1 Aim

This tutorial explains how to process Illumina data with the Dada2 suite as implemented in R (dada2 is also implemented in Qiime). It is adapted from:
https://vaulot.github.io/tutorials/R_dada2_tutorial.html
and https://benjjneb.github.io/dada2/tutorial.html.

A Github repository for the workshop can be found here:
https://github.com/krabberod/AeN-workshop-2020
The repository contains several R-scripts used during the workshop. The commands used particularly for the DADA2 pipeline can be found in the file DADA2_workshop.R.

2 Directory structure

The following directories are used in the pipeline, and are relative to the main working directory. The first two contains data that needs to be downloaded:

- **../fastq** : Illumina data in fastq format. Can be downloaded here from Github hor here: https://www.dropbox.com/s/erhdug0lun797iu/fastq.zip?dl=0
- **../databases** : PR2 database files (contains PR2 database formatted for dada2 - https://github.com/pr2database/pr2database/releases/)

The following will be generated during the analysis and contains output from the pipeline:

- **../fastq_filtered** : fastq files after filtration
- **../qual_pdf** : qual pdf files
- **../dada2** : dada2 processed files
- **../blast** : BLAST files output
- **../img** : Images

3 Downloads

Install the following software:

- R : https://pbil.univ-lyon1.fr/CRAN/
- R studio : https://www.rstudio.com/products/rstudio/download/#download
- Download and install the following libraries by running these lines in Rstudio

```r
install.packages("readr")     # To read and write files
install.packages("readxl")    # To read excel files
install.packages("dplyr")     # To manipulate dataframes
install.packages("tibble")    # To work with data frames
install.packages("tidyr")     # To work with data frames
install.packages("stringr")   # To manipulate strings
install.packages("ggplot2")   # To do plots
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

BiocManager::install(version = "3.10")
BiocManager::install(c("dada2", "phyloseq","Biostrings"))
```

4 Data used

The samples were collected during 2016-2017 from the sampling station DK1 in the middle of the inner Oslofjord. 2L of water was filtrated on a Sterivex filter, which collect every organism bigger than 0.2 μm.

The V4 region of the18S rRNA gene have been amplified with PCR and PCR products have been sequenced by 1 run of Illumina MiSeq 2*250 bp. The data consist of fastq files that have been subsampled with 10 000 sequences per sample.

4.1 References

- Gerikas Ribeiro C, Marie D, Lopes dos Santos A, Pereira Brandini F, Vaulot D. (2016). Estimating microbial populations by flow cytometry: Comparison between instruments. Limnol Oceanogr Methods 14:750â“758.
- Gerikas Ribeiro C, Lopes dos Santos A, Marie D, Brandini P, Vaulot D. (2018). Relationships between photosynthetic eukaryotes and nitrogen-fixing cyanobacteria off Brazil. ISME J in press.

- Gerikas Ribeiro C, Lopes dos Santos A, Marie D, Helena Pellizari V, Pereira Brandini F, Vaulot D. (2016). Pico and nanoplankton abundance and carbon stocks along the Brazilian Bight. PeerJ 4:e2587.

5 Tutorial description

5.1 Load the necessary libraries

```
library("dada2")
library("phyloseq")
library("Biostrings")
library("ggplot2")
library("dplyr")
library("tidyr")
library("tibble")
library("readxl")
library("readr")
library("stringr")
library("kableExtra")  # necessary for nice table formatting with knitr
```

5.2 Set up directories

Create directories that will be used to store the files at the different stage of the processing

```
#Make a folder that you call “Workshop Dada2” in Documents on your computer


setwd("~/Documents/DADA2_workshop") # change working directory to the
directory of your choice

fastq_dir <- "fastq/"  # fastq directory with the samples we are using
database_dir <- "databases/"  # folder with the PR2 database
https://github.com/vaulot/metabarcodes_tutorials/tree/master/databases

filtered_dir <- "fastq_filtered/"   # fastq filtered
qual_dir <- "qual_pdf/"             # qual pdf
dada2_dir <- "dada2/"               # dada2 results
blast_dir <- "blast/                # blast2 results

dir.create(filtered_dir)
dir.create(qual_dir)
dir.create(dada2_dir)
dir.create(blast_dir)
```

5.3 Primers

Note that the primers are degenerated. Dada2 has an option to remove primers (FilterandTrim) but this function will not accept degeneracy.

```
primer_set_fwd = c("CCAGCASCYGCGGTAATTCC")
primer_set_rev = c("ACTTTCGTTCTTGATYRATGA")

primer_length_fwd <- str_length(primer_set_fwd[1])
primer_length_rev <- str_length(primer_set_rev[1])
```

5.4 PR2 tax levels

```
PR2_tax_levels <- c("Kingdom", "Supergroup", "Division", "Class", "Order",
"Family", "Genus", "Species")
```

5.5 Examine the fastQ files

5.5.1 Construct a list of the fastq files

It is assumed that the sample names are at the start of file name and separated by _. I.e. A file called *S01_L001_R1_001.fastq* is assumed to be from the sample *S01*. The rest of the information is related to the sequencing.

