TreeFix-TP Manual

Samuel Sledzieski
Department of Computer Science
University of Connecticut

Mukul Bansal
Department of Computer Science
University of Connecticut

April 4, 2019

Contents

1 Introduction

TreeFix-TP is a program for reconstructing highly accurate viral phylogenetic trees. TreeFix-TP incorporates host information with sequence data to reconstruct the tree using a mixed maximum likelihood / maximum parsimony framework. Given a maximum likelihood phylogeny and a multiple sequence alignment, TreeFix-TP searches among multiple topologies which are supported equally by sequence data (using the Shimodaiara-Hasegawa test statistic). The selected topology is the one that minimizes the number of necessary transmissions.

Requirements

- Python (2.5.4 or greater)
- C compiler (gcc)
- SWIG (1.3.29 or greater)
- Numpy (1.5.1 or greater)
- Scipy (0.7.1 or greater)
- Additionally, Python modules are required for computing the p-value for likelihood equivalence.

Likelihood TreeFix-TP uses the Shimodaira-Hasegawa (SH) test statistic with RAxML site-wise likelihoods to compute p-values for each candidate tree.

Parsimony TreeFix-TP scores each statistically equivalent candidate tree using Fitch's algorithm. Given host labels at the leaf nodes, the Fitch module computes a score equivalent to the minimum necessary number of transmissions needed to label the internal nodes.

2 Tutorial

Usage `treefixTP -[options] <treefile>`

Input TreeFix-TP requires a seed tree, generally a maximum likelihood tree, and a multiple sequence alignment.

Options TreeFix-TP assumes that the multiple sequence alignment and seed tree have the same root name, with different extensions. The default extensions are ".align" and ".tree", but other extensions can be specified. The output tree file will have the same root name, and will have either the default extension ".treefix.tree", or a user specified extension.

- `-A <alignment file extension>, --alignext=<alignment file extension>`
- `-o <old tree file extension>, --oldext=<old tree file extension>`
- `-n <new tree file extension>, --newext=<new tree file extension>`

The default likelihood model uses RAxML site-wise likelihoods, but a different user-defined module can be substituted. The default likelihood test is the SH test, but again a user-defined test can be used.

- `-m <module for likelihood calculations>, --module=<module for likelihood calculations>`
- `-e <extra arguments to module>, --extra=<extra arguments to module>`
- `-t <test statistic>, --test=<test statistic>`
- `--alpha=<alpha>`: alpha threshold (default: 0.05)
- `-p <alpha>, --pval=<alpha>`: same as alpha

The default module for transmission cost uses Fitch's algorithm. A user defined cost model can be substituted.

- `-M <module for transmission cost calculation>, --smodule=<module for transmission cost calculation>`

A number of options can be specified to guide the search conducted by TreeFix-TP.

- `-x <seed>, --seed=<seed>`: seed value for random generator
- `--niter=<number of iterations>`: number of iterations (default: 1000)
- `--nquickiter=<number of quick iterations>`: number of sub-proposals (default: 100)

Finally, there are a handful of informational options.

- `--version`: show program's version number and exit
- `-h, --help`: show the help message and exit
- `-V <verbosity level>, --verbose=<verbosity level>`
- `-l <log file>, --log=<log file>`: log file name. Use '-' to display on stdout.

3 Test Parameters

For testing of TreeFix-TP, we simulated viral transmission and sequence data using FAVITES (https://github.com/niemasd/FAVITES). The baseline for testing uses the Barabasi-Albert model for contact network generation, the SEIR model of transmission, the coalescent model for phylogenetic evolution, and the GTR+Gamma model for sequence evolution. We varied the length of viral sequence, the height of the phylogenetic tree, and the number of sequences sampled from each host to evaluate the effectiveness of TreeFix-TP under different conditions. In addition, we tested using the SIR model of transmission to evaluate the effects of different transmission patterns on TreeFix-TP. Under the SEIR model of transmission, each host has a latent period in which they are infected but not yet infectious. Once an individual becomes infectious, it can begin to infect multiple neighbors within the contact network, which do not immediately become infectious. This leads to a transmission network with a few very high degree nodes, and transmission occurring in a burst-like pattern. Under the SIR model of transmission, a host may transmit the disease as soon as it becomes infectious, leading to fewer high degree nodes and more low degree nodes in the transmission network. As a result, the transmissions occur consistently through the entire time window.

