Processing nanopore reads

Marko Suokas

Preprocess reads

Dorado doesn't support demultiplexing of dual indexes on 5' and 3' ends. Additionally, ligated library reads can be either orientation. Most straightforward approach to demultiplex reads, is to utilise cutadapt. You can demultiplex index pairs using liked adapters approach in forward and reverse orientation, then process with scripts to reverse complement reverse read files and merge each of thme with forward reads.

Extracting forward reads to fastq file can be performed with following command

cutadapt -e 0 -O 12 -g file:~/scripts/barcodes.fasta–trimmed-only -m 1200 -o "fdemuxed/{name}.fastq.gz reads.fastq.gz

Command will extract barcodes defined in barcodes.fasta file and output matches into individual files in fdemuxed subdirectory. Minimum length is sen in example to 1200 bp.

Extracting reverse reads using reverse complemented barcodes.fasta file

cutadapt -e 0 -O 12 -g file:~/scripts/rev_barcodes.fasta–trimmed-only -m 1200 -o "rdemuxed/{name}.fastq.gz reads.fastq.gz

Reads are demultiplexed into separate directory

Tip! O, e, m and M parameters can be used to reduce chances of misaligned matches

Next we use bash script that will process each reverse read file and reverse complement them using basic command

seqkit seq -rp –seq-type DNA -o reverse_comp.fastq.gz reverse_out.fastq.gz

Final step is merging. You can use also for this simple bash script that merge files with same base name from two separate directories using basic command

zcat forward_out.fastq.gz reverse_comp.fastq.gz >merged_reads.fastq.gz

Trimming PCR amplification primers

Finally, you can use cutadapt and bash scripts to trim forward and reverse PCR primers from demultiplexed sequence files.

Import set2

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(mia);packageVersion("mia")
```

```
[1] '1.12.0'
```

```
library(qiime2R);packageVersion("qiime2R")
```

```
[1] '0.99.6'
```

Set variables

```
# Path variables
path <- "data/processed/set2"
training <- "~/feature_classifiers/SILVA_SSU_r138_2019.RData"
meta_file <- "data/set2_meta.tsv"
exportloc <- "results_set2/"
# Variables: truncation length, phix (Illumina)
truncation <- 1400
#Creates results directory
dir.create(exportloc)
#metadata
metadata <- data.frame(read_tsv(meta_file))
```

For the project, we take advantage of computing power of CSC puhti server and import already executed data objects from there. R code is unaltered and can can be executed by turning eval to TRUE.

