Project 5 (due Nov 22): NJ tree construction

Vedhæftede filer:

example_slide4.phy (136 B)

distance_matrices.zip (21,769 MB)

pfam_alignments.zip (91,972 MB)

This project is about making an efficient implementation of the neighbor-joining (NJ) algorithm as shown on slide 62 in the slides about tree reconstruction

and compare its performance to the NJ programs QuickTree and RapidNJ that you know from project 4.

Problem

You should make a program that implements the NJ algorithm as shown in the slides about tree reconstruction. Your program program should take a distance matrix in phylip-format as input and produce a tree in newick-format as output. You should know these formats from project 4. Your aim is to make your implementation as efficient as possible.

The file example_slide4.phy

contains the distance matrix (in phylip-format) from slide 4 in the slides about tree reconstruction. With this matrix as input, your program should produce the tree that is also shown on slide 4 in the slides about tree reconstruction.

Experiment

From project 4, you know the programs QuickTree and RapidNJ that are implementations of the NJ methods. QuickTree implements the basic cubic time algorithm while RapidNJ implements an algorithm the is faster in practice.

You should compare the performance of your program against these two program in the following way.

The archive distance_matrices.zip

contains 14 distance matrices (in phylip-format) ranging in size from 89 to 1849 species. For each distance matrix, you should do the following:

- Measure the time it takes to construct the corresponding NJ tree using QuickTree, RapidNJ, and your program.
- Compute the RF-distances (using your program rfdist from project 4) between the trees produced by QuickTree, RapidNJ, and your program.

(If you want to investigate the running time of your program on more examples than provided in distance_matrices.zip, then you are welcome to download pfam_alignments.zip

that contains 128 of alignment in Stockholm-format (from the Pfam database) aligning from 58 to 71535 species that you can convert to distance matrices in phylip-format using e.g. QuickTree. However, converting the big alignments to distance matrices would probably take too long and require too much space.)

Report

You must hand in a report (in pdf-format) and your implementation of NJ (in a zip-archive) via Blackboard no later than **Thursday, Nov 22, 2018, at 16:00**. Your report should be no more than 2 pages and must contain:

- Status of your work, does your program work correctly on the test case in example_slide4.phy, i.e. from this matrix produces the tree on slide 4. If not, what do you think is wrong and what have you done to alleviate it.
- A description of your implementation of nj explaining what you have done in order to make it as efficient as possible, including which programming language(s) you have used.
- A description of the of the machine (cpu, ram, os, ...) that you have used to perform the experiment and how you have measured the running time.
  A table summarizing the results of your experiment. For each of 14 distance matrix in distance_matrices.zip, your table should contain a row with 8 entries that report:
    - Running time of QuickTree.
    - Running time of RapidNJ.
    - Running time of your program.
    - The speed-up achieved by your program relative to QuickTree, i.e. "Running time of QuickTree" / "Running time of your program".
    - The speed-up achieved by your program nj relative to RapidNJ, i.e. "Running time of RapidNJ" / "Running time of your program".
    - The RF-distance (as computed by your program rfdist from project 4) between the tree produced by QuickTree and the tree produced by your program.
    - The RF-distance (as computed by your program rfdist from project 4) between the tree produced by RapidNJ and the tree produced by your program.
    - The RF-distance (as computed by your program rfdist from project 4) between the tree produced by RapidNJ and the tree produced by QuickTree.