4 Testing Pipeline

This section describes the pipeline which we have developed for testing of TreeFix-TP. In this pipeline, ground-truth phylogenies, contact networks, and gene sequences are generated using FAVITES [1]. The sequences are then used to generate a maximum likelihood phylogeny using RAxML [2]. The RAxML phylogeny and sequences are provided as input to TreeFix-TP, which will generate the error-collected phylogeny. The two phylogenies are then scored using the transmission cost calculated by Fitch's algorithm, and are compared to the ground-truth phylogeny using Robinson-Foulds distance. Transmission networks are generated using Phyloscanner [5] or Tnet, which implements Sankoff's Algorithm, and these networks are compared to the true networks and scored on the basis of false positive, false negative, and true positive edges. This process can be done using four separate scripts, `data_gen`, `tree_gen`, `tree_score`, and `net_gen`, or by running `RunTreeFixTest`. All steps require the inclusion of a `run name` argument, which will be used to connect the different scripts. A directory with the run name will be created in the current working directory containing all files generated by the test. The testing pipeline can be found at https://github.com/samsledje/treefixTP-testing.

Additional Requirements

- RAxML
- FastTree
- ete3
- Phyloscanner
- Tnet

4.1 RunTreeFixTest

This script will call `data_gen`, `tree_gen`, and `tree_score` consecutively to perform a full simulation testing run. All options from each script (detailed below) can be used with RunTreeFixTest. If none are specified, the default options are used.

Usage: `bin/RunTreeFixTest [options] -c <config file> <run name>`

4.2 RunPhyloscanner

Will generate networks with Phyloscanner from a given starting tree.

Usage: `bin/RunPhyloScanner <run name> <base tree>`

4.3 CheckProgress

If multiple TreeFix-TP testing runs are going, this will output the end of the RAxML and TreeFix-TP log files, to check how far along each run is. At the bottom, the number of complete runs will be displayed.

4.4 CheckDone

If multiple TreeFix-TP testing runs are going, this will display the outfiles of only the complete runs.

4.5 data_gen

This script will generate test data using a FAVITES configuration file. Some preprocessing is done to ensure that the sequences and phylogeny can be processed by TreeFix-TP, including renaming of tree nodes. A directory will be created using the provided run name, and two sub-directories will also be created. `FAVITES_output/` is set as the output directory for FAVITES, while `clean_data/` contains all data necessary for subsequent steps of the pipeline.

Usage: `lib/data_gen.py [options] <run name>`

Options:

- `-h`: Show help message
- `-s <extension>`: Sequence file extension. Default is '.fasta'
- `-t <extension>`: Tree file extension. Default is '.tree'
- `-n <extension>`: Network file extension. Default is '.network'
- `-c <file>`: Required argument. FAVITES configuration file
- `--favites=<path>`: Path to run_favites.py if not on path
- `-l <file>`: Log file name. If not set, default is `<run name>.log`

4.6 tree_gen

This script will calculate a maximum likelihood phylogeny using RAxML. The default settings for RAxML are outlined below, but non-default parameters can be passed using `--raxopts`. The RAxML tree and the sequence alignment from `data_gen` are used as inputs to TreeFix-TP, which will generate an error corrected phylogeny in `<run name>/clean_data/`. By default 1000 iterations of TreeFix-TP are run, however the number of iterations as well as other non-default parameters can be passed to TreeFix-TP using `--treefixopts`. All RAxML output files are written to `RAxML_output/`, and the maximum likelihood tree is copied into `clean_data/`

Usage: `lib/tree_gen.py [options] <run name>`
Note that this run name must match the run name provided to `data_gen`.

Options:

- `-h`: Show help message
- `-s <extension>`: Sequence file extension. Should match the extension used for `data_gen`
- `-r <extension>`: ML tree file extension. Default is '.raxml'
- `-f <extension>`: TreeFix tree file extension. Default is '.treefix'
- `--fast`: This flag can be set to use FastTree instead of RAxML for quick ML tree generation
- `--ml=<path>`: Path to RAxML if not on path
- `-l <file>`: If a non-default log file was used for `data_gen`, it should be specified here to maintain a singular logging location

4.7 tree_score

This script will score both the maximum likelihood and TreeFix-TP trees using a variety of metrics. Fitch Score represents the minimum necessary number of transmissions, calculated using Fitch's algorithm for the small parsimony problem. The Fitch score of the simulated tree should be equal to the number of transmissions performed by FAVITES. Each tree will also be compared to the true tree generated by FAVITES using the Robinson-Foulds distance. For more information about these metrics see Sections 1.6 and 4.3 of *Computational Phylogenetics* [3]. When used with the –csv option, percent decreases in cost and RF distance will be calculated, and multiple runs are written to a single .csv file.

