DADA2 Processing Script for Type 2 Diabetes association

AUTHOR
Stuyck, Olivia M

Objective

The purpose of this script is to conduct the following:

1. Data Import - take the trimmed fastq files
2. Data Wrangling / QA
3. DADA2
    1. Error rate estimation
    2. Dereplication
    3. ASV Calling
    4. Read pair merging
    5. Chimera removal
4. DECIPHER - Taxonomic Classification
5. Data Export
    1. ASV counts
    2. Taxonomy matrix
    3. ASV Fasta

1. Data Import

Lets start by loading the libraries, setting the working directory, and importing the fastq files output from trimmomatic

```
library(dada2)
```

```
Loading required package: Rcpp
```

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
✔ dplyr     1.1.2     ✔ readr     2.1.4
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.2     ✔ tibble    3.2.1
✔ lubridate 1.9.2     ✔ tidyr     1.3.0
✔ purrr     1.0.1
```

```
── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force
all conflicts to become errors
```

```
setwd("/home/jochum00/lmatz/LM_RawSequences/paired")

########################################
### DATA WRANGLING
#import the dataframe with the manifest,tsv and make vectors of the fwd
########################################

df<-read.table("../manifest.tsv",header = F,sep = "\t",row.names = NULL
colnames(df)<-"samples"
df<-df%>%mutate(fwd=paste0(samples,".1.paired.fq"),
                rev=paste0(samples,".2.paired.fq"),
                filt_fwd=paste0(samples,".1.paired.filt.fq"),
```

```
                filt_rev=paste0(samples,".2.paired.filt.fq"),
                )
```

2. Data Wrangling / QA / Filter and trimming

Make a vector containing the forward, reverse, filtered forward, and filtered reverse read name

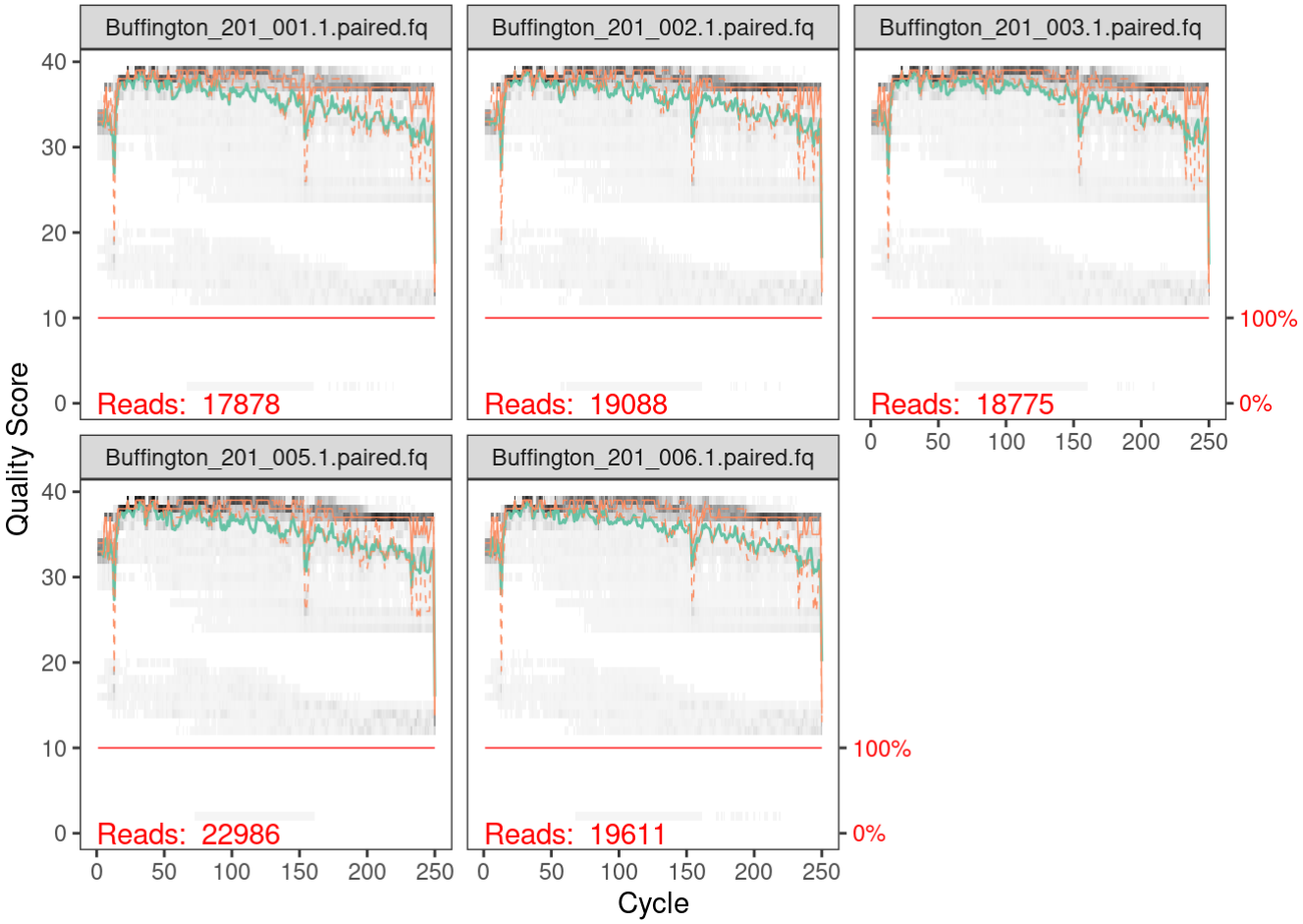
```
# one holding the file names of all the forward reads
forward_reads <- df$fwd
# and one with the reverse
reverse_reads <- df$rev

