Processing nanopore reads

Marko Suokas

Preprocess reads

Dorado does not support demultiplexing dual indexes located on both the 5' and 3' ends. Additionally, in ligated libraries, the reads can appear in either orientation. To address this, we use `cutadapt` for demultiplexing. Index pairs are identified using the linked adapters approach in both forward and reverse orientations, after which scripts are applied to reverse complement the reverse reads. Finally, the reads are merged.

Note: Be aware that autocorrect might change double dashes in command-line examples.

Extracting Forward Reads

You can extract forward reads into a FASTQ file using the following command:

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

This command extracts barcodes defined in the `barcodes.fasta` file and outputs matching reads into individual files within the `fdemuxed` subdirectory. In this example, the minimum read length is set to 1200 bp.

Extracting Reverse Reads

To extract reverse reads, use the reverse-complemented barcode file:

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

The reads are demultiplexed into a separate directory.

Tip: Parameters `-O`, `-e`, `-m`, and `-M` can help reduce the chances of mismatched alignments.

Reverse Complementing Reverse Reads

Next, we use a bash script to process each reverse read file and reverse complement them using the following command:

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

Merging Forward and Reverse Reads

For the final step, you can merge forward and reverse reads with the same base name from two directories. Here's a simple bash command for that:

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

Trimming Primers

Finally, `cutadapt` and bash scripts can be employed to trim forward and reverse PCR primers from the sequence reads.

Import set1 to R

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/set1"
training <- "~/feature_classifiers/SILVA_SSU_r138_2019.RData"
silva <- "~/feature_classifiers/silva_nr99_v138.1_train_set.fa.gz"
species <- "~/feature_classifiers/silva_species_assignment_v138.1.fa.gz"
meta_file <- "data/set1_meta.tsv"
exportloc <- "set1/"
# Variable truncation length
truncation <- 1400
#Creates results directory
dir.create(exportloc)
#metadata file to df
metadata <- read_tsv(meta_file, show_col_types = F)
metadata <- column_to_rownames(metadata, var = "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.

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

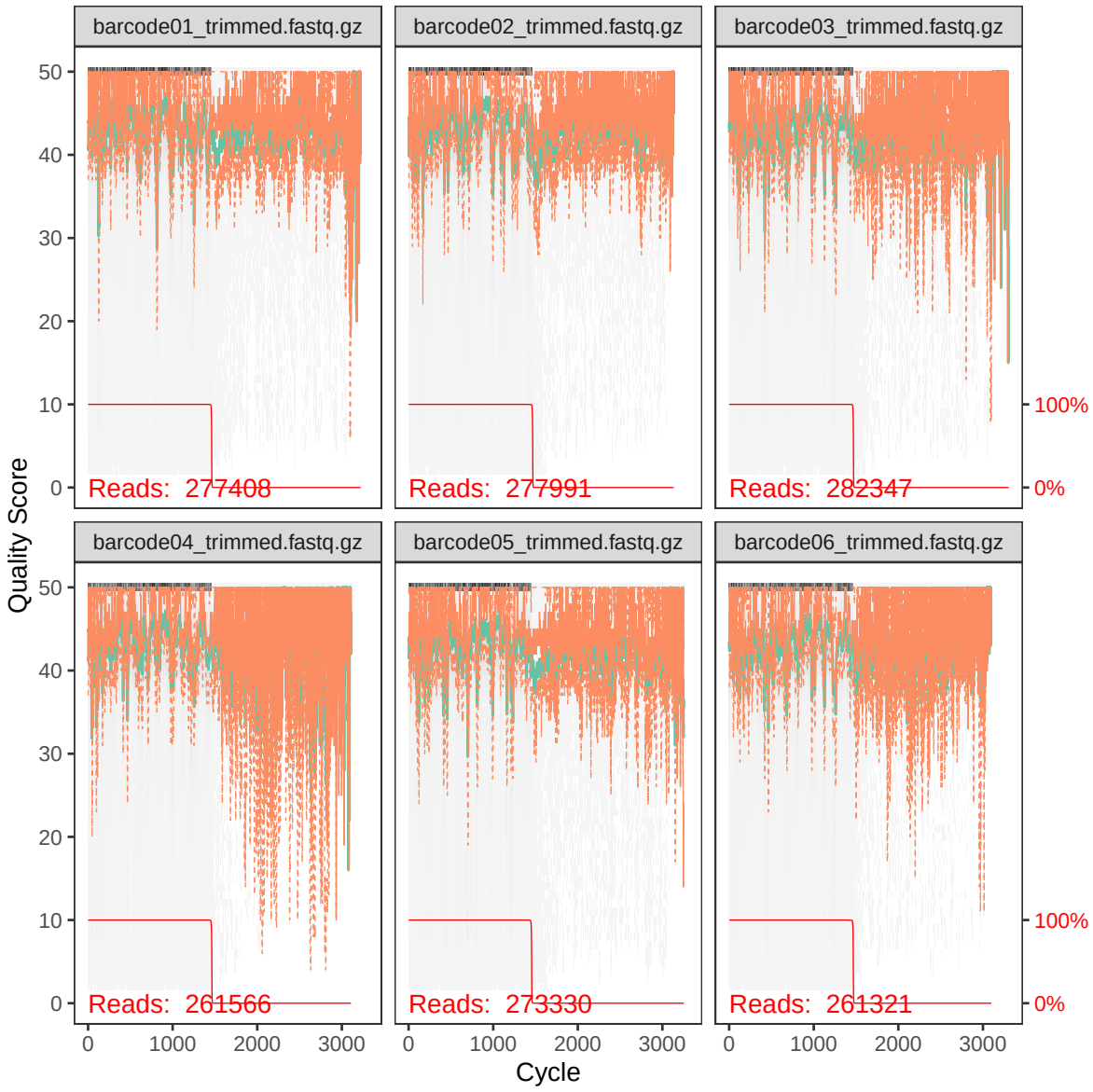
```
[1] "barcode01_trimmed.fastq.gz" "barcode02_trimmed.fastq.gz"
[3] "barcode03_trimmed.fastq.gz" "barcode04_trimmed.fastq.gz"
[5] "barcode05_trimmed.fastq.gz" "barcode06_trimmed.fastq.gz"
```

```
# Forward fastq filenames have format: SAMPLENAME_R1_001.fastq
fnFs <- sort(list.files(path, pattern="_trimmed_all.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

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

