IQ-TREE version 2.2.0: Tutorials and Manual
Phylogenomic software by maximum likelihood

http://www.iqtree.org

Bui Quang Minh, Rob Lanfear, Nhan Ly-Trong
Jana Trifinopoulos, Dominik Schrempf, Heiko A. Schmidt

March 25, 2022

Preface

Contents

Chapter 1

Introduction

1.1 Why IQ-TREE?

Thanks to the recent advent of next-generation sequencing techniques, the amount of phylogenomic/transcriptomic data have been rapidly accumulated. This extremely facilitates resolving many "deep phylogenetic" questions in the tree of life. At the same time it poses major computational challenges to analyze such big data, where most phylogenetic software cannot handle. Moreover, there is a need to develop more complex probabilistic models to adequately capture realistic aspects of genomic sequence evolution.

This trends motivated us to develop the IQ-TREE software with a strong emphasis on phylogenomic inference. Our goals are:

- **Accuracy**: Proposing novel computational methods that perform better than existing approaches.
- **Speed**: Allowing fast analysis on big data sets and utilizing high performance computing platforms.
- **Flexibility**: Facilitating the inclusion of new (phylogenomic) models and sequence data types.
- **Versatility**: Implementing a broad range of commonly-used maximum likelihood analyses.

IQ-TREE has been developed since 2011 and freely available at http://www.iqtree.org/ as open-source software under the GNU-GPL license version 2. It is actively maintained by the core development team (see below) and a number of collabrators.

The name IQ-TREE comes from the fact that it is the successor of **IQ**PNNI and **TREE**-PUZZLE software.

1.2 Key features

- **Efficient search algorithm**: Fast and effective stochastic algorithm to reconstruct phylogenetic trees by maximum likelihood. IQ-TREE compares favorably to RAxML and PhyML in terms of likelihood while requiring similar amount of computing time (Nguyen et al., 2015).
- **Ultrafast bootstrap**: An ultrafast bootstrap approximation (UFBoot) to assess branch supports. UFBoot is 10 to 40 times faster than RAxML rapid bootstrap and obtains less biased support values (Minh et al., 2013; Hoang et al., 2018).
- **Ultrafast model selection**: An ultrafast and automatic model selection (ModelFinder) which is 10 to 100 times faster than jModelTest and ProtTest. ModelFinder also finds best-fit partitioning scheme like PartitionFinder.
- **Big Data Analysis**: Supporting huge datasets with thousands of sequences or millions of alignment sites via checkpointing, safe numerical and low memory mode. Multicore CPUs and parallel MPI system are utilized to speedup analysis.
- **Phylogenetic testing**: Several fast branch tests like SH-aLRT and aBayes test (Anisimova et al., 2011) and tree topology tests like the approximately unbiased (AU) test (Shimodaira, 2002).

The strength of IQ-TREE is the availability of a wide variety of phylogenetic models:

- **Common models**: All common substitution models for DNA, protein, codon, binary and morphological data with rate heterogeneity among sites and ascertainment bias correction for e.g. SNP data.
- **Partition models**: Allowing individual models for different genomic loci (e.g. genes or codon positions), mixed data types, mixed rate heterogeneity types, linked or unlinked branch lengths between partitions.
- **Mixture models**: fully customizable mixture models and empirical protein mixture models and.
- **Polymorphism-aware models**: Accounting for *incomplete lineage sorting* to infer species tree from genome-wide population data (Schrempf et al., 2016).

1.3 Free web server

For a quick start you can also try the IQ-TREE web server, which performs online computation using a dedicated computing cluster. It is very easy to use with as few as just 3 clicks! Try it out at

http://iqtree.cibiv.univie.ac.at

1.4 User support

Please refer to the user documentation and frequently asked questions. If you have further questions, feedback, feature requests, and bug reports, please sign up the following Google group (if not done yet) and post a topic to the

https://groups.google.com/d/forum/iqtree

The average response time is two working days.

1.5 Documentation

IQ-TREE has an extensive documentation with several tutorials and manual:

- Getting started guide: recommended for users who just downloaded IQ-TREE.
- Web Server Tutorial: A quick starting guide for the IQ-TREE Web Server.
- Beginner's tutorial: recommended for users starting to use IQ-TREE.
- Advanced tutorial: recommended for more experienced users who want to explore more features of IQ-TREE.
- Command Reference: Comprehensive documentation of command-line options available in IQ-TREE.
- Substitution Models: All common substitution models and usages.
- Complex Models: Complex models such as partition and mixture models.
- Polymorphism Aware Models: Polymorphism-aware phylogenetic Models (PoMo) related documentation.
- Compilation guide: for advanced users who wants to compile IQ-TREE from source code.
- Frequently asked questions (FAQ): recommended to have a look before you post a question in the IQ-TREE group.

1.6 How to cite IQ-TREE?

To maintain IQ-TREE, support users and secure fundings, it is important for us that you cite the following papers, whenever the corresponding features were applied for your analysis.

- Example 1: *We obtained branch supports with the ultrafast bootstrap (Hoang et al., 2018) implemented in the IQ-TREE software (Nguyen et al., 2015).*
- Example 2: *We inferred the maximum-likelihood tree using the edge-linked partition model in IQ-TREE (Chernomor et al., 2016; Nguyen et al., 2015).*

If you performed the tests of symmetry, please cite:

- **S. Naser-Khdour, B.Q. Minh, W. Zhang, E.A. Stone, R. Lanfear** (2019) The prevalence and pmpact of model violations in phylogenetic analysis, *Genome Biol. Evol.*, in press. https://doi.org/10.1093/gbe/evz193

If you used the polymorphism-aware models please cite:

- **D. Schrempf, B.Q. Minh, A. von Haeseler, and C. Kosiol** (2019) Polymorphism-aware species trees with advanced mutation models, bootstrap, and rate heterogeneity. *Mol. Biol. Evol.*, 36:1294-1301. https://doi.org/10.1093/molbev/msz043

If you used the heterotachy model (GHOST) please cite:

- **S.M. Crotty, B.Q. Minh, N.G. Bean, B.R. Holland, J. Tuke, L.S. Jermiin, A. von Haeseler** (2019) GHOST: Recovering historical signal from heterotachously-evolved sequence alignments. *Syst. Biol.*, in press. https://doi.org/10.1093/sysbio/syz051

If you performed the ultrafast bootstrap (UFBoot) please cite:

- **D.T. Hoang, O. Chernomor, A. von Haeseler, B.Q. Minh, and L.S. Vinh** (2018) UFBoot2: Improving the ultrafast bootstrap approximation. *Mol. Biol. Evol.*, 35:518–522. https://doi.org/10.1093/molbev/msx281

If you used posterior mean site frequency model please cite:

- **H.C. Wang, B.Q. Minh, S. Susko and A.J. Roger** (2018) Modeling site heterogeneity with posterior mean site frequency profiles accelerates accurate phylogenomic estimation. *Syst. Biol.*, 67:216-235. https://doi.org/10.1093/sysbio/syx068

If you used ModelFinder please cite:

- **S. Kalyaanamoorthy, B.Q. Minh, T.K.F. Wong, A. von Haeseler, and L.S. Jermiin** (2017) ModelFinder: Fast Model Selection for Accurate Phylogenetic Estimates, *Nature Methods*, 14:587–589. https://doi.org/10.1038/nmeth.4285

If you performed tree reconstruction please cite:

- **L.-T. Nguyen, H.A. Schmidt, A. von Haeseler, and B.Q. Minh** (2015) IQ-TREE: A fast and effective stochastic algorithm for estimating maximum likelihood phylogenies. *Mol. Biol. Evol.*, 32:268-274. https://doi.org/10.1093/molbev/msu300

If you used partition models e.g., for phylogenomic analysis please cite:

- **O. Chernomor, A. von Haeseler, and B.Q. Minh** (2016) Terrace aware data structure for phylogenomic inference from supermatrices. *Syst. Biol.*, 65:997-1008. https://doi.org/10.1093/sysbio/syw037

If you used the IQ-TREE web server please cite:

- **J. Trifinopoulos, L.-T. Nguyen, A. von Haeseler, and B.Q. Minh** (2016) W-IQ-TREE: a fast online phylogenetic tool for maximum likelihood analysis. *Nucleic Acids Res.*, 44 (W1):W232-W235. https://doi.org/10.1093/nar/gkw256

1.7 Development team

IQ-TREE is actively developed by:

Bui Quang Minh, *Team leader*, Designs and implements software core, tree search, ultrafast bootstrap, model selection.

Robert Lanfear, *Co-leader*, Model selection.

Olga Chernomor, *Developer*, Implements partition models.

Heiko A. Schmidt, *Developer*, Integrates TREE-PUZZLE features.

Dominik Schrempf, *Developer*, Implements polymorphism-aware models (PoMo).

Michael Woodhams, *Developer*, Implements Lie Markov models.

Diep Thi Hoang, *Developer*, Improves ultrafast bootstrap.

Arndt von Haeseler, *Advisor*.

Past members:

Lam Tung Nguyen, *Developer*, Implemented tree search algorithm.

Jana Trifinopoulos, *Developer*, Implemented web service.

1.8 Credits and acknowledgements

Some parts of the code were taken from the following packages/libraries: Phylogenetic likelihood library, TREE-PUZZLE, BIONJ, Nexus Class Libary, Eigen library, SPRNG library, Zlib library, gzstream library, vectorclass library, GNU scientific library.

IQ-TREE was funded by the Austrian Science Fund (grant no. I760-B17 from 2012-2015 and I 2508-B29 from 2016-2017), the University of Vienna (Initiativkolleg I059-N from 2012-2015), the Australian National University (2018-onwards), Chan-Zuckerberg Initiative (2020).

Chapter 2

Getting started

Recommended for users who just downloaded IQ-TREE the first time.

2.1 IQ-TREE web server

The quickest is to try out the IQ-TREE web server, where you only need to upload an alignment, choose the options and start the analysis. There is a web server tutorial here.

If you want to use the command-line version, follow the instructions below.

2.2 Installation

For reasons of performance, IQ-TREE is a command-line program, i.e., IQ-TREE needs to be run from a terminal/console (command prompt under Windows).

2.2.1 Packages and bundles

Ready made IQ-TREE packages are available for the following distributions/repositories (command to install iqtree):

- Debian Linux: `sudo apt-get install iqtree`
- Arch Linux (AUR)
- Anaconda: `conda install -c bioconda iqtree`
- Homebrew: `brew install brewsci/bio/iqtree2`
- FreeBSD: `pkg install iqtree`

2.2.2 Manual download

IQ-TREE for Windows, MacOSX and Linux can be downloaded here.

- Extract the `.zip` (Windows, MacOSX) or `.tar.gz` (Linux) file to create a directory `iqtree-X.Y.Z-OS`, where `X.Y.Z` is the version number and `OS` is the operating system (Windows, MacOSX or Linux).
- You will find the executable in the `bin` sub-folder. Copy all files in `bin` folder to your system search path such that you can run IQ-TREE by entering `iqtree` from the Terminal.

Now you need to open a Terminal (or Console) to run IQ-TREE. See below the guide for Windows users and Mac OS X users.

2.3 For Windows users

Since IQ-TREE is a command-line program, clicking on `iqtree.exe` will not work. You have to open a Command Prompt for all analyses:

1. Click on "Start" menu (below left corner of Windows screen).
2. Type in "cmd" and press "Enter". It will open the Command Prompt window (see Figure below).
3. Go into IQ-TREE folder you just extracted by entering e.g. (assuming you downloaded version 1.5.0):

   ```
   cd Downloads\iqtree-1.5.0-Windows
   ```

   (assuming that IQ-TREE was downloaded into `Downloads` folder).
4. Now you can try an example run by entering:

   ```
   bin\iqtree -s example.phy
   ```

   (`example.phy` is the example PHYLIP alignment file also extracted in that folder).
5. After a few seconds, IQ-TREE finishes and you may see something like this:

Congratulations ;-) You have finished the first IQ-TREE analysis.