# and variables holding file names for the forward and reverse
# filtered reads we're going to generate below
filtered_forward_reads <-df$filt_fwd
filtered_reverse_reads <- df$filt_rev
```

Lets take a look at the quality profiles of the trimmed reads for the first 5 samples

```
plotQualityProfile(reverse_reads[1:5])
```

```
Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use
"none" instead as
of ggplot2 3.3.4.
ℹ The deprecated feature was likely used in the dada2 package.
  Please report the issue at
<https://github.com/benjjneb/dada2/issues>.
```


```
plotQualityProfile(forward_reads[1:5])
```



Ok looks like the read quality drops off around 200bp for the reverse, so lets truncate reverse reads starting at 200bp for quality thresholds less than Q20, and lets remove reads that are less than 50b[in length

```
filtered_out <- filterAndTrim(forward_reads,
                              filtered_forward_reads,
                              reverse_reads,
                              filtered_reverse_reads,
                              maxEE=c(2,2),
                              rm.phix=TRUE,
                              multithread = T,
                              minLen=50,
                              truncLen=c(250,200))
```

Ok lets take a look at what that did to our dataset

```
class(filtered_out) # matrix
```

```
[1] "matrix" "array"
```

```
dim(filtered_out) # 20 2
```

```
[1] 12  2
```

```
filtered_out
```

```
                                reads.in reads.out
Buffington_201_001.1.paired.fq    17878     15287
Buffington_201_002.1.paired.fq    19088     17182
Buffington_201_003.1.paired.fq    18775     16559
Buffington_201_005.1.paired.fq    22986     19944
Buffington_201_006.1.paired.fq    19611     17239
Buffington_201_007.1.paired.fq    23630     20788
Buffington_201_008.1.paired.fq    19337     17473
Buffington_201_009.1.paired.fq    13076     11231
Buffington_201_010.1.paired.fq    22531     19794
Buffington_201_011.1.paired.fq    19323     15999
Buffington_201_013.1.paired.fq    17213     15100
Buffington_201_014.1.paired.fq    21178     17680
```

```
#
# reads.in reads.out
# Buffington_201_001.1.paired.fq    17878     15287
# Buffington_201_002.1.paired.fq    19088     17182
# Buffington_201_003.1.paired.fq    18775     16559
# Buffington_201_005.1.paired.fq    22986     19944
# Buffington_201_006.1.paired.fq    19611     17239
# Buffington_201_007.1.paired.fq    23630     20788
# Buffington_201_008.1.paired.fq    19337     17473
# Buffington_201_009.1.paired.fq    13076     11231
# Buffington_201_010.1.paired.fq    22531     19794
# Buffington_201_011.1.paired.fq    19323     15999
# Buffington_201_013.1.paired.fq    17213     15100
# Buffington_201_014.1.paired.fq    21178     17680
```

Not too bad, we didnt loose too many reads so lets keep going

3. DADA2

3.1 Error rate estimation

Learn the error rates

```
# learn the error rates
err_forward_reads <- learnErrors(filtered_forward_reads, multithread=TRU
```

```
51069000 total bases in 204276 reads from 12 samples will be used for
learning the error rates.
```

