a sequence abundance table (like our ASV table), taxonomic information, sample metadata, and a phylogenetic tree into one object, which makes working with these disparate tables much simpler. Output file is a compressed phyloseq object stored as an R object file.

6. Build phylogeny (bacteria only - optional)

If you're working with bacterial 16S, it can be a good idea to build a phylogeny of your ASVs. Some analyses, such as UniFrac community distance measures, use phylogeny-aware methods to improve comparisons between communities (Lozupone and Knight, 2005). However, fungal ITS reads are generally not alignable, and thus cannot be used to build phylogenies. This step is not required for most downstream analyses, so if you've got ITS reads, just skip to the next section.

It is beyond the scope of this paper to cover all the methods for building phylogenies, so we will focus on a simple and effective "good enough" approach within the R environment, using the *msa* and *phangorn* R packages (Schliep, 2011; Bodenhofer *et al*., 2015). The general workflow is to align your ASVs to each other, build a distance matrix, and then construct a phylogenetic tree from that matrix. The script "02_Build_and_add_Phylogeny.R" walks you through a typical workflow.

Our phyloseq object currently contains our ASV table, taxonomic assignments, and sample metadata. The sequences associated with each ASV can be easily extracted from our phyloseq object, and aligned using the MUSCLE algorithm (Edgar, 2004) using the *msa()* function. This alignment is then used to generate a maximum likelihood distance matrix with the *dist.ml()* function. We can then construct an initial tree using the Neighbor-Joining method, implemented with the *NJ()* function. The *phangorn* package has a range of tools for optimizing and updating trees based on different models, but for our purposes, a simple Neighbor-Joining tree will suffice as "good enough for now", since none of our anticipated downstream analyses will depend on having a phylogeny. If phylogenetic methods are essential for your own analyses, readers are encouraged to explore the *phangorn* package in further detail (see *phangorn* vignette at https://cran.r-project.org/web/packages/phangorn/vignettes/Trees.html). After we have a tree that we are comfortable with, we can add it to our *phyloseq* object, and move on to tidying our dataset in preparation for exploratory analyses.

7. Final cleanup and preparation for analyses (bacteria and fungi)

Our 16S primers can amplify non-bacterial DNA, such as chloroplasts or Archaea, in addition to bacteria. Unless these sequences are part of your research question, now is a good time to remove them. The script "03_Clean_and_Explore_Data.R" walks through a demonstration of how to use *phyloseq* to subset and manipulate our full data set.

First, we use the *subset_taxa()* function to remove any taxa that were not unambiguously assigned to the kingdom Bacteria. This removes any archaea or unknown sequences that were detected. Next, we use the same function to remove taxa that were assigned to known chloroplast identities. Finally, we can clean up our ASV table by removing any samples that are now "empty", by using the *subset_samples()* function. In other words, some samples may have *only* contained reads that were non-bacterial, so now that we have removed those reads, those samples will contain no information, and they should be dropped. This is usually only an issue when sequencing depth is very shallow. With this cleanup completed, we are now ready to start exploring our data set.

8. Exploration and hypothesis testing (bacteria and fungi)

The script "04_Explore_and_Test_Hypotheses.R" walks you through an example of how to use phyloseq to explore a meta-amplicon data set. Of course, this exploration and analysis process will vary greatly between studies, as it is completely contingent on what questions you set out to address. In our case, the example data were collected to assess seagrass-associated bacteria in pristine and impacted seagrass communities. The metadata may be different, but the techniques will not change, regardless of whether you are studying plant endophytes or human gut microbiomes. Additionally, there are no differences with regard to fungal or bacterial amplicons at this point. In fact, it is possible to combine fungal and bacterial phyloseq objects at this point, if you are studying them simultaneously for the same set of samples using the *merge_phyloseq()* function. Here, we will briefly outline some steps in a typical analysis. The code documentation in the associated script goes into more detail, but this is meant to be simply an example for how to use the functions mentioned here, and it is unlikely that any other study will follow these exact steps.

The script begins with a simple exploration of the various *phyloseq* accessor functions. These make it straightforward to access information without having to understand the structure of the phyloseq object. For example, we can access any of the individual data components independently: *sample_data()* returns our metadata, *otu_table()* returns our sequence (ASVs, in our case) abundance table, *tax_table()* returns our taxonomic assignments, and *phy_tree()* returns our phylogenetic tree.

