

TNet (version 1.2)

Description

TNet is a phylogeny-based method for reconstructing transmission networks for infectious diseases. It takes as input a phylogeny of the strain (pathogen) sequences sampled from infected hosts and analyzes it to estimate the underlying transmission network. TNet relies on the availability of multiple strain sequences from each sampled host to infer transmissions and is simpler and more accurate than existing approaches. Each run of TNet on the same input tree can result in a different estimate of the transmission network, and so TNet should be executed multiple times (say 100) on the input phylogeny and an aggregated transmission network should be constructed from the resulting outputs. The method is parameter-free and highly scalable and can be easily applied within seconds to datasets with hundreds of strain sequences and hosts.

TNet implements algorithms described in the following paper:

[TNet: Phylogeny-Based Inference of Disease Transmission Networks Using Within-Host Strain Diversity](#)

Saurav Dhar, Chengchen Zhang, Ion Mandoiu, Mukul S. Bansal

Bioinformatics Research and Applications; ISBRA 2020, LNCS 12304: 203-216

TNet is implemented in Python (version 3.0 or greater) and is freely available open source under GNU GPL. TNet also uses the Biopython library that can be freely downloaded as described at <https://biopython.org/>. Further details on Biopython appear in the following paper:

Cock, P.J.A. et al. *Biopython: freely available Python tools for computational molecular biology and bioinformatics*. *Bioinformatics* 2009 Jun 1; 25(11) 1422-3.

Usage

TNet takes as input a single rooted binary phylogeny on all strain sequences sampled from the infected hosts considered in the analysis. This phylogeny must be in Newick format. Such a phylogeny can be constructed using standard phylogeny construction tools such as RAxML or PhyML and then rooting the resulting unrooted phylogeny using standard rooting methods. Each leaf label in this phylogeny must be of the form <hostID>_<sequenceID>. Only <hostID> is used by TNet.

To use TNet, an input file and an output file must be specified as follows:

```
tnet.py inputFile outputFile  
or  
python3 tnet.py inputFile outputFile
```

By default, TNet executes the minimum back-transmission sampling algorithm described in the manuscript cited above, which greedily samples from among all optimal solutions those that minimize the number of back-transmissions. However, options for uniform random sampling of all optimal

solutions (not necessarily minimizing the number of back-transmissions), as well as for computing a highest-probability optimal solution are also available as described below.

Optional command-line parameters

usage: tnet.py [-h] [-sd SEED] [-rs] [-mx] [-t TIMES] [-lt] [-v] INPUT_TREE_FILE OUTPUT_FILE

positional arguments:

INPUT_TREE_FILE	input file name
OUTPUT_FILE	output file name

optional arguments:

-h, --help	show this help message and exit
-sd SEED, --seed SEED	seed for random number generator
-rs, --randomsampling	sample optimal solutions uniformly at random
-mx, --maxprob	compute highest-probability optimal solution
-t, --times	sample TNet multiple times and output the summary
-lt, --labeledtrees	output labeled trees in addition to transmission network
-v, --version	show program's version number and exit

Interpretation of the output

By default, each execution of TNet on an input file outputs a single transmission network based on a single sampled solution for the underlying computational problem (i.e., for minimum back-transmission sampling, uniform sampling among all optimal solutions, or highest-probability optimal solution). The output file lists all inferred edges (connecting two hosts) in the transmission network. A sample output follows:

```
None 124
124 123
124 122
124 121
124 121
```

In this output, the first line indicates that host 124 is the source of the transmission, the second line indicates that there was a transmission from host 124 to host 123, and so on. Note that some transmission edges may be listed multiple times (e.g., the transmission from 124 to 121 in the sample output above). Any repeated edges can be ignored; alternatively, repeated edges may indicate that multiple distinct pathogen lineages were transmitted during the transmission.

Aggregating results across multiple samples

As noted above, each execution of TNet on an input file outputs a single transmission network based on an appropriately sampled solution to an underlying computational problem. Thus, TNet should be executed multiple times (say 100 times) on a single input file and results should be aggregated across all

output transmission networks. This aggregation across multiple executions can be performed automatically using the `-t` command line option. For example, the following command will compute 100 transmission network samples and will output an aggregated transmission network obtained by aggregating across these 100 samples.

```
tnet.py -t 100 inputFile outputFile  
or  
python3 tnet.py -t 100 inputFile outputFile
```

The output aggregated transmission network shows each inferred transmission edge along with its support value (i.e., number of sampled transmission networks that contained that edge). A sample output follows:

```
174->37      90  
453->62      74  
177->59      55  
59->228      5  
37->211      4
```

In this output, each line indicates a transmission edge. The first line indicates that there was a transmission from host 174 to host 37 and that this transmission edge was observed in 90 out of the 100 sampled transmission networks. Note that in computing these support values transmission edges are only counted once per transmission network sample even if they appear multiple times in a single sample.

To further improve inference accuracy, we also suggest aggregating across multiple bootstrap replicates of the input phylogeny, as done in the manuscript cited above.

Outputting labeled trees

TNet infers transmission networks by first assigning a host label to each internal node of the input tree. If needed, TNet can output the specific labeled tree(s) used to infer the computed transmission network(s). This can be done by using the `-lt` command line option. The labeled tree (or labeled trees, if the `-t` option is used) is then written in Newick format to the file “outputFile.newick”, where “outputFile” is the output file name specified in the command.

Example datasets

The “input” directory in the git repository contains some sample rooted phylogenies that can be used as input for TNet.