SBCC26: IQ-TREE Instructions

Gauri Renjith (UCSD Supercomputing)

All clarifications made in green (please review full document!)

Overview

IQ-TREE is a bioinformatics software tool used to reconstruct phylogenetic trees from molecular sequence data such as DNA, RNA, or proteins. Through tasks that resemble real-world scientific analyses, teams will demonstrate understanding of the IQ-TREE software and its biological context.

Teams are required to use IQ-TREE version 2.4.0. Any source code modifications were expected to be disclosed to the committee by April 7th.

Submission Instructions

Teams will upload a tarball titled `iqtree.tar` to the shared Google Drive folder. The expected directory structure for the submission is as follows:

```
.
├── task_1 (directory with all deliverables for task 1)
├── task_2 (directory with all deliverables for task 2)
├── task_3 (directory with all deliverables for task 3)
├── task_4 (directory with all deliverables for task 4)
└── task_5 (directory with all deliverables for task 5)
```

General Clarifications

- For tasks 1-4 and task 5 baseline, you may only use additional flags from IQ-TREE's general options and checkpoint options (from general options, `--fconst` and `--epsilon` cannot be used). Flags from other sections should not be used unless necessary for the task. For the second part of task 5, any flag may be used.

Task 1: Viper Venom (8%)

You wake up 30 years in the future and realize you are now working as a bioinformatician. Your role is to specialize in IQ-TREE and perform high-quality phylogenetic analyses.

Before you can contribute to real research projects, you need to understand how IQ-TREE behaves under different settings—how model choice, initialization, and search parameters affect the resulting tree and likelihood. You have a small dataset of viper venom protein genes that you can use to learn along the way.

Dataset: `M6414.nex`
Random seed: `10042026` (use for all IQ-TREE commands)

Task 1A
Perform the following tree searches:

- Fixed model: `GTR+G4`
    - use `--prefix fixed`
- Tree search and model selection using ModelFinder (`-m MFP`)
    - use `--prefix mfp`

Deliverables:

- `fixed.tar`: tarball of all files from running the fixed model
- `mfp.tar`: tarball of all files from running MFP

Task 1B
Using the best-fit model found in Task 1A, perform the following tree searches:

- Vary the number of initial trees: `-ninit 10` and `-ninit 200`
    - use `--prefix ninit_10` and `--prefix ninit_200`
- Vary the number of top trees retained: `-ntop 5` and `-ntop 50`
    - use `--prefix ntop_5` and `--prefix ntop_50`
- Vary the number of best trees: `-nbest 1` and `-nbest 20`
    - use `--prefix nbest_1` and `--prefix nbest_20`

Deliverables:

- `ninit_10.tar, ninit_200.tar, ntop_5.tar, ntop_50.tar, nbest_1.tar, nbest_20.tar`: tarballs that contain all IQ-TREE output files generated by the corresponding commands
- `results.csv`: table with log-likelihood and total CPU time used for each variation

Task 1C

Let n be the max number of threads on one node. Using the best fit model found in Task 1A, perform tree search with:

- 1 MPI process, n threads (single node run)
    - use `--prefix 1p_nt` (do not change n to a number)
- 2 MPI processes, n/2 threads (single node run)
    - use `--prefix 2p_nby2t` (do not change n to a number)
- 4 MPI processes, n/4 threads (single node run)
    - use `--prefix 4p_nby4t` (do not change n to a number)
- 2 MPI processes, n threads (two-node run)
    - use `--prefix 2p_nt` (do not change n to a number)
- 4 MPI processes, n threads (four-node run)
    - use `--prefix 4p_nt` (do not change n to a number)

Deliverables:

- `1p_nt.tar, 2p_nby2t.tar, 4p_nby4t.tar, 2p_nt.tar, 4p_nt.tar`: tarballs that contain all IQ-TREE output files generated by the corresponding commands.
- `results.csv`: table with log-likelihood and total CPU time used for each variation.

Task 2: Pokémon (14%)

Scientists have discovered a distant planet that hosts 200 Pokémon species. From each species, 5 genes have been sequenced. Your research team wants to understand how these genes evolved; you will analyze partitioned data, compare models, and simulate new datasets to better understand the evolutionary structure of this ecosystem.

