Processing and studying nanopore reads

Marko Suokas

Preprocess reads

Dorado doesn't support demultiplexing of dual indexes on both 5' and 3' ends. Additionally, in ligated libraries reads can be either orientation. Our approach to demultiplex reads is using cutadapt. Index pairs are searched using linked adapters approach in forward and reverse orientation, then scripts are used to reverse complement reverse reads. Finally reads are merged

Please note that autocorrect often change double dashes in command examples.

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 set 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 merge. You can use simple bash script that will 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
```

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

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

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

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

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

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

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

```
library(ape);packageVersion("ape")
```

```
[1] '5.8'
```

Set variables

```
# Path variables
path <- "data/processed/set2"
silva <- "~/feature_classifiers/silva_nr99_v138.1_train_set.fa.gz"
species <- "~/feature_classifiers/silva_species_assignment_v138.1.fa.gz"
meta_file <- "data/set2_meta.tsv"
exportloc <- "set2/"
# Variables: truncation length, phix (Illumina)
truncation <- 1400
#Creates results directory
dir.create(exportloc)
#metadata
metadata <- data.frame(read_tsv(meta_file, show_col_types = F))
metadata <- column_to_rownames(metadata, "Sampleid")
```

For project, we took advantage of computing power of CSC and imported already executed data objects. R code is unaltered. Execution is controlled by eval parameter in code chunk. RDS files also save resources and time when document is edited and checked.

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

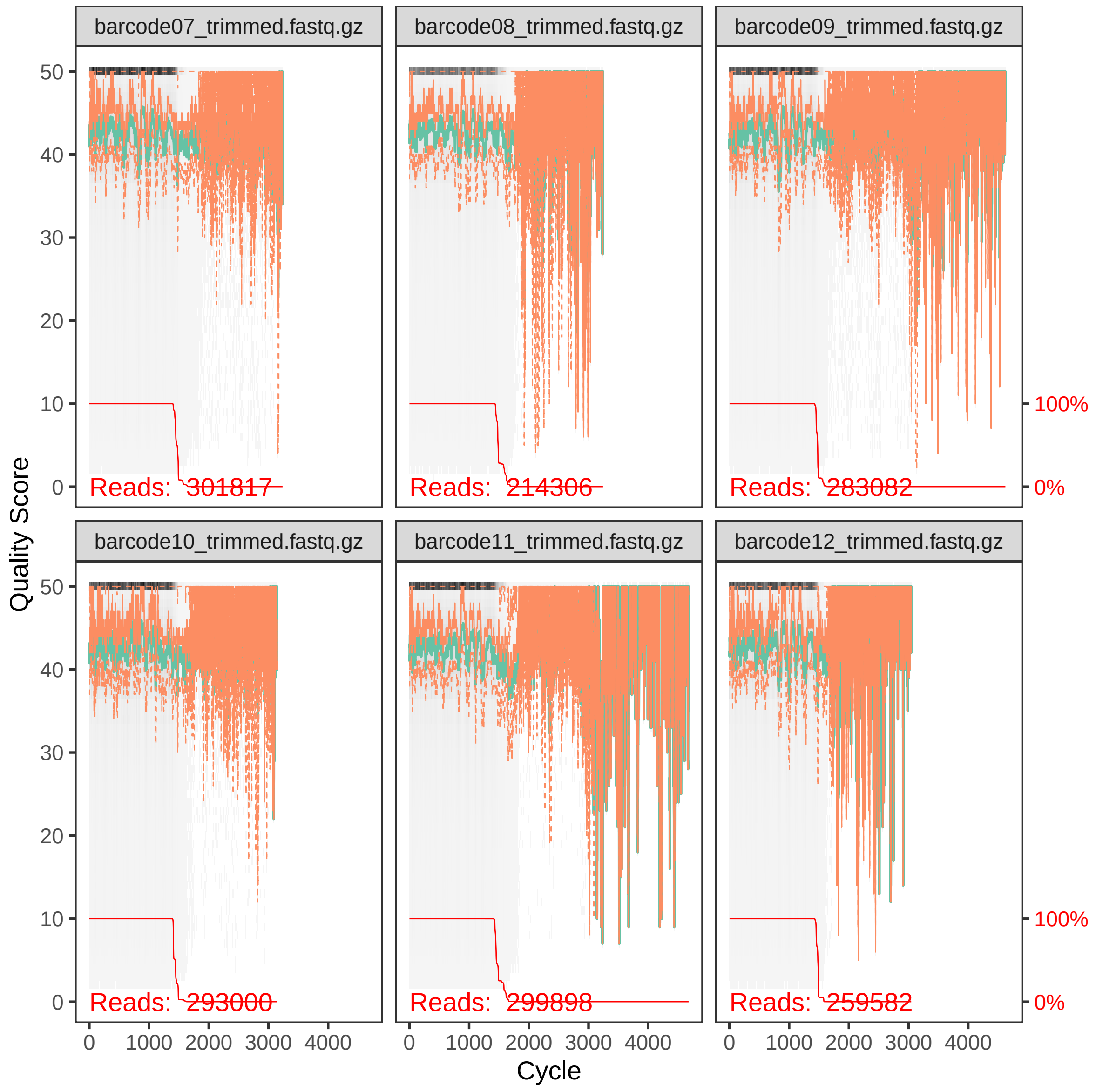
```
[1] "barcode07_trimmed.fastq.gz" "barcode08_trimmed.fastq.gz"
[3] "barcode09_trimmed.fastq.gz" "barcode10_trimmed.fastq.gz"
[5] "barcode11_trimmed.fastq.gz" "barcode12_trimmed.fastq.gz"
[7] "filtered"
```

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

Plot sequence quality profile for samples

