# Installation

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

mamba create -n REPGENR -y python=3.9 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

# 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  
drep\_workdir 🡨 dRep software working directory  
genomes\_derep\_representants 🡨 folder of dereplicated genome representatives  
derep\_ANI\_threshold.txt 🡨 average nucleotide identity used for cluster generation  
genomes\_derep\_representants.dnd 🡨 phylogenetic tree (newick) of cluster-representatives  
drep\_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:

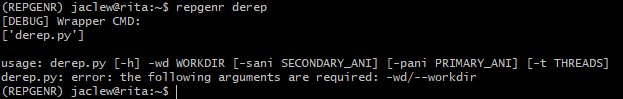
metadata\_level.txt 🡨 stores the specified output taxonomic level  
metadata\_selected.tsv 🡨 metadata for output samples  
metadata\_summary.tsv 🡨 summary of samples per taxonomic level  
metadata\_summary\_number\_in\_level.png 🡨 bar-plot of sample abundance at each taxonomic level  
metadata\_summary\_number\_per\_level.png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

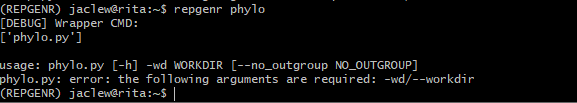
### derep



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 derep module requires great computation effort and may take multiple hours to finish.

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 “drep\_clustered\_genomes.tsv” where column 1 holds the cluster-representative sequence and column 2 holds the cluster-contained sequences. The specified ANI-threshold for the clustering is saved in the text-file named “derep\_ANI\_threshold.txt.” The folder “drep\_workdir” is a working-directory for the dRep software.

### 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.

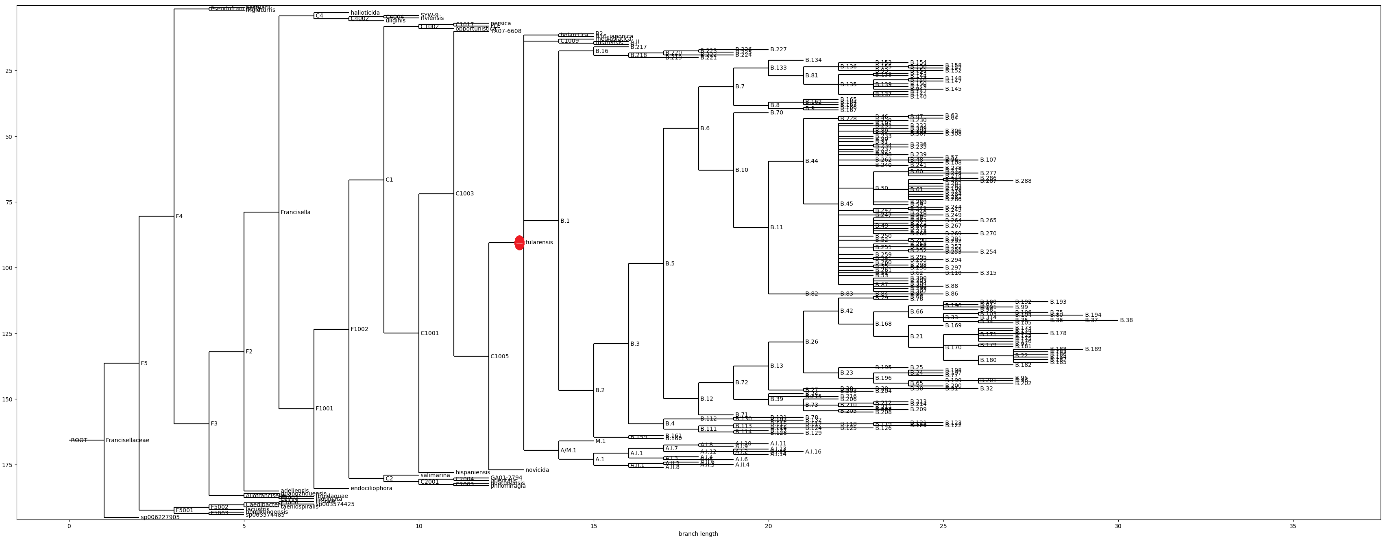
The parent-child representation of phylogeny is found in file “derep\_genomes\_tree2tax.tsv” and the path to genome files is found in file “derep\_genomes\_map.tsv.”