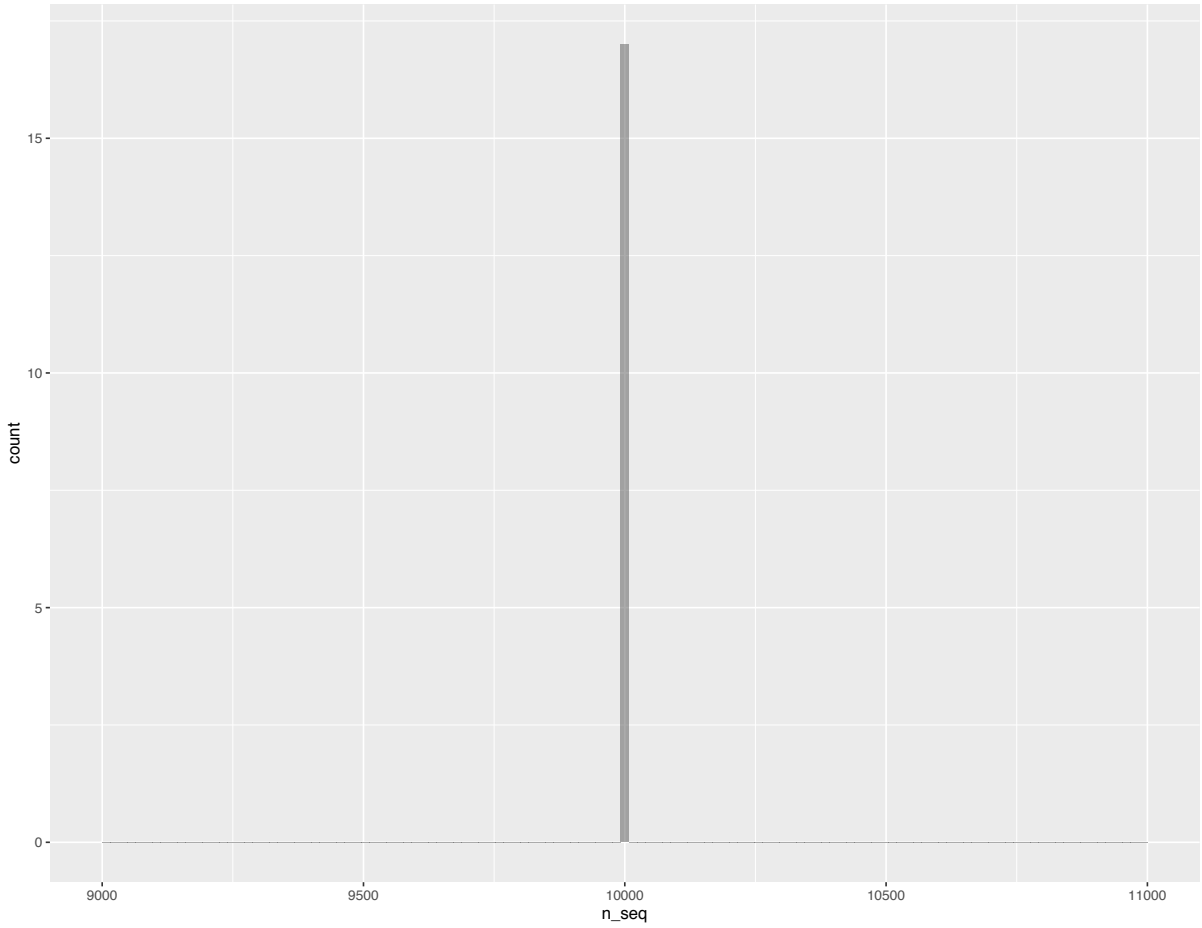
```
# get a list of all fastq files in the ngs directory and separate R1 and R2

fns <- sort(list.files(fastq_dir, full.names = TRUE))
fns <- fns[str_detect(basename(fns), ".fastq")]
fns_R1 <- fns[str_detect(basename(fns), "R1")]
fns_R2 <- fns[str_detect(basename(fns), "R2")]


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

sample.names <- str_split(basename(fns_R1), pattern = "_", simplify = TRUE)
sample.names <- sample.names[, 1]
```

