16s rRNA analysis workshop

Hena R. Ramay

2022-10-02

Contents

1 Introduction **5**
1.1 Workshop Schedule . . . 5
1.2 Important links . . . 6

2 Basic Information **7**
2.1 What are we trying to achieve . . . 7
2.2 Basics & Background . . . 7

3 Primer Removal **9**
3.1 CutAdapt . . . 9

4 DADA2 **11**
4.1 DADA2 pipeline (v1.18) . . . 11
4.2 Some tips: . . . 14

5 Downstream Analysis **17**
5.1 Phyloseq . . . 17
5.2 Alpha Diversity . . . 21
5.3 Beta Diversity . . . 22
5.4 Differential Abundance Analysis . . . 26

6 Project **31**
6.1 Setup . . . 31
6.2 Task 1 (Ignore this task) . . . 31
6.3 Task 2 . . . 32
6.4 Task 3 . . . 33
6.5 Project solutions . . . 33

7 What Next? **45**

Chapter 1

Introduction

Our intention is to develop workshop material as we go along. For each day of the workshop, the basic material will be uploaded and more details will be added based on your questions and problems we encounter. So please ask as many questions are you can to help us in making this workshop better!!

1.1 Workshop Schedule

We will try to cover the following material in the course:

- Day1
  - 9:30 - 10:00 am Introductions and a presentation of the basic concepts
  - 10:00 - 10:30 am CutAdapt- how remove primers
  - 10:30 - 10:45 am Break
  - 10:45 - 12 pm Installations of the required packages and data download
  - 12 - 1 pm Lunch break
  - 1 - 3 pm Reading data in and inspecting read quality
- Day2
  - 9:30 -10:30 am Filter and trim + learn error rates + Sample inference + Merge samples
  - 10:30 - 10:45 am Break
  - 10:45 - 12 Generate sequence table and remove Chimeras
  - 12 - 1 pm Lunch break
  - 1 - 3 pm Taxonomy explanation + assign your sequences. If time permits, make a phyloseq object
- Day 3
  - 9:30 - 12 pm onwards we will use a real world dataset to re-do what we have learned here

1.2 Important links

- DADA2 Tutorial : link
- Presentation slides: link
- Workshop dataset : link
- Project dataset : link

Chapter 2

Basic Information

We are very excited to teach this workshop and share what we know with you all !

So lets start by talking about the very basics:

2.1 What are we trying to achieve

Our goal is very similar to gathering data on a city neighbourhood to find out who lives there, how the demographic changes over time or in case of a drastic event. We can gather more information by asking about neighbours, quality of life etc. Similarly when we are looking at microbial communities our first question is who is there, how abundant and how their presence changes over time or when conditions change. We can also ask questions like how the microbiomes are interacting with each other (metabolites).

For the scope of this workshop we will stick to the simple questions: who and how much?

2.2 Basics & Background

Here is the link to the lecture we will start with today: workshop

Key points are:

- Think of a hypothesis before doing an experiment
- Spend time on experiment design.
  - Sample size, 16s region to amplify etc
  - Talk to a bioinformatician
  - Think about the depth of sequencing if you want to capture the less abundant taxa

  - Add negative control to account for contamination
- Thoughtful data analysis is critical for successful identification of microbes

“If you torture the data long enough, it will confess.”- Ronald Coase, Economist

Chapter 3

Primer Removal

There are multiple ways in which you can remove primers sequences from your fastq files. CutAdapt and trimmomatic are two widely used tools for short-read data. DADA2 package has its own removePrimers sequence function which is recommended for PacBio data.

3.1 CutAdapt

The first step that everyone performs before doing an analysis is data cleaning. Data cleaning can mean multiple things in this context: primer removal, quality trimming, removing very short sequences etc. Remember inconsistent and incorrect data leads to false conclusions. In short, garbage in, garbage out applies to all data.

The first step that we will do with amplicon data is check which 16s rRNA gene region was sequenced, find the primer that were used and remove them. For this purpose there are a few software available like, cutAdapt, Trimmomatic, skewer. For the purpose of this workshop we will use cutAdapt.

Cutadapt finds and removes adapter sequences, primers, poly-A tails and other types of unwanted sequence from your high-throughput sequencing reads. It also alows you to perform quality trimming on your reads.

3.1.1 Primer removal for paired end reads

Here, the input reads are in reads.1.fastq and reads.2.fastq, and the result will be written to out.1.fastq and out.2.fastq.

In paired-end mode, the options -a, -b, -g and -u that also exist in single-end mode are applied to the forward reads only. To modify the reverse read, these options have uppercase versions -A, -B, -G and -U that work just like their counterparts. In the example above, ADAPTER_FWD will therefore be trimmed from the forward reads and ADAPTER_REV from the reverse reads.

Single-end/R1 option	Corresponding option for R2
–adapter, -a	-A
–front, -g	-G
–anywhere, -b	-B
–cut, -u	-U
–output, -o	–paired-output, -p

In paired-end mode, Cutadapt checks whether the input files are properly paired. An error is raised if one of the files contains more reads than the other or if the read names in the two files do not match. The read name comparison ignores a trailing /1 or /2 to allow processing some old Illumina paired-end files.

Cutadapt supports compressed input and output files. Whether an input file needs to be decompressed or an output file needs to be compressed is detected automatically by inspecting the file name: For example, if it ends in .gz, then gzip compression is assumed

3.1.2 Quality trimming

-q 15,10

3.1.3 Call cutAdapt from R

Follow this link to see how you can call cutAdapt from within R https://benjjneb.github.io/dada2/ITS_workflow.html

Chapter 4

DADA2

4.1 DADA2 pipeline (v1.18)

From now on, we will be working on the DADA2 package version 1.18. DADA2 has great documentation and an excellent tutorial online.Please go to the following link http://benjjneb.github.io/dada2/tutorial.html

All the notes from now on are my additional comments to help you follow the pipeline:

4.1.1 Data for the tutorial

The data to use for the tutorial can be downloaded from here

4.1.2 Getting ready (load packages and get file list)

Functions that we will be using here are :

- list.files()
- sort()
- strsplit()
- basename()
- sapply()

4.1.3 Inspect read quality profiles

Read in files

```
library(dada2);
```

```
## Loading required package: Rcpp
```

```
library(limma)
path <- "MiSeq_SOP/"
list.files(path)
```

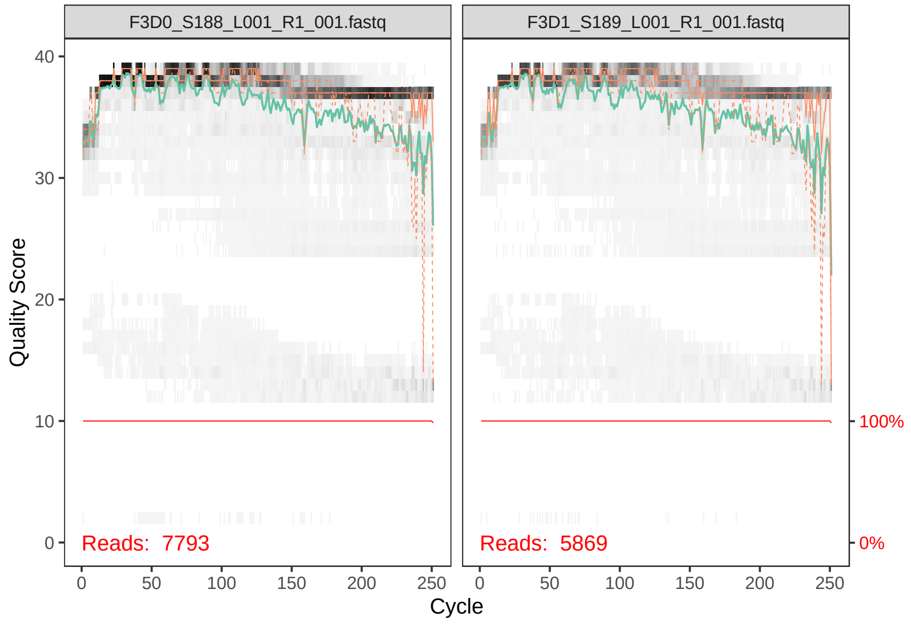
```
##  [1] "F3D0_S188_L001_R1_001.fastq"   "F3D0_S188_L001_R2_001.fastq"
##  [3] "F3D1_S189_L001_R1_001.fastq"   "F3D1_S189_L001_R2_001.fastq"
##  [5] "F3D141_S207_L001_R1_001.fastq" "F3D141_S207_L001_R2_001.fastq"
##  [7] "F3D142_S208_L001_R1_001.fastq" "F3D142_S208_L001_R2_001.fastq"
##  [9] "F3D143_S209_L001_R1_001.fastq" "F3D143_S209_L001_R2_001.fastq"
## [11] "F3D144_S210_L001_R1_001.fastq" "F3D144_S210_L001_R2_001.fastq"
## [13] "F3D145_S211_L001_R1_001.fastq" "F3D145_S211_L001_R2_001.fastq"
## [15] "F3D146_S212_L001_R1_001.fastq" "F3D146_S212_L001_R2_001.fastq"
## [17] "F3D147_S213_L001_R1_001.fastq" "F3D147_S213_L001_R2_001.fastq"
## [19] "F3D148_S214_L001_R1_001.fastq" "F3D148_S214_L001_R2_001.fastq"
## [21] "F3D149_S215_L001_R1_001.fastq" "F3D149_S215_L001_R2_001.fastq"
## [23] "F3D150_S216_L001_R1_001.fastq" "F3D150_S216_L001_R2_001.fastq"
## [25] "F3D2_S190_L001_R1_001.fastq"   "F3D2_S190_L001_R2_001.fastq"
## [27] "F3D3_S191_L001_R1_001.fastq"   "F3D3_S191_L001_R2_001.fastq"
## [29] "F3D5_S193_L001_R1_001.fastq"   "F3D5_S193_L001_R2_001.fastq"
## [31] "F3D6_S194_L001_R1_001.fastq"   "F3D6_S194_L001_R2_001.fastq"
## [33] "F3D7_S195_L001_R1_001.fastq"   "F3D7_S195_L001_R2_001.fastq"
## [35] "F3D8_S196_L001_R1_001.fastq"   "F3D8_S196_L001_R2_001.fastq"
## [37] "F3D9_S197_L001_R1_001.fastq"   "F3D9_S197_L001_R2_001.fastq"
## [39] "HMP_MOCK.v35.fasta"            "Mock_S280_L001_R1_001.fastq"
## [41] "Mock_S280_L001_R2_001.fastq"   "mouse.dpw.metadata"
## [43] "mouse.time.design"             "stability.batch"
## [45] "stability.files"
```

