Comparative Genomics 2018
Practical 3: Phylogenetic Reconstruction

Group 11
Fuqi Xu, Milda Valiukonyte, Shuhan Xu

Summary

In this practical we learned how to analyze evolutionary relations between genes. By performing a BLAST search we identified homologs to *E.coli* 16S gene in four of our genomes. Then we extracted the sequences from the output file using a parser script and aligned the sequences using KALIGN. Next, we created four phylogenetic trees with Belvu, using two different tree building methods (neighbor-joining and UPGMA) and two different distance correction methods (Scoredist and Jukes-Cantor). We then built the maximum likelihood tree with RAxML and compared this method with the previous tree building methods. Finally, we applied bootstrapping support values to build three phylogenetic trees and evaluate the confidence of each topology.

Exercise 1

1.1

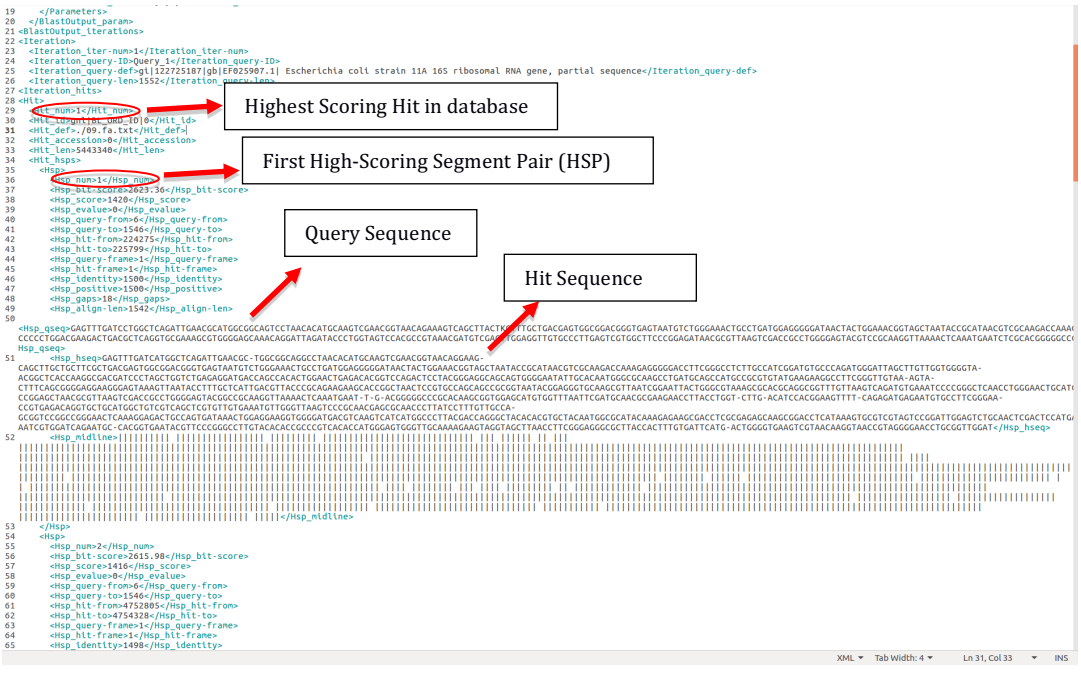
–in	name of the input genome file
-dbtype	database type (nucl - nucleotide)

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.

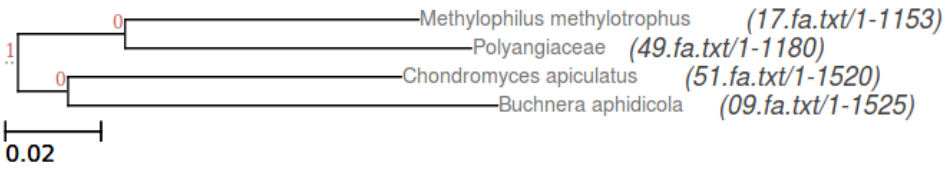
In our alignment, KALIGN gap open score = 217, gap extension score =39.40000153, terminal gap score = 292.60000610 , bonus score = 283. The alignment with the highest score would be the best alignment.

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.

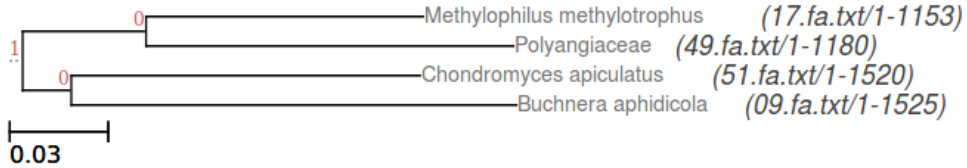
Exercise 4.