5.5.2 Compute number of paired reads

```
# create an empty data frame
df <- data.frame()

# loop through all the R1 files (no need to go through R2 which should be
# the same)

for (i in 1:length(fns_R1)) {

    # use the dada2 function fastq.geometry
    geom <- fastq.geometry(fns_R1[i])

    # extract the information on number of sequences and file name
    df_one_row <- data.frame(n_seq = geom[1], file_name =
basename(fns_R1[i]))

    # add one line to data frame
    df <- bind_rows(df, df_one_row)
}

# display number of sequences and write data to small file
knitr::kable(df)

View(df)
```

	n_seq	File_name
1	10000	S01_L001_R1_001.fastq
2	10000	S02_L001_R1_001.fastq
3	10000	S03_L001_R1_001.fastq
4	10000	S04_L001_R1_001.fastq
5	10000	S05_L001_R1_001.fastq
6	10000	S06_L001_R1_001.fastq
7	10000	S07_L001_R1_001.fastq
8	10000	S08_L001_R1_001.fastq
9	10000	S09_L001_R1_001.fastq
10	10000	S10_L001_R1_001.fastq
11	10000	S11_L001_R1_001.fastq
12	10000	S12_L001_R1_001.fastq
13	10000	S13_L001_R1_001.fastq
14	10000	S14_L001_R1_001.fastq
15	10000	S15_L001_R1_001.fastq
16	10000	S16_L001_R1_001.fastq
17	10000	S17_L001_R1_001.fastq

```
# write.table(df, file = 'n_seq.txt', sep='\t', row.names = FALSE, na='',
# quote=FALSE)

# plot the histogram with number of sequences

ggplot(df, aes(x = n_seq)) + geom_histogram(alpha = 0.5, position =
       "identity", binwidth = 100) + xlim(0, 20000)
```



5.5.3 Plot quality for reads

```
for (i in 1:length(fns)) {

    # Use dada2 function to plot quality
    p1 <- plotQualityProfile(fns[i])

    # Only plot on screen for first 2 files
    if (i <= 2) {
        print(p1)
    }
```

```
    # save the file as a pdf file (uncomment to execute)

    p1_file <- paste0(qual_dir, basename(fns[i]), ".qual.pdf")
    ggsave(plot = p1, filename = p1_file, device = "pdf", width = 15,
height = 15, scale = 1, units = "cm")
}
```

Question: Have a look at the quality plots you created. By visual estimation from the graphs, what is the sequence length where quality drops below acceptable for downstream analysis? Hint: read about “phred scores».

5.6 Filter and Trim the reads

The dada2 algorithm requires primers to be removed prior to processing.

- Using dada2 there are 2 possibilities
  - Remove by sequence, but dada2 does not allow for ambiguities
  - Remove by position, which is not a problem for Illumina sequences but is a problem for 454

- For complex situation we recommend to use **cutadapt** to remove the primers : http://cutadapt.readthedocs.io/en/stable/guide.html#.
  The program is really very powerful.

5.6.1 Create names for the filtered files

We create the name of the files that will be generated by the `filterAndTrim` function in the step below. These names are composed by the path name ("../fastq_filtered/"), the sample names, the read number (R1 or R2) and a "_filt" suffix.

```
filt_R1 <- str_c(filtered_dir, sample.names, "_R1_filt.fastq")
filt_R2 <- str_c(filtered_dir, sample.names, "_R2_filt.fastq")
```

5.6.2 Removing the primers by sequence (DO NOT EXECUTE THIS STEP)

- **Go to next step (5.6.3)**

The next piece of code *could be used* to remove the primers by **sequence**. The dada2 package does not allow for primer degeneracy. Since our forward primer is degenerated at two positions, all four combinations need to be tested. However, it will be necessary to re-assemble after that the 4 fastQ files created (which has not done). A better strategy in this case is to remove primer by truncation (see next step).

```
# On Windows set multithread=FALSE

out_all <- data.frame(id = length(fns_R1))

for (i in 1:4) {out <- filterAndTrim(fns_R1, filt_R1, fns_R2, filt_R2,
        truncLen = c(250, 200), trimLeft = c(0, 0), maxN = 0, maxEE =
        c(Inf, Inf), truncQ = 10, rm.phix = TRUE, primer.fwd =
        primer_set_fwd[i], compress = FALSE, multithread = FALSE)out_all <-
        cbind(out_all, out)}

knitr::kable(out_all, "latex") %>% kable_styling(bootstrap_options =
        "striped", font_size = 7)
```