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


tmp<-strsplit(basename(fnFs), "_")


# As tmp is a list of vectors, we will have to use sapply function to get the first po

samples.names<-sapply(tmp,'[',1)
```

4.1.4 Lets make plots to look at the quality for multiple files

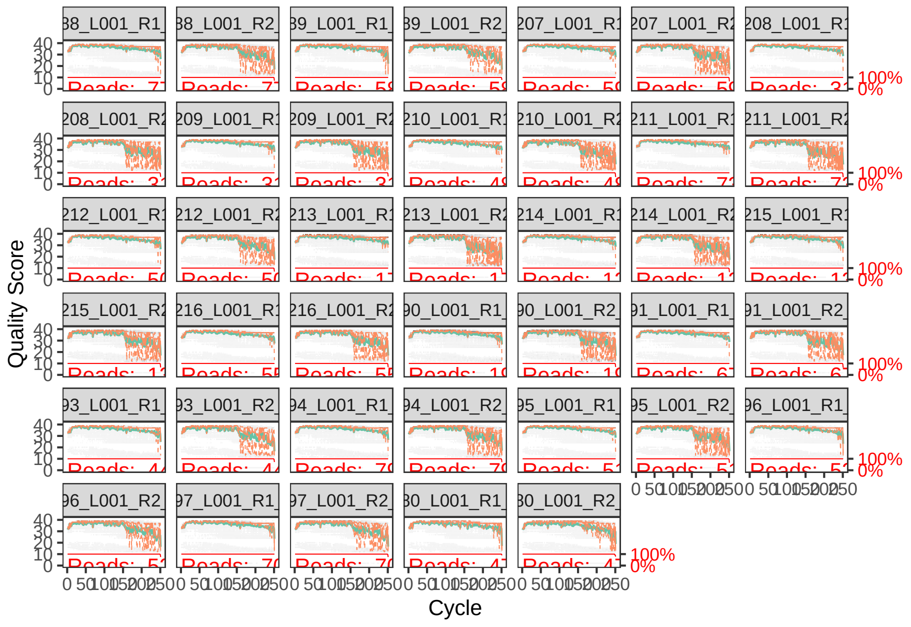
```r
plotQualityProfile(fnFs[1:2])
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



```r
joint_fnames<-c(rbind(fnFs,fnRs))

plotQualityProfile(joint_fnames)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



if there are too many files, use arrangegrob. See here here for details

For the rest of the DADA2 pipelien please visit

4.2 Some tips:

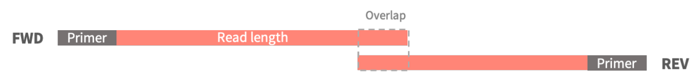
1. At every step check how many reads are getting filtered. This is very important to make sure that you are not losing too much data.

2. If in the filtering step you are losing too many reads, relax the maxEE from 2 to 3 or more. Also check the length that you are truncating at.

3. If you are using cutAdapt for quality trimming, you don't have to use truncation in filterandtrim function of DADA2

4. Always keep in mind the overlap length for your region of interest. see the figure below for calculation:

How to calculate V region overlap



Overlap=Read length *2 – primer_F- primer_R – VR_length

Example: V4 VR_length=254 bp, read length =250 x2 bp and primers=20 bp

Overlap= 250 *2 – 20- 20 – 254
= 206

5. If you lose too many sequences to chimera removal, please check that the primers are removed properly.

Chapter 5

Downstream Analysis

5.1 Phyloseq

Once we are finished using the dada2 package, we will have a sequence table and taxonomy table.

Lets look at the metadata files we have to get more information about these samples.

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
time_info<-read.delim("~/projects/16sanalysis_workshop/workshoptutorial/MiSeq_SOP/mouse.time.desi
dpw_info<-read.delim("~/projects/16sanalysis_workshop/workshoptutorial/MiSeq_SOP/mouse.dpw.metada

sample_info<-left_join(time_info,dpw_info,by="group")

rownames(sample_info)<- sample_info$group

sample_info
```

```
##         group  time dpw
## F3D0     F3D0 Early   0
## F3D1     F3D1 Early   1
## F3D141 F3D141  Late 141
## F3D142 F3D142  Late 142
## F3D143 F3D143  Late 143
## F3D144 F3D144  Late 144
## F3D145 F3D145  Late 145
## F3D146 F3D146  Late 146
## F3D147 F3D147  Late 147
## F3D148 F3D148  Late 148
## F3D149 F3D149  Late 149
## F3D150 F3D150  Late 150
## F3D2     F3D2 Early   2
## F3D3     F3D3 Early   3
## F3D5     F3D5 Early   5
## F3D6     F3D6 Early   6
## F3D7     F3D7 Early   7
## F3D8     F3D8 Early   8
## F3D9     F3D9 Early   9
```

Lets make a phyloseq object

Now that we have sample_info lets try to make a phyloseq object out of this

```r
library(phyloseq)
taxa<-readRDS("~/projects/16sanalysis_workshop/workshoptutorial/output/taxa.rds")
seq<-readRDS("~/projects/16sanalysis_workshop/workshoptutorial/output/seq.rds")

ps <- phyloseq(otu_table(seq,taxa_are_rows = F),
               sample_data(sample_info),
               tax_table(taxa))

ps
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 234 taxa and 19 samples ]
## sample_data() Sample Data:       [ 19 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 234 taxa by 7 taxonomic ranks ]
```

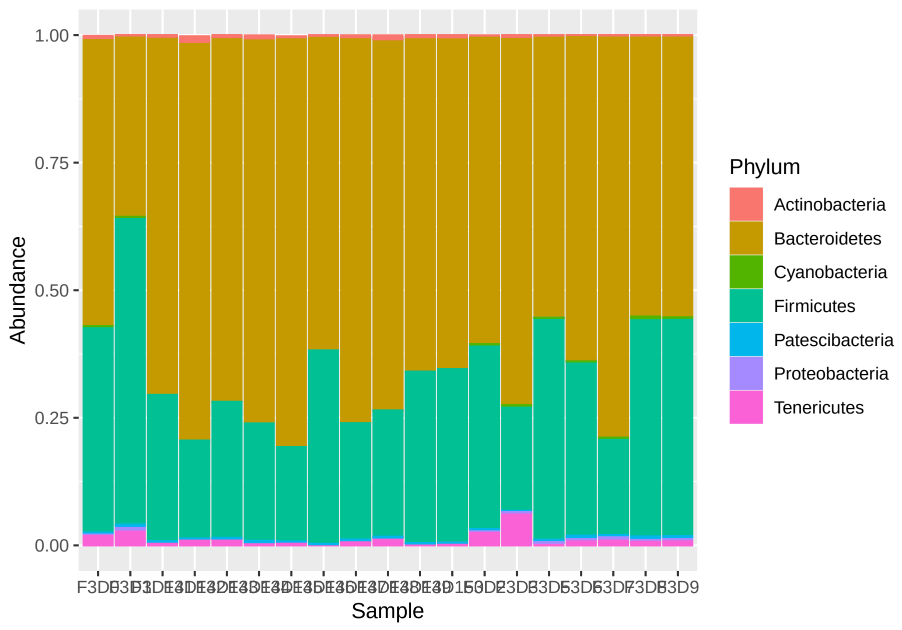
You can see that this object has an OTU table(ASV table), sample data and tax_table. You can use functions tax_table(), sample_data() and otu_table() to access the data.