# Example: Francisella tularensis sub-group

Suppose we have a database representing *Francisellaceae* and want to replace the *tularensis-*branch of the database with an average nucleotide identity-based clustering strategy.

## Step 1: Visualize database before modification

We can visualize the original database using FlexTaxD (flextaxd -db francisellaceae.db --vis\_type tree --vis\_depth 0 --visualise\_node Francisellaceae).

  
Figure 1. Database of *Francisellaceae* generated from FlexTaxD. *Tularensis*-branch is indicated by a red dot.

## Step 2: Run repGenR-software

### a. Downloading metadata (including outgroup)

repgenr metadata -r 207.0 -v bac120 -d all -l species -tg francisella -ts tularensis -wd tularensis

### b. Downloading genomes (including outgroup)

repgenr genome –wd tularensis

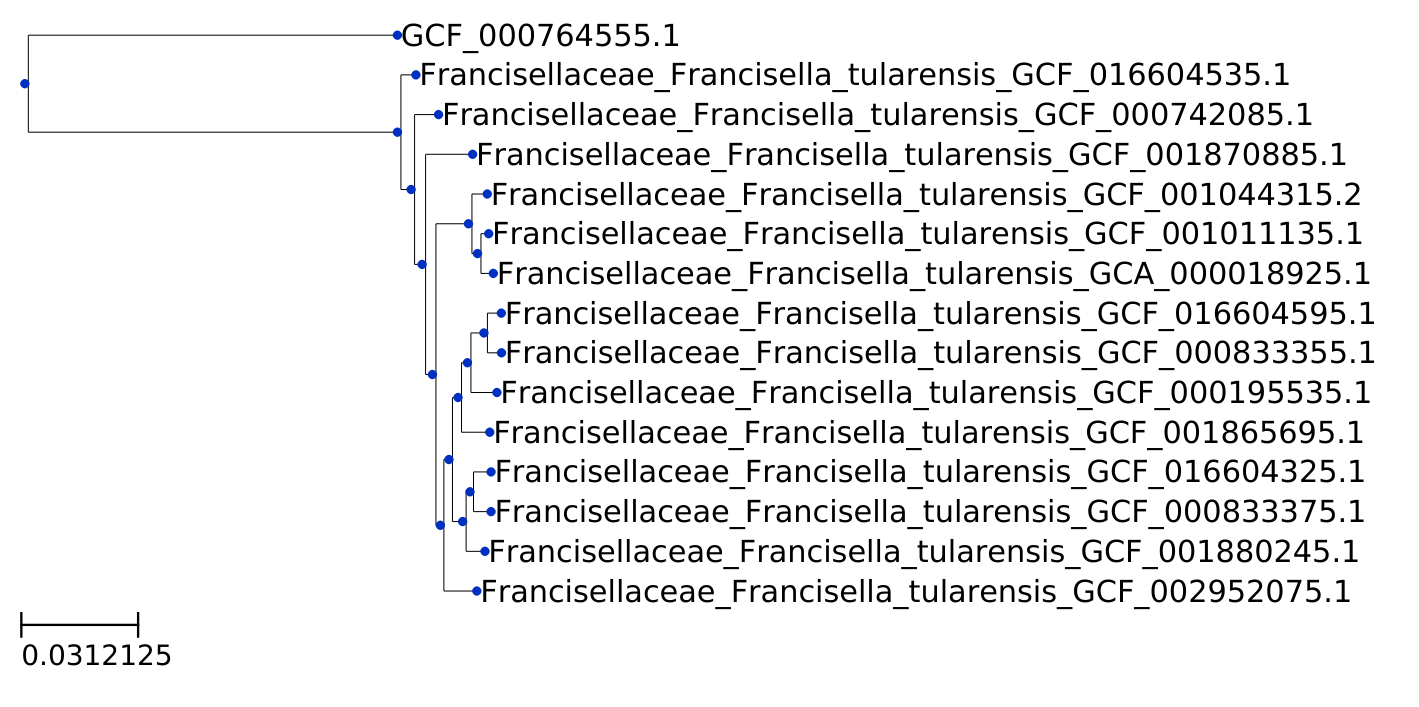
### c. De-replicating genomes at 0.995 average nucleotide identity using 70 threads

repgenr derep –wd tularensis –sani 0.995 –t 70

### d. Computing phylogeny of cluster representants (with outgroup)

repgenr phylo –wd tularensis

The output newick tree can be loaded into a tree-viewer:

  
Figure 2. Phylogeny between representative sequences of clusters. The outgroup sample is “GCF\_000764555.1.”

