Processing fungal nanopore sequences

Preprocessing

The Nanopore basecaller lacks support for demultiplexing dual indexes located on both the 5' and 3' ends of reads. Additionally, ligated libraries may contain reads in either orientation. To address these, we use `Cutadapt` for demultiplexing and `SeqKit` for reverse-complementing reads in the reverse orientation. The sequences are then merged, and PCR primers are trimmed. A Python wrapper script has been developed to automate these steps. End result is demultiplexed fastq.gz read files.

ITSxpress (standalone or Qiime plugin)

`Itsxpress` is used to trim ITS regions. Thus, after the step, we have three datasets: one containing ITS1 sequences, one with ITS2 sequences, and one with both regions. The code will also remove reads that are shorter than 20 nt as zero length sequences will cause error with `vsearch`.

```
#!/bin/bash
source /etc/profile.d/conda.sh
conda activate base

# Define the input and output directories
input_dir="/data/projects/fungi"
output_dir="/data/projects/fungi"

# Create output subdirectories if they don't exist
mkdir -p "$output_dir/ITS1" "$output_dir/ITS2" "$output_dir/all"

# Loop over each file in the input directory
for file in "$input_dir"/*
do
    # Get the base name of the file (without path and extension)
    base_name=$(basename "$file" .fastq.gz)
    # Process with ITS1 option
    itsxpress --single_end --fastq "$file" --threads 20 \
    --region ITS1 --outfile "$output_dir/ITS1/${base_name}_ITS1.fastq.gz"

    cutadapt -m 20 -o "$output_dir/ITS1/${base_name}_ITS1_trimmed.fastq.gz" \
    "$output_dir/ITS1/${base_name}_ITS1.fastq.gz"

    # Process with ITS2 option
    itsxpress --single_end --fastq "$file" --threads 20 \
    --region ITS2 --outfile "$output_dir/ITS2/${base_name}_ITS2.fastq.gz"

    cutadapt -m 20 -o "$output_dir/ITS2/${base_name}_ITS2_trimmed.fastq.gz" \
    "$output_dir/ITS2/${base_name}_ITS2.fastq.gz"

    # Process with all regions option
    itsxpress --single_end --fastq "$file" --threads 20 \
    --region ALL --outfile "$output_dir/all/${base_name}_all.fastq.gz"

    cutadapt -m 20 -o "$output_dir/all/${base_name}_all_trimmed.fastq.gz" \
    "$output_dir/all/${base_name}_all.fastq.gz"
done

echo "Processing complete! Check the output directory for results."
```

Process extracted ITS1 reads

Import to Qiime via manifest file as usual. It's also possible to request AI to prepare file for you by providing necessary information.

```
# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-amplicon-2024.5

# Run import command
qiime tools import \
  --type 'SampleData[SequencesWithQuality]' \
  --input-path ITS1/manifest.csv \
  --output-path q2/demux_its1.qza \
  --input-format SingleEndFastqManifestPhred33
```

Dereplicate sequences with `vsearch` plugin

```
# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-amplicon-2024.5

# Dereplicate
qiime vsearch dereplicate-sequences \
--i-sequences q2/demux_its1.qza \
--o-dereplicated-table q2/its1/derep_table.qza \
--o-dereplicated-sequences q2/its1/derep_seq.qza \
```

De-novo otu picking at 97 % identity level using `vsearch` plugin

```
# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-amplicon-2024.5

# Pick otus
qiime vsearch cluster-features-de-novo \
--i-sequences q2/its1/derep_seq.qza \
--i-table q2/its1/derep_table.qza \
--p-perc-identity 0.97 --p-strand plus \
--p-threads 20 --output-dir q2/its1/features
```

