Processing short nanopore reads with dada2

Marko Suokas

This document utilises previously computed objects that are saved as rds files. Result objects are loaded from files. Original code can be executed by changing eval to TRUE.

#### Preprocess Ion Torrent adapter reads

*Trim forward reads with Adapter A and trP1(rc) sequences*

cutadapt -g “CCATCTCATCCCTGCGTGTCTCCGACTCAG;o=30…ATCACCGACTGCCCATAGAGAGG;o=23” –trimmed-only -e 0.05 -o ev\_forward.fastq.gz ev\_reads\_hq.fastq.gz

*Trim reverse reads with trP1 and Adapter A(rc) sequences*

cutadapt -g ”CCTCTCTATGGGCAGTCGGTGAT;o=23…CTGAGTCGGAGACACGCAGGGATGAGATGG;o=30” –trimmed-only -e 0.05 -o ev\_reverse.fastq.gz ev\_reads\_hq.fastq.gz

*Reverse-complement reverse reads*

seqkit seq -rp -t DNA -o ev\_rcomp.fasta.gz ev\_reverce.fasta.gz

*Merge with forward reads*

cat ev\_forward.fasta.gz ev\_rcomp.fasta.gz > raw\_005.fasta.gz

*Import data to qiime*

qiime tools import –type MultiplexedSingleEndBarcodeInSequence –input-path raw\_005.fasta.gz –output-path raw\_005.qza

*Demultiplex*

qiime cutadapt demux-single –i-seqs raw\_005.qza –m-barcodes-file jt\_meta.tsv –m-barcodes-column Barcode\_seq –output-dir demuxed –p-error-rate 0 –p-anchor-barcode

*Trim pcr primers (519F and 926R)*

qiime cutadapt trim-single –i-demultiplexed-sequences per\_sample\_sequences.qza –p-overlap 15 –p-discard-untrimmed –p-front ACAGCMGCCGCGGTAATWC –o-trimmed-sequences trim1.qza

qiime cutadapt trim-single –i-demultiplexed-sequences trim1.qza –p-adapter AAACTCAAAKGAATTGACGG –o-trimmed-sequences trimmed-sequences.qza

*Decompress read files*

unzip trimmed-sequences.qza

**Note.** Parameters allow one error in sequencing adapters, no errors in barcode sequence and 1 and 2 errors in pcr primers, respectively.

**Note.** Some options in commands require double dash and are not displayed correctly in rendered documents.

#### Load libraries

library(dada2);packageVersion("dada2")

[1] '1.30.0'

library(knitr);packageVersion("knitr")

[1] '1.45'

library(Biostrings);packageVersion("Biostrings")

[1] '2.70.2'

library(DECIPHER);packageVersion("DECIPHER")

[1] '2.30.0'

library(phyloseq);packageVersion("phyloseq")

[1] '1.46.0'

library(tidyverse);packageVersion("tidyverse")

[1] '2.0.0'

library(kableExtra);packageVersion("kableExtra")

[1] '1.4.0'

library(mia);packageVersion("mia")

[1] '1.10.0'

#### Set parameters

# Path variables  
path <- "data/reads/"  
training <- "~/feature\_classifiers/SILVA\_SSU\_r138\_2019.RData"  
meta\_file <- "data/jt\_meta.tsv"  
exportloc <- "results/"  
# Variables: truncation length, phix (Illumina)  
truncation <- 350  
#Creates results directory  
dir.create(exportloc)  
#metadata  
metadata <- data.frame(read\_tsv(meta\_file))  
#set knitr cache path  
knitr::opts\_chunk$set(cache.path = "cache/")

#### Import reads

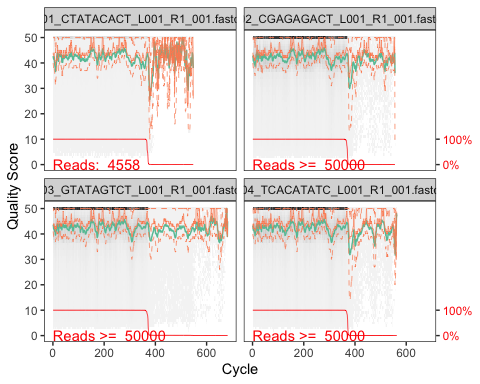
nr072 was removed from dataset (0 reads causing error in denoising step).