```
#List files inside directory
list.files(path)
```

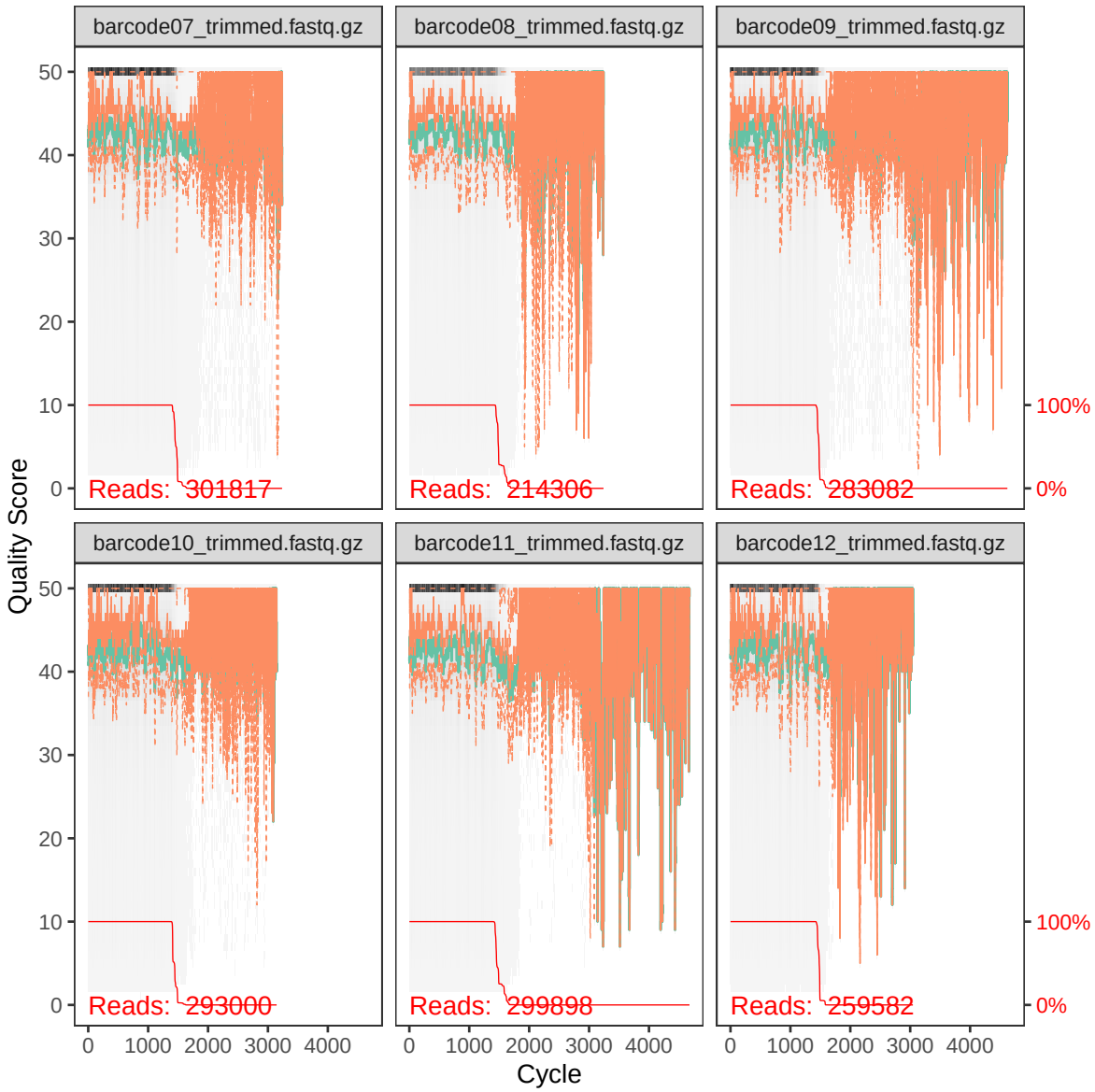
```
[1] "barcode07_trimmed.fastq.gz" "barcode08_trimmed.fastq.gz"
[3] "barcode09_trimmed.fastq.gz" "barcode10_trimmed.fastq.gz"
[5] "barcode11_trimmed.fastq.gz" "barcode12_trimmed.fastq.gz"
[7] "filtered"
```

```
# Forward fastq filenames have format: SAMPLENAME_R1_001.fastq
fnFs <- sort(list.files(path, pattern="_trimmed_all.fastq.gz", full.names = TRUE))
# Extract sample names, assuming filenames have format: SAMPLENAME_XXX.fastq
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
```

Plot sequence quality profile for samples

```
# Base quality plot
prsetII <- plotQualityProfile(fnFs[1:6])
prsetII
```

```
prsetII <- readRDS("rds/set2_rds/prset2.rds")
prsetII
```



Filter sequence data

Filtering reads (maxEE ≈ 1 error/200 bp sequence is good starting point for this amplicon)

```
# 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 = 7 , truncQ = 2,
                     compress = TRUE, multithread = TRUE,
                     rm.phix = FALSE)
```

