**Comparative Genomics 2018**

**Practical 3: Phylogenetic Reconstruction**

Group 11

Fuqi Xu, Milda Valiukonyte, Shuhan Xu

**Summary**

**<Text>**

**Exercise 1**

1.1

–in name of the input genome file

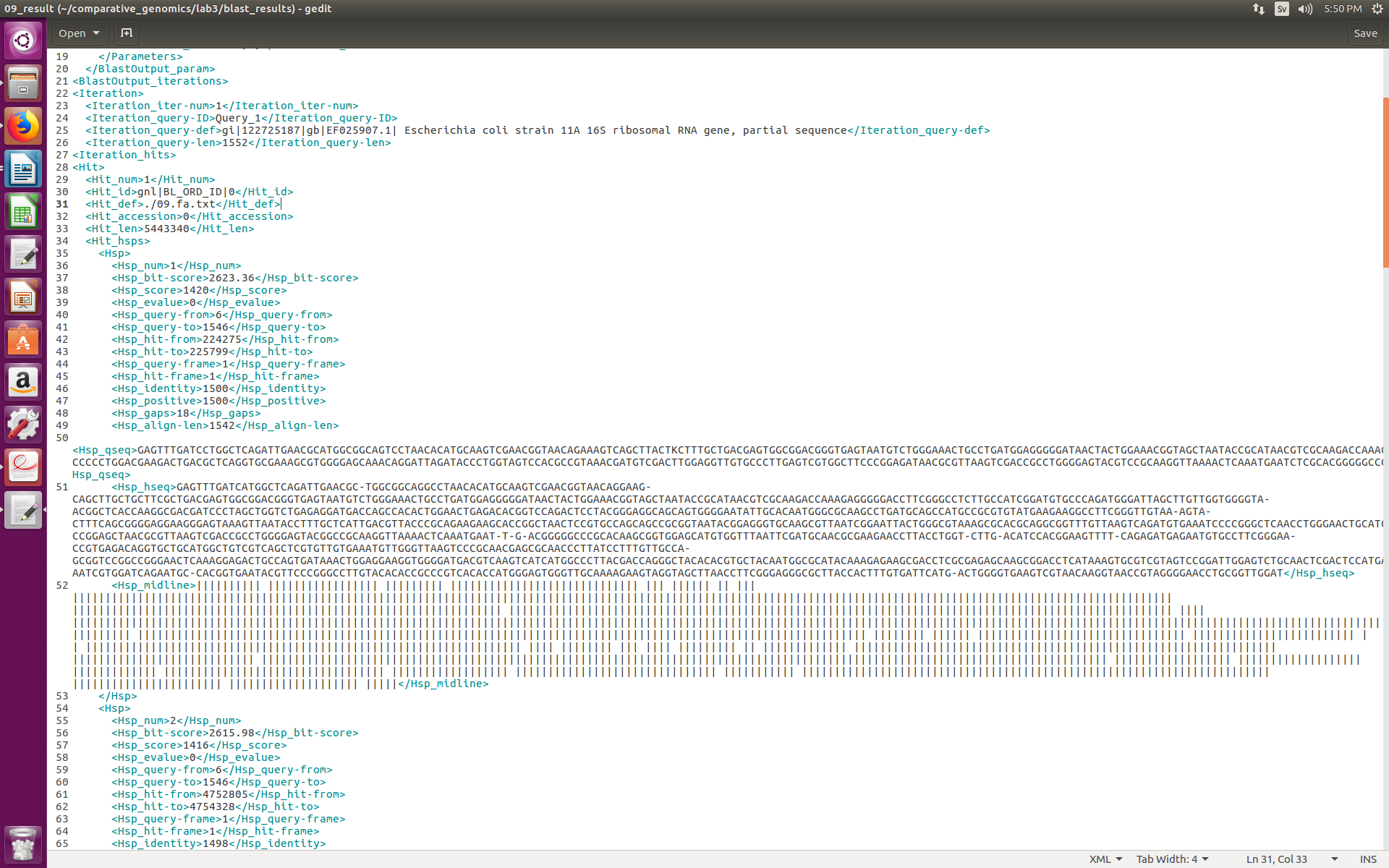
-dbtype database type (nucl - nucleotide)

1.2 We ran the program makeblastdb for each FASTA file.

2. We gathered all the genomes in one file called genomes\_all and made a database out of it. Querying the combined genome database for 16S rRNA is equivalent to querying the individual genome database and concatenating the results together.

3.1 Best hits in a FASTA file: see attachment all\_result.fasta

3.2



3.2.1

–outfmt Output format:

(5 = BLAST XML)

-query Name of the query file

-db Name of the database file

-out Name of the output file

3.2.2 If ‘-out’ flag is used together with the name of the output file, the output file can be found within the directory the command was run in. Otherwise, the output will be printed to the screen (standard output). In the output one can see hits and for each hit the lists of high-scoring segment pairs (hsps) aligned with the query sequence aligned to the hit sequence, as well as various parameters for each of the alignments such as identities, gaps, etc.