Filter rare features

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

# Filter rare features
qiime feature-table filter-features \
--i-table q2/its1/features/clustered_table.qza \
--p-min-frequency 10 \
--o-filtered-table q2/its1/features/f1_table.qza

qiime feature-table filter-seqs \
--i-data q2/its1/features/clustered_sequences.qza \
--i-table q2/its1/features/f1_table.qza \
--o-filtered-data q2/its1/features/f1_sequences.qza
```

Identify chimeric sequences

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

# Detect chimeras
qiime vsearch uchime-denovo \
--i-sequences q2/its1/features/f1_sequences.qza \
--i-table q2/its1/features/f1_table.qza \
--o-chimeras q2/its1/features/chimeras.qza \
--o-stats q2/its1/features/stats.qza \
--o-nonchimeras q2/its1/features/nonchimeras.qza
```

Filter chimeras from feature table

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

# Keep nonchimeric features in the table
qiime feature-table filter-features \
--i-table q2/its1/features/f1_table.qza \
--m-metadata-file q2/its1/features/nonchimeras.qza \
--o-filtered-table q2/its1/features/otu_table.qza
```

Filter sequence file

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

# Keep nonchimeric sequences
qiime feature-table filter-seqs \
--i-data q2/its1/features/f1_sequences.qza \
--i-table q2/its1/features/otu_table.qza \
--o-filtered-data q2/its1/features/otu_sequences.qza
```

Load R libraries

```r
library(tidyverse)
library(kableExtra)
library(ggthemes)
library(dada2)
library(mia)
```

Import feature table and metadata to TreeSummarizedExperiment

```r
# Import feature table and sort sample names alphabetically
tse <- importQIIME2(featureTableFile = "q2/its1/features/otu_table.qza")
tse <- tse[, sort(colnames(tse))]

# Import metadata file and add data to colData
metadata <- data.frame(read_tsv("meta.tsv",
show_col_types = F))
metadata <- column_to_rownames(metadata, "sampleid")
colData(tse) <- DataFrame(metadata)

# Check dimensions
tse
```

```
class: TreeSummarizedExperiment
dim: 9569 24
metadata(0):
assays(1): counts
rownames(9569): bdff85154aeee8c692bc09c0e6036175d2c13fc9
  512752594d8c4edc51da1ef084be93b8dbefec79 ...
  0b1a693dd82b3309f69c8c0a71d2a03f3ac48f5d
  359935bddb5d54d4dd105c173ed0776cce2da5a0
rowData names(0):
colnames(24): Barcode001 Barcode002 ... Barcode023 Barcode024
colData names(5): Labnro Alue Kasvillisuus Soil_pH Maanäyte
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
```

Import sequences and rearrane them to correspond count table

```r
# Import squence file
ref_sequences <- importQZA("q2/its1/features/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
```

```r
# Included ordered sequences to data object
referenceSeq(tse) <- ref_sequences_ordered
```

Classify taxonomy. This step is computionally heavy.

```r
# Unite reference file
unite <- "~/feature_classifiers/sh_general_release_dynamic_04.04.2024.fasta"
# Assign taxonomy using dada2 assignTaxonomy function
taxa <- assignTaxonomy(referenceSeq(tse),unite, minBoot=50, multithread=2)
# Save result to rds file
saveRDS(taxa, "q2/its1_taxa.rds")
```

Process and included taxonomic results

```
# Read results
taxa <- data.frame(readRDS("q2/its1_taxa.rds"))
# Remove taxa prefixes from each column
taxa <- as.data.frame(lapply(taxa, function(x) sub("^[a-z]__","", x)))
#Add taxonomy to rowData
rownames(taxa) <- NULL
rowData(tse) <- DataFrame(taxa)
# Rename rows
rownames(tse) <- paste0("OTU_", seq_len(nrow(tse)))
```

Prune non-fungal taxa from data

```
# Non-bacterial taxa
tse <- tse[rowData(tse)$Kingdom %in% "Fungi",]
# Final dimensions
tse
```

```
class: TreeSummarizedExperiment
dim: 9553 24
metadata(0):
assays(1): counts
rownames(9553): OTU_1 OTU_2 ... OTU_9568 OTU_9569
rowData names(7): Kingdom Phylum ... Genus Species
colnames(24): Barcode001 Barcode002 ... Barcode023 Barcode024
colData names(5): Labnro Alue Kasvillisuus Soil_pH Maanäyte
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (9553 sequences)
```