2.4 For Mac OS X users

1. Open the "Terminal", e.g., by clicking on the Spotlight icon (top-right corner), typing "terminal" and press "Enter".

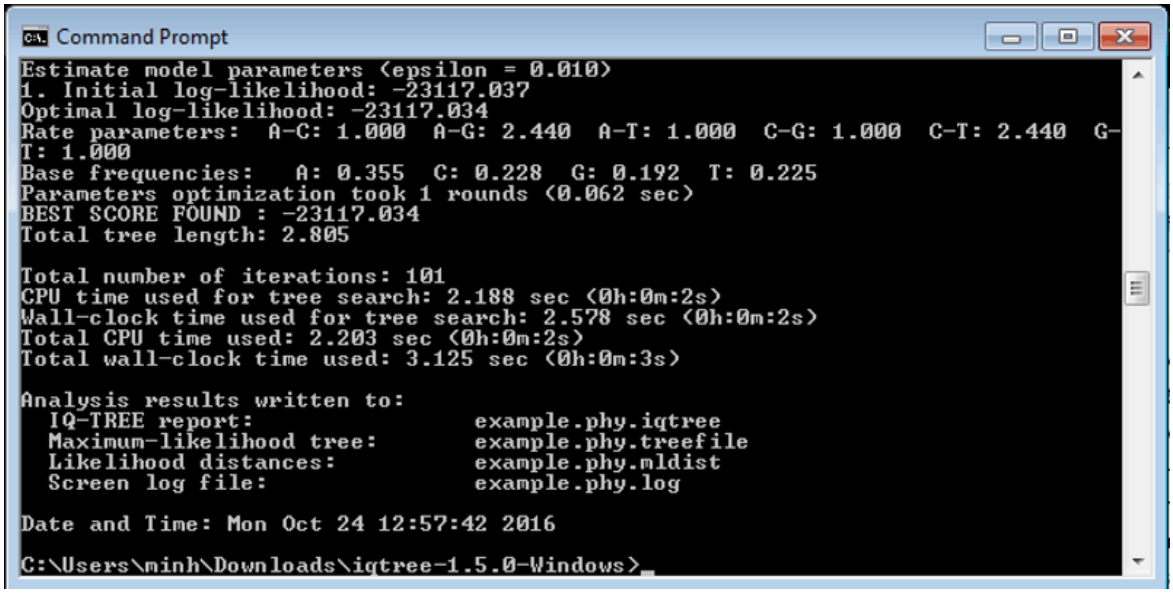


Figure 2.1: Windows command prompt

2. Go into IQ-TREE folder by entering (assuming you downloaded version 1.5.0):

```
cd Downloads/iqtree-1.5.0-MacOSX
```

(assuming that IQ-TREE was downloaded into `Downloads` folder).

3. Now you can try an example run by entering

```
bin/iqtree -s example.phy
```

(`example.phy` is the example PHYLIP alignment file also extracted in that folder).

4. After a few seconds, IQ-TREE finishes and you may see something like this:

Congratulations ;-) You have finished the first IQ-TREE analysis.

2.5 Minimal command-line examples

A few typically analyses are listed in the following. Note that it is assumed that `iqtree` executable was already copied into system search path. If not, please replace `iqtree` with actual path to executable.

- Infer maximum-likelihood tree from a sequence alignment (`example.phy`) with the best-fit model automatically selected by ModelFinder:

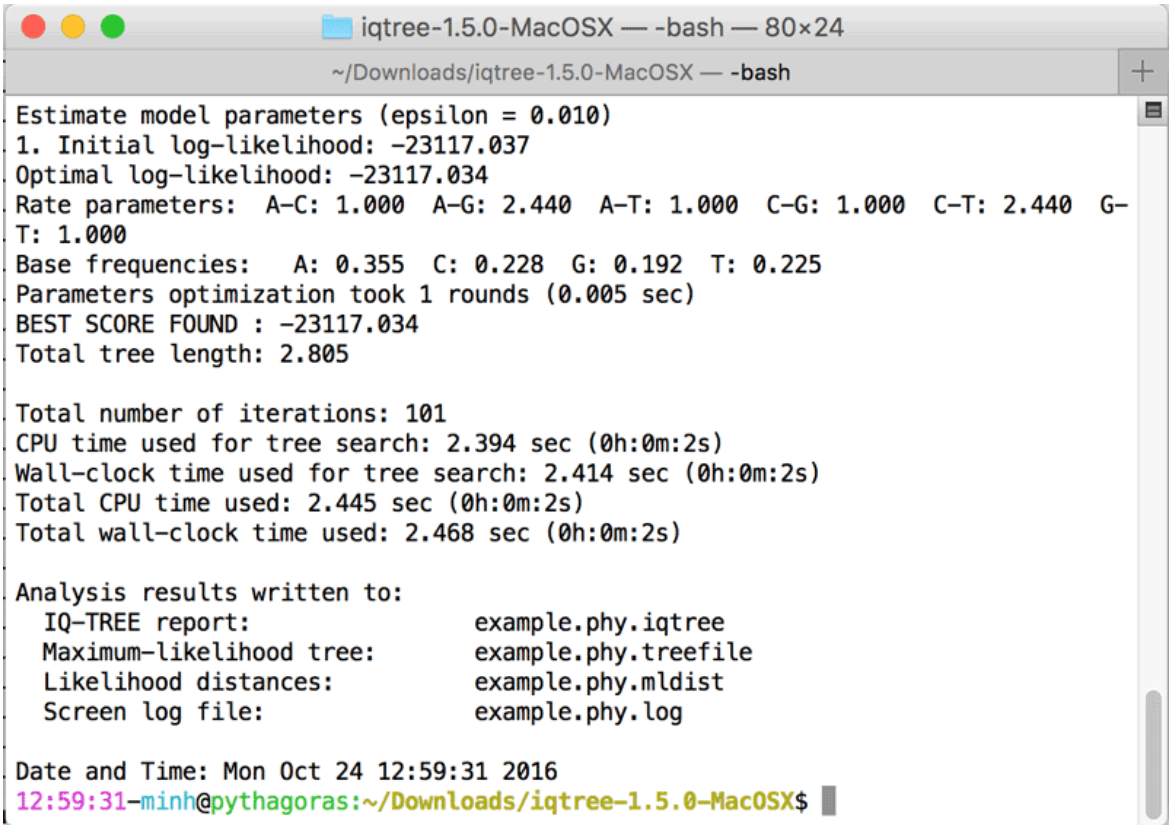


Figure 2.2: Mac terminal

```
iqtree -s example.phy
```

- Infer maximum-likelihood tree using `GTR+I+G` model:

```
iqtree -s example.phy -m GTR+I+G
```

- Perform ModelFinder without subsequent tree inference:

```
iqtree -s example.phy -m MF
```

- Combine ModelFinder, tree search, SH-aLRT test and ultrafast bootstrap with 1000 replicates:

```
iqtree -s example.phy -B 1000 -alrt 1000
# for version 1.x, change -B to -bb
```

- Perform edge-linked proportional partition model (`example.nex`):

```
iqtree -s example.phy -p example.nex
# for version 1.x change -p to -spp
```

- Find best partition scheme by possibly merging partitions:

```
iqtree -s example.phy -p example.nex -m MF+MERGE
```

- Find best partition scheme followed by tree inference and ultrafast bootstrap:

```
iqtree -s example.phy -p example.nex -m MFP+MERGE -B 1000
# for version 1.x change -B to -bb
```

- Use 4 CPU cores to speed up computation:

```
iqtree -s example.phy -T 4
# for version 1.x change -T to -nt
```

- Determine the best number of cores to use under `GTR+R4` model:

```
iqtree -s example.phy -m GTR+R4 -T AUTO
# for version 1.x change -T to -nt
```

- Show all available options:

```
iqtree -h
```

2.6 Where to go from here?

Please continue with the **Beginner's tutorial** for further usages.

Chapter 3

Web server tutorial

A quick starting guide for the IQ-TREE Web Server.

This tutorial explains briefly how to use the IQ-TREE web server for fast online phylogenetic inference, accessible at iqtree.cibiv.univie.ac.at.

There are three tabs: Tree Inference, Model Selection and Analysis Results.

3.1 Tree Inference

Tree Inference provides the most frequently used features of IQ-TREE and allows users to carry out phylogenetic analysis on a multiple sequence alignment (MSA). In the most basic case, no more than an MSA file is required to submit the job. Without further input, IQ-TREE will run with the default parameters and auto-detect the sequence type as well as the best-fitting substitution model. Additionally, Ultrafast Bootstrap (Hoang et al., 2018) and the SH-aLRT branch test (Guindon et al., 2010) will be performed.

You can either try out the web server with an example alignment by ticking the corresponding box or upload your own alignment file. By clicking on 'Browse' a dialog will open where you can select your MSA; the file formats Phylip, Fasta, Nexus, Clustal and MSF are supported.

After that you can submit the job. If you provide an email address, a notification will be sent to you once the job is finished. In case you don't specify an email address, you will receive a link in the next step; you can bookmark this link to retrieve your results after the job is finished.

