# Installation

## Step 1: Create conda environment with required packages and software

mamba create -n REPGENR -y python matplotlib drep checkm-genome ncbi-datasets-cli mashtree ete3

## Step 2: Activate the environment

conda activate REPGENR

## Step 3: Download RepGenR and install using pip:

git clone <https://github.com/FOI-Bioinformatics/repgenr>

(or download in browser from <https://github.com/FOI-Bioinformatics/repgenr/archive/refs/heads/main.zip> and unpack repgenr script directory)

pip install repgenr

## Step 4: Manual corrections needed

### Export data path for Checkm

conda env config vars set CHECKM\_DATA\_PATH=<path to checkm\_data folder inside the environment, e.g.: ../anaconda\_path/envs/REPGENR/checkm\_data>

### Library-file for libgsl.so.25 (use conda config --set channel\_priority strict before testing the solution below)

If you get error during dRep (fastANI step?) requiring libgsl.so.25, you can copy or softlink a higher version of the file, at ../anaconda\_path/envs/REPGENR/lib.

e.g. soft-link .27 as .25: ln -s libgsl.so.27 libgsl.so.25

This is no longer the case since changing procedure of installation.

# Inputs

* Name of species/Genus/Family
* Working directory

# Outputs

metadata\_level.txt 🡨 stores the specified output taxonomic level  
metadata\_selected.tsv 🡨 metadata for output samples  
metadata\_summary.tsv 🡨 summary of samples per taxonomic level  
metadata\_summary\_number\_in\_level.png 🡨 bar-plot of sample abundance at each taxonomic level  
metadata\_summary\_number\_per\_level.png 🡨 bar-plot of taxonomic level abundance  
outgroup\_accession.txt 🡨 NCBI accession number of outgroup  
genomes 🡨 folder of downloaded genomes, formatted as family\_genus\_species\_NCBIaccession  
outgroup 🡨 folder of downloaded outgroup, formatted as family\_genus\_species\_NCBIaccession  
ncbi\_acc\_download\_list.txt 🡨 list of downloaded genomes as NCBIaccession  
genomes\_derep\_representants 🡨 folder of dereplicated genome representatives  
derep\_parameters.txt 🡨 saves parameters used during dereplication (e.g., ANI thresholds)  
genomes\_derep\_representants.dnd 🡨 phylogenetic tree (newick) of cluster-representatives  
derep\_chunks\_clustered\_genomes.tsv 🡨 dereplication information, see next (if chunking dataset)  
derep\_clustered\_genomes.tsv 🡨 cluster-representing and cluster-contained datasets  
derep\_genomes\_tree2tax.tsv 🡨 parent-child relations of cluster-representatives phylogeny  
derep\_genomes\_map.tsv 🡨 accession-to-representative map

# Usage

The software is a workflow of modules that populates the “Work directory” folder.

Main script (wrapper): **repgenr**



## Modules

### metadata



The metadata module fetches the GTDB metadata table according to input criteria. It will output the NCBI accession numbers for all samples at requested taxonomic level. A random outgroup sample is selected at one taxonomic level above the specified taxonomic level. The outgroup sample is used in a later module for phylogeny to infer placement into the database. Optionally, the outgroup may be user-specified as an NCBI-accession number.

Metadata information can be found in the work-directory specified to the software:

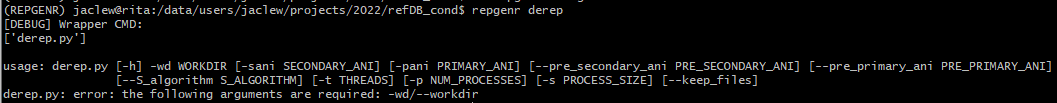
metadata\_level.txt 🡨 stores the specified output taxonomic level  
metadata\_selected.tsv 🡨 metadata for output samples  
metadata\_summary.tsv 🡨 summary of samples per taxonomic level  
metadata\_summary\_number\_in\_level.png 🡨 bar-plot of sample abundance at each taxonomic level  
metadata\_summary\_number\_per\_level.png 🡨 bar-plot of taxonomic level abundance  
outgroup\_accession.txt 🡨 NCBI accession number of outgroup