**Exercise 2.**

**B.**

a. First, it builds an iterator storing all blast record. For every element in the iterator, it contains the description of the sequence and the alignments result. Then it takes the 0 indexed (first) element of the list of high scoring pairs (hsps) of the alignments.

→ print alignment.hsps [0].sbjct

Ref: biopython documentation

b. BLAST record corresponds to results of a BLAST run for a single query sequence. If multiple query sequences are used, the output file will contain multiple records. In this case, only one query sequence was used (16S rRNA), hence the output file will be parsed into a single BLAST record object. A BLAST record object contains all the information of a BLAST run, including information about the program, query sequence, database, as well as the alignments.

c. BLAST XML output is an assorted list of alignments. It assumes that the first hsp is the single best BLAST result.

d. The script prints the first hit in the blast in fasta format, including the name of the hit and its aligned sequence.

**Exercise 3.**

1.1 Gaps are unfavorable in alignment, so when we introduce a gap, we need to decrease the alignment score accordingly, which is called gap penalty. Gap penalty includes two major parts: gap open penalty and gap extend penalty. Gap open penalty is the cost to introduce a new gap. Gap extension is the cost to enlong the existing gap. Also, if the gap locates at the beginning or the end of the alignment, which is more unfavorable, terminal gap penalty should be included. Bonus score is added to every pair of aligned residues.

In our alignment, KALIGN gap open score = 217, gap extension score =39.40000153, terminal gap score = 292.60000610 , bonus score = 283. The alignment with the highest score would be the best alignment.

1.2

The default gap open penalty in the web server is 80 and the default gap extension penalty is 3. The gap open and extension penalties chosen by kalign are much higher than the default value. As our sequences are very long, we need high gap penalties for the alignment to be sensitive to gaps.

**Exercise 4.**

1.

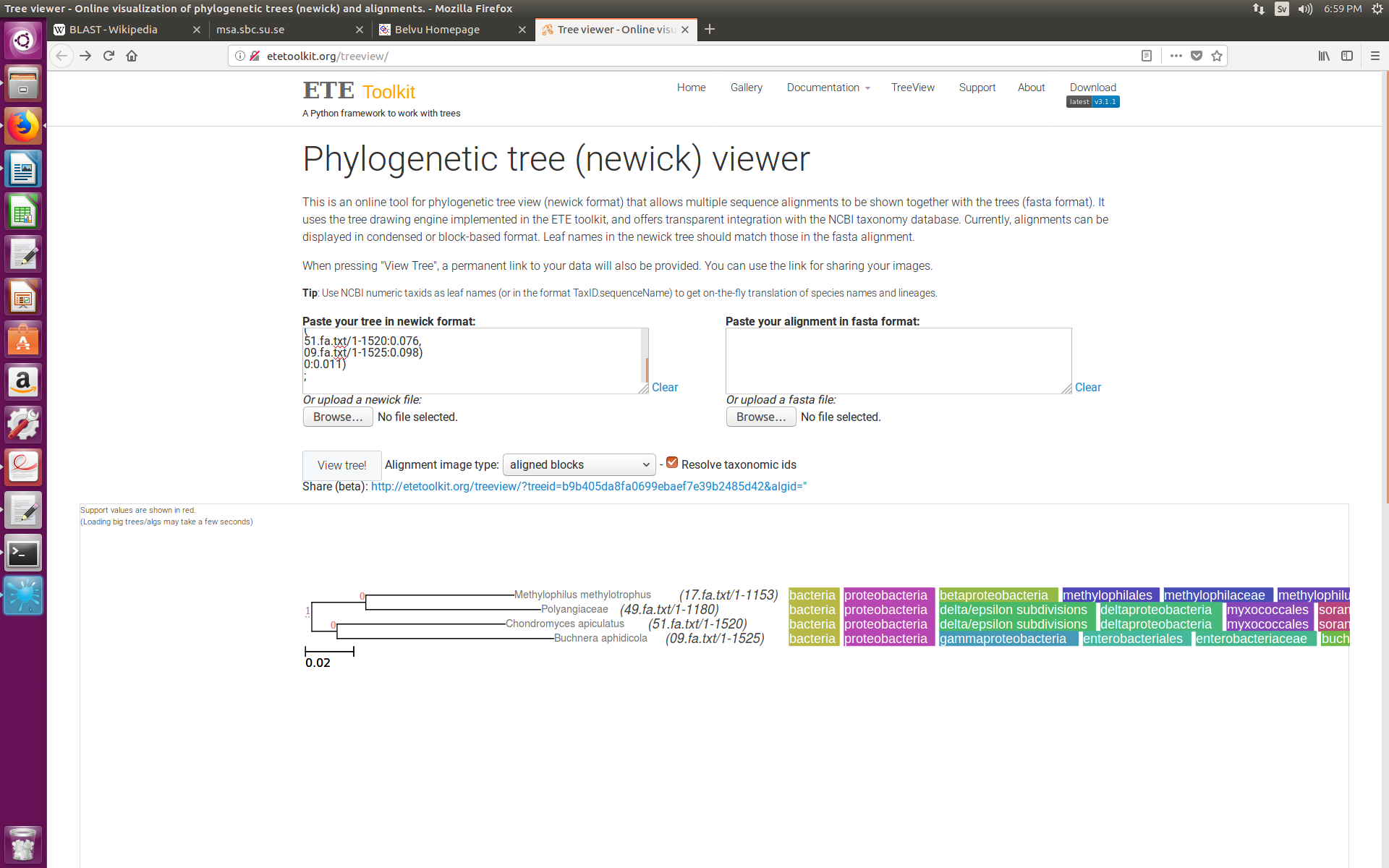
1.1 Scoredist distance correction (default), Jukes-Cantor distance correction, Kimura distance correction, Storm & Sonnhammer distance correction, and uncorrected distance