Dataset: `pokemonn.phy`
Random seed: `10042026` (use for all IQ-TREE commands)

Task 2A
We want to find the best model for this dataset

1. Run tree search with ModelFinder (`-m MFP`)
    a. use `--prefix pokemon_original`
2. Find the best model for each individual gene in the alignment
    a. the total sequence is equivalent to gene 1 + gene 2 + gene 3 + gene 4 + gene 5
        i. gene 1 has length 1324, gene 2 has length 898, gene 3 has length 1371, gene 4 has length 564, gene 5 has length 3863
    b. for each individual gene from 1-5, use `--prefix pokemon_gene_$i` where `$i` is the gene's number
3. Run tree search with a partition file based on the best-fit models for each gene
    a. use `--prefix pokemon_bestfit`
    b. use an edge-linked equal partition model

Deliverables:

- `pokemon_original.tar`: tarball of all output files from the first tree search
- `pokemon_gene_1.tar, pokemon_gene_2.tar, pokemon_gene_3.tar, pokemon_gene_4.tar, pokemon_gene_5.tar`: tarballs that contain all IQ-TREE output files generated for the corresponding gene
- `pokemon_bestfit.tar`: tarball of all output files from the tree search with the partition file AND
    - `pokemon_bestfit.nex:` the partition file used

`pokemon_bestfit.nex` should be of the format:

```
#nexus
begin sets;
    charset gene_1 = DNA, XXX-XXX;
    charset gene_2 = DNA, XXX-XXX;
    charset gene_3 = DNA, XXX-XXX;
    charset gene_4 = DNA, XXX-XXX;
    charset gene_5 = DNA, XXX-XXX;
    charpartition mine =
        XXX:gene_1,
```

```
        XXX:gene_2,
        XXX:gene_3,
        XXX:gene_4,
        XXX:gene_5;
end;
```

Task 2B

1. Simulate alignments with the partition file from task 2A and the `pokemon_bestfit` treefile
    a. Alignments should have the same gene lengths and structure
2. Run tree search and ModelFinder (`-m MFP`) on simulated alignment
    a. Use `--prefix pokemon_sim_search`
3. Compute all-to-all Robinson-Foulds distances between the 3 trees

Deliverables

- `pokemon_sim.aln`: alignment simulated from partition file
- `pokemon_sim_search.tar`: all files from MFP on simulated alignment
- `pokemon_rf.rfdist`: file with Robinson-Foulds distances

Task 3: Ebola (8%)

You are part of a rapid-response team tracking an ongoing Ebola outbreak. You currently have 400 viral samples; suddenly, 50 new samples arrive from remote regions, and you need to integrate these samples into your analyses.

Datasets: `ebola.400.01.true.aln, ebola.50.aln`
Random seed: `10042026` (use for all IQ-TREE commands)

1. Construct a tree using the initial 400 sequences in `ebola.400.01.true.aln`
    a. Use `--prefix ebola_original`
2. Insert 50 new sequences from `ebola.50.aln` into the existing tree and identify closest relatives
    a. Use `--prefix ebola_add_seq`
3. Reoptimize the new tree
    a. Use `--prefix ebola_reopt`

Deliverables

- `original.tar`: all IQ-TREE output files from the initial tree
- `add_seq.tar`: all IQ-TREE output files from sequence insertion
- `reopt.tar`: all IQ-TREE output files from reoptimization
- `relatives.csv`: table listing closest related sequence(s) for each of the 50 new sequences
- `log_likehoods.csv`: table of log-likelihoods for the initial tree, tree after insertion, and reoptimized tree

Task 4: HIV-1 (25%)

Your research team has just acquired new HPC hardware to scale up phylogenetic analyses using IQ-TREE. Before deploying it on large projects, you need to understand how well your system scales—both within a single node (multithreading) and across multiple nodes.

Using UFBoot bootstrap analyses on HIV datasets, you will measure how runtime improves as you increase computational resources, and evaluate how efficiently your system uses those resources.