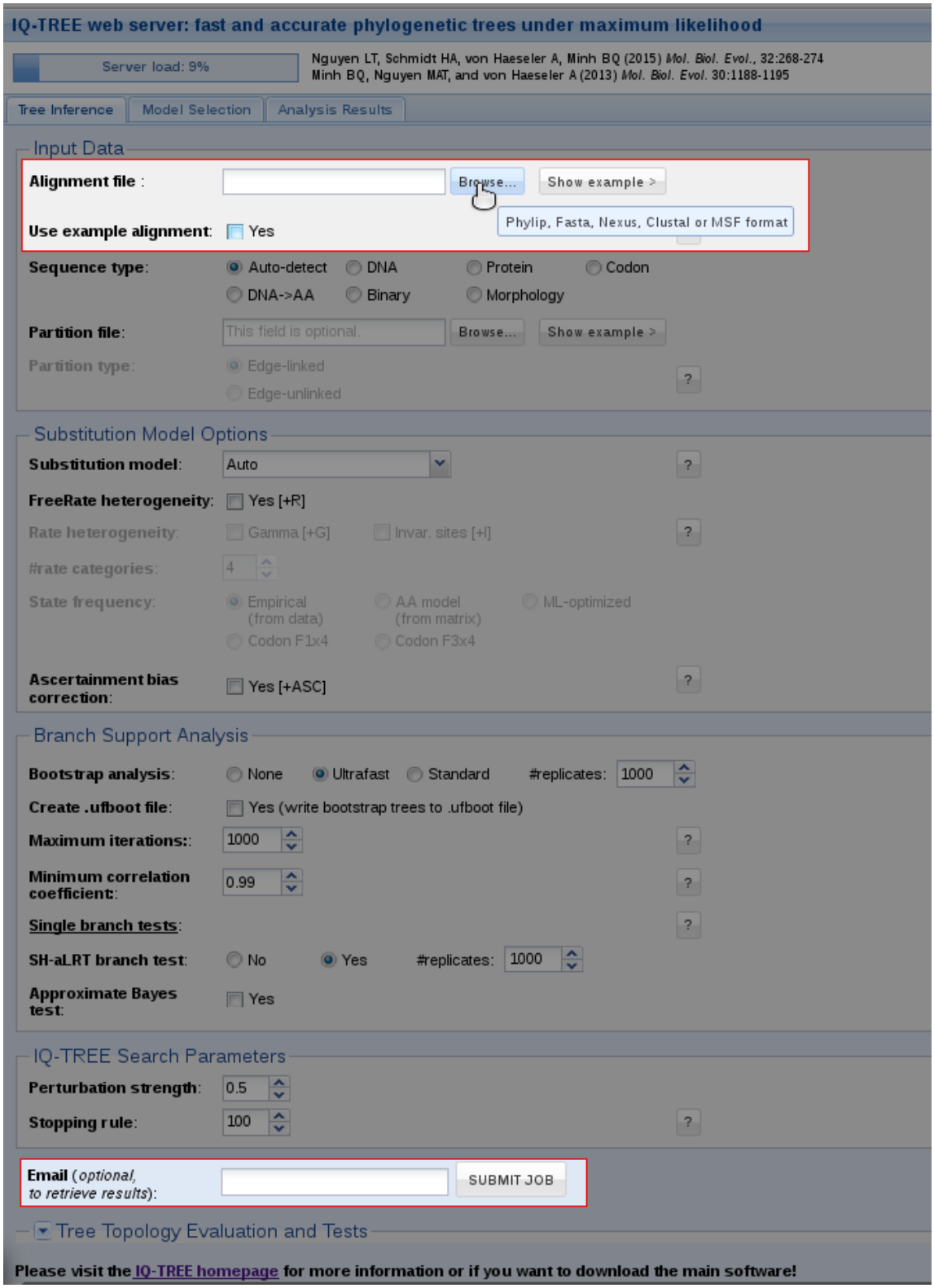


Figure 3.1: Tree Inference Tab

3.2 Model Selection

IQ-TREE supports a wide range of substitution models for DNA, protein, codon, binary and morphological alignments. In case you do not know which model is appropriate for your data, IQ-TREE can automatically determine the best-fit model for your alignment. Use the Model Selection tab if you only want to find the best-fit model without doing tree reconstruction.

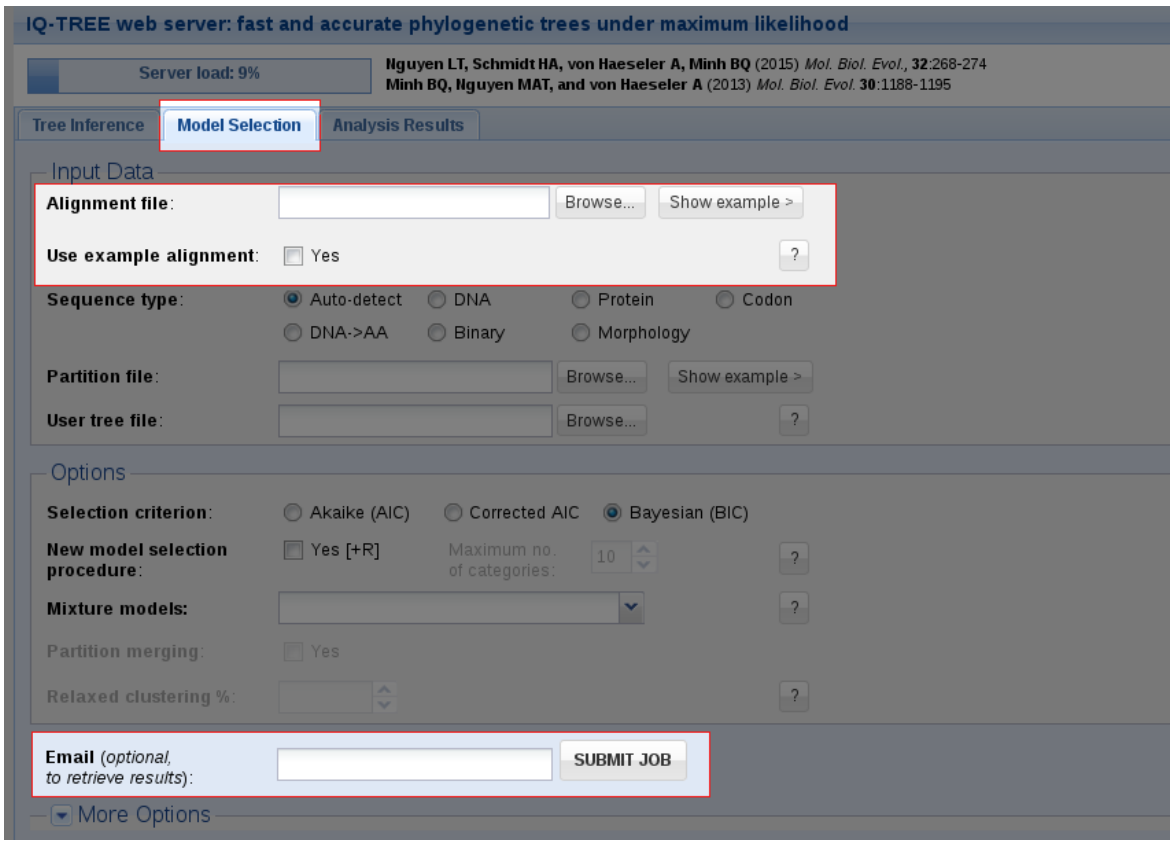


Figure 3.2: Model Selection Tab

Like with Tree Inference, the only obligatory input is a multiple sequence alignment. You can either upload your own **alignment file** or use the **example alignment** to try out the web server and then **submit the job**.

3.3 Analysis Results

In the tab Analysis Results you can monitor your jobs. With our example file, a run will only take a few seconds, depending on the server load. For your own alignments

the CPU time limit is 24 hours. If you provided an email address when submitting the job, you will get an email once it is finished.

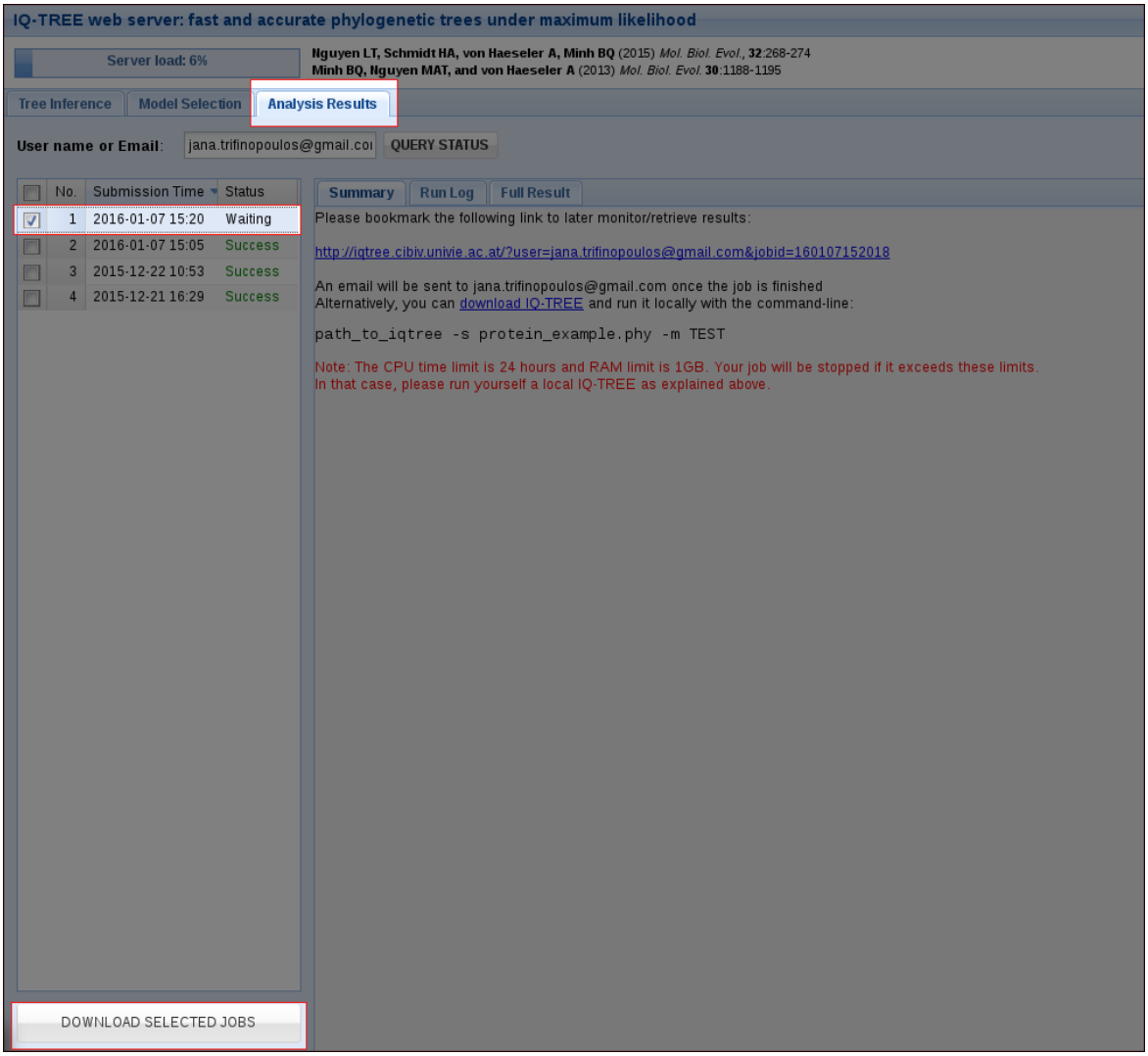


Figure 3.3: Analysis Results

Once a job is finished, you can select it by checking the corresponding box and then **download the selected jobs** as a zip file. This zip file will contain the results of your run, including the **Run Log** and the **Full Result** which are also accessible in the webserver.

Suffix	Explanation
`.iqtree`	Full result of the run, this is the main report file

Suffix	Explanation
`.log`	Run log
`.treefile`	Maximum likelihood tree in NEWICK format, can be visualized with treeviewer programs
`.svg`	Graphical tree representation in SVG format, done with ete view
`.pdf`	Graphical tree representation in PDF format, done with ete view
`.contree`	Consensus tree with assigned branch supports where branch lengths are optimized on the original alignment; printed if Ultrafast Bootstrap is selected
`.ckp.gz`	Checkpoint file; included if a job was stopped because of RAM/CPU limits

NOTE: Jobs which require more than 24 hours or 1GB RAM will be stopped. In such a case, you can download the stopped job and resume the run from the last checkpoint on your local PC as described here.

Chapter 4

Beginner's tutorial

This tutorial gives a beginner's guide.

Please first download and install the binary for your platform. For the next steps, the folder containing your `iqtree` executable should be added to your PATH enviroment variable so that IQ-TREE can be invoked by simply entering `iqtree` at the command-line. Alternatively, you can also copy `iqtree` binary into your system search.

TIP: For quick overview of all supported options in IQ-TREE, run the command `iqtree -h`.

4.1 Input data

IQ-TREE takes as input a *multiple sequence alignment* and will reconstruct an evolutionary tree that is best explained by the input data. If you have raw (unaligned) sequences, you need to first run an alignment program like MAFFT or ClustalW to align the sequences, before feeding them into IQ-TREE.