1.2 Storm & Sonnhammer distance correction method makes the tree more compact whereas Kimura distance correction makes it more spaced out. UPGMA tree has a root whereas neighbor-joining based tree does not have a root.

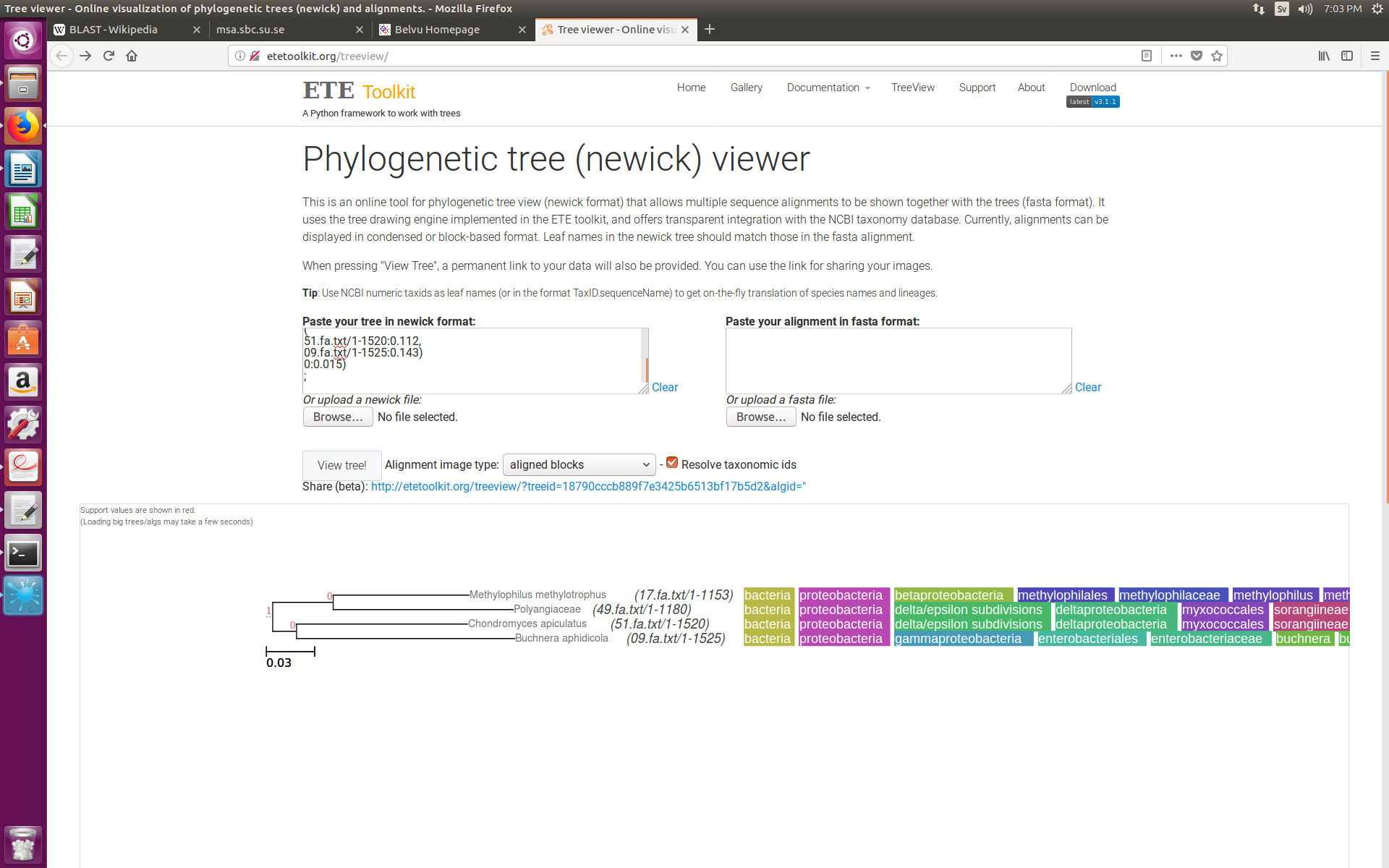
# Different distance corrections don’t change the splits, but the branch length was changed.

Include two screenshots.