```
# 51069000 total bases in 204276 reads from 12 samples will be used for
err_reverse_reads <- learnErrors(filtered_reverse_reads, multithread=TRU
```

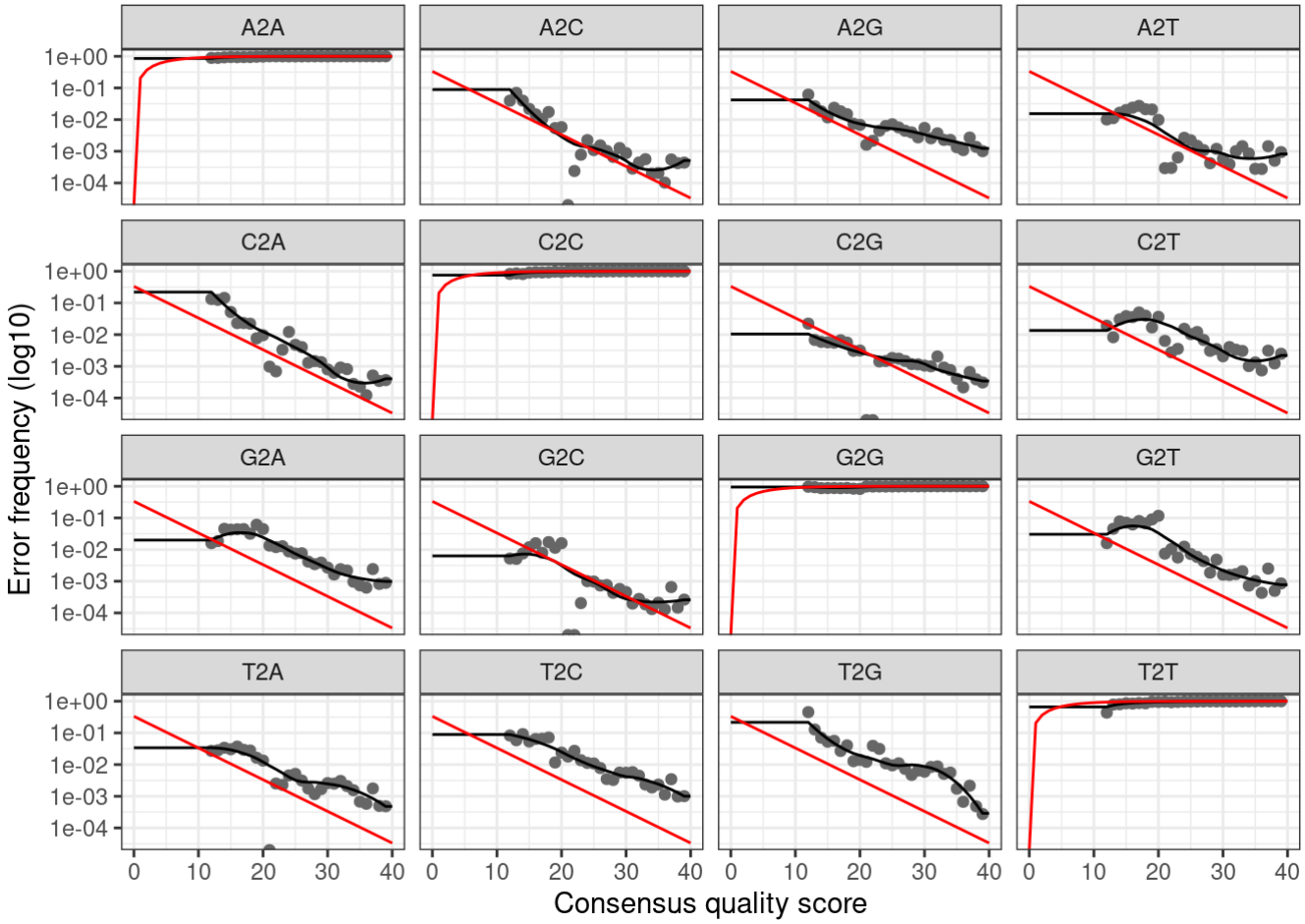
```
40855200 total bases in 204276 reads from 12 samples will be used for
learning the error rates.
```

