The Maternal Newborn Oral Microbiome

Irene Yang

3/28/2018

Purpose: The purpose of this final project is to replicate analysis of 16SrRNA sequence data from a recently completed pilot study. The initial analysis was completed using the MOTHUR platform. I will attempt to replicate the analysis by using an R-based workflow modeled on the one presented in class which is based on the method introduced by Callahan and colleagues (2017)

This milestone demonstrates that I have acquired and begun the exploration of my dataset. I am, at this point, on target with the timeline I set out in Milestone 1.

Weekly Goal	Task	Status
By 2/21	Install qiime2	Using R-based workflow
By 3/7	Demultiplexing and sequence quality control	Complete
By 3/14	FeatureTable and FeatureData summaries	Complete
By 3/21	Diversity analyses	Complete
By 3/28	Milestone 2 Assignment	
By 4/7	Complete Taxonomic analysis	
By 4/14	Differential abundance testing	
By 4/21	Trouble shoot analyses	
By 4/28	Manuscript and presentation prep	
By 5/2	Final Project due	

A few notes: (notes:*)

1. The raw fastq files that support this project can be found within the secure Emory Box location at: https://emory.app.box.com/folder/48209374654 (https://emory.app.box.com/folder/48209374654)
2. I did decide to change my workflow from a qiime2 platform to an R-based platform. This seemed logical since this was what we went over in class and because the advanced visualization with qiime2 requires use of R anyway.
3. I do have a few questions about my results so far, which I will be working to address:

- Determine if merge paired reads are adequate
- Identify the reason for 0 bimeras.

4. I will complete my taxonomic analysis and attempt some basic differential abundance testing before the final due date.

Step 1: Processing fastq files to come up with OTU table.

Load packages

```
library(dada2); packageVersion("dada2")
```

```
## [1] '1.6.0'
```

```
library(ShortRead); packageVersion("ShortRead")
```

```
## [1] '1.36.1'
```

```
library(phyloseq); packageVersion("phyloseq")
```

```
## [1] '1.22.3'
```

```
library(ggplot2); packageVersion("ggplot2")
```

```
## [1] '2.2.1'
```

Change the path in the next chunk to where your files sit.

```
# Set the path to the sequence data files

path <- "~/Desktop/N741/2018Week7/AWHONN Fastq Files"
fileNames <- list.files(path)

# Listing of filenames omitted to save space in rmarkdown
```

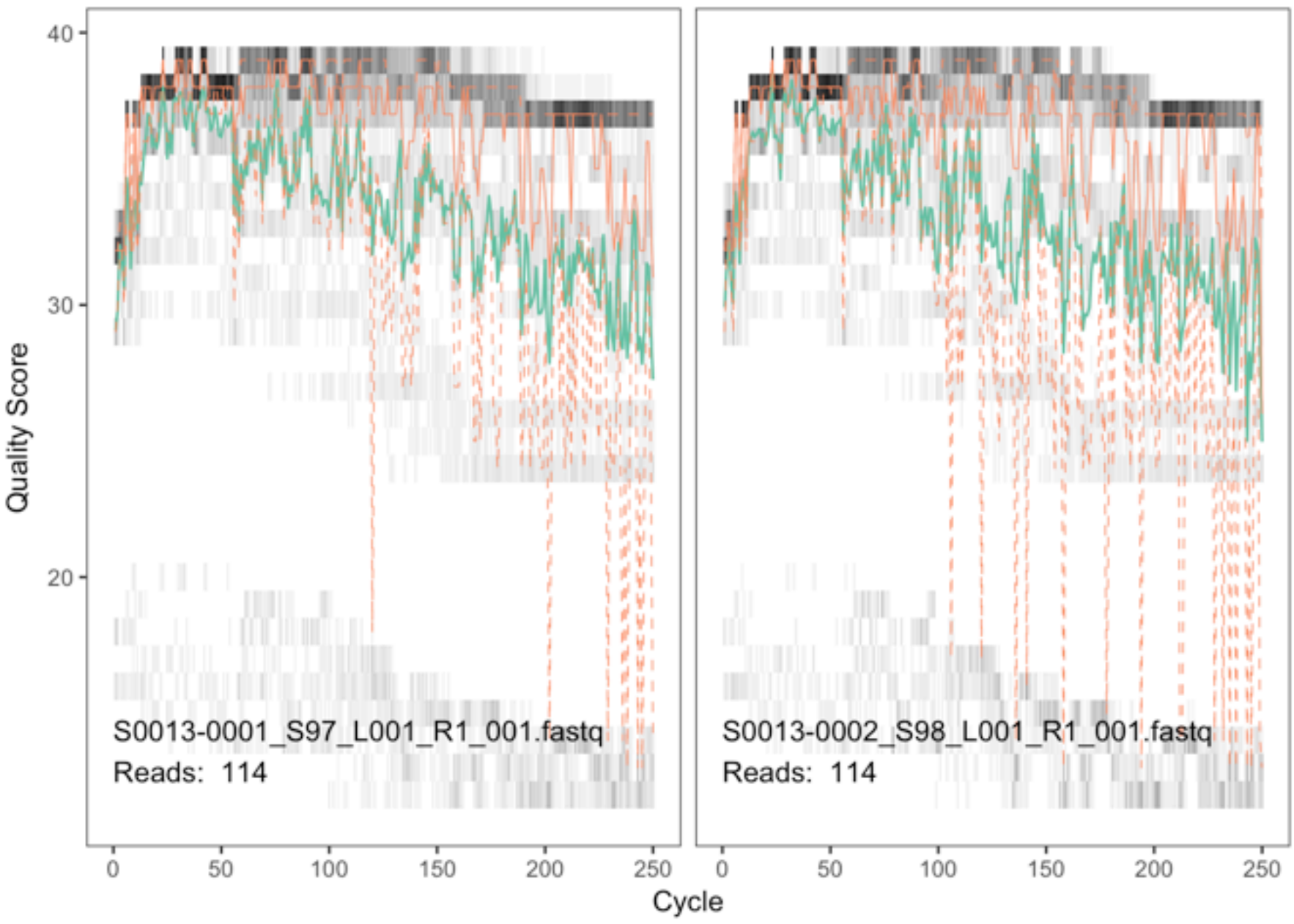
Read in sample names

Using the `dada2` pipeline, first read in the names of the .fastq files. Then manipulate those names as character variables, using regular expressions to create lists of the forward and reverse read .fastq files in *matched* order.