```
prsetI <- readRDS("rds/set1_rds/prsetI.rds")
prsetI
```



Filter sequence data

Filtering reads (maxEE ≈ 1 error/200 bp sequence should be 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 = T, multithread = T, rm.phix = F)
```

```
out <- readRDS("rds/set1_rds/out.rds")
```

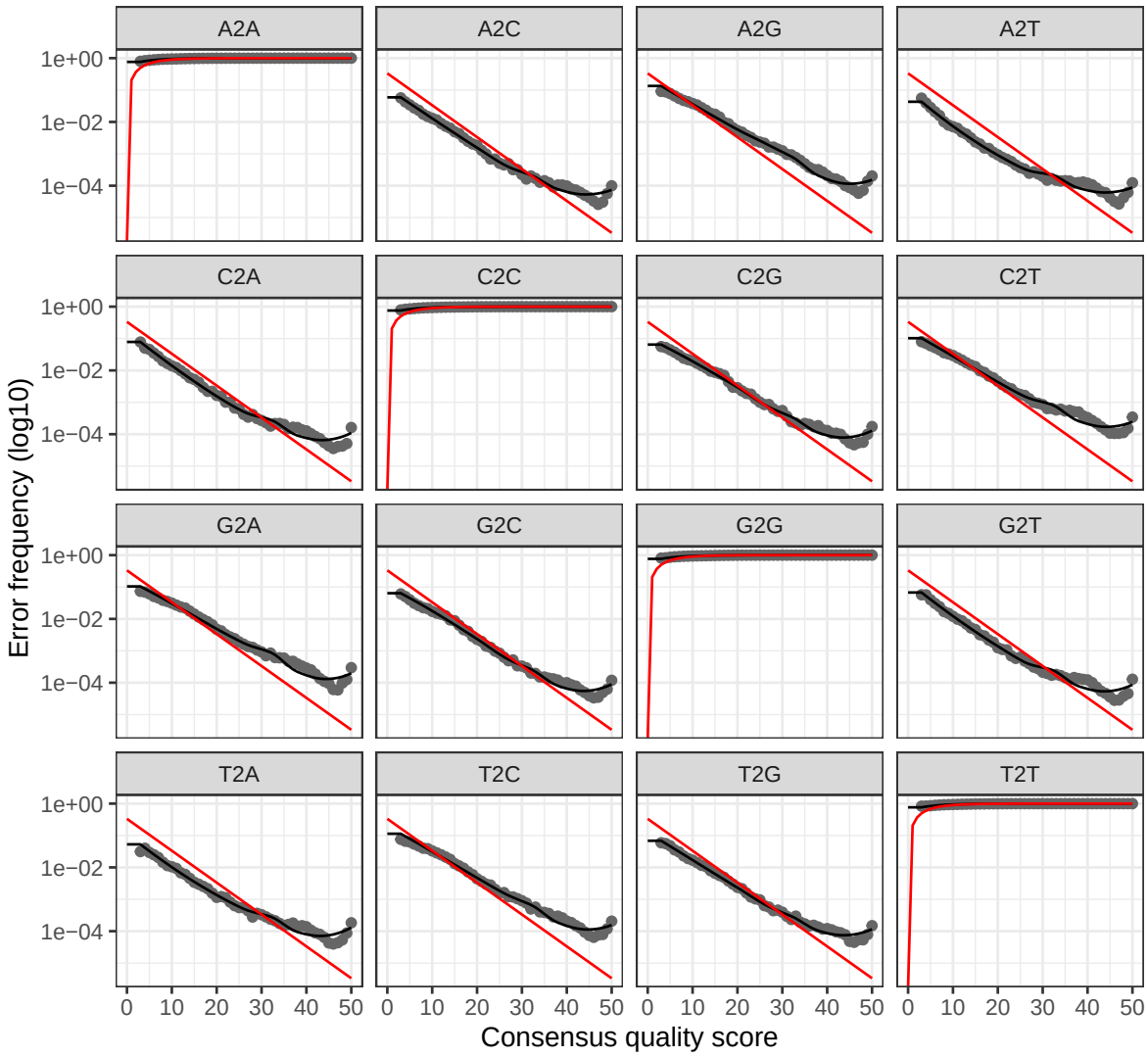
Learn error rates

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

```
errF <- readRDS("rds/set1_rds/errF.rds")
```

Plot error rates

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



Denoise

```
dadaFs <- dada(derepFs, err = errF, multithread = T)
```

```
dadaFs <- readRDS("rds/set1_rds/dadaFs.rds")
```

Build asv table

Dimensions tell us number of samples and variants

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

```
[1]  6 72
```

Chimera removal

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

```
[1]  6 63
```

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) <- rownames(metadata)
kable(track, caption="Summary table")  %>%
  kable_styling(latex_options=c("striped", "HOLD_position"), font_size = 12) %>%
                row_spec(0,background = "teal", color = "ivory")
```

Table 1: Summary table

	Input	Filtered	DenoisedF	Nonchimeric	N:o of variants
barcode01	277408	225976	225599	223766	19
barcode02	277991	223708	223395	223395	12
barcode03	282347	221198	220613	220613	11
barcode04	261566	215868	215550	210277	8
barcode05	273330	203550	203212	203212	16
barcode06	261321	204075	203814	202829	8

Taxonomy assignment