### e. Generating modification-files for FlexTaxD (with local tree root set as tularensis and removing the outgroup)

repgenr tree2tax –wd tularensis --root\_name \”Francisella tularensis\” --remove\_outgroup

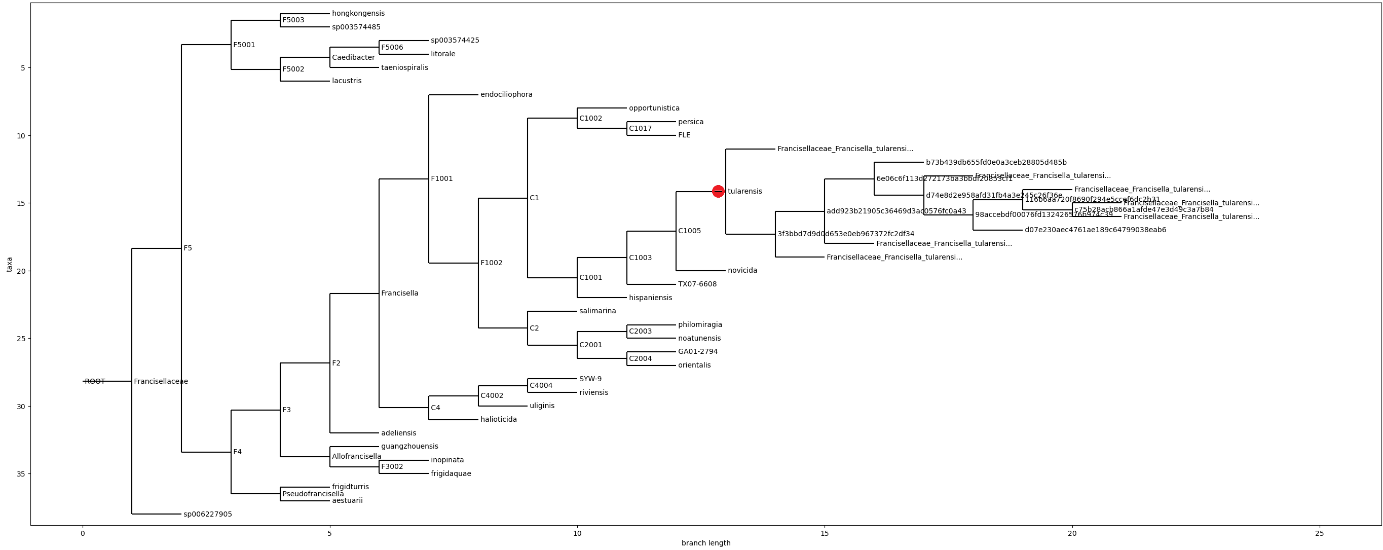
Make sure that you include quotes with preceding back-slash, \”, if a root\_name is specified that contains spaces.

## Step 3: Modify the database using FlexTaxD (replacing branch under tularensis)

flextaxd -db francisellaceae.db --mod\_file tularensis/derep\_genomes\_tree2tax.tsv --genomeid2taxid tularensis/derep\_genomes\_map.tsv --replace --parent "Francisella tularensis"

## Step 4: Visualize database after modification

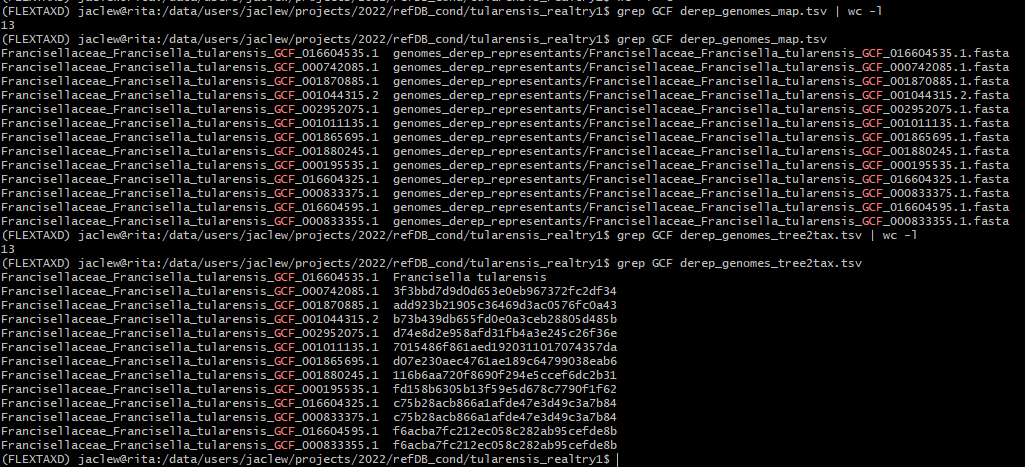
flextaxd -db francisellaceae.db --vis\_type tree --vis\_depth 0 --visualise\_node Francisellaceae

