Using RDPTools Output with Phyloseq

John Quensen & Qiong Wang

August 2018

Introduction

The purpose of this document is to demonstrate how to generate RDPTools (Cole et al., 2014) output from sequencing data and import it into the R/Bioconductor package `phyloseq` (McMurdie and Holmes, 2012). Once this is done, the data can be analyzed not only using `phyloseq`'s wrapper functions, but by any method available in R. For examples see the section **Examples of Data Analysis** below.

You may install R from the **Comprehensive R Archive Network** (CRAN) repository at http://cran.us.r-project.org/. As an aid to working with R, we suggest that you also install RStudio, an IDE interface for R, available at http://www.rstudio.com/. By loading the Rmd file included with the tutorial files (see below) into RStudio, you can easily recreate the R portions of these tutorials on your own computer. And you can get an explanation of how to use any function in a code block by placing the cursor anywhere in the function name and pressing key F1.

RDPTools

The Ribosomal Database Project (RDP) at Michigan State University has long provided web-based tools for processing pyrosequencing data. These tools were originally developed for handling data for bacterial and archaeal 16S rRNA genes, but since then their capabilities have been expanded to include functional genes, 28S rRNA and ITS fungal sequences, and Illumina data. To handle the increased demands of analyzing larger data sets, command line versions of the tools have been made available as `RDPTools` on GitHub (https://github.com/rdpstaff/RDPTools). With these tools sequencing data can be analyzed by either a "supervised" approach in which sequences are binned using the RDP classifier(Wang et al., 2007), or an "unsupervised" approach in which sequences are clustered according to their similarities. Tutorials are available on the RDP website (http://rdp.cme.msu.edu/index.jsp), and detailed instructions for installation and use of the command line tools are available on GitHub.

At this time installation of `RDPTools` is practical only on Linux-like systems. Window users may use Ubuntu Linux in Oracle's VM Virtual Box, but with large data sets these programs are best run on a computer cluster.

Phyloseq

`Phyloseq` is an R/Bioconductor package that provides a means of organizing all data related to a sequencing project and includes a growing number of convenience wrappers for exploratory data analysis, some of which are demonstrated below. But perhaps `phyloseq`'s greater utility is that it makes it easy to subset and merge both samples and taxa. Methods for doing so are also demonstrated below.

If you need to install `phyloseq` in R, install it from Bioconductor with these commands:

```
source("http://bioconductor.org/biocLite.R")
biocLite("phyloseq")
```

RDPutils

This tutorial is concerned primarily with how the command-line programs in `RDPTools` can be used to generate files to fully populate a `phyloseq` object with an OTU table, sample data table, classification table, tree file, and reference sequences. RDP's web-based tools are currently more limited, but the vignette included in the R package `RDPutils` describes how to fully populate a `phyloseq` object using output from the web-based tools.

In any event, if you use the supervised method you will need a function in `RDPutils` to import the `classifier`'s result into `phyloseq`. See section **Processing Classifier Output** below. `RDPutils` depends on `phyloseq` and `reshape2`, so they should both be installed prior to installing `RDPutils`. `Phyloseq` also depends on `reshape2`, so `reshape2` may be installed automatically when you install `phyloseq`. If it is not, you may install `reshape2` by either of two methods.

If using RStudio, `reshape2` may be installed by clicking on "Packages," "Install," typing in "reshape2" (without the quotation marks), and clicking "Install." Whenever installing packages in this manner, be sure the "Install dependencies" box is checked.

Or from the R console, the `reshape2` package may be installed with the command:

```
install.packages("reshape2", dependencies=TRUE)
```

To install `RDPutils`, download the tar ball from http://rdp.cme.msu.edu/download/users/RDPutils_1.4.2.tar.gz and install it in R using the command:

```
install.packages("/path/RDPutils_1.4.2.tar.gz",  type="source", repos=NULL)
```

If `RDPutils_1.4.2.tar.gz` is not in your R working directory, you must include the path (indicated by `/path/` in the command above) with the file name. Use forward slashes in the path even if you are using Windows.

Other R Packages

The R portions of the tutorial below make use of several other R packages available from CRAN. These are:

- `ggplot2`
- `GUniFrac`
- `phangorn`
- `vegan`

If necessary, these may be installed following the instructions for installation of `reshape2` above.

Tutorial Files

Example data files used in this tutorial can be downloaded from http://rdp.cme.msu.edu/download/users/RDPTools_phyloseq_tutorial.zip. There are two folders inside `RDPTools_phyloseq_tutorial` after uncompressing the zip file, `supervised` and `unsupervised`. These contain example files such that all of the `RDPTools` commands in section **Generating Cluster Output (Unsupervised Method)** can be run from the `unsupervised` directory and all of the `RDPTools` commands in section **Generating Classifier Output (Supervised Method)** can be run from the `supervised` directory. Example files for all of the R commands in this tutorial are in the parent directory `RDPTools_phyloseq_tutorial`.

Some of the functions in phyloseq require a phylogenetic tree file. We recommend `FastTree` (http://www.microbesonline.org/fasttree) for creation of the tree file.

Initial Processing

First step sorting & trimming chimera filtering For unsupervised method, aligning

Generating Cluster Output (Unsupervised Method)

With RDPTools as of May 2014, it is possible to output results as a biom file with OTU table, classification table, and sample data. Assuming RDPTools and FastTree are installed, the commands below generate output to fill all phyloseq slots: `otu_table, tax_table, sample_data, phy_tree, and refseq`.

For this tutorial, the commands below are run from the directory `C:/RDPTools_phyloseq_tutorial/unsupervised`, which contains aligned sample fasta files inside the sub-directory `alignment`. The alignment files were created using RDP's Infernal bacterial 16S rRNA aligner. For your own work you can use any of the following RDP Aligners: Infernal bacterial and archaeal 16S Aligners, RDP fungal 28S Aligner, or Hmmer Aligner. Modify the path to `Clustering.jar` as appropriate to your installation of `RDPTools`. Each command must be entered as a single line. In this document, long commands are carried over to the next line(s) and indented for readability.

A helpful hint: The aligned sample fasta files should be named exactly as you wish your samples to be named in tables or figures to be generated from your data. The names should not contain prefixes or suffixes such as `aligned_` or `_trimmed` commonly introduced by previous processing steps.

Clustering

Complete-linkage clustering is performed by RDP's `mcClust` routine in the following three steps:

Dereplicate:

```
java -Xmx2g -jar ~/RDPTools/Clustering.jar derep -m '#=GC_RF' -o derep.fa
  all_seqs.ids all_seqs.samples alignment/*.fasta
```

Calculate distance matrix:

```
java -Xmx2g -jar ~/RDPTools/Clustering.jar dmatrix --id-mapping
  all_seqs.ids --in derep.fa --outfile derep_matrix.bin   -l 200
  --dist-cutoff 0.1
```

Cluster:

```
java -Xmx2g -jar ~/RDPTools/Clustering.jar cluster --dist-file
  derep_matrix.bin --id-mapping all_seqs.ids --sample-mapping
  all_seqs.samples --method complete --outfile all_seq_complete.clust
```

OTU File

If desired, you can also make a flat OTU table for distance = 0.03 that can be imported into R. This is a tab-delimited text file with samples as rows and OTUs as columns.

```
java -Xmx2g -jar ~/RDPTools/Clustering.jar cluster_to_Rformat
  all_seq_complete.clust . 0.03 0.03
```

Representative Sequences

To get representative sequences for each OTU, it is first necessary to merge all of the aligned sample fasta files into one aligned fasta file. To import the representative sequences into `phyloseq`, they must be renamed to match the cluster/OTU names.

Merge the aligned fasta files:

```
java -jar ~/RDPTools/AlignmentTools/dist/AlignmentTools.jar
  alignment-merger alignment merged_aligned.fasta
```

Get and rename sequences:

```
java -Xmx2g -jar ~/RDPTools/Clustering.jar rep-seqs -c --id-mapping
  all_seqs.ids --one-rep-per-otu   all_seq_complete.clust 0.03
  merged_aligned.fasta
```

Create Biom file

Make a biom file containing only OTUs from the cluster file for distance 0.03:

```
java -Xmx2g -jar ~/RDPTools/Clustering.jar cluster-to-biom
  all_seq_complete.clust 0.03 > all_seq_complete.clust.biom
```

Add Classification and Sample Data

The following command adds both classification and sample data to the biom file just created. The sample data to be added (`sam.data.txt` in this example) is in the form of a tab-delimited text file with sample names in the first column and attribute names in the first row. It is added with the -d switch in the command line. The sample names must match the names of the aligned sample fasta files exactly. Alternatively, the sample data file could be added later in R.

```
java -Xmx2g -jar ~/RDPTools/classifier.jar classify -c 0.5 -f biom -m
  all_seq_complete.clust.biom -d sam.data.txt -o  all_seq_complete.clust_classified.biom
  all_seq_complete.clust_rep_seqs.fasta
```

Reference Sequences

Recent versions of `phyloseq` objects may hold the representative sequences themselves, but the alignment is destroyed in the importation step. This is done by removing all .'s in the sequences, but not the -'s (which pad the sequences to full length). Thus it makes sense to also remove the -'s here. It is also necessary to remove the sequence descriptions before importing the representative sequences into `phyloseq`; this so that the names match the OTU names exactly. Lower case nucleotides will be converted to upper case when imported, so that does not need to be handled here.

Unalign sequences and remove descriptions:

```
java -Xmx2g -jar ~/RDPTools/Clustering.jar to-unaligned-fasta
  all_seq_complete.clust_rep_seqs.fasta | cut -f1 -d ' ' >
  unaligned_short_names.fasta
```

Tree File

If you have `FastTree` installed, you may tree the representative sequences.

Extract comparable sequences

If (as is the case in this tutorial), the alignment was produced by a model-based aligner such as Infernal or Hmmer, first extract the comparable (model) positions for tree building:

```
java -Xmx2g -jar ~/RDPTools/Clustering.jar derep -f -o
  all_seq_complete.clust_rep_seqs_modelonly.fasta rep_seqs.ids
  rep_seqs.sample all_seq_complete.clust_rep_seqs.fasta
```

Build the tree:

```
fasttree -nt -gtr < all_seq_complete.clust_rep_seqs_modelonly.fasta > my_expt_tree.nwk
```

Processing Cluster Output

Phyloseq includes a function for importing the biom, tree, and representative sequence files directly. In R, set the working directory (`RDPTools_phyloseq_tutorial` for this tutorial), load `phyloseq`, and create the phyloseq object `clst.expt`:

```
# setwd("C:/RDPTools_phyloseq_tutorial") # modify path as appropriate
suppressMessages(suppressWarnings(library(phyloseq)))
packageVersion("phyloseq")
```

```
## [1] '1.24.2'
```

```
clst.expt <- import_biom("all_seq_complete.clust_classified.biom",
  treefilename="my_expt_tree.nwk",
  refseqFunction=readDNAStringSet,
  parseFunction=parse_taxonomy_qiime)

clst.expt
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 3280 taxa and 8 samples ]
## sample_data() Sample Data:       [ 8 samples by 4 sample variables ]
## tax_table()   Taxonomy Table:    [ 3280 taxa by 6 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 3280 tips and 3278 internal nodes ]
```

If one is not able to use the command line `RDPTools`, the `RDPutils` package for R includes a vignette for creating a `phyloseq` object from a cluster file, a set of corresponding representative sequences, and the classification of the representative sequences.

Generating Classifier Output (Supervised Method)

The RDP classifier is provided with training sets for the identification of bacterial and archaeal 16S rRNA and fungal 28S rRNA and ITS gene sequences, and can classify multiple samples at one time. In this example, we use it to classify fungal 28S rRNA gene sequences and import the results into a `phyloseq` object. Clustering is not appropriate in this case because the primers are too far apart to read through the entire sequence.

Instead, identical bar codes were put on both forward and reverse primers so that for each sample sequences were obtained from each direction. Thus they could not be aligned, a necessary prerequisite to clustering, but they could still be classified.