Taxonomy classification against Silva 138.1 including species information.

```
taxonomy <- assignTaxonomy(seqtab.nochim, silva, multithread=3)
taxonomy <- addSpecies(taxonomy, species)
saveRDS(taxonomy, "rds/set1_rds/taxonomy.rds")
```

```
taxonomy <- readRDS("rds/set1_rds/taxonomy.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
#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: 63 6
metadata(0):
assays(1): counts
rownames(63): ASV1 ASV2 ... ASV62 ASV63
rowData names(7): Kingdom Phylum ... Genus Species
colnames(6): barcode01 barcode02 ... barcode05 barcode06
colData names(1): Name
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (63 sequences)
```

Write results to files

Abundance table into tsv file

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

Taxonomy table into tsv file

```
taxdf <- data.frame(ASV_names, rowData(tse_dada))
#write
write_tsv(taxdf, paste0(exportloc,"taxonomy_dada.tsv"))
```

Variant sequences into fasta file

```
writeXStringSet(repseq, paste0(exportloc, "repseq_dada.fasta"),
                                      append = F, compress = F,
                                      format = "fasta")
```

Metadata into tsv file

```
metadf <- metadata %>% rownames_to_column(var = "Sampleid")
#write
write_tsv(metadf, paste0(exportloc,"metadata_dada.tsv"))
```

Add phylotree and save object

```
tree <- read.tree("set1/tree.nwk")
rowTree(tse_dada) <- tree
saveRDS(tse_dada, "set1/tse_dada.rds")
```

Vsearch@97%

Data has been processed in qiime, except taxonomic classification