Write final object data to files

RDS

```
saveRDS(tse, "results/tse_its1_original.rds")
```

Abundance table

```
#FeatureID will be rowname
abd <- data.frame(FeatureID = rownames(tse),assays(tse)$counts)
#Write
write_tsv(abd, "results/its1_feature_table.tsv")
```

Taxonomy table

```
#FeatureID will be rowname
taxt <- data.frame(FeatureID = rownames(tse), rowData(tse))
#Write
write_tsv(taxt, "results/its1_taxonomy.tsv")
```

Feature sequences

```
# Write fasta file
writeXStringSet(referenceSeq(tse), "results/its1_repseq.fasta",
append = F, compress = F,
format = "fasta")
```

Metadata file

```
metadf <- data.frame(colData(tse)) %>% rownames_to_column(var="Sampleid")
# Write
write_tsv(metadf, "results/its1_metadata.tsv")
```

Process extracted ITS2 reads

Import to Qiime via manifest file as usual. It's also possible to request AI to prepare file for you by providng necessary information.

```
# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-amplicon-2024.5

# Import command
qiime tools import \
  --type 'SampleData[SequencesWithQuality]' \
  --input-path ITS2/manifest.csv \
  --output-path q2/demux_its2.qza \
  --input-format SingleEndFastqManifestPhred33
```

Dereplicate imported sequences with `vsearch` plugin

```
#!/bin/bash

# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-amplicon-2024.5

#Dereplicate
qiime vsearch dereplicate-sequences \
--i-sequences q2/demux_its2.qza \
--o-dereplicated-table q2/its2/derep_table.qza \
--o-dereplicated-sequences q2/its2/derep_seq.qza \
```

De-novo otu picking at 97 % identity level using `vsearch` plugin

```
# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-amplicon-2024.5
```

```
# Pick otus
qiime vsearch cluster-features-de-novo \
--i-sequences q2/its2/derep_seq.qza \
--i-table q2/its2/derep_table.qza \
--p-perc-identity 0.97 --p-strand plus \
--p-threads 20 --output-dir q2/its2/features
```

Filter rare features

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

# Filter rare features
qiime feature-table filter-features \
--i-table q2/its2/features/clustered_table.qza \
--p-min-frequency 10 \
--o-filtered-table q2/its2/features/f1_table.qza

qiime feature-table filter-seqs \
--i-data q2/its2/features/clustered_sequences.qza \
--i-table q2/its2/features/f1_table.qza \
--o-filtered-data q2/its2/features/f1_sequences.qza
```

Identify chimeric sequences

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

# Detect chimeras
qiime vsearch uchime-denovo \
--i-sequences q2/its2/features/f1_sequences.qza \
--i-table q2/its2/features/f1_table.qza \
--o-chimeras q2/its2/features/chimeras.qza \
--o-stats q2/its2/features/stats.qza \
--o-nonchimeras q2/its2/features/nonchimeras.qza
```

Filter chimeras from feature table

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

# Keep nonchimeric features in the table
qiime feature-table filter-features \
--i-table q2/its2/features/f1_table.qza \
--m-metadata-file q2/its2/features/nonchimeras.qza \
--o-filtered-table q2/its2/features/otu_table.qza
```

Filter 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 q2/its2/features/f1_sequences.qza \
--i-table q2/its2/features/otu_table.qza \
--o-filtered-data q2/its2/features/otu_sequences.qza
```