```r
# Base quality plot
prsetII <- plotQualityProfile(fnFs[1:6])
prsetII
saveRDS("rds/set2_rds/prset2.rds")
```

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



Filter sequence data

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

```r
# 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 = T, multithread = 2,
                     rm.phix = F)
saveRDS(out, "rds/set2_rds/out.rds")
```

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

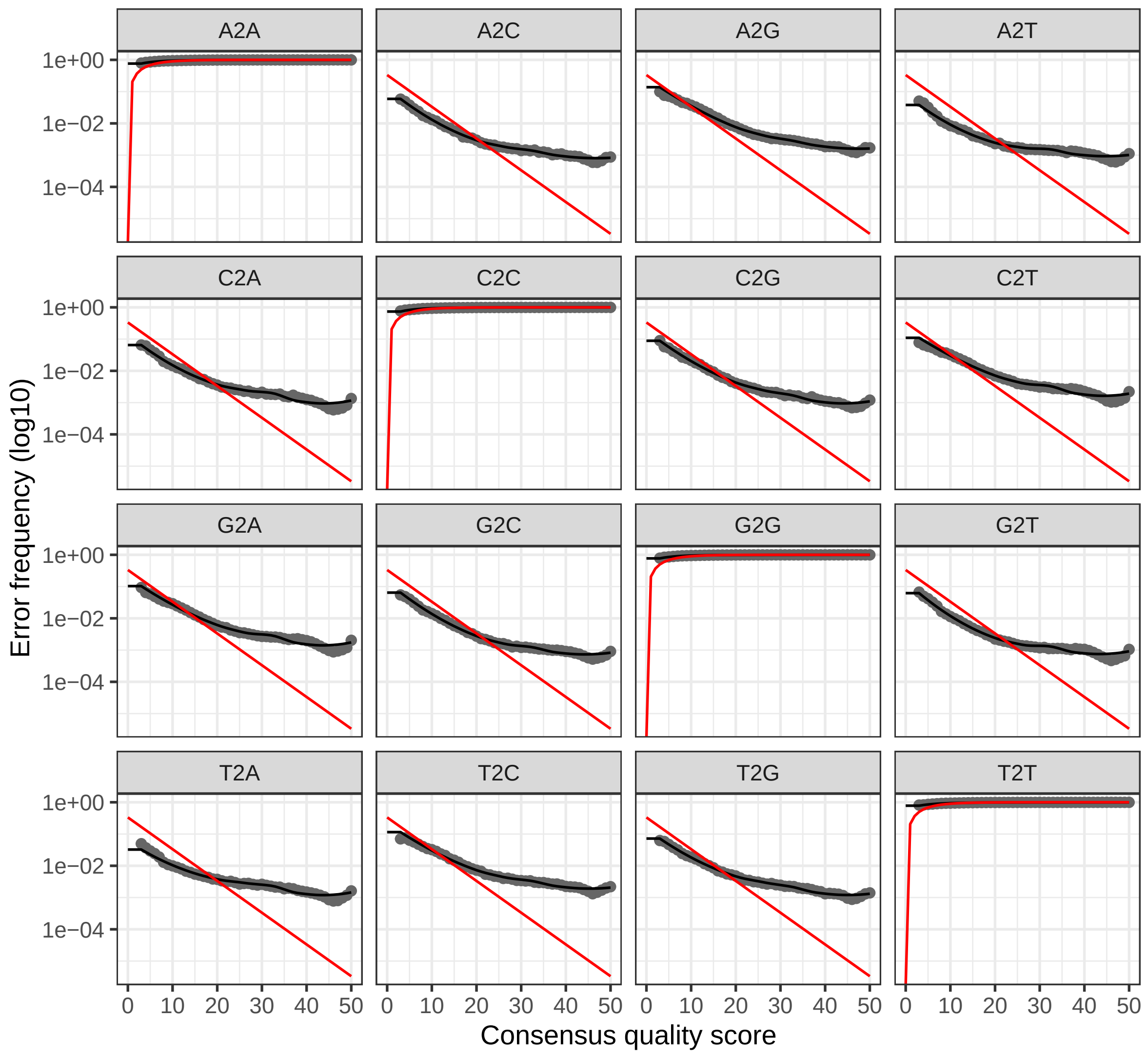
Learn error rates