Random seed: `10042026` (use for all IQ-TREE commands)

Task 4A: Multithread Scaling (single node)
Dataset: `hiv.100.01.true.aln`

- Use 1000 bootstrap replicates and 200 as the maximum number of iterations
- Vary number of threads within one node

Deliverables:

- `multithread_scaling.tar`: all IQ-TREE output files in one tarball (use different prefixes for different thread counts and different repeats) AND
    - `README.md`: documentation explaining the naming convention
- `cpu_scaling.png`: plot of total CPU time vs. thread count
- `speedup.png`: plot of speedup vs. thread count
- `efficiency.png`: plot of parallel efficiency vs. thread count

Task 4B: Multinode Scaling
Dataset: `hiv.200.01.true.aln`

- Use 2000 bootstrap replicates and 800 as the maximum number of iterations
- Run across node counts from 1 to the maximum available

Deliverables:

- `multinode_scaling.tar`: all IQ-TREE output files in one tarball (use different prefixes for different core counts and different repeats) AND
    - `README.md`: documentation explaining the naming convention on
- `cpu_scaling.png`: plot of total CPU time vs. thread count
- `speedup.png`: plot of speedup vs. thread count
- `efficiency.png`: plot of parallel efficiency vs. thread count

Additional information:

- Use powers of 2 where possible:
    - Example (cores): 1, 2, 4, 8

    - Example (nodes): 1, 2, 4, 8, 16
- If fewer than 8 nodes are available:
    - Must include at least 4 data points
        - Example:
            - 4–5 nodes → {1, 2, 3, 4}
            - 6–7 nodes → {1, 2, 4, 6}
- Teams are encouraged to repeat runs for variance/error bars
    - Teams may skip repeats if the runtime is prohibitive
- Speedup is calculated as S(n) = T(1) / T(n)
    - T(1) is the total wall clock runtime on 1 thread (task 4A) or 1 node (task 4B)
    - T(n) is the runtime on n thread (task 4A) or n node (task 4B)
- Efficiency is calculated E(n) = S(n) / n

Task 5: SARS-CoV-2 (20%)

Following the COVID-19 pandemic, you now have access to a large dataset of 1,600 SARS-CoV-2 sequences. Your goal is to construct a phylogenetic tree that maximizes the log-likelihood. Starting from a baseline analysis, you will experiment with different strategies in IQ-TREE to improve the result.

Dataset: `covid.1600.01.viralmsa.aln`

Task 5A

Run tree search with ModelFinder (`-m MFP`) to get a baseline.

- Use `--prefix covid_mfp` and `--seed 10042026`

Deliverables

- `covid_mfp.tar`: tarball with all output files from the IQ-TREE command

Task 5B

Run tree search to get the best possible log-likelihood score; teams can modify search parameters, model selections, IQ-TREE flags (including seed), etc.

Deliverables

- `covid_best.tar`: contains all output files from the IQ-TREE tree search run with the best log-likelihood score AND
    - `covid_strategy.txt`: a short paragraph on the approach

Task 6: Interview (25%)

This interview will likely be 15-20 minutes long. Be prepared to answer questions on your thought process, approaches to the tasks, and understanding of IQ-TREE!

Acknowledgements

Ebola, HIV-1, and SARS-CoV-2 alignments were downloaded from the ViralMSA repository (Moshiri, 2021). Ebola and HIV-1 "true" alignments were originally sourced from the LANL sequence database; the SARS-CoV-2 alignment was generated with ViralMSA. The M6414 alignment was downloaded from TreeBASE (study TB2:S10791; Casewell et al., 2010).

Special thanks to Professor Niema Moshiri (UC San Diego) for his guidance in identifying datasets for this competition.

References

Casewell, N.R., Wagstaff, S.C., Harrison, R.A., & Wüster, W. (2010). Gene tree parsimony of multi-locus snake venom protein families reveals species tree conflict as a result of multiple parallel gene loss. Molecular Biology and Evolution, 28(3), 1157–1172.

Moshiri, N. (2021). ViralMSA: massively scalable reference-guided multiple sequence alignment of viral genomes. Bioinformatics, 37(5), 714–716. https://doi.org/10.1093/bioinformatics/btaa743

Piel, W. H., Chan, L., Dominus, M. J., Ruan, J., Vos, R. A., and V. Tannen 2009. TreeBASE v. 2: A Database of Phylogenetic Knowledge. In: e-BioSphere 2009

Vos, R. A., Balhoff, J. P., Caravas, J. A., Holder, M. T., Lapp, H. Maddison, W. P., Midford, P. E., Priyam, A., Sukumaran, J., Xia, X. and A. Stoltzfus 2012. NeXML: rich, extensible, and verifiable representation of comparative data and metadata. Systematic Biology 61(4): 675-689.