5.6.3 Remove primers by truncation and filter

Filter out all sequences with N

```
out <- filterAndTrim(fns_R1, filt_R1, fns_R2, filt_R2, truncLen = c(250,
        200), trimLeft = c(primer_length_fwd, primer_length_rev), maxN = 0,
        maxEE = c(2, 2), truncQ = 2, rm.phix = TRUE, compress = FALSE,
        multithread = FALSE)
```

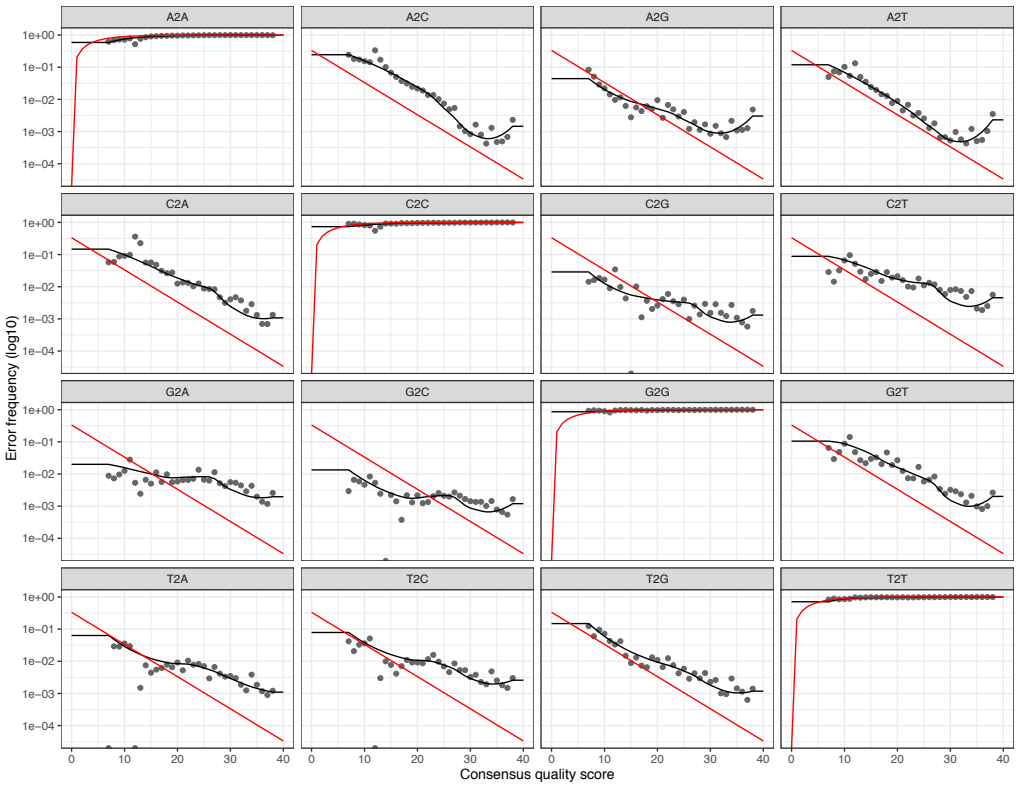
5.7 Dada2 processing

5.7.1 Learn error rates

The error rates are plotted.

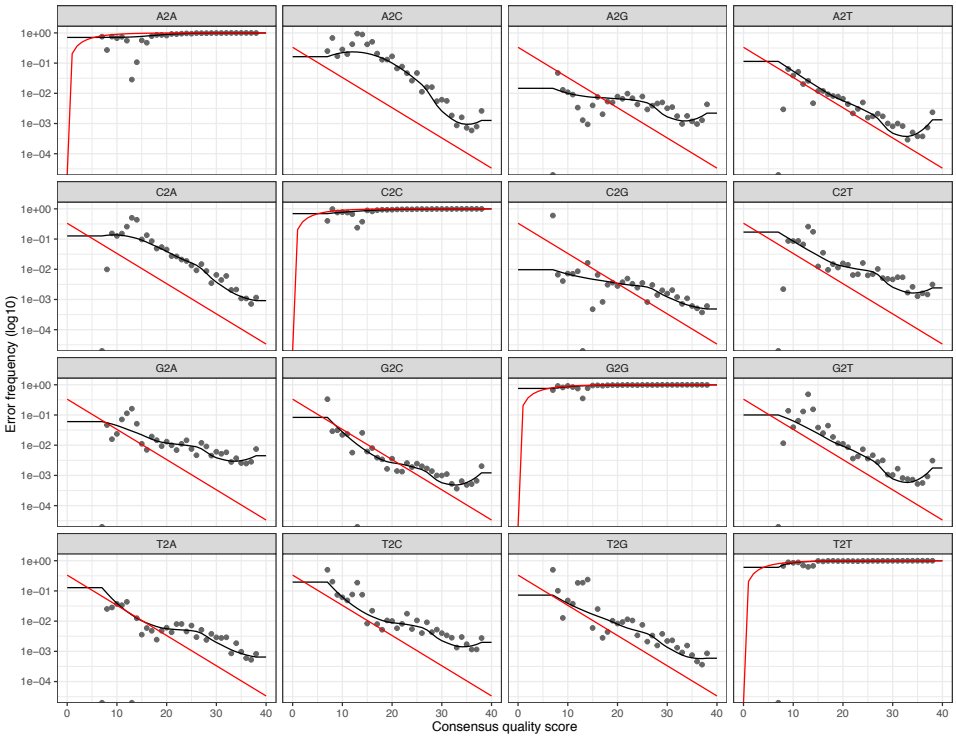
```
err_R1 <- learnErrors(filt_R1, multithread = FALSE)

plotErrors(err_R1, nominalQ = TRUE)
```

```
4542380 total bases in 106706 reads from 17 samples will be used for
learning the error rates.
```




```
err_R2 <- learnErrors(filt_R2, multithread = FALSE)

plotErrors(err_R2, nominalQ = TRUE)
```

```
9100374 total bases in 106706 reads from 17 samples will be used for
learning the error rates.
```