Initially, many analyses start with a quick exploration of alpha diversity. The *estimate_richness()* and *plot_richness()* functions make this a simple task (Figure 5). Since we cannot guarantee equal sequencing depth between all samples, for most analyses it is important to realize that amplicon sequence data is compositional in nature (Gloor *et al*., 2017). Traditionally, rarefying data to a common sequence depth was the preferred method to adjust for discrepancies in sequencing effort, but this has been shown to needlessly remove potentially useful data (McMurdie and Holmes, 2014). We will simply convert our read counts into relative abundance values using the *transform_sample_counts()* function.

Next, we can take a look at beta-diversity, using ordination (*ordinate()* from the *phyloseq* package*)* and permANOVA (*adonis()* function from the *vegan* package). It is reasonable to try several ordination methods as we have done (Figure 6), depending on your data set. Figure 6 seems to show a strong separation of bacterial communities based on "Colony_Color." We can test this statistically using *adonis()*, where we model our ASV table as the response, and Colony_Color and Island as predictors (Table 3).

While permANOVA can tell us whether any patterns in community dissimilarity exist, it doesn't tell us which taxa are driving those differences. To answer that question, we must turn to special methods designed to detect differential abundant taxa. There are many commonly used tools for this, such as DESeq2 (Love *et al.*, 2014), and ALDEx2 (Fernandes *et al*., 2013), but here we will use the *corncob* package (Martin, B. D. *et al*., 2020) because it is capable of modeling differential variance in addition to abundance in a user-friendly way, and is well integrated with *phyloseq*. Note that all of these methods depend on non-transformed data (raw counts), unlike our beta-diversity measures, which needed normalized counts.

Table 3. PermANOVA model output. Island is a significant predictor of community dissimilarity, but Status is not.

term	df	SumOfSqs	R^2	statistic	P-value
meta$Status	1	0.1860	0.01445	0.6425	0.630
meta$Island	1	1.3744	0.10679	4.7467	0.003
meta$Status:meta$Island	1	0.0170	0.00132	0.0585	1.000
Residual	39	11.2925	0.87743	*NA*	*NA*
Total	42	12.8699	1.00000	*NA*	*NA*

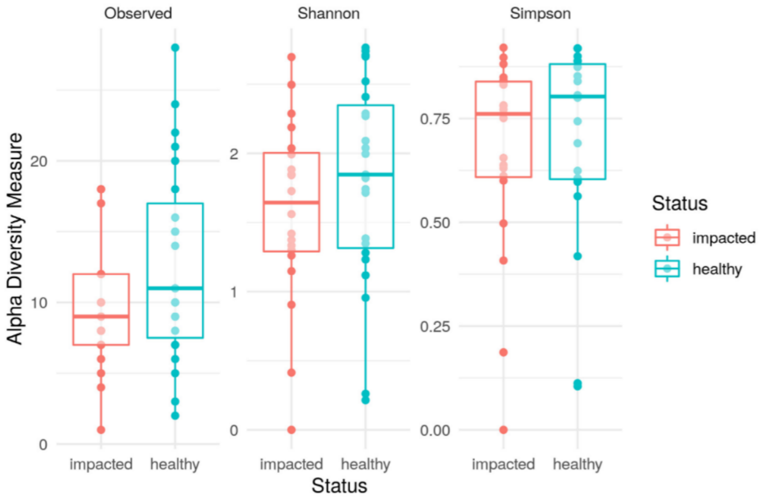

Figure 5. Alpha diversity boxplots for observed counts, Shannon, and Simpson diversity measures.
Figure code:

```
plot_richness(ps, x = "Colony_Color", measures = c("Observed","Shannon","Simpson"),
                      color = "Colony_Color", sortby = "Observed") +
  geom_boxplot(alpha = .5) + theme_minimal()
```