To follow the example below, use the sample data downloaded from http://rdp.cme.msu.edu/download/users/RDPTools_phyloseq_tutorial.zip and run the command from the directory named `supervised`. Sample fasta files are in the sub-directory `sequences`.

```
cd c:/RDPTools_phyloseq_tutorial/supervised
java -Xmx2g -jar ~/RDPTools/classifier.jar classify -g fungallsu
  -c 0.5 -f filterbyconf -o test_classified.txt -h test_hier.txt sequences/*.fasta
```

Processing Classifier Output

The file `test_hier.txt` is then converted to a phyloseq object with the function `hier2phyloseq` in the package `RDPutils`.

Open an R session and do the following:

```
# setwd("C:/RDPTools_phyloseq_tutorial") # modify path as appropriate
suppressWarnings(suppressMessages(library(phyloseq)))
packageVersion("phyloseq")
```

```
## [1] '1.24.2'
```

```
suppressPackageStartupMessages(library(RDPutils))
packageVersion("RDPutils")
```

```
## [1] '1.4.1'
```

Create the phyloseq object `class.expt` from classifier output in hier format.

```
class.expt <- hier2phyloseq("test_hier.txt")
class.expt
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 615 taxa and 9 samples ]
## tax_table()   Taxonomy Table:    [ 615 taxa by 6 taxonomic ranks ]
```

`class.expt` contains an OTU table and a taxonomy table.

Add sample data from a comma delimited file created in Excel. The sample names must match exactly.

```
sam.data <- read.csv(file="sample.data.csv", row.names=1, header=TRUE)
sample_data(class.expt) <- sam.data
class.expt
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 615 taxa and 9 samples ]
## sample_data() Sample Data:       [ 9 samples by 25 sample variables ]
## tax_table()   Taxonomy Table:    [ 615 taxa by 6 taxonomic ranks ]
```

The classifier includes non-fungal 28S rRNA gene sequences to improve classification. The next step is to remove all of the non-fungal OTUs.

Get rank names.

```
rank_names(class.expt)
```

```
## [1] "Domain" "Phylum" "Class"  "Order"  "Family" "Genus"
```

Get unique domain names.

```
get_taxa_unique(class.expt, taxonomic.rank="Domain")
```

```
## [1] "Fungi"                    "Animalia"
## [3] "Eukaryota incertae sedis" "Viridiplantae"
## [5] "Alveolata"                ""
## [7] "Amoebozoa"
```

Subset to include only the Fungi.

```
fungi <- subset_taxa(class.expt, Domain=="Fungi")
fungi
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 588 taxa and 9 samples ]
## sample_data() Sample Data:       [ 9 samples by 25 sample variables ]
## tax_table()   Taxonomy Table:    [ 588 taxa by 6 taxonomic ranks ]
```

```
get_taxa_unique(fungi, taxonomic.rank="Domain")
```

```
## [1] "Fungi"
```

`fungi` can now be used for data analysis. It is based on fungal sequences only and contains counts for OTUs by sample, classification of those OTUs, and sample data such as treatment factors and environmental variables.

Examples of Data Analysis

class.expt

First we will use `class.expt` to demonstrate some of the analyses possible once we have our data in a `phyloseq` object. The nine samples in `class.expt` are 28S rRNA gene libraries prepared from soil collected from three Wisconsin fields planted for at least 10 years with each of three crops: corn, switchgrass, and mixed prairie species. Because the sequences could not be aligned and treed, `class.expt` contains only an OTU table, taxonomy table, and sample data table. The tree and refseq slots in `class.expt` are empty.

Unless stated otherwise, the examples below make use of `phyloseq` functions and wrappers.

Inspect sample variables.

```
sample_variables(fungi)
```

```
##  [1] "ext.int"   "treatment" "crop"      "location"  "labels"
##  [6] "soil"      "P"         "K"         "Ca"        "Mg"
## [11] "S"         "Zn"        "B"         "Mn"        "Fe"
## [16] "Cu"        "Al"        "Na"        "Total.C"   "Total.N"
## [21] "pH"        "Sand"      "Silt"      "Clay"      "Texture"
```

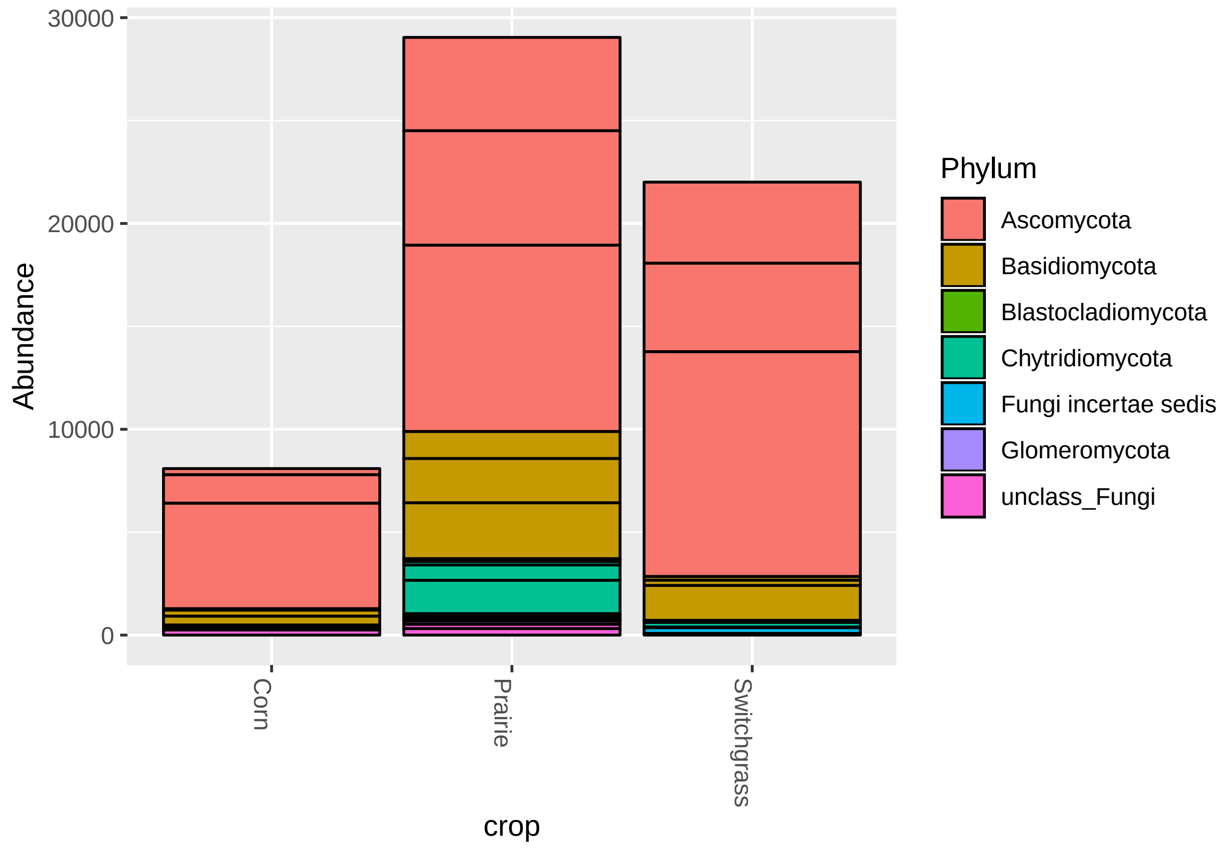
List the crops.

```
unique(sample_data(fungi)[,"crop"])
```

```
##                  crop
## WIE.Co.1         Corn
## WIE.Pr.1      Prairie
## WIE.Sw.1 Switchgrass
```

Make a bar plot of phyla by crop. First agglomerate sequences by phyla.

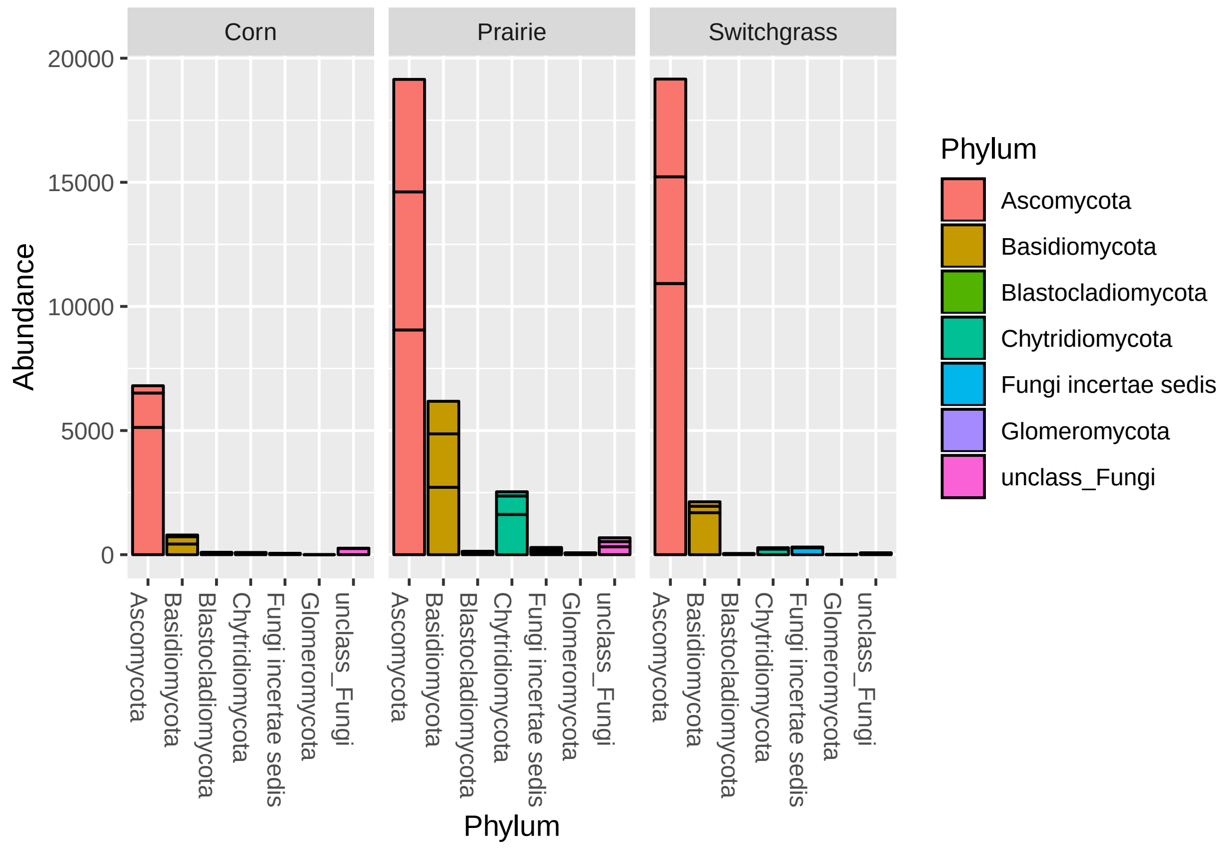
```r
fungi.p <- tax_glom(fungi, taxrank="Phylum")
plot_bar(fungi.p, x="crop", fill="Phylum")
```



Values (number of sequences) are stacked in order in each bar, with the greatest at the bottom, and replicates are depicted individually. Thus for a given phylum the replicates may not be contiguous.

Make separate plots for each crop.

