Plot phylogenetic tree and identify outliers

Madeleine J.S. Gundersen

2023-05-09

This R-notebook document shows how a user can investigate their phylogenetic tree and identify outliers. The outline is as follows

1. Input your phyloseq object and plot is with cirlular layout using ggtree package.
2. If outliers are present identify them
3. Give the script your fasta file and it will make a new fasta file containing only the “suspicious” sequences
4. Input this new fasta file to BLAST and download the XML report for all samples.
5. Input the XML report back into R
6. In this script the top 5 BLAST matches per OTU/ASV is kept. But you can modify this to your preference.
7. Match the accession number for each match to taxonomic information from NCBI. If all 5 matches are “Bacteria” we keep the ASV. If they are not bacteria, the program will display the BLAST information to you and you have to decide if the OTU/ASV should be removed.
8. You are given a vector containing the biased sequences and can make a new phyloseq object without these biased ASVs/OTUs.
9. Plot the tree again and see if you need to repeat.

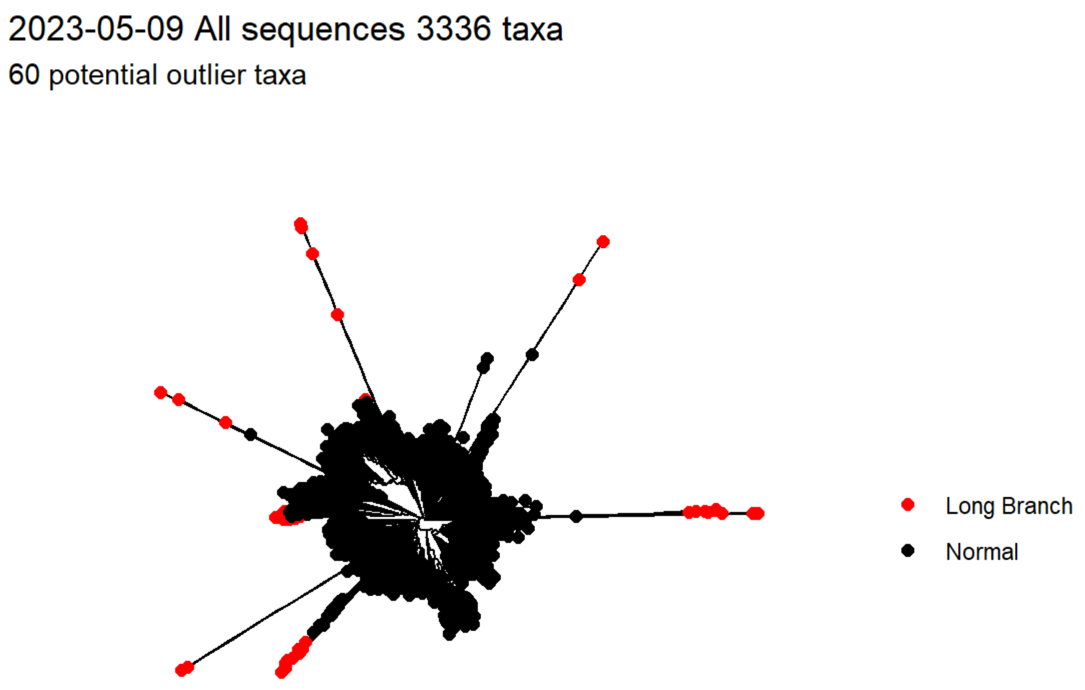
```
# load packages
library(phyloseq)
library(ggtree)
library(ggplot2)
library(tibble)
library(dplyr)
library(xml2)
library(rentrez)
library(kableExtra)
```

```
# Set filepaths to your data and for saving figures and resuls
filepath= "the_path_to_your_folder"
# e.g. "C:/Users/madel/Project_Cod/R_analysis
filepath_results = paste0(filepath,"/figures/", Sys.Date(), "_")

#load the phyloseq object
phyloseq_all_sequences = readRDS("your_phyloseq_object.rds")
fasta_file_experiment = "your_project_fasta_file.fasta"
```

In this document an experiment containing outlier sequences was used to exemplify the code and method.