Take a look at:

- subset_samples()
- subset_taxa()
- tax_glom()
- sample_sums()

- prune_samples()
- transform_sample_counts()
- psmelt()

```
ps2<-tax_glom(ps,taxrank = "Phylum")
ps2 = transform_sample_counts(ps2, function(x) x/sum(x))
pmelt<-psmelt(ps2) %>% arrange(desc(Abundance))

cutoff<-0.005
pmelt_filt<-pmelt %>% group_by(Phylum,time) %>% filter(sum(Abundance) >cutoff)

ggplot(pmelt_filt,aes(x=Sample,y=Abundance,color=Phylum,fill=Phylum)) +geom_bar(stat = "identity'
```



```
pmelt_filt %>% group_by(time,Phylum) %>% summarise(mean_Abundance=mean(Abundance)) %>% ggplot(.,a
```

```
## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.
```

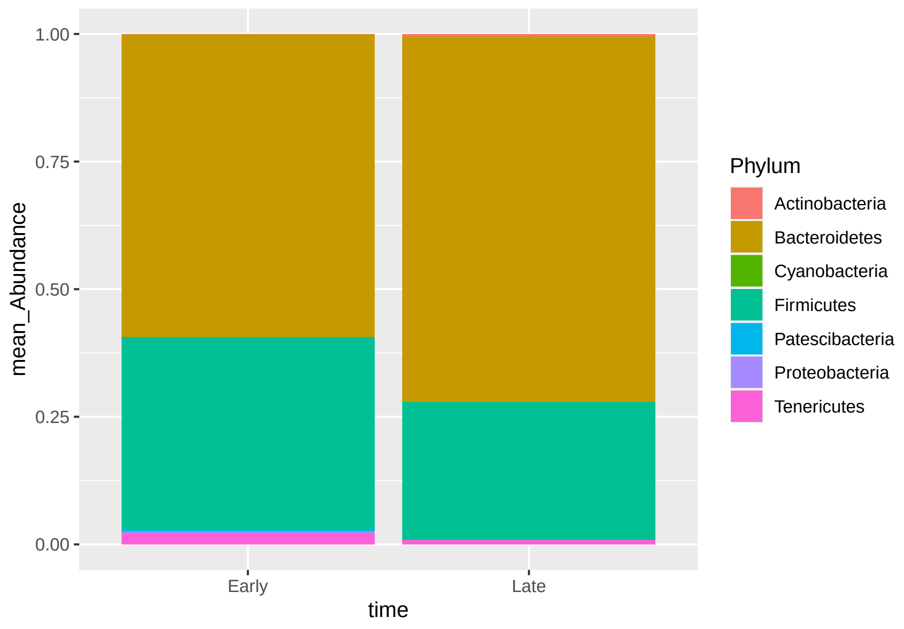


Figure out how to change the colors to custom colors!!

```
pmelt_filt %>% group_by(time,Phylum) %>% summarise(mean=mean(Abundance))
```

```
## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 12 x 3
## # Groups:   time [2]
##    time  Phylum              mean
##    <chr> <chr>              <dbl>
##  1 Early Actinobacteria  0.00133
##  2 Early Bacteroidetes   0.592
##  3 Early Cyanobacteria   0.000594
##  4 Early Firmicutes      0.378
##  5 Early Patescibacteria 0.00285
##  6 Early Proteobacteria  0.00331
##  7 Early Tenericutes     0.0218
##  8 Late  Actinobacteria  0.00472
##  9 Late  Bacteroidetes   0.716
## 10 Late  Firmicutes      0.269
## 11 Late  Patescibacteria 0.000993
## 12 Late  Tenericutes     0.00845
```

5.2 Alpha Diversity

Alpha Diversity: Variety and abundance of organisms within a sample

Species richness: The number of different kinds of organisms present in a particular community

Species evenness Compares the uniformity of the population size of each of the species present

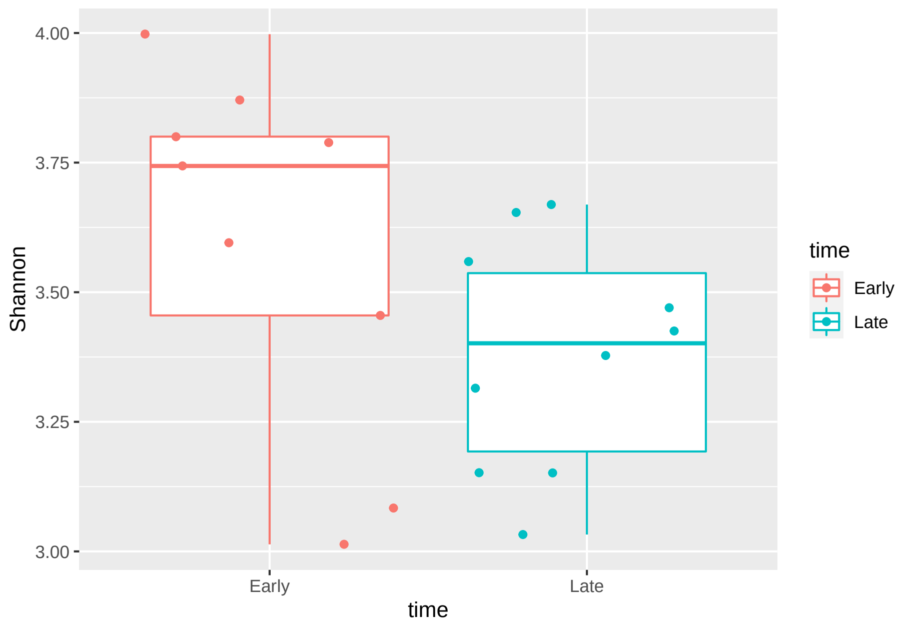
There are different alpha diversity measures to calculate the biodiversity including Simpson, Shannon, and Observed. These indexes show different estimates of magnitude of change in the niche giving us a more complete picture. There are differences in these indexes in weighting and unit:

Simpson: Estimator of species richness and species evenness, but provides more weight to evenness resulted from the sum of squared proportions.

Shannon: Estimator of species richness and species evenness, but provides more weight to richness than evenness and is shown as a logarithmic value.

Observed: Abundance-based estimator of species richness. This index calculates the minimal number of OTUs (rare OTUs) present in a sample. This index gives more weight to the low abundant species.

```
richness<-estimate_richness(ps)
richness$time<-ps@sam_data$time

ggplot(richness,aes(x=time,y=Shannon,color=time)) + geom_boxplot() +geom_jitter()
```



```
wilcoxon_shannon <- wilcox.test(Shannon ~ time, data = richness)

wilcoxon_shannon
```

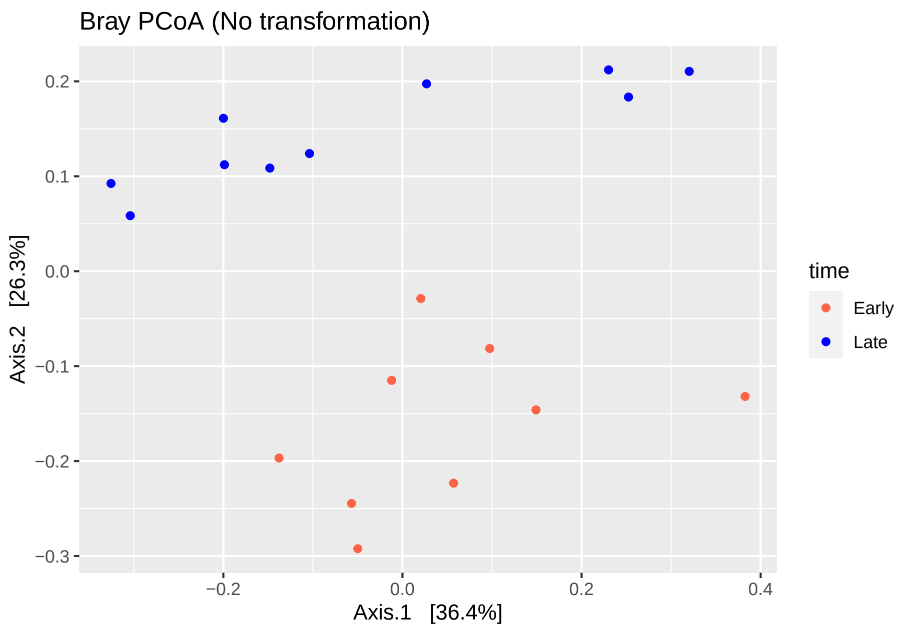
```
##
##  Wilcoxon rank sum exact test
##
## data:  Shannon by time
## W = 65, p-value = 0.1128
## alternative hypothesis: true location shift is not equal to 0
```

