Dada2 pipeline in R

Marko Suokas

Load libraries

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

```
[1] '1.32.0'
```

```
library(knitr);packageVersion("knitr")
```

```
[1] '1.48'
```

```
library(Biostrings);packageVersion("Biostrings")
```

```
[1] '2.72.1'
```

```
library(DECIPHER);packageVersion("DECIPHER")
```

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

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

```
[1] '1.48.0'
```

```
library(tidyverse);packageVersion("tidyverse")
```

```
[1] '2.0.0'
```

```
library(kableExtra);packageVersion("kableExtra")
```

```
[1] '1.4.0'
```

```
library(patchwork);packageVersion("patchwork")
```

```
[1] '1.2.0'
```

Parameters

```
#Path variables
path <- "data/reads"
training <- "~/feature_classifiers/SILVA_SSU_r138_2019.RData"
meta_file <- "data/metadata.tsv"
exportloc <- "result_tables"
#Truncation length and phix (Illumina)
truncation <- 245
phi <- FALSE
#Name of first column in metadata file
meta_1stcol <- "Sampleid"
#Create results directory
dir.create(exportloc)
```

Sample lists

```
#List files in path
list.files(path)
```

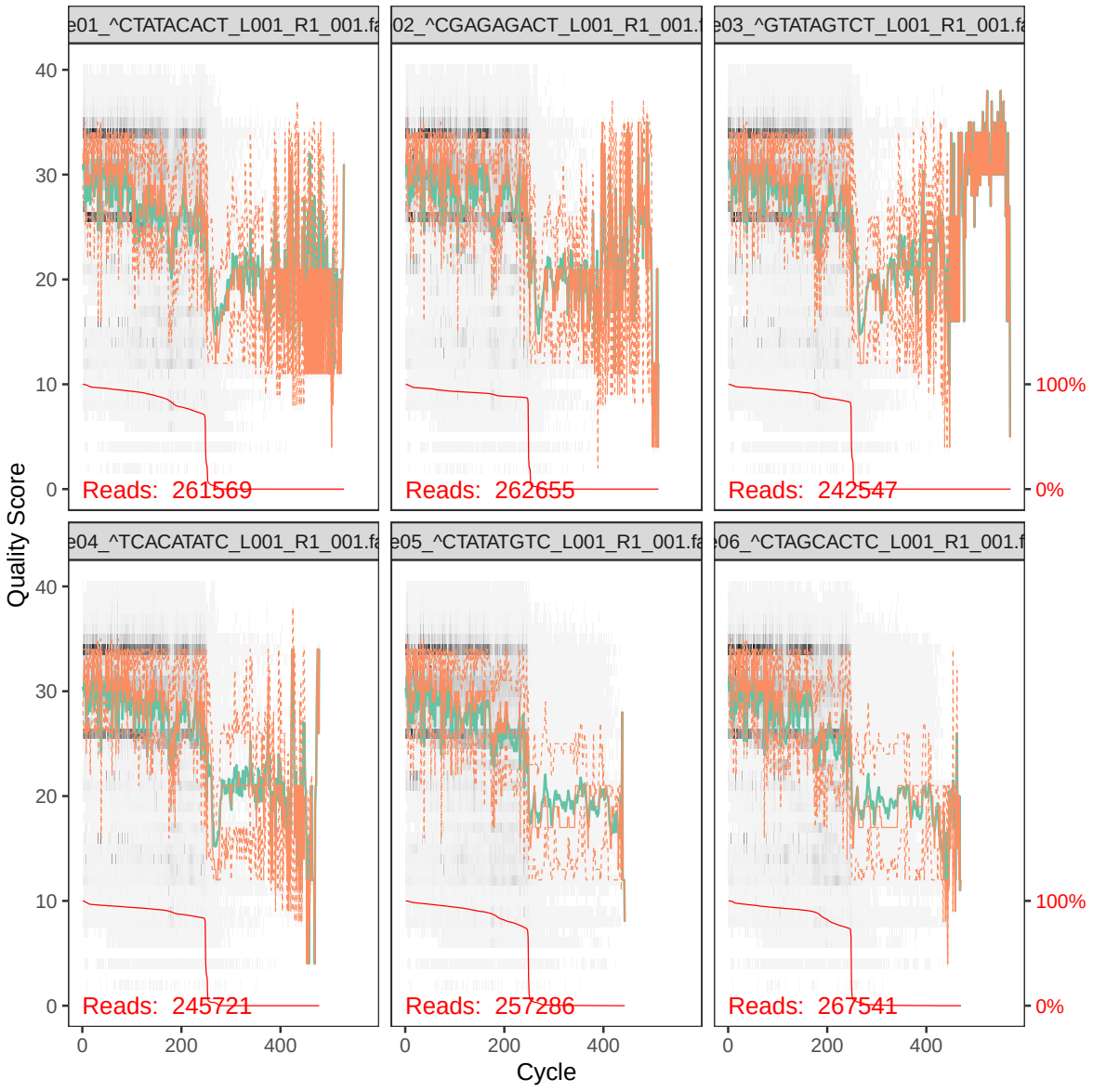
```
 [1] "Sample01_^CTATACACT_L001_R1_001.fastq.gz"
 [2] "Sample02_^CGAGAGACT_L001_R1_001.fastq.gz"
 [3] "Sample03_^GTATAGTCT_L001_R1_001.fastq.gz"
 [4] "Sample04_^TCACATATC_L001_R1_001.fastq.gz"
 [5] "Sample05_^CTATATGTC_L001_R1_001.fastq.gz"
 [6] "Sample06_^CTAGCACTC_L001_R1_001.fastq.gz"
 [7] "Sample07_^ATAGCTGTC_L001_R1_001.fastq.gz"
 [8] "Sample08_^ATACGACTC_L001_R1_001.fastq.gz"
 [9] "Sample09_^ACATGATCT_L001_R1_001.fastq.gz"
[10] "Sample10_^CTACGCATC_L001_R1_001.fastq.gz"
[11] "Sample11_^TCATGCGTC_L001_R1_001.fastq.gz"
[12] "Sample12_^TCATGTACT_L001_R1_001.fastq.gz"