# Forward fastq filenames have format: SAMPLENAME\_R1\_001.fastq  
fnFs <- sort(list.files(path, pattern="L001\_R1\_001.fastq.gz", full.names = TRUE))  
# Extract sample names, assuming filenames have format: SAMPLENAME\_XXX.fastq.gz  
sample.names <- sapply(strsplit(basename(fnFs), "\_"), `[`, 1)

Checking read quality of first samples

# Base quality plot  
p <- plotQualityProfile(fnFs[1:4], n = 50000)  
p

# Load base quality plot from saved object  
p <- readRDS("rds/qplot.rds")  
p



#### Filter and trim reads

# Filtered files are placed in filtered subdirectory  
filtFs <- file.path(path, "filtered", paste0(sample.names,  
 "\_F\_filt.fastq.gz"))  
# For single end data sets without phix control  
names(filtFs) <- sample.names  
out <- filterAndTrim(fnFs, filtFs, truncLen=truncation,  
 maxN = 0, maxEE = 2 , truncQ = 2,  
 compress = TRUE, multithread = FALSE,  
 rm.phix = FALSE)  
saveRDS(out, file = "rds/out.rds")

# Filtered files are placed in filtered subdirectory  
filtFs <- file.path(path, "filtered", paste0(sample.names,  
 "\_F\_filt.fastq.gz"))  
# For single end data sets without phix control  
names(filtFs) <- sample.names  
# Load previously saved object  
out <-readRDS("rds/out.rds")

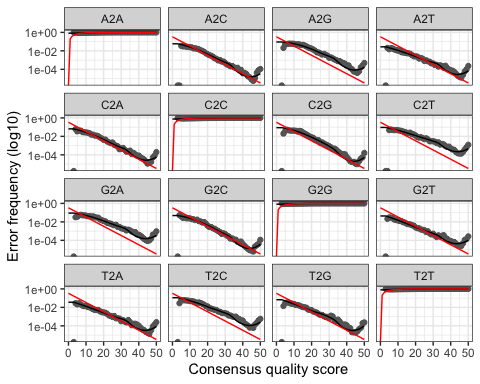
#### Learn and plot error profile

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

# Load previously saved object  
errF <- readRDS("rds/errF.rds")

Plot error profile

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



#### Denoise sequences

dadaFs <- dada(filtFs, err = errF, multithread = TRUE, verbose = FALSE)  
saveRDS(dadaFs, file = "rds/dadaFs.rds")

# Load previously saved object  
dadaFs <- readRDS("rds/dadaFs.rds")

#### Build ASV table

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

[1] 131 8404

Remove chimeric variants

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

[1] 131 7634

Amount of data remaining after chimera removal

sum(seqtab.nochim)/sum(seqtab)

[1] 0.9959133

#### Summary table

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  
colnames(track) <- c("Input", "Filtered", "DenoisedF", "Nonchimeric",  
 "N:o of variants")  
rownames(track) <- sample.names  
kable(track, caption="Summary table", booktabs = TRUE, longtable = TRUE) %>%  
 kable\_styling(latex\_options=c("striped", "HOLD\_position", "repeat\_header")) %>%  
 row\_spec(0,background = "teal", color = "ivory")