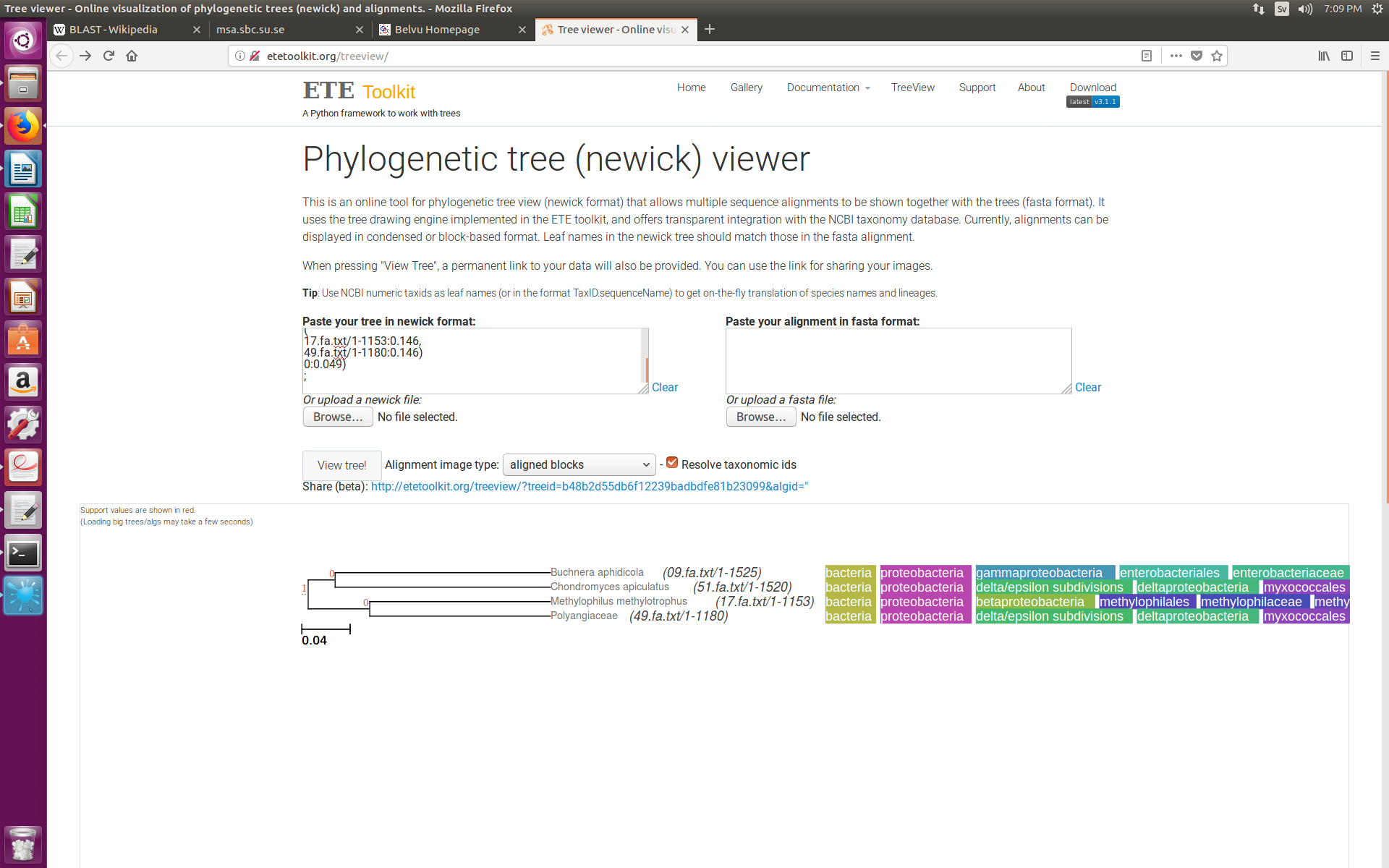
Neighbor-joining and Jukes-Cantor distance correction