Inspect the phylogenetic tree and identify outliers

```
# Name title to be displayed in plot
dataset_title = "All sequences"

# Display the tree in a circular layout using the ggtree package
ggtree_obj = ggtree::ggtree(phyloseq_all_sequences) +
  ggtree::layout_circular()

# Identify taxa with long branches
branch_cutoff_outlier = 0.32#adjust if needed
long_branch_taxa = ggtree_obj$data$label[ggtree_obj$data$branch > branch_cutoff_outlier]

# Find number and names of outlier taxa
long_branch_taxa_df = data.frame(outliers = long_branch_taxa) %>% unique() %>% na.omit()

# Highlight taxa with long branch lengths in red
# Display the updated tree
ggtree_obj +
  ggtree::geom_tippoint(aes(color = ifelse(label %in% long_branch_taxa, "Long Branch", "Norma
l")),
                        size = 2) +
  ggplot2::scale_color_manual(values = c("Normal" = "black", "Long Branch" = "red")) +
  ggplot2::theme(legend.position = "right",
                 legend.title = element_blank()) +
  ggplot2::ggtitle(label = paste0(Sys.Date(), " ", dataset_title, " ", ntaxa(phyloseq_all_sequen
ces), " taxa"),
                   subtitle = paste0(nrow(long_branch_taxa_df), " potential outlier taxa"))
```



```
# Save the plot if desired
plotname = "Phylogenetic_tree_all_sequences_with_outliers"
#ggplot2::ggsave(filename = paste0(filepath_results, plotname, ".png"), width = 10, height = 10)
```

```
# You can also filter out outliers using other indicators

## Based on taxonomy

#unclassified_bacteria = ggtree_obj$data$label[ggtree_obj$data$Phylum == "uncl_d_Bacteria"]
```

Export outlier sequences in fasta format

```
seq_data <- Biostrings::readDNAStringSet(fasta_file_experiment)
```

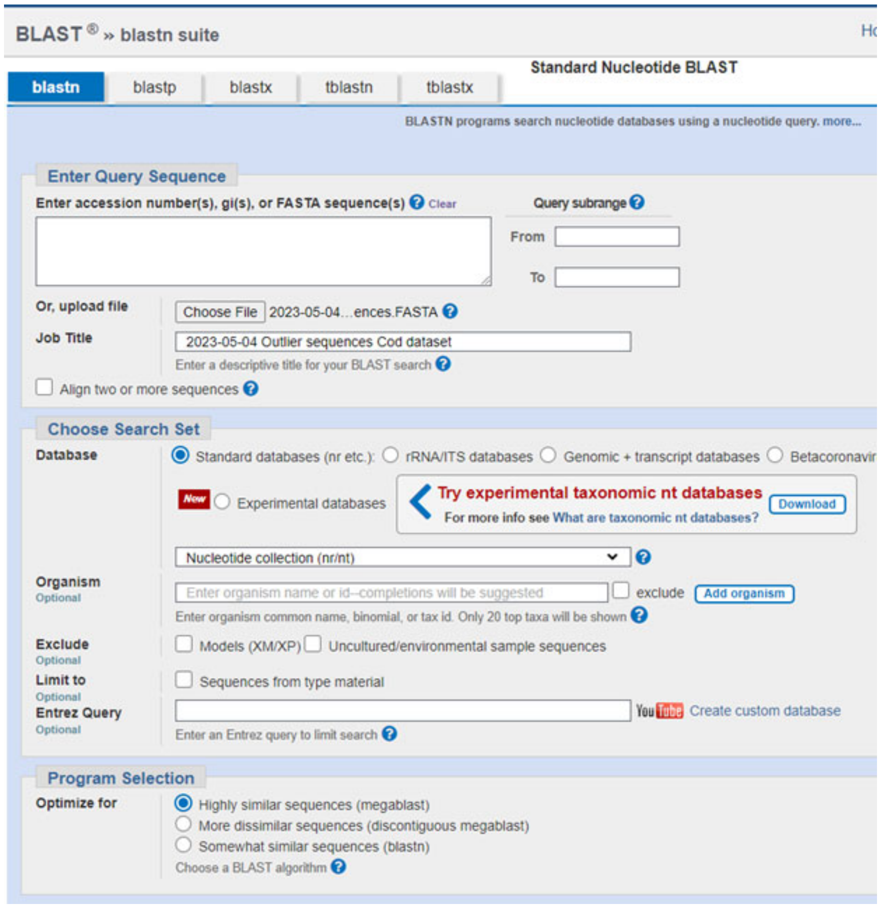
```
# Filter sequencing data to only contain outliers
outlier_seq_data <- seq_data[names(seq_data) %in% long_branch_taxa_df$outliers, ]

#export outlier sequences in fasta format for input to BLAST
fasta_file_name = "Outlier_sequences"
Biostrings::writeXStringSet(x = outlier_seq_data,
                            filepath = paste0(filepath_results,
                                              fasta_file_name, "_",
                                              length(outlier_seq_data), "sequences.fasta"))
```

```
# Create data frame with information for outlier taxa. Can come if handy if you want to check up
specific taxa
taxa_df <- data.frame(taxaID = rownames(tax_table(phyloseq_all_sequences)),
                      tax_table(phyloseq_all_sequences)) %>%
  dplyr::filter(taxaID %in% long_branch_taxa_df$outliers)

sequence_df =
data.frame(taxaID = names(outlier_seq_data),
  Sequence = as.character(outlier_seq_data))

outlier_info_df = left_join(taxa_df, sequence_df, by = "taxaID")

#save as a csv
write.csv(x = outlier_info_df, file = paste0(filepath_results, "outlier_phylogenetic_tree_ID_seq
uence.csv"))
```