5.3 Beta Diversity

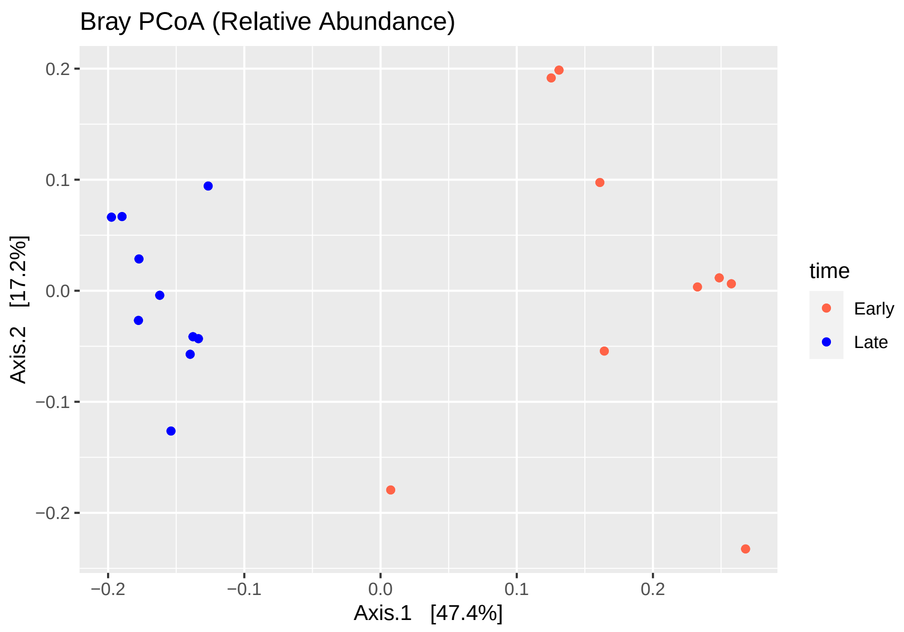
Beta diversity is a measure of dissimilarity in microbial composition **between** two samples.

Distance metrics: Bray–Curtis method does not look at the phylogenetic relatedness and takes the taxa abundance into consideration. This distance is defined as the difference of the abundance divided by the total abundance contributed by both samples. Unifrac method calculates the distances between samples based on their phylogenetic relatedness. There are two unifrac measures: unweighted UniFrac, a qualitative measure that takes presence/absence of data into consideration to compare community composition. This suggests if ecological factors prevent a taxonomic group from occupying certain habitats.Weighted UniFrac, a quantitative measure weights the branches of a phylogenetic tree based on the the the relative abundance of species/taxa. This suggests if ecological differences between habitats have caused the abundance of taxonomic groups to change.

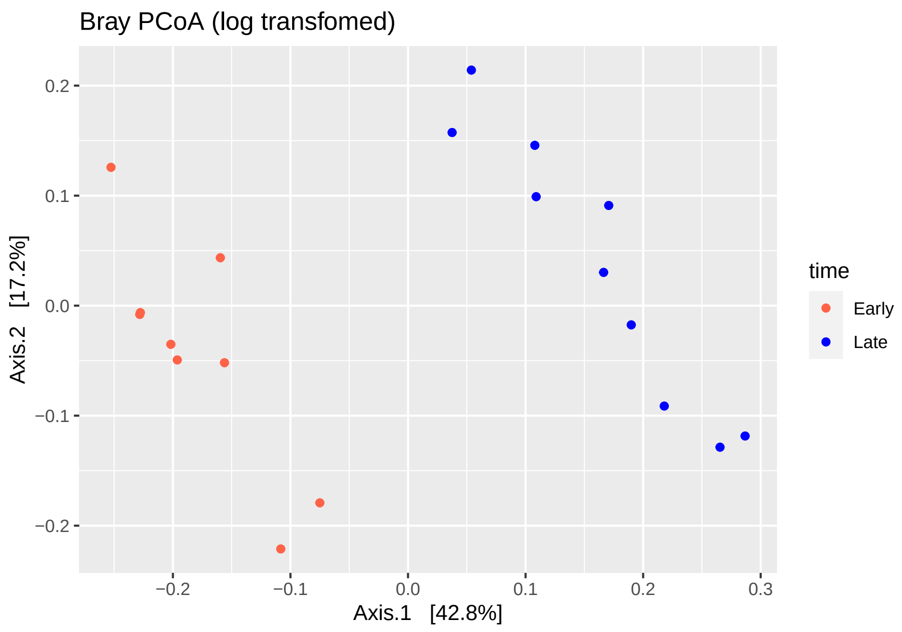
```
# Calculate Bray-curtis dissimilarity
ord<-ordinate(ps,method = "PCoA",distance = "bray")

# plot ordination
plot_ordination(ps,ordination = ord,color="time",title="Bray PCoA (No transformation)")+ scale_co
```



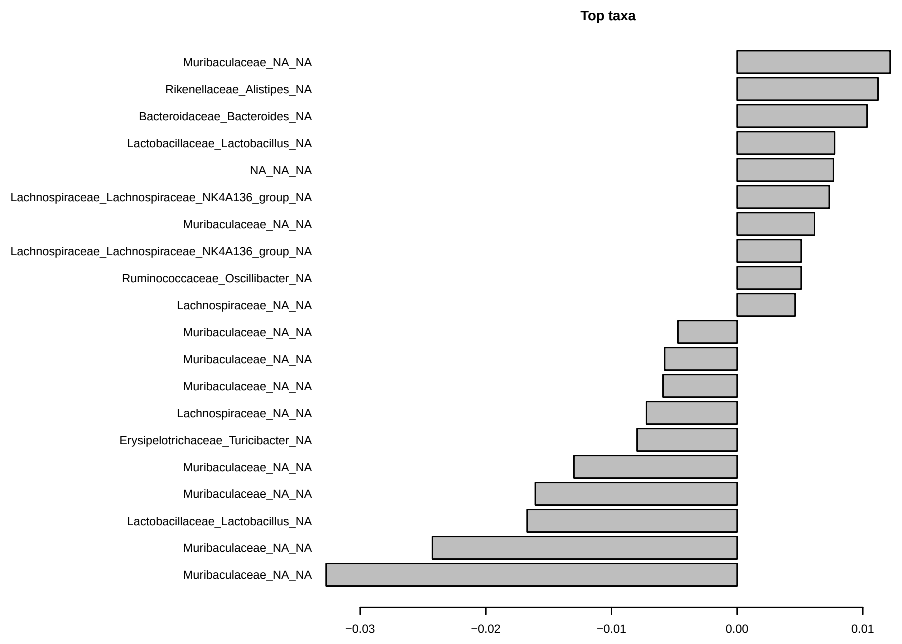
```
# plot ordination with relative abundance values
ps %>%
transform_sample_counts(., function(x) x/sum(x)) %>%
plot_ordination(., ordinate(., method="PCoA", distance="bray"),
color="time" ,title="Bray PCoA (Relative Abundance)")+ scale_colour_manual(values=(c("tomato","bl
```



```
# plot ordination with log transfromed values
ps %>%
transform_sample_counts(., function(x) log(1 + x)) %>%
plot_ordination(., ordinate(., method="PCoA", distance="bray"),
color="time" ,title="Bray PCoA (log transfomed)")+ scale_colour_manual(values=(c("toma
```



```r
# Permanova ()
ps_rel<- ps %>% transform_sample_counts(., function(x) x/sum(x))
asv <- as.matrix(ps_rel@otu_table@.Data)
colnames(asv)<-paste0(ps_rel@tax_table[,5],"_",ps_rel@tax_table[,6],"_",ps_rel@tax_table[,7])

meta <- data.frame(ps_rel@sam_data)

permanova <- vegan::adonis(asv ~ time,
               data = meta, permutations=999, method = "bray")

coef <- coefficients(permanova)["time1",]
top.coef <- coef[rev(order(abs(coef)))[1:20]]
par(mar = c(2, 24, 2, 1),cex=0.5)
barplot(sort(top.coef), horiz = T, las = 1, main = "Top taxa")
```



5.4 Differential Abundance Analysis

Differential abundance analysis is performed between two condition or time points to find taxa that have either bloomed or depleted from one condition compared to another. There is a variety of tests that can be used to perform this analysis like **DESeq2**, **ANCOM-BC**, **Corncob** to name a few.