The input alignment can be in various common formats. For example the PHYLIP format which may look like:

```
7 28
Frog        AAATTTGGTCCTGTGATTCAGCAGTGAT
Turtle      CTTCCACACCCCAGGACTCAGCAGTGAT
Bird        CTACCACACCCCAGGACTCAGCAGTAAT
Human       CTACCACACCCCAGGAAACAGCAGTGAT
Cow         CTACCACACCCCAGGAAACAGCAGTGAC
Whale       CTACCACGCCCCAGGACACAGCAGTGAT
```

```
Mouse       CTACCACACCCCAGGACTCAGCAGTGAT
```

This tiny alignment contains 7 DNA sequences from several animals with the sequence length of 28 nucleotides. IQ-TREE also supports other file formats such as FASTA, NEXUS, CLUSTALW. The FASTA file for the above example may look like this:

```
>Frog
AAATTTGGTCCTGTGATTCAGCAGTGAT
>Turtle
CTTCCACACCCCAGGACTCAGCAGTGAT
>Bird
CTACCACACCCCAGGACTCAGCAGTAAT
>Human
CTACCACACCCCAGGAAACAGCAGTGAT
>Cow
CTACCACACCCCAGGAAACAGCAGTGAC
>Whale
CTACCACGCCCCAGGACACAGCAGTGAT
>Mouse
CTACCACACCCCAGGACTCAGCAGTGAT
```

TIP: From version 2 you can input a directory of alignment files. IQ-TREE 2 will load and concatenate all alignments within the directory, eliminating the need for users to manually perform this step.

Not all special characters are allowed in sequence names, because they may interfere with the structure encoding in the Newick tree files. To avoid problems with downstream software (like tree viewers), IQ-Tree (and also other phylogenetic software) checks the names for such potentially interfering characters and substitutes them by underscores `_`. Permitted characters in sequence names are alphanumeric letters, underscores `_`, dash `-`, dot `.`, slash `\` and vertical bar `|`. All other characters are substituted, like e.g. `hawk's-eye` is converted to `hawk_s-eye` as which it will appear in the tree.

Please note, this can lead to duplicate names if you, for instance, already have two sequences named `hawk_s-eye` and `hawk's-eye`. In such cases you will obtain an error and you need to adjust the names in the original input alignment.

4.2 First running example

From the download there is an example alignment called `example.phy` in PHYLIP format. This example contains parts of the mitochondrial DNA sequences of several

animals (Source: Phylogenetic Handbook).

You can now start to reconstruct a maximum-likelihood tree from this alignment by entering (assuming that you are now in the same folder with `example.phy`):

```
iqtree -s example.phy
```

`-s` is the option to specify the name of the alignment file that is always required by IQ-TREE to work. At the end of the run IQ-TREE will write several output files including:

- `example.phy.iqtree`: the main report file that is self-readable. You should look at this file to see the computational results. It also contains a textual representation of the final tree (see below).
- `example.phy.treefile`: the ML tree in NEWICK format, which can be visualized by any supported tree viewer programs like FigTree or iTOL.
- `example.phy.log`: log file of the entire run (also printed on the screen). To report bugs, please send this log file and the original alignment file to the authors.

For this example data the resulting maximum-likelihood tree may look like this (extracted from `.iqtree` file):

```
NOTE: Tree is UNROOTED although outgroup taxon 'LngfishAu' is
   drawn at root

+--------------LngfishAu
|
|        +--------------LngfishSA
+--------|
|        +--------------LngfishAf
|
|      +--------------------Frog
+------|
       |                +-----------------Turtle
       |          +-----|
       |          |     |      +------------------------Sphenodon
       |          |     |   +--|
       |          |     |   |  +--------------------------Lizard
       |          |     +---|
       |          |         |      +---------------------Crocodile
       |          |         +------|
       |          |                +------------------Bird
       |          +---------|
       |                    |                  +----------------Human
```

```
                  |                +--|
                  |                |  |  +--------Seal
                  |                |  +--|
                  |                |     |   +-------Cow
                  |                |     +---|
                  |                |         +---------Whale
                  |           +----|
                  |           |    |         +------Mouse
                  |           |    +---------|
                  |           |              +--------Rat
                  |           +----------|
                              |   +----------------Platypus
                              +---|
                                  +-------------Opossum
```

This makes sense as the mammals (`Human` to `Opossum`) form a clade, whereas the reptiles (`Turtle` to `Crocodile`) and `Bird` form a separate sister clade. Here the tree is drawn at the *outgroup* Lungfish which is more accient than other species in this example. However, please note that IQ-TREE always produces an **unrooted tree** as it knows nothing about this biological background; IQ-TREE simply draws the tree this way as `LngfishAu` is the first sequence occuring in the alignment.

During the example run above, IQ-TREE periodically wrote to disk a checkpoint file `example.phy.ckp.gz` (gzip-compressed to save space). This checkpoint file is used to resume an interrupted run, which is handy if you have a very large data sets or time limit on a cluster system. If the run did not finish, invoking IQ-TREE again with the very same command line will recover the analysis from the last stopped point, thus saving all computation time done before.

If the run successfully completed, running again will issue an error message:

```
ERROR: Checkpoint (example.phy.ckp.gz) indicates that a previous
   run successfully finished
Use `-redo` option if you really want to redo the analysis and
   overwrite all output files.
```

This prevents lost of data if you accidentally re-run IQ-TREE. However, if you really want to re-run the analysis and overwrite all previous output files, use `-redo` option:

```
iqtree -s example.phy -redo
```

Finally, the default prefix of all output files is the alignment file name. You can change the prefix with:

```
iqtree -s example.phy --prefix myprefix
# for version 1.x change --prefix to -pre
```

This prevents output files being overwritten when you perform multiple analyses on the same alignment within the same folder.

4.3 Choosing the right substitution model

NOTE: If you use model selection please cite the following paper:

> **S. Kalyaanamoorthy, B.Q. Minh, T.K.F. Wong, A. von Haeseler, and L.S. Jermiin** (2017) ModelFinder: fast model selection for accurate phylogenetic estimates. *Nat. Methods*, 14:587–589. DOI: 10.1038/nmeth.4285

IQ-TREE supports a wide range of substitution models for DNA, protein, codon, binary and morphological alignments. If you do not know which model is appropriate for your data, you can use ModelFinder to determine the best-fit model:

```
iqtree -s example.phy -m MFP
# change -m MFP to -m TEST to resemble jModelTest/ProtTest
```

`-m` is the option to specify the model name to use during the analysis. The special `MFP` key word stands for *ModelFinder Plus*, which tells IQ-TREE to perform ModelFinder and the remaining analysis using the selected model. ModelFinder computes the log-likelihoods of an initial parsimony tree for many different models and the *Akaike information criterion* (AIC), *corrected Akaike information criterion* (AICc), and the *Bayesian information criterion* (BIC). Then ModelFinder chooses the model that minimizes the BIC score (you can also change to AIC or AICc by adding the option `-AIC` or `-AICc`, respectively).

> **TIP**: Starting with version 1.5.4, `-m MFP` is the default behavior. Thus, this run is equivalent to `iqtree -s example.phy`.

Here, IQ-TREE will write an additional file:

- `example.phy.model`: log-likelihoods for all models tested. It serves as a checkpoint file to recover an interrupted model selection.

If you now look at `example.phy.iqtree` you will see that IQ-TREE selected `TIM2+I+G4` as the best-fit model for this example data. Thus, for additional analyses you do not

have to perform the model test again and can use the selected model:

```
iqtree -s example.phy -m TIM2+I+G
```

Sometimes you only want to find the best-fit model without doing tree reconstruction, then run:

```
iqtree -s example.phy -m MF
# change -m MF to -m TESTONLY to resemble jModelTest/ProtTest
```

By default, the maximum number of categories is limitted to 10 due to computational reasons. If your sequence alignment is long enough, then you can increase this upper limit with the `cmax` option:

```
iqtree -s example.phy -m MF -cmax 15
```

will test `+R2` to `+R15` instead of at most `+R10`.

To reduce computational burden, one can use the option `-mset` to restrict the testing procedure to a subset of base models instead of testing the entire set of all available models. For example, `-mset WAG,LG` will test only models like `WAG+...` or `LG+...`. Another useful option in this respect is `-msub` for AA data sets. With `-msub nuclear` only general AA models are included, whereas with `-msub viral` only AA models for viruses are included.

If you have enough computational resource, you can perform a thorough and more accurate analysis that invokes a full tree search for each model considered via the `-mtree` option:

```
iqtree -s example.phy -m MF -mtree
```

4.4 Using codon models

IQ-TREE supports a number of codon models. You need to input a protein-coding DNA alignment and specify codon data by option `-st CODON` (Otherwise, IQ-TREE applies DNA model because it detects that your alignment has DNA sequences):

```
iqtree -s coding_gene.phy -st CODON
```

If your alignment length is not divisible by 3, IQ-TREE will stop with an error message. IQ-TREE will group sites 1,2,3 into codon site 1; sites 4,5,6 to codon site 2; etc. Moreover, any codon, which has at least one gap/unknown/ambiguous nucleotide, will be treated as unknown codon character.

Note that the above command assumes the standard genetic code. If your sequences follow 'The Invertebrate Mitochondrial Code' (see the full list of supported genetic code here), then run:

```
iqtree -s coding_gene.phy -st CODON5
```

4.5 Binary, morphological and SNP data

Binary alignments contain sequences with characters 0 and 1, which can be in any common formats supported by IQ-TREE, for example, in PHYLIP format:

```
4 6
S1    010101
S2    110011
S3    0--100
S4    10--10
```

Morphological alignments have an extended characeter alphabet of 0-9 and A-Z (for states 10-31). For example (PHYLIP format):

```
4 10
S1    0123401234
S2    03---20432
S3    3202-04--0
S4    4230120340
```

IQ-TREE will automatically determine the sequence type and the alphabet size. To run IQ-TREE on such alignments:

```
iqtree -s morphology.phy
```

or

```
iqtree -s morphology.phy -st MORPH
```

IQ-TREE implements to two morphological ML models: MK and ORDERED. Morphological data typically do not have constant (uninformative) sites. In such cases, you should apply ascertainment bias correction model by e.g.:

```
iqtree -s morphology.phy -st MORPH -m MK+ASC
```

You can again select the best-fit binary/morphological model:

```
iqtree -s morphology.phy -st MORPH
```

For SNP data (DNA) that typically do not contain constant sites, you can explicitly tell the model to include ascertainment bias correction:

```
iqtree -s SNP_data.phy -m GTR+ASC
```

You can explicitly tell model testing to only include `+ASC` model with:

```
iqtree -s SNP_data.phy -m MFP+ASC
```

4.6 Assessing branch supports with ultrafast bootstrap approximation

To overcome the computational burden required by the nonparametric bootstrap, IQ-TREE introduces an ultrafast bootstrap approximation (UFBoot) (Minh et al., 2013; Hoang et al., 2018) that is orders of magnitude faster than the standard procedure and provides relatively unbiased branch support values. Citation for UFBoot:

> **D.T. Hoang, O. Chernomor, A. von Haeseler, B.Q. Minh, and L.S. Vinh** (2018) UFBoot2: Improving the ultrafast bootstrap approximation. *Mol. Biol. Evol.*, 35:518–522. https://doi.org/10.1093/molbev/msx281

To run UFBoot:

```
iqtree -s example.phy -m TIM2+I+G -B 1000
# for version 1.x change -B to -bb
```

`-B` specifies the number of bootstrap replicates where 1000 is the minimum number recommended. The section `MAXIMUM LIKELIHOOD TREE` in `example.phy.iqtree` shows a textual representation of the maximum likelihood tree with branch support values in percentage. The NEWICK format of the tree is printed to the file `example.phy.treefile`. In addition, IQ-TREE writes the following files:

- `example.phy.contree`: the consensus tree with assigned branch supports where branch lengths are optimized on the original alignment.
- `example.phy.splits.nex`: support values in percentage for all splits (bipartitions), computed as the occurence frequencies in the bootstrap trees. This file can be viewed with the program SplitsTree to explore the conflicting signals in

the data. So it is more informative than consensus tree, e.g. you can see how highly supported the second best conflicting split is, which had no chance to enter the consensus tree.