[13] "Sample13_^CTACGTGCT_L001_R1_001.fastq.gz"
[14] "Sample14_^TCAGTATCT_L001_R1_001.fastq.gz"
[15] "Sample15_^GCAGTCGTC_L001_R1_001.fastq.gz"
[16] "Sample16_^TCAGTGCTC_L001_R1_001.fastq.gz"
[17] "Sample17_^AGCTACACT_L001_R1_001.fastq.gz"
[18] "Sample18_^CGCTAGTCT_L001_R1_001.fastq.gz"
[19] "filtered"
```

```
#Filenames have format: SAMPLENAME_R1_001.fastq
fnFs <- sort(list.files(path, pattern = "_R1_001.fastq", full.names = TRUE))
# Extract sample names, assuming pattern
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
#Filtered files will be placed in filtered/ subdirectory
filtFs <- file.path(path, "filtered", paste0(sample.names, "_F_filt.fastq.gz"))
names(filtFs) <- sample.names
```

Tip: If you have numbered samples, use 0X format. Otherwise you have problems in sort order.

Quality profile

```
# Base quality plot in first 6 samples
plotQualityProfile(fnFs[1:6])
```



Filtering and trimming reads

```
#Filtered files will be placed in filtered/ subdirectory
out <- filterAndTrim(fnFs, filtFs, truncLen=245,
                     maxN=0, maxEE=2, truncQ=2,
                     compress=TRUE, multithread=FALSE)
#Output is saved to rds file, so we don't have to recalculate, if we make changes
#If you are making changes to chunk, change eval = TRUE
saveRDS(out,"rds/out.rds")
```

```
#read rds file
out <- readRDS("rds/out.rds")
```

Considerations: The standard parameters are starting points. If you want to speed up downstream computation, consider tightening `maxEE`. If too few reads are passing the filter, consider relaxing `maxEE`.

For ITS sequencing, it is usually undesirable to truncate reads to a fixed length due to the large length variation at that locus. You can omit in this case truncLen parameter.