Nucleotide BLAST the outlier sequences

Go to https://blast.ncbi.nlm.nih.gov/Blast.cgi (https://blast.ncbi.nlm.nih.gov/Blast.cgi) and choose “Nucleotide BLAST”.

Upload the fasta file continaing the potential outlier sequences and BLAST against Nucleotide collection (nr/nt). Try first using standard algorithm parameters.



BLAST ® » blastn suite

blastn | blastp | blastx | tblastn | tblastx

Standard Nucleotide BLAST

BLASTN programs search nucleotide databases using a nucleotide query. more...

Enter Query Sequence

Enter accession number(s), gi(s), or FASTA sequence(s) Clear

Query subrange: From ___ To ___

Or, upload file: Choose File 2023-05-04...ences.FASTA

Job Title: 2023-05-04 Outlier sequences Cod dataset

Enter a descriptive title for your BLAST search

☐ Align two or more sequences

Choose Search Set

Database: ● Standard databases (nr etc.): ○ rRNA/ITS databases ○ Genomic + transcript databases ○ Betacoronavir

New ○ Experimental databases — Try experimental taxonomic nt databases. For more info see What are taxonomic nt databases? Download

Nucleotide collection (nr/nt)

Organism (Optional): Enter organism name or id--completions will be suggested ☐ exclude Add organism