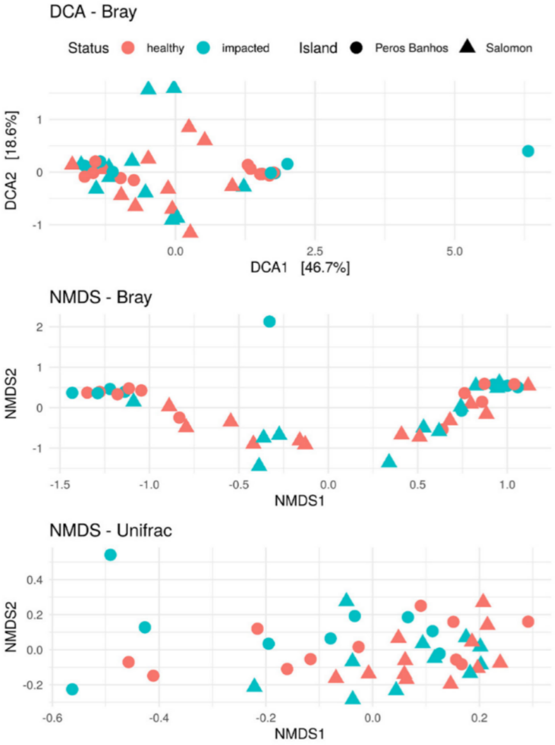

Figure 6. Comparison of ordination methods.
Top = Detrended correspondence analysis (DCA) using Bray-Curtis dissimilarity, Middle = Non-metric multidimensional scaling (NMDS) using Bray-Curtis dissimilarity, Bottom = NMDS using UniFrac dissimilarity. The same pattern is seen with each method.

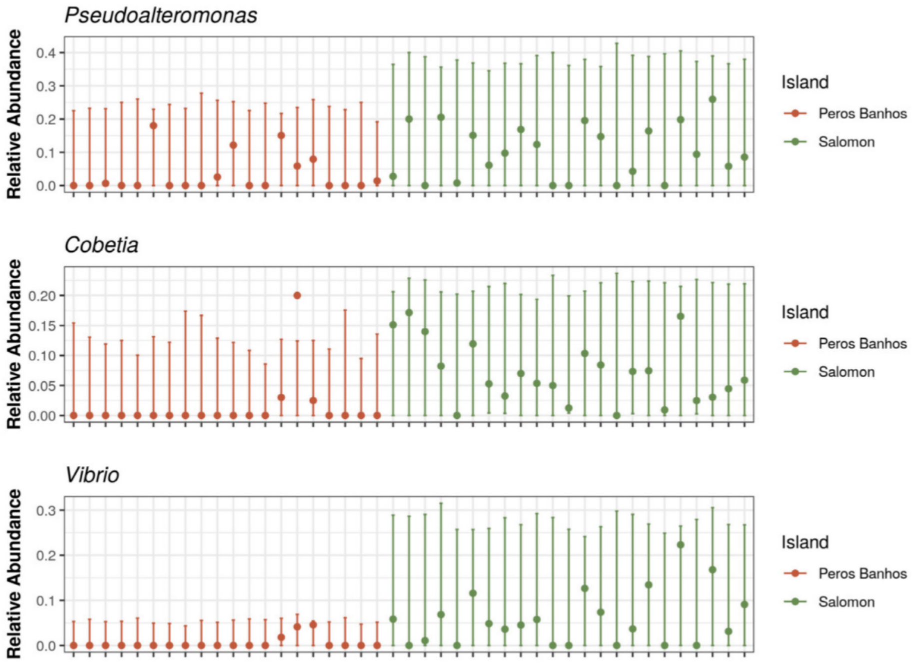

Figure 7 . Relative abundance plots for taxa identified as differentially abundant in a corncob model. Individual samples are on the x-axis, and are colored by "Island." Points show relative abundance, and bars show dispersion.

Result interpretation

Our raw forward and reverse reads showed rather sharp quality score dropoff at around position 250 and 200, respectively (Figure 2). Our quality control steps were effective at removing low quality regions and filtering out ambiguous reads (Figure 3). In fact, an average of 50% of all reads from each sample were removed (Table 1). This is rather a high proportion, but not unheard of. Your own data may fare better or worse than this, and it is up to you to determine meaningful quality thresholds that balance fidelity with data retention. In general, it is advised to lean towards a conservative approach, retaining only those reads in which you are confident.

