Project 4 (due Nov 8): Tree Comparison

Vedhæftede filer:

- testdata.zip (12,376 KB)
- quicktree.zip (68,526 KB)
- patbase_aibtas.fasta (400,95 KB)
- patbase_aibtas_permuted.fasta (400,95 KB)
- quicktree.newick (22,599 KB)
- rapidnj.newick (22,616 KB)

This project is about comparing evolutionary trees constructed using the Neighbor Joining (NJ) methods on different datasets. The objective is to implement an efficient algorithm for computing the RF distance between two trees and use this implementation in an experiment.

Problem

You should implement an algorithm for computing the RF distance between two unrooted evolutionary trees over the same set of species. The algorithm can e.g. be Day's algorithm as explained in class. You should make a program called rfdist which as input takes two evolutionary trees in Newick format (also referred to as 'New Hampshire format'), and outputs the RF distance between them.

The archive testdata.zip

contains two trees with RF-distance 8. These can e.g. be used for testing.

Experiment

The programs QuickTree and RapidNJ are implementations of the NJ methods. QuickTree implements the basic cubic time algorithm while RapidNJ implements an algorithm the is faster in practice. You might want to take a look at the QuickTree paper and the RapidNJ paper.

Downloading quicktree_1.1.tar.gz (in case of problems downloading, you can use this local copy of quicktree source code

) and rapidnj-src-2.3.2.zip and compiling on a Linux-platform is straightforward (just run 'make').

The file patbase_aibtas.fasta

contains 395 protein sequences from the P-Type ATPase Database in Fasta-format.

You should make an experiment where you first construct a number of trees for these 395 sequences using NJ (as implemented in QuickTree and RapidNJ) based on different multiple alignments of the 395 sequences, and secondly, compare the construted trees using your program rfdist in order to investigate the influence of using different multiple alignment methods and tree reconstruction methods. Biologically, the 395 sequences are grouped into five groups. The group which a sequenence belongs to is indicated by the first character (1-5) in its name in Fasta-file. If you visualize your constructed trees using e.g. the program Dendroscope, you might want to inspect to what extend sequences from one group are in the same subtree.

The alignment methods you most use are:

- Clustal Omega: http://www.ebi.ac.uk/Tools/msa/clustalo/
- Kalign: http://www.ebi.ac.uk/msa/Tools/kalign/
- MAFFT: http://www.ebi.ac.uk/Tools/msa/mafft/
- MUSCLE: http://www.ebi.ac.uk/Tools/msa/muscle/

You should use the default parameters of each program. Beware of the output format. QuickTree can only read multiple alignments in Stockholm-format, and RapidNJ can read multiple alignments in both Fasta- and single lineStockholm-format. You can use the online converter available at http://hcv.lanl.gov/content/sequence/FORMAT_CONVERSION/form.htmlto convert different alignment formats to Stockholm- and Fasta-format. **Note**: To convert to single line Stockholm-format, you must set the 'Output line width' to 'as wide as possible'.

Experiment 1: For each alignment method (Clustal Omega, Kalign, MAFFT, MUSCLE), you build a NJ tree using QuickTree and RapidNJ, and compute the RF-distance between each combination of these eight tree. The outcome of your experiment, is an 8x8 table showing the RF-distance between each pair of constructed trees. (You might want to use the program Dendroscopeto visualize the constructed trees.)

Experiment 2: Redo the above experiment where you use 395 input sequences in patbase_aibtas_permuted.fasta

. This yields another 8x8 table.

Experiment 3: Compute the RF-distance between the trees produced in 'Experiment 1' and 'Experiment 2' using the same alignment and tree reconstruction method. This yields 8 distances.

Experiment 4 (OPTIONAL): Redo experiment 1-3 with quartet distance instead of RF-distance. You can e.g. use tqDist to compute the quartet distance.

Experiment 5: Make an experiment that shows the running time of your implementation of rfdist. You must choose test data your self. If the running time is not linear, you should explain why.

Note: Depending on your Newick-parser, you might have problems comparing trees constructed by QuickTree and RapidNJ. The reason is that RapidNJ encloses the leaf names in apostrophes, while QuickTree does not. See e.g. these examples of a QuickTree newick output (quicktree.newick

) and a RapidNJ newick output (rapidnj.newick) with names enclosed in apostrophes. It is easy to solve the problem by removing all apostrophes from the RapidNJ output. On a Unix-system like Linux or Mac OS X, you can easily do this by piping the output of RapidNJ through sed like this: rapidnj -i sth somealignment.stockholm | sed -e "s/'//g" > tree_without_apostrophes.newick

Report

You (i.e. your group) must hand in a report (in pdf-format) and your implementation of rfdist (in a zip-archive) via Blackboard no later than **Thurday, November 8, 2018, at 16:00**. Your report should be no more than 3 pages, and must cover:

- Status of your work.
- A short description of your implementation of rfdist, explaning what algorithm you implemented, how you read in trees in Newick-format, how you validated the correctness of your implementation, and how to access and run your program rfdist.
- References to the 8 alignments (in Stockholm-format) and trees (in Newick-format) that you have produced in Experiment 1 and 2.
- Your results of Experiment 1-3, i.e. an 8x8 table showing the RF-distance between each pair of constructed trees based on the patbase_aibtas.fasta, an 8x8 table showing the RF-distance between each pair of constructed trees based on the patbase_aibtas_permuted.fasta, and a 1x8 table showing the RF-distance between the trees constructed based on the same alignment and NJ method for normal and the permuted dataset. Comment on your results. Is everything as expected, or are you surprised?
- Your results of Experiment 4, if have done it. Comment on your results. How do they compare to Experiment 1-3?
- Your results of Experiment 5. Remember to argue your choice of test data and why (or why not) your experiment shows that the running time is linear.