- `example.phy.splits` (if using `-wsplits` option): This file contains the same information as `example.phy.splits.nex` but in star-dot format.

NOTE: UFBoot support values have a different interpretation to the standard bootstrap. Refer to FAQ: UFBoot support values interpretation for more information.

4.7 Reducing impact of severe model violations with UFBoot

Starting with IQ-TREE version 1.6 we provide a new option `-bnni` to reduce the risk of overestimating branch supports with UFBoot due to severe model violations. With this option UFBoot will further optimize each bootstrap tree using a hill-climbing nearest neighbor interchange (NNI) search based directly on the corresponding bootstrap alignment.

Thus, if severe model violations are present in the data set at hand, users are advised to append `-bnni` to the regular UFBoot command:

```
iqtree -s example.phy -m TIM2+I+G -B 1000 -bnni
# for version 1.x change -B to -bb
```

4.8 Assessing branch supports with standard nonparametric bootstrap

The standard nonparametric bootstrap is invoked by the `-b` option:

```
iqtree -s example.phy -m TIM2+I+G -b 100
```

`-b` specifies the number of bootstrap replicates where 100 is the minimum recommended number. The output files are similar to those produced by the UFBoot procedure.

4.9 Assessing branch supports with single branch tests

IQ-TREE provides an implementation of the SH-like approximate likelihood ratio test (Guindon et al., 2010). To perform this test, run:

```
iqtree -s example.phy -m TIM2+I+G -alrt 1000
```

`-alrt` specifies the number of bootstrap replicates for SH-aLRT where 1000 is the minimum number recommended.

IQ-TREE also supports other tests such as the aBayes test (Anisimova et al., 2011) and the local bootstrap test (Adachi and Hasegawa, 1996b). See single branch tests for more details.

You can also perform both SH-aLRT and the ultrafast bootstrap within one single run:

```
iqtree -s example.phy -m TIM2+I+G -alrt 1000 -B 1000
# for version 1.x change -B to -bb
```

The branches of the resulting `.treefile` will be assigned with both SH-aLRT and UFBoot support values, which are readable by any tree viewer program like FigTree, Dendroscope or ETE. You can also look at the textual tree figure in `.iqtree` file:

```
NOTE: Tree is UNROOTED although outgroup taxon 'LngfishAu' is
   drawn at root
Numbers in parentheses are SH-aLRT support (%) / ultrafast
   bootstrap support (%)

+-------------LngfishAu
|
|       +--------------LngfishSA
+-------| (100/100)
|       +------------LngfishAf
|
|      +---------------------Frog
+------| (99.8/100)
       |                      +-----------------Turtle
       |                   +--| (85/72)
       |                   |  |    +------------------------
          Crocodile
       |                   |  +----| (96.5/97)
       |                   |       +------------------Bird
```

```
       |               +--| (39/51)
       |               |  +----------------------------Sphenodon
       |         +-----| (98.2/99)
       |         |     +--------------------------------Lizard
       +---------| (100/100)
                 |                     +--------------Human
                 |                  +--| (92.3/93)
                 |                  |  |  +------Seal
                 |                  |  +--| (68.3/75)
                 |                  |     |  +-----Cow
                 |                  |     +--| (99.7/100)
                 |                  |        +-------Whale
                 |            +-----| (99.1/100)
                 |            |     |         +---Mouse
                 |            |     +---------| (100/100)
                 |            |               +------Rat
                 +------------| (100/100)
                              |  +--------------Platypus
                              +--| (93/98)
                                 +-----------Opossum
```

From this figure, the branching patterns within reptiles are poorly supported (e.g. `Sphenodon` with SH-aLRT: 39%, UFBoot: 51% and `Turtle` with SH-aLRT: 85%, UFBoot: 72%) as well as the phylogenetic position of `Seal` within mammals (SH-aLRT: 68.3%, UFBoot: 75%). Other branches appear to be well supported.

4.10 Utilizing multi-core CPUs

IQ-TREE can utilize multiple CPU cores to speed up the analysis. A complement option `-T` (or `-nt` for version 1.x) allows specifying the number of CPU cores to use. For example:

```
iqtree -s example.phy -m TIM2+I+G -T 2
# for version 1.x change -T to -nt
```

Here, IQ-TREE will use 2 CPU cores to perform the analysis.

Note that the parallel efficiency is only good for long alignments. A good practice is to use `-T AUTO` to determine the best number of cores:

```
iqtree -s example.phy -m TIM2+I+G -T AUTO
# for version 1.x change -T to -nt
```

Then while running IQ-TREE may print something like this on to the screen:

```
Measuring multi-threading efficiency up to 8 CPU cores
Threads: 1 / Time: 8.001 sec / Speedup: 1.000 / Efficiency: 100%
   / LogL: -22217
Threads: 2 / Time: 4.346 sec / Speedup: 1.841 / Efficiency: 92% /
    LogL: -22217
Threads: 3 / Time: 3.381 sec / Speedup: 2.367 / Efficiency: 79% /
    LogL: -22217
Threads: 4 / Time: 4.385 sec / Speedup: 1.825 / Efficiency: 46% /
    LogL: -22217
BEST NUMBER OF THREADS: 3
```

Therefore, I would only use 3 cores for this example data. For later analysis with your same data set, you can stick to the determined number.

Depending on the compute system it might be required to set an upper limit of CPU cores that can automatically be assigned. Use the `-ntmax` option to do so. For instance

```
iqtree -s example.phy -m TIM2+I+G -T AUTO -ntmax 8
# for version 1.x change -T to -nt
```

does the same as above, but only allows to use up to 8 CPU cores. By default all cores of the current machine would be used as maximum.

4.11 Where to go from here?

Once confident enough you can go on with a **more advanced tutorial**, which covers topics like phylogenomic (multi-gene) analyses using partition models or mixture models.

Chapter 5

Advanced tutorial

Recommended for experienced users to explore more features.

To get started, please read the Beginner's Tutorial first if not done so yet.

5.1 Partitioned analysis for multi-gene alignments

If you used partition model in a publication please cite:

> **O. Chernomor, A. von Haeseler, and B.Q. Minh** (2016) Terrace aware data structure for phylogenomic inference from supermatrices. *Syst. Biol.*, 65:997-1008. https://doi.org/10.1093/sysbio/syw037

In the partition model, you can specify a substitution model for each gene/character set. IQ-TREE will then estimate the model parameters separately for every partition. Moreover, IQ-TREE provides edge-linked or edge-unlinked branch lengths between partitions:

- `-q partition_file`: all partitions share the same set of branch lengths (like `-q` option of RAxML).
- `-p partition_file` (`-spp` in version 1.x): like above but allowing each partition to have its own evolution rate.
- `-Q partition_file` (`-sp` in version 1.x): each partition has its own set of branch lengths (like combination of `-q` and `-M` options in RAxML) to account for, e.g. *heterotachy* (Lopez et al., 2002).

NOTE: `-p` is recommended for typical analysis. `-q` is unrealistic and `-Q` is very parameter-rich. One can also perform all three analyses and compare e.g. the BIC scores to determine the best-fit partition model.

IQ-TREE supports RAxML-style and NEXUS partition input file. The RAxML-style partition file may look like:

```
DNA, part1 = 1-100
DNA, part2 = 101-384
```

If your partition file is called `example.partitions`, the partition analysis can be run with:

```
iqtree -s example.phy -p example.partitions -m GTR+I+G
# for version 1.x change -p to -spp
```

Note that using RAxML-style partition file, all partitions will use the same rate heterogeneity model given in `-m` option (`+I+G` in this example). If you want to specify, say, `+G` for the first partition and `+I+G` for the second partition, then you need to create the more flexible NEXUS partition file. This file contains a `SETS` block with `CharSet` and `CharPartition` commands to specify individual genes and the partition, respectively. For example:

```
#nexus
begin sets;
    charset part1 = 1-100;
    charset part2 = 101-384;
    charpartition mine = HKY+G:part1, GTR+I+G:part2;
end;
```

If your NEXUS file is called `example.nex`, then you can use the option `-p` to input the file as following:

```
iqtree -s example.phy -p example.nex
# for version 1.x change -p to -spp
```

Here, IQ-TREE partitions the alignment `example.phy` into 2 sub-alignments named `part1` and `part2` containing sites (columns) 1-100 and 101-384, respectively. Moreover, IQ-TREE applies the subtitution models `HKY+G` and `GTR+I+G` to `part1` and `part2`, respectively. Substitution model parameters and trees with branch lengths can be found in the result file `example.nex.iqtree`.

Moreover, the `CharSet` command allows to specify non-consecutive sites with e.g.:

```
charset part1 = 1-100 200-384;
```

That means, `part1` contains sites 1-100 and 200-384 of the alignment. Another example is:

```
charset part1 = 1-100\3;
```

for extracting sites 1,4,7,...,100 from the alignment. This is useful for getting codon positions from the protein-coding alignment.

5.2 Partitioned analysis with mixed data

IQ-TREE also allows combining sub-alignments from different alignment files, which is helpful if you want to combine mixed data (e.g. DNA and protein) in a single analysis. Here is an example for mixing DNA, protein and codon models:

```
#nexus
begin sets;
    charset part1 = dna.phy: 1-100 201-300;
    charset part2 = dna.phy: 101-200;
    charset part3 = prot.phy: 1-400;
    charset part4 = prot.phy: 401-600;
    charset part5 = codon.phy:CODON, *;
    charpartition mine = HKY:part1, GTR+G:part2, LG+G:part3, WAG+
       I+G:part4, GY:part5;
end;
```

Here, `part1` and `part2` contain sub-alignments from alignment file `dna.phy`, whereas `part3` and `part4` are loaded from alignment file `prot.phy` and `part5` from `codon.phy`. The `:` is needed to separate the alignment file name and site specification. Note that, for convenience `*` in `part5` specification means that `part5` corresponds to the entire alignment `codon.phy`.

TIP: For `part5` the `CODON` keyword is specified so that IQ-TREE will apply a codon model. Moreover, this implicitly assumes the standard genetic code. If you want to use another genetic code, append `CODON` with the code ID described here

Because the alignment file names are now specified in this NEXUS file, you can omit the `-s` option:

```
iqtree -p example.nex
# for version 1.x change -p to -spp
```

Note that `aln.phy` and `prot.phy` does not need to contain the same set of sequences. For instance, if some sequence occurs in `aln.phy` but not in `prot.phy`, IQ-TREE will treat the corresponding parts of sequence in `prot.phy` as missing data. For your convenience IQ-TREE writes the concatenated alignment into the file `example.nex.conaln`.

5.3 Choosing the right partitioning scheme

ModelFinder implements a greedy strategy (Lanfear et al., 2012) that starts with the full partition model and subsequentially merges two genes until the model fit does not increase any further:

```
iqtree -s example.phy -p example.nex -m MFP+MERGE
# for version 1.x change -p to -spp
```

Note that this command considers the FreeRate heterogeneity model (see model selection tutorial). If you want to resemble PartitionFinder by just considering the invariable site and Gamma rate heterogeneity (thus saving computation times), then run:

```
iqtree -s example.phy -p example.nex -m TESTMERGE
# for version 1.x change -p to -spp
```

After ModelFinder found the best partition, IQ-TREE will immediately start the tree reconstruction under the best-fit partition model. Sometimes you only want to find the best-fit partition model without doing tree reconstruction, then run:

```
iqtree -s example.phy -p example.nex -m MF+MERGE
# for version 1.x change -p to -spp
```

To resemble PartitionFinder and save time:

```
iqtree -s example.phy -p example.nex -m TESTMERGEONLY
# for version 1.x change -p to -spp
```

To reduce the computational burden IQ-TREE implements the *relaxed hierarchical clustering algorithm* (Lanfear et al., 2014), which is invoked via `-rcluster` option:

```
iqtree -s example.phy -p example.nex -m MF+MERGE -rcluster 10
# for version 1.x change -p to -spp
```

to only examine the top 10% partition merging schemes (similar to the `--rcluster-percent 10` option in PartitionFinder).