Summary table

|  | Input | Filtered | DenoisedF | Nonchimeric | N:o of variants |
| --- | --- | --- | --- | --- | --- |
| nr001 | 4558 | 3737 | 3589 | 3589 | 96 |
| nr002 | 101469 | 82272 | 82017 | 82017 | 449 |
| nr003 | 139054 | 113872 | 113654 | 113612 | 359 |
| nr004 | 256204 | 207763 | 207374 | 207160 | 361 |
| nr005 | 104757 | 85575 | 85340 | 85153 | 413 |
| nr006 | 125276 | 101867 | 101438 | 101135 | 407 |
| nr007 | 16776 | 13466 | 13352 | 13352 | 179 |
| nr008 | 60779 | 50372 | 50261 | 50261 | 259 |
| nr009 | 57496 | 46090 | 45844 | 45777 | 250 |
| nr010 | 58884 | 48417 | 48302 | 48287 | 239 |
| nr011 | 57804 | 46828 | 46648 | 46648 | 315 |
| nr012 | 63184 | 51624 | 51355 | 51279 | 291 |
| nr013 | 184184 | 151003 | 150506 | 150337 | 305 |
| nr014 | 37034 | 30255 | 30059 | 30059 | 167 |
| nr015 | 52921 | 42498 | 42295 | 42214 | 256 |
| nr016 | 57780 | 47377 | 47221 | 47198 | 184 |
| nr017 | 32249 | 27015 | 26761 | 26761 | 223 |
| nr018 | 32343 | 26457 | 26235 | 26204 | 238 |
| nr019 | 211 | 182 | 152 | 152 | 8 |
| nr020 | 17466 | 14548 | 14312 | 14309 | 271 |
| nr021 | 13454 | 10843 | 10407 | 10368 | 298 |
| nr022 | 36958 | 30203 | 29950 | 29916 | 330 |
| nr023 | 5863 | 4939 | 4760 | 4760 | 169 |
| nr024 | 39062 | 32944 | 32682 | 32675 | 287 |
| nr025 | 18358 | 15039 | 14839 | 14839 | 234 |
| nr026 | 54181 | 45020 | 44705 | 44689 | 461 |
| nr027 | 57462 | 46962 | 46734 | 46734 | 281 |
| nr028 | 6385 | 5190 | 4976 | 4976 | 135 |
| nr029 | 1149 | 915 | 809 | 809 | 57 |
| nr030 | 2978 | 2463 | 2280 | 2280 | 72 |
| nr031 | 55110 | 44610 | 44399 | 44191 | 303 |
| nr032 | 53814 | 42979 | 42625 | 42284 | 219 |
| nr033 | 41001 | 32923 | 32685 | 32664 | 287 |
| nr034 | 25124 | 20580 | 20298 | 20296 | 288 |
| nr035 | 1028 | 835 | 761 | 761 | 34 |
| nr036 | 55738 | 45478 | 45334 | 45302 | 332 |
| nr037 | 52126 | 42022 | 41630 | 41550 | 325 |
| nr038 | 63365 | 52054 | 51762 | 51391 | 404 |
| nr039 | 62495 | 50482 | 50215 | 50090 | 428 |
| nr040 | 66486 | 54623 | 54394 | 54325 | 364 |
| nr041 | 50049 | 41199 | 41048 | 41028 | 288 |
| nr042 | 56406 | 47601 | 47412 | 47302 | 201 |
| nr043 | 62973 | 52026 | 51849 | 51794 | 248 |
| nr044 | 51860 | 42460 | 42297 | 42212 | 259 |
| nr045 | 62931 | 51359 | 50996 | 50953 | 410 |
| nr046 | 57953 | 47598 | 47331 | 46941 | 259 |
| nr047 | 59138 | 49654 | 49488 | 49442 | 260 |
| nr048 | 55998 | 46407 | 46245 | 46231 | 284 |
| nr049 | 59015 | 49029 | 48942 | 48940 | 252 |
| nr050 | 58770 | 48128 | 47929 | 47881 | 367 |
| nr051 | 58199 | 48129 | 48011 | 47938 | 241 |
| nr052 | 59722 | 48648 | 48421 | 48387 | 456 |
| nr053 | 52295 | 42909 | 42812 | 42810 | 206 |
| nr054 | 54222 | 44000 | 43814 | 43761 | 255 |
| nr055 | 44896 | 36482 | 36250 | 36211 | 327 |
| nr056 | 56564 | 46213 | 46048 | 46037 | 340 |
| nr057 | 57220 | 44611 | 44384 | 44384 | 340 |
| nr058 | 29071 | 21680 | 21628 | 21628 | 101 |
| nr059 | 49673 | 37678 | 37585 | 37585 | 184 |
| nr060 | 56441 | 44838 | 44765 | 44720 | 204 |
| nr061 | 39464 | 31746 | 31592 | 31558 | 203 |
