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

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 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/processed/set1/manifest.tsv \
  --output-path data/work/set1/demux.qza \
  --input-format SingleEndFastqManifestPhred33
```

Dereplicate sequences

```
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/work/dereplicated_table.qza \
    --o-dereplicated-sequences data/work/set1/derep_sequences.qza \
    --i-sequences data/work/demux.qza
```

Pick otus using closed Silva reference

Step performed 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 derep_sequences.qza \
   --i-table derep_table.qza \
   --i-reference-sequences ../silva-138-99-seqs.qza \
   --p-strand plus --p-threads 32 --p-perc-identity 0.97 \
   --output-dir  results
```

Detect chimeric otus

Step performed at CSC

```
#!/bin/bash
#SBATCH --job-name=chimera_detection
#SBATCH --account=project_2010620
#SBATCH --time=24:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=24G
#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 uchime-denovo --i-sequences clustered_sequences.qza \
--i-table clustered_table.qza  \
--output-dir chimeric
```

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/work/set1/clustered_table.qza \
    --m-metadata-file data/work/set1/nonchimeras.qza \
    --p-min-frequency 2 \
    --o-filtered-table data/work/set1/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/work/set1/clustered_sequences.qza \
   --i-table data/work/set1/otu_table.qza \
   --o-filtered-data data/work/set1/otu_sequences.qza
```

Libraries

```
library(mia)
library(Biostrings)
library(ShortRead)
library(tidyverse)
library(kableExtra)
library(ggthemes)
```

Import qiime files and metadata

```
#sequence file
tse <- importQIIME2(featureTableFile = "data/work/set1/otu_table.qza",
                    refSeqFile = "data/work/set1/otu_sequences.qza")
tse <- tse[, sort(colnames(tse))]
#add metadata
metadata <- data.frame(read_tsv("data/set1_meta.tsv",
                                show_col_types = F))
metadata <- column_to_rownames(metadata, "Sampleid")
colData(tse) <- DataFrame(metadata)
tse
```

```
class: TreeSummarizedExperiment
dim: 163 6
metadata(0):
assays(1): counts
rownames(163): CP016294.613081.614615 JF312983.1.1520 ...
  KU533726.1.1442 KF941215.1.1481
rowData names(0):
colnames(6): barcode001 barcode002 ... barcode005 barcode006
colData names(1): Name
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (163 sequences)
```

Create taxonomy table from Silva

```
#
silva_tax <- read_tsv("~/feature_classifiers/silva_taxonomy.tsv",
                      show_col_types = F)
rows <- data.frame(FeatureID = rownames(tse))
taxonomy <- dplyr::left_join(rows, silva_tax, by = "FeatureID")
taxonomy <- taxonomy %>%
  separate(Taxon,
           into = c("Kingdom", "Phylum", "Class", "Order", "Family", "Genus", "Species"),
           sep = "; ",
           fill = "right")
taxonomy <- taxonomy %>%
  mutate(across(everything(), ~ sub("^[a-z]__", "", .)))
taxonomy <- column_to_rownames(taxonomy, "FeatureID")
```

Add taxonomy to rowData

```
#Add taxonomy
rownames(taxonomy) <- NULL
rowData(tse) <- DataFrame(taxonomy)
#Rename rows (alternative to Silva ID)
#rownames(tse) <- paste0("OTU_", seq_len(nrow(tse)))
tse
```

```
class: TreeSummarizedExperiment
dim: 163 6
metadata(0):
assays(1): counts
rownames(163): CP016294.613081.614615 JF312983.1.1520 ...
  KU533726.1.1442 KF941215.1.1481
rowData names(7): Kingdom Phylum ... Genus Species
colnames(6): barcode001 barcode002 ... barcode005 barcode006
colData names(1): Name
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (163 sequences)
```

Write object data to files

Write RDS. It can be reloaded as unchanged object to R

```
saveRDS(tse, "set1/tse.rds")
```

Write abundance table

```
#FeatureID will be rowname
abd <- data.frame(FeatureID = rownames(tse),assays(tse)$counts)
#Write
write_tsv(abd, "set1/otu_table.tsv")
```

Write taxonomy table

```
#FeatureID will be rowname
taxt <- data.frame(FeatureID = rownames(tse), assays(tse)$counts)
#Write
write_tsv(taxt, "set1/taxonomy.tsv")
```

Variant sequences to fasta file

```
writeXStringSet(referenceSeq(tse), "set1/repseq.fasta",
                                    append = F, compress = F,
                                    format = "fasta")
```

Write metadata file

```
metadf <- data.frame(colData(tse)) %>% rownames_to_column(var="Sampleid")
#write
write_tsv(metadf, "set1/metadata.tsv")
```

Microbe data analysis

Agglomerate taxonomy to genus rank and count relative abundance

```
tse <- agglomerateByRank(tse, rank = "Genus",
                             onRankOnly = T, na.rm = F)
#relabundance
tse <- transformAssay(tse, assay.type = "counts", method = "relabundance")
```

Pick ten most abundant features