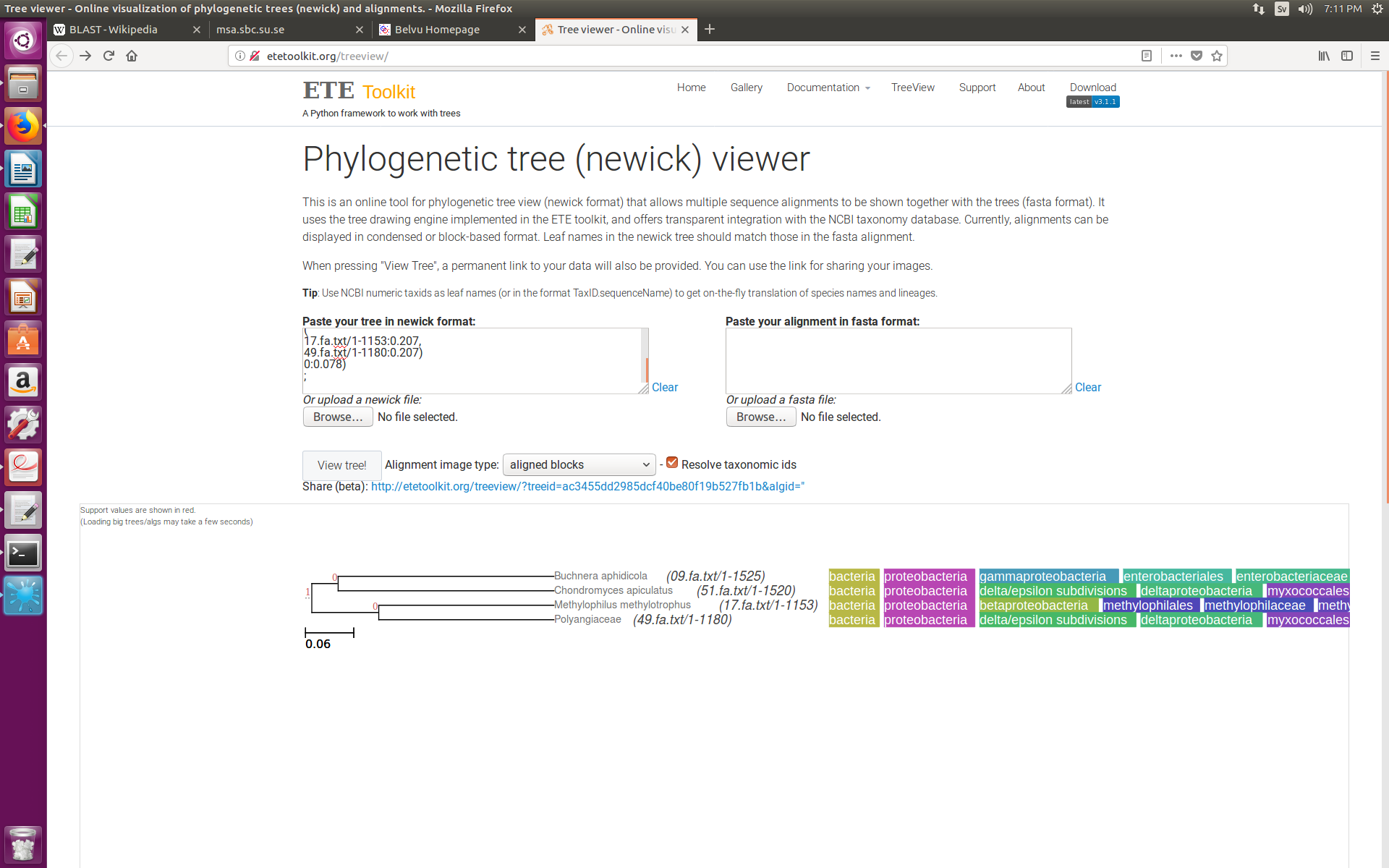
Neighbor-joining and Scoredist distance correction



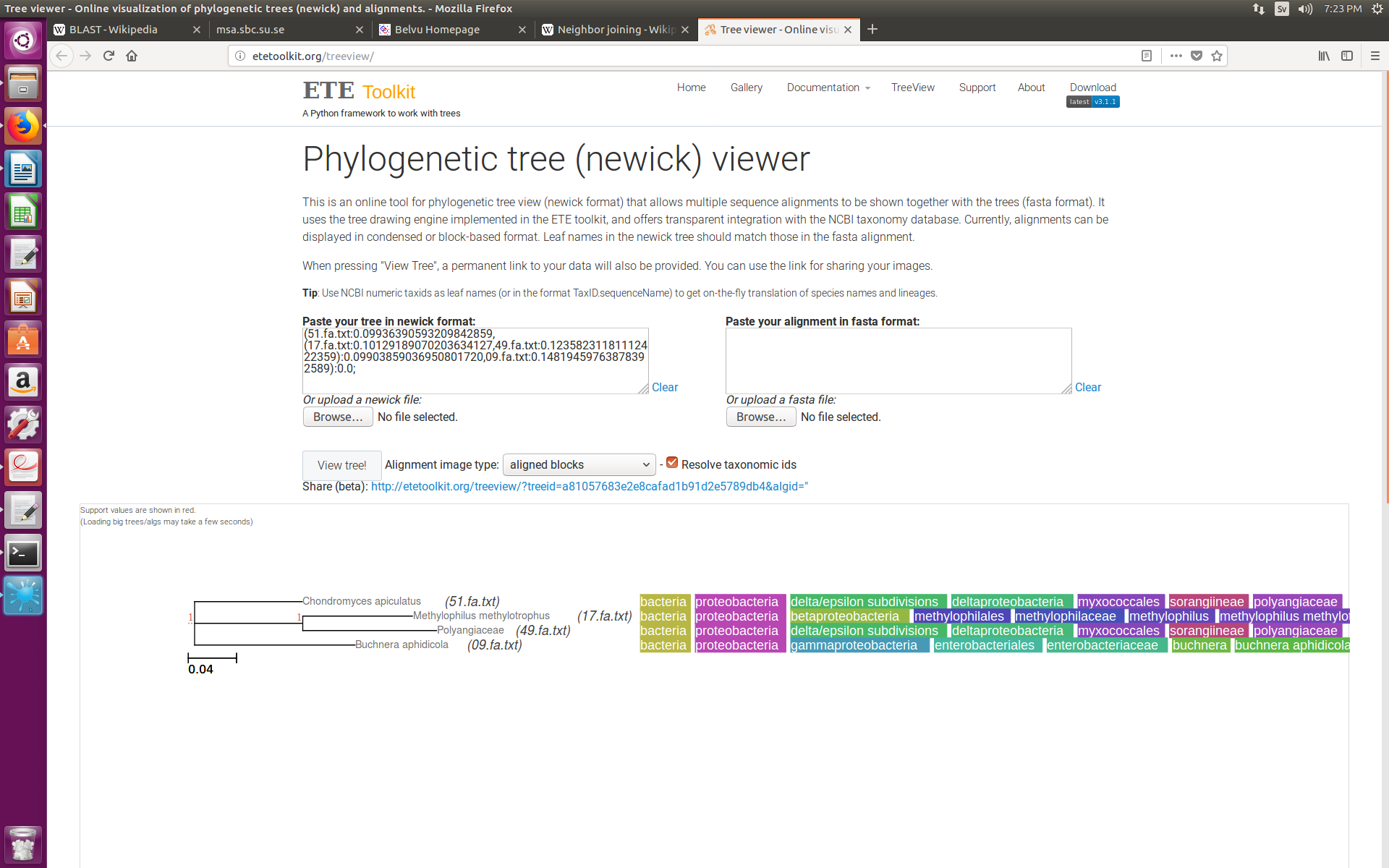
UPGMA and Jukes-Cantor distance correction