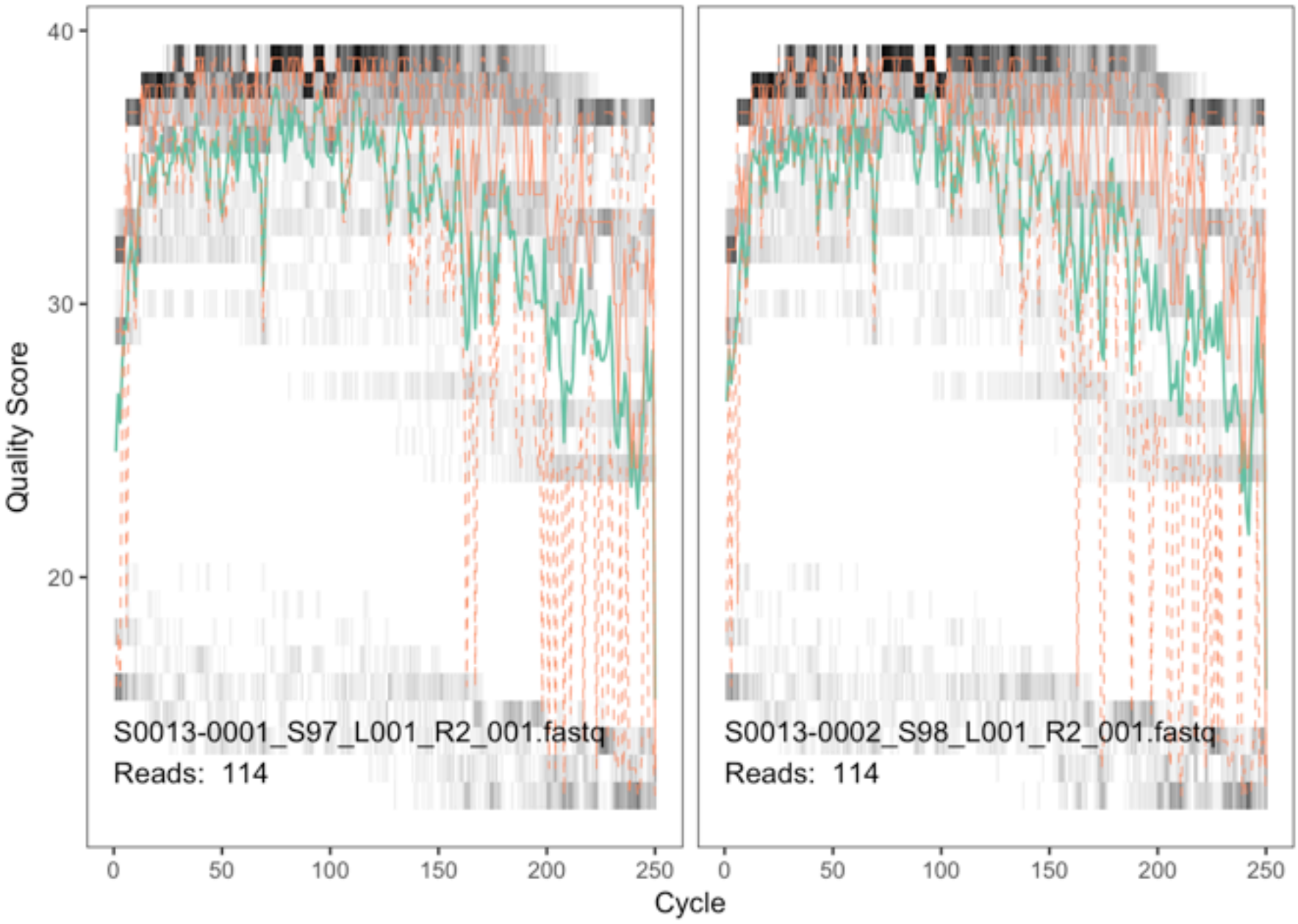
```
# Forward and reverse fastq filenames should have format: SAMPLENAME_R1_001.fastq and
SAMPLENAME_R2_001.fastq

# Start by reading in the names of the .fastq files

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

# Extract sample names, assuming filenames have format: SAMPLENAME_XXX.fastq

sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
```

Generate Quality Profiles of the reads

```
# Visualize the quality profile of the first two files containing forward reads

plotQualityProfile(fnFs[1:2])
```



```
# Visualize the quality profile of the first two files containing reverse reads

plotQualityProfile(fnRs[1:2])
```



Filter and Trim

Typical filtering parameters were used:
- `maxN = 0` – `dada2` requires that there be no N's in a sequence - `truncQ = 2` – truncate reads at the first instance of a quality less than or equal to #. - `maxEE` = 2 – sets the maximum number of expected errors allowed in a read, which is a better filter than simply averaging quality scores.

Note: Decision made to trim conservatively given the robustness of dada2 to lower quality sequences. Trimmed at 200 (forward) and 190 (reverse). Overlap between forward and reverse reads was ensured.

```
# Make a directory and filenames for the filtered fastqs

# Place filtered files in a filtered/ subdirectory

filt.path <- file.path(path, "filtered")
if(!file_test("-d", filt.path)) dir.create(filt.path)
filtFs <- file.path(filt.path, paste0(sample.names, "_F_filt.fastq.gz"))
filtRs <- file.path(filt.path, paste0(sample.names, "_R_file.fastq.gz"))

# Filter the forward and reverse reads

out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, truncLen = c(200, 190),
                     maxN=0, maxEE =c(2,2), truncQ = 2, rm.phix = TRUE,
                     compress=TRUE, multithread=TRUE)

head(out)
```