| nr062 | 57638 | 47417 | 47311 | 47311 | 121 |
| nr063 | 41622 | 34042 | 33991 | 33847 | 116 |
| nr064 | 57665 | 45751 | 45672 | 45672 | 151 |
| nr065 | 31057 | 24009 | 23923 | 23923 | 115 |
| nr066 | 46974 | 36323 | 36192 | 36192 | 160 |
| nr067 | 28068 | 23098 | 22969 | 22967 | 116 |
| nr068 | 29823 | 24238 | 24203 | 24203 | 86 |
| nr069 | 30168 | 24183 | 24129 | 24127 | 128 |
| nr070 | 60493 | 50010 | 49839 | 49598 | 200 |
| nr071 | 53242 | 41175 | 41058 | 40969 | 179 |
| nr073 | 49981 | 40478 | 40360 | 40357 | 179 |
| nr074 | 57395 | 46109 | 46003 | 46003 | 153 |
| nr075 | 56378 | 46525 | 46362 | 46362 | 181 |
| nr076 | 50105 | 39042 | 38965 | 38965 | 198 |
| nr077 | 53839 | 45055 | 44930 | 44903 | 272 |
| nr078 | 49264 | 40522 | 40432 | 40428 | 324 |
| nr079 | 1023 | 865 | 775 | 775 | 25 |
| nr080 | 76644 | 64132 | 63928 | 63928 | 348 |
| nr081 | 77967 | 62167 | 61521 | 61008 | 1391 |
| nr082 | 39055 | 32857 | 32731 | 32731 | 161 |
| nr083 | 33607 | 27830 | 27742 | 27719 | 227 |
| nr084 | 6891 | 5655 | 5515 | 5515 | 88 |
| nr085 | 63220 | 52028 | 51855 | 51853 | 218 |
| nr086 | 28118 | 21641 | 21539 | 21515 | 165 |
| nr087 | 1547 | 1240 | 1214 | 1214 | 44 |
| nr088 | 355 | 291 | 248 | 248 | 19 |
| nr089 | 94933 | 75244 | 74715 | 73698 | 62 |
| nr090 | 49193 | 39043 | 38697 | 38660 | 56 |
| nr091 | 43492 | 34421 | 34160 | 34137 | 35 |
| nr092 | 87487 | 69403 | 69153 | 69153 | 64 |
| nr093 | 170090 | 138581 | 137950 | 137467 | 90 |
| nr094 | 183908 | 136075 | 135427 | 135250 | 92 |
| nr095 | 187778 | 149971 | 149441 | 149364 | 86 |
| nr096 | 152388 | 120640 | 120247 | 120041 | 86 |
| nr097 | 151106 | 122605 | 122173 | 122018 | 69 |
| nr098 | 61905 | 49312 | 49062 | 49001 | 50 |
| nr099 | 6228 | 5126 | 5003 | 5003 | 32 |
| nr100 | 181 | 137 | 91 | 91 | 11 |
| nr101 | 118 | 86 | 47 | 47 | 7 |
| nr102 | 65 | 48 | 23 | 23 | 4 |
| nr103 | 196 | 145 | 78 | 78 | 9 |
| nr104 | 198 | 153 | 100 | 100 | 13 |
| nr105 | 179 | 145 | 82 | 82 | 11 |
| nr106 | 107 | 83 | 42 | 42 | 5 |
| nr107 | 176 | 144 | 68 | 68 | 7 |
| nr108 | 92 | 71 | 29 | 29 | 5 |
| nr109 | 139 | 113 | 67 | 67 | 9 |
| nr110 | 447 | 360 | 269 | 269 | 32 |
| nr111 | 122 | 96 | 52 | 52 | 4 |
| nr112 | 212 | 174 | 114 | 114 | 11 |
| nr113 | 144 | 116 | 74 | 74 | 5 |
| nr114 | 127 | 103 | 57 | 57 | 7 |
| nr115 | 502 | 393 | 378 | 378 | 11 |
| nr116 | 73 | 52 | 16 | 16 | 2 |
| nr117 | 127 | 103 | 46 | 46 | 6 |
| nr118 | 120 | 94 | 47 | 47 | 5 |
| nr119 | 170311 | 133480 | 132659 | 129636 | 87 |
| nr120 | 157 | 126 | 64 | 64 | 8 |
| nr121 | 161 | 131 | 72 | 72 | 11 |
| nr122 | 199 | 161 | 117 | 117 | 14 |
| nr123 | 60 | 48 | 14 | 14 | 2 |
| nr124 | 100 | 83 | 47 | 47 | 10 |
| nr125 | 84 | 66 | 35 | 35 | 6 |
| nr126 | 81 | 63 | 25 | 25 | 2 |
| nr127 | 208327 | 163285 | 162177 | 159439 | 112 |
| nr128 | 72407 | 59091 | 58754 | 58652 | 56 |
| nr129 | 45752 | 37244 | 37008 | 36922 | 44 |
| nr130 | 192968 | 154678 | 153444 | 145777 | 122 |
| nr131 | 19 | 17 | 13 | 13 | 2 |
| nr132 | 14 | 11 | 1 | 1 | 1 |