Question: Why do we need an error model to denoise our sequences? What are the potential sources of errors and consequences for downstream data interpretation?

5.7.2 Dereplicate the reads

```
derep_R1 <- derepFastq(filt_R1, verbose = FALSE)
derep_R2 <- derepFastq(filt_R2, verbose = FALSE)


# Name the derep-class objects by the sample names
names(derep_R1) <- sample.names
names(derep_R2) <- sample.names
```

5.7.3 Sequence-variant inference algorithm to the dereplicated data

```
dada_R1 <- dada(derep_R1, err = err_R1, multithread = FALSE, pool = FALSE)
```

```
Sample 1 - 6272 reads in 2844 unique sequences.
Sample 2 - 5988 reads in 2488 unique sequences.
Sample 3 - 5534 reads in 2405 unique sequences.
Sample 4 - 6329 reads in 2504 unique sequences.
```

```
Sample 5 - 6457 reads in 2380 unique sequences.
Sample 6 - 6183 reads in 2807 unique sequences.
Sample 7 - 6222 reads in 2674 unique sequences.
Sample 8 - 6375 reads in 2310 unique sequences.
Sample 9 - 6259 reads in 2209 unique sequences.
Sample 10 - 6560 reads in 2438 unique sequences.
Sample 11 - 6549 reads in 2334 unique sequences.
Sample 12 - 6174 reads in 2034 unique sequences.
Sample 13 - 6313 reads in 2281 unique sequences.
Sample 14 - 6209 reads in 1792 unique sequences.
Sample 15 - 6285 reads in 2262 unique sequences.
Sample 16 - 6776 reads in 2255 unique sequences.
Sample 17 - 6221 reads in 2685 unique sequences.
```

```
dada_R2 <- dada(derep_R2, err = err_R2, multithread = FALSE, pool = FALSE)
```

```
Sample 1 - 6272 reads in 2813 unique sequences.
Sample 2 - 5988 reads in 2551 unique sequences.
Sample 3 - 5534 reads in 2207 unique sequences.
Sample 4 - 6329 reads in 2930 unique sequences.
Sample 5 - 6457 reads in 2408 unique sequences.
Sample 6 - 6183 reads in 2763 unique sequences.
Sample 7 - 6222 reads in 2834 unique sequences.
Sample 8 - 6375 reads in 2485 unique sequences.
Sample 9 - 6259 reads in 2350 unique sequences.
Sample 10 - 6560 reads in 2685 unique sequences.
Sample 11 - 6549 reads in 2301 unique sequences.
Sample 12 - 6174 reads in 2161 unique sequences.
Sample 13 - 6313 reads in 2593 unique sequences.
Sample 14 - 6209 reads in 2130 unique sequences.
Sample 15 - 6285 reads in 2341 unique sequences.
Sample 16 - 6776 reads in 2486 unique sequences.
Sample 17 - 6221 reads in 2458 unique sequences.
```

```
dada_R1[[1]]
```

```
dada-class: object describing DADA2 denoising results
146 sequence variants were inferred from 2844 input unique sequences.
Key parameters: OMEGA_A = 1e-40, OMEGA_C = 1e-40, BAND_SIZE = 16
```

```
dada_R2[[1]]
```

```
dada-class: object describing DADA2 denoising results
```

```
85 sequence variants were inferred from 2813 input unique sequences.
Key parameters: OMEGA_A = 1e-40, OMEGA_C = 1e-40, BAND_SIZE = 16
```

Question: What happens during the dereplication and sample inference step, respectively?