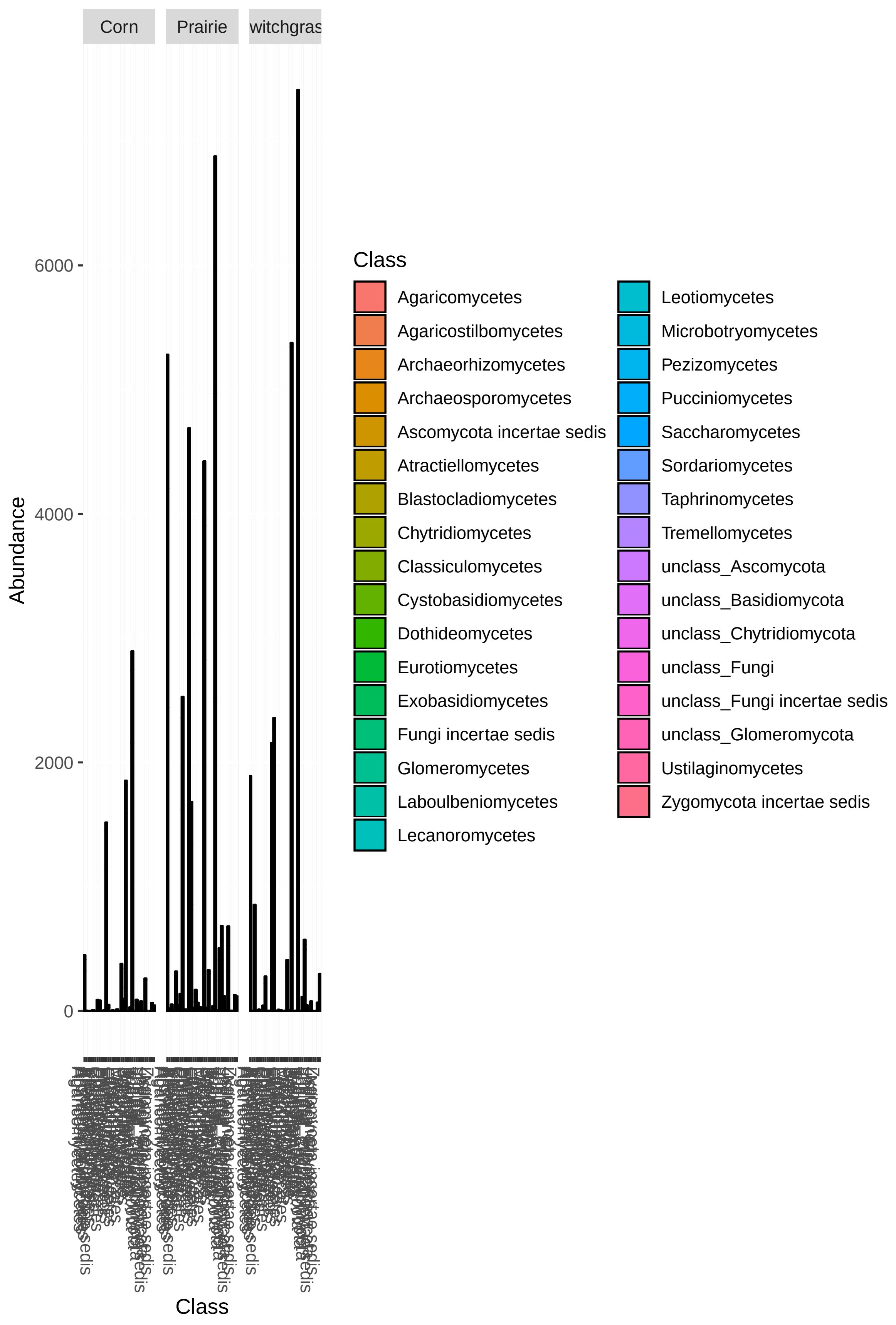
```r
plot_bar(fungi.p, x="Phylum", fill="Phylum", facet_grid=~crop)
```



This presentation is easier to interpret. Notice, for example, that while Ascomycota is present in all samples, it is much less abundant in two of the three corn replicates. Also, nearly all of the Chytridiomycota sequences occur in two of the prairie samples.

Look at classes within Ascomycota (the most abundant phylum).

```
asco <- subset_taxa(fungi, Phylum="Ascomycota")
asco.c <- tax_glom(asco, taxrank="Class")
plot_bar(asco.c, x="Class", fill="Class", facet_grid=~crop)
```



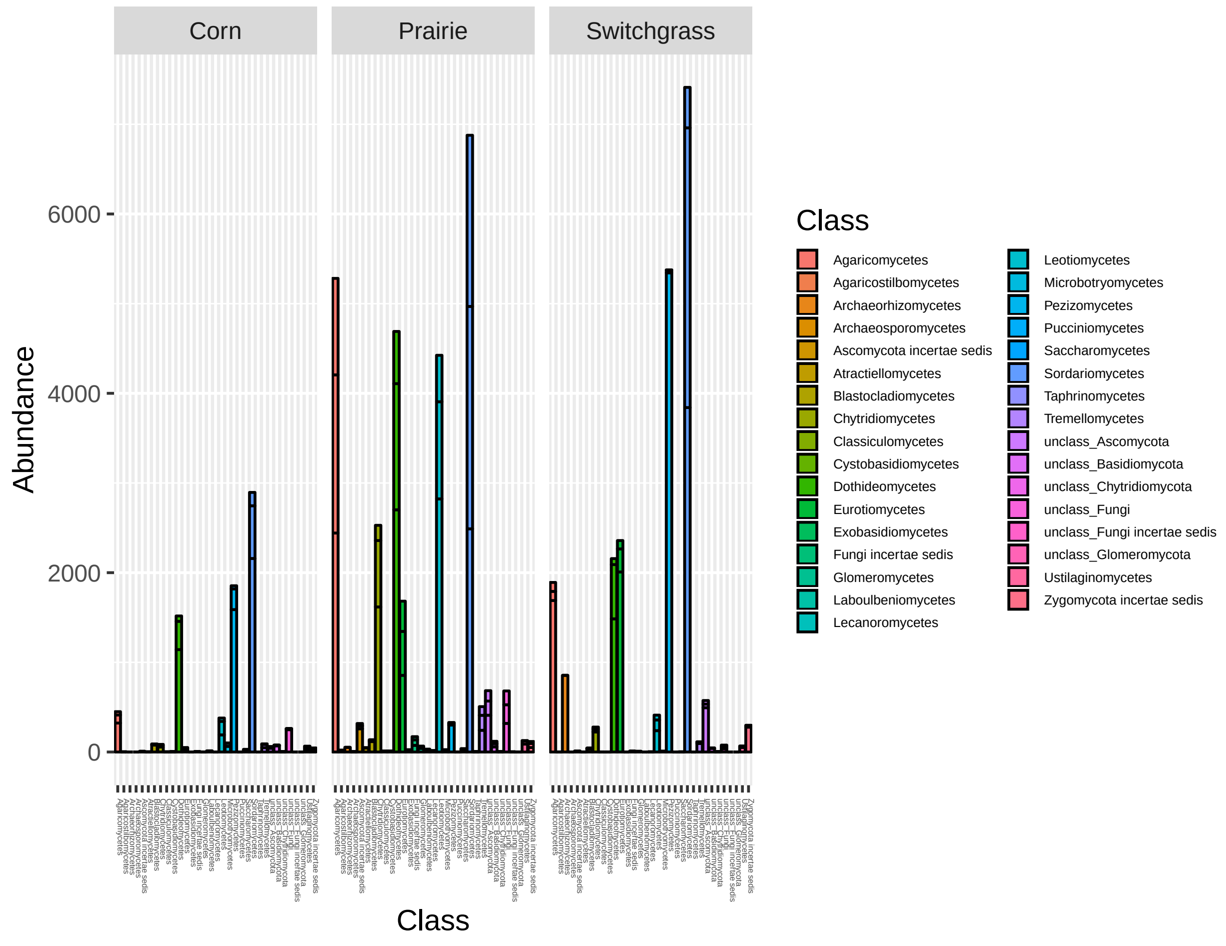
Corn
Prairie
witchgras
Abundance
6000
4000
2000
0
Class
Class
Agaricomycetes
Agaricostilbomycetes
Archaeorhizomycetes
Archaeosporomycetes
Ascomycota incertae sedis
Atractiellomycetes
Blastocladiomycetes
Chytridiomycetes
Classiculomycetes
Cystobasidiomycetes
Dothideomycetes
Eurotiomycetes
Exobasidiomycetes
Fungi incertae sedis
Glomeromycetes
Laboulbeniomycetes
Lecanoromycetes
Leotiomycetes
Microbotryomycetes
Pezizomycetes
Pucciniomycetes
Saccharomycetes
Sordariomycetes
Taphrinomycetes
Tremellomycetes
unclass_Ascomycota
unclass_Basidiomycota
unclass_Chytridiomycota
unclass_Fungi
unclass_Fungi incertae sedis
unclass_Glomeromycota
Ustilaginomycetes
Zygomycota incertae sedis

This plot is messy because there are so many classes and we have used `phyloseq`'s default values for plotting. Phyloseq uses `ggplot2` graphics. If we dig into how these graphics work, we can modify the plot for better readability. We will do the same for plots later in this tutorial.

```r
suppressPackageStartupMessages(library(ggplot2))# Necessary to modify phyloseq graphs.
packageVersion("ggplot2")
```

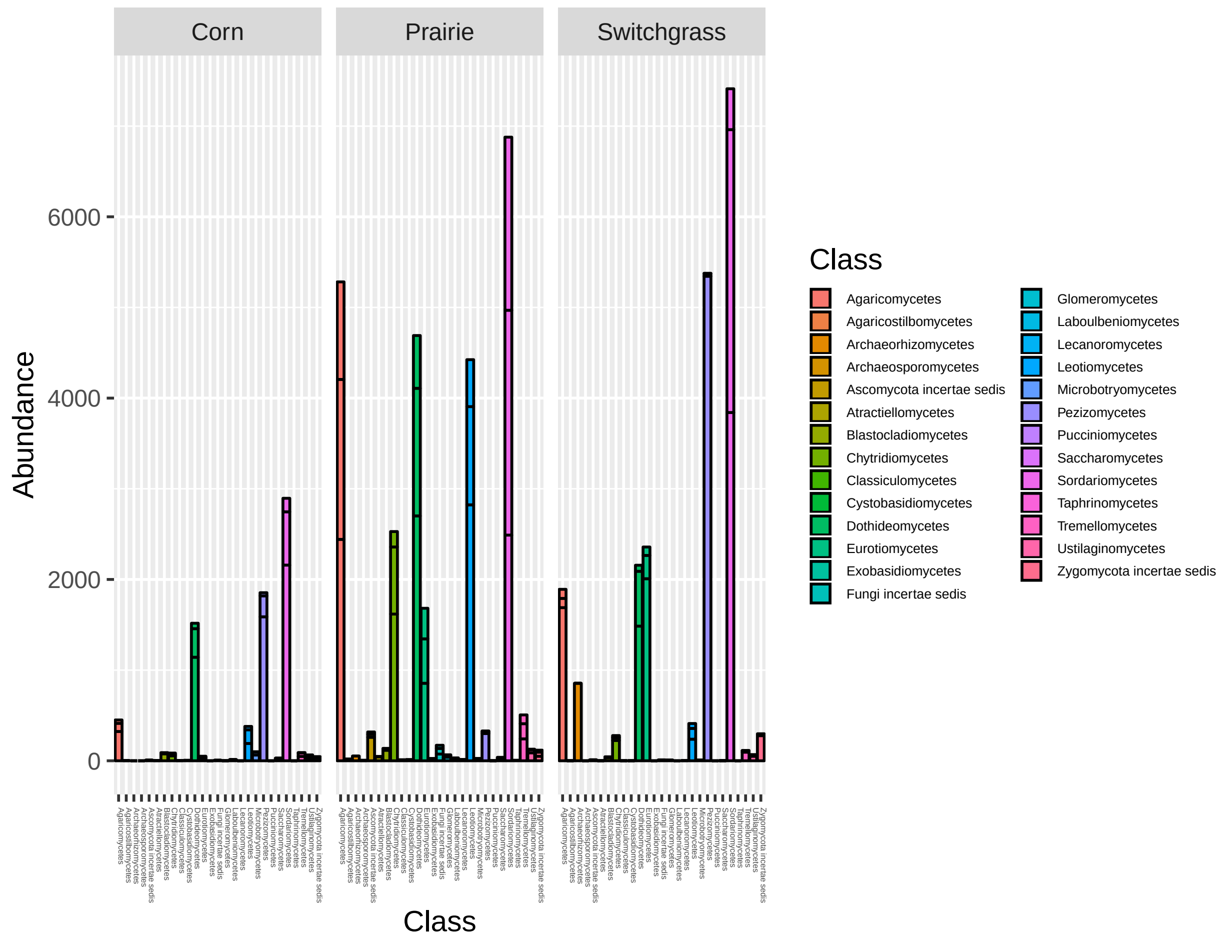
```
## [1] '3.0.0'
```

```r
library(grid) # Otherwise, function unit (below) is not recognized;
#             part of base R installation but must be called.
p1 <- plot_bar(asco.c, x="Class", fill="Class", facet_grid=~crop)
p1 <- p1 + theme(legend.key.size = unit(0.3, "cm"),
           axis.text.x = element_text(size=3, vjust=0),
           legend.text = element_text(size=5))
p1
```