### genome



The genome module will read the accession numbers obtained through the metadata command and download them from NCBI. It produces a folder named “genomes” in where all downloaded genomes appear, formatted as “family”\_”genus”\_”species”\_”NCBI-accession-number”.fasta. A list of downloaded accessions is stored in “ncbi\_acc\_download\_list.txt.” The outgroup sample is downloaded to folder “outgroup.”

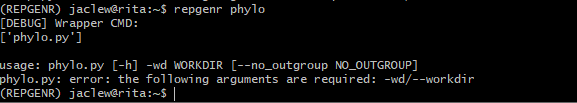
### derep



The derep module clusters the downloaded genomes based on average nucleotide identity (ANI) using the dRep software. The user may optionally change the ANI for final clustering (SECONDARY\_ANI, default=0.99) using a sensitive alignment strategy and for primary clustering (PRIMARY\_ANI, default=0.9) using a rough alignment-free strategy. The algorithm used to perform ANI-comparison can be controlled through S\_ALGORITHM (default: fastANI). The derep module requires great computation effort and may take multiple hours to finish. Large datasets can be split into chunks that are input to dRep as separate processes to reduce the number of all-vs-all comparisons. In a second step, dRep is executed on the output of the separate processes. This strategy reduces computation time for large datasets. The chunk size is controlled by PROCESS\_SIZE (a dataset of 2500 genomes can be divided to 5 chunks by setting PROCESS\_SIZE to 500). The number of chunks to compute in parallel is specified by NUM\_PROCESSES. The total number of threads used across processes is determined by THREADS. If desired, the ANI-thresholds for dRep primary and secondary clustering can be individually controlled for the chunk-processes (PRE\_SECONDARY\_ANI and PRE\_PRIMARY\_ANI – by default these parameters are SECONDARY\_ANI and PRIMARY\_ANI, respectively)

The derep module produces a folder named “genomes\_derep\_representants” where the representative genomes for each cluster is found. Contained genomes for each cluster is found in the file “derep\_clustered\_genomes.tsv” where column 1 holds the cluster-representative sequence and column 2 holds the cluster-contained sequences. If a large dataset was split into chunks an additional file “derep\_chunks\_clustered\_genomes.tsv” is produced that holds the cluster-representative sequence (column 1), cluster-contained sequence (column 2), and which chunk processed the data (column 3). The parameters used during dereplication is saved in the text-file named “derep\_parameters.txt” that includes ANI-thresholds and chunk size. The working-directory (“dereplication\_workdir”) of the derep module is removed by default but can be kept by specifying the KEEP\_FILES parameter.

### phylo



The phylo module computes a phylogenetic tree based on the representative genome sequences generated by the derep module. Optionally, the outgroup can be ignored when creating the tree. The module outputs the tree in newick format to the file “genomes\_derep\_representants.dnd.”

### tree2tax



The tree2tax module produces modification-files that can be input to FlexTaxD to modify an existing database. Node basenames that will be enumerate may be specified by the user, resulting in node naming like so: <basename>\_1,…,<basename>\_N. If unspecified, each node receives a MD5 hash based on all its’ descending leaves and thus produces unique node names in any non-redundant database. If the output is going to replace a branch in the database, the parent node for that branch should be specified as the root using the –r parameter. If the –remove\_outgroup parameter is specified the outgroup is removed from the output.

The parent-child representation of phylogeny is found in file “derep\_genomes\_tree2tax.tsv” and the path to genome files is found in file “derep\_genomes\_map.tsv.”

# Example: Francisellaceae

Suppose we want to create a database for Francisellaceae using representative-genomes in GTDB but we want to increase the resolution of *Francisella tularensis* species. We will use FlexTaxD to create a database and later modify it.