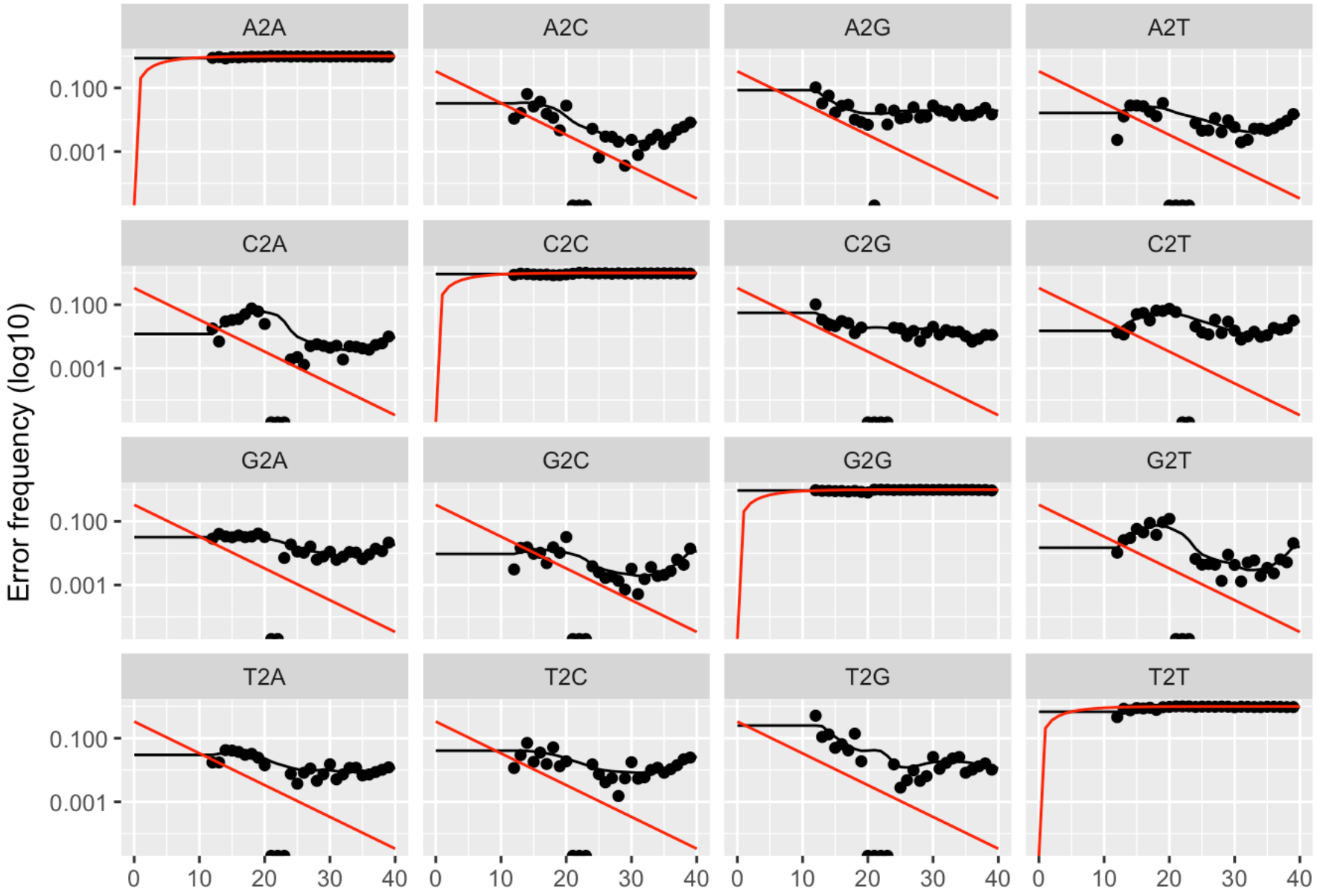
```
##                                    reads.in reads.out
## S0013-0001_S97_L001_R1_001.fastq        114        99
## S0013-0002_S98_L001_R1_001.fastq        114        94
## S0013-0003_S99_L001_R1_001.fastq        114       103
## S0013-0004_S100_L001_R1_001.fastq       114        93
## S0013-0005_S101_L001_R1_001.fastq       114        99
## S0013-0006_S102_L001_R1_001.fastq       114        95
```

Learn the Error Rates

```
errF <- learnErrors(filtFs, multithread = TRUE)
errR <- learnErrors(filtRs, multithread = TRUE)
```

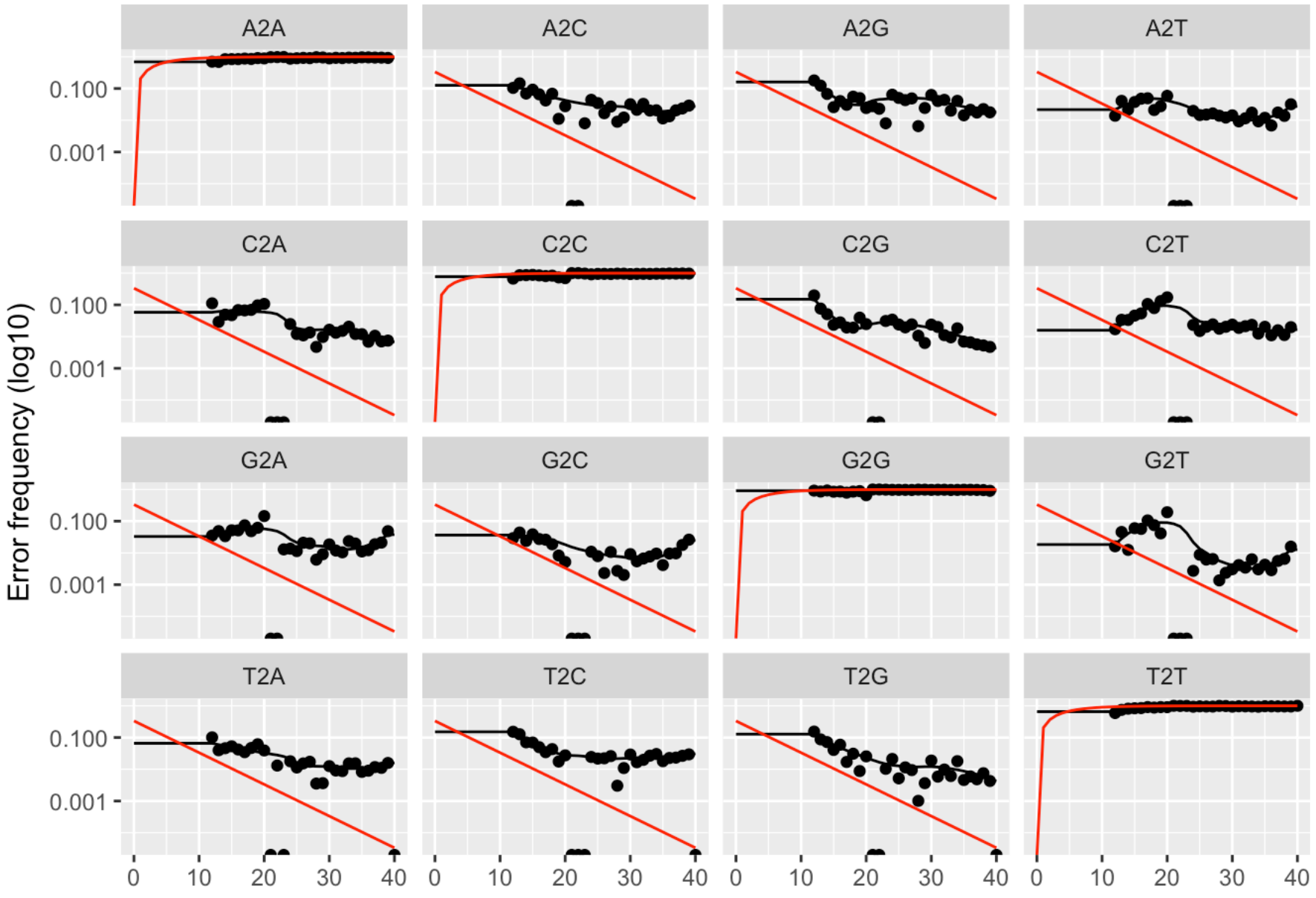
```
# Visualize the estimated error rates by plotting the forward and reverse reads

plotErrors(errF, nominalQ=TRUE)
```