  
Figure 3. Database of *Francisellaceae* generated from FlexTaxD after. *Tularensis*-branch is indicated by a red dot.

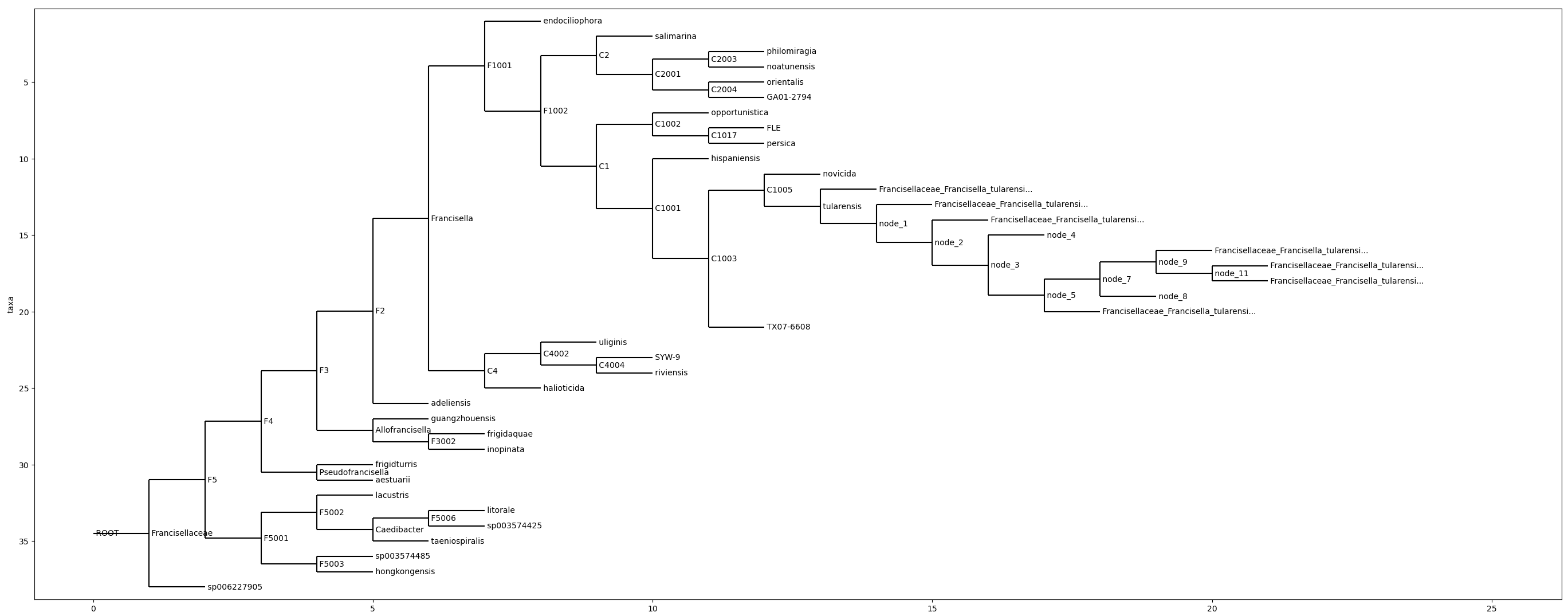
# Problem: All samples are not displayed in database after using FlexTaxD

The tree from genome representants (Figure 2) does not match the tree in the database (Figure 3).