5.4.1 DESEQ2

For demonstration purpose, we will use DESeq2 here.It internally performs data normalization and does p-value correction. It used a negative binomial distribution to detect differences in read counts between groups.To fully understand the method you can watch this detailed video by Statquest here

```
library(DESeq2)
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:dplyr':
##
```

```
##     combine, intersect, setdiff, union

## The following object is masked from 'package:limma':
##
##     plotMA

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:phyloseq':
##
##     distance

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## Loading required package: GenomicRanges

## Loading required package: GenomeInfoDb

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats
```

```
##
## Attaching package: 'matrixStats'

## The following object is masked from 'package:dplyr':
##
##     count

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'Biobase'

## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians

## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians

## The following object is masked from 'package:phyloseq':
##
##     sampleNames
```

```
dds<-phyloseq_to_deseq2(ps,design = ~time)
```

```
## converting counts to integer mode

## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
dds <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 22 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

```
res <- results(dds)
df <- as.data.frame(res)
# Orders the rows of data frame in increasing order firstly based on column
# "log2FoldChange" and secondly based on "padj" column
df <- df %>% arrange(log2FoldChange, padj)
```

Chapter 6

Project

6.1 Setup

Please download the raw data here.

These samples are from a study looking at the effect of a prebiotic on the gut microbiota of children with type 1 diabetes. To invesigate this, the researchers took samples at three timepoints (baseline, 3 months, and 6 months) and sequenced the **V3-V4** region of the 16S rRNA gene.

6.2 Task 1 (Ignore this task)

The first step with all sequence data is to ensure the primers have been removed. The two most common tools to accomplish this are **Cutadapt** and **Trimmomatic**.

Note: both of these tools can be used for quality trimming as well as primer removal but here we will perform the quality trimming in dada2 instead.

6.2.1 Cutadapt

You can find the documentation for this tool here.

These are the parameters that contain the primer sequences:

- `-g : forward primer`
- `-G : reverse primer`
- `-a : reverse primer, reverse complement`
- `-A : forward primer, reverse complement`

For this dataset, these are the appropriate primers:

```
-g TCGTCGGCAGCGTCAGATGTGTATAAGAGACAGCCTACGGGNGGCWGCAG
-G GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAGGACTACHVGGGTATCTAATCC
-a GGATTAGATACCCBDGTAGTCCTGTCTCTTATACACATCTCCGAGCCCACGAGAC
-A CTGCWGCCNCCCGTAGGCTGTCTCTTATACACATCTGACGCTGCCGACGA
```

Here is an example of how to execute Cutadapt from the command line:

```
cutadapt -g <sequence> -G <sequence> -a <sequence> -A <sequence> -o output_forward.fas
```

Note that you can also run cutadapt from within R. You can find instructions on how to do that here.

6.2.2 Trimmomatic

You can find the documentation for this tool here.

Instead of specifying forward and reverse reads, Trimmomatic takes a fasta file as input and searches the reads for all sequences in the files. When using Trimmomatic on paired-end reads, you will get four output files: forward and reverse reads that are still paired (reads for which both forward and reverse were kept) and forward and reverse reads that are unpaired. In general, the number of unpaired reads should be small and you can continue with your analysis without these.

For this dataset, you can use this fasta.

Here is an example of how to execute Trimmomatic from the command line:

```
trimmomatic PE input_forward.fq.gz input_reverse.fq.gz output_forward_paired.fq.gz out
```

6.2.3 Primers removed

If you don't care about using either of these tools and want to move on to the next step, you can find the sequences with the primers removed here.

6.3 Task 2

Use the dada2 pipeline to generate a table of ASVs and a taxonomy table.

1. Plot quality profiles of the reads for all samples. Save these as pdf files.
2. Try a couple of sets of filtering params and determine the best choice.
3. Filter and trim the data and generate the quality plots again. Save these as well.
4. Learn the error rates and visually inspect the estimated error rates.
5. Infer samples.
6. Merge paired reads.
7. Construct sequence table.
8. Remove chimeras.

9. Track reads through the pipeline and ensure that no step of the analysis removed a large/unreasonable proportion of the reads.
10. Assign taxonomy using the Silva databse.
11. Save the ASV table and taxonomy table using the `saveRDS()` function.

6.4 Task 3

Use the phyloseq package to generate do some basic analysis. For this section you will need the project metadata files that you already downloaded

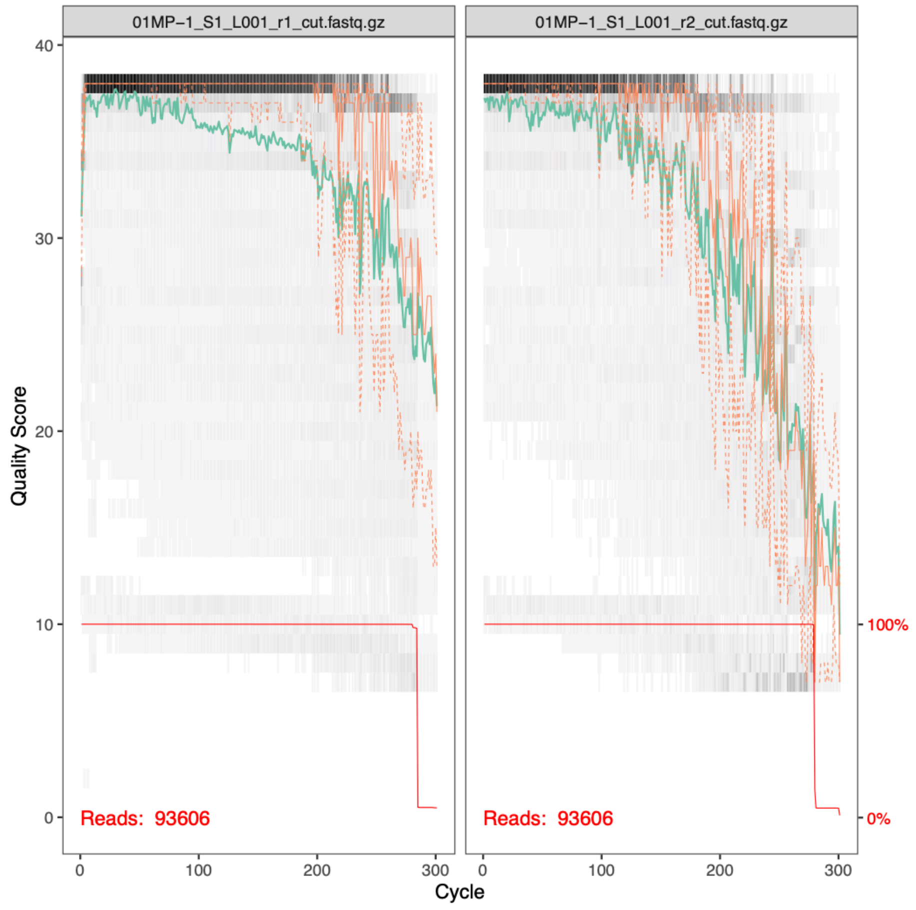
1. Generate a histogram of the number of reads per sample. Does this look okay?
2. Generate a table of alpha diversity measures (Shannon, Simspon, and Chao1) and use this table to plot alpha diversity by treatment and time-point.
3. Transform the data to relative abundances and remove ASVs that are only present in less than 5% of the samples.
4. Perform a principal component analysis (PCoA) using the `ordinate()` function and `distance = "bray"`. Plot beta diversity.
5. Explore the composition of each sample by generating bar plots of the relative abundances at the level of phylum, family, and genus. For family and genus, plot only the top 10 most abundant taxa.

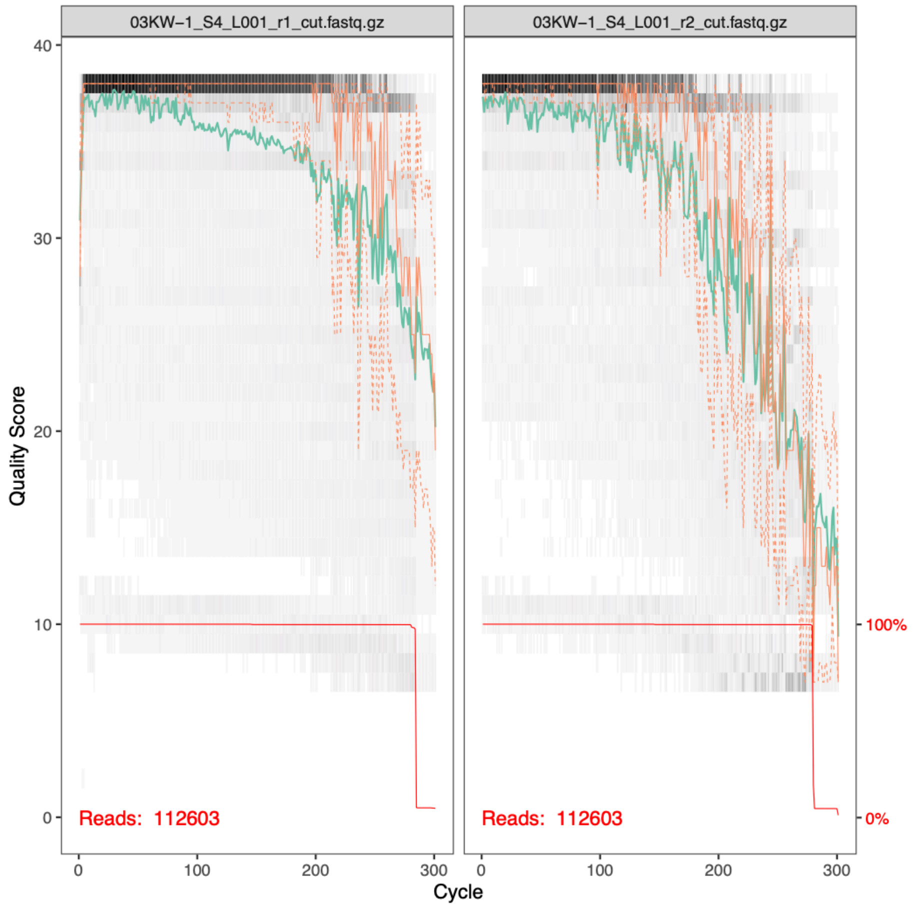
6.5 Project solutions

6.5.1 Task 2 - dada2

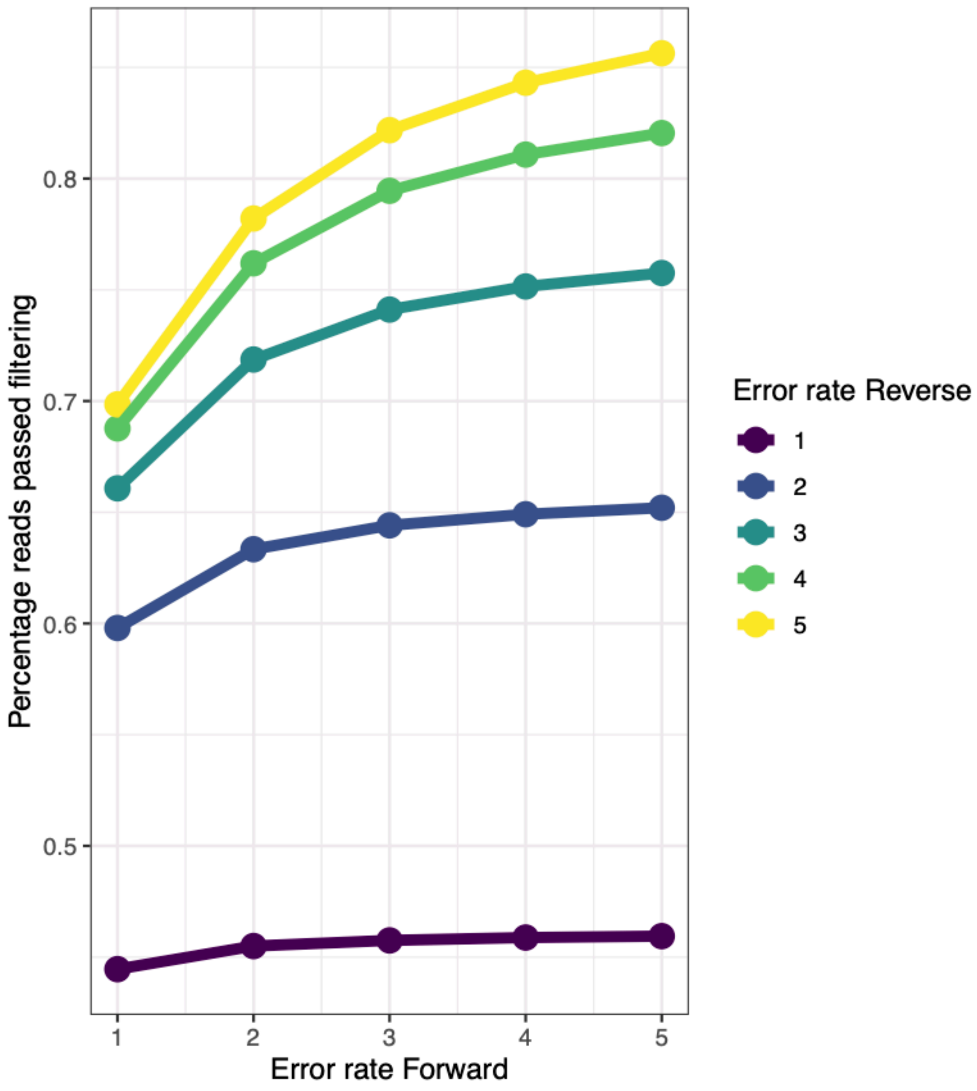
For this section, the solution closely resembles the code from the dada2 tutorial. The main challenge is to determine the appropriate filtering parameters.

Here are the quality profiles for the forward and reverse reads for samples 01MP-1 and 03KW-1. You can see that the quality of the ends of the reverse reads are much worse than the forward reads.


01MP−1_S1_L001_r1_cut.fastq.gz
01MP−1_S1_L001_r2_cut.fastq.gz
Quality Score
Cycle
Reads: 93606
Reads: 93606
100%
0%
0
10
20
30
40
0
100
200
300
0
100
200
300

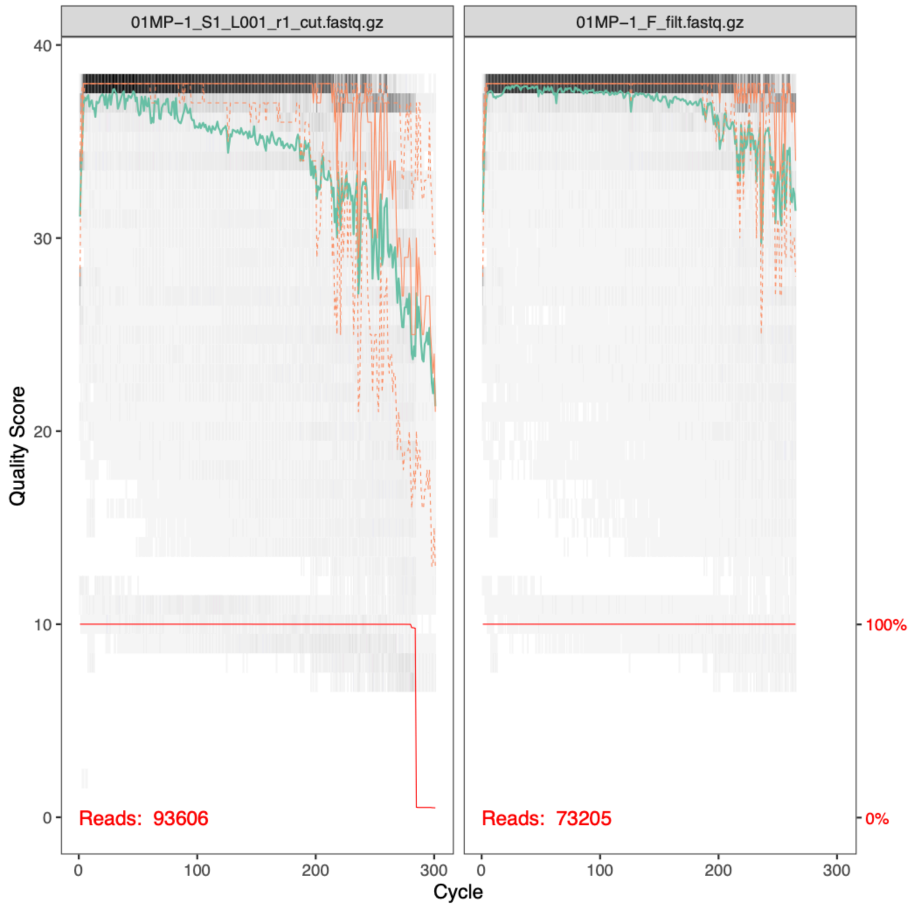


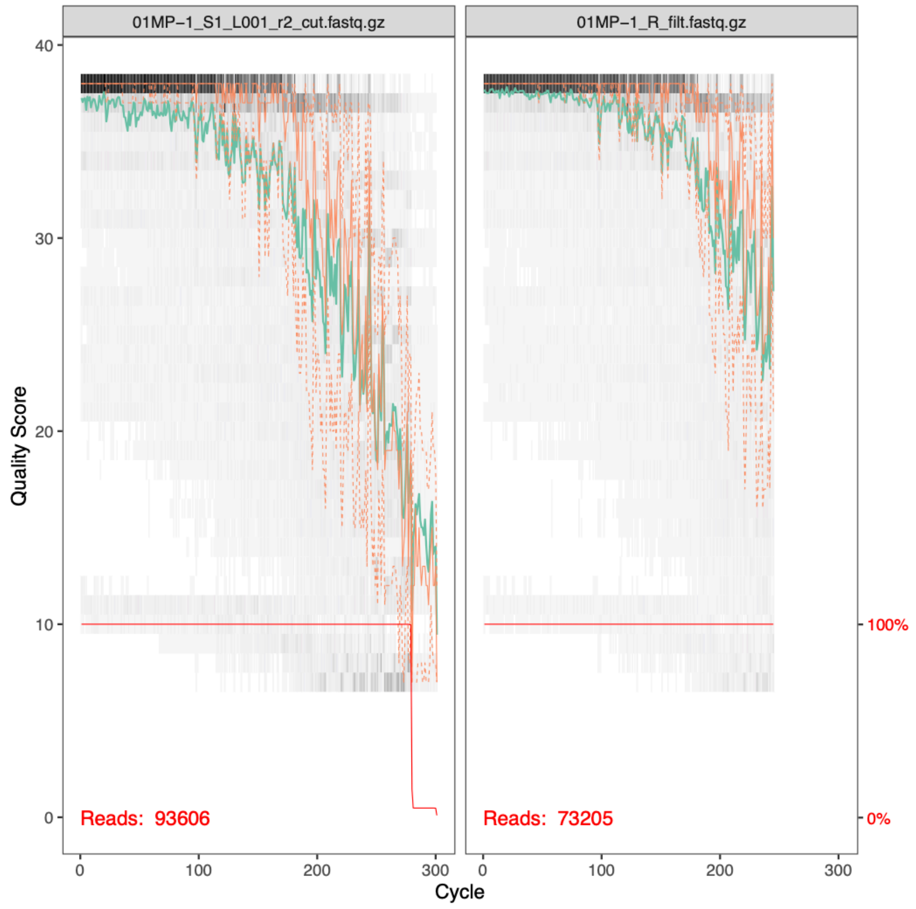
To help guide our parameter choice, here is a plot showing the percentage of reads that passed filtering as a produce of changing both the forward error rate and the reverse error rate.



Based on this, we should choose an error rate of `c(3,4)` or something similar. Note that the truncation length used here was `c(265,245)`.

After filtering, we look the effect of the filtering on the quality profiles. Below are the before and after filtering quality profiles of the forward and reverse reads of sample 01MP-1.


01MP−1_S1_L001_r1_cut.fastq.gz
01MP−1_F_filt.fastq.gz
Quality Score
40
30
20
10
0
Reads: 93606
Reads: 73205
100%
0%
0
100
200
300
Cycle

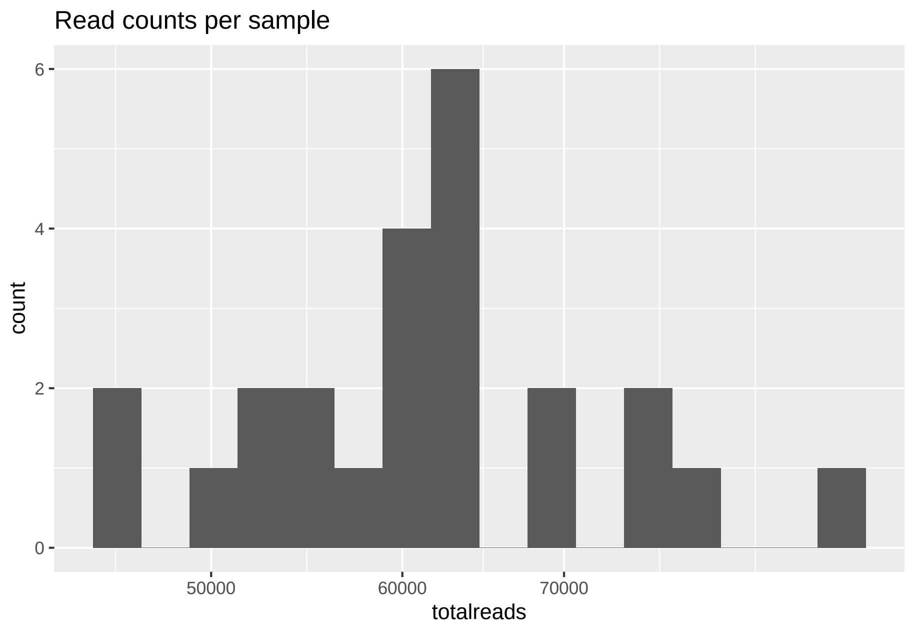


6.5.2 Task 3 - phyloseq

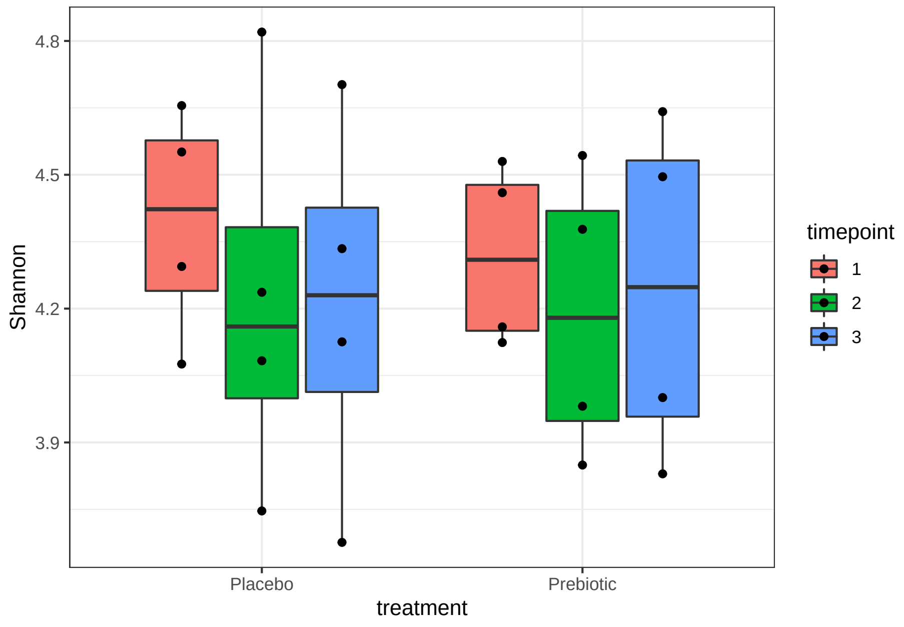
Following the dada2 pipeline, we will have a sequence table and a taxonomy table.

