The size of fungal ITS1 sequences is highly variable and can differ from one species to another. The sequence length dissimilarity as well as chimaera amplicons generated during Polymerase Chain Reaction (PCR) create bias during similarity clustering and consequently affect community abundance estimation in the downstream analysis. This script manual instructs the pre-processing and processing steps of a comparative analysis of a newly designed pipeline, The Hierarchical Clustering with Kraken (HCK), implemented with Abundance-Base Alternative Approach (ABaa) to filter chimeric sequences and classify ITS1 amplicons in fungi metabarcoding datasets versus existing Qiime version1, DADA2 and Kraken version.

The database used for reference filtering was downloaded from the following link (<https://unite.ut.ee/sh_files/sh_qiime_release_31.01.2016.zip>).

The NCBI NT, and taxdb database downloaded locally from the following link (<https://ftp.ncbi.nlm.nih.gov/blast/db/>) released on the 2020-09-14.

UNITE database (UNITE Community 2019) downloaded from <http://unite.ut.ee/repository.php> for QIIME and DADA2. In this study, we used the version 8.2 released on the 2018-11-18 containing 9407 reference sequences and 26260 representative sequences.

1. Illumina reads sequences quality check

# install fastqc and multiqc locally

*mkdir -p QC\_bf*

*fastqc -o QC\_bf -f fastq --extract raw\_data/\*.fastq*

*multiqc -o merged\_QC -f -v QC\_bf*

1. Illumina raw sequence trimming

module load trimmomatic/0.36 or install locally trimmomatic (minimum version 0.35)

*mkdir $1*

*# $1=output\_dir(trim),*

*$2=raw\_data*

*for file in $(ls $2 | sed 's/R[12]\.fastq//')*

*do*

 *trimmomatic PE -phred33 \*

*$2/${file}R1.fastq $2/${file}R2.fastq \*

 *$1/${file}R1.paired.fastq $1/${file}R1.unpaired.fastq $1/${file}R2.paired.fastq $1/${file}R2.unpaired.fastq \*

*ILLUMINACLIP:$HOME/my\_env/TruSeq3-PE.fa:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 \*

*-threads 30 \*

*MINLEN:50*

*done*

1. Merging forward and reverse sequence reads

module load pandaseq/2.11 or install locally pandaseq (minimum version 2.10)

*mkdir -p $1 ## $1=output\_dir, $2=trimmed\_fastq.gz*

*for file in $(ls $2/\*.paired.fastq | cut -f2 -d "/" | sed 's/\.R[12].paired\.fastq//');*

*do*

*pandaseq \*

*-f $2/${file}.R1.paired.fastq \*

*-r $2/${file}.R2.paired.fastq \*

*-N -l 50 -o 20 -B -t 0.9 -T 50 \*

*-A simple\_bayesian -w $1/${file}.fasta;*

*done*

1. Combining all samples fasta files into one single fasta file (demultiplexing)

*add\_label.py -i raw\_fasta -o SEQ/raw\_fasta*

1. Non-specific amplicons detection with Abaa

*abaa.py -i input\_dir -o output\_dir* ## input\_dir should contain amplicons fasta files

1. Chimera filtering with UCHIME\_REF and UCHIME\_DENOVO

*vsearch --uchime\_denovo SEQ/raw.fasta --threads 20 --nonchimeras SEQ/denovo.fasta & vsearch --uchime\_ref SEQ/raw.fasta --threads 40 --db $silva\_ref --nonchimeras SEQ/ref.fasta*

1. Blast search

*blastn -query SEQ/raw.fasta -out BLAST/All.tsv -task megablast -strand both -db $DB/nt -num\_threads 10 -evalue 1e-10 -outfmt "6 qseqid qlen qstart qend sseqid slen sstart send pident evalue qcovs bitscore staxids sskingdoms sscinames" -max\_target\_seqs 100*

1. DADA2 adapted for fungi ITS amplicons

#preparing the working environment

*args = commandArgs(trailingOnly=TRUE)*

*library(dada2)*

*packageVersion("dada2")*

*library(ShortRead)*

*packageVersion("ShortRead")*

*library(Biostrings)*

*packageVersion("Biostrings")*

*library(phyloseq)*

*packageVersion("phyloseq")*

# load reads files

*path <- args[1]*

*#list.files(path)*

#match foreward and reverse files

*fnFs <- sort(list.files(path, pattern = "R1.fastq.gz", full.names = TRUE))*

*fnRs <- sort(list.files(path, pattern = "R2.fastq.gz", full.names = TRUE))*

# Forward and reverse fastq filenames have the format:

*cutFs <- sort(list.files(path, pattern = "R1.fastq.gz", full.names = TRUE))*

*cutRs <- sort(list.files(path, pattern = "R2.fastq.gz", full.names = TRUE))*

# Extract sample names, assuming filenames have format:

*get.sample.name <- function(fname) strsplit(basename(fname), "\_")[[1]][1]*

*sample.names <- unname(sapply(cutFs, get.sample.name))*

*#head(sample.names)*

#triming

*filtFs <- file.path(path, "filtered", basename(cutFs))*

*filtRs <- file.path(path, "filtered", basename(cutRs))*

*out <- filterAndTrim(cutFs, filtFs, cutRs, filtRs, maxN = 0, maxEE = c(2, 2), truncQ = 2, minLen = 50, rm.phix = TRUE, compress = TRUE, multithread = TRUE) # on windows, set multithread = FALSE*