Usage: `lib/tree_score.py [options] <run name>`

Note that this run name must match the run name provided to `data_gen`.

Options:

- `-h`: Show help message
- `-r <extension>`: ML tree file extension. Should match the extension used for `tree_gen`
- `-f <extension>`: TreeFix-TP tree file extension. Should match the extension used for `tree_gen`
- `-t <extension>`: Simulated tree file extension. Should match the extension used for `data_gen`
- `--fitch=<path>`: Path to fitch.linux if not on path
- `--csv`: Write multiple runs to a single .csv file
- `-o <file>`: Desired outfile for scores. By default, scores will be written to `<run name>.out`
- `-l <file>`: If a non-default log file was used for `data_gen`, it should be specified here to maintain a singular logging location.

4.8 net_gen

This script will reconstruct transmission networks using both Phyloscanner and Tnet, and score those networks. Reconstructed networks are compared to the true network by looking at the number of false positive, false negative, and true positive edges. When used with the –csv option, TPR, PPV, and F1 scores are calculated, and multiple runs are written to a single .csv file.

Usage: `lib/net_gen.py [options] <run name>`
Note that this run name must match the run name provided to `data_gen`.

Options:

- `-h`: Show help message
- `-r <extension>`: ML tree file extension. Should match the extension used for `tree_gen`

- `-f <extension>`: TreeFix-TP tree file extension. Should match the extension used for `tree_gen`
- `-l <file>`: If a non-default log file was used for `data_gen`, it should be specified here to maintain a singular logging location
- `-o <file>`: Desired outfile for scores. Should match the outfile used for `tree_gen`
- `--csv`: Write multiple runs to a single .csv file
- `--tnet`: Reconstruct networks with Tnet in addition to Phyloscanner

5 Changes from TreeFix-DTL

TreeFix-TP was built by modifying TreeFix-DTL [4], with the most significant change being the replaced cost module (fitch.linux rather than ranger-dtl). All changes are documented below.

5.1 treefixTP

- Based on treefixDTL script
- Removed most options in parser, only need alignment extension, old extension, new extension
- Changed default smodule to FitchModel
- Call treefix_for_TP rather than treefix

5.2 fitchmodel.py

- Copy of rangerdtlmodel.py
- Changed compute_cost() to call fitch.linux executable
- Removed stree, smap from FitchModel.optimize_model
- Removed call to CostModel.optimize_model()

5.3 treefix_for_TP

- Renamed treefix to treefix_for_TP - this avoids path collision with treefix and treefixDTL
- Removed smap and stree as required arguments
- Changed to common.check_req_options(species=False)
- Removed stree and gene2species from check_input_tree()
- Removed stree and gene2species from search_landscape()
- Removed check_congruent_tree() function, no longer applies
- Removed 'if flag: return mintree'
- Removed reading species tree and species map from main()
- Removed stree and gene2species from calls to optimize_model(), check_input_tree(), and search_landscape() in main

5.4 General

- Removed all ranger executables
- Removed treefix_compute
- Removed rangerdtl model from models module
- Modified setup files to install the proper executables
- Updated treefixTP.py with version and software info
- Updated INSTALL.txt, CHANGES.txt, and README.txt
- Added README.pdf

References

[1] Niema Moshiri. FAVITES: simultaneous simulation of transmission networks, phylogenetic trees, and sequences. https://www.biorxiv.org/content/early/2018/04/18/297267, 2018.

[2] A. Stamatakis. RAxML Version 8: A tool for Phylogenetic Analysis and Post-Analysis of Large Phylogenies. *Bioinformatics*, 2014.

[3] Tandy Warnow. *Computational Phylogenetics.* Cambridge University Press, 2017.

[4] Yi-Chieh Wu, Matthew D. Rasmussen, Mukul S. Bansal, and Manolis Kellis. Treefix: Statistically informed gene tree error correction using species trees. *Systematic Biology*, 2013.

[5] Chris Wymant, Matthew Hall, Oliver Ratmann, David Bonsall, Tanya Golubchik, Mariateresa de Cesare, Astrid Gall, Marion Cornelissen, Christophe Fraser, STOP-HCV Consortium, The Maela Pneumococcal Collaboration, and The BEEHIVE Collaboration. PHYLOSCANNER: Inferring transmission from within- and between-host pathogen genetic diversity. *Molecular Biology and Evolution*, 2017.