```
out <- readRDS("rds/set2_rds/out2.rds")
```

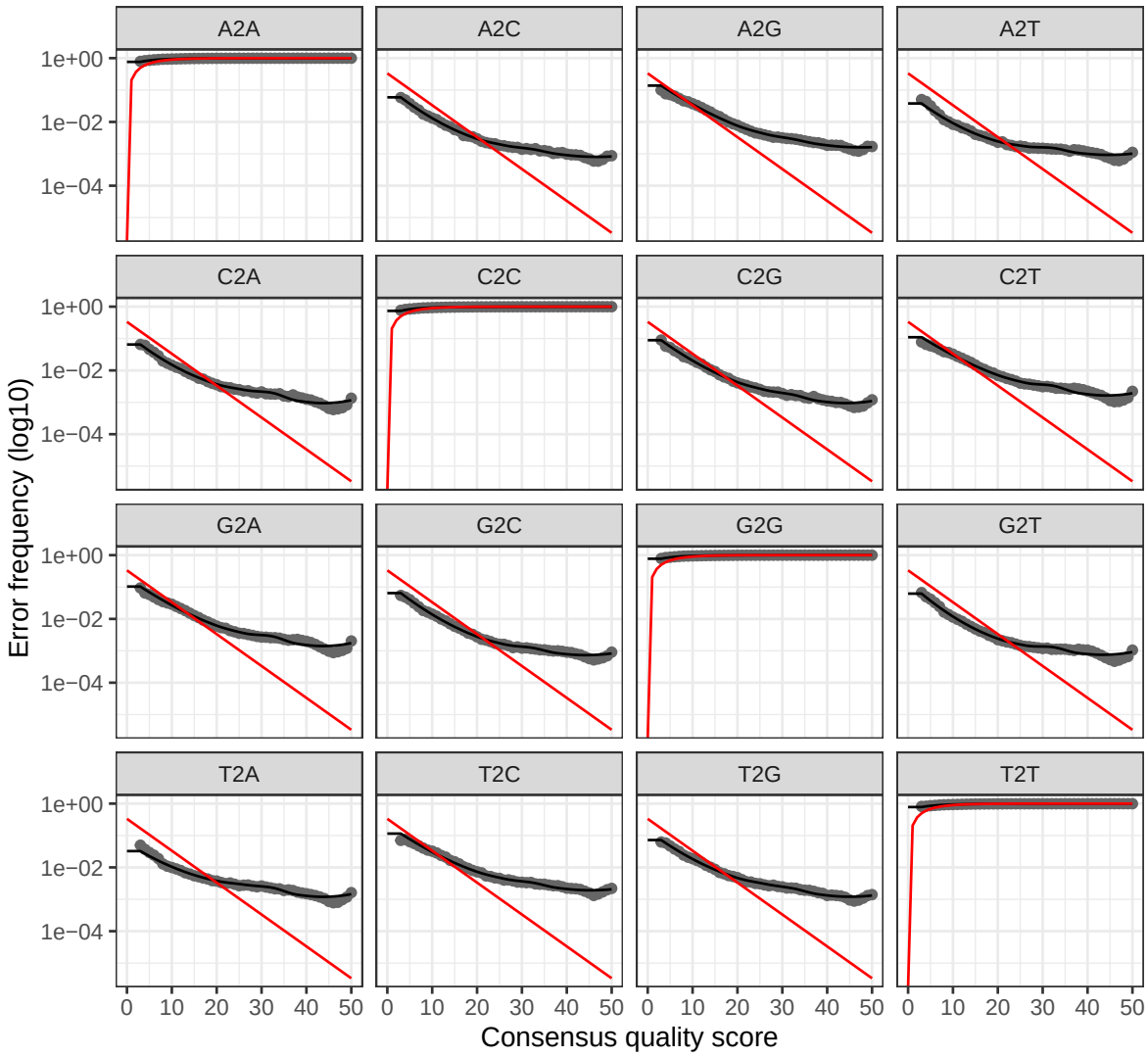
Learn error rates

```
# Forward read error rate
errF <- learnErrors(filtFs, multithread = TRUE)
```

```
errF <- readRDS("rds/set2_rds/errF2.rds")
```

Plott error rates

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



Denoise

```
dadaFs <- dada(filtFs, err = errF, multithread = TRUE)
```

```
dadaFs <- readRDS("rds/set2_rds/dadaFs2.rds")
```

Build asv table

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

```
[1]    6 1566
```

Chimera removal

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

```
[1]   6 933
```

Summary

```
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) <- metadata$Name
kable(track, caption="Summary table")  %>%
  kable_styling(latex_options=c("striped", "HOLD_position")) %>%
                row_spec(0,background = "teal", color = "ivory")
```

Table 1: Summary table

	Input	Filtered	DenoisedF	Nonchimeric	N:o of variants
A5_3	301817	229567	211982	199348	190
A6_3	214306	164005	144520	124792	194
A7_3	283082	214821	208768	202384	145
E5_3	293000	224099	218197	206612	129
E6_3	299898	228111	209010	188314	189
E7_3	259582	201157	176699	151760	175

Taxonomy assignment

IdTaxa from DECIPHER package

```
#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)
```

```
taxid <- readRDS("rds/set2_rds/taxid2.rds")
```

Create 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:         [ 933 taxa and 6 samples ]
sample_data() Sample Data:       [ 6 samples by 3 sample variables ]
tax_table()   Taxonomy Table:    [ 933 taxa by 7 taxonomic ranks ]
```