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

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




```
plotErrors(errR, nominalQ = TRUE)
```

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

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



Dereplication

```
# Dereplicate

derepFs <- derepFastq(filtFs, verbose=FALSE)
derepRs <- derepFastq(filtRs, verbose=FALSE)

# Name the derep-class objects by the sample names

names(derepFs) <- sample.names
names(derepRs) <- sample.names
```

Sample Inference

Infer the sequence variants in each sample (second dada pass)

```
# First with the Forward reads

dadaFs <- dada(derepFs, err = errF, multithread = TRUE)

# Then with the Reverse reads

dadaRs <- dada(derepRs, err = errR, multithread = TRUE)
```

```
# Inspect the dada-class objects returned by the dada function

dadaFs[[1]]
```

```
## dada-class: object describing DADA2 denoising results
## 6 sample sequences were inferred from 57 input unique sequences.
## Key parameters: OMEGA_A = 1e-40, BAND_SIZE = 16, USE_QUALS = TRUE
```

```
dadaRs[[1]]
```

```
## dada-class: object describing DADA2 denoising results
## 9 sample sequences were inferred from 61 input unique sequences.
## Key parameters: OMEGA_A = 1e-40, BAND_SIZE = 16, USE_QUALS = TRUE
```

We can see that the algorithm has inferred 6 unique sequence variants from the forward reads and 8 from the reverse reads.

Merge Paired Reads

We can eliminate further spurious sequence variants by merging overlapping reads. The core function is `mergePairs` and it depends on the forward and reverse reads being in matching order at the time they were dereplicated.

```
# Merge the denoised forward and reverse reads

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

```
# Inspect the merged data.frame from the first sample

head(mergers[[1]])
```

```
##
sequence
## 1   TACGTAGGTGGCAAGCGTTGTCCGGAATTATTGGGCGTAAAGCGCGCGCAGGCGGATAGGTCAGTCTGTCTTAAAAGT
TCGGGGCTTAACCCCGTGATGGGATGGAAACTGCCAATCTAGAGTATCGGAGAGGAAAGTGGAATTCCTAGTGTAGCGGTGAAAT
GCGTAGATATTAGGAAGAACACCAGTGGCGAAGGCGACTTTCTGGACGAAAACTGACGCTGAGGCGCGAAAGCCAGGGGAGCGAA
CGGG
## 3   TACGTAGGTCCCGAGCGTTGTCCGGATTTATTGGGCGTAAAGCGAGCGCAGGCGGTTAGATAAGTCTGAAGTTAAAGG
CTGTGGCTTAACCATAGTATGCTTTGGAAACTGTTTAACTTGAGTGCAGAAGGGGAGAGTGGAATTCCATGTGTAGCGGTGAAAT
GCGTAGATATATGGAGGAACACCGGTGGCGAAAGCGGCTCTCTGGTCTGTAACTGACGCTGAGGCTCGAAAGCGTGGGGAGCAAA
CAGG
## 4  TACGTAGGGTGCGAGCGTTAATCGGAATTACTGGGCGTAAAGCGGGCGCAGACGGTTACTTAAGCAGGATGTGAAATCC
CCGGGCTCAACCTGGGAACTGCGTTCTGAACTGGGTAACTAGAGTGTGTCAGAGGGAGGTAGAATTCCACGTGTAGCAGTGAAAT
GCGTAGAGATGTGGAGGAATACCGATGGCGAAGGCAGCCTCCTGGGATAACACTGACGTTCATGCCCGAAAGCGTGGGTAGCAAA
CAGG
## 5 TACGTAGGGTGCGAGCGTTGTCCGGAATTACTGGGCGTAAAGAGCTCGTAGGTGGTTTGTTGCGTCGTCTGTGAAATTCC
GGGGCTTAACTTCGGGGTGGCAGGCGATACGGGCATAACTAGAGTGCTGTAGGGGAGACTGGAATTCCTGGTGTAGCGGTGGAAT
GCGCAGATATCAGGAGGAACACCGATGGCGAAGGCAGGTCTCTGGGCAGTAACTGACGCTGAGGAGCGAAAGCATGGGGAGCGAA
CAGG
## 6   TACGTAGGTGGCAAGCGTTGTCCGGAATTATTGGGCGTAAAGCGCGCGCAGGCGGATCAGTCAGTCTGTCTTAAAAGT
TCGGGGCTTAACCCCGTGATGGGATGGAAACTGCTGATCTAGAGTATCGGAGAGGAAAGTGGAATTCCTAGTGTAGCGGTGAAAT
GCGTAGATATTAGGAAGAACACCAGTGGCGAAGGCGACTTTCTGGACGAAAACTGACGCTGAGGCGCGAAAGCCAGGGGAGCGAA
CGGG
## 7  TACGGAGGGTGCGAGCGTTAATCGGAATAACTGGGCGTAAAGGGCACGCAGGCGGTGACTTAAGTGAGGTGTGAAAGCC
CCGGGCTTAACCTGGGAATTGCATTTCATACTGGGTCGCTAGAGTACTTTAGGGAGGGGTAGAATTCCACGTGTAGCGGTGAAAT
GCGTAGAGATGTGGAGGAATACCGAAGGCGAAGGCAGCCCCTTGGGAATGTACTGACGCTCATGTGCGAAAGCGTGGGGAGCAAA
CAGG
##   abundance forward reverse nmatch nmismatch nindel prefer accept
## 1        23       1       1    138          0      0      1   TRUE
## 3        14       4       6    138          0      0      2   TRUE
## 4        12       2       3    137          0      0      1   TRUE
## 5        11       3       8    136          0      0      1   TRUE
## 6         9       5       9    138          0      0      1   TRUE
## 7         5       6       4    137          0      0      1   TRUE
```

Sequence Table Construction

We will now construct the sequence table, this being analogous to the “OTU table” produced by other methods.