```
path <- "./book/project_files/"

# Read in files
seqtab <- readRDS(file.path(path, "seqtab.rds"))
taxa <- readRDS(file.path(path, "taxa.rds"))
info <- read.table(file.path(path, "project_metadata.txt"), header = TRUE)

# Match sample names
rownames(info) <- rownames(seqtab)

# Make a phyloseq object
```

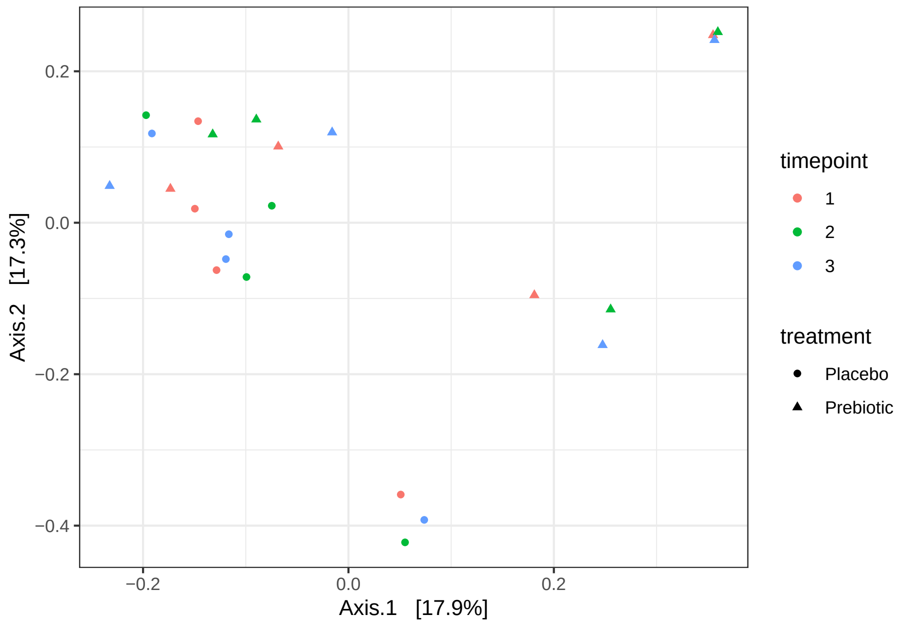
```
ps <- phyloseq(otu_table(seqtab, taxa_are_rows=FALSE), sample_data(info), tax_table(taxa))