```
# 40855200 total bases in 204276 reads from 12 samples will be used for
```

Plot the learned error rates

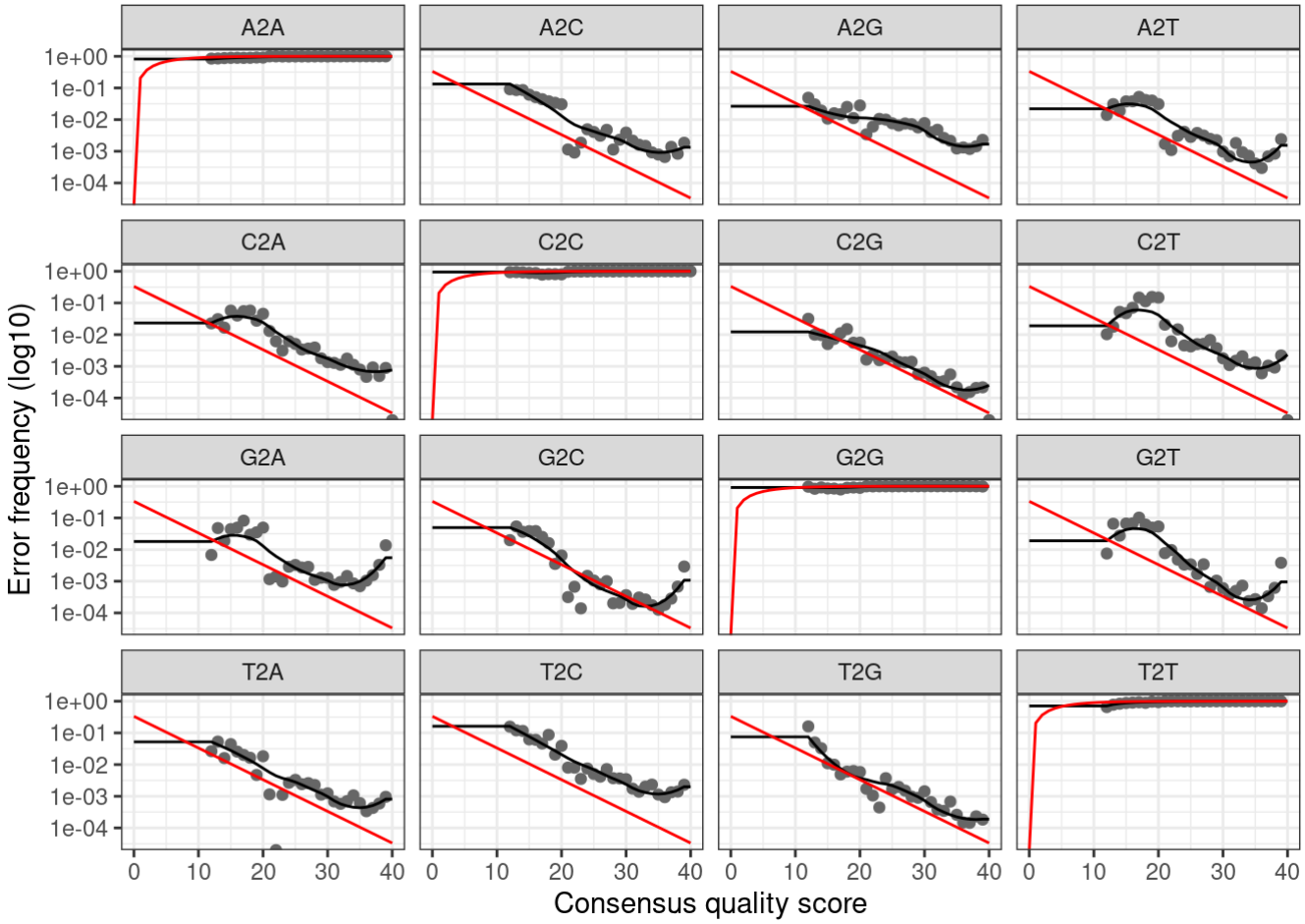
```
plotErrors(err_forward_reads, nominalQ=TRUE)
```

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




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

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



3.2 Dereplication

dereplicate the forward and reverse reads and give them identical names for matching

```
derep_forward <- derepFastq(filtered_forward_reads, verbose=TRUE)
```

```
Dereplicating sequence entries in Fastq file:
Buffington_201_001.1.paired.filt.fq

Encountered 4602 unique sequences from 15287 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_002.1.paired.filt.fq

Encountered 3557 unique sequences from 17182 total sequences read.
```

```
Dereplicating sequence entries in Fastq file:
Buffington_201_003.1.paired.filt.fq

Encountered 5143 unique sequences from 16559 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_005.1.paired.filt.fq

Encountered 5919 unique sequences from 19944 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_006.1.paired.filt.fq

Encountered 4054 unique sequences from 17239 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_007.1.paired.filt.fq