## Initiating the Francisellaceae database

Download metadata for *Francisellaceae* representative species using repgenr:  
>> repgenr metadata -r 207.0 -v bac120 -d rep -l family -tf Francisellaceae -wd francisellaceae\_repset

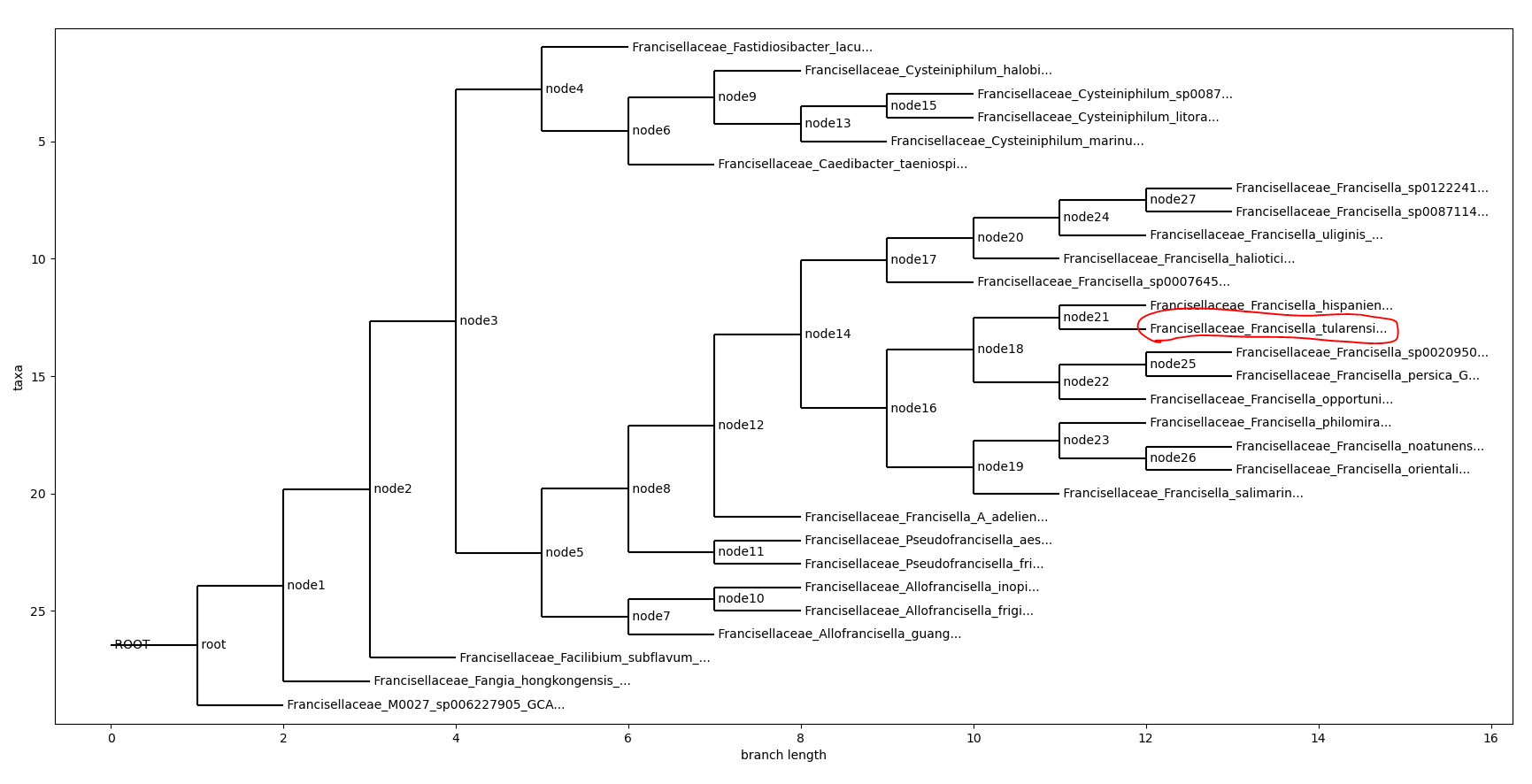
Download the genomes:  
>> repgenr genome -wd francisellaceae\_repset