UPGMA and Scoredist distance correction



2.2



2.3

-f Selects the algorithm used by the program (-f a is the rapid Bootstrap analysis that looks for the highest scoring ML tree in one run)

-x Specifies a random seed number and switches on rapid bootstrapping

-N the number of alternative runs on different initial trees

-T Specifies the number of threads to run

-p Specifies the parsimony inferences random seed number

-m model of amino acid substitution (PROTCATBLOSUM62 is BLOSUM62)

-s Name of the alignment file

-n Name of the output file

2.4

Distance based methods derive the distance measure for each pair of aligned sequences and the maximum likelihood methods estimate the likelihood of each tree topology and reports the one with the maximum likelihood.

Distance-based

Advantages: Suitable for large datasets as only a single tree is constructed

Disadvantages:

Maximum likelihood

Advantages: Evaluates all possible tree topologies and gives the one with the highest probability.

Disadvantages: Computationally demanding and cannot analyze large datasets

#

Distance-based method is based on the number of identical/nonidential residues.

ML method is based on the likelihood of a tree topology.

The distance based method is based on the sequence information. So if there is enough information provided, like base composition, then distance based method is more accurate. Also, distance method is faster. However, it’s performance relies on the distance correction method. It depends on the selected model and may lead to overfitting problems.

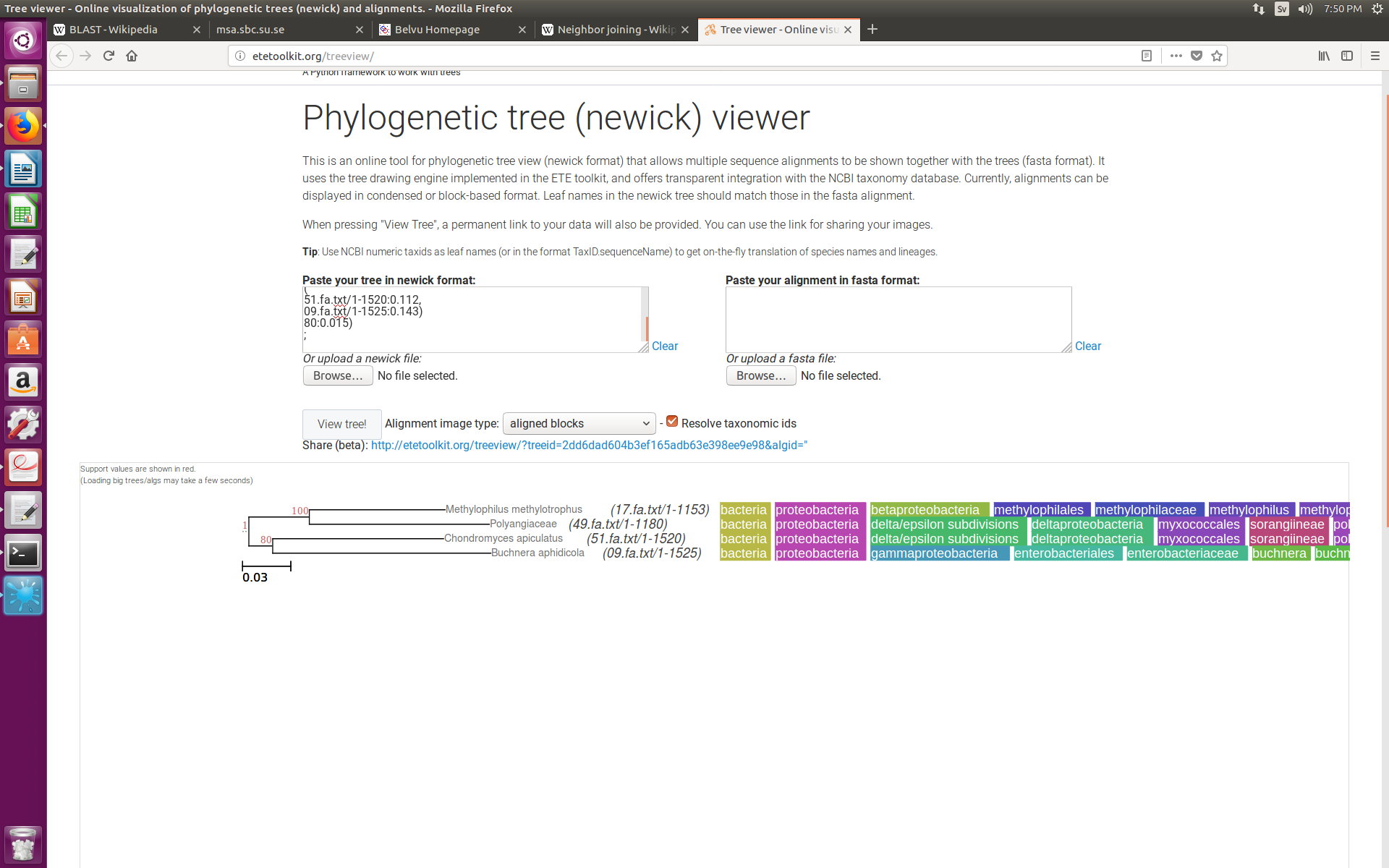
While, the maximum likelihood method is based on statistical techniques. It can fit parameters along with the tree and model the parameters at the same time. However, the maximum liklyhood method is not as rigor as the distance based method so far.