# Make timepoint a factor
sample_data(ps)$timepoint <- as.factor(sample_data(ps)$timepoint)

## Histogram of reads per sample
sums <- rowSums(otu_table(ps))
counts <- data.frame(as(sample_data(ps), "data.frame"), totalreads = sums)

qq <- ggplot(counts, aes(totalreads)) + geom_histogram(binwidth = 0.02) + ggtitle("Read counts pe
qq
```



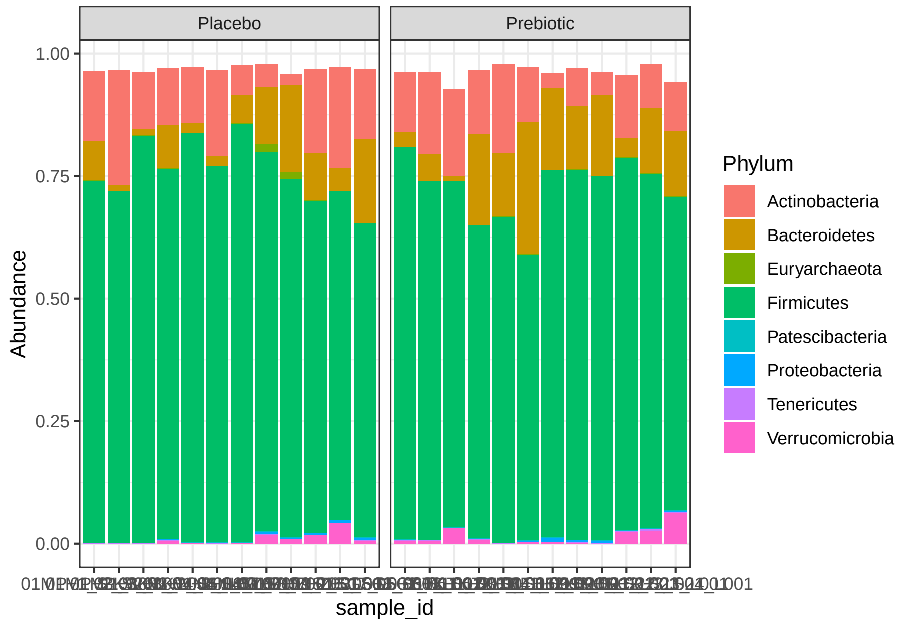
```
## Alpha diversity

## Make table of alpha diversity calculations
alpha <- estimate_richness(ps, measures = c("Chao1", "Shannon", "Simpson"))
alpha_info <- sample_data(ps)
aa <- cbind(alpha, alpha_info)


a0 <- ggplot(aa, aes(x = treatment, y = Shannon, fill = timepoint)) +
  geom_boxplot(outlier.fill = NULL, outlier.shape = 21) +
  theme_bw() +
  geom_point(position = position_dodge(width = 0.75))
a0
```



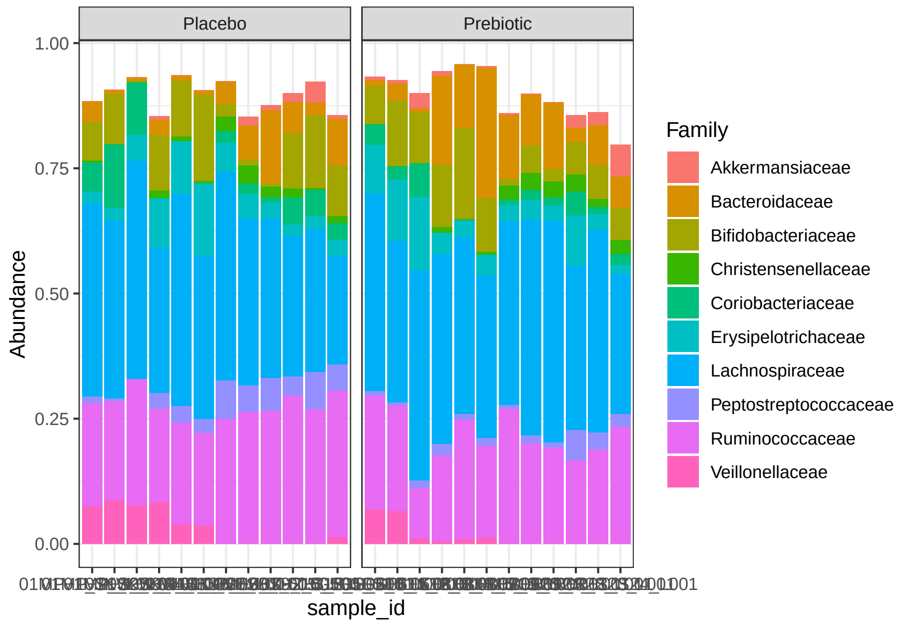
```r
## Beta diversity

# transform to relative abundace
rel <- transform_sample_counts(ps, function(x) x / sum(x))
# filter out low prevalence ASVs
prevdf <- apply(X = otu_table(ps), MARGIN = ifelse(taxa_are_rows(ps), yes = 1, no = 2)
prevdf <- data.frame(prevalence = prevdf, total_abundance = taxa_sums(ps), tax_table(ps
prevalence_threshold <- 0.05 * nsamples(ps)
keeptaxa <- rownames(prevdf)[prevdf$prevalence >= prevalence_threshold]
relf <- prune_taxa(keeptaxa, rel)
psf <- prune_taxa(keeptaxa, ps)

ord <- ordinate(relf, method = "PCoA", distance = "bray")
b0 <- plot_ordination(relf, ord, shape = "treatment", color = "timepoint") +
  theme_bw()
b0
```



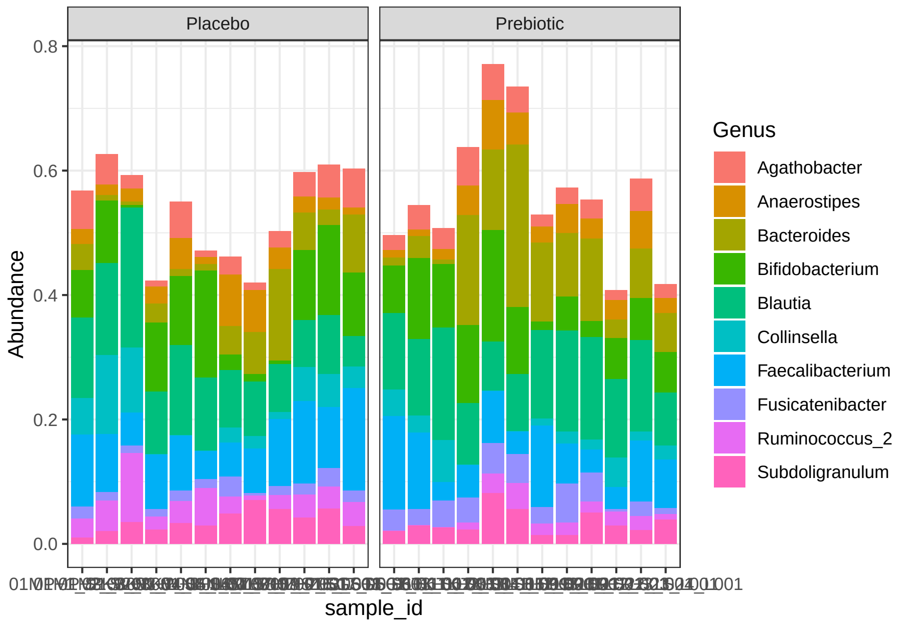
```
## Composition

# Agglomerate to phylum, family, and genus level
phy <- tax_glom(relf, "Phylum")
fam <- tax_glom(relf, "Family")
gen <- tax_glom(relf, "Genus")

phymelt <- psmelt(phy)
c0 <- ggplot(phymelt, aes(sample_id, Abundance, fill = Phylum)) +
  geom_bar(stat = "identity") +
  theme_bw() +
  facet_wrap(~treatment, scales = "free_x")
c0
```



```
top10 <- names(sort(taxa_sums(fam), TRUE))[1:10]
fam10 <- prune_taxa(top10, fam)
fammelt <- psmelt(fam10)
c1 <- ggplot(fammelt, aes(sample_id, Abundance, fill = Family)) +
  geom_bar(stat = "identity") +
  theme_bw() +
  facet_wrap(~treatment, scales = "free_x")
c1
```



```r
top10 <- names(sort(taxa_sums(gen), TRUE))[1:10]
gen10 <- prune_taxa(top10, gen)
genmelt <- psmelt(gen10)
c2 <- ggplot(genmelt, aes(sample_id, Abundance, fill = Genus)) +
  geom_bar(stat = "identity") +
  theme_bw() +
  facet_wrap(~treatment, scales = "free_x")
c2
```


Placebo
Prebiotic
Abundance
0.8
0.6
0.4
0.2
0.0
sample_id
Genus
Agathobacter
Anaerostipes
Bacteroides
Bifidobacterium
Blautia
Collinsella
Faecalibacterium
Fusicatenibacter
Ruminococcus_2
Subdoligranulum

Chapter 7

What Next?

What you have learnt so far is just the beginging!! There is a lot more ot learn and do.

For example, after looking at beta diversity, you can also look at differential abundance to see which taxa are different in each treatment group or over time.

Most commonly used methods for doing this is deseq2 and a new one is corncob. Before using them, please read the papers carefully to know the limitations and correct usage of these pacakages.

Some people also try to do functional analaysis with 16s data and for this PICRUST2 pacakge is usually used.