TriMouNet User Manual

Mao Qiyun

Overview

This manual describes a complete TriMouNet workflow in three stages:

1. Quartet extraction and preprocessing (extract_outgroup_quartets.py): extract all outgroup-rooted quartets from full gene trees, clean/normalize branch lengths, filter outliers, and generate a branch-length–free, numeric-encoded file for fast split counting.
2. Parameter estimation and TNET generation (infer_trinet_types_and_generate_tnets.py): estimate model parameters from quartets (or quartet-derived statistics) and aggregate them into a *.tnets file compatible with TriMouNet.
3. Network reconstruction (TriMouNet.jar): infer the final phylogenetic network from the *.tnets input and output the network topology and associated parameters (e.g., w1 for reticulated cherries).

Part I — Quartet extraction and preprocessing (extract_outgroup_quartets.py)

I.1 Purpose

The goal of this script is to convert full gene trees (spanning all taxa) into outgroup-rooted quartet trees required by downstream TriMouNet processing. In addition, it applies a sequence of cleaning and normalization steps to ensure:

Quartet trees are comparable (consistent Newick layout with outgroup placement fixed);

Branch-length values are numerically stable (avoid exact zeros and pathological rooting artifacts);

An auxiliary file is produced with branch lengths removed and taxa mapped to integers, enabling efficient split counting in later stages.

I.2 Inputs

A text file containing Newick gene trees, one tree per line, each ending with ';'.

In practice, users should first reconstruct gene trees for multiple loci (e.g., using IQ-TREE on per-locus alignments). The resulting locus-wise maximum-likelihood trees should then be combined into a single .tre file, where each line corresponds to one locus tree in Newick format. This combined multi-locus tree file is used as the input to the quartet-extraction script. In this manual, we illustrate the workflow using the example file "simulatednetwork_middle.tre", which contains 4000 simulated gene trees for eight ingroup taxa {A..H}, a single outgroup taxon O. Each gene tree was reconstructed from a simulated alignment of 5,000 bp.

I.3 Outputs

The script produces intermediate files for transparency and debugging:

- ***_quartet.tre** (Step 1): all pruned quartet trees containing {triplet + O}, written with fixed branch-length precision.
- ***_quartet_adjust.tre** (Step 2): quartets after branch-length cleanup and root–outgroup adjustment.
- ***_deleteoutlier.tre** (Step 3): quartets after removing outlier branch-length configurations.
- ***_modify.tre** (Step 4): normalized Newick with outgroup O placed consistently on the outside.
- ***_withoutweight_num.tre** (Step 5): branch-length–stripped, outgroup-removed, and numeric-encoded taxa (A..H → 1..8), for downstream split statistics.

I.4 Step-by-step description

Step 1 — Quartet extraction

Enumerate all 3-taxon subsets of the ingroup and prune each full gene tree down to {triplet + O}.

Step 2 — Branch-length cleanup and root–outgroup adjustment

Replace exact zero-length branches with 0.00001 to avoid invalid or collapsed trees.

If the root has exactly two child clades and one child is the outgroup O, then push the internal root edge length onto the outgroup edge and set the subtree root length to 0. This prevents pruning-induced artifacts where O becomes artificially close to the root.

Step 3 — Outlier filtering

Remove quartet trees with extreme branch-length patterns to reduce numerical artifacts that can bias downstream parameter estimation. We apply two filters: Ultra-short branch filter: discard any quartet tree in which at least one branch length is lower than 0.0002. In this manual, the simulated sequence length is 5,000 bp, so we use 1/5000 as the default cutoff. In real analyses, users may adjust this threshold; we recommend setting it to the reciprocal of the average alignment length (in bp) across loci; Dominant long-branch filter: discard any quartet tree in which a single branch length accounts for more than 50% of the total tree length, to avoid cases where an unusually long branch dominates the branch-length structure.

Step 4 — Newick normalization (fixing outgroup placement)

Rewrite Newick strings so that outgroup O appears in a consistent, standardized position (e.g., as the outermost first child). For example, a normalized quartet tree can be written as:

(O:0.24311,A:0.05052,(B:0.04465,C:0.04931):0.00594);

Step 5 — Remove branch lengths and encode taxa as integers

Strip all branch lengths, remove O, and map A..H to 1..8. This produces a compact file optimized for split counting. For example, after deleting **O**, the resulting three-taxon tree can be written as "(1,(2,3));".

Part II—Parameter estimation and TNET generation (infer_trinet_types_and_generate_tnets.py)

II.1 Purpose

This stage converts quartet trees (or quartet-derived statistics) into a ".tnets" input file for TriMouNet. It consists of:

1. estimating model parameters for each multilocus3-taxa subset;
2. assigning an appropriate trinet type to each 3-taxa subset based on the estimated parameters;
3. writing a ".tnets" file containing one entry per 3-taxon subset, plus a loci-count line.

II.2 Inputs

Depending on your implementation, typical inputs include:the normalized quartet file from Part I (*_modify.tre) and the numeric-encoded file (*_withoutweight_num.tre) for split counting.

II.3 Outputs: .tnets file format

A .tnets file is a whitespace-delimited text file. Each trinet line begins with three taxa labels and a trinet type, followed by the parameters required by that type:

A B C T1/N2/N3/N4 gap_pending_edge gap_cherry_edge w1_cherry u1_cherry u2_cherry w1_pending u1_pending u2_pending p_value_second p_value_first

A D B S1 gap_common_edge p_value_second p_value_first weight1 weight2

and the final file ends with a line recording the locus count, e.g.:

N U M Num 4000 (To approximate the average number of loci per trinet, we recommend dividing the number of trees in "_modify.tre" by the number of assigned trinets.)

II.4 Step-by-step description of extract_outgroup_quartets.py

Step 1 — Enumerate all ingroup triplets (8 choose 3)

Step 2 — Count split/topology frequencies from *_withoutweight_num.tre

Step 3 — Parameter fitting by optimization (BFGS / L-BFGS-B) ...

Step 4 — Write the final.tnets file (trinet type assignment)

Part III—Network reconstruction (TriMouNet.jar)

II.1 Purpose

This step uses TriMouNet to convert the inferred trinet set (.tnets) into a single phylogenetic network over all taxa, exported as a DOT graph for visualization with GraphViz.

II.2 Command

Run TriMouNet

From the command line, run TriMouNet and specify an output name/prefix:

```
java -jar TriMouNet.jar dataset.tnets
```

Visualize with GraphViz

Render the DOT file to a figure using GraphViz:

```
dot -Tpdf myOutput.dot -o myOutput.pdf
# or
dot -Tpng myOutput.dot -o myOutput.png
```