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)

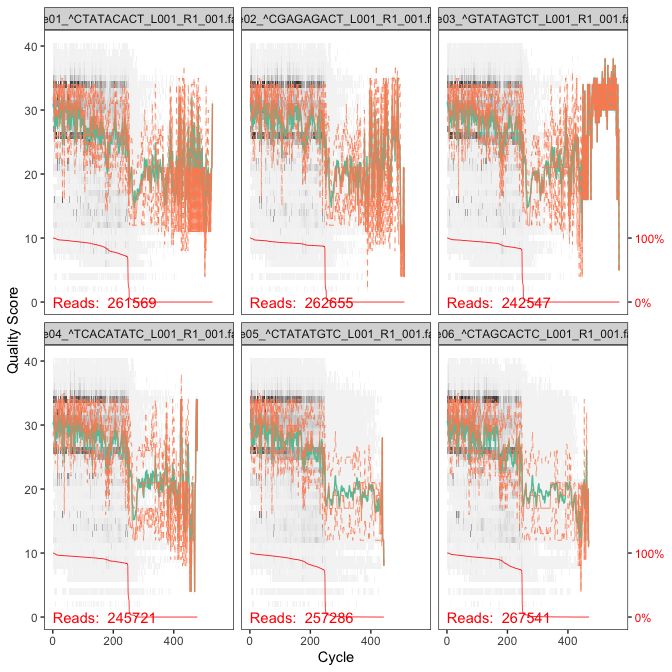
[1] "Sample01\_^CTATACACT\_L001\_R1\_001.fastq.gz"  
 [2] "Sample02\_^CGAGAGACT\_L001\_R1\_001.fastq.gz"  
 [3] "Sample03\_^GTATAGTCT\_L001\_R1\_001.fastq.gz"  
 [4] "Sample04\_^TCACATATC\_L001\_R1\_001.fastq.gz"  
 [5] "Sample05\_^CTATATGTC\_L001\_R1\_001.fastq.gz"  
 [6] "Sample06\_^CTAGCACTC\_L001\_R1\_001.fastq.gz"  
 [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

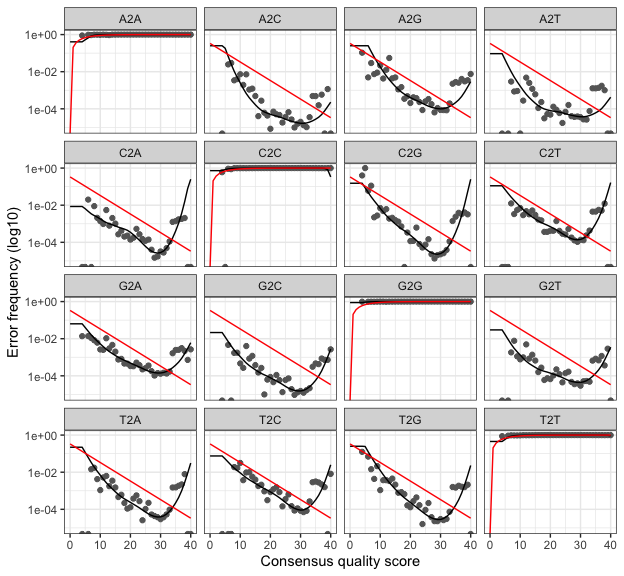
Step determinates error rate of dataset using *learnErrors* function.

# 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, remove the sapply calls: e.g. 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")

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"))