```r
# Forward read error rate
errF <- learnErrors(filtFs, multithread = 4)
saveRDS(errF, "rds/set2_rds/errF2.rds")
```

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

Plott error rates

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



Denoise

```
dadaFs <- dada(filtFs, err = errF, multithread = 4)
saveRDS(dadaFs, "rds/set2_rds/dadaFs.rds")
```

```
dadaFs <- readRDS("rds/set2_rds/dadaFs.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 = T)
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$Sampleid
kable(track, caption="Summary table")  %>%
  kable_styling(latex_options=c("striped", "HOLD_position"), font_size = 11) %>%
              row_spec(0,background = "MidnightBlue", color = "white")
```

Table 1: Summary table

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

Taxonomy assignment

```
taxonomy <- assignTaxonomy(seqtab.nochim, silva, multithread=3)
taxonomy <- addSpecies(taxonomy, species)
saveRDS(taxonomy, "rds/set2_rds/taxonomy_dada.rds")
```

```
taxonomy <- readRDS("rds/set2_rds/taxonomy_dada.rds")
```

Create TSE object

```
#Preparing counts and variant sequences
counts <- t(seqtab.nochim)
repseq <- DNAStringSet(rownames(counts))
ASV_names <- paste0("ASV", seq(nrow(counts)))
names(repseq) <- ASV_names
rownames(counts) <- NULL
#Preparing taxonomy
rownames(taxonomy) <- NULL
#Metadata
rownames(metadata) <- NULL
#Create tse
tse_dada <- TreeSummarizedExperiment(assays = list(counts = counts),
                                     rowData = DataFrame(taxonomy),
                                     colData = DataFrame(metadata))
rownames(tse_dada) <- ASV_names
#Reference sequences
referenceSeq(tse_dada) <- repseq

#The object
tse_dada
```

```
class: TreeSummarizedExperiment
dim: 933 6
metadata(0):
assays(1): counts
rownames(933): ASV1 ASV2 ... ASV932 ASV933
rowData names(7): Kingdom Phylum ... Genus Species
colnames(6): barcode07 barcode08 ... barcode11 barcode12
colData names(2): Name Media
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (933 sequences)
```

Write results to files

Counts table (equals assays(object_name)$counts) is written to text file

```
#sample names will be columns
ASVdf <- (data.frame(ASV_names,assays(tse_dada)$counts))
#write
write_tsv(ASVdf, paste0(exportloc,"asv_dada.tsv"))
```

Likewise taxonomy table from rowData

```r
#taxonomy ranks in columns
taxonomy <- data.frame(ASV_names, rowData(tse_dada))
#write
write_tsv(taxonomy,paste0(exportloc,"taxonomy_dada.tsv"))
```

Variant sequences are saved into fasta file

```r
tse_dada %>% referenceSeq() %>%
  writeXStringSet(paste0(exportloc, "repseq_dada.fasta"),
                                    append = F, compress = F,
                                    format = "fasta")
```

Writing also metadata ensures that it is compatible with data set

```r
data.frame(colData(tse_dada)) %>% rownames_to_column(var = "Sampleid") %>% write_tsv(paste0(exportloc,"metadata_dada.tsv"))
```

Final step is adding externally created phylogenic tree to object and save object as rds file

```r
phylotree <- read.tree(paste0(exportloc, "tree_dada.nwk"))
rowTree(tse_dada) <- phylotree
saveRDS(tse_dada, paste0(exportloc, "tse_dada.rds"))
```

Vsearch@97%

Data has been processed in qiime, except feature classification

