# Processing nanopore sequence data

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

Next, forward and reverse reads with the same base name are merged from two directories. Here's a simple bash command for that:

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

### Trimming Primers

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

### Import sequence data to Qiime 2

```
#Activate qiime environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-2024.2
# QIIME 2 command to import sequence data
qiime tools import \
  --type 'SampleData[SequencesWithQuality]' \
  --input-path data/reads/set2/manifest.csv \
  --output-path data/demux.qza \
  --input-format SingleEndFastqManifestPhred33
```

### Dereplicate sequences

Dereplication removes unncessary redundancy from sequence files

```
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-2024.2
# Dereplicate sequences with vsearch plugin
qiime vsearch dereplicate-sequences \
    --p-min-seq-length 1200 \
    --o-dereplicated-table data/derep_table.qza \
    --o-dereplicated-sequences data/derep_sequences.qza \
    --i-sequences data/demux.qza
```

### Pick closed-reference features

Step executed at CSC.

```
#!/bin/bash
#SBATCH --job-name=cluster
#SBATCH --account=project_2010620
#SBATCH --time=48:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=32
#SBATCH --mem=48G
#SBATCH --partition=small
#SBATCH --gres=nvme:100

#set up qiime
module load qiime2/2024.2-amplicon
# run task. Don't use srun in submission as it resets TMPDIR
qiime vsearch cluster-features-closed-reference \
   --i-sequences data/derep_sequences.qza \
   --i-table data/derep_table.qza \
   --i-reference-sequences ~/feature_classifiers/silva-138-99-seqs.qza
   --p-strand plus --p-threads 32 --p-perc-identity 0.97 \
   --output-dir data/closed_ref
```

### Detect chimeric features

```
#Activate qiime environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-2024.2

# run task. Don't use srun in submission as it resets TMPDIR
qiime vsearch uchime-denovo \
   --i-sequences data/closed_ref/clustered_sequences.qza \
   --i-table data/closed_ref/clustered_table.qza  \
   --o-chimeras data/closed_ref/chimeras.qza --o-stats data/closed_ref/stats.qza \
   --o-nonchimeras data/closed_ref/nonchimeras.qza
```

### Filter chimeras from table file

```
#Activate qiime environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-2024.2
# QIIME 2 command to keep nonchimeric features
 qiime feature-table filter-features \
```

```
    --i-table data/closed_ref/clustered_table.qza \
    --m-metadata-file data/closed_ref/nonchimeras.qza \
    --o-filtered-table data/closed_ref/otu_table.qza
```

### Filter chimeras from sequence file

```
#Activate qiime environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-2024.2
# QIIME 2 command to keep nonchimeric sequences
qiime feature-table filter-seqs \
   --i-data data/closed_ref/clustered_sequences.qza \
   --i-table data/closed_ref/otu_table.qza \
   --o-filtered-data data/closed_ref/otu_sequences.qza
```

### R libraries

```
library(mia)
library(scater)
library(vegan)
library(tidyverse)
library(kableExtra)
library(ggthemes)
```

Import qiime otu table and metadata

```
#sequence file
tse <- importQIIME2(featureTableFile = "data/closed_ref/otu_table.qza")
tse <- tse[, sort(colnames(tse))]
#add metadata
metadata <- data.frame(read_tsv("data/set2_meta.tsv",
                                show_col_types = F))
metadata <- column_to_rownames(metadata, "Sampleid")
colData(tse) <- DataFrame(metadata)
tse
```

```
class: TreeSummarizedExperiment
dim: 292 6
metadata(0):
assays(1): counts
rownames(292): EF087979.1.1401 CP001219.333233.334702 ...
  AP007209.9185.10635 CP003108.3632996.3634442
rowData names(0):
```

```
colnames(6): barcode007 barcode008 ... barcode011 barcode012
colData names(2): Name Media
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
```

Qiime arranges sequence file alphabetically, while TSE expects reference sequences in same order as rownames. Thus we need to rearrange sequences

```
ref_sequences <- importQZA("data/closed_ref/otu_sequences.qza")
ref_ids <- names(ref_sequences)
tse_ids <- rownames(tse)
# Check if all rownames are present in the reference IDs
if (!all(tse_ids %in% ref_ids)) {
  stop("Not all rownames from tse are present in the reference sequences.")
}
# Reorder `ref_sequences` to match the order of `tse` rownames
ref_sequences_ordered <- ref_sequences[match(tse_ids, ref_ids)]
all(names(ref_sequences_ordered) == rownames(tse))
```

```
[1] TRUE
```

```
referenceSeq(tse) <- ref_sequences_ordered
```

### Assign taxonomy

You can fetch taxonomy directly from Silva reference taxonomy file. However, dada2 classifier has more polished reference, so we reassign taxonomy.

```
taxa <- assignTaxonomy(referenceSeq(tse), refFasta = "~/feature_classifiers/silva_nr99_v138.1
saveRDS(taxa, "data/closed_ref/taxa.rds")
```

Add taxonomy to rowData

```
taxa <- readRDS("data/closed_ref/taxa.rds")
#Add taxonomy
rownames(taxa) <- NULL
rowData(tse) <- DataFrame(taxa)
#Rename rows (alternative to Silva ID)
rownames(tse) <- paste0("OTU_", seq_len(nrow(tse)))
tse
```

```
class: TreeSummarizedExperiment
dim: 292 6
metadata(0):
assays(1): counts
rownames(292): OTU_1 OTU_2 ... OTU_291 OTU_292
rowData names(6): Kingdom Phylum ... Family Genus
colnames(6): barcode007 barcode008 ... barcode011 barcode012
colData names(2): Name Media
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (292 sequences)
```

### Write object data to files

Write RDS. It can be reloaded as a ready to be used data object to R

```
saveRDS(tse, "results/closedref/tse.rds")
```

Write an abundance table

```
#FeatureID will be rowname
abd <- data.frame(FeatureID = rownames(tse),assays(tse)$counts)
#Write
write_tsv(abd, "results/closedref/otu_table.tsv")
```

Write a taxonomy table

```
#FeatureID will be rowname
taxt <- data.frame(FeatureID = rownames(tse), rowData(tse))
#Write
write_tsv(taxt, "results/closedref/taxonomy.tsv")
```

Write variant sequences to fasta file

```
writeXStringSet(referenceSeq(tse), "results/closedref/repseq.fasta",
                                    append = F, compress = F,
                                    format = "fasta")
```

Write a metadata file

```
metadf <- data.frame(colData(tse)) %>% rownames_to_column(var="Sampleid")
#write
write_tsv(metadf, "results/closedref/metadata.tsv")
```