Import feature table and metadata to tse Import feature table and metadata to tse

```
# Import feature table and sort sample names alphabetically
tse <- importQIIME2(featureTableFile = "q2/its2/features/otu_table.qza")
tse <- tse[, sort(colnames(tse))]

# Import metadata file and add data to colData
metadata <- data.frame(read_tsv("meta.tsv",
show_col_types = F))
metadata <- column_to_rownames(metadata, "sampleid")
colData(tse) <- DataFrame(metadata)

# Check dimensions
tse
```

```
class: TreeSummarizedExperiment
dim: 9709 24
metadata(0):
assays(1): counts
rownames(9709): 26f4071d1add3a49098c6a0911bd63d950f4a289
  154c9b9d47ea38a3d4eb3ab2fb91d6aaf40af3e8 ...
  121bf0dc5425a52dbc02b7c3f01815d934c187e6
  0b344b50e92e220b6add5f0deb0a154a38634a2d
rowData names(0):
colnames(24): Barcode001 Barcode002 ... Barcode023 Barcode024
colData names(5): Labnro Alue Kasvillisuus Soil_pH Maanäyte
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
```

```
rowTree: NULL
colLinks: NULL
colTree: NULL
```

Import sequences and rearrange to correspond count table

```r
# Import sequence file
ref_sequences <- importQZA("q2/its2/features/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
```

```r
# Included ordered sequences to data object
referenceSeq(tse) <- ref_sequences_ordered
```

Classify taxonomy. This step is computionally heavy.

```r
# Unite reference file
unite <- "~/feature_classifiers/sh_general_release_dynamic_04.04.2024.fasta"
# Assign taxonomy using dada2 assignTaxonomy function
taxa <- assignTaxonomy(referenceSeq(tse),unite, minBoot=50, multithread=2)
# Save result to rds file
saveRDS(taxa, "q2/its2_taxa.rds")
```

Process and included taxonomic results

```r
# Read results
taxa <- data.frame(readRDS("q2/its2_taxa.rds"))
# Remove taxa prefixes from each column
taxa <- as.data.frame(lapply(taxa, function(x) sub("^[a-z]__","", x)))
#Add taxonomy to rowData
rownames(taxa) <- NULL
rowData(tse) <- DataFrame(taxa)
# Rename rows
rownames(tse) <- paste0("OTU_", seq_len(nrow(tse)))
```

Prune non-fungal taxa from data

```r
# Non-bacterial taxa
tse <- tse[rowData(tse)$Kingdom %in% "Fungi",]
# Final dimensions
tse
```

```
class: TreeSummarizedExperiment
dim: 9709 24
metadata(0):
assays(1): counts
rownames(9709): OTU_1 OTU_2 ... OTU_9708 OTU_9709
rowData names(7): Kingdom Phylum ... Genus Species
colnames(24): Barcode001 Barcode002 ... Barcode023 Barcode024
colData names(5): Labnro Alue Kasvillisuus Soil_pH Maanäyte
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (9709 sequences)
```

Write results to files

RDS

```
saveRDS(tse, "results/tse_its2_original.rds")
```

Abundance table

```
#FeatureID will be rowname
abd <- data.frame(FeatureID = rownames(tse),assays(tse)$counts)
#Write
write_tsv(abd, "results/its2_feature_table.tsv")
```

Taxonomy table

```
#FeatureID will be rowname
taxt <- data.frame(FeatureID = rownames(tse), rowData(tse))
#Write
write_tsv(taxt, "results/its2_taxonomy.tsv")
```

Feature sequences

```
# Write fasta file
writeXStringSet(referenceSeq(tse), "results/its2_repseq.fasta",
append = F, compress = F,
format = "fasta")
```

Metadata file

```
metadf <- data.frame(colData(tse)) %>% rownames_to_column(var="Sampleid")
#write
write_tsv(metadf, "results/its2_metadata.tsv")
```

Process extracted full ITS region reads

Import reads

```
# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-amplicon-2024.5

# import command
qiime tools import \
  --type 'SampleData[SequencesWithQuality]' \
  --input-path all/processed/manifest.csv \
  --output-path q2/demux_all.qza \
  --input-format SingleEndFastqManifestPhred33
```

