# TNet-Geo (version 1.0)

## Description

TNet-Geo is a tool for Transmission Network analysis on multiple strain sequences from different geographic regions. It uses the TNet tool with some modifications and instead of multiple samples per host it uses multiple samples per region.

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
*International Symposium on Bioinformatics Research and Applications (ISBRA) 2020*: LNCS; to appear.

TNet-Geo 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-Geo 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 in this phylogeny must be labeled with the "strainID" or in the format <regionName>_<strainID>.

To do the time series regional analysis user has to use a separate metadata file to specify the "date" for each of the tree nodes and "regionName" for each of the leaf nodes. The node labels should under the "strain" column. The corresponding dates should be under the "date" column and the corresponding regionNames should be under the "country" column. All the node labels should match the node names in the Newick file. The dates should be in "YYYY-MM-DD" format. For the non-leaf nodes the "country" column should be empty.

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

```
Tnet_geo.py inputFile outputFile
```
or
```
python3 tnet_geo.py inputFile outputFile
```
or
```
python3 tnet_geo.py inputFile outputFile -md metadata.csv
```

By default, TNet-Geo executes the uniform random sampling algorithm described in the manuscript cited above. However, options for minimum back-transmission sampling of all optimal solutions, as well as for highest-probability solution sampling are also available as described below.

## Optional command-line options

```
usage: tnet_geo.py INPUT_TREE_FILE OUTPUT_FILE [-h] [-sd SEED] [-bs] [-mx] [-md METADATA] [-t
TIMES] [-ex] [-v]

positional arguments:
  INPUT_TREE_FILE       input file name
  OUTPUT_FILE           output file name

optional arguments:
  -h, --help                    show this help message and exit

  -md, --metadata               csv file with meta data for the tree nodes
  -sd SEED, --seed SEED         random number generator seed
  -bs, --biasedsampling         sample optimal solutions with minimum back-transmission
  -mx, --maxprob                compute highest-probability solution
  -t, --times                   sample TNet multiple times

  -ex, --extradata              output a json file with extra output data for further analysis
  -v, --version                 show program's version number and exit
```

## Interpretation of the output

The output file gives the labeled trees in Newick format. If you use "-t" options all the labeled trees from multiple runs will be in the output file. If you use the "-ex" option TNet-Geo will output a JSON file with three useful list for further analysis. The first one is "Country of exposure" which gives you the region of exposure for each strain along with their counts. The second one is "Transmission edges" which gives the summary of all the transmission edges along with their counts for the multiple runs. The last one is the "Dated edges" which is a collection of all the transmission edges from multiple runs and their corresponding dates. These "Dated edges" can further be analyzed for time series analysis.

For Country of exposure the output format is:

```
        "EPI_ISL_426890": {"count": {"CzechRepublic": 100}
```
For Transmission edges the output format is:

```
        "Germany->Italy": 88
```

Dated edges the output format is:

```
        ["Italy->Hungary", "2020-02-10"]
```

As noted above, each execution of TNet-Geo on an input file outputs a single labeled tree based on an appropriately sampled optimal solution to an underlying computational problem. Thus, TNet-Geo should be executed multiple times (say 100 times) on a single input file and results should be aggregated across all output transmission networks. To further improve inference accuracy, we also suggest aggregating across multiple bootstrap replicates of the input phylogeny, as done in the manuscript cited above.

## Example datasets

The “input” directory in the git repository contains some sample rooted phylogenies that can be used as input for TNet. The ‘’output” folder contains the corresponding outputs of them.

Try,

```
python3 tnet_geo.py input/0.tree output/0.out -md input/0.metadata.csv
-t 100 -ex
```

## Further analysis of the JSON file

Here we provide another script “analyze_json.py” for some further analysis of the JSON file we got from TNet-Geo. Currently this script can group all the transmission edges by months using the “-eg” or “--edgedategroup” argument. It will generate a CSV file containing all the transmission edges and their respective counts during each month. Please check the “output” folder to find the demo output “0.edgegroups.csv” for better understanding.

Try,

```
python3 analyze_json.py output/0.out.json -eg output/0.edgegroups.csv
```