```r
#Preparing counts and variant sequences
counts <- read_tsv("data/set2/feature-table97.tsv", show_col_types = F)
counts <- column_to_rownames(counts, "FeatureID")
counts <- counts[,sort(colnames(counts))]
repseq <- readDNAStringSet("data/set2/dna-sequences97.fasta")
ASV_names <- paste0("ASV", seq(nrow(counts)))
names(repseq) <- ASV_names
rownames(counts) <- NULL
#Preparing taxonomy
taxonomy <- readRDS("rds/set2_rds/taxonomy_vsearch97.rds")
rownames(taxonomy) <- NULL
#Metadata
rownames(metadata) <- NULL
#Create tse
tse_vs97 <- TreeSummarizedExperiment(assays = list(counts = counts),
                                     rowData = DataFrame(taxonomy),
                                     colData = DataFrame(metadata))
rownames(tse_vs97) <- ASV_names
#Reference sequences
referenceSeq(tse_vs97) <- repseq

#The object
tse_vs97
```

```
class: TreeSummarizedExperiment
dim: 2519 6
metadata(0):
assays(1): counts
rownames(2519): ASV1 ASV2 ... ASV2518 ASV2519
rowData names(7): Kingdom Phylum ... Genus Species
colnames(6): barcode07 barcode08 ... barcode11 barcode12
colData names(2): Name Media
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (2519 sequences)
```

Write vsearch97 results. Metadata file remains same.

```r
#sample names will be columns
ASVdf <- (data.frame(ASV_names,assays(tse_vs97)$counts))
#write
write_tsv(ASVdf, paste0(exportloc,"asv_vs97.tsv"))
#taxonomy ranks in columns
taxonomy <- data.frame(ASV_names, rowData(tse_vs97))
#write
write_tsv(taxonomy,paste0(exportloc,"taxonomy_vs97.tsv"))
#sequence file
tse_vs97 %>% referenceSeq() %>%
  writeXStringSet(paste0(exportloc, "repseq_vs97.fasta"),
                                      append = F, compress = F,
                                      format = "fasta")
#read external tree file
vs97_tree <- read.tree("set2/tree_vs97.nwk")
rowTree(tse_vs97) <- vs97_tree
saveRDS(tse_vs97, "set2/tse_vs97.rds")
```

Vsearch@99%

```
#Preparing counts and variant sequences
counts <- read_tsv("data/set2/feature-table99.tsv", show_col_types = F)
counts <- column_to_rownames(counts, "FeatureID")
counts <- counts[,sort(colnames(counts))]
repseq <- readDNAStringSet("data/set2/dna-sequences99.fasta")
ASV_names <- paste0("ASV", seq(nrow(counts)))
names(repseq) <- ASV_names
rownames(counts) <- NULL
#Preparing taxonomy
taxonomy <- readRDS("rds/set2_rds/taxonomy_vsearch99.rds")
taxonomy <- data.frame(taxonomy) %>% mutate(Species = "NA")
rownames(taxonomy) <- NULL
#Metadata
rownames(metadata) <- NULL
#Create tse
tse_vs99 <- TreeSummarizedExperiment(assays = list(counts = counts),
                                     rowData = DataFrame(taxonomy),
                                     colData = DataFrame(metadata))
rownames(tse_vs99) <- ASV_names
#Reference sequences
referenceSeq(tse_vs99) <- repseq
#The object
tse_vs99
```

```
class: TreeSummarizedExperiment
dim: 18314 6
metadata(0):
assays(1): counts
rownames(18314): ASV1 ASV2 ... ASV18313 ASV18314
rowData names(7): Kingdom Phylum ... Genus Species
colnames(6): barcode07 barcode08 ... barcode11 barcode12
colData names(2): Name Media
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (18314 sequences)
```

Write vsearch99 results. Metadata file remains same.

```
#sample names will be columns
ASVdf <- (data.frame(ASV_names,assays(tse_vs99)$counts))
#write
write_tsv(ASVdf, paste0(exportloc,"asv_vs99.tsv"))
#taxonomy ranks in columns
taxonomy <- data.frame(ASV_names, rowData(tse_vs99))
#write
write_tsv(taxonomy,paste0(exportloc,"taxonomy_vs99.tsv"))
#sequence file
tse_vs99 %>% referenceSeq() %>%
  writeXStringSet(paste0(exportloc, "repseq_vs99.fasta"),
                                      append = F, compress = F,
                                      format = "fasta")
#read external tree file
vs99_tree <- read.tree("set2/tree_vs99.nwk")
rowTree(tse_vs99) <- vs99_tree
saveRDS(tse_vs99, "set2/tse_vs99.rds")
```

Emu abundance estimator

Emu results file is pretty messy. Data is processed in separate document and saved as TSE object.

Observations

With larger number of variants and 1400 bp sequence length, error rate profile quite clearly deviates from the expected. Though, variant numbers seem realistic, it's unclear if denoising is alternative for ONT long reads.

From same data, vsearch produced >2500 and >18000 variants, respectively. Filtering of low abundant "OTUs" might be necessary.