5.4 Ultrafast bootstrapping with partition model

IQ-TREE can perform the ultrafast bootstrap with partition models by e.g.,

```
iqtree -s example.phy -p example.nex -B 1000
# for version 1.x change -p to -spp and -B to -bb
```

Here, IQ-TREE will resample the sites *within* partitions (i.e., the bootstrap replicates are generated per partition separately and then concatenated together). The same holds true if you do the standard nonparametric bootstrap.

IQ-TREE supports the partition-resampling strategy as suggested by (Nei et al., 2001):

```
iqtree -s example.phy -p example.nex -B 1000 --sampling GENE
# for version 1.x change -p to -spp and -B to -bb and --sampling
    to -bsam
```

to resample partitions instead of sites. Moreover, IQ-TREE allows an even more complicated strategy: resampling partitions and then sites within resampled partitions (Gadagkar et al., 2005; Seo et al., 2005). This may help to reduce false positives (i.e. wrong branch receiving 100% support):

```
iqtree -s example.phy -p example.nex -B 1000 --sampling GENESITE
# for version 1.x change -p to -spp and -B to -bb and --sampling
    to -bsam
```

5.5 Constrained tree search

IQ-TREE supports constrained tree search via `-g` option, so that the resulting tree must obey a constraint tree topology. The constraint tree can be multifurcating and need not to contain all species. To illustrate, let's return to the first running example, where we want to force Human grouping with Seal whereas Cow with Whale. If you use the following constraint tree (NEWICK format):

```
((Human,Seal),(Cow,Whale));
```

Save this to a file `example.constr0` and run:

```
iqtree -s example.phy -m TIM2+I+G -g example.constr0 --prefix
   example.constr0
# for version 1.x change --prefix to -pre
```

(We use a prefix in order not to overwrite output files of the previous run). The resulting part of the tree extracted from `example.constr0.iqtree` looks exactly like a normal unconstrained tree search:

```
        +--------------Human
     +--|
     |  |  +------Seal
     |  +--|
     |     |  +-----Cow
     |     +--|
     |        +-------Whale
+----|
|    |         +---Mouse
|    +---------|
|              +------Rat
```

This is the correct behavior: although Human and Seal are not monophyletic, this tree indeed satisfies the constraint, because the induced subtree separates (Human,Seal) from (Cow,Whale). This comes from the fact that the tree is *unrooted*. If you want them to be sister groups, then you need to include *outgroup* taxa into the constraint tree. For example:

```
((Human,Seal),(Cow,Whale),Mouse);
```

Save this to `example.constr1` and run:

```
iqtree -s example.phy -m TIM2+I+G -g example.constr1 --prefix
   example.constr1
# for version 1.x change --prefix to -pre
```

The resulting part of the tree is then:

```
           +---------------Human
        +--|
        |  +------Seal
     +--|
     |  |  +-----Cow
     |  +--|
```

```
      |     +-------Whale
+-----|
|     |          +---Mouse
|     +----------|
|                +------Rat
```

which shows the desired effect.

NOTE: While this option helps to enforce the tree based on prior knowledge, it is advised to always perform tree topology tests to make sure that the resulting constrained tree is NOT significantly worse than an unconstrained tree! See tree topology tests and testing constrained tree below for a guide how to check this.

5.6 Tree topology tests

IQ-TREE can compute log-likelihoods of a set of trees passed via the `-z` option:

```
iqtree -s example.phy -z example.treels -m GTR+G
```

assuming that `example.treels` is an existing file containing a set of trees in NEWICK format. IQ-TREE first reconstructs an ML tree. Then, it will compute the log-likelihood of the trees in `example.treels` based on the estimated parameters done for the ML tree. `example.phy.iqtree` will have a section called `USER TREES` that lists the tree IDs and the corresponding log-likelihoods. The trees with optimized branch lengths can be found in `example.phy.treels.trees` If you only want to evaluate the trees without reconstructing the ML tree, you can run:

```
iqtree -s example.phy -z example.treels -n 0
```

Here, the number of search iterations is set to 0 (`-n 0`), such that model parameters are quickly estimated from an initial parsimony tree, which is normally accurate enough for our purpose. If you, however, prefer to estimate model parameters based on a tree (e.g. reconstructed previously), use `-te <treefile>` option.

IQ-TREE also supports several tree topology tests using the RELL approximation (Kishino et al., 1990). This includes bootstrap proportion (BP), Kishino-Hasegawa test (Kishino and Hasegawa, 1989), Shimodaira-Hasegawa test (Shimodaira and Hasegawa, 1999), expected likelihood weights (Strimmer and Rambaut, 2002):

```
iqtree -s example.phy -z example.treels -n 0 -zb 1000
```

Here, `-zb` specifies the number of RELL replicates, where 1000 is the recommended minimum number. The `USER TREES` section of `example.phy.iqtree` will list the results of BP, KH, SH, and ELW methods.

If you also want to perform the weighted KH and weighted SH tests, simply add `-zw` option:

```
iqtree -s example.phy -z example.treels -n 0 -zb 1000 -zw
```

Starting with version 1.4.0 IQ-TREE supports approximately unbiased (AU) test (Shimodaira, 2002) via `-au` option:

```
iqtree -s example.phy -z example.treels -n 0 -zb 1000 -zw -au
```

This will perform all above tests plus the AU test.

Finally, note that IQ-TREE will automatically detect duplicated tree topologies and omit them during the evaluation.

HINTS:

- The KH, SH and AU tests return p-values, thus a tree is rejected if its p-value < 0.05 (marked with a `-` sign).
- bp-RELL and c-ELW return posterior weights which *are not p-value*. The weights sum up to 1 across the trees tested.
- The KH test (Kishino and Hasegawa, 1989) was designed to test 2 trees and thus has no correction for multiple testing. The SH test (Shimodaira and Hasegawa, 1999) fixes this problem.
- However, the SH test becomes too conservative (i.e., rejecting fewer trees than expected) when testing many trees. The AU test (Shimodaira, 2002) fixes this problem and is thus recommended as replacement for both KH and SH tests.

5.7 Testing constrained tree

We now illustrate an example to use the AU test (see above) to test trees from unconstrained versus constrained search, which is helpful to know if a constrained search is sensible or not. Thus:

1. Perform an unconstrained search:

```
iqtree -s example.phy -m TIM2+I+G --prefix example.unconstr
# for version 1.x change --prefix to -pre
```

2. Perform a constrained search, where `example.constr1` file contains: `((Human,Seal),(Cow,Whale),Mouse);`:

```
iqtree -s example.phy -m TIM2+I+G -g example.constr1 --
   prefix example.constr1
# for version 1.x change --prefix to -pre
```

3. Perform another constrained search, where `example.constr2` file contains `((Human,Cow,Whale),Seal,Mouse);`:

```
iqtree -s example.phy -m TIM2+I+G -g example.constr2 --
   prefix example.constr2
# for version 1.x change --prefix to -pre
```

4. Perform the last constrained search, where `example.constr3` file contains `((Human,Mouse),(Cow,Rat),Opossum);`:

```
iqtree -s example.phy -m TIM2+I+G -g example.constr3 --
   prefix example.constr3
# for version 1.x change --prefix to -pre
```

5. Concatenate all trees into a file:

```
# for Linux or macOS
cat example.unconstr.treefile example.constr1.treefile
   example.constr2.treefile example.constr3.treefile >
   example.treels

# for Windows
type example.unconstr.treefile example.constr1.treefile
   example.constr2.treefile example.constr3.treefile >
   example.treels
```

6. Test the set of trees:

```
iqtree -s example.phy -m TIM2+I+G -z example.treels -n 0 -
   zb 1000 -au
```

Now look at the resulting `.iqtree` file:

```
USER TREES
----------

See example.phy.trees for trees with branch lengths.
```

```
Tree      logL    deltaL  bp-RELL    p-KH     p-SH    c-ELW     p
   -AU
--------------------------------------------------------------------------

  1   -21152.617   0.000  0.7110 + 0.7400 + 1.0000 + 0.6954 +
     0.7939 +
  2   -21156.802   4.185  0.2220 + 0.2600 + 0.5910 + 0.2288 +
     0.3079 +
  3   -21158.579   5.962  0.0670 + 0.1330 + 0.5130 + 0.0758 +
     0.1452 +
  4   -21339.596 186.980  0.0000 - 0.0000 - 0.0000 - 0.0000 -
     0.0000 -

deltaL  : logL difference from the maximal logl in the set.
bp-RELL : bootstrap proportion using RELL method (Kishino et al.
   1990).
p-KH    : p-value of one sided Kishino-Hasegawa test (1989).
p-SH    : p-value of Shimodaira-Hasegawa test (2000).
c-ELW   : Expected Likelihood Weight (Strimmer & Rambaut 2002).
p-AU    : p-value of approximately unbiased (AU) test (Shimodaira
   , 2002).

Plus signs denote the 95% confidence sets.
Minus signs denote significant exclusion.
All tests performed 1000 resamplings using the RELL method.
```

One sees that the AU test does not reject the first 3 trees (denoted by `+` sign below the `p-AU` column), whereas the last tree is significantly excluded (`-` sign). All other tests also agree with this. Therefore, groupings of `(Human,Mouse)` and `(Cow,Rat)` do not make sense. Whereas the phylogenetic position of `Seal` based on 3 first trees is still undecidable. This is in agreement with the SH-aLRT and ultrafast bootstrap supports done in the Tutorial.

5.8 Consensus construction and bootstrap value assignment

IQ-TREE can construct an extended majority-rule consensus tree from a set of trees written in NEWICK or NEXUS format (e.g., produced by MrBayes):

```
iqtree -con mytrees
```

To build a majority-rule consensus tree, simply set the minimum support threshold to 0.5:

```
iqtree -con mytrees -minsup 0.5
```

If you want to specify a burn-in (the number of beginning trees to ignore from the trees file), use `-bi` option:

```
iqtree -con mytrees -minsup 0.5 -bi 100
```

to skip the first 100 trees in the file.

IQ-TREE can also compute a consensus network and print it into a NEXUS file by:

```
iqtree -net mytrees
```

Finally, a useful feature is to read in an input tree and a set of trees, then IQ-TREE can assign the support value onto the input tree (number of times each branch in the input tree occurs in the set of trees). This option is useful if you want to compute the support values for an ML tree based on alternative topologies.

```
iqtree -sup input_tree set_of_trees
```

5.9 User-defined substitution models

Users can specify any DNA model using a 6-letter code that defines which rates should be equal. For example, `010010` corresponds to the HKY model and `012345` to the GTR model. In fact, IQ-TREE uses this specification internally to simplify the coding. The 6-letter code is specified via the `-m` option, e.g.:

```
iqtree -s example.phy -m 010010+G
```

Moreover, with the `-m` option one can input a file which contains the 6 rates (A-C, A-G, A-T, C-G, C-T, G-T) and 4 base frequencies (A, C, G, T). For example:

```
iqtree -s example.phy -m mymodel+G
```

where `mymodel` is a file containing the 10 entries described above, in the correct order. The entries can be seperated by either empty space(s) or newline character. One can even specify the rates within `-m` option by e.g.:

```
iqtree -s example.phy -m 'TN{2.0,3.0}+G8{0.5}+I{0.15}'
```

That means, we use Tamura-Nei model with fixed transition-transversion rate ratio of 2.0 and purine/pyrimidine rate ratio of 3.0. Moreover, we use 8-category Gamma-distributed site rates with the shape parameter (alpha) equal to 0.5 and a proportion of invariable sites p-inv=0.15.