In this example, we conducted only a few of many possible analyses with this mock data set, but it seems clear that a couple of patterns stand out. First, the environmental status of sites for seagrasses (impacted vs healthy) does not appear to have a significant effect on seagrass-associated bacterial communities, either in alpha- or beta-diversity measures (Figure 5; Table 3). However, location (Island) influences community structure (Table 3). Further, differential tests revealed three bacterial taxa (*Pseudoalteromonas, Cobetia,* and *Vibrio*) that were differentially abundant in samples between the two islands (Figure 7). The visualizations and tests you employ depend on the underlying questions in your study and the metadata available. These example analyses should serve as a starting point for working with microbiome data sets in R.

Acknowledgments

This protocol and the raw sequence data were derived from previous work with plant-associated microbial communities. In particular, Wainwright *et al.* (2018, 2019).

Competing interests

The author declares no financial or non-financial competing interests.

References

Abarenkov, K., Zirk, A., Piirmann, T., Pöhönen, R., Ivanov, F., Nilsson, R. H. and Kõljalg, U. (2020). UNITE general FASTA release for eukaryotes [Application/gzip]. UNITE Community. https://doi.org/10.15156/BIO/786370.

Afgan, E., Baker, D., Batut, B., van den Beek, M., Bouvier, D., Cech, M., Chilton, J., Clements, D., Coraor, N., Gruning, B. A., *et al*. (2018). The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Res* 46(W1): W537-W544.

Bengtsson-Palme, J., Ryberg, M., Hartmann, M., Branco, S., Wang, Z., Godhe, A., De Wit, P., Sánchez-García, M., Ebersberger, I., de Sousa, F., *et al*. (2013). Improved software detection and extraction of ITS1 and ITS2 from ribosomal ITS sequences of fungi and other eukaryotes for analysis of environmental sequencing data. *Methods Ecol Evol* 4(10): 914-919.

Bodenhofer, U., Bonatesta, E., Horejs-Kainrath, C. and Hochreiter, S. (2015). msa: an R package for multiple sequence alignment. *Bioinformatics* 31(24): 3997-3999.

Bolyen, E., Rideout, J. R., Dillon, M. R., Bokulich, N. A., Abnet, C. C., Al-Ghalith, G. A., Alexander, H., Alm, E. J., Arumugam, M., Asnicar, F., *et al*. (2019). Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. *Nat Biotechnol* 37(8): 852-857.

Bryan, J. (2017). Project-oriented workflow. https://www.tidyverse.org/blog/2017/12/workflow-vs-script/.

Busby, P. E., Peay, K. G. and Newcombe, G. (2016). Common foliar fungi of Populus trichocarpa modify Melampsora rust disease severity. *New Phytol* 209(4): 1681-1692.

Callahan, B. J., McMurdie, P. J. and Holmes, S. P. (2017). Exact sequence variants should replace operational taxonomic units in marker-gene data analysis. *ISME J* 11: 2639-2643.

Callahan, B. J., McMurdie, P. J., Rosen, M. J., Han, A. W., Johnson, A. J. and Holmes, S. P. (2016). DADA2: High-resolution sample inference from Illumina amplicon data. *Nat Methods* 13(7): 581-583.

Darcy, J. L., Swift, S. O. I., Cobian, G. M., Zahn, G. L., Perry, B. A. and Amend, A. S. (2020). Fungal communities living within leaves of native Hawaiian dicots are structured by landscape-scale variables as well as by host plants. *Mol Ecol 29*(16): 3102-3115.

Davis, N. M., Proctor, D. M., Holmes, S. P., Relman, D. A. and Callahan, B. J. (2018). Simple statistical identification and removal of contaminant sequences in marker-gene and metagenomics data. *Microbiome* 6(1): 226.

Edgar, R. C. (2004). MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* 5: 113.

Fernandes, A. D., Macklaim, J. M., Linn, T. G., Reid, G. and Gloor, G. B. (2013). ANOVA-like differential expression (ALDEx) analysis for mixed population RNA-Seq. *PLoS One* 8(7): e67019.

Gardes, M. and Bruns, T. D. (1993). ITS primers with enhanced specificity for basidiomycetes--application to the identification of mycorrhizae and rusts. *Mol Ecol* 2(2): 113-118.

Gloor, G. B., Macklaim, J. M., Pawlowsky-Glahn, V. and Egozcue, J. J. (2017). Microbiome Datasets Are Compositional: And This Is Not Optional. *Front Microbiol* 8: 2224.

Love, M. I., Huber, W. and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol* 15(12): 550.