Sequence data is stored as taxa_names. We will store sequences as refseq and create numbered variant names

```
#create DNA object and store sequences
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")
```

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")
```

Vsearch@97%

Data has been processed in qiime, except feature classification

```
#read qiime2 table
vs97 <- readQZA("data/vsearch2/97/filtered-table.qza")
#reorder samples
vs97 <- vs97[, order(colnames(vs97))]
#read taxonomy
otu_taxonomy <- readRDS("data/vsearch2/97/taxid.rds")
pseq_vsearch97 <- phyloseq(otu_table(vs97, taxa_are_rows = TRUE),
                  tax_table(otu_taxonomy))
#change names
taxa_names(pseq_vsearch97) <- paste0("ASV", seq(ntaxa(pseq_vsearch97)))
#capitalise taxonomic ranks
colnames(tax_table(pseq_vsearch97)) <- c("Kingdom", "Phylum", "Class",
  "Order", "Family", "Genus", "Species")
#create DNA object and store sequences
seqs <- readDNAStringSet("data/vsearch2/97/dna-sequences.fasta")
names(seqs) <- taxa_names(pseq_vsearch97)
pseq_vsearch97 <- merge_phyloseq(pseq_vsearch97, seqs)
```

Write vsearch97 after data wrangling

```
#create_directory
results <- "results_vsearch97/"
dir.create(results)
#variant_table
ASV_names <- taxa_names(pseq_vsearch97)
ASV_counts <- otu_table(pseq_vsearch97)
ASVdf <- (data.frame(ASV_names,ASV_counts))
write_tsv(ASVdf, paste0(results,"asvs_set2.tsv"))
#taxonomy
ASV_names <- taxa_names(pseq_vsearch97)
taxonomy <- (data.frame(ASV_names, tax_table(pseq_vsearch97)))
write_tsv(taxonomy,paste0(results,"taxonomy_set2.tsv"))
#sequences
refseq(pseq_vsearch97) %>% writeXStringSet(paste0(results,"repseq_set2.fasta"),
                                    append = FALSE, compress = FALSE,
                                    format = "fasta")
```

Vsearch@99%

```
#read qiime2 table
vs99 <- readQZA("data/vsearch2/99/filtered-table.qza")
#reorder samples
vs99 <- vs99[, order(colnames(vs99))]
#read taxonomy
otu_taxonomy <- readRDS("data/vsearch2/99/taxid.rds")
pseq_vsearch99 <- phyloseq(otu_table(vs99, taxa_are_rows = TRUE),
                  tax_table(otu_taxonomy))
#change names
taxa_names(pseq_vsearch99) <- paste0("ASV", seq(ntaxa(pseq_vsearch99)))
#capitalise taxonomic ranks
colnames(tax_table(pseq_vsearch99)) <- c("Kingdom", "Phylum", "Class",
  "Order", "Family", "Genus", "Species")
#create DNA object and store sequences
seqs <- readDNAStringSet("data/vsearch2/99/dna-sequences.fasta")
names(seqs) <- taxa_names(pseq_vsearch99)
pseq_vsearch99 <- merge_phyloseq(pseq_vsearch99, seqs)
```

Write vsearch99 files after data wrangling

```
#create_directory
results <- "results_vsearch99/"
dir.create(results)
#variant_table
ASV_names <- taxa_names(pseq_vsearch99)
ASV_counts <- otu_table(pseq_vsearch99)
ASVdf <- (data.frame(ASV_names,ASV_counts))
write_tsv(ASVdf, paste0(results,"asvs_set2.tsv"))
#taxonomy
ASV_names <- taxa_names(pseq_vsearch99)
taxonomy <- (data.frame(ASV_names, tax_table(pseq_vsearch99)))
write_tsv(taxonomy,paste0(results,"taxonomy_set2.tsv"))
#sequences
refseq(pseq_vsearch99) %>% writeXStringSet(paste0(results,"repseq_set2.fasta"),
                                           append = FALSE, compress = FALSE,
                                           format = "fasta")
```

Observations

With larger number of variants and 1,4 kbp sequence length, error rate profiles start slightly to deviate from expected. It doesn't seem more than we usually see with short-read sequencers. Even though, variant numbers seem realistic, it's unclear if denoising is best choice for data.

From same samples, vsearch produced >2500 and >18000 variants, respectively. Note that in this set, sequences were truncated to 1400 prior dereplication and clustering.