We can also remove groups unclassified at the class level. These are likely disparate groups, so little can be said about them. Removing them makes the plot slightly less busy.

```r
classified <- as.vector(substr(tax_table(asco.c)[,"Class"], 0, 7)!="unclass")
asco.c.classified <- prune_taxa(classified, asco.c)
p2 <- plot_bar(asco.c.classified, x="Class", fill="Class", facet_grid=~crop)
p2 <- p2 + theme(legend.key.size = unit(0.3, "cm"),
           axis.text.x = element_text(size=3, vjust=0),
          legend.text = element_text(size=5))
p2
```



With some finagling you can reduce the number of classes in the plot further, say to only those representing at least 1% of sequences in at least one sample. First extract the OTU table with samples in columns (the orientation in `phyloseq`. Convert to percentages.

```
otu <- otu_table(asco.c.classified)
otu.pc <- prop.table(otu, margin = 2)*100
```

Get a logical vector of OTUs to keep.

```
keep <- apply(otu.pc, 1, max)>=1
table(keep)
```

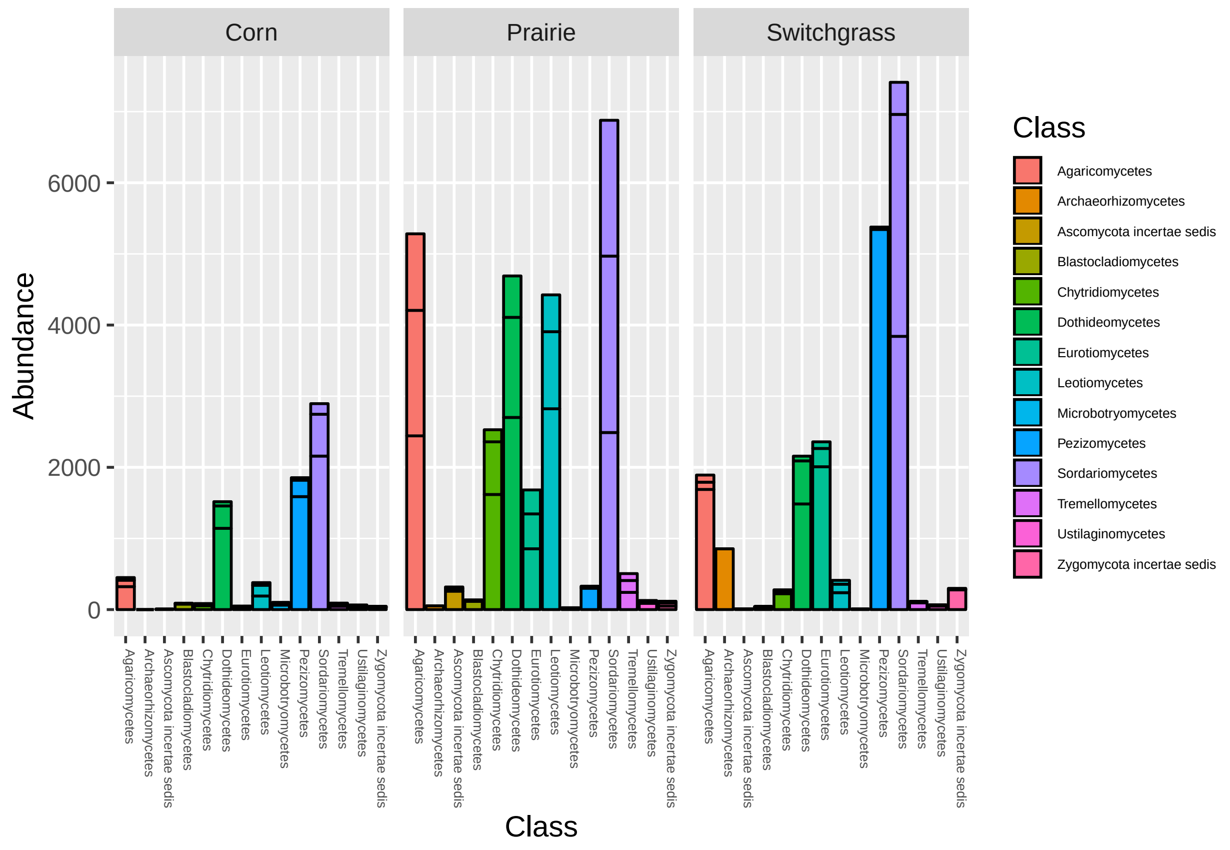
```
## keep
## FALSE  TRUE
##    13    14
```

Only 14 of the 27 OTUs/classes represent at least 1% of the sequences in at least one sample. Prune to these 14 classes.

```
temp <- prune_taxa(keep, asco.c.classified)
```

Plot as above.

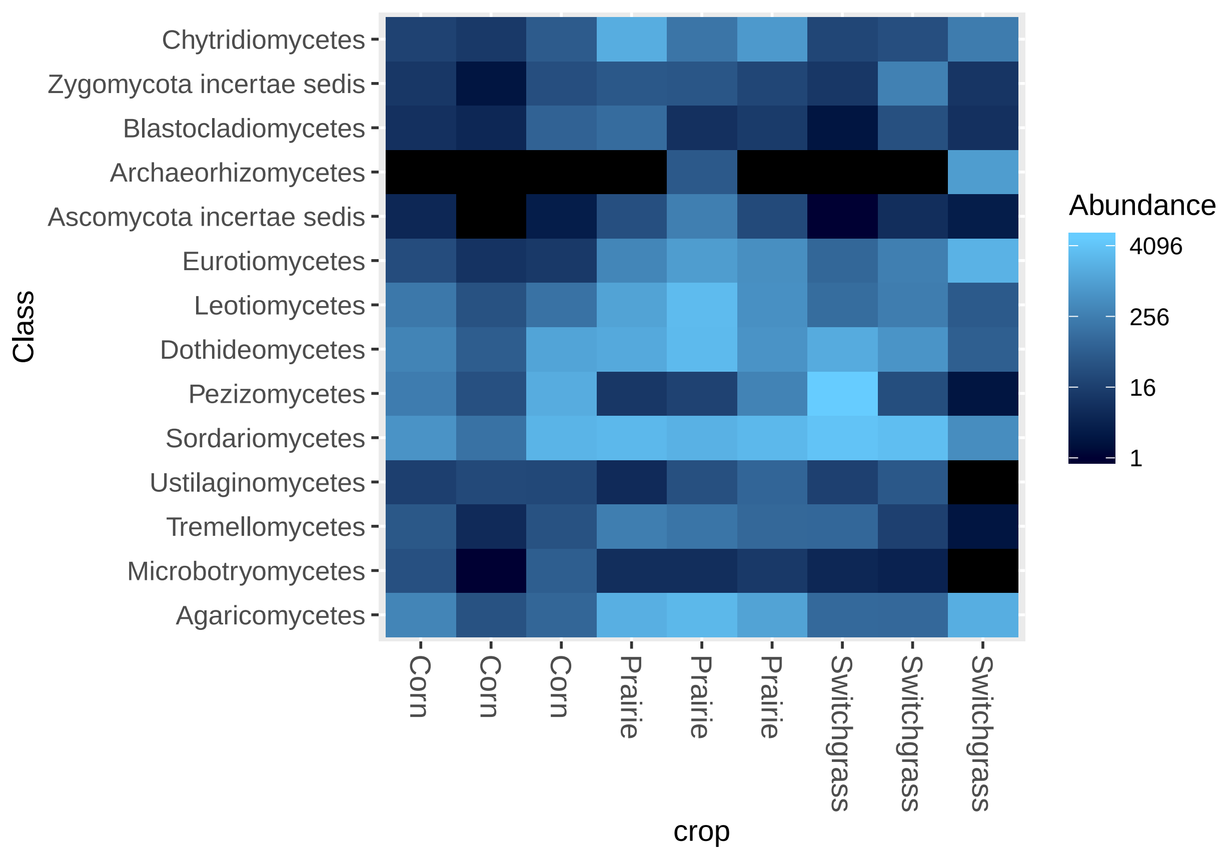
```
p3 <- plot_bar(temp, x="Class", fill="Class", facet_grid=~crop)
p3 + theme(legend.key.size = unit(0.4, "cm"),
           axis.text.x = element_text(size=5, vjust=0),
           legend.text = element_text(size=5))
```



You can easily plot this same information as a heat map.

```
plot_heatmap(temp,  method=NULL, sample.label="crop", taxa.label="Class")
```

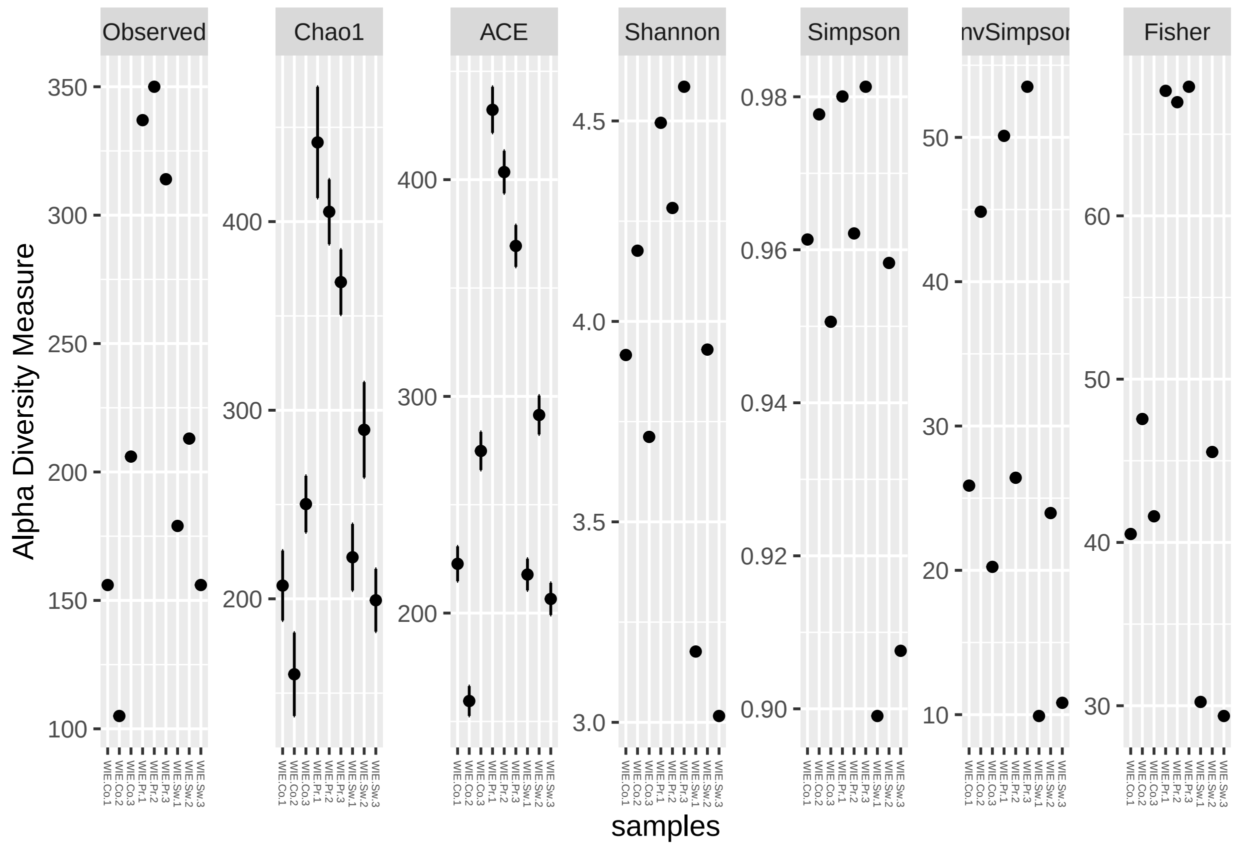
```
## Warning: Transformation introduced infinite values in discrete y-axis
```



Plot richness estimates for the entire fungal data set.

```
p4 <- plot_richness(fungi)
p4 + theme(axis.text.x = element_text(size=4))
```

```
## Warning: Removed 45 rows containing missing values (geom_errorbar).
```



By most measures, the prairie samples are more diverse.