*errF <- learnErrors(filtFs, multithread = TRUE)*

*errR <- learnErrors(filtRs, multithread = TRUE)*

#dereplicate

*derepFs <- derepFastq(filtFs, verbose = TRUE)*

*derepRs <- derepFastq(filtRs, verbose = TRUE)*

# Name the derep-class objects by the sample names

*names(derepFs) <- sample.names*

*names(derepRs) <- sample.names*

#sample inference

*dadaFs <- dada(derepFs, err = errF, multithread = TRUE)*

*dadaRs <- dada(derepRs, err = errR, multithread = TRUE)*

#merging

*mergers <- mergePairs(dadaFs, derepFs, dadaRs, derepRs, verbose=TRUE)*

#sequence table

*seqtab <- makeSequenceTable(mergers)*

#remove chimeras

*seqtab.nochim <- removeBimeraDenovo(seqtab, method="consensus", multithread=TRUE, verbose=TRUE)*

#track reads

*getN <- function(x) sum(getUniques(x))*

*track <- cbind(out, sapply(dadaFs, getN), sapply(dadaRs, getN), sapply(mergers, getN), rowSums(seqtab.nochim))*

# If processing a single sample, remove the sapply calls: e.g. replace

*# sapply(dadaFs, getN) with getN(dadaFs)*

*colnames(track) <- c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim")*

*rownames(track) <- sample.names*

*write.table(track, "track.tsv")*

#taxonomy

*unite.ref <- args[2]*

*taxa <- assignTaxonomy(seqtab.nochim, unite.ref, multithread = TRUE, tryRC = TRUE)*

*taxa.print <- taxa # Removing sequence rownames for display only*

*rownames(taxa.print) <- NULL*

*write.table(taxa, "taxa.tsv")*

*write.table(taxa.print, "taxa\_print.tsv")*

#export otu table

*ps <- phyloseq(otu\_table(seqtab.nochim, taxa\_are\_rows=FALSE), tax\_table(taxa))*

*dna <- Biostrings::DNAStringSet(taxa\_names(ps))*

*names(dna) <- taxa\_names(ps)*

*ps <- merge\_phyloseq(ps, dna)*

*taxa\_names(ps) <- paste0("ASV", seq(ntaxa(ps)))*

*t = t(otu\_table(ps))*

*write.table(t, "otu\_table.tsv")*

*t2 = tax\_table(ps)*

*write.table(t2, "taxa\_table.tsv")*

## users can make a script out of these command lines and name it dada2.R, he will simply need to run thefollowing command

*Rsripts dada2.R arg1 (raw\_data\_dir) arg2 (Unite\_db\_dir)*

1. QIIME version 1 scripts

#unite\_db

*for i in $(ls SEQ/ | sed 's/\.fasta//'); do pick\_open\_reference\_otus.py -i SEQ/$i.fasta -r $unite\_ref -o QIIME\_$i -p $unite\_setting -f -a -O 5 -v; summarize\_taxa.py -i QIIME\_$i/otu\_table\_mc2.biom -a -o QIIME\_$i/tax -L 1,2,3,4,5,6,7; done*

#unite\_setting

pick\_otus:enable\_rev\_strand\_match True

assign\_taxonomy:reference\_seqs\_fp dir\_to\_Unite\_db

assign\_taxonomy:id\_to\_taxonomy\_fp dir\_to\_Unite\_taxa\_file

#ITSdb

*for i in $(ls SEQ/ | sed 's/\.fasta//'); do pick\_open\_reference\_otus.py -i SEQ/$i.fasta -r fungi\_QIIME\_DB.fasta.sorted -o QIIME\_$i -a -O 2 -p fungi\_setting.txt -f -v; summarize\_taxa.py -i QIIME\_$i/otu\_table\_mc2\_w\_tax.biom -a -o QIIME\_$i/tax -L 1,2,3,4,5,6,7; done*

#itsdb\_setting

*pick\_otus:enable\_rev\_strand\_match True*

*assign\_taxonomy:reference\_seqs\_fp dir\_to\_ITSdb*

*assign\_taxonomy:id\_to\_taxonomy\_fp dir\_ITSdb\_taxa\_file*

1. **Kraken analysis**

*mkdir -p $1* #create results directory

*in=$2* # fasta file

*db=$3* # db\_dir

*kraken --db $db --preload --unclassified-out $1/unclassified.fna --classified-out $1/classified.fna $in > $1/metagenome.kraken*

#generate representative report

*kraken-report $1/metagenome.kraken --db $db > $1/metagenome.report*

*ls fasta |sed 's/\.fasta//g' > $1/list.txt* # fasta is the directory containing the list of samples fasta files, the aim is to generate the sample list name

#generate kraken profile for all study samples

*cd $1*

*mkdir -p sample*

*cat list.txt | while read line; do grep $line metagenome.kraken > sample/$line.kraken; echo "just finish $line"; done*

#generate kraken report for all samples

*mkdir -p report*

*mkdir -p abundance*

*for file in $(ls sample); do d="$(echo ${file} | sed 's/kraken/report/')"; kraken-report sample/${file} --db $db > report/$d; echo "just finish $d"; done*

#plot abundance

*Rscript filter2.R report abundance*

*echo "All done...!"*

# users can make a script out of theses command lines , name it kraken.sh and run the following command:

*bash kraken.sh $1(results\_dir) $2 (input\_dir) $3(db\_dir)*

1. Alpha diversity

*alpha\_diversity.py -i QIIME/otu\_table\_mc2.biom -m chao1,shannon -o adiv\_chao1\_shan.tsv*