```
# Construct sequence table

seqtab <- makeSequenceTable(mergers)
```

```
## The sequences being tabled vary in length.
```

```
# Consider the table

dim(seqtab)
```

```
## [1]  56 135
```

```
class(seqtab)
```

```
## [1] "matrix"
```

```
# Inspect the distribution of sequence lengths

table(nchar(getSequences(seqtab)))
```

```
##
## 252 253 254
##  39  93   3
```

Remove Chimeras

```
# Remove chimeric sequences

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

```
## Identified 0 bimeras out of 135 input sequences.
```

```
dim(seqtab.nochim)
```

```
## [1]  56 135
```

```
sum(seqtab.nochim)/sum(seqtab)
```

```
## [1] 1
```

Track Reads through the Pipeline

```
getN <- function(x) sum(getUniques(x))
pctSurv <- rowSums(seqtab.nochim)*100/out[,1]
track <- cbind(out, sapply(dadaFs, getN), sapply(mergers, getN), rowSums(seqtab), row
Sums(seqtab.nochim), pctSurv)
colnames(track) <- c("input", "filtered", "denoised", "merged", "tabled", "nonchimeri
c", "% passing")
rownames(track) <- sample.names
head(track)
```

```
##             input filtered denoised merged tabled nonchimeric % passing
## S0013-0001   114       99       99     74     74           74  64.91228
## S0013-0002   114       94       94     78     78           78  68.42105
## S0013-0003   114      103      103     62     62           62  54.38596
## S0013-0004   114       93       93     74     74           74  64.91228
## S0013-0005   114       99       99     99     99           99  86.84211
## S0013-0006   114       95       95     94     94           94  82.45614
```

Assign Taxonomy

GreenGenes 13_8 reference will be used.

```
# Assign taxonomy

# First initialize random number generator for reproducibility

set.seed(100)
getwd()
```

```
## [1] "/Users/ireneyang/Desktop/N741/Final-Project/Milestone-2"
```

```
path
```

```
## [1] "~/Desktop/N741/2018Week7/AWHONN Fastq Files"
```

```
# list.files omitted to save space on rmarkdown

taxa <- assignTaxonomy(seqtab.nochim, "~/Desktop/N741/2018Week7/AWHONN Fastq Files/gg
_13_8_train_set_97.fa", multithread = TRUE)
unname(head(taxa))
```

```
##      [,1]          [,2]               [,3]
## [1,] "k__Bacteria" "p__Firmicutes"    "c__Bacilli"
## [2,] "k__Bacteria" "p__Firmicutes"    "c__Bacilli"
## [3,] "k__Bacteria" "p__Firmicutes"    "c__Clostridia"
## [4,] "k__Bacteria" "p__Actinobacteria" "c__Actinobacteria"
## [5,] "k__Bacteria" "p__Fusobacteria"  "c__Fusobacteriia"
## [6,] "k__Bacteria" "p__Fusobacteria"  "c__Fusobacteriia"
##      [,4]                [,5]                    [,6]
## [1,] "o__Lactobacillales" "f__Streptococcaceae"   "g__Streptococcus"
## [2,] "o__Lactobacillales" "f__Streptococcaceae"   "g__Streptococcus"
## [3,] "o__Clostridiales"   "f__Veillonellaceae"    "g__Veillonella"
## [4,] "o__Actinomycetales" "f__Corynebacteriaceae" "g__Corynebacterium"
## [5,] "o__Fusobacteriales" "f__Fusobacteriaceae"   "g__Fusobacterium"
## [6,] "o__Fusobacteriales" "f__Fusobacteriaceae"   "g__Fusobacterium"
##      [,7]
## [1,] "s__"
## [2,] "s__"
## [3,] "s__dispar"
## [4,] "s__"
## [5,] "s__"
## [6,] "s__"
```

Inspect the taxonomic assignments:

```
taxa.print <- taxa #Removing sequence rownames for display only
rownames (taxa.print) <- NULL
head(taxa.print)
```

```
##      Kingdom       Phylum              Class
## [1,] "k__Bacteria" "p__Firmicutes"     "c__Bacilli"
## [2,] "k__Bacteria" "p__Firmicutes"     "c__Bacilli"
## [3,] "k__Bacteria" "p__Firmicutes"     "c__Clostridia"
## [4,] "k__Bacteria" "p__Actinobacteria" "c__Actinobacteria"
## [5,] "k__Bacteria" "p__Fusobacteria"   "c__Fusobacteriia"
## [6,] "k__Bacteria" "p__Fusobacteria"   "c__Fusobacteriia"
##      Order                Family                   Genus
## [1,] "o__Lactobacillales" "f__Streptococcaceae"    "g__Streptococcus"
## [2,] "o__Lactobacillales" "f__Streptococcaceae"    "g__Streptococcus"
## [3,] "o__Clostridiales"   "f__Veillonellaceae"     "g__Veillonella"
## [4,] "o__Actinomycetales" "f__Corynebacteriaceae" "g__Corynebacterium"
## [5,] "o__Fusobacteriales" "f__Fusobacteriaceae"    "g__Fusobacterium"
## [6,] "o__Fusobacteriales" "f__Fusobacteriaceae"    "g__Fusobacterium"
##      Species
## [1,] "s__"
## [2,] "s__"
## [3,] "s__dispar"
## [4,] "s__"
## [5,] "s__"
## [6,] "s__"
```

Construct a Phylogenetic Tree

```
library(DECIPHER)
```

```
## Loading required package: RSQLite
```

```
seqs <- getSequences(seqtab.nochim)

# This next command will allow propagation of sequence names to the tip labels of the
tree
names(seqs) <- seqs
alignment <- AlignSeqs(DNAStringSet(seqs), anchor=NA)

# Construct tree

library(phangorn)
```

```
## Loading required package: ape
```

```
##
## Attaching package: 'ape'
```

```
## The following object is masked from 'package:ShortRead':
##
##     zoom
```

```
## The following object is masked from 'package:Biostrings':
##
##     complement
```

```
phang.align <- phyDat(as(alignment, "matrix"), type="DNA")
dm <- dist.ml(phang.align)
treeNJ <- NJ(dm) # Tip order will not equal sequence order
fit <- pml(treeNJ, data=phang.align)
```

```
## negative edges length changed to 0!
```

```
## negative edges length changed to 0.

fitGTR <- update(fit, k=4, inv=0.2)
fitGTR <- optim.pml(fitGTR, model="GTR", optInv=TRUE, optGamma=TRUE,
                    rearrangement = "stochastic", control=pml.control(trace=0))
detach("package:phangorn", unload=TRUE)
```