Inspect sample sizes. From the above bar plots it is evident that there are fewer sequences in the corn samples.

```
sample_sums(fungi)
```

```
## WIE.Co.1 WIE.Co.2 WIE.Co.3 WIE.Pr.1 WIE.Pr.2 WIE.Pr.3 WIE.Sw.1 WIE.Sw.2
##     1864      385     5842     9786    12403     6852    11239     4849
## WIE.Sw.3
##     5921
```

WIE.Co.2 should be removed for having too few sequences.

```
fungi.8 <- prune_samples(sample_names(fungi)!="WIE.Co.2", fungi)
sample_names(fungi.8)
```

```
## [1] "WIE.Co.1" "WIE.Co.3" "WIE.Pr.1" "WIE.Pr.2" "WIE.Pr.3" "WIE.Sw.1"
## [7] "WIE.Sw.2" "WIE.Sw.3"
```

You can also sub-sample the samples to have the same number of sequences. `rngseed=TRUE` sets the random seed generator (to 711 by default) for reproducibility.

```
fungi.8.r <- rarefy_even_depth(fungi.8, rngseed=TRUE)
```

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

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

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

## ...
## 171OTUs were removed because they are no longer
## present in any sample after random subsampling
```

```
## ...
```

```r
sample_sums(fungi.8.r)
```

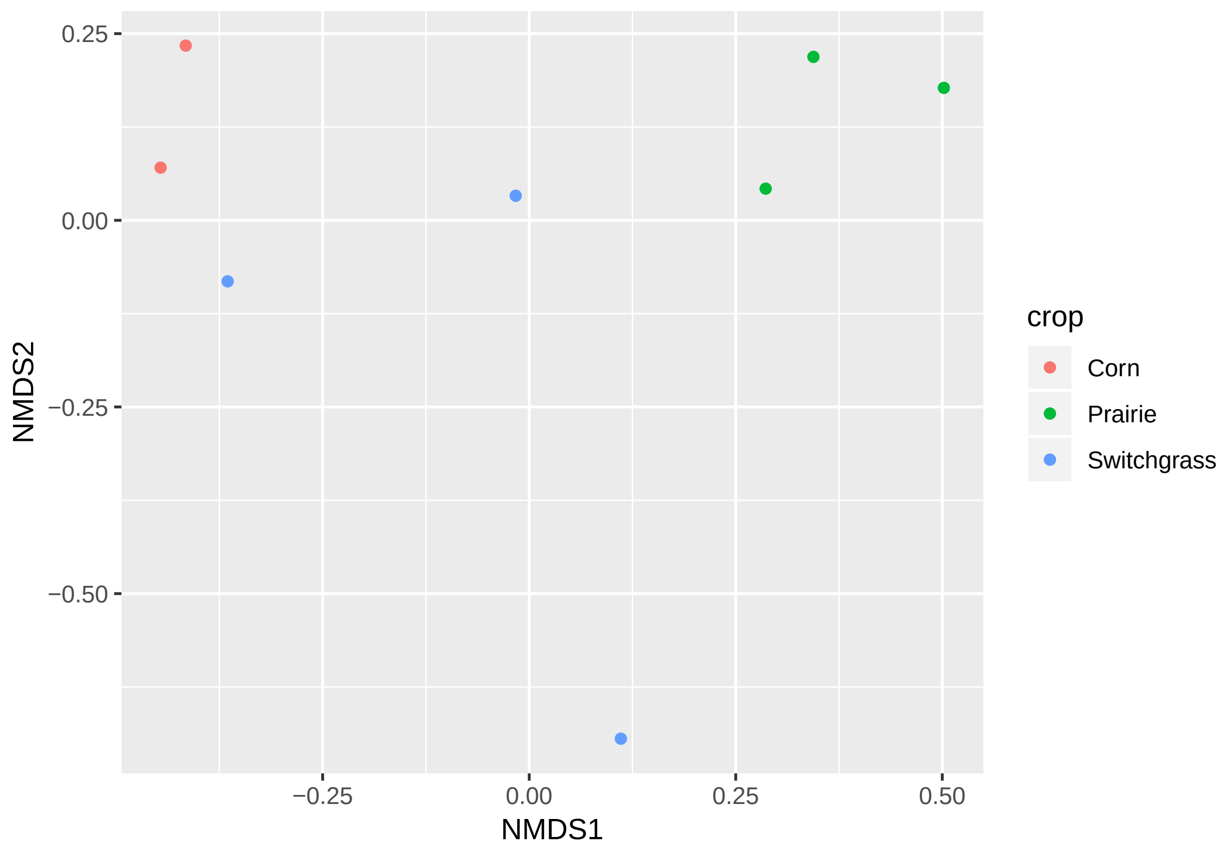
```
## WIE.Co.1 WIE.Co.3 WIE.Pr.1 WIE.Pr.2 WIE.Pr.3 WIE.Sw.1 WIE.Sw.2 WIE.Sw.3
##     1864     1864     1864     1864     1864     1864     1864     1864
```

Now make an NMDS ordination plot based on Bray distances.

```r
ord1 <- ordinate(fungi.8.r, method="NMDS", distance="bray")
```

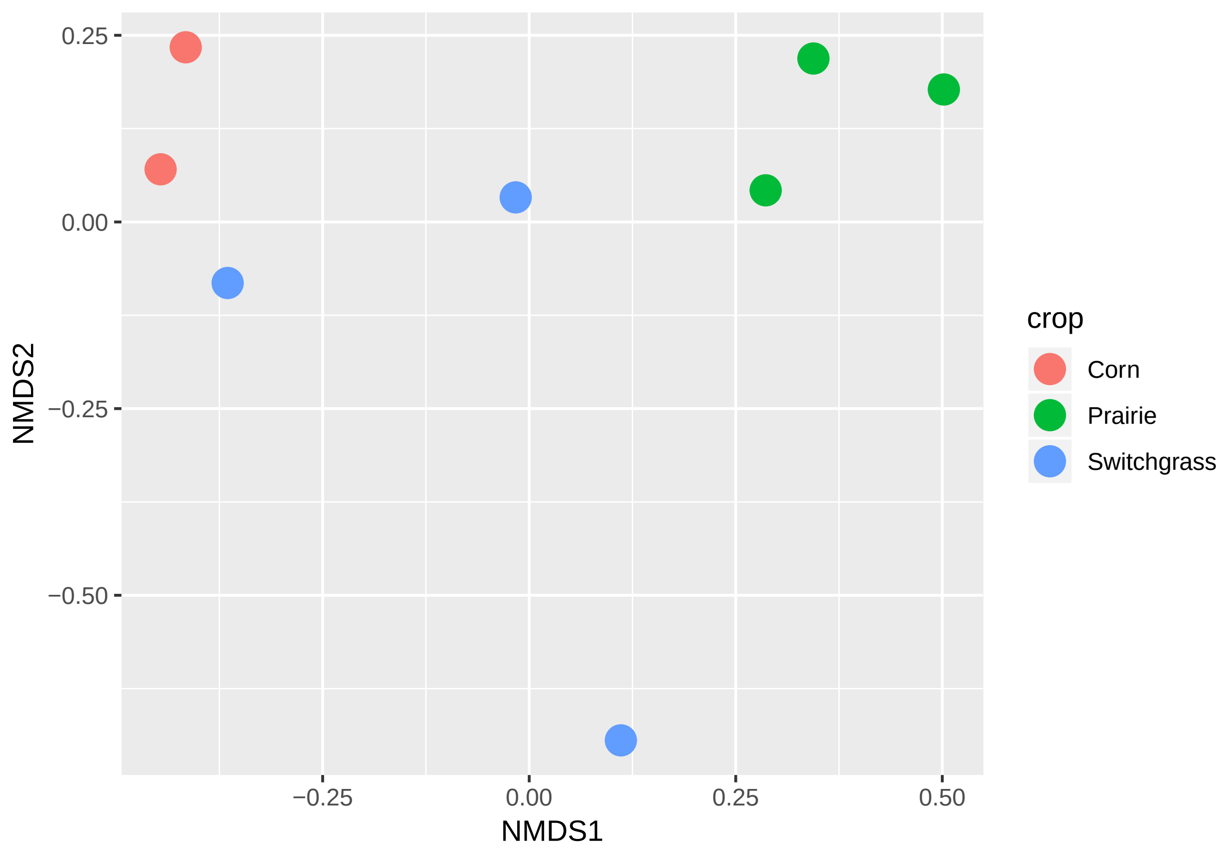
```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.04353246
## Run 1 stress 0.2180039
## Run 2 stress 0.2262215
## Run 3 stress 0.04353172
## ... New best solution
## ... Procrustes: rmse 0.0003044651  max resid 0.0005428531
## ... Similar to previous best
## Run 4 stress 0.04353174
## ... Procrustes: rmse 2.808062e-05  max resid 4.497213e-05
## ... Similar to previous best
## Run 5 stress 0.04353215
## ... Procrustes: rmse 0.0002134629  max resid 0.0003717185
## ... Similar to previous best
## Run 6 stress 0.2165193
## Run 7 stress 0.04353175
## ... Procrustes: rmse 3.841148e-05  max resid 8.210201e-05
## ... Similar to previous best
## Run 8 stress 0.05283507
## Run 9 stress 0.04353179
## ... Procrustes: rmse 4.50035e-05  max resid 7.918342e-05
## ... Similar to previous best
## Run 10 stress 0.04353168
## ... New best solution
## ... Procrustes: rmse 6.990801e-05  max resid 0.000112127
## ... Similar to previous best
## Run 11 stress 0.2165193
## Run 12 stress 0.04353168
## ... Procrustes: rmse 4.429026e-05  max resid 6.867676e-05
## ... Similar to previous best
## Run 13 stress 0.0435318
## ... Procrustes: rmse 0.0001279522  max resid 0.0002097576
## ... Similar to previous best
## Run 14 stress 0.04353172
## ... Procrustes: rmse 0.0002260578  max resid 0.0003734653
## ... Similar to previous best
## Run 15 stress 0.04357853
## ... Procrustes: rmse 0.04493335  max resid 0.08582427
## Run 16 stress 0.04353227
## ... Procrustes: rmse 0.0003179837  max resid 0.0005647814
## ... Similar to previous best
## Run 17 stress 0.05194343
## Run 18 stress 0.04353165
## ... New best solution
## ... Procrustes: rmse 6.456963e-05  max resid 0.000102943
## ... Similar to previous best
## Run 19 stress 0.0435321
## ... Procrustes: rmse 0.0002569775  max resid 0.0004465224
## ... Similar to previous best
## Run 20 stress 0.04353164
## ... New best solution
## ... Procrustes: rmse 3.685276e-05  max resid 5.952211e-05
## ... Similar to previous best
## *** Solution reached
```