Learn error rates

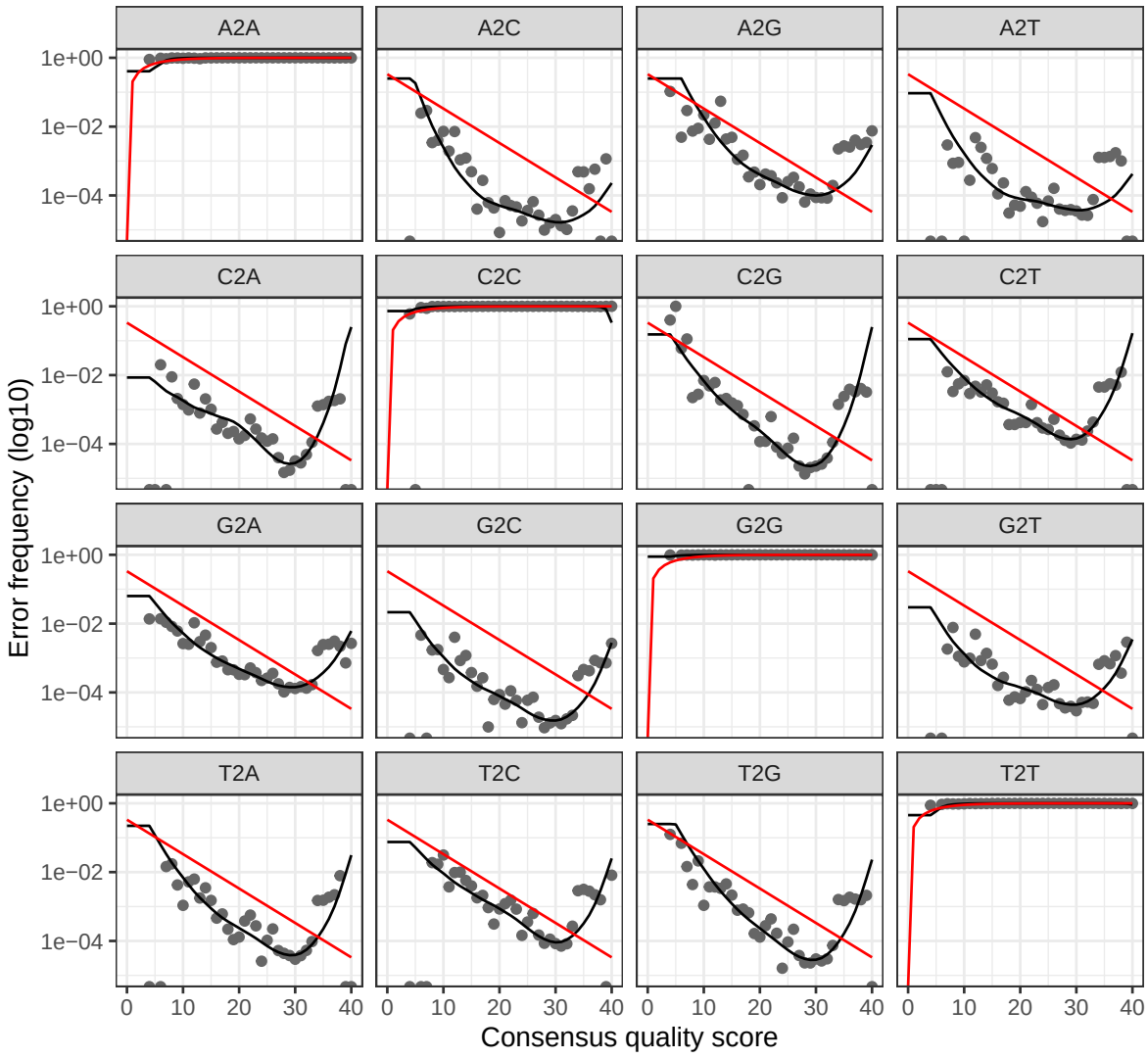
Step determinates error rate of dataset using *learnErrors* function.

```
# Forward read error rate
errF <- learnErrors(filtFs, multithread=TRUE)
# saverds
saveRDS(errF,"rds/errF.rds")
```

```
errF <- readRDS("rds/errF.rds")
```