```
#top10 features
top10 <- getTopFeatures(tse, top = 10, method = "mean",
                        assay.type = "relabundance")
#create and filter table
table <- data.frame(assays(tse)$relabundance)
table <- table %>% rownames_to_column(var = "Genus") %>%
  filter(Genus %in% top10)
```

Abundance table

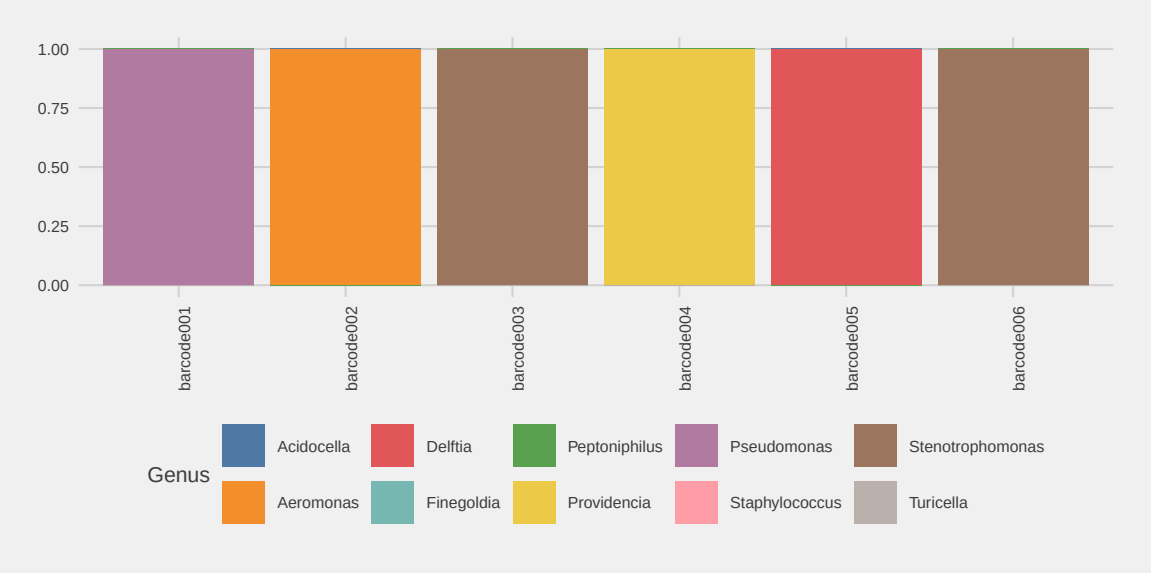
```
kable(table, digits=3) %>%
  kable_styling(latex_options = c("HOLD_position", "striped"),
                font_size = 10) %>%
  row_spec(0, background = "teal", color = "white")
```

Genus	barcode001	barcode002	barcode003	barcode004	barcode005	barcode006
Stenotrophomonas	0	0	1	0	0	1
Pseudomonas	1	0	0	0	0	0
Delftia	0	0	0	0	1	0
Providencia	0	0	0	1	0	0
Aeromonas	0	1	0	0	0	0
Acidocella	0	0	0	0	0	0
Finegoldia	0	0	0	0	0	0
Staphylococcus	0	0	0	0	0	0
Turicella	0	0	0	0	0	0
Peptoniphilus	0	0	0	0	0	0

Composition plot

Change data to long table format

```
df_long <- table %>% pivot_longer(cols = starts_with("barcode"),
names_to = "Sample", values_to = "Abundance")
#Plot stacked barplot
ggplot(df_long, aes(x = Sample, y = Abundance, fill = Genus)) +
geom_bar(stat = "identity") + theme_fivethirtyeight(base_size = 8) + scale_fill_tableau() + theme(axis.text.x =
element_text(angle = 90))
```



Summary of closed reference otu picking strategy

Count first number of sequences in the raw sequence files

```
# List all compressed fastq files in the folder
fastq_files <- list.files("data/processed/set1", pattern = "\\.fastq\\.gz$", full.names = T)

# Function to count sequences in a compressed FASTQ file
count_sequences_in_fastq <- function(file) {
  # Use gzfile to read the compressed file
  fq <- readFastq(file)
  return(length(fq))
}
# Apply the function to each file and store the counts
sequence_counts <- sapply(fastq_files, count_sequences_in_fastq)
# Print the result
raw_data <- data.frame(File = fastq_files, Sequences = sequence_counts)
```

Create summary table

```
summary <- data.frame(Sample = colnames(tse), Raw_Counts = raw_data$Sequences, Counts = colSums(assays(tse)$counts))
summary <- summary %>% mutate(Percentage = Counts/Raw_Counts)
rownames(summary) <- NULL
kable(summary, digits=2) %>%
  kable_styling(latex_options = c("HOLD_position", "striped"),
                font_size = 12) %>%
  row_spec(0, background = "teal", color = "white")
```

Sample	Raw_Counts	Counts	Percentage
barcode001	391434	308417	0.79
barcode002	394282	362634	0.92
barcode003	401407	400245	1.00
barcode004	370675	358424	0.97
barcode005	388126	382892	0.99
barcode006	371125	359834	0.97

Results

Due to nature of nanopore reads, closed reference otu picking was performed against Silva 138.1 database at 97 % identity level. Results in the samples remain unchanged compared to earlier. However, methologically earlier results are based on wrong assumptions.

Otu picking percentage was high across all samples. This can be interpreted as there is no problem in sequence quality per se. Problem is that nanopore is unable to predict quality at single nucleotide level in which most modern analysis methods are based on.