```
p1 <- plot_ordination(fungi.8.r, ord1, color="crop")
p1
```



The symbols always seem too small. You can plot larger ones over the top of the smaller ones thus:

```
p1 <- p1 + geom_point(size=5)
p1
```



And of course you can always take a `phyloseq` object apart and use functions in other R packages such as `vegan, labdsv, ade4, ape`.... and your own functions.

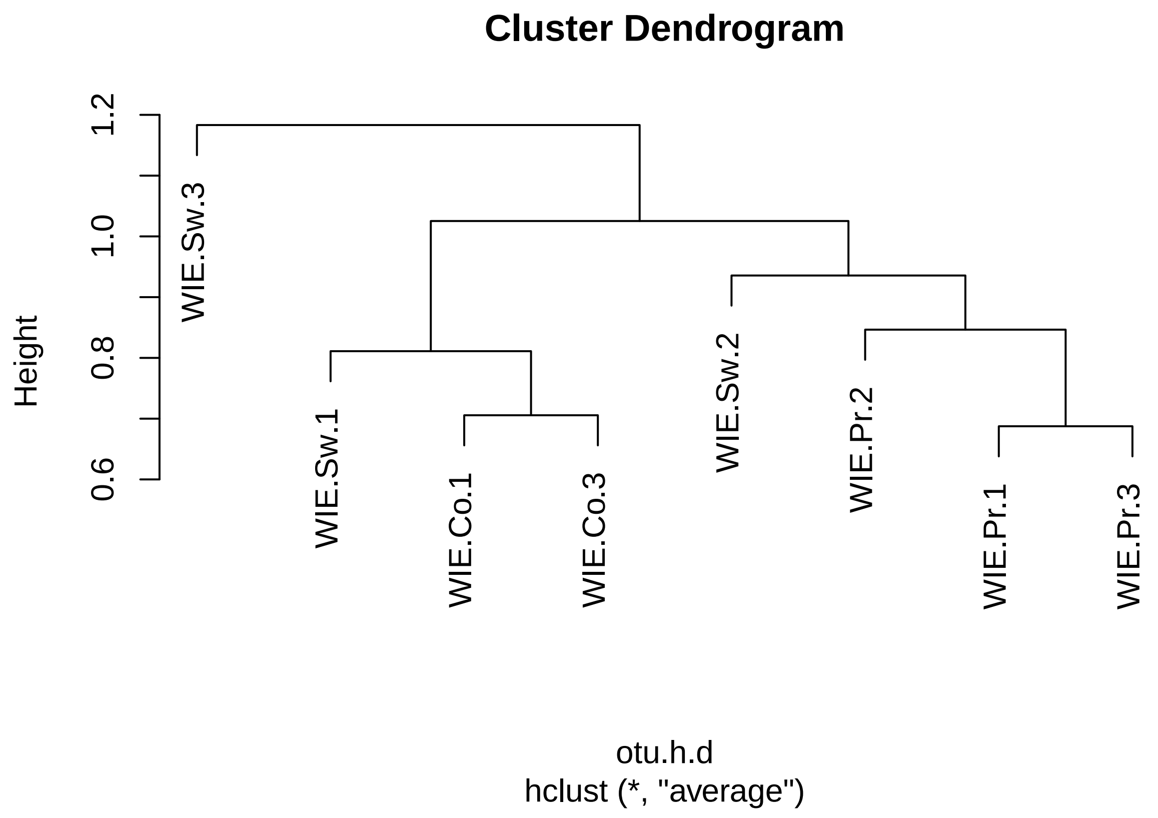
As an example, `phyloseq` does not provide a wrapper for plotting a cluster dendogram of the samples, but you can do this in `vegan` (Oksanen et al., 2013).

First define functions to extract vegan-compatible OTU and sample data tables from a `phyloseq` object:

```
veganotu <- function(physeq) {
  OTU <- otu_table(physeq)
  if (taxa_are_rows(OTU)) {
    OTU <- t(OTU)
  }
  OTU <- as(OTU, "matrix")
  return(OTU)
}
vegansam <- function(physeq) {
  sam <- sample_data(physeq)
  i <- sapply(sam, is.numeric)
  j <- sapply(sam, is.character)
  k <- sapply(sam, is.factor)
  sam <- as.matrix(sam)
  sam <- as.data.frame(sam)
  sam[i] <- lapply(sam[i], as.numeric)
  sam[j] <- lapply(sam[j], as.character)
  sam[k] <- lapply(sam[k], as.factor)
  return(sam)
}
```

And then use `veganotu` to extract the OTU table from `fungi.8.r` and plot a dendrogram based on Hellinger distance and average linkage clustering.

```
suppressPackageStartupMessages(library(vegan))
otu <- veganotu(fungi.8.r)
otu.h <- decostand(otu, "hellinger")
otu.h.d <- vegdist(otu.h, "euclidean")
plot(hclust(otu.h.d, method="average"))
```

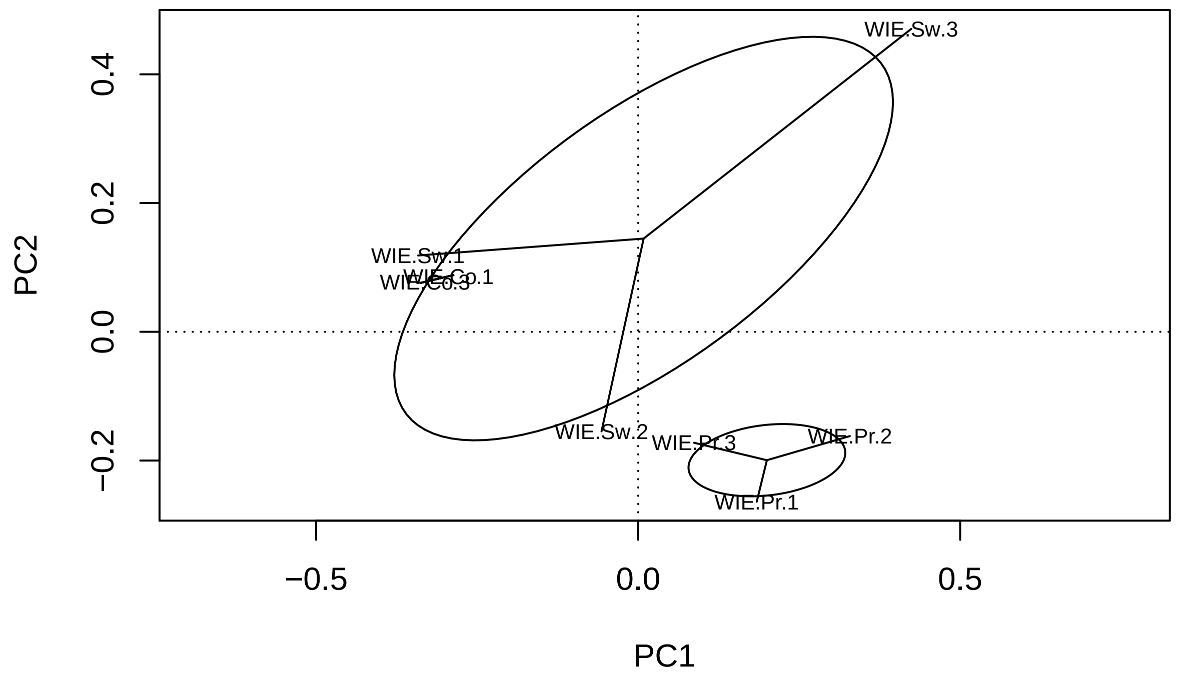


We can also extract the sample data and plot a PCA ordination based on the Hellinger distances with ellipses and spiders differentiating the crops. When given only an OTU matrix, the `rda` function performs a principal components analysis based on euclidean distances. So to get a PCA ordination based on Hellinger distances, just supply an OTU matrix that has been Hellilnger transformed.

```
pca <- rda(otu.h)
plot(pca, display="sites", scaling=1, type="text")
sam.data <- vegansam(fungi.8.r)
ordiellipse(pca, groups=sam.data$crop, display="sites", scaling=1, kind="sd")
```

```
## Warning in chol.default(cov, pivot = TRUE): the matrix is either rank-
## deficient or indefinite
```

```
ordispider(pca, groups=sam.data$crop, display="sites", scaling=1)
```



The result is similar to the NMDS plot, showing the corn and prairie samples well separated, with switchgrass samples intermediate, more variable, and partially overlapping the corn samples. The cluster results yield a similar interpretation. The warning message is because we have only two corn samples.

Clst.expt

There are additional possibilities when you have a tree file in your `phyloseq` object, as is the case with `clst.expt`. The eight samples in `clst.expt` are 16S rRNA gene libraries prepared from microcosm experiments conducted with sediments collected from Thompson Creek (TC in the sample names) and Choptank River (CR). Replicate libraries (A and B) were prepared from the sediments as they were received in the laboratory (Pr_R in the sample name), and from control samples after 15 weeks of incubation (Co15). For this tutorial, the libraries were sub-sampled to 1,000 sequences per sample.

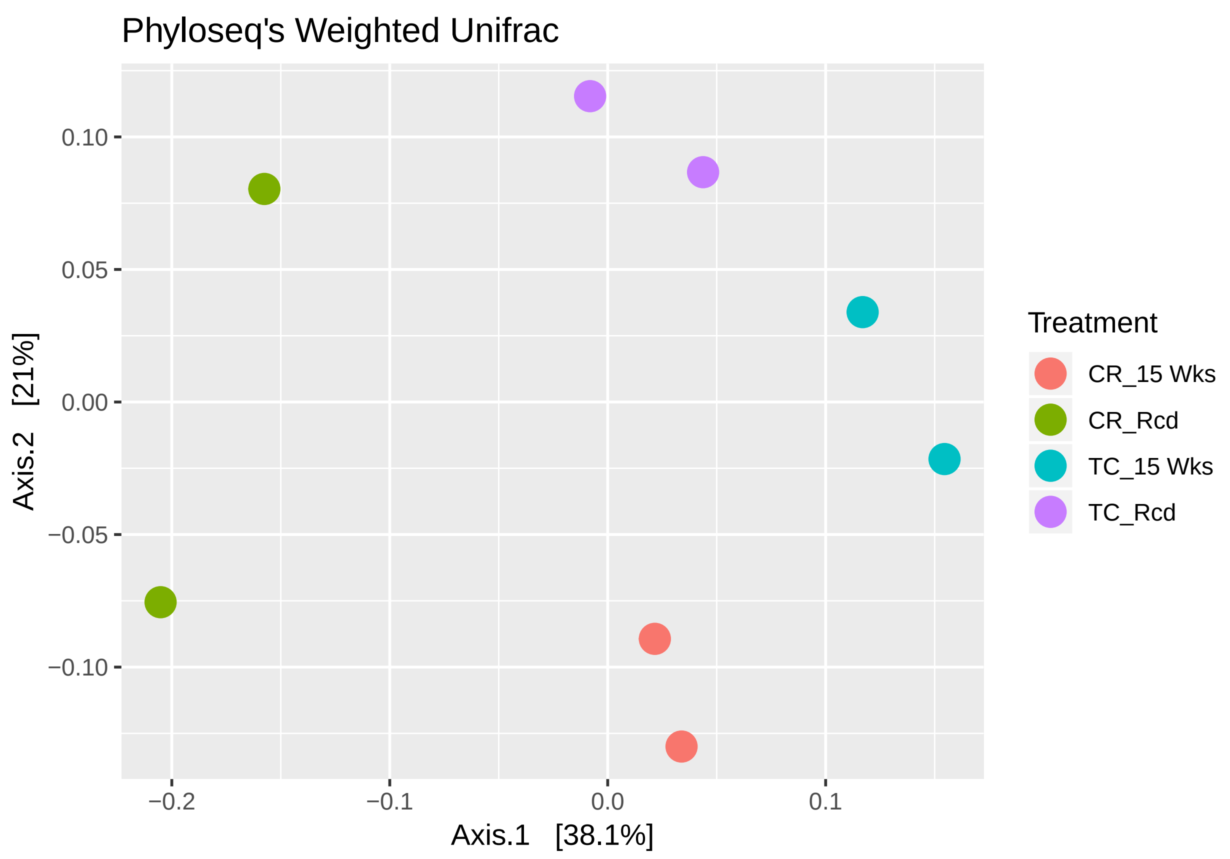
Because `clst.expt` contains a tree of the representative sequences, we can make ordination plots based on unifrac distances among samples and tree plots showing OTU distributions among samples.

Make an PCoA ordination plot based on abundance based unifrac distances with the following commands:

```
ord2 <- ordinate(clst.expt, method="PCoA", distance="unifrac", weighted=TRUE)
```

```
## Warning in UniFrac(physeq, ...): Randomly assigning root as -- cluster_100
## -- in the phylogenetic tree in the data you provided.
```

```
p2 <- plot_ordination(clst.expt, ord2, color="Treatment",
                      title="Phyloseq's Weighted Unifrac")
p2 <- p2 + geom_point(size=5)
p2
```



The GUniFrac package (Chen et al., 2012) available on CRAN can also be used to calculate unifrac distances and has additional features. Unifrac distances are traditionally calculated on either presence/absence data, or abundance data. The former can be affected by PCR and sequencing errors leading to a high number of spurious and usually rare OTUs, and the latter can give undue weight to the more abundant OTUs. GUniFrac's methods include use of a parameter alpha that controls the weight given to abundant OTUs and also a means of adjusting variances.

The function GUniFrac requires a rooted tree, but unlike phyloseq's ordination function will not try to root an unrooted one. We will apply mid-point rooting with the midpoint function from the phangorn package (Schliep, 2011).