Encountered 5756 unique sequences from 20788 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_008.1.paired.filt.fq

Encountered 3211 unique sequences from 17473 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_009.1.paired.filt.fq

Encountered 3771 unique sequences from 11231 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_010.1.paired.filt.fq

Encountered 5630 unique sequences from 19794 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_011.1.paired.filt.fq

Encountered 4498 unique sequences from 15999 total sequences read.
```

```
Dereplicating sequence entries in Fastq file:
Buffington_201_013.1.paired.filt.fq

Encountered 3422 unique sequences from 15100 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_014.1.paired.filt.fq

Encountered 4404 unique sequences from 17680 total sequences read.
```

```
names(derep_forward) <-df$samples # the sample names in these objects a
derep_reverse <- derepFastq(filtered_reverse_reads, verbose=TRUE)
```

```
Dereplicating sequence entries in Fastq file:
Buffington_201_001.2.paired.filt.fq

Encountered 4534 unique sequences from 15287 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_002.2.paired.filt.fq

Encountered 3064 unique sequences from 17182 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_003.2.paired.filt.fq

Encountered 4290 unique sequences from 16559 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_005.2.paired.filt.fq

Encountered 5010 unique sequences from 19944 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_006.2.paired.filt.fq

Encountered 4532 unique sequences from 17239 total sequences read.
```

```
Dereplicating sequence entries in Fastq file:
Buffington_201_007.2.paired.filt.fq

Encountered 4292 unique sequences from 20788 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_008.2.paired.filt.fq

Encountered 3827 unique sequences from 17473 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_009.2.paired.filt.fq

Encountered 3780 unique sequences from 11231 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_010.2.paired.filt.fq

Encountered 5462 unique sequences from 19794 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_011.2.paired.filt.fq

Encountered 4286 unique sequences from 15999 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_013.2.paired.filt.fq

Encountered 3668 unique sequences from 15100 total sequences read.

Dereplicating sequence entries in Fastq file:
Buffington_201_014.2.paired.filt.fq

Encountered 4328 unique sequences from 17680 total sequences read.
```

```
names(derep_reverse) <- df$samples
```

3.3 ASV Calling

Run the DADA2 algorithm to generate the ASVS

```
#run DADA2
dada_forward <- dada(derep_forward, err=err_forward_reads, pool=T, mult
```

```
12 samples were pooled: 204276 reads in 46370 unique sequences.
```

```
# 12 samples were pooled: 204276 reads in 46370 unique sequences.
dada_reverse <- dada(derep_reverse, err=err_reverse_reads, pool=T, mult
```

```
12 samples were pooled: 204276 reads in 44362 unique sequences.
```

```
# 12 samples were pooled: 204276 reads in 44362 unique sequences.
```

3.4 Read pair merging

merging forward and reverse reads

```
#Merging forward and reverse reads
merged_amplicons <- mergePairs(dada_forward, derep_forward, dada_revers
                               derep_reverse, trimOverhang=TRUE, minOve

# this object holds a lot of information that may be the first place yo
```

Generate a count table

```
seqtab <- makeSequenceTable(merged_amplicons)
class(seqtab) # matrix
```

```
[1] "matrix" "array"
```

```
dim(seqtab) #  12 729
```

```
[1]  12 729
```

3.5 Chimera removal

```
#Chimera identification
seqtab.nochim <- removeBimeraDenovo(seqtab, verbose=T)
```

```
Identified 195 bimeras out of 729 input sequences.
```

```
# Identified 195 bimeras out of 729 input sequences.
# though we only lost 17 sequences, we don't know if they held a lot in
sum(seqtab.nochim)/sum(seqtab)
```

```
[1] 0.9485947
```

```
#  0.9485947
```

Generate an overview of the SV counts throughout all the samples

```
#Overview of counts throughout
# set a little function
getN <- function(x) sum(getUniques(x))

# making a little table
summary_tab <- data.frame(row.names=df$samples,
                          dada2_input=filtered_out[,1],
                          filtered=filtered_out[,2],
                          dada_f=sapply(dada_forward, getN),
                          dada_r=sapply(dada_reverse, getN),
                          merged=sapply(merged_amplicons, getN),
                          nonchim=rowSums(seqtab.nochim),
                          final_perc_reads_retained=round(rowSums(seqtal
```

lets write the summary table to a file for reference

```
write.table(summary_tab, "read-count-tracking.tsv", quote=FALSE, sep="\
```

4. DECIPHER - Taxonomic Classification

Lets start by downloading the silva138 reference database (if you dont already have it) and loading it into our environment