#### Assign taxonomy

#Create a DNAStringSet from the ASV sequences  
repseq <- DNAStringSet(getSequences(seqtab.nochim))  
# CHANGE TO THE PATH OF YOUR TRAINING SET  
load(training)  
ids <- IdTaxa(repseq, trainingSet, strand = "top",  
 processors = 3, verbose = FALSE,  
 threshold = 50)  
ranks <- c("domain", "phylum", "class", "order", "family",  
 "genus", "species")   
# Convert the output 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)  
saveRDS(taxid, file = "rds/taxid.rds")

# Load previously saved object  
taxid <- readRDS("rds/taxid.rds")

#### Build phyloseq object

pseq <- phyloseq(otu\_table(seqtab.nochim, taxa\_are\_rows = FALSE),  
 tax\_table(taxid))  
row.names(metadata) <- sample\_names(pseq)  
sample\_data(pseq) <- metadata  
pseq

phyloseq-class experiment-level object  
otu\_table() OTU Table: [ 7634 taxa and 131 samples ]  
sample\_data() Sample Data: [ 131 samples by 5 sample variables ]  
tax\_table() Taxonomy Table: [ 7634 taxa by 7 taxonomic ranks ]

#create sequences from rownames to refseq   
seqs <- DNAStringSet(taxa\_names(pseq))  
names(seqs) <- taxa\_names(pseq)  
pseq <- merge\_phyloseq(pseq, seqs)  
#new variant names  
taxa\_names(pseq) <- paste0("ASV", seq(ntaxa(pseq)))  
#capitalise taxonomic ranks  
colnames(tax\_table(pseq)) <- c("Kingdom", "Phylum", "Class",   
 "Order", "Family", "Genus", "Species")

Remove non-bacterial variants

#remove non-bacterial taxa  
pseq <- subset\_taxa(pseq, Kingdom != is.na(Kingdom))  
pseq <- subset\_taxa(pseq, Kingdom != "Eukaryota")  
pseq

phyloseq-class experiment-level object  
otu\_table() OTU Table: [ 7280 taxa and 131 samples ]  
sample\_data() Sample Data: [ 131 samples by 5 sample variables ]  
tax\_table() Taxonomy Table: [ 7280 taxa by 7 taxonomic ranks ]  
refseq() DNAStringSet: [ 7280 reference sequences ]

#### Write results to files

Abundance table is transponed and written as tsv file

#variant names in rows  
ASV\_names <- taxa\_names(pseq)  
#sample names will be columns  
ASV\_counts <- t(otu\_table(pseq))  
ASVdf <- (data.frame(ASV\_names,ASV\_counts))  
#write  
write\_tsv(ASVdf, paste0(exportloc,"asvs.tsv"))

Likewise taxonomy table is saved as tsv

#variant names in rows  
ASV\_names <- taxa\_names(pseq)  
#taxonomy ranks in columns  
taxonomy <- (data.frame(ASV\_names, tax\_table(pseq)))  
#write  
write\_tsv(taxonomy,paste0(exportloc,"taxonomy.tsv"))

Variant sequences are saved into fasta file

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

Compatible metadata file as tsv

sampleid <- sample\_names(pseq)  
metafile <- sample\_data(pseq)  
metadf <- data.frame(sampleid,metafile)  
write\_tsv(metadf, paste0(exportloc,"metadata.tsv"))

#### Observations

Customised sup basecalling of nanopore sequences produce quality matching Illumina ja Ion Torrent

Error profiles of short amplicon (truncated to 350 bp) follow expected frequency

Proportion of unique reads is smaller when compared to long amplicons

Thus, denoising seem to work normally on shorter read lengths in contrast to 1,5 kbp full-length 16S rRNA gene. This is expected as algorithm relies on error-free reads that are used to build variant clusters. In long reads, even 99,5 % accuracy is simply not enough. At 1400 bp 0,5 % mean error rate means 7 sequencing errors per read

#### Advantages of nanopore

Read length is not limiting factor while designing amplicon targets

Base quality doesn’t decrease as a function of read length

Low diversity libraries are not problem in sequencing

Libraries prepared for other platforms can be conveniently converted to nanopore

Live basecalling allows controlling sequencing throughput and in some cases flow-cell can be reused

Cost per bp in amplicon sequencing is great compared to MiSeq

#### Disadvantages of nanopore

Homopolymer region accuracy is not quite as good as in Illumina

High accuracy basecalling is computationally intensive

Software tools are not quite at the same level as in other platforms and require more knowledge

Consistency of flow-cells (number of functional pores) and repeatibility of sequencing is so far unclear

Pores might die if library preparation contains contaminants originating from sample (concerns mainly genomic or transcriptomic sequencing)