Handoff to `phyloseq`

Our next activity will be to hand off the data to the `phyloseq` package for analysis. This package requires three items: the “OTUtable,” the taxonomy table, and data about the samples. The first two items are directly available at the end of your `dada2` run, and you can import the latter as a .csv file.

```
# Import metadata file.

samdf <- read.csv("~/Desktop/N741/2018Week7/Metadata.csv",header=TRUE)

rownames(samdf) <- samdf$Sample_ID

rownames(samdf)
```

```
##  [1] "S0013-0001" "S0013-0002" "S0013-0003" "S0013-0004" "S0013-0005"
##  [6] "S0013-0006" "S0013-0007" "S0013-0008" "S0013-0009" "S0013-0010"
## [11] "S0013-0011" "S0013-0012" "S0013-0013" "S0013-0014" "S0013-0015"
## [16] "S0013-0016" "S0013-0017" "S0013-0018" "S0013-0019" "S0013-0020"
## [21] "S0013-0021" "S0013-0022" "S0013-0023" "S0013-0024" "S0013-0025"
## [26] "S0013-0026" "S0013-0027" "S0013-0028" "S0013-0029" "S0013-0030"
## [31] "S0013-0031" "S0013-0032" "S0013-0033" "S0013-0034" "S0013-0035"
## [36] "S0013-0036" "S0013-0037" "S0013-0038" "S0013-0039" "S0013-0040"
## [41] "S0013-0041" "S0013-0042" "S0013-0043" "S0013-0044" "S0013-0045"
## [46] "S0013-0046" "S0013-0047" "S0013-0048" "S0013-0049" "S0013-0050"
## [51] "S0013-0051" "S0013-0052" "S0013-0053" "S0013-0054" "S0013-0055"
## [56] "S0013-0056"
```

```
rownames(seqtab.nochim)
```

```
##  [1] "S0013-0001" "S0013-0002" "S0013-0003" "S0013-0004" "S0013-0005"
##  [6] "S0013-0006" "S0013-0007" "S0013-0008" "S0013-0009" "S0013-0010"
## [11] "S0013-0011" "S0013-0012" "S0013-0013" "S0013-0014" "S0013-0015"
## [16] "S0013-0016" "S0013-0017" "S0013-0018" "S0013-0019" "S0013-0020"
## [21] "S0013-0021" "S0013-0022" "S0013-0023" "S0013-0024" "S0013-0025"
## [26] "S0013-0026" "S0013-0027" "S0013-0028" "S0013-0029" "S0013-0030"
## [31] "S0013-0031" "S0013-0032" "S0013-0033" "S0013-0034" "S0013-0035"
## [36] "S0013-0036" "S0013-0037" "S0013-0038" "S0013-0039" "S0013-0040"
## [41] "S0013-0041" "S0013-0042" "S0013-0043" "S0013-0044" "S0013-0045"
## [46] "S0013-0046" "S0013-0047" "S0013-0048" "S0013-0049" "S0013-0050"
## [51] "S0013-0051" "S0013-0052" "S0013-0053" "S0013-0054" "S0013-0055"
## [56] "S0013-0056"
```

Create the phyloseq object.

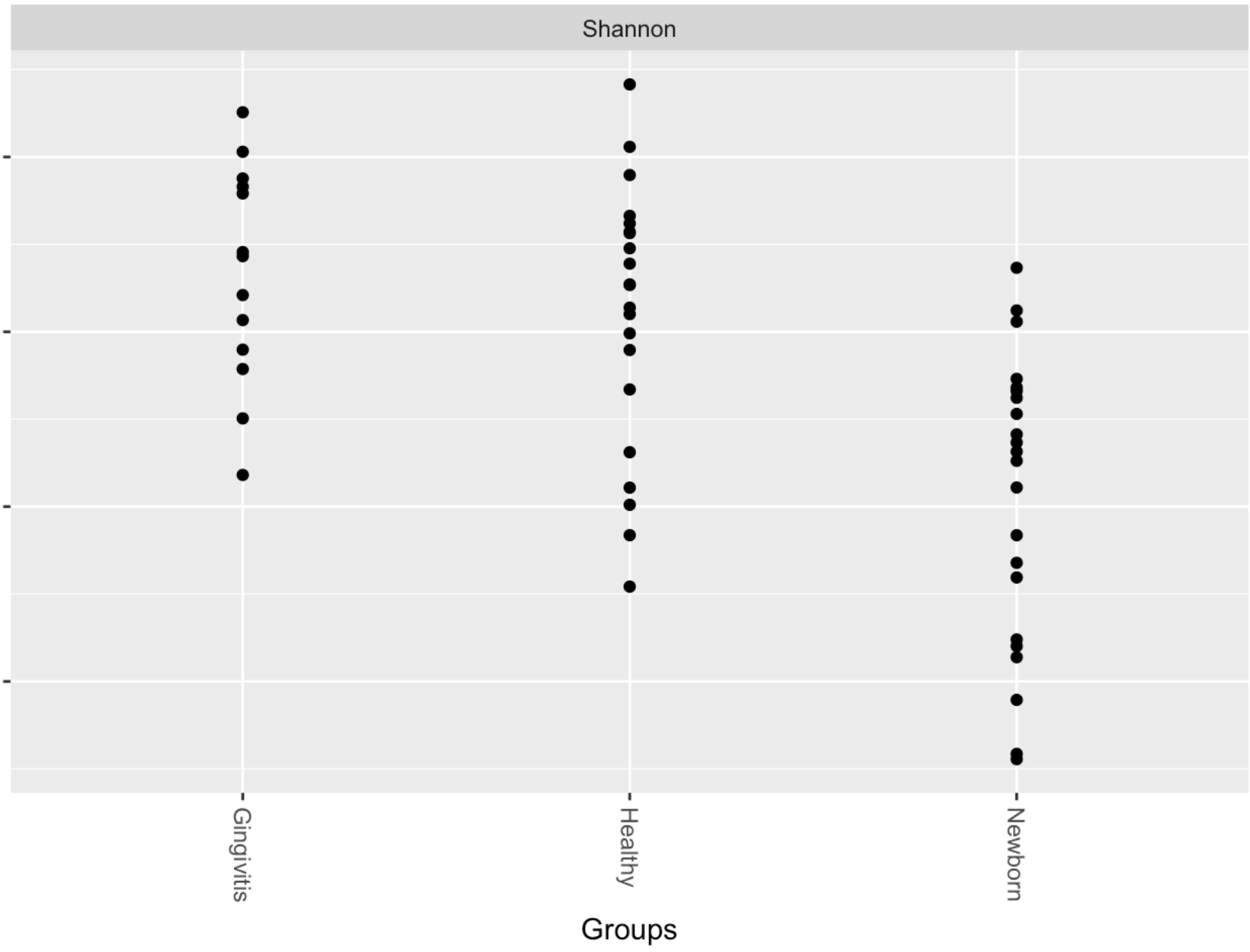
```
library(phyloseq)

# Create phyloseq object

ps <- phyloseq(otu_table(seqtab.nochim, taxa_are_rows=FALSE),
               sample_data(samdf),
               tax_table(taxa),
               phy_tree(fitGTR$tree))

# Describe it

ps
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 135 taxa and 56 samples ]
## sample_data() Sample Data:       [ 56 samples by 6 sample variables ]
## tax_table()   Taxonomy Table:    [ 135 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 135 tips and 133 internal nodes ]
```

Diversity in Microbial Ecology

```
# Plot alpha-diversity

plot_richness(ps, x="Groups", measures = c("Shannon"))
```

```
## Warning in estimate_richness(physeq, split = TRUE, measures = measures): The data
you have provided does not have
## any singletons. This is highly suspicious. Results of richness
## estimates (for example) are probably unreliable, or wrong, if you have already
## trimmed low-abundance taxa from the data.
##
## We recommended that you find the un-trimmed data and retry.
```