Dereplicate full its sequences

```
# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-amplicon-2024.5

# Dereplicate
qiime vsearch dereplicate-sequences \
--i-sequences q2/demux_all.qza \
--o-dereplicated-table q2/itsall/derep_table.qza \
--o-dereplicated-sequences q2/itsall/derep_seq.qza \
--verbose
```

De-novo otu picking at 97 % identity level using `vsearch` plugin

```
# Activate the QIIME 2 environment
source /opt/miniconda3/etc/profile.d/conda.sh
conda activate qiime2-2024.2

# Pick otus
qiime vsearch cluster-features-de-novo \
--i-sequences q2/itsall/derep_seq.qza \
--i-table q2/itsall/derep_table.qza \
--p-perc-identity 0.97 --p-strand plus \
--p-threads 2 --output-dir q2/itsall/features
```

Filter rare features

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

# QIIME 2 command to filter rare features
qiime feature-table filter-features \
--i-table q2/itsall/features/clustered_table.qza \
--p-min-frequency 10 \
--o-filtered-table q2/itsall/features/f1_table.qza

qiime feature-table filter-seqs \
--i-data q2/itsall/features/clustered_sequences.qza \
--i-table q2/itsall/features/f1_table.qza \
--o-filtered-data q2/itsall/features/f1_sequences.qza
```

Identify chimeric sequences

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

# Detect chimeras
qiime vsearch uchime-denovo \
--i-sequences q2/itsall/features/f1_sequences.qza \
--i-table q2/itsall/features/f1_table.qza \
--o-chimeras q2/itsall/features/chimeras.qza \
--o-stats q2/itsall/features/stats.qza \
--o-nonchimeras q2/itsall/features/nonchimeras.qza
```

Filter chimeric features from the table

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

# Keep nonchimeric features in the table
qiime feature-table filter-features \
--i-table q2/itsall/features/f1_table.qza \
--m-metadata-file q2/itsall/features/nonchimeras.qza \
--o-filtered-table q2/itsall/features/otu_table.qza
```

Filter chimeric sequences

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

# Keep nonchimeric sequences
qiime feature-table filter-seqs \
--i-data q2/itsall/features/f1_sequences.qza \
--i-table q2/itsall/features/otu_table.qza \
--o-filtered-data q2/itsall/features/otu_sequences.qza
```

Import feature table and metadata to tse

```r
# Import feature table and sort sample names alphabetically
tse <- importQIIME2(featureTableFile = "q2/itsall/features/otu_table.qza")
tse <- tse[, sort(colnames(tse))]

# Import metadata file and add data to colData
metadata <- data.frame(read_tsv("meta.tsv",
show_col_types = F))
metadata <- column_to_rownames(metadata, "sampleid")
colData(tse) <- DataFrame(metadata)

# Check dimensions
tse
```

```
class: TreeSummarizedExperiment
dim: 8292 24
metadata(0):
assays(1): counts
rownames(8292): a5010f85c93eb041d28b76b98645d95d6dfaa7db
  3d4bab7221d32b82095a0ebc4cc8e45909aa1e21 ...
  40ef4e0761d396974af1ce8c18fdd6fc9aa7c559
  1dc8b1f77b4dd8807445d704b3deceab60a33885
rowData names(0):
colnames(24): Barcode001 Barcode002 ... Barcode023 Barcode024
colData names(5): Labnro Alue Kasvillisuus Soil_pH Maanäyte
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
```

Import sequences and rearrange them to correspond count table

```r
# Import squence file
ref_sequences <- importQZA("q2/itsall/features/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
```

```r
# Included ordered sequences to data object
referenceSeq(tse) <- ref_sequences_ordered
```

Classify taxonomy. This step is computionally heavy.

```r
# Unite reference file
unite <- "~/feature_classifiers/sh_general_release_dynamic_04.04.2024.fasta"
# Assign taxonomy using dada2 assignTaxonomy function
taxa <- assignTaxonomy(referenceSeq(tse),unite, minBoot=50, multithread=2)
# Save result to rds file
saveRDS(taxa, "q2/itsall_taxa.rds")
```