```
suppressPackageStartupMessages(library(GUniFrac))
packageVersion("GUniFrac")
```

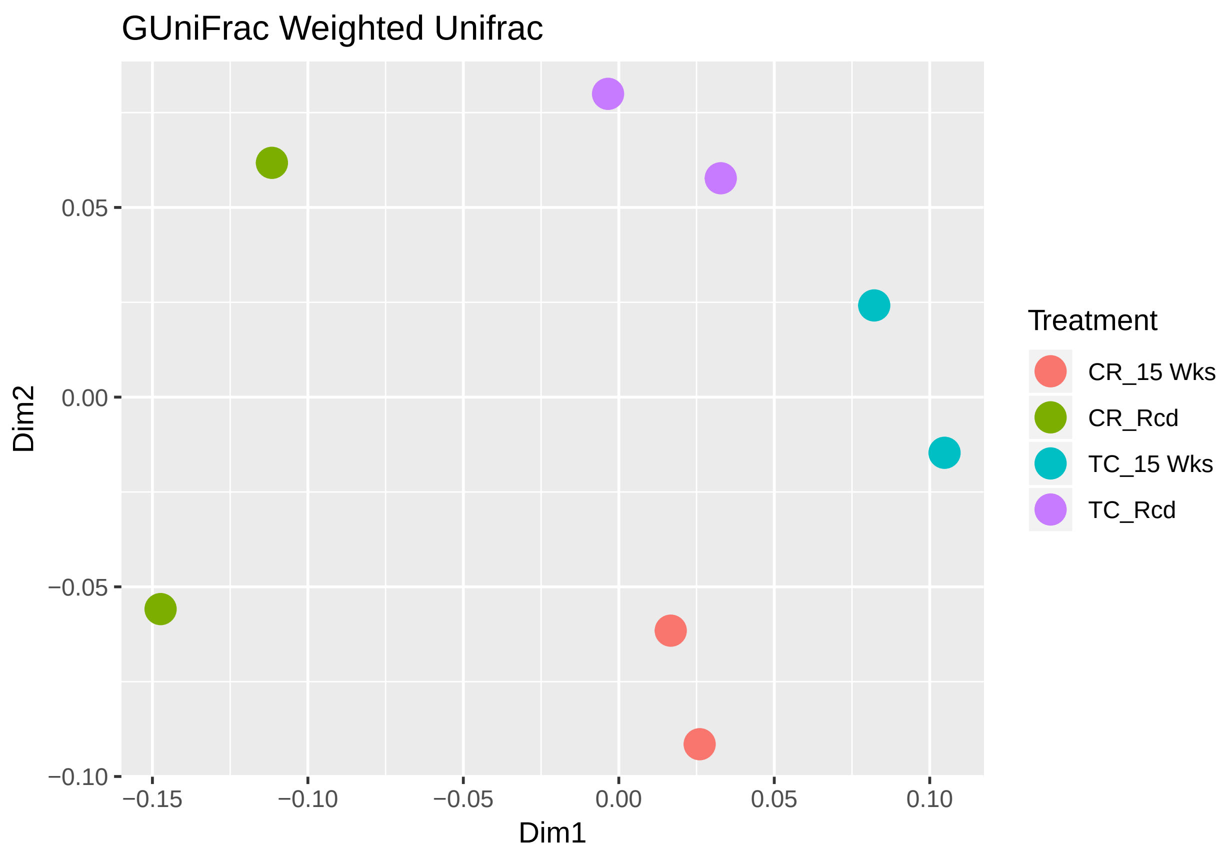
```
## [1] '1.1'
```

```
suppressPackageStartupMessages(library(phangorn))
packageVersion("phangorn")
```

```
## [1] '2.4.0'
```

```
ext.tree <- phy_tree(clst.expt)
new.tree <- midpoint(ext.tree)
com <- veganotu(clst.expt)
unifracs <- GUniFrac(com, new.tree, alpha = c(0, 0.5, 1))$unifracs
 # We can extract a variety of distance matrices with different weightings.
dw <- unifracs[, , "d_1"]  # Weighted UniFrac
du <- unifracs[, , "d_UW"]  # Unweighted UniFrac
```

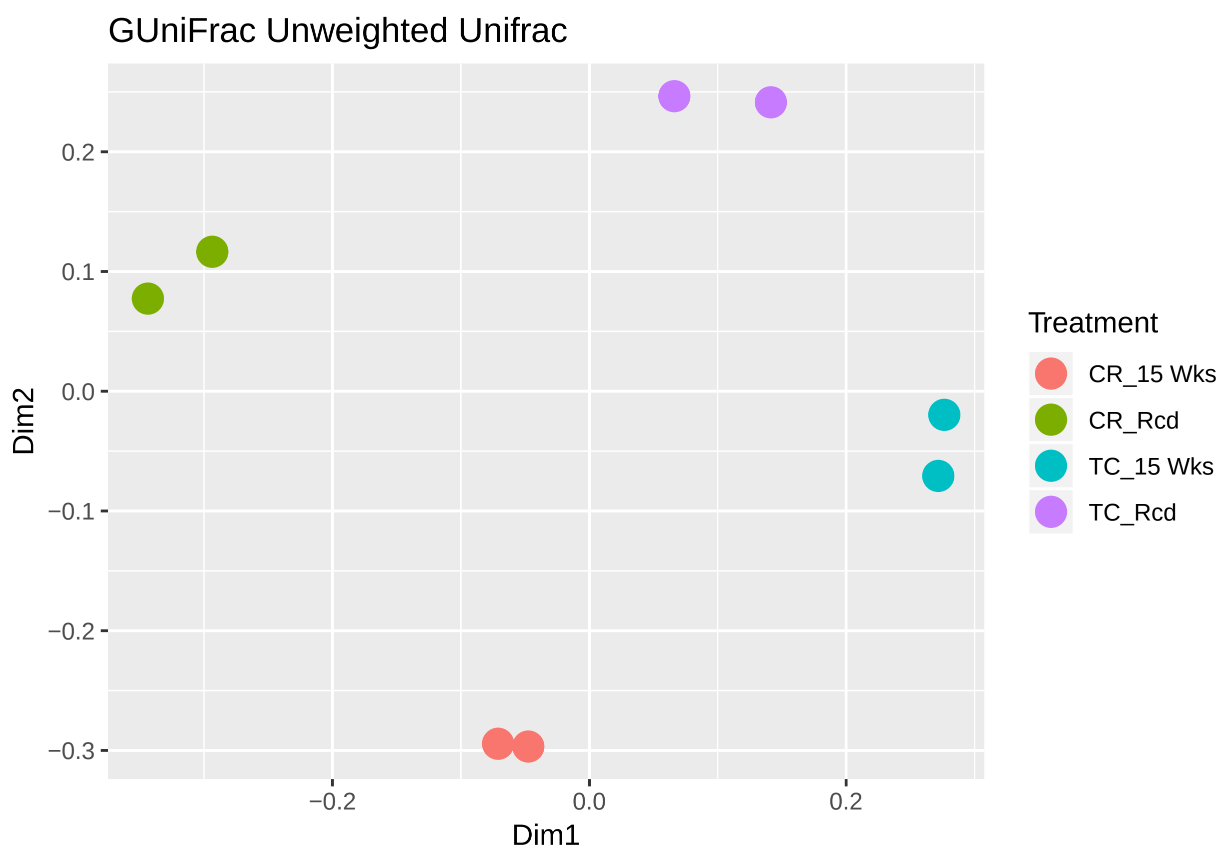
```r
dv <- unifracs[, , "d_VAW"]  # Variance adjusted weighted UniFrac
d0 <- unifracs[, , "d_0"]  # GUniFrac with alpha 0
d5 <- unifracs[, , "d_0.5"]  # GUniFrac with alpha 0.5
# use vegan's cmdscale function to make a PCoA ordination from a distance matrix.
pcoa <- cmdscale(dw, k = nrow(com) - 1, eig = TRUE, add = TRUE)
# Use phyloseq's plotting function to generate a ggplot.
plot_ordination(clst.expt, pcoa, color="Treatment",
                title="GUniFrac Weighted Unifrac") + geom_point(size=5)
```



Compare the above method to `phyloseq`'s wrapper function; the results are the same.

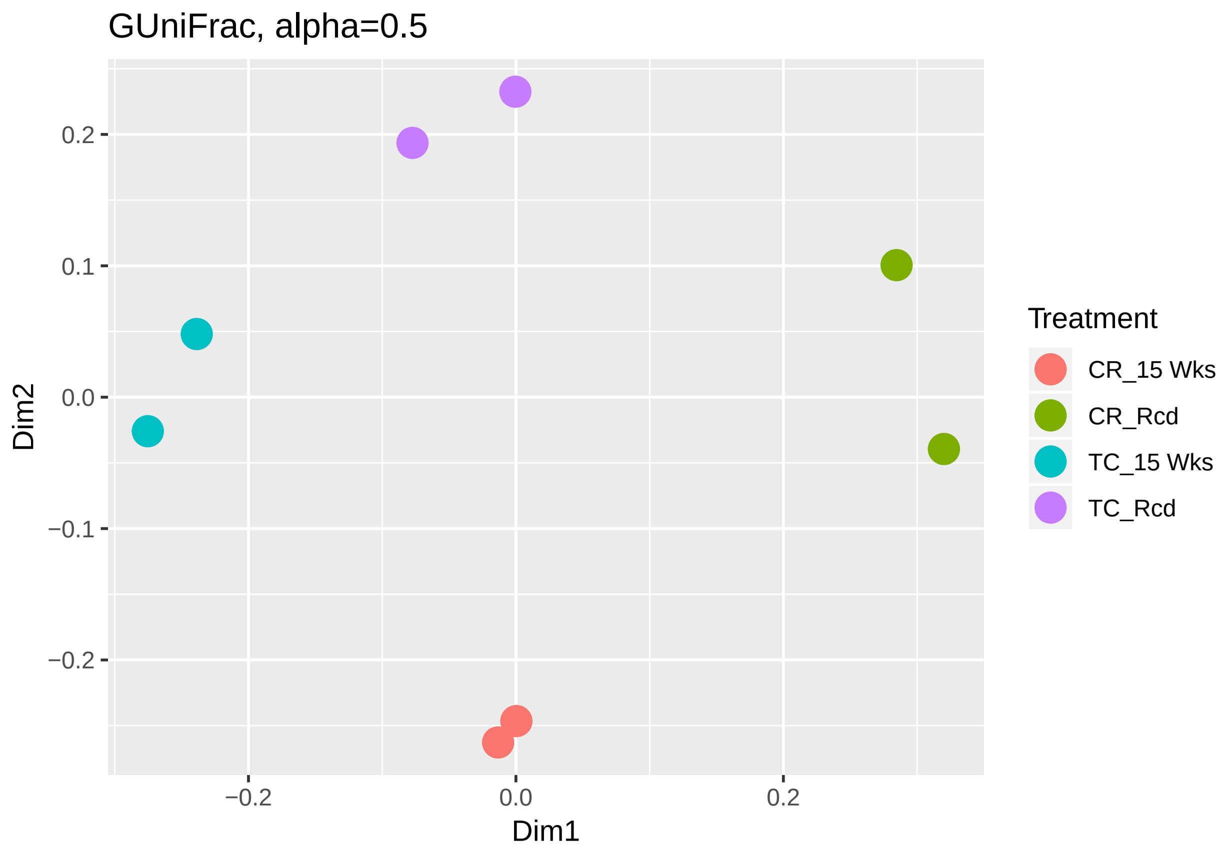
Now compare the results using other distance matrices.