Compute phylogeny on downloaded genomes (without dereplication) by supplying --all\_genomes:  
>> repgenr phylo -wd francisellaceae\_repset –all\_genomes

Produce parent-child and genome files of phylogeny for FlexTaxD (removing outgroup and specifying root name):  
>> repgenr tree2tax -wd francisellaceae\_repset –all\_genomes –-node\_basename node –-root\_name root --remove\_outgroup

Compile database using FlexTaxD:  
>> flextaxd -db francisellaceae.db --taxonomy\_file francisellaceae\_repset/genomes\_tree2tax.tsv --genomeid2taxid francisellaceae\_repset/genomes\_map.tsv

Visualize database:  
>> flextaxd -db francisellaceae.db --vis\_type tree --visualise\_node root --vis\_depth 0

  
Phylogenetic tree of *Francisellaceae* with *tularensis*-species indicated in red. Below we will replace the *tularensis*-species to expand the genetic resolution using an average nucleotide identity-based strategy.

## Modifying the database to increase the resolution of the *tularensis*-branch

To increase tree resolution of *tularensis* species, we download metadata using repgenr:  
>> repgenr metadata -r 207.0 -v bac120 -d all –l species -tg Francisella –ts tularensis -wd tularensis

Download genomes:  
>> repgenr genome -wd tularensis

(Started 12.09, ended 14.12; runtime = 2h03m) Perform dereplication at 0.99 average nucleotide identity (ANI) and splitting the dataset (N=860 genomes) into three chunks (300+300+260) that are run in parallel (all three) at a total of 70 threads – to increase performance (possibly at a slightly reduced accuracy):  
>> repgenr derep -wd tularensis -sani 0.99 --process\_size 300 --num\_processes 3 --threads 70