Process and include taxonomic results

```
# Read results
taxa <- data.frame(readRDS("q2/itsall_taxa.rds"))
# Remove taxa prefixes from each column
taxa <- as.data.frame(lapply(taxa, function(x) sub("^[a-z]__","", x)))
#Add taxonomy to rowData
rownames(taxa) <- NULL
rowData(tse) <- DataFrame(taxa)
# Rename rows
rownames(tse) <- paste0("OTU_", seq_len(nrow(tse)))
```

Prune non-fungal taxa from data

```
# Non-fungal taxa
tse <- tse[rowData(tse)$Kingdom %in% "Fungi",]
# Final dimensions
tse
```

```
class: TreeSummarizedExperiment
dim: 8292 24
metadata(0):
assays(1): counts
rownames(8292): OTU_1 OTU_2 ... OTU_8291 OTU_8292
rowData names(7): Kingdom Phylum ... Genus Species
colnames(24): Barcode001 Barcode002 ... Barcode023 Barcode024
colData names(5): Labnro Alue Kasvillisuus Soil_pH Maanäyte
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (8292 sequences)
```

Write results to files

RDS

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

Abundance table

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

Taxonomy table

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

Feature sequences

```
# Write fasta file
writeXStringSet(referenceSeq(tse), "results/itsall_repseq.fasta",
append = F, compress = F,
format = "fasta")
```

Metadata file

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

Summary table

```r
# Read data
its1 <- readRDS("results/tse_its1.rds")
its2 <- readRDS("results/tse_its2.rds")
both <- readRDS("results/tse_itsall.rds")
# Create data frame with counts information
summary_df <- data.frame(Samples = colData(its1)$Labnro,
                         ITS1 = colSums(assay(its1, "counts")),
                         ITS2 = colSums(assay(its2, "counts")),
                         Both = colSums(assay(both, "counts")))
rownames(summary_df) <- NULL
```

```
kable(summary_df, caption = "ITS sequence summary") %>%
kable_styling(latex_options = c("HOLD_position", "striped"),
font_size = 11) %>% row_spec(0, background = "teal",
color = "white")
```

Table 1: ITS sequence summary

Samples	ITS1	ITS2	Both
R08	192986	189009	198378
R16	224500	223081	232812
R07	241390	240808	252158
R37	128771	125651	132975
R33	246158	241450	252766
R13	264401	259406	271416
M06	184969	182707	190069
R38	242126	239200	240816
R05	309193	302698	319647
R34	231660	224780	217338
R32	221635	218142	210455
R04	198082	194667	197552
M04	274001	267374	261328
M07	260081	254465	258038
M01	250043	245839	242923
R21	261731	251855	260505
R18	200344	196712	202929
R40	186000	182146	188609
R42	281324	277085	279398
M05	221907	217972	223169
M02	269483	259828	261025
R31	231214	226903	222414
R23	268175	254423	266089
negative	30	29	26