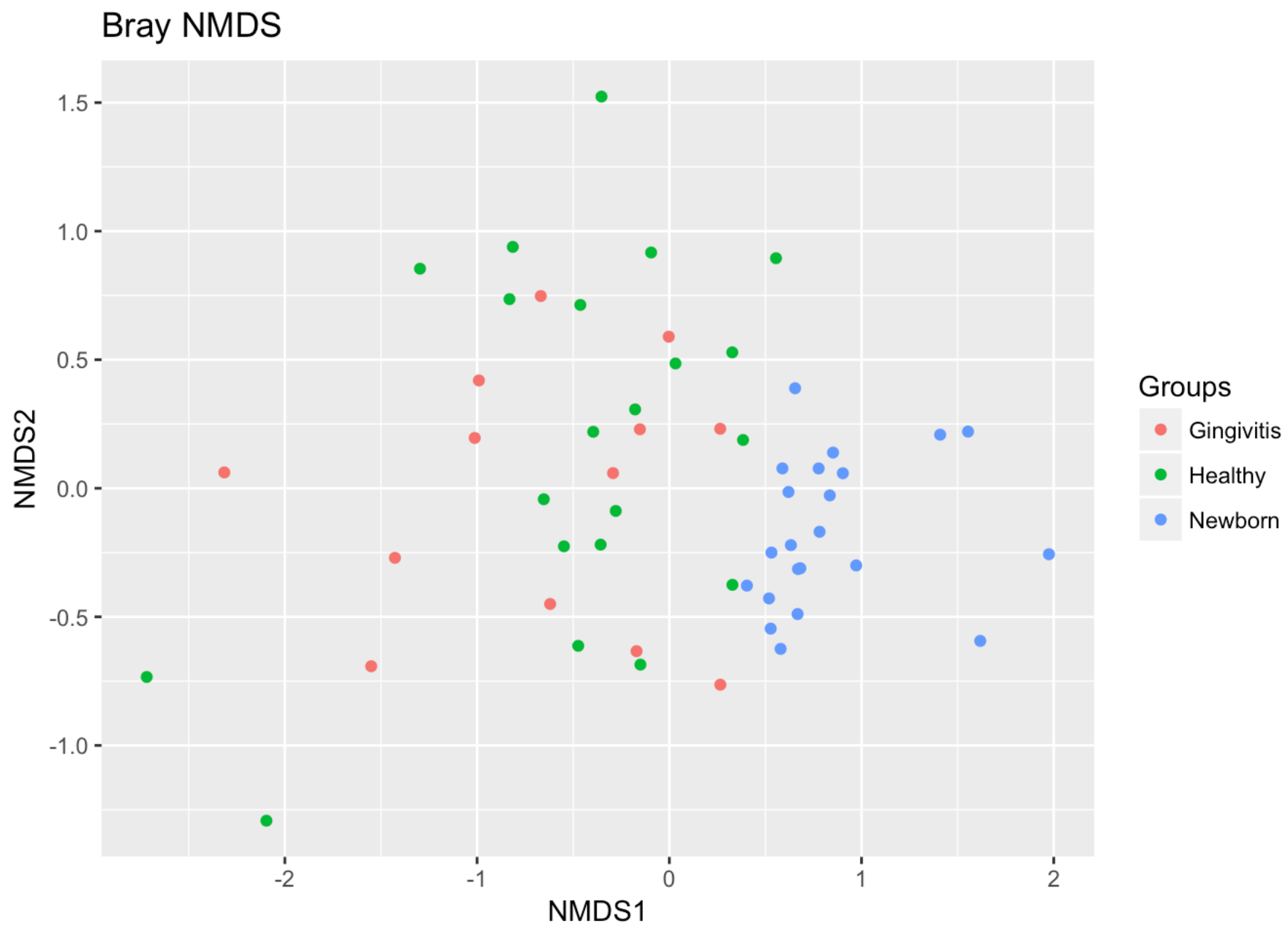
```
        theme_bw()
```

Ordinate

Using the Bray-Curtis dissimilarity index.

```
# Ordinate with Bray-Curtis

ord.nmds.bray <- ordinate(ps, method="NMDS", distance="bray")
```

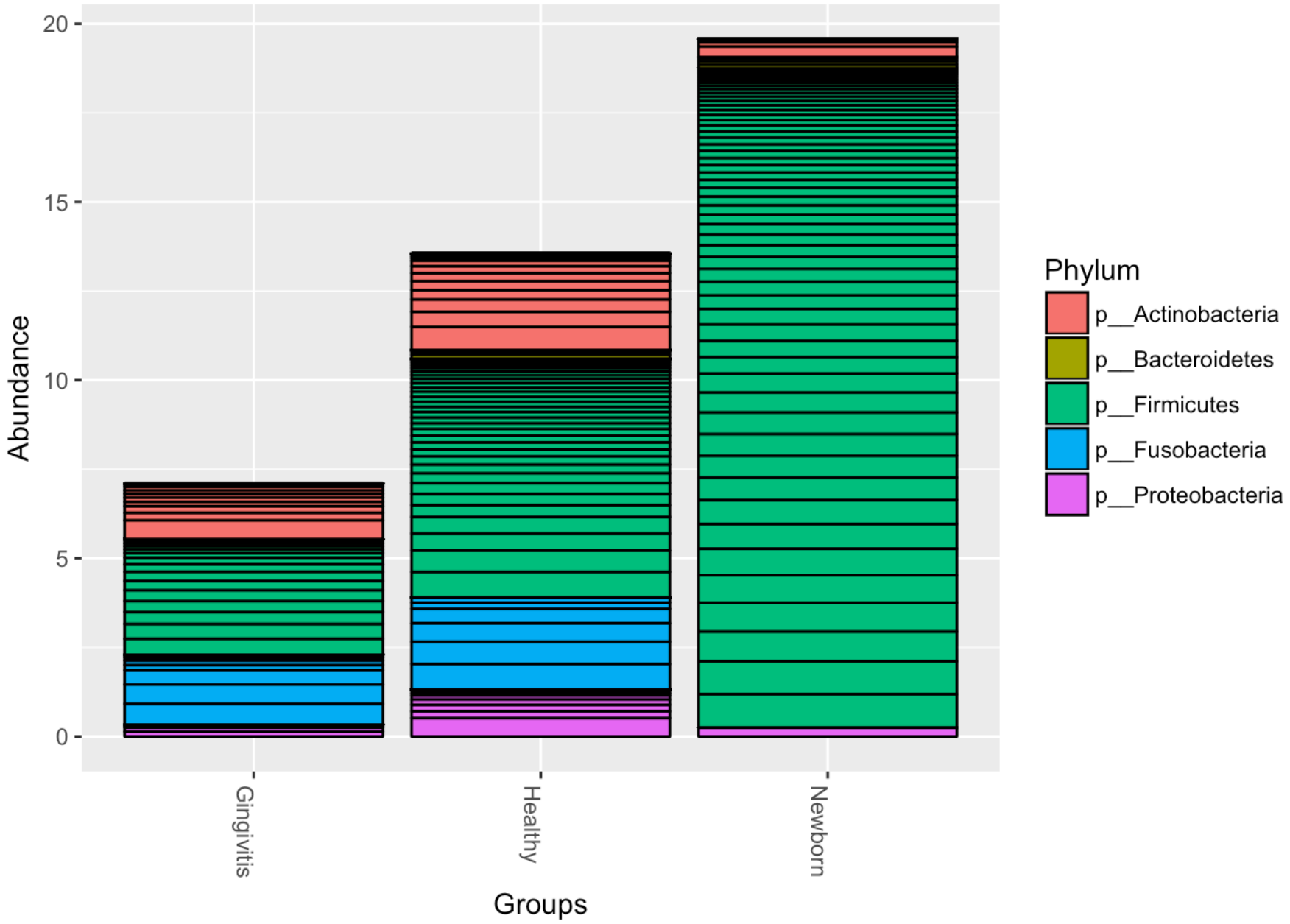
```
plot_ordination(ps, ord.nmds.bray, color="Groups", title="Bray NMDS")
```