5.7.4 Merge sequences

```
mergers <- mergePairs(dada_R1, derep_R1, dada_R2, derep_R2, verbose = TRUE)


# Inspect the merger data.frame from the first sample

knitr::kable(head(mergers[[1]]))
```

sequence	abundance	forward	reverse	nmatch	nmismatch	nindel	prefer	accept
AGCTCCA..	736	1	2	31	0	0	1	TRUE
AGCTCCA..	404	2	1	31	0	0	2	TRUE
AGCTCCA..	308	4	3	27	0	0	2	TRUE
AGCTCTA..	282	3	5	37	0	0	1	TRUE
AGCTCCA..	265	5	4	26	0	0	1	TRUE
AGCTCCA..	172	6	9	31	0	0	2	TRUE

5.7.5 Make sequence table

```
seqtab <- makeSequenceTable(mergers)

dim(seqtab)
```

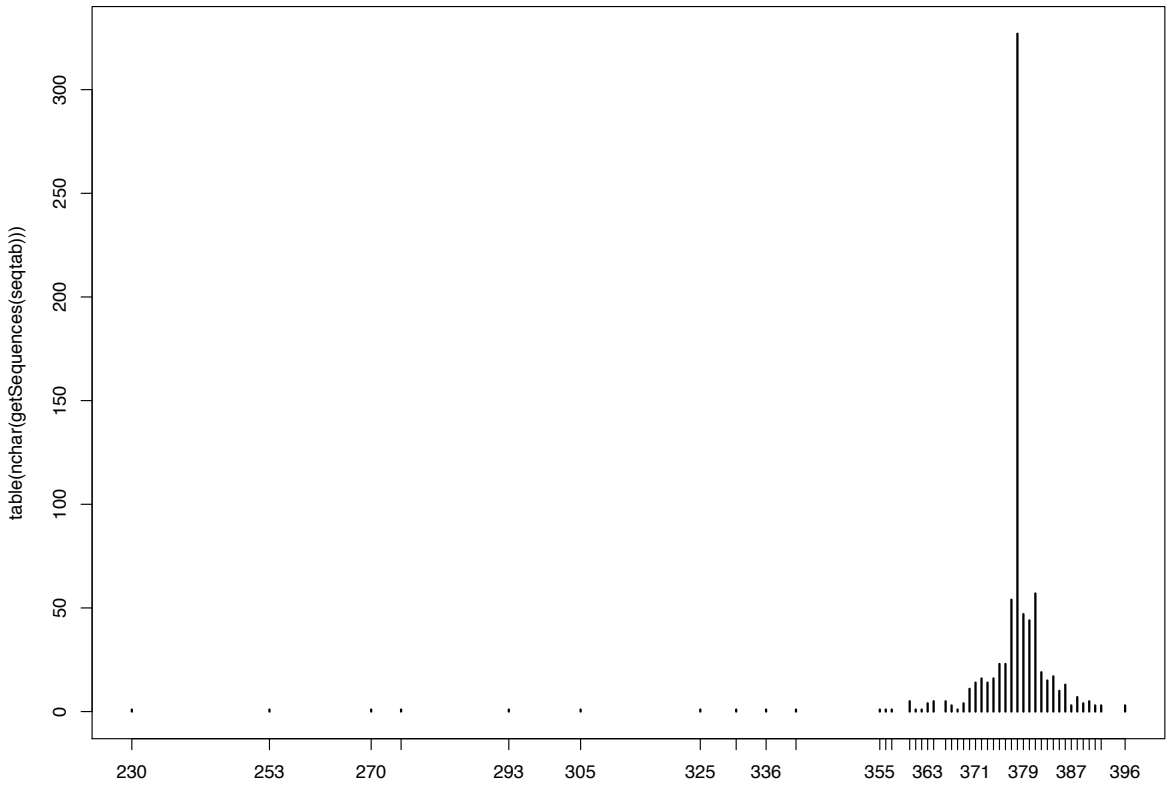
```
[1] 17 790
```

```
# Make a transposed of the seqtab to make it be similar to mothur database

t_seqtab <- t(seqtab)


# Inspect distribution of sequence lengths

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

```
30 253 270 275 293 305 325 331 336 341 355 356 357 360 361 362 363 364 366 367
```

```
  1   1   1   1   1   1   1   1   1   1   1   1   1   5   1   1   4   5   5   3 
368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 
  1   4  11  14  16  14  16  23  23  54 327  47  44  57  19  15  17  10  13   3 
388 389 390 391 392 396 
  7   4   5   3   3   3 
```

```
#simple plot of length distribution

plot(nchar(getSequences(seqtab)))
```



5.7.6 Remove chimeras

Note that remove chimeras will produce spurious results if primers have not be removed. The parameter `methods` can be `pooled` or `consensus`

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


# Compute % of non chimeras

paste0("% of non chimeras : ", sum(seqtab.nochim)/sum(seqtab) * 100)
```

```
[1] "% of non chimeras : 98.5313859477355"
```

```
paste0("total number of sequences : ", sum(seqtab.nochim))
```

```
[1] "total number of sequences : 97618"
```

Question: What are chimeras? And why do they need to be removed?