Plot error profiles

```
# Plotting error rate profile for forward reads
plotErrors(errF, nominalQ=TRUE)
```



Denoise data

```
#denoise command
dadaFs <- dada(filtFs, err=errF, multithread=TRUE)
#save to rds file
saveRDS(dadaFs,"rds/dadaFs.rds")
```

```
#read rds
dadaFs <- readRDS("rds/dadaFs.rds")
```

Create ASV table

```
seqtab <- makeSequenceTable(dadaFs)
# Dimensions of ASV table
dim(seqtab)
```

```
[1]   18 1797
```

Remove chimeric variants

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

```
[1]   18 1558
```

Summary

Summary can help to pinpoint if at some stage, abnormal amount of the data is lost

```
getN <- function(x) sum(getUniques(x))
track <- cbind(out, sapply(dadaFs, getN), rowSums(seqtab.nochim),
               rowSums(seqtab.nochim != 0))
# If processing a single sample, replace sapply(dadaFs, getN) with getN(dadaFs)
colnames(track) <- c("Input", "Filtered", "Denoised", "Nonchimeric", "Variants")
rownames(track) <- sample.names
kable(track, caption="Denoising summary") %>%
  kable_styling(latex_options = c("HOLD_position","striped")) %>%
  row_spec(0, background="indigo", color="ivory")
```

Table 1: Denoising summary

	Input	Filtered	Denoised	Nonchimeric	Variants
Sample01	261569	137665	137385	136842	47
Sample02	262655	203573	203354	201985	59
Sample03	242547	174515	173923	172596	366
Sample04	245721	180881	180445	179564	380
Sample05	257286	147881	147568	146855	73
Sample06	267541	154404	154150	153195	57
Sample07	246034	147910	147210	144670	755
Sample08	265660	153425	152732	149897	727
Sample09	226624	186066	185870	185285	43
Sample10	216757	163175	162997	162556	36
Sample11	225419	164347	164114	163551	99
Sample12	232630	159525	159275	158958	110
Sample13	222337	155976	155772	154595	37
Sample14	232938	183999	183779	182886	42
Sample15	222467	162987	162557	162207	207
Sample16	227439	153865	153543	152411	181
Sample17	332584	138388	137843	136599	480
Sample18	403	247	228	228	8

Assign taxonomy

We use idTaxa from DECIPHER and Silva database to assign taxonomic information.

```
#Create a DNAStringSet from the ASVs
sequences <- DNAStringSet(getSequences(seqtab.nochim))
# CHANGE TO THE PATH OF YOUR TRAINING SET
load("~/feature_classifiers/SILVA_SSU_r138_2019.RData")
#IdTaxa
ids <- IdTaxa(sequences, trainingSet, strand="top", processors = 3, verbose = FALSE)
ranks <- c("domain", "phylum", "class", "order", "family", "genus", "species")
#Convert the output object of class "Taxa" to a matrix analogous to the output from assignTaxonomy
taxid <- t(sapply(ids, function(x) {
        m <- match(ranks, x$rank)
        taxa <- x$taxon[m]
        taxa[startsWith(taxa, "unclassified_")] <- NA
        taxa
}))
colnames(taxid) <- ranks; rownames(taxid) <- getSequences(seqtab.nochim)
#save end result to rds
saveRDS(taxid, "rds/taxid.rds")
```

```
taxid <- readRDS("rds/taxid.rds")
```

Create phyloseq object

```
# Reading tsv file, arranging first column to rownames and creating phyloseq object pseq
samples_meta <- read_tsv("data/metadata.tsv", show_col_types = FALSE)
samples_meta <- samples_meta %>% tibble::column_to_rownames("Sampleid")
sampletable = sample_data(samples_meta)
pseq <- phyloseq(otu_table(seqtab.nochim, taxa_are_rows = FALSE),
                 tax_table(taxid),
                 sampletable)
#Viewing basic information of pseq object
pseq
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 1558 taxa and 18 samples ]
sample_data() Sample Data:       [ 18 samples by 5 sample variables ]
tax_table()   Taxonomy Table:    [ 1558 taxa by 7 taxonomic ranks ]
```