```r
pcoa <- cmdscale(du, k = nrow(com) - 1, eig = TRUE, add = TRUE)
plot_ordination(clst.expt, pcoa, color="Treatment",
                title="GUniFrac Unweighted Unifrac") + geom_point(size=5)
```



Unweighted means that the OTU table is first converted to presence/absence data. Replicates are very close to each other.

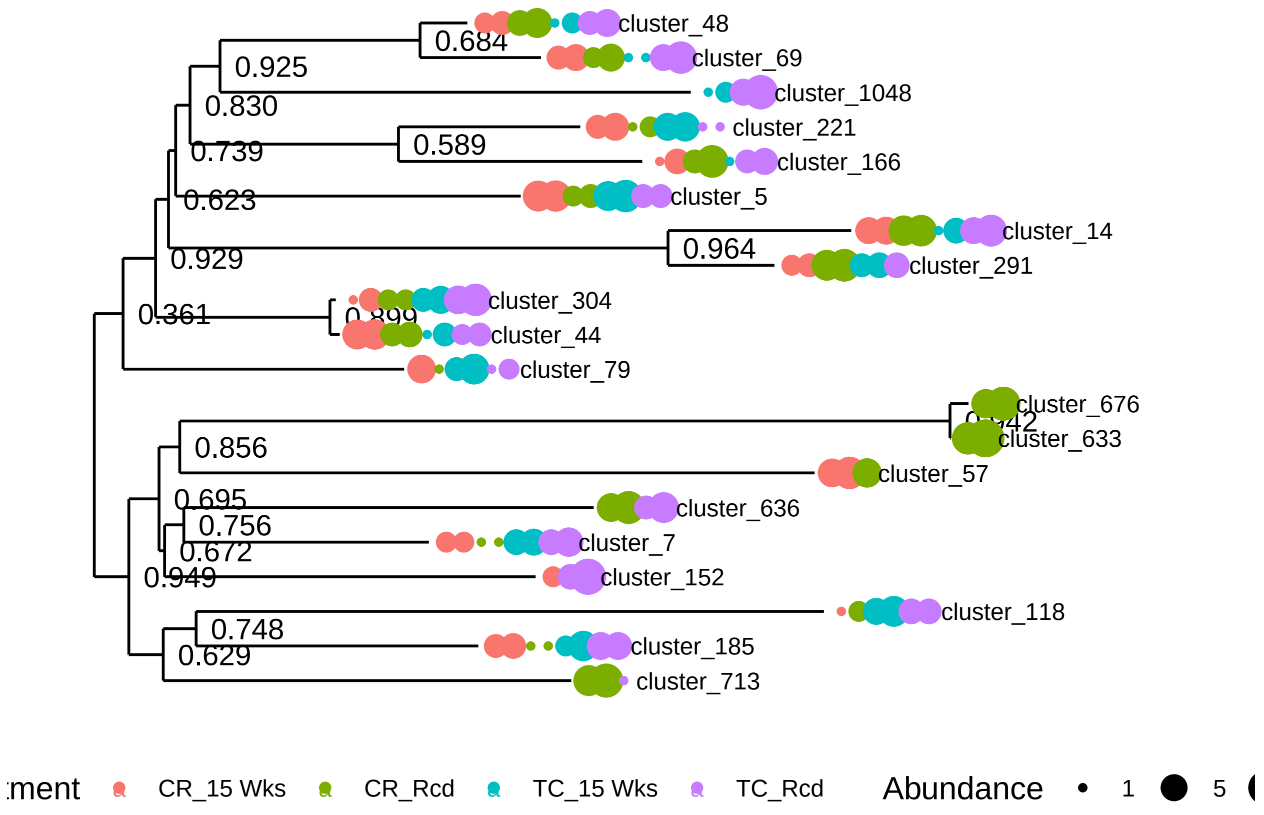
```
pcoa <- cmdscale(d5, k = nrow(com) - 1, eig = TRUE, add = TRUE)
plot_ordination(clst.expt, pcoa, color="Treatment",
                title="GUniFrac, alpha=0.5") + geom_point(size=5)
```



With alpha=0.5, abundance is still taken into account, but its importance is not as great as with the weighted option. As should be expected, results are intermediate between the weighted and unweighted versions. (As sometimes happens with ordination but is of no importance, the signs for the first axis are reversed from the previous plot.)

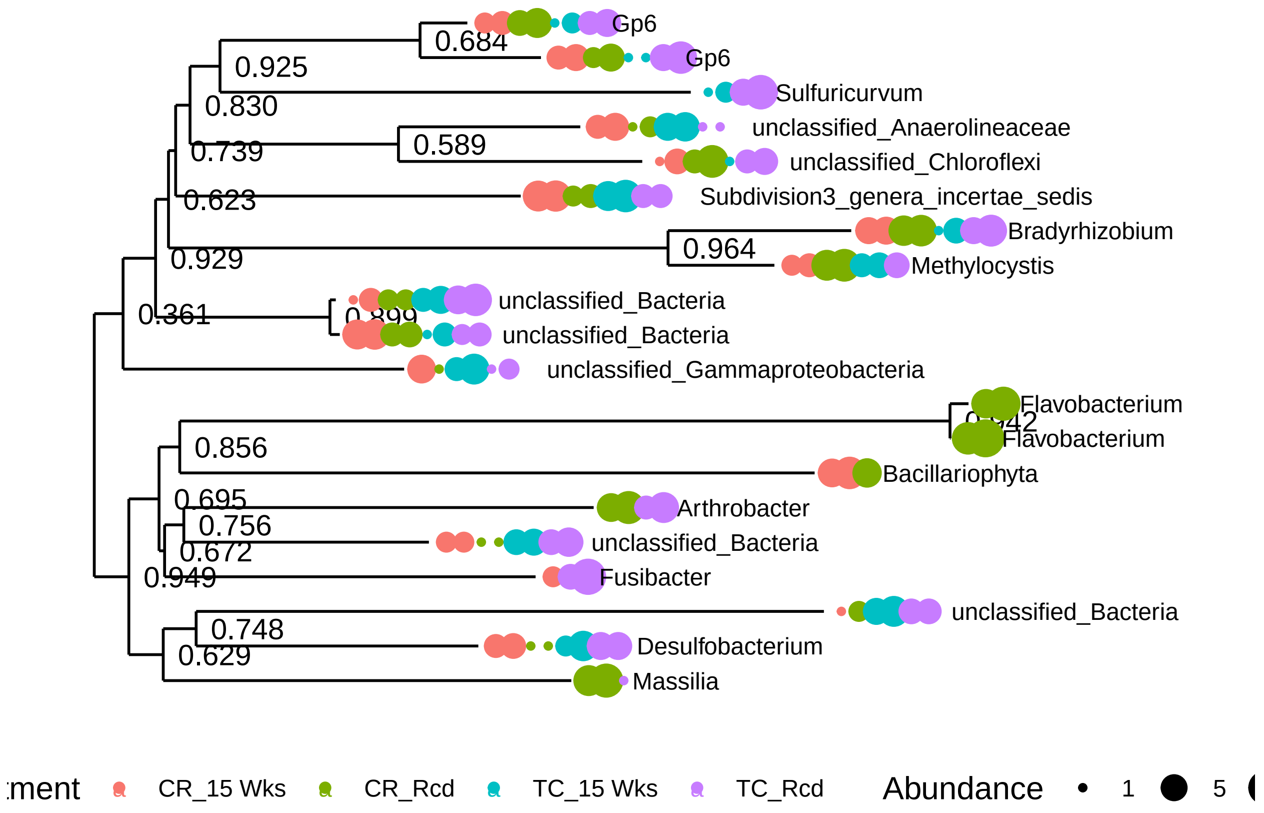
Let us examine the distribution of taxa among samples by making a tree plot of the 20 most abundant taxa.

```
clst.expt.20 = prune_taxa(names(sort(taxa_sums(clst.expt), TRUE)[1:20]), clst.expt)
tree.plot.1 <- plot_tree(clst.expt.20, color="Treatment", size="abundance",
                    label.tips="taxa_names", text.size=3, ladderize="left")
tree.plot.1 <- tree.plot.1 + theme(legend.position = "bottom",
                       legend.title = element_text(size=12),
                       legend.key = element_blank())
tree.plot.1
```



This plot makes it easy to pick out both temporal and source differences in the distributions of OTUs, and see how closely related they are at the same time. It is simple to replace the cluster number with a taxon name if you wish:

```
tree.plot.2 <- plot_tree(clst.expt.20, color="Treatment", size="abundance",
                         label.tips="Genus", text.size=3, ladderize="left")
tree.plot.2 <- tree.plot.2 + theme(legend.position = "bottom",
                       legend.title = element_text(size=12),
                       legend.key = element_blank())
tree.plot.2
```



For further examples using `phyloseq`, see the `phyloseq` GitHub page at https://github.com/joey711/phyloseq.

References

Chen, J., K. Bittinger, E. S. Charlson, C. Hoffmann, J. Lewis, G. D. Wu, R. G. Collman, F. D. Bushman, and H. Z. Li. 2012. Associating microbiome composition with environmental covariates using generalized UniFrac distances. Bioinformatics **28**:2106-2113. (http://bioinformatics.oxfordjournals.org/content/28/16/2106.full?sid=c2f0827a-bf57-4369-a482-d2d7736b515d)

Cole, J. R., Q. Wang, J. A. Fish, B. Chai, D. M. McGarrell, Y. Sun, C. T. Brown, A. Porras-Alfaro, C. R. Kuske, and J. M. Tiedje. 2014. Ribosomal Database Project: data and tools for high throughput rRNA analysis Nucl. Acids Res. **41** (Database issue):D633-D642. (http://nar.oxfordjournals.org/content/early/2013/11/26/nar.gkt1244)

McMurdie, P. J., and S. Holmes. 2012. phyloseq: A Bioconductor Package for Handling and Analysis of High-Throughput Phylogenetic Sequence Data. Pacific Symposium on Biocomputing **17**:235-246. http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0061217

Oksanen, Jari, F. Guillaume Blanchet, Roeland Kindt, Pierre Legendre, Peter R. Minchin, R. B. O'Hara, Gavin L. Simpson, Peter Solymos, M. Henry H. Stevens and Helene Wagner. 2013. vegan: Community Ecology Package. R package version 2.0-10. http://CRAN.R-project.org/package=vegan

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

Wang, Q, G. M. Garrity, J. M. Tiedje, and J. R. Cole. 2007. Naive Bayesian Classifier for Rapid Assignment of rRNA Sequences into the New Bacterial Taxonomy. Appl Environ Microbiol. **73**(16):5261-5267. (http://aem.asm.org/content/73/16/5261)