We see that ordination picks out a separation between maternal and newborn samples.

Bar Plots

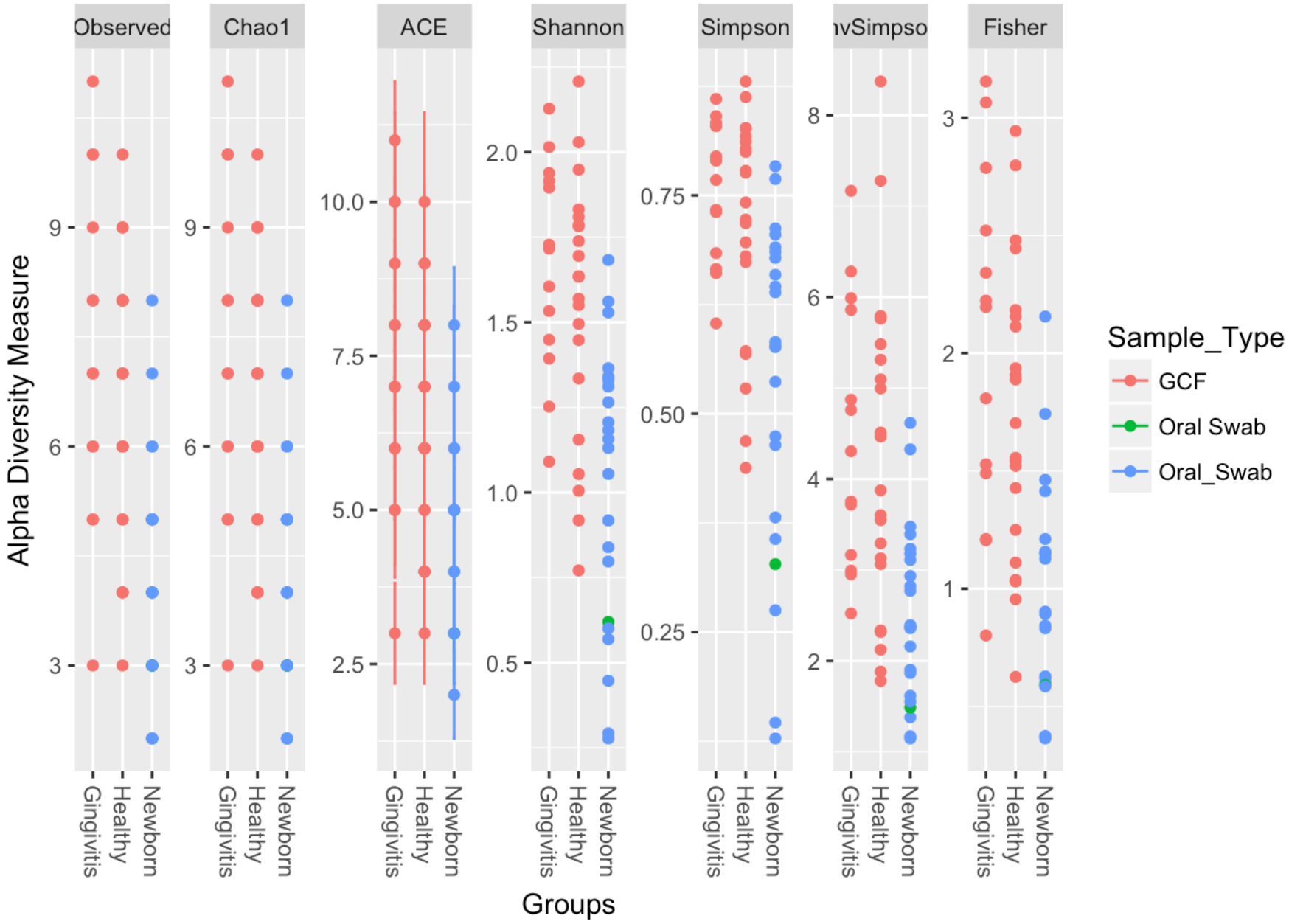
```
# Create bar plots for top 20 OTUs

top20 <- names(sort(taxa_sums(ps), decreasing = TRUE))[1:20]
ps.top20 <- transform_sample_counts(ps, function(OTU) OTU/sum(OTU))
ps.top20 <- prune_taxa(top20, ps.top20)
plot_bar(ps.top20, x="Groups", fill="Phylum")
```




```
# Plot richness

plot_richness(ps, "Groups", "Sample_Type")
```

```
## Warning in estimate_richness(physeq, split = TRUE, measures = measures): The data
you have provided does not have
## any singletons. This is highly suspicious. Results of richness
## estimates (for example) are probably unreliable, or wrong, if you have already
## trimmed low-abundance taxa from the data.
##
## We recommended that you find the un-trimmed data and retry.
```

```
## Warning: Removed 286 rows containing missing values (geom_errorbar).
```



References

Callahan, B. J., Sankaran, K., Fukuyama, J. A., McMurdie, P. J., & Holmes, S. P. (2017). Bioconductor workflow for microbiome data analysis: From raw reads to community analyses. Retrieved from: https://bioconductor.org/help/course-materials/2017/BioC2017/Day1/Workshops/Microbiome/MicrobiomeWorkflowII.html#references (https://bioconductor.org/help/course-materials/2017/BioC2017/Day1/Workshops/Microbiome/MicrobiomeWorkflowII.html#references)