**Exercise 5.**

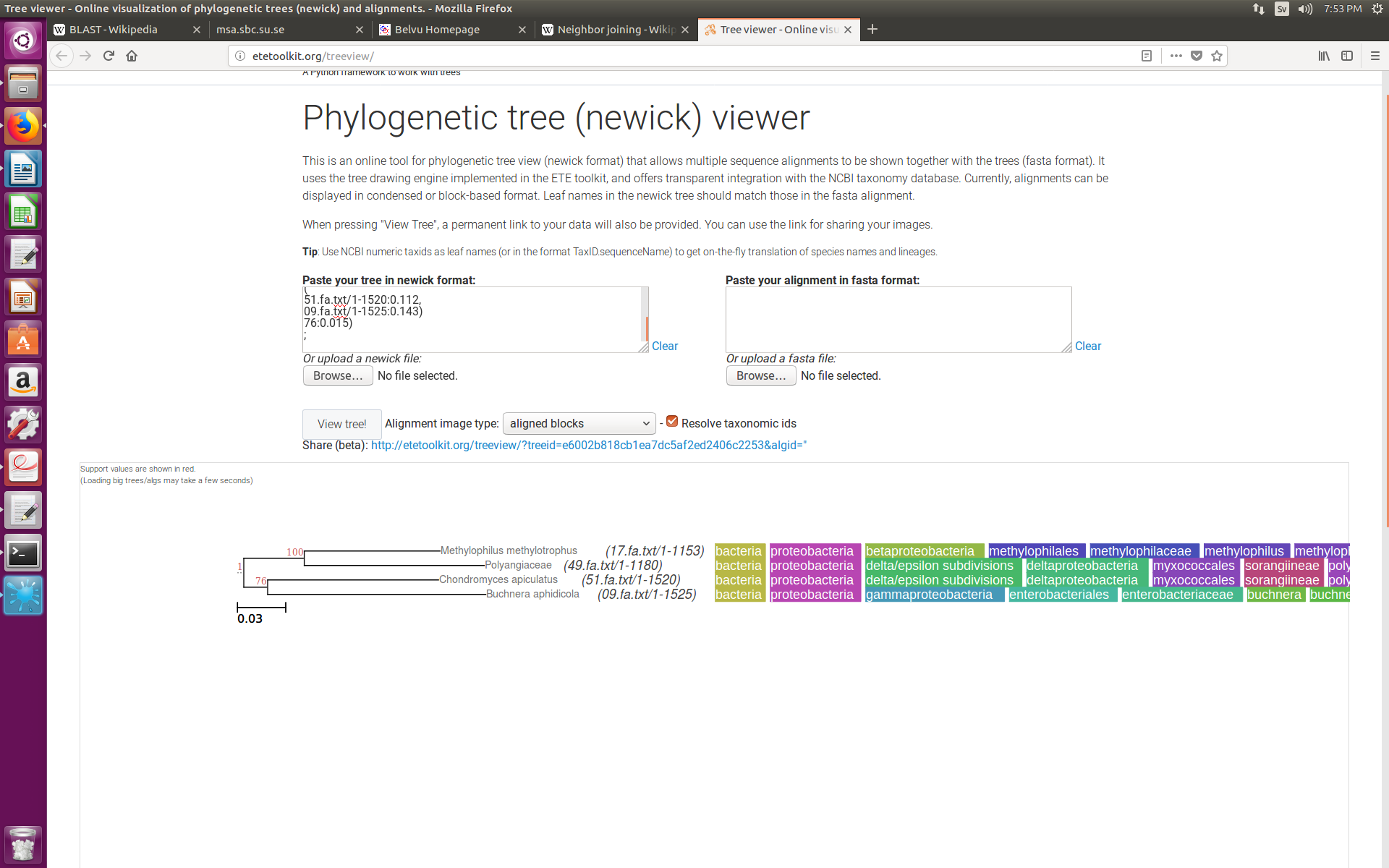
1. Bootstrapping is a method to evaluate the uncertainty of the result when one cannot perform a repeated measurement. It uses the original dataset with N data points and picks another N datapoints from it, which is then called a bootstrap replicate (might be the same points – sampling with replacement). The replicate dataset is then analyzed as the original dataset. Multiple replicate datasets can be created this way and analyzed. In the case of sequences, new sequences are generated from the original ones and then branch lengths are calculated.