Enter organism common name, binomial, or tax id. Only 20 top taxa will be shown

Exclude (Optional): ☐ Models (XM/XP) ☐ Uncultured/environmental sample sequences

Limit to (Optional): ☐ Sequences from type material

Entrez Query (Optional): ___ YouTube Create custom database

Enter an Entrez query to limit search

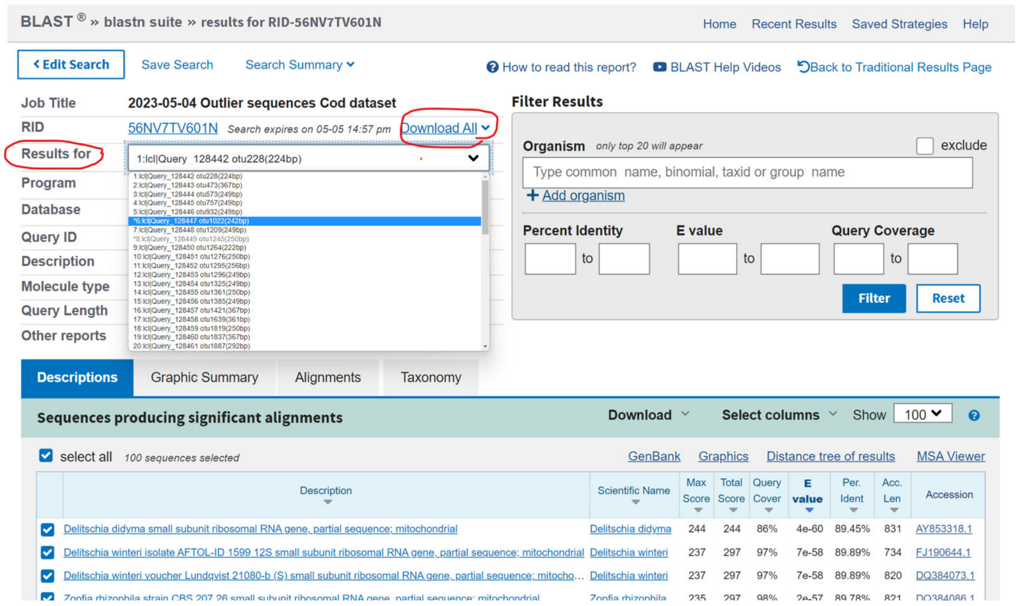
Program Selection

Optimize for: ● Highly similar sequences (megablast) ○ More dissimilar sequences (discontiguous megablast) ○ Somewhat similar sequences (blastn)

Choose a BLAST algorithm

BLAST search window

When the search is finished you will get up a result window. In this case I uploaded 60 sequences, and should get 60 Query summaries back. Check this by clicking “Results for” button. Then click “Download All” and select “XML”. The XML file should now be in your download folder. Transfer this file to your project folder and rename if wanted.



BLAST ® » blastn suite » results for RID-56NV7TV601N

Home | Recent Results | Saved Strategies | Help

< Edit Search | Save Search | Search Summary | How to read this report? | BLAST Help Videos | Back to Traditional Results Page

Job Title: 2023-05-04 Outlier sequences Cod dataset

RID: 56NV7TV601N Search expires on 05-05 14:57 pm Download All

Results for: 1:lcl|Query_128442 otu228(224bp)

```
1:lcl|Query_128442 otu228(224bp)
2:lcl|Query_128443 otu473(367bp)
3:lcl|Query_128444 otu573(249bp)
4:lcl|Query_128445 otu757(249bp)
5:lcl|Query_128446 otu932(249bp)
*6:lcl|Query_128447 otu1022(242bp)
7:lcl|Query_128448 otu1209(249bp)
*8:lcl|Query_128449 otu1245(250bp)
9:lcl|Query_128450 otu1264(222bp)
10:lcl|Query_128451 otu1276(250bp)
11:lcl|Query_128452 otu1295(256bp)
12:lcl|Query_128453 otu1296(249bp)
13:lcl|Query_128454 otu1325(249bp)
14:lcl|Query_128455 otu1361(250bp)
15:lcl|Query_128456 otu1385(249bp)
16:lcl|Query_128457 otu1421(367bp)
17:lcl|Query_128458 otu1639(361bp)
18:lcl|Query_128459 otu1819(250bp)
19:lcl|Query_128460 otu1837(367bp)
20:lcl|Query_128461 otu1887(292bp)
```

Program | Database | Query ID | Description | Molecule type | Query Length | Other reports

Filter Results

Organism (only top 20 will appear) ☐ exclude — Type common name, binomial, taxid or group name — + Add organism

Percent Identity ___ to ___ | E value ___ to ___ | Query Coverage ___ to ___

Filter | Reset

Descriptions | Graphic Summary | Alignments | Taxonomy

Sequences producing significant alignments — Download | Select columns | Show 100

✓ select all 100 sequences selected — GenBank | Graphics | Distance tree of results | MSA Viewer

	Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
✓	Delitschia didyma small subunit ribosomal RNA gene, partial sequence; mitochondrial	Delitschia didyma	244	244	86%	4e-60	89.45%	831	AY853318.1
✓	Delitschia winteri isolate AFTOL-ID 1599 12S small subunit ribosomal RNA gene, partial sequence; mitochondrial	Delitschia winteri	237	297	97%	7e-58	89.89%	734	FJ190644.1
✓	Delitschia winteri voucher Lundqvist 21080-b (S) small subunit ribosomal RNA gene, partial sequence; mitocho...	Delitschia winteri	237	297	97%	7e-58	89.89%	820	DQ384073.1
✓	Zopfia rhizophila strain CBS 207.26 small subunit ribosomal RNA gene, partial sequence; mitochondrial	Zopfia rhizophila	235	297	98%	2e-57	89.78%	821	DQ384086.1