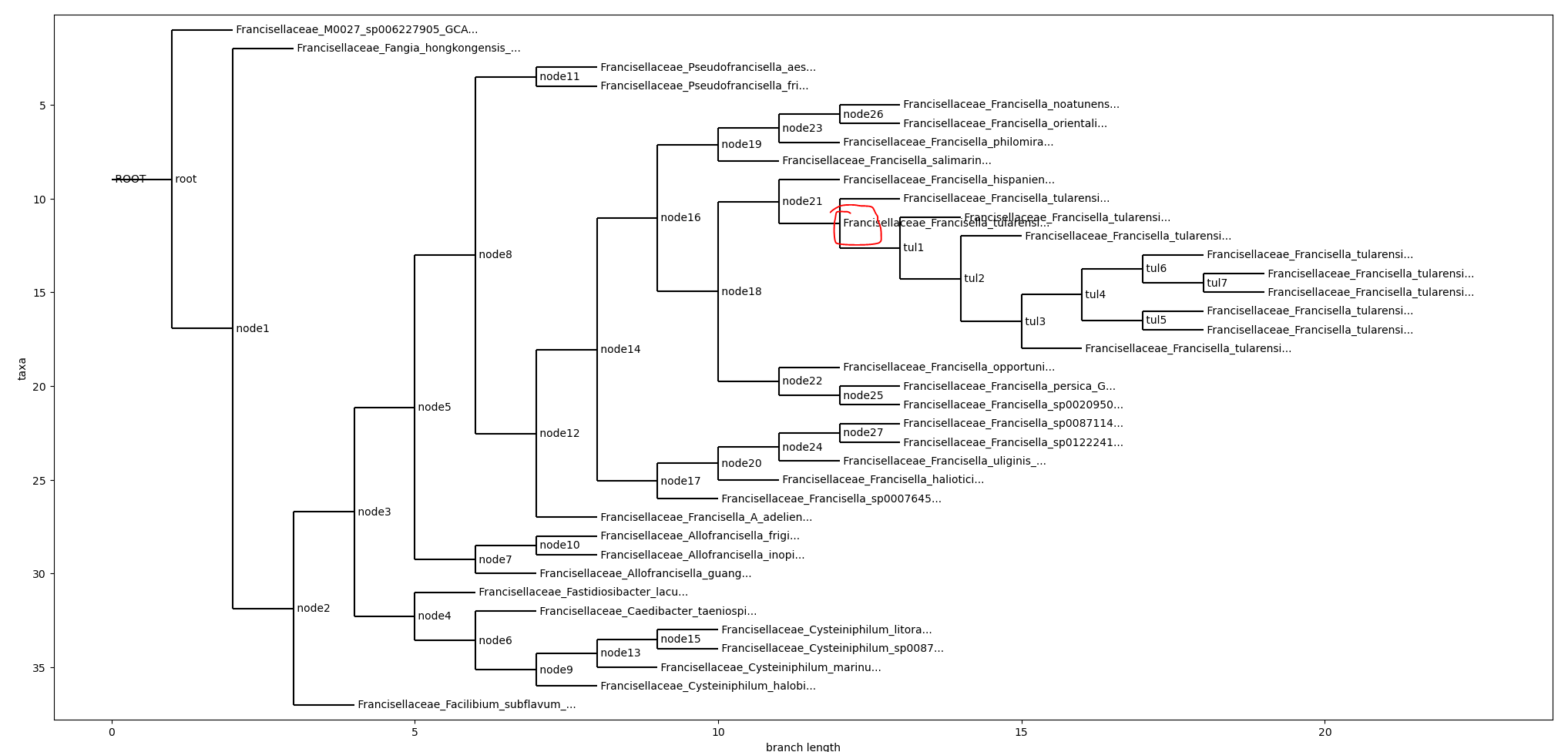
* (Started X, ended X, runtime=XhXm) Alternative single-process command:  
  >> repgenr derep -wd tularensis -sani 0.99 --threads 70

Compute phylogeny on dereplicated genomes:  
>> repgenr phylo –wd tularensis

Produce parent-child and genome files of phylogeny for FlexTaxD (defining root as the node in our *Francisellaceae* database to expand – in this case the *tularensis* species):  
>> repgenr tree2tax -wd tularensis –-node\_basename tul –-root\_name Francisellaceae\_Francisella\_tularensis\_GCF\_000008985.1 --remove\_outgroup