Lozupone, C. and Knight, R. (2005). UniFrac: a new phylogenetic method for comparing microbial communities. *Appl Environ Microbiol* 71(12): 8228-8235.

Martin, B. D., Witten, D. and Willis, A. D. (2020). Modeling microbial abundances and dysbiosis with β-binomial regression. *Ann Appl Stat* 14(1): 94-115.

Martin, M. (2011). Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet J* 17(1): 10-12.

McMurdie, P. J. and Holmes, S. (2013). phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data. *PLoS One* 8(4): e61217.

McMurdie, P. J. and Holmes, S. (2014). Waste not, want not: why rarefying microbiome data is inadmissible. *PLoS Comput Biol* 10(4): e1003531.

Meyer, F., Paarmann, D., D'Souza, M., Olson, R., Glass, E. M., Kubal, M., Paczian, T., Rodriguez, A., Stevens, R., Wilke, A., Wilkening, J. and Edwards, R. A. (2008). The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics* 9: 386.

Morgan, M., Anders, S., Lawrence, M., Aboyoun, P., Pages, H. and Gentleman, R. (2009). ShortRead: a bioconductor package for input, quality assessment and exploration of high-throughput sequence data. *Bioinformatics* 25(19): 2607-2608.

Okansen, J., Blanchet, F. G., Friendly, M., Kindt, R., Legendre, P., McGlinn, D., Minchin, P. R., O'Hara, R. B., Simpson, G. L., Solymos, P., *et al*. (2016). vegan: Community Ecology Package (2.4-0) [Computer software].

Pagès, H., Aboyoun, P., Gentleman, R. and DebRoy, S. (2021). Biostrings: Efficient manipulation of biological strings (2.58.0) [Computer software]. Bioconductor version: Release (3.12).

Pedersen, T. L. (2020). patchwork: The Composer of Plots (1.1.1) [Computer software].

R Core Team. (2017). R: A Language and Environment for Statistical Computing (3.6.3) [Computer software]. R Foundation for Statistical Computing.

Rivers, A. R., Weber, K. C., Gardner, T. G., Liu, S. and Armstrong, S. D. (2018). ITSxpress: Software to rapidly trim internally transcribed spacer sequences with quality scores for marker gene analysis. *F1000Res* 7: 1418.

Rosen, M. J., Callahan, B. J., Fisher, D. S. and Holmes, S. P. (2012). Denoising PCR-amplified metagenome data. *BMC Bioinformatics* 13: 283.

Schliep, K. P. (2011). phangorn: phylogenetic analysis in R. *Bioinformatics* 27(4): 592-593.

Tipton, L., Zahn, G., Datlof, E., Kivlin, S. N., Sheridan, P., Amend, A. S. and Hynson, N. A. (2019). Fungal aerobiota are not affected by time nor environment over a 13-y time series at the Mauna Loa Observatory. *Proc Natl Acad Sci U S A* 116(51): 25728-25733.

Tipton, L., Zahn, G. L., Darcy, J. L., Amend, A. S. and Hynson, N. A. (2022). Hawaiian Fungal Amplicon Sequence Variants Reveal Otherwise Hidden Biogeography. *Microb Ecol* 83(1): 48-57.

Wainwright, B. J., Zahn, G. L., Arlyza, I. S. and Amend, A. S. (2018). Seagrass-associated fungal communities follow Wallace's line, but host genotype does not structure fungal community. *J Biogeogr* 45(4)*:* 762-770.

Wainwright, B. J., Afiq-Rosli, L., Zahn, G. L. and Huang, D. (2019). Characterisation of coral-associated bacterial communities in an urbanised marine environment shows strong divergence over small geographic scales. *Coral Reefs* 38: 1097-1106.

Wang, Q., Garrity, G. M., Tiedje, J. M. and Cole, J. R. (2007). Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Appl Environ Microbiol* 73(16): 5261-5267.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., *et al*. (2019). Welcome to the Tidyverse. *J Open Source Software* 4(43): 1686.

Zahn, G. and Amend, A. S. (2017). Foliar microbiome transplants confer disease resistance in a critically-endangered plant. *Peer J* 5: e4020.

Supplementary information

1. Data and code availability: All data and code have been deposited to GitHub: https://github.com/Bio-protocol/metaamplicon-recipe.