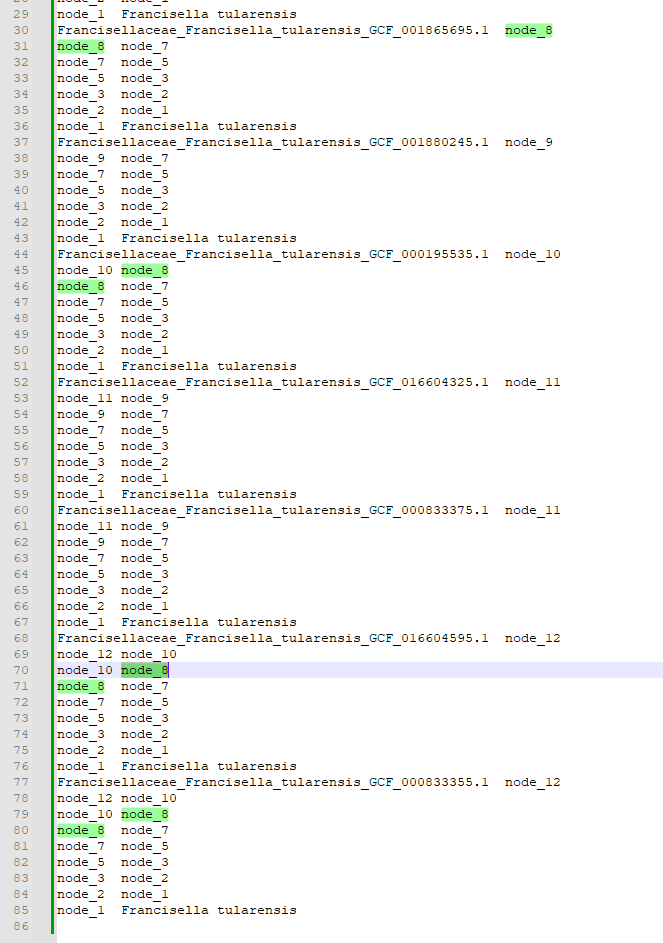
I confirm that all entries exists in the input files to FlexTaxD (--genomeid2taxid and --mod\_file)



Represented with “node\_” as node basename:



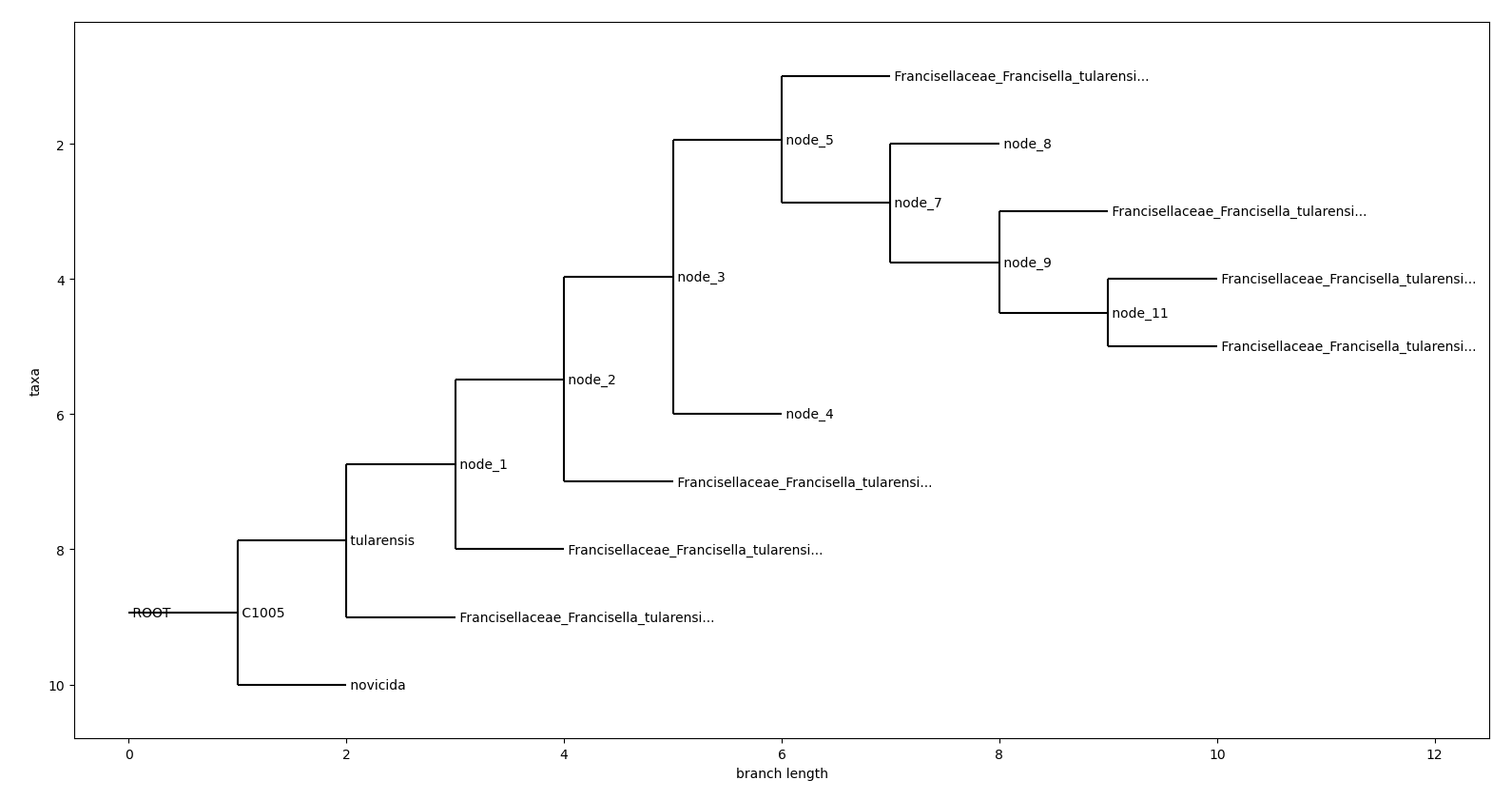
Parent-child relations file:



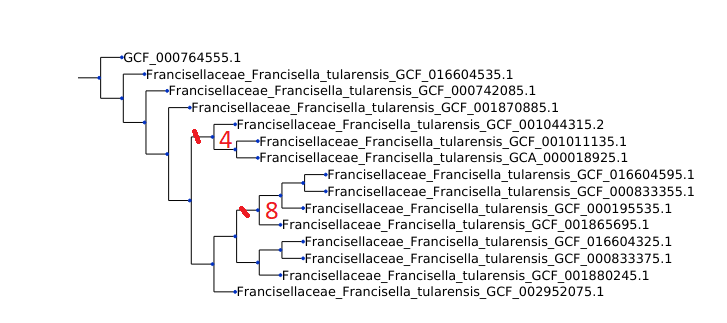
In other words, we miss node\_10 in tree and everything downstream of node\_10.

## Furher characterization of the problem

I investigate further concentrated on the tularensis-part of the tree:



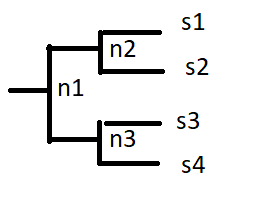
Brances downstream of nodes 4 and 8 are clipped.



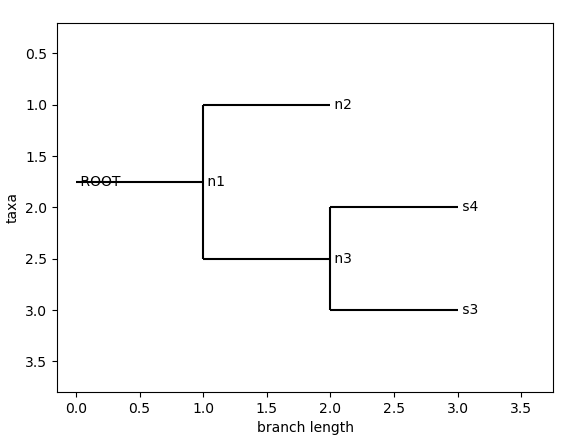
What node 4 and 8 have in common is that their parent branch (node 3 and node 7, respectively) both give childs that are branches. My best guess is that there is a bug in FlexTaxD that makes it unable to handle nodes that have childs which are both branches.

## Minimal example

Goal tree



Produced tree



Taxonomy-file

parent child

n1 n2

n2 s1

n2 s2

n1 n3

n3 s3

n3 s4

Genome-file

s1.fasta s1

s2.fasta s2

s3.fasta s3

s4.fasta s4