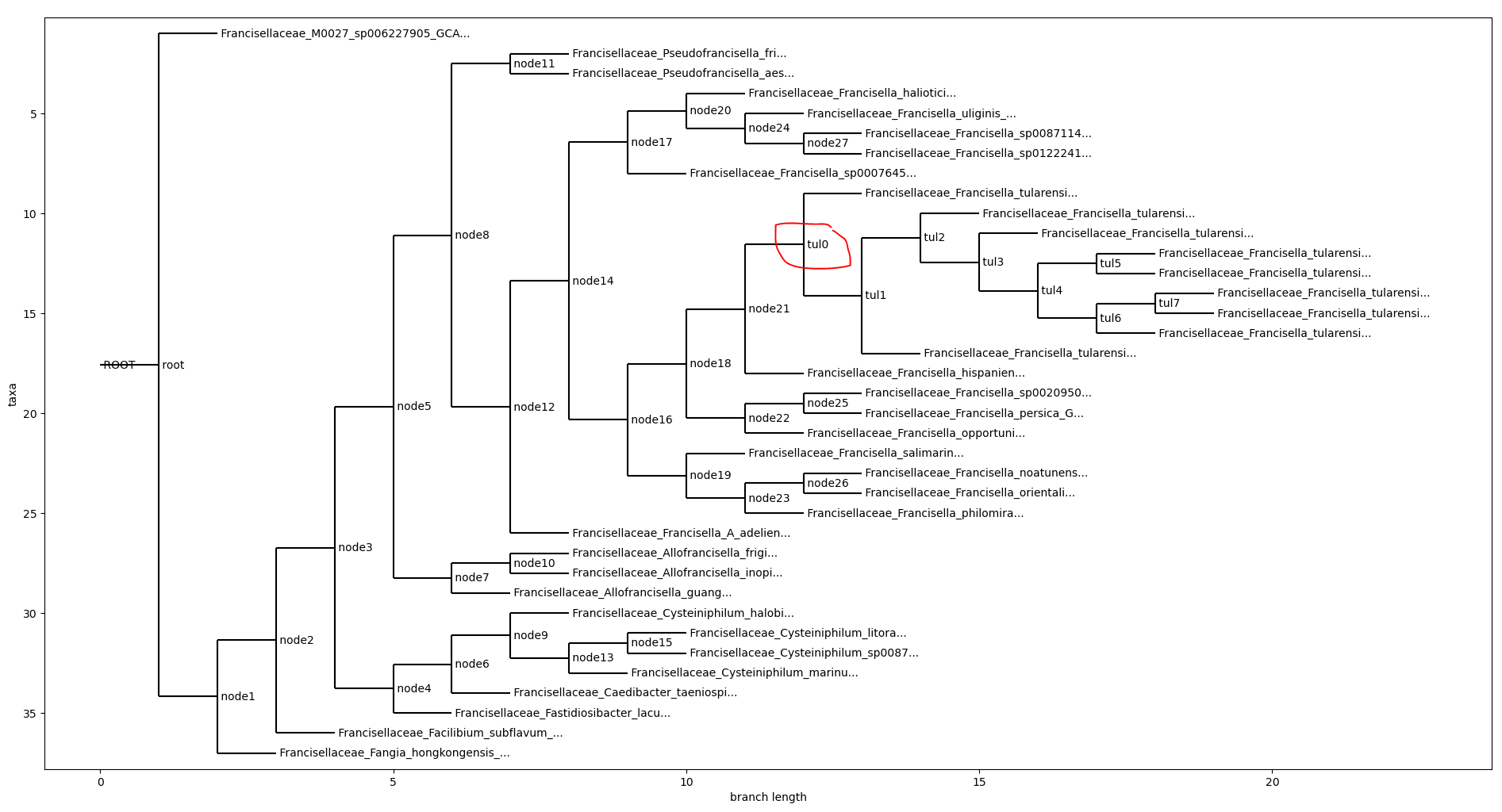
Modify *Francisellaceae* database using FlexTaxD, specifying the node to replace:  
>> flextaxd -db francisellaceae.db --mod\_file tularensis/derep\_genomes\_tree2tax.tsv --genomeid2taxid tularensis/derep\_genomes\_map.tsv --replace --parent Francisellaceae\_Francisella\_tularensis\_GCF\_000008985.1

Visualize database:  
>> flextaxd -db francisellaceae.db --vis\_type tree --visualise\_node root --vis\_depth 0

  
Phylogenetic tree of *Francisellaceae* with *tularensis*-branch expanded. The red marking indicates where the representative *tularensis*-dataset has been replaced by our ANI-based dereplication method (nodes with basename “tul”) to increase the resolution of this species.

Rename node using FlexTaxD (requires update of FlexTaxD, this is a new function):  
>> flextaxd -db francisellaceae.db –rename\_from Francisellaceae\_Francisella\_tularensis\_GCF\_000008985.1 --rename\_to tul0

Visualize database after renaming:  
>> flextaxd -db francisellaceae.db --vis\_type tree --visualise\_node root --vis\_depth 0

  
Showing the renaming of *tularensis* local root.