By default IQ-TREE computes empirical state frequencies from the alignment by counting, but one can also estimate the frequencies by maximum-likelihood with `+Fo` in the model name:

```
iqtree -s example.phy -m GTR+G+Fo
```

For amino-acid alignments, IQ-TREE use the empirical frequencies specified in the model. If you want frequencies as counted from the alignment, use `+F`, for example:

```
iqtree -s myprotein_alignment -m WAG+G+F
```

Note that all model specifications above can be used in the partition model NEXUS file.

5.10 Inferring site-specific rates

IQ-TREE allows to infer site-specific evolutionary rates if a site-rate heterogeneity model such as Gamma or FreeRate is the best model. Here, IQ-TREE will estimate model parameters and then apply an empirical Bayesian approach to assign site-rates as the mean over rate categories, weighted by the posterior probability of the site falling into each category. This approach is provided in IQ-TREE because such empirical Bayesian approach was shown to be most accurate (Mayrose et al., 2004). An example run:

```
iqtree -s example.phy --rate
# for version 1.x change --rate to -wsr
```

IQ-TREE will write an output file `example.phy.rate` that looks like:

```
Site    Rate    Category        Categorized_rate
1       0.26625 2       0.24393
2       0.99345 3       0.81124
3       2.69275 4       2.91367
4       0.25822 2       0.24393
5       0.25822 2       0.24393
6       0.42589 2       0.24393
7       0.30194 2       0.24393
8       0.72790 3       0.81124
```

```
9       0.25822 2       0.24393
10      0.09177 1       0.03116
```

The 1st column is site index of the alignment (starting from 1), the 2nd column `Rate` shows the mean site-specific rate as explained above, and the 3rd and 4th columns show the category index and rate of the Gamma rate category with the highest probability for this site (1 for slow and 4 for fast rate).

The above run will perform a full tree search. To speed up you can use `-n 0` to only use a parsimony tree for site rate estimates. Or if you have already infered an ML tree, you can specify it to improve the rate estimate:

```
iqtree -s example.phy -t ml.treefile -n 0 --rate
# for version 1.x change --rate to -wsr
```

where `-t` is the option to input a fixed tree topology and `ml.treefile` is the ML tree reconstructed previously.

If you already know the best-fit model for the alignment, you can use specify it via `-m` option to omit model selection and hence speed it up:

```
iqtree -s example.phy -m GTR+R10 -n 0 --rate
# for version 1.x change --rate to -wsr
```

Finally, IQ-TREE 2 allows to estimate rates by maximum likelihood via `--mlrate` option:

```
iqtree -s example.phy -n 0 --mlrate
```

This will print an output file `example.phy.mlrate` that looks like:

```
# Site-specific subtitution rates determined by maximum
   likelihood
# This file can be read in MS Excel or in R with command:
#   tab=read.table('example.phy.mlrate',header=TRUE)
# Columns are tab-separated with following meaning:
#   Site:   Alignment site ID
#   Rate:   Site rate estimated by maximum likelihood
Site    Rate
1       2.51550
2       12.89129
3       34.31350
4       2.44313
5       2.44313
```

```
6         4.41889
7         2.69577
8         9.27503
9         2.44313
10        0.00001
```

5.11 Where to go from here?

See Command Reference for a complete list of all options available in IQ-TREE.

Chapter 6

Assessing phylogenetic assumptions

It is important to know that phylogenetic models rely on various simplifying assumptions to ease computations. If your data severely violate these assumptions, it might cause bias in phylogenetic estimates of tree topologies and other model parameters. Some common assumptions include *treelikeness* (all sites in the alignment have evolved under the same tree), *stationarity* (nucleotide/amino-acid frequencies remain constant over time), *reversibility* (substitutions are equally likely in both directions), and *homogeneity* (substitution rates remain constant over time).

This document shows several ways to check some of these assumptions that you should perform before doing phylogenetic analysis.

6.1 Tests of symmetry

IQ-TREE provides three matched-pairs tests of symmetry (Naser-Khdour et al., 2019) to test the two assumptions of *stationarity* and *homogeneity*. A simple analysis:

```
iqtree2 -s example.phy -p example.nex --symtest-only
```

will perform the three tests of symmetry on every partition of the alignment and print the result into a `.symtest.csv` file. `--symtest-only` option tells IQ-TREE to only perform the tests of symmetry and then exit. In this example the content of `example.nex.symtest.csv` looks like this:

```
# Matched-pair tests of symmetry
# This file can be read in MS Excel or in R with command:
#    dat=read.csv('example.nex.symtest.csv',comment.char='#')
# Columns are comma-separated with following meanings:
```

```
#    Name:    Partition name
#    SymSig:  Number of significant sequence pairs by test of
   symmetry
#    SymNon:  Number of non-significant sequence pairs by test of
    symmetry
#    SymPval: P-value for maximum test of symmetry
#    MarSig:  Number of significant sequence pairs by test of
   marginal symmetry
#    MarNon:  Number of non-significant sequence pairs by test of
    marginal symmetry
#    MarPval: P-value for maximum test of marginal symmetry
#    IntSig:  Number of significant sequence pairs by test of
   internal symmetry
#    IntNon:  Number of non-significant sequence pairs by test of
    internal symmetry
#    IntPval: P-value for maximum test of internal symmetry
Name,SymSig,SymNon,SymPval,MarSig,MarNon,MarPval,IntSig,IntNon,
   IntPval
part1,44,92,0.475639,50,86,0.722371,4,132,0.23869
part2,43,93,0.142052,49,87,0.205232,5,131,0.169618
part3,53,83,0.00499855,58,78,0.00164132,6,130,0.343127
```

The three important columns are:

- SymPval: a small p-value (say < 0.05) indicates that the assumptions of stationarity or homogeneity or both is rejected. In this case, partition `part3` does not comply with these two assumptions (p-value = 0.00499855), whereas the other two partitions are "good".
- MarPval: a small p-value means that the assumption of stationarity is rejected. In this case, only partition `part3` does not comply with the stationary condition (p-value = 0.00164132).
- IntPval: a small p-value means that the homogeneity assumption is reject. In this case, no partitions are "bad" according to this test, i.e., they all comply with the homogeneity assumption.

This little example shows that only `part3` is problematic by not complying with the stationary assumption.

Now you may want to perform the phylogenetic analysis excluding all "bad" partitions by:

```
iqtree2 -s example.phy -p example.nex --symtest-remove-bad
```

that will remove all "bad" partitions where SymPval < 0.05 and continue the analysis with the remaining "good" partitions. You may then compare the trees from "all" partitions and from "good" only partitions to see if there is significant difference between them with tree topology tests.

Other options can be seen when running `iqtree2 -h`:

```
TEST OF SYMMETRY:
  --symtest                Perform three tests of symmetry
  --symtest-only           Do --symtest then exist
  --symtest-remove-bad     Do --symtest and remove bad partitions
  --symtest-remove-good    Do --symtest and remove good partitions
  --symtest-type MAR|INT   Use MARginal/INTernal test when
     removing partitions
  --symtest-pval NUMER     P-value cutoff (default: 0.05)
  --symtest-keep-zero      Keep NAs in the tests
```

6.2 Likelihood mapping

Likelihood mapping (Strimmer and von Haeseler, 1997) is a visualisation method to display the phylogenetic information of an alignment. It visualises the *treelikeness* of all quartets in a single triangular graph and therefore renders a quick interpretation of the phylogenetic content.

A simple likelihood mapping analysis can be conducted with:

```
iqtree -s example.phy -lmap 2000 -n 0
```

where `-lmap` option specify the number of quartets of taxa that will be drawn randomly from the alignment. `-n 0` tells IQ-TREE to stop the analysis right after running the likelihood mapping. IQ-TREE will print the result in the `.iqtree` report file as well as the likelihood mapping plot `.lmap.svg` (in SVG format) and `.lmap.eps` file (in EPS figure format).

You can now view the likelihood mapping plot file `example.phy.lmap.svg`, which looks like this:

It shows phylogenetic information of the alignment `example.phy`.

- Top sub-figure: distribution of quartets depicted by dots on the likelihood mapping plot.
- Left sub-figure: percentages of quartets falling in each of the three areas. The three areas show support for one of the different groupings like (a,b)-(c,d).

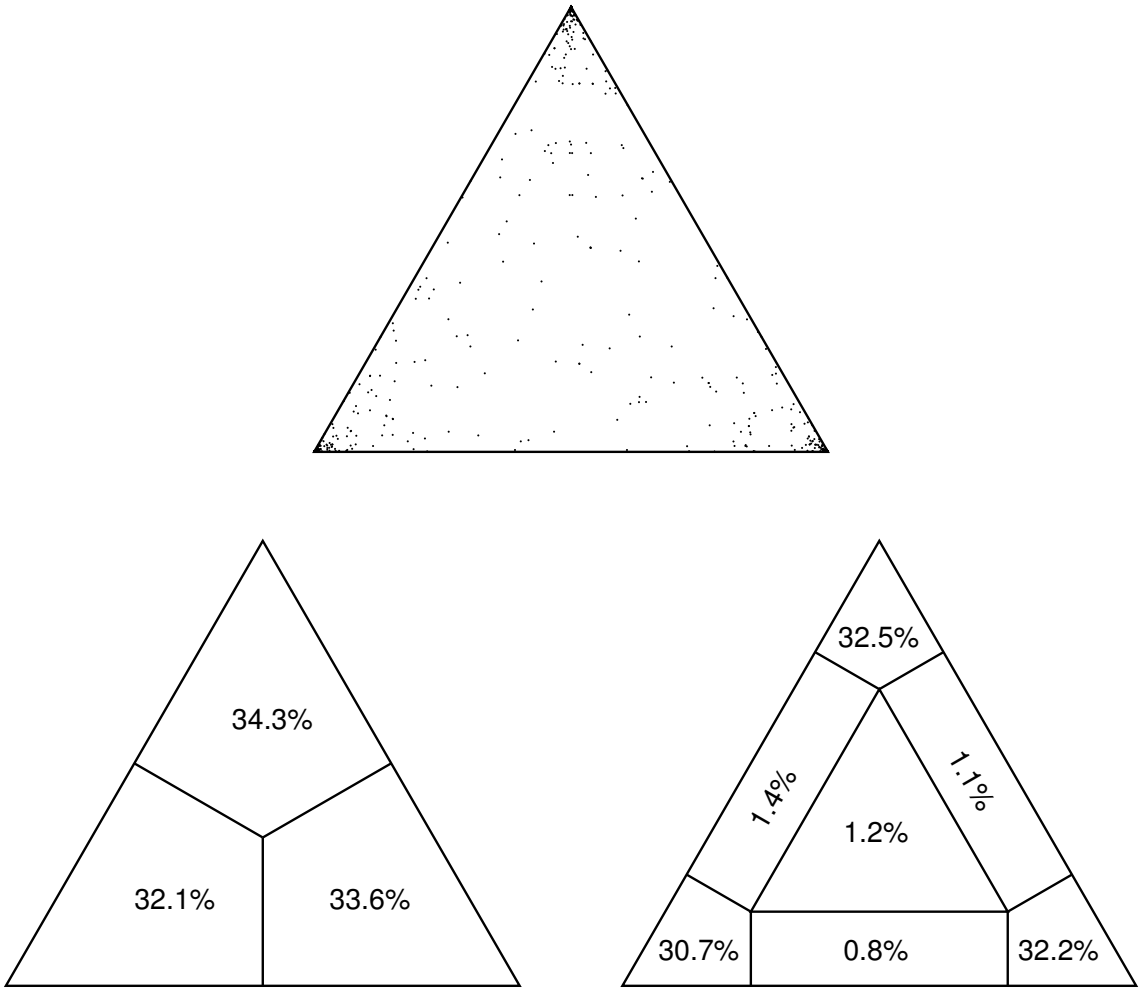


Figure 6.1: Likelihood mapping plot.

