Processing 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 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)

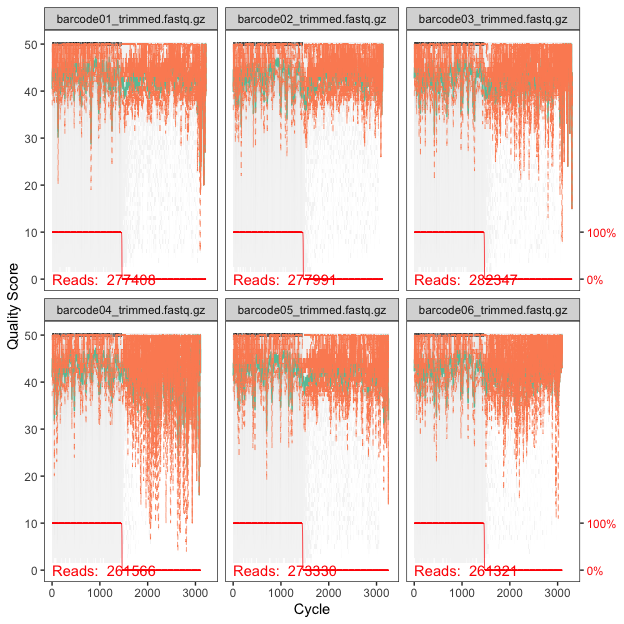
[1] "barcode01\_trimmed.fastq.gz" "barcode02\_trimmed.fastq.gz"  
[3] "barcode03\_trimmed.fastq.gz" "barcode04\_trimmed.fastq.gz"  
[5] "barcode05\_trimmed.fastq.gz" "barcode06\_trimmed.fastq.gz"  
[7] "results"

# 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")

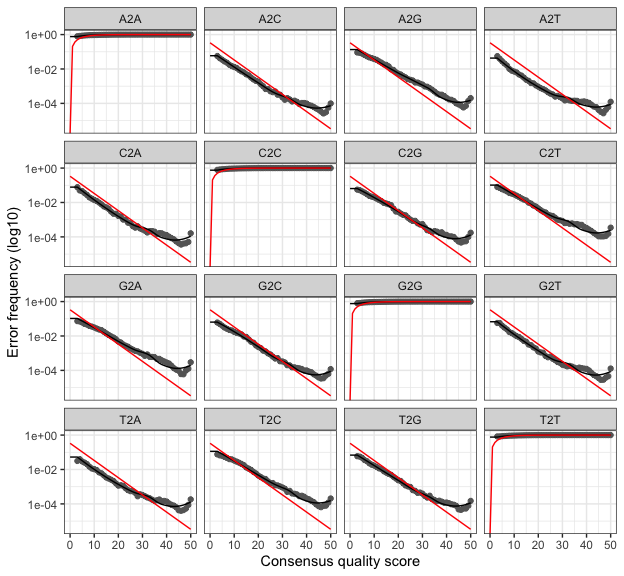
#### Learn error rates

# 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")

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.