Analyse BLAST results

Load XML files and format information to data frame structure

```r
# Load the XML file (one XML file per BLAST)
# set the directory where the XML files are located
xml_dir <- paste0(filepath)
# get the filenames of the XML files
xml_files <- list.files(xml_dir, pattern = "^.*\\.xml$")
# read the XML files into a list
xml_files = paste0(filepath, xml_files)
xml_list <- lapply(xml_files, xml2::read_xml)
```

You can inspect the XML file by clinking the file in the “Files” window to get a grasp of the structure. Here we are saving the description, accession number , and E-value

```r
# Initialize empty data frame to store results.
results_df = data.frame(matrix(ncol = 4, nrow = 0))
colnames(results_df) = c("Query", "Accession", "Match", "E_Value")

for (list_position in 1:length(xml_list)) {
  # if you have more XML files this script will loop over them
  xmlfile = xml_list[[list_position]]

  # Get all query nodes (one node per taxa)
  query_nodes <-
    xml2::xml_find_all(xmlfile, "//BlastOutput_iterations/Iteration")

  # Loop through query nodes and extract information
  for (i in 1:length(query_nodes)) {
    # Get query name
    query_name = xml2::xml_find_first(query_nodes[i],
                     ".//Iteration_query-def") %>%
      xml2::xml_text()

    # Get top 5 matches
    hit_nodes <- xml_find_all(query_nodes[i], ".//Hit") %>%
      head(5) #choose desired matches to keep

    # Extract match information
    for (j in 1:length(hit_nodes)) {
      match_name = xml2::xml_find_first(hit_nodes[j], ".//Hit_def") %>% xml2::xml_text()
      accession_id = xml2::xml_find_first(hit_nodes[j], ".//Hit_accession") %>% xml2::xml_text()
      evalue = xml2::xml_find_first(hit_nodes[j], ".//Hsp_evalue") %>% xml2::xml_text()
      if (length(accession_id) == 0) {
        accession_id = "no BLAST matches_check up"
        match_name = "no BLAST matches_check up"
        evalue = "no BLAST matches_check up"}
    #Save information in result_df
      results_df = base::rbind(results_df,
                              data.frame(Query = query_name,
                                         Accession = accession_id,
                                         Match = match_name,
                                         E_Value = evalue  ))}}}
```

Query	Accession	Match	E_Value
otu228	AY853318	Delitschia didyma small subunit ribosomal RNA gene, partial sequence; mitochondrial	4.06238e-60
otu228	FJ190644	Delitschia winteri isolate AFTOL-ID 1599 12S small subunit ribosomal RNA gene, partial sequence; mitochondrial	6.79779e-58
otu228	DQ384073	Delitschia winteri voucher Lundqvist 21080-b (S) small subunit ribosomal RNA gene, partial sequence; mitochondrial	6.79779e-58
otu228	DQ384086	Zopfia rhizophila strain CBS 207.26 small subunit ribosomal RNA gene, partial sequence; mitochondrial	2.44492e-57

Query	Accession	Match	E_Value
otu228	KT225540	Lophium mytilinum isolate AFTOL-ID 1609 12S ribosomal RNA gene, partial sequence; mitochondrial	1.47146e-54
otu473	KY980331	Caecitellus paraparvulus isolate BH56_230 small subunit ribosomal RNA gene, partial sequence	0
otu473	KY980321	Caecitellus paraparvulus isolate BH56_212 small subunit ribosomal RNA gene, partial sequence	0
otu473	KY980317	Caecitellus paraparvulus isolate BH56_205 small subunit ribosomal RNA gene, partial sequence	0
otu473	KY980271	Caecitellus paraparvulus isolate BH56_108 small subunit ribosomal RNA gene, partial sequence	0
otu473	KY980264	Caecitellus paraparvulus isolate BH56_74 small subunit ribosomal RNA gene, partial sequence	0
otu573	KY124855	Uncultured bacterium clone A494_Bac213 16S ribosomal RNA gene, partial sequence	1.95201e-123
otu573	KP937587	Uncultured bacterium clone OTU95932_AL220_211882 16S ribosomal RNA gene, partial sequence	4.25535e-115
otu573	KP909788	Uncultured bacterium clone 4368878_AL223_1289544 16S ribosomal RNA gene, partial sequence	4.25535e-115
otu573	KR846765	Uncultured bacterium clone OTU_15481 16S ribosomal RNA gene, partial sequence	9.21121e-112
otu573	KU115717	Uncultured bacterium clone 19455 16S ribosomal RNA gene, partial sequence	9.21121e-112