- Right sub-figure: percentages of quartets falling in each of the seven areas. Quartets falling into the three corners are informative and called fully-resolved quartets. Those in three rectangles are partly informative (partly resolved quartets) and those in the center are uninformative (unresolved quartets). A good data set should have high number of fully resolved quartets and low number of unresolved quartets.

The meanings can also be found in the `LIKELIHOOD MAPPING STATISTICS` section of the report file `example.phy.iqtree`:

```
LIKELIHOOD MAPPING STATISTICS
-----------------------------

           (a,b)-(c,d)                                  (a,b)-(c,d)
                /\                                           /\
               /  \                                         /  \
              /    \                                       /  1 \
             /  a1  \                                     / \  / \
            /\      /\                                   /   \/   \
           /  \    /  \                                 /    /\    \
          /    \  /    \                               / 6  /  \  4 \
         /      \/      \                             /\   /  7 \   /\
        /        |       \                           /  \ /______\ /
           \
       /   a3    |    a2  \                        / 3  |    5   |  2
           \
      /__________|_________\                      /_____|________|
          _____\
(a,d)-(b,c)            (a,c)-(b,d)      (a,d)-(b,c)            (a
   ,c)-(b,d)

Division of the likelihood mapping plots into 3 or 7 areas.
On the left the areas show support for one of the different
   groupings
like (a,b|c,d).
On the right the right quartets falling into the areas 1, 2 and 3
    are
informative. Those in the rectangles 4, 5 and 6 are partly
   informative
and those in the center (7) are not informative.
.....
```

The command reference will provide more options and how to perform 2-, 3-, or 4-

cluster likelihood mapping analysis.

Chapter 7

Concordance Factor

Since IQ-TREE 2, we provide two measures for quantifying genealogical concordance in phylogenomic datasets: the gene concordance factor (gCF) and the site concordance factor (sCF). For every branch of a reference tree, gCF is defined as the percentage of "decisive" gene trees containing that branch. gCF is already in wide usage, but here we allow to calculate gCF while correctly accounting for variable taxon coverage among the gene trees. sCF is defined as the percentage of decisive alignment sites supporting a branch in the reference tree. sCF is a novel measure that is particularly useful when individual gene alignments are relatively uninformative, such that gene trees are uncertain. gCF and sCF complement classical measures of branch support (e.g. bootstrap) in phylogenetics by providing a full description of underlying disagreement among loci and sites.

If you use this feature please cite:

Minh B.Q., Hahn M.W., Lanfear R. (2020) New methods to calculate concordance factors for phylogenomic datasets. *Molecular Biology and Evolution*, 37:2727–2733. https://doi.org/10.1093/molbev/msaa106

For sCF we recommend that you use the more accurate version of sCF based on maximum likelihood (`--scfl` option instead of `--scf`) that is available since IQ-TREE v2.2.2. In that case please cite:

Mo Y.K., Lanfear R., Hahn M.W., and Minh B.Q. (2022) Updated site concordance factors minimize effects of homoplasy and taxon sampling. *Bioinformatics*, in press. https://doi.org/10.1093/bioinformatics/btac741

HINT: See very nice tips on how to use and interpret concordance factors written by Rob Lanfear.

7.1 Inferring species tree

First, you need to infer a reference tree (e.g. a species tree), on which the concordance factors will be annotated. The species tree can be reconstructed by a concatenation/-supermatrix approach or a coalescent/reconciliation/supertree approach. Here, we will use the concatenation approach in IQ-TREE.

As an example, you can apply an edge-linked proportional partition model with ultra-fast bootstrap (1000 replicates; for comparison with concordance factors):

```
iqtree2 -s ALN_FILE -p PARTITION_FILE --prefix concat -B 1000 -T
   AUTO
```

where `ALN_FILE` and `PARTITION_FILE` are your input files. `-T AUTO` is to detect the best number of CPU cores. Here we use a prefix `concat`, so that all output files (`concat.*`) do not interfere with analyses below. If `--prefix` is omitted, all output files will be `PARTITION_FILE.*`.

Moreover, IQ-TREE 2 provides a new convenient feature: if you have a directory with many (locus) alignments, you can specify this directory directly with `-p` option:

```
iqtree2 -p ALN_DIR --prefix concat -B 1000 -T AUTO
```

IQ-TREE detects if `-p` argument is a directory and automatically load all alignment files and concatenate them into a supermatrix for the partition analysis.

7.2 Inferring gene/locus trees

We now construct a set of gene/locus trees. One can manually do a for-loop, but IQ-TREE 2 provides a new convenient option `-S` to compute individual locus trees given a partition file or a directory:

```
iqtree2 -s ALN_FILE -S PARTITION_FILE --prefix loci -T AUTO
# or
iqtree2 -S ALN_DIR --prefix loci -T AUTO
```

In the second case, IQ-TREE automatically detects that `ALN_DIR` is a directory and will load all alignment files within the directory. So `-S` takes the same argument as `-p`

except that it performs model selection (ModelFinder) and tree inference separately for each partition/alignment. The output files are similar to those from a partitioned analysis, except that `loci.treefile` now contains a set of trees.

7.3 Gene concordance factor (gCF)

Given the species tree `concat.treefile` and the set of locus trees `loci.treefile` computed above, you can calculate gCF for each branch of the species tree as the fraction of decisive gene trees concordant with this branch:

```
iqtree2 -t concat.treefile --gcf loci.treefile --prefix concord
```

Note that `-t` accepts any reference tree (e.g., by coalescent/reconciliation approach) and `--gcf` accepts any set of trees (e.g. locus trees and bootstrap trees), which may contain a subset of taxa from the reference tree. IQ-Tree will write three files:

- `concord.cf.tree`: Newick tree with gCF assigned for each internal branch of the reference tree. If the reference tree already has some branch label (such as bootstrap support in this case), gCF will be appended to the existing label separated by a `/`.
- `concord.cf.branch`: Newick tree with internal branch IDs.
- `concord.cf.stat`: A tab-separated table with gCF and gDF (gene discordance factor) for every internal branch (rows of the table). The ID column can be linked with `concord.cf.branch` file. This file can be read in R to do some plot (see below).

If you omit `--prefix`, all output files will be written to `concat.treefile.*`.

7.4 Site concordance factor (sCF)

NOTE: From version 2.2.2 IQ-TREE provides a new and more accurate sCF based on likelihood via `--scfl` option (Mo et al., 2022), whereas the original sCF is based on parsimony. You can download this version from here.

Given the species tree `concat.treefile` and the alignment, you can calculate sCF for each branch of the species tree as the fraction of decisive alignment sites supporting that branch:

```
# for version 2.2.2 or above
iqtree2 -te concat.treefile -s ALN_FILE --scfl 100 --prefix
   concord
```

```
# older versions
iqtree2 -t concat.treefile -s ALN_FILE --scf 100 --prefix concord
    -T 10
```

`--scf` specifies the number of quartets (randomly sampled around each internal branch) for computing sCF. We recommend at least 100 quartets for stable sCF values. Note that running this command several times may lead to slightly different sCF due to randomness. To make it reproducible, you need to use `-seed` option to provide a random number generator seed.

Note that the `--scfl` option from IQ-TREE v2.2.2 will invoke model selection with ModelFinder and also tree search if you don't specify a tree with `-te` option. If you already have a best-fit model from a previous run, you can ignore ModelFinder (and thus speed up this run) by provide the model with `-m` option.

Instead of `-s`, you can alternatively provide a directory or a partition file. IQ-Tree then computes sCF for the concatenated alignment:

```
# for version 2.2.2 or above
iqtree2 -te concat.treefile -p ALN_DIR --scfl 100 --prefix
   concord
# older versions
iqtree2 -t concat.treefile -p ALN_DIR --scf 100 --prefix concord
   -T 10
```

Finally, you can combine gCF and sCF within a single run:

```
# only for the original sCF
iqtree2 -t concat.treefile --gcf loci.treefile -p ALN_DIR --scf
   100 --prefix concord -T 10
```

Here, each branch of `concord.cf.tree` will be assigned (or appended) with `gCF/sCF` values and `concord.cf.stat` will be written with both gCF and sCF values.

7.5 Putting it all together

If you have separate alignments for each locus in a folder, then perform the following commands:

```
# infer a concatenation-based species tree with 1000 ultrafast
   bootstrap and an edge-linked partition model
iqtree2 -p ALN_DIR --prefix concat -B 1000 -T AUTO
```

```
# infer the locus trees
iqtree2 -S ALN_DIR --prefix loci -T AUTO

# compute gene concordance factors
iqtree2 -t concat.treefile --gcf loci.treefile --prefix concord

# compute site concordance factor using likelihood with v2.2.2
iqtree2 -te concat.treefile -p ALN_DIR --scfl 100 --prefix
   concord2
```

If you have a single concatenated alignment with a partition file that defines loci:

```
# infer a concatenation-based species tree with 1000 ultrafast
   bootstrap and an edge-linked partition model
iqtree2 -s ALN_FILE -p PARTITION_FILE --prefix concat -B 1000 -T
   AUTO

# infer the locus trees
iqtree2 -s ALN_FILE -S PARTITION_FILE --prefix loci -T AUTO

# compute gene concordance factors
iqtree2 -t concat.treefile --gcf loci.treefile --prefix concord

# compute site concordance factor using likelihood with v2.2.2
iqtree2 -te concat.treefile -s ALN_FILE --scfl 100 --prefix
   concord2
```

Note that you can adjust `-T 10` if you have fewer/larger CPU cores.

7.6 Calculating concordance factors on very large datasets

If you have a dataset which takes a long time to analyse on your machine, there are a couple of adjustments you can make to the above process to keep things as fast as possible.

Specifically, because the new version of the site concordance factor uses likelihoods, we can make sure to re-use as much information as possible.

So, suppose that in the first step of the analysis you ran the command as above:

```
iqtree2 -s ALN_FILE -p PARTITION_FILE --prefix concat -B 1000 -T
   AUTO
```

That command will have figured out for you the model of evolution, all the parameters of that model, and the branch lengths of the corresponding tree. We can re-use all of that useful information in the final step. It just takes a little bit of effort to find what you need.

First we'll get the model parameters we need. If you take a look at the end of the `concat.log` file you will find a little section called `ALISIM COMMAND`. You can find it like this on mac/linux (or just open the `concat.log` file in a text editor and scroll to the end:

```
tail concat.log
```

You should see something like this:

```
ALISIM COMMAND
--------------
--alisim simulated_MSA -t concat.treefile -m "Q.plant+I
   {0.177536}+R8
   {0.147295,0.0935335,0.114418,0.190578,0.108376,0.538389,0.113777,0.804005,
   " --length 432014
```

That bit after the `-m` (not including the `--length` stuff) is what you need to specify the Maximum Likelihood model parameters when you run the `--scfl` command. Note that it's vital that you use the model from YOUR analysis, not the example provided here. (That's why this bit is an a longer and more detailed section at the end of the tutorial.)

We also want to re-use the branch lengths we calculated in step 1, and we can do that easily with the `-blfix` option.

To put all of that together, we are going to change the final command of the tutorial above, where we calculate the site concordance factors from one of these two options (depending on if your alignments are per-locus, or all concatentated):

```
# simple command, with per-locus alignments
# compute site concordance factor using likelihood with v2.2.2
iqtree2 -te concat.