```
#process qiime2 feature table
vs97 <- read_tsv("data/set1/feature-table97.tsv", show_col_types = F)
ASV_names <- paste0("ASV", seq(nrow(vs97)))
vs97 <- vs97[, order(colnames(vs97))]
vs97[,1] <- NULL
rownames(vs97) <- NULL
#process decipher taxonomy
taxonomy <- readRDS("rds/set1_rds/taxonomy_vsearch97.rds")
rownames(taxonomy) <- NULL
#process repseq fasta
seqs <- readDNAStringSet("data/set1/dna-sequences97.fasta")
names(seqs) <- ASV_names
#create tse
tse_vs97 <- TreeSummarizedExperiment(assays = list(counts = vs97),
                                     rowData = DataFrame(taxonomy),
                                     colData = DataFrame(metadata))
rownames(tse_vs97) <- ASV_names
#Reference sequences
referenceSeq(tse_vs97) <- seqs
#The object
tse_vs97
```

```
class: TreeSummarizedExperiment
dim: 985 6
metadata(0):
assays(1): counts
rownames(985): ASV1 ASV2 ... ASV984 ASV985
rowData names(7): Kingdom Phylum ... Genus Species
colnames(6): barcode01 barcode02 ... barcode05 barcode06
colData names(1): Name
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (985 sequences)
```

Write vsearch97 object

```
#variant_table
ASVdf <- data.frame(ASV_names, assays(tse_vs97)$counts)
write_tsv(ASVdf, "set1/asv_vs97.tsv")
#taxonomy
taxonomy <- data.frame(ASV_names, rowData(tse_vs97))
write_tsv(taxonomy, "set1/taxonomy_vs97.tsv")
#sequences
tse_vs97 %>% referenceSeq() %>% writeXStringSet("set1/repseq97.fasta",
                                     append = F, compress = F,
                                     format = "fasta")
#read and add tree
tree <- read.tree("set1/tree_vs97.nwk")
rowTree(tse_vs97) <- tree
#save rds
saveRDS(tse_vs97, "set1/tse_vs97.rds")
```

Vsearch@99%

```
#process qiime2 feature table
vs99 <- read_tsv("data/set1/feature-table99.tsv", show_col_types = F)
ASV_names <- paste0("ASV", seq(nrow(vs99)))
vs99 <- vs99[, order(colnames(vs99))]
vs99[,1] <- NULL
rownames(vs99) <- NULL
#process decipher taxonomy
taxonomy <- readRDS("rds/set1_rds/taxonomy_vsearch99.rds")
rownames(taxonomy) <- NULL
#process repseq fasta
seqs <- readDNAStringSet("data/set1/dna-sequences99.fasta")
names(seqs) <- ASV_names
#create tse
tse_vs99 <- TreeSummarizedExperiment(assays = list(counts = vs99),
                                     rowData = DataFrame(taxonomy),
                                     colData = DataFrame(metadata))
rownames(tse_vs99) <- ASV_names
#Reference sequences
referenceSeq(tse_vs99) <- seqs
#The object
tse_vs99
```

```
class: TreeSummarizedExperiment
dim: 9195 6
metadata(0):
assays(1): counts
rownames(9195): ASV1 ASV2 ... ASV9194 ASV9195
rowData names(7): Kingdom Phylum ... Genus Species
colnames(6): barcode01 barcode02 ... barcode05 barcode06
colData names(1): Name
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (9195 sequences)
```

Write vsearch99 object

```
#variant_table
ASVdf <- data.frame(ASV_names, assays(tse_vs99)$counts)
write_tsv(ASVdf, "set1/asv_vs99.tsv")
#taxonomy
taxonomy <- data.frame(ASV_names, rowData(tse_vs99))
write_tsv(taxonomy, "set1/taxonomy_vs99.tsv")
#sequences
tse_vs99 %>% referenceSeq() %>% writeXStringSet("set1/repseq99.fasta",
                                     append = F, compress = F,
                                     format = "fasta")
#read and add tree
tree <- read.tree("set1/tree_vs99.nwk")
rowTree(tse_vs99) <- tree
#save rds
saveRDS(tse_vs99, "set1/tse_vs99.rds")
```

Observations

The low bacterial diversity in these samples may explain why denoising yields good results for long 16S rRNA sequences. The error rate plot appears flawless for this data. However, it is noteworthy that all samples contain over 150,000 unique reads.

In contrast, vsearch clustering generated a significantly higher number of variants, exceeding 900 and 9,000, respectively. Lowest number of variants (37) was observed with emu.