Collect taxonomic information from NCBI

```r
# create an empty dataframe to store taxonomy information
taxonomy_df <- data.frame(Organism = character(),
                          taxonomy = character(),
                          Accession = character(),
                          stringsAsFactors = FALSE)

# save the accession numbers in a dataframe and remove duplicate numbers
Accession_IDs = results_df %>% select(Accession) %>% unique()

# match the accession numbers with taxonomic information
for (i in 1:nrow(Accession_IDs)) {
  # skip if Accession is missing
  if (Accession_IDs$Accession[i] == "no BLAST matches_check up") {
    taxonomy_df <- rbind(taxonomy_df, data.frame(Organism = "no BLAST matches_check up",
                                                 taxonomy = "no BLAST matches_check up",
                                                 Accession = "no BLAST matches_check up",
                                                 stringsAsFactors = FALSE))
    next
  }
  # use efetch to retrieve the record from GenBank
  gb = rentrez::entrez_fetch(db = "nuccore", id = Accession_IDs$Accession[i], rettype = "XML")
  # convert the XML object to a list
  xml_list = XML::xmlToList(gb)
  # extract the required information
  organism = xml_list$GBSeq$GBSeq_organism
  taxonomy = xml_list$GBSeq$GBSeq_taxonomy
  accession = xml_list$GBSeq$`GBSeq_primary-accession`
  # create a data frame
  taxonomy_df = rbind(taxonomy_df, data.frame(Organism = organism,
                                               taxonomy = taxonomy,
                                               Accession = accession,
                                               stringsAsFactors = FALSE))}

# merge taxonomy information with results dataframe
outliers_with_accession = left_join(results_df, taxonomy_df, by = "Accession")
saveRDS(object = outliers_with_accession, file = paste0(filepath_results, "outliers_BLASTed_with
accession.RDS"))
```

```r
outliers_with_accession = readRDS(paste0(filepath,"2023-05-04_outliers_BLASTed_withaccession.RD
S"))
# Check that all outliers have been analyed
# The numer of queries should be the same as possible outliers
outliers_with_accession %>% select(Query) %>% unique() %>% nrow()
```

```
## [1] 60
```

Quality control of outlier taxa

```
# Load quality_control script
source(file = paste0(filepath, "quality_control_sequences.R"))

# COPY the command below into the Console window and make a decition for each taxa
quality_control_decition = quality_control(df = outliers_with_accession)
outliers_with_accession_decition = dplyr::left_join(outliers_with_accession, quality_control_dec
ition, by = "Query")
saveRDS(outliers_with_accession_decition, file = paste0(filepath_results, "Quality_control_decit
ion.RDS"))
```

After running the script we now have a decition for each outlier taxa

```
outliers_with_accession_decition[1:15,] %>%
  kbl() %>%
  kable_paper("hover", full_width = F)
```