5.7.7 Track number of reads at each step

```
# define a function

getN <- function(x) sum(getUniques(x))


track <- cbind(out, sapply(dada_R1, getN), sapply(mergers, getN),
        rowSums(seqtab), rowSums(seqtab.nochim))


colnames(track) <- c("input", "filtered", "denoised", "merged", "tabled",
        "nonchim")


rownames(track) <- sample.names


knitr::kable(track)

# View the output

track
```

	input	filtered	denoised	merged	tabled	nonchim
S01	10000	6272	6052	5564	5564	5410
S02	10000	5988	5848	5556	5556	5517
S03	10000	5534	5376	4988	4988	4962
S04	10000	6329	6172	5841	5841	5761
S05	10000	6457	6334	6111	6111	6012
S06	10000	6183	5924	5512	5512	5421
S07	10000	6222	6016	5653	5653	5557
S08	10000	6375	6170	5878	5878	5809
S09	10000	6259	6162	6038	6038	5985
S10	10000	6560	6427	6335	6335	6249
S11	10000	6549	6437	6242	6242	6037
S12	10000	6174	6061	5846	5846	5819
S13	10000	6313	6194	5960	5960	5828
S14	10000	6209	6059	5936	5936	5898
S15	10000	6285	6172	5877	5877	5813
S16	10000	6776	6610	6292	6292	6220
S17	10000	6221	5852	5444	5444	5320

```
write_tsv(data.frame(track), str_c(dada2_dir, "read_numbers_dada2.tsv"))
```

Questions: How many sequences remain after quality trimming (in each sample)? How many sequences remain after running derepFastq command? How many unique sequences were identified from each sample?

5.7.8 Transforming and saving the ASVs sequences

In the OTU put of dada2, OTU names are the sequences. We change to give a Otuxxx name and the sequences are stored in the taxonomy table.

```
seqtab.nochim_trans <- as.data.frame(t(seqtab.nochim)) %>%
        rownames_to_column(var = "sequence") %>% rowid_to_column(var =
        "OTUNumber") %>% mutate(OTUNumber = sprintf("otu%04d", OTUNumber))
        %>% mutate(sequence = str_replace_all(sequence, "(-|\\.)", ""))


df <- seqtab.nochim_trans

seq_out <- Biostrings::DNAStringSet(df$sequence)


names(seq_out) <- df$OTUNumber


Biostrings::writeXStringSet(seq_out, str_c(dada2_dir,
        "CARBOM_ASV_no_taxo.fasta"), compress = FALSE, width = 20000)
```

5.7.9 Assigning taxonomy

This step is quite long (20-30 min. depending on the computer)… Start before you take a break.

```
pr2_file <- paste0(database_dir, "pr2_version_4.72_dada2.fasta.gz")

taxa <- assignTaxonomy(seqtab.nochim, refFasta = pr2_file, taxLevels =
        PR2_tax_levels, minBoot = 0, outputBootstraps = TRUE, verbose =
        TRUE)

saveRDS(taxa, str_c(dada2_dir, "CARBOM.taxa.rds"))
```

Why is the choice of database important?

5.7.10 Export data as produced by Dada2

```
taxa <- readRDS(str_c(dada2_dir, "CARBOM.taxa.rds"))

write_tsv(as.tibble(taxa$tax), path = str_c(dada2_dir, "taxa.txt"))

write_tsv(as.tibble(taxa$boot), path = str_c(dada2_dir, "taxa_boot.txt"))

write_tsv(as.tibble(seqtab.nochim), path = str_c(dada2_dir, "seqtab.txt"))
```

5.7.11 Appending taxonomy and boot to the sequence table

```
taxa_tax <- as.data.frame(taxa$tax)

taxa_boot <- as.data.frame(taxa$boot) %>% rename_all(funs(str_c(.,
"_boot")))

seqtab.nochim_trans <- taxa_tax %>% bind_cols(taxa_boot) %>%
        bind_cols(seqtab.nochim_trans)
```

5.7.12 Filter for 18S

Remove the sequences are not 18S by selecting only bootstrap value for Supergroup in excess of 80.

```
bootstrap_min <- 80

# Filter based on the bootstrap
```

```
seqtab.nochim_18S <- seqtab.nochim_trans %>% dplyr::filter(Supergroup_boot
        >= bootstrap_min)


# Create a database like file for dada2

write_tsv(seqtab.nochim_18S, str_c(dada2_dir, "CARBOM_dada2.database.tsv"))
```

Question: Why do have sequences that are not recognized as 18S in the data set?