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

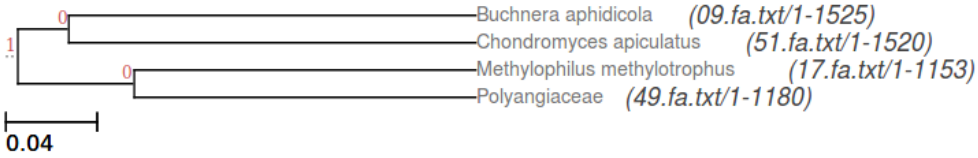
1.2
Neighbor-joining and Jukes-Cantor distance correction



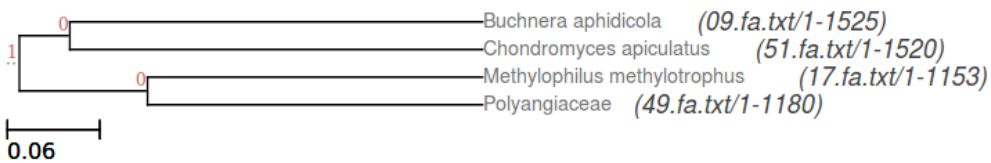
Neighbor-joining and Scoredist distance correction



UPGMA and Jukes-Cantor distance correction



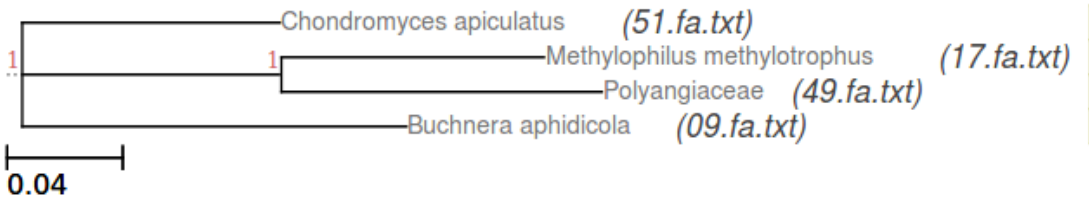
UPGMA and Scoredist distance correction



Within the same tree building method, Scoredist distance correction tends to suggest a greater evolutionary distance (number of mutations per site or time) than Jukes-Cantor distance correction.

Within the same evolutionary model (distance correction), neighbor-joining method gives different branch length for each descendant from their common ancestor. In the case of UPGMA, the distance from a common ancestor to its descendants is the same.

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 a distance measure for each pair of aligned sequences and use these distances to construct the tree. The maximum likelihood methods construct the tree from the multiple sequence alignment directly and assess all sequences at each alignment position. In particular, the maximum likelihood method calculates the likelihood of producing the data given a tree topology. It then reports the tree which has the highest likelihood.

Distance-based
Advantages: It is faster than maximum likelihood method especially when large datasets are used.
Disadvantages: It is unlikely to find the tree which represents the true evolutionary history. It is also unable to statistically evaluate the different distance correction methods (evolutionary models).

Maximum likelihood
Advantages: It is able to find the optimal tree. In addition, since it is based on statistical techniques, it is able to statistically evaluate the different evolutionary models
Disadvantages: It is computationally demanding and is slow when analyzing large datasets

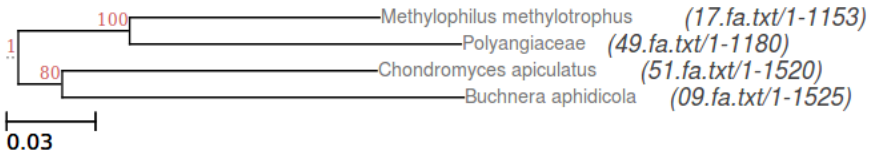
Exercise 5.

1. In bootstrapping, one repeatedly samples the same dataset and reconstructs the trees from the new pseudoreplicate alignments. The bootstrap value is the fraction of pseudoreplicate trees containing the split in the original tree.

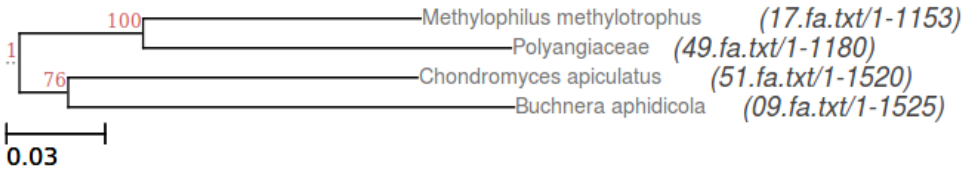
Bootstrapping enables one to estimate the amount of support in the data for the branches in a tree topology. It is used when one wants to know how well the tree represents the data and whether small changes in sampling would change the tree.

2.

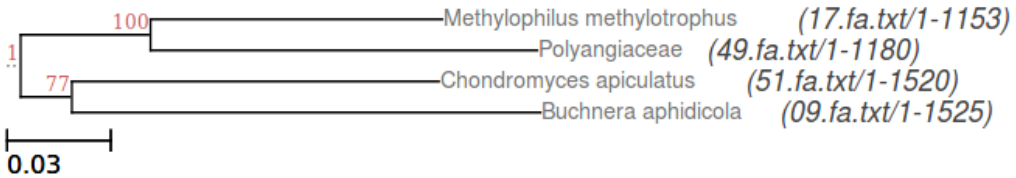
10 bootstrap samples



100 bootstrap samples



1000 bootstrap samples



2.1 It specifies the number of bootstrap samples.

2.2. Base on the bootstrap values above, we would choose 100 bootstrap samples. It is small enough so that it is fast to generate. At the same time, it is large enough for us to have confidence in the estimate as it seems to converge to around 77.