Query	Accession	Match	E_Value	Organism	taxonomy	Decition
otu228	AY853318	Delitschia didyma small subunit ribosomal RNA gene, partial sequence; mitochondrial	4.06238e-60	Delitschia didyma	Eukaryota; Fungi; Dikarya; Ascomycota; Pezizomycotina; Dothideomycetes; Pleosporomycetidae; Pleosporales; Delitschiaceae; Delitschia	OUTLIER
otu228	FJ190644	Delitschia winteri isolate AFTOL-ID 1599 12S small subunit ribosomal RNA gene, partial sequence; mitochondrial	6.79779e-58	Delitschia winteri	Eukaryota; Fungi; Dikarya; Ascomycota; Pezizomycotina; Dothideomycetes; Pleosporomycetidae; Pleosporales; Delitschiaceae; Delitschia	OUTLIER
otu228	DQ384073	Delitschia winteri voucher Lundqvist 21080-b (S) small subunit ribosomal RNA gene, partial sequence; mitochondrial	6.79779e-58	Delitschia winteri	Eukaryota; Fungi; Dikarya; Ascomycota; Pezizomycotina; Dothideomycetes; Pleosporomycetidae; Pleosporales; Delitschiaceae; Delitschia	OUTLIER
otu228	DQ384086	Zopfia rhizophila strain CBS 207.26 small subunit ribosomal RNA gene, partial sequence; mitochondrial	2.44492e-57	Zopfia rhizophila	Eukaryota; Fungi; Dikarya; Ascomycota; Pezizomycotina; Dothideomycetes; Dothideomycetes incertae sedis; Zopfiaceae; Zopfia	OUTLIER
otu228	KT225540	Lophium mytilinum isolate AFTOL-ID 1609 12S ribosomal RNA gene, partial sequence; mitochondrial	1.47146e-54	Lophium mytilinum	Eukaryota; Fungi; Dikarya; Ascomycota; Pezizomycotina; Dothideomycetes; Pleosporomycetidae; Mytilinidiales; Mytilinidiaceae; Lophium	OUTLIER
otu473	KY980331	Caecitellus paraparvulus isolate BH56_230 small subunit ribosomal RNA gene, partial sequence	0	Caecitellus paraparvulus	Eukaryota; Sar; Stramenopiles; Bigyra; Opalozoa; Bicosoecida; Caecitellus	OUTLIER

Query	Accession	Match	E_Value	Organism	taxonomy	Decition
otu473	KY980321	Caecitellus paraparvulus isolate BH56_212 small subunit ribosomal RNA gene, partial sequence	0	Caecitellus paraparvulus	Eukaryota; Sar; Stramenopiles; Bigyra; Opalozoa; Bicosoecida; Caecitellus	OUTLIER
otu473	KY980317	Caecitellus paraparvulus isolate BH56_205 small subunit ribosomal RNA gene, partial sequence	0	Caecitellus paraparvulus	Eukaryota; Sar; Stramenopiles; Bigyra; Opalozoa; Bicosoecida; Caecitellus	OUTLIER
otu473	KY980271	Caecitellus paraparvulus isolate BH56_108 small subunit ribosomal RNA gene, partial sequence	0	Caecitellus paraparvulus	Eukaryota; Sar; Stramenopiles; Bigyra; Opalozoa; Bicosoecida; Caecitellus	OUTLIER
otu473	KY980264	Caecitellus paraparvulus isolate BH56_74 small subunit ribosomal RNA gene, partial sequence	0	Caecitellus paraparvulus	Eukaryota; Sar; Stramenopiles; Bigyra; Opalozoa; Bicosoecida; Caecitellus	OUTLIER
otu573	KY124855	Uncultured bacterium clone A494_Bac213 16S ribosomal RNA gene, partial sequence	1.95201e-123	uncultured bacterium	Bacteria; environmental samples	KEEP
otu573	KP937587	Uncultured bacterium clone OTU95932_AL220_211882 16S ribosomal RNA gene, partial sequence	4.25535e-115	uncultured bacterium	Bacteria; environmental samples	KEEP
otu573	KP909788	Uncultured bacterium clone 4368878_AL223_1289544 16S ribosomal RNA gene, partial sequence	4.25535e-115	uncultured bacterium	Bacteria; environmental samples	KEEP
otu573	KR846765	Uncultured bacterium clone OTU_15481 16S ribosomal RNA gene, partial sequence	9.21121e-112	uncultured bacterium	Bacteria; environmental samples	KEEP
otu573	KU115717	Uncultured bacterium clone 19455 16S ribosomal RNA gene, partial sequence	9.21121e-112	uncultured bacterium	Bacteria; environmental samples	KEEP

```r
outliers_remove = outliers_with_accession_decition %>% dplyr::filter(Decition == "OUTLIER") %>%
select(Query) %>% unique()
# Convert taxonomy table to data frame and add taxa id as a column
taxa_df <- data.frame(taxaID = rownames(tax_table(phyloseq_all_sequences)), tax_table(phyloseq_a
ll_sequences))
tax_table(phyloseq_all_sequences) = tax_table(as.matrix(taxa_df))

# Remove outliers from phyloseq object
quality_ps = subset_taxa(physeq = phyloseq_all_sequences, !taxaID %in% outliers_remove$Query)
quality_ps = filter_taxa(quality_ps, function(x) sum(x) > 0, prune = TRUE)
#saveRDS(quality_ps, file = paste0(filepath_results, "phyloseq_quality1_3308taxa.RDS") )
```