5.7.13 Write FASTA file for BLAST analysis with taxonomy

Use the Biostrings library

```
df <- seqtab.nochim_18S

seq_out <- Biostrings::DNAStringSet(df$sequence)


names(seq_out) <- str_c(df$OTUNumber, df$Supergroup, df$Division, df$Class,
        df$Order, df$Family, df$Genus, df$Species, sep = "|")


Biostrings::writeXStringSet(seq_out, str_c(blast_dir, "CARBOM_ASV.fasta"),
        compress = FALSE, width = 20000)
```

This file can be sent to a server and a BLAST analysis can be done using the following slurm (which is used by for instance Saga).

```
#!/bin/bash

#### EXTRA Example for blast on Saga ####
##!/bin/sh
##SBATCH --job-name=blastn
##SBATCH --account=[insert account here]
##SBATCH --output=slurm-%j.base
##SBATCH --cpus-per-task=16
##SBATCH --time=100:00:00
##SBATCH --mem-per-cpu=6G
#
#module purge
#module load BLAST+/2.8.1-intel-2018b
#
#FASTA=OsloFjord_ASV.fasta
#BLAST_TSV=OsloFjord_.blast.tsv
```

```
#DB=/cluster/shared/databases/blast/latest/nt
#
#
#OUT_FMT="6 qseqid sseqid sacc stitle sscinames staxids sskingdoms
sblastnames pident slen length mismatch gapopen qstart qend sstart send
evalue bitscore"
#
#blastn -max_target_seqs 100 -evalue 1.00e-10 -query $FASTA -out $BLAST_TSV
-db "$DB" -outfmt "$OUT_FMT" -num_threads 16
###############
```

Or for using qsub file for other servers

```
#!/bin/bash
# Commands starting with  '#$' are interpreted by SGE
# Shell to be used for the job
#$ -S /bin/bash
# User to be informed
#$ -M vaulot@sb-roscoff.fr
# Export all environment variable
#$ -V
# Send a message by email  at beginning (b), end (e) and abort (a) of job
#$ -m bea
# Standard output.  Can use '-j y' to add stderr with stdout
#$ -o repl
# Send the commande from the curent directory where the script reside
#$ -cwd
# Define environmental variables
# submitted with
# qsub -q short.q qsub_blast_antar.sh
# Replace the next line by the location of the directory where you have
your data

DIR_PROJECT="/projet/sbr/ccebarcodep1408/workshop_nz_2018/blast/"

cd $DIR_PROJECT

FILE="CARBOM_ASV"

FASTA=$DIR_PROJECT$FILE".fasta"

BLAST_TSV=$DIR_PROJECT$FILE".blast.tsv"
```

```
OUT_FMT="6 qseqid sseqid sacc stitle sscinames staxids sskingdoms
sblastnames pident slen length mismatch gapopen qstart qend sstart send
evalue bitscore"

blastn -max_target_seqs 100 -evalue 1.00e-10 -query $FASTA -out $BLAST_TSV
-db /db/blast/all/nt -outfmt "$OUT_FMT"
```

5.8 Phyloseq

- We can now create a phyloseq object from dada2 results. This object can be used for further statistical analyses in R.

```
samdf <- data.frame(sample_name = sample.names)

rownames(samdf) <- sample.names


OTU <- seqtab.nochim_18S %>% column_to_rownames("OTUNumber") %>%
        select_if(is.numeric) %>% select(-contains("_boot")) %>%
        as.matrix() %>% otu_table(taxa_are_rows = TRUE)


TAX <- seqtab.nochim_18S %>% column_to_rownames("OTUNumber") %>%
        select(Kingdom:Species) %>% as.matrix() %>% tax_table()


ps_dada2 <- phyloseq(OTU, sample_data(samdf), TAX)

#Save the Rdata:

saveRDS(ps_dada2, str_c(dada2_dir, "CARBOM_phyloseq.rds"))

# Alternatively, save the entire workspace. This is highly recommended if
you run the script on a server and want to look at the data afterwards.
save.image(str_c(dada2_dir, "DADA2_workspace.Rdata"))

# This image can easily be loaded with
# load(str_c(dada2_dir, "DADA2_workspace.Rdata"))
```