```
## loading DECIPHER
library(DECIPHER)
```

```
Loading required package: Biostrings

Loading required package: BiocGenerics


Attaching package: 'BiocGenerics'

The following objects are masked from 'package:lubridate':

    intersect, setdiff, union

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

    combine, intersect, setdiff, union

The following objects are masked from 'package:stats':

    IQR, mad, sd, var, xtabs

The following objects are masked from 'package:base':

    anyDuplicated, aperm, 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

Loading required package: S4Vectors
```

```
Loading required package: stats4


Attaching package: 'S4Vectors'

The following objects are masked from 'package:lubridate':

    second, second<-

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

    first, rename

The following object is masked from 'package:tidyr':

    expand

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:lubridate':

    %within%

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

    collapse, desc, slice

The following object is masked from 'package:purrr':

    reduce

Loading required package: XVector
```

```
Attaching package: 'XVector'

The following object is masked from 'package:purrr':

    compact

Loading required package: GenomeInfoDb


Attaching package: 'Biostrings'

The following object is masked from 'package:base':

    strsplit

Loading required package: RSQLite

Loading required package: parallel
```

```
# download the reference taxonomy data
## downloading DECIPHER-formatted SILVA v138 reference
download.file(url="http://www2.decipher.codes/Classification/TrainingSe
```

```
Warning in download.file(url =
"http://www2.decipher.codes/Classification/TrainingSets/SILVA_SSU_r138_
2019.RData",
: URL
http://www2.decipher.codes/Classification/TrainingSets/SILVA_SSU_r138_2
019.RData:
cannot open destfile 'SILVA_SSU_r138_2019.RData', reason 'Permission
denied'

Warning in download.file(url =
"http://www2.decipher.codes/Classification/TrainingSets/SILVA_SSU_r138_
2019.RData",
: download had nonzero exit status
```

```
## loading reference taxonomy object
load("SILVA_SSU_r138_2019.RData")
## creating DNAStringSet object of our ASVs
dna <- DNAStringSet(getSequences(seqtab.nochim))
## and classifying
tax_info <- IdTaxa(test=dna, trainingSet=trainingSet, strand="both", pr
```

```
=========================================================================
=========

Time difference of 271.17 secs
```

```
#########################################################################
```

Now lets make a DNAStringSet object using the nonchimeric sequences that we have generated from out ASVS

```
## creating DNAStringSet object of our ASVs
dna <- DNAStringSet(getSequences(seqtab.nochim))
```

And finally lets taxonomically classify our ASVs based on the Silva db using DECIPHER

NOTE: FOR parallel processing, you need to change the number “cores” to the number of processors that the computer running the analysis has

```
tax_info <- IdTaxa(test=dna, trainingSet=trainingSet, strand="both", pr
```

```
=========================================================================
=========

Time difference of 264.61 secs
```

5. Data Export

Lets make our sequence headers more manageable names (ASV_1, ASV_2...)

```
#Extracting the standard goods from DADA2
# giving our seq headers more manageable names (ASV_1, ASV_2...)
asv_seqs <- colnames(seqtab.nochim)
asv_headers <- vector(dim(seqtab.nochim)[2], mode="character")

for (i in 1:dim(seqtab.nochim)[2]) {
  asv_headers[i] <- paste(">ASV", i, sep="_")
}
```

5.1 ASV Counts

write out the count matrix to a file

```
# making and writing out a fasta of our final ASV seqs:
asv_fasta <- c(rbind(asv_headers, asv_seqs))
write(asv_fasta, "ASVs.fa")
```

5.2 Taxonomy

Make rank names for the taxonomy data, wrangle out the > sign for the ASV headers, and export the taxonomy data

```
ranks <- c("domain", "phylum", "class", "order", "family", "genus", "sp
asv_tax <- t(sapply(tax_info, function(x) {
  m <- match(ranks, x$rank)
  taxa <- x$taxon[m]
  taxa[startsWith(taxa, "unclassified_")] <- NA
  taxa
}))
colnames(asv_tax) <- ranks
rownames(asv_tax) <- gsub(pattern=">", replacement="", x=asv_headers)

write.table(asv_tax, "ASVs_taxonomy.tsv", sep = "\t", quote=F, col.name
```

5.3 ASV Fasta

Lets write out our fasta file containing our final ASV seqs

```
asv_fasta <- c(rbind(asv_headers, asv_seqs))
write(asv_fasta, "ASVs.fa")
```