Our variant sequences are currently stored as names. They will moved to refseq and taxa names will be replaced by more convenient format

```
repseq <- Biostrings::DNAStringSet(taxa_names(pseq))
names(repseq) <- taxa_names(pseq)
pseq <- merge_phyloseq(pseq, repseq)
taxa_names(pseq) <- paste0("ASV", seq(ntaxa(pseq)))
pseq
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 1558 taxa and 18 samples ]
sample_data() Sample Data:       [ 18 samples by 5 sample variables ]
tax_table()   Taxonomy Table:    [ 1558 taxa by 7 taxonomic ranks ]
refseq()      DNAStringSet:      [ 1558 reference sequences ]
```

Finally, minor modifications for dataset. Number of taxa lost is checked at each step

```
#We  capitalise taxonomic rank names
colnames(tax_table(pseq)) <- c("Kingdom", "Phylum", "Class",
  "Order", "Family", "Genus", "Species")
#Sample18 negative control in unnecessary as there is nothing to investigate
pseq <- subset_samples(pseq, Name != "control")
pseq
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 1558 taxa and 17 samples ]
sample_data() Sample Data:       [ 17 samples by 5 sample variables ]
tax_table()   Taxonomy Table:    [ 1558 taxa by 7 taxonomic ranks ]
refseq()      DNAStringSet:      [ 1558 reference sequences ]
```

```
# Keeping all taxa that are not unknown at Kingdom rank
pseq <- subset_taxa(pseq, Kingdom != "NA")
pseq
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 1431 taxa and 17 samples ]
sample_data() Sample Data:       [ 17 samples by 5 sample variables ]
tax_table()   Taxonomy Table:    [ 1431 taxa by 7 taxonomic ranks ]
refseq()      DNAStringSet:      [ 1431 reference sequences ]
```

```
# Keeping all that are not Chloroplastic at Order rank
pseq <- subset_taxa(pseq, Order != "Chloroplast" | is.na(Order))
pseq
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 1420 taxa and 17 samples ]
sample_data() Sample Data:       [ 17 samples by 5 sample variables ]
tax_table()   Taxonomy Table:    [ 1420 taxa by 7 taxonomic ranks ]
refseq()      DNAStringSet:      [ 1420 reference sequences ]
```

```
# Keeping all that are not Mitochondrial at Family rank
pseq <- subset_taxa(pseq, Family != "Mitochondria" | is.na(Family))
pseq
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 1417 taxa and 17 samples ]
sample_data() Sample Data:       [ 17 samples by 5 sample variables ]
tax_table()   Taxonomy Table:    [ 1417 taxa by 7 taxonomic ranks ]
refseq()      DNAStringSet:      [ 1417 reference sequences ]
```

In the end we have 17 samples and 1417 taxa

Writing data

Last step is to save data to suitable file formats.

All variant sequences are save to fasta

```
pseq %>% refseq() %>% writeXStringSet(paste0(exportloc,"/repseq.fasta"), append=FALSE,
                                      compress=FALSE, format="fasta")
```

Taxonomy table is converted to dataframe and written as tsv

```
taxonomy <- as.data.frame(tax_table(pseq))
write_tsv(taxonomy, paste0(exportloc, "/taxonomy.tsv"))
```

For metadata we add sampleid colum and write as tsv

```
sampleid <- sample_names(pseq)
metafile <- sample_data(pseq)
metadf <- data.frame(sampleid,metafile)
write_tsv(metadf, paste0(exportloc, "/metadata.tsv"))
```

ASV count data need to be transposed prior writing

```
ASV_names <- taxa_names(pseq)
ASV_counts <- t(otu_table(pseq))
ASVdf <- (data.frame(ASV_names,ASV_counts))
write_tsv(ASVdf, paste0(exportloc, "/fishery_asvs.tsv"))
```