2.

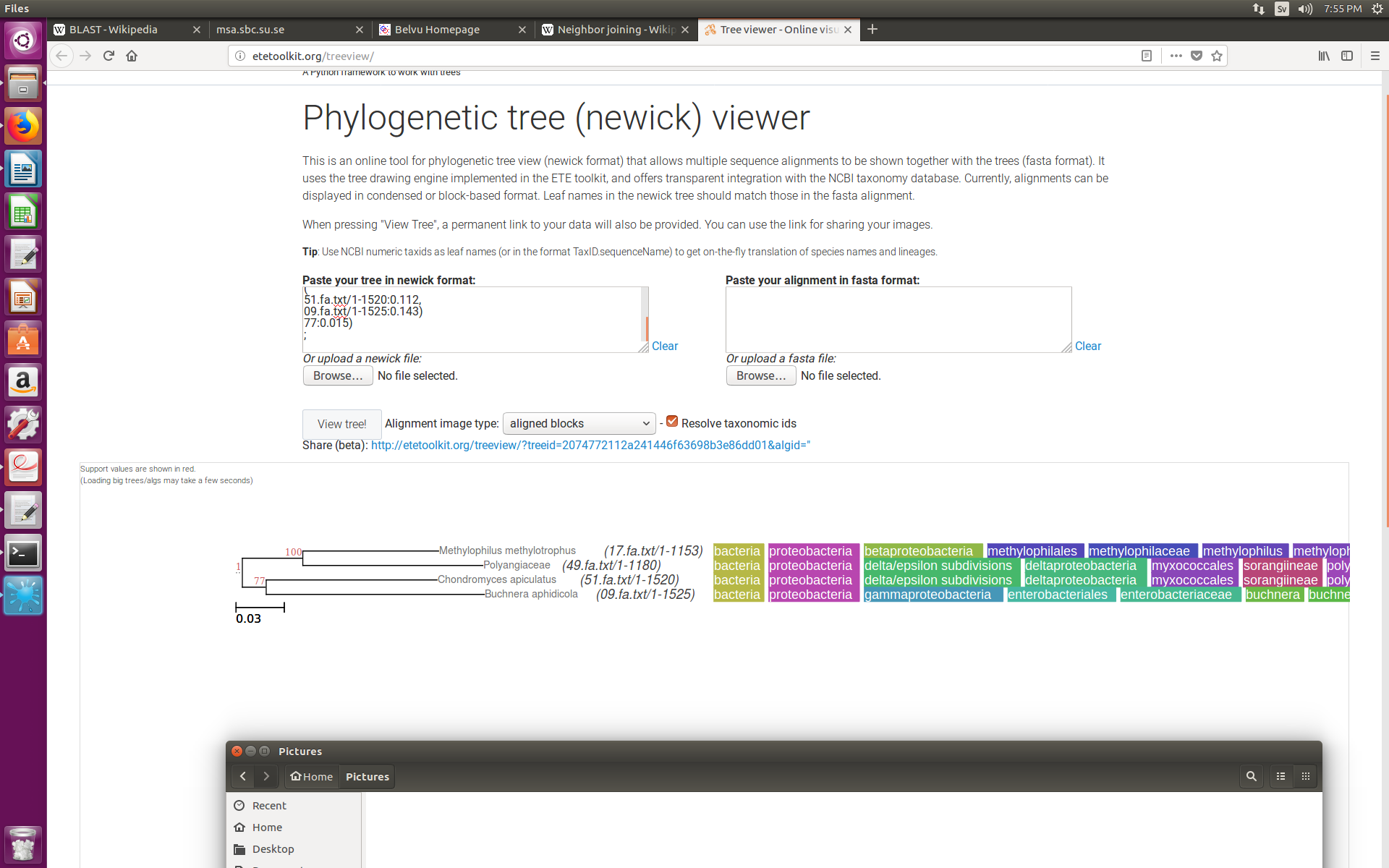
10 bootstrap samples



100 bootstrap samples



1000 bootstrap samples



2.1 It specifies the number of bootstrap samples.

2.2