Processing and studying nanopore reads

Marko Suokas

#### Preprocessing Reads

Dorado does not support demultiplexing dual indexes on both the 5’ and 3’ ends. Additionally, in ligated libraries, reads can occur 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, followed by scripts that reverse complement the reverse reads. Finally, the reads are merged.

**Note:** Autocorrect may alter double dashes in command-line examples, so ensure they are correctly formatted.

#### Extracting Forward Reads

To extract forward reads into a FASTQ file, use 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 specified in the barcodes.fasta file and outputs the matched reads into individual files within the fdemuxed subdirectory. In this example, the minimum read length is set to 1200 bp.

#### Extracting Reverse Reads

For 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 reverse reads are demultiplexed into the rdemuxed directory.

**Tip:** You can use parameters -O, -e, -m, and -M to reduce the chances of mismatched alignments.

#### Reverse Complementing Reverse Reads

Next, we use a bash script to reverse complement each reverse read file with the following command:

seqkit seq -rp --seq-type DNA -o reverse\_comp.fastq.gz reverse\_out.fastq.gz

#### Merging Forward and Reverse Reads

Finally, you can merge forward and reverse reads with the same base name from two directories using a simple bash script:

zcat forward\_out.fastq.gz reverse\_comp.fastq.gz > merged\_reads.fastq.gz

#### Trimming Primers

Once the reads are merged, cutadapt and bash scripts can be used to trim forward and reverse PCR primers from the 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.

#List files inside directory  
list.files(path)

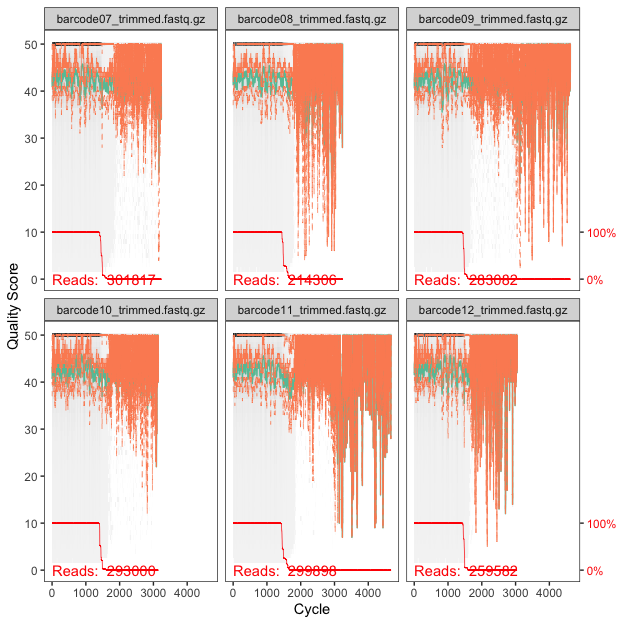
[1] "barcode07\_trimmed.fastq.gz" "barcode08\_trimmed.fastq.gz"  
[3] "barcode09\_trimmed.fastq.gz" "barcode10\_trimmed.fastq.gz"  
[5] "barcode11\_trimmed.fastq.gz" "barcode12\_trimmed.fastq.gz"

# 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

# Base quality plot  
prsetII <- plotQualityProfile(fnFs[1:6])  
prsetII  
saveRDS("rds/set2\_rds/prset2.rds")

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)

# 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")

out <- readRDS("rds/set2\_rds/out2.rds")

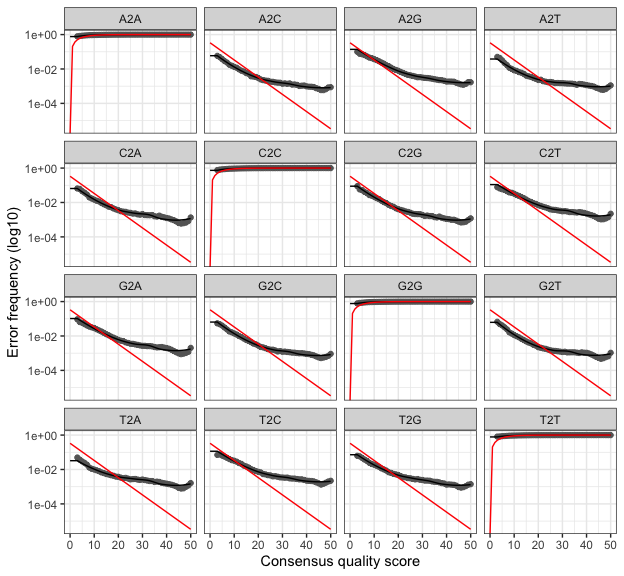
#### Learn error rates

# Forward read error rate  
errF <- learnErrors(filtFs, multithread = 4)  
saveRDS(errF, "rds/set2\_rds/errF2.rds")

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")

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

#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

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

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

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

#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.

#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.