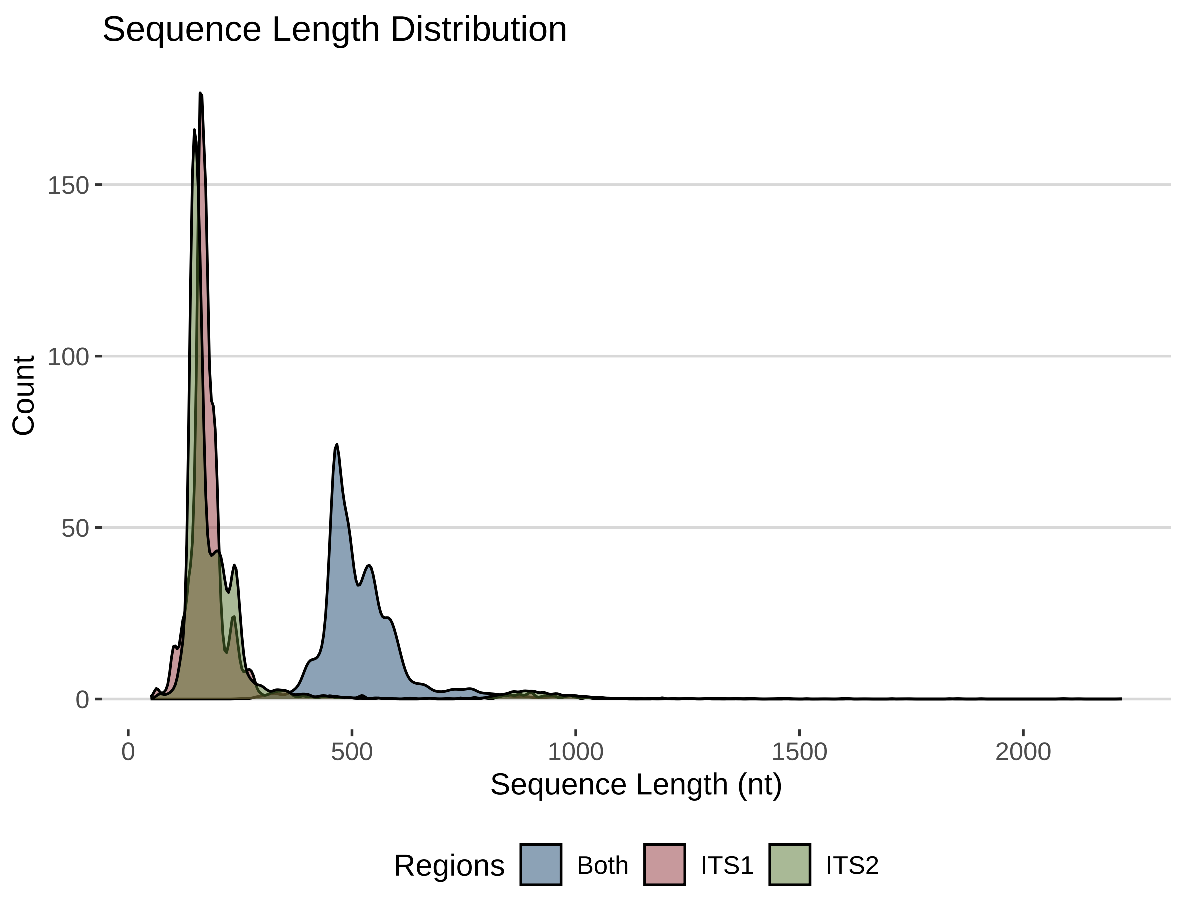
Sequence length distribution

```
# Get sequences
seqset1 <- referenceSeq(its1)
seqset2 <- referenceSeq(its2)
seqset3 <- referenceSeq(both)

# Calculate sequence lengths
sequence_lengths1 <- nchar(seqset1)
sequence_lengths2 <- nchar(seqset2)
sequence_lengths3 <- nchar(seqset3)

# Combine lengths into a single data frame in long format
its_df <- data.frame(
  Length = c(sequence_lengths1, sequence_lengths2, sequence_lengths3),
  Regions = factor(c(rep("ITS1", length(sequence_lengths1)),
                     rep("ITS2", length(sequence_lengths2)),
                     rep("Both", length(sequence_lengths3))))
)

# Plot the density distribution
ggplot(its_df, aes(x = Length, fill = Regions)) +
  geom_density(aes(y = ..count..), alpha = 0.5) +
  labs(title = "Sequence Length Distribution",
       x = "Sequence Length (nt)", y = "Count") +
  theme_hc() +
  scale_fill_stata()
```


Sequence Length Distribution
Count
Sequence Length (nt)
Regions
Both
ITS1
ITS2