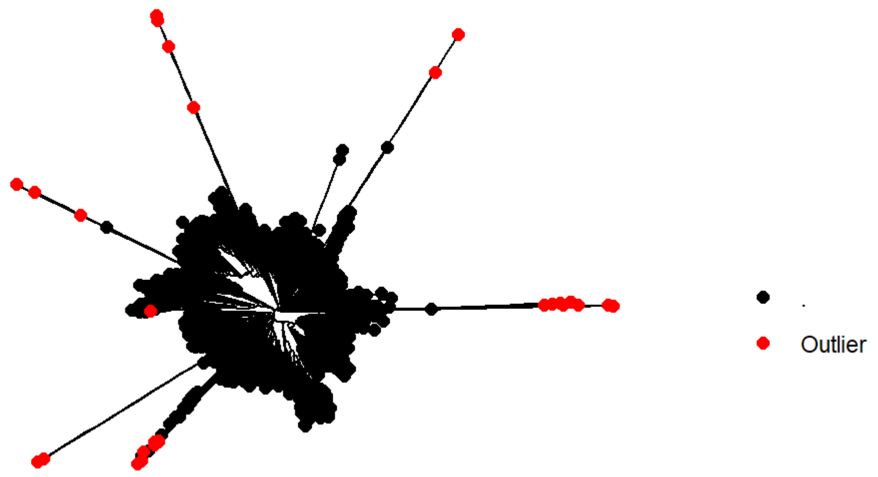
Plot new phylogenetic tree and filter out outliers

Finally, we remove the outliers from the phyloseq object and plot the new phylogenetic tree to evaluate if we need to repeat the process.

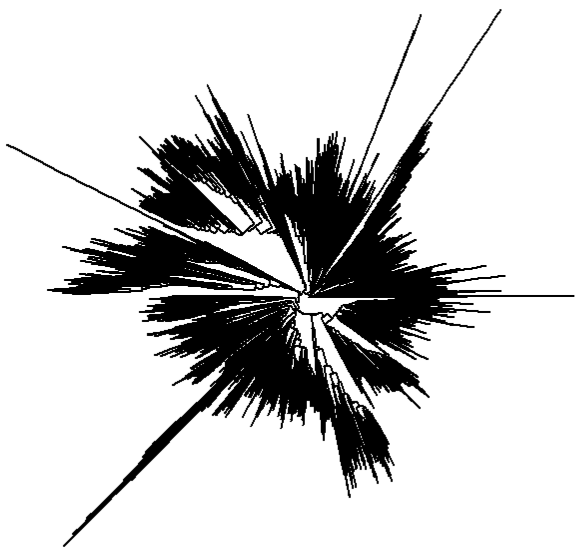
```
removed_tree=
ggtree(phyloseq_all_sequences) + layout_circular() +
  ggtree:: geom_tippoint(aes(color = ifelse(label %in% outliers_remove$Query, "Outlier", ".")),
size = 2) +
  ggplot2::scale_color_manual(values = c("." = "black", "Outlier" = "red")) +
  ggplot2::theme(legend.position = "right",
                 legend.title = element_blank()) +
  ggplot2::ggtitle(label = paste0(Sys.Date(), " Full dataset ", ntaxa(phyloseq_all_sequences), "
taxa"),
                   subtitle = paste0(nrow(outliers_remove), " outlier taxa"))
quality_tree =
ggtree(quality_ps) + layout_circular() +
  ggplot2::theme(legend.position = "right",
                 legend.title = element_blank()) +
  ggplot2::ggtitle(label = paste0(Sys.Date(), " Without outliers ", ntaxa(quality_ps), " taxa"))

ggpubr::ggarrange(removed_tree,quality_tree,nrow = 2)
```

2023-05-09 Full dataset 3336 taxa

28 outlier taxa



2023-05-09 Without outliers 3308 taxa



```
#ggsave(filename = paste0(filepath_results, #"removed_outliers_and_new_tree.png"), width = 6, he
ight = 10)
```

In this example it can be advisable to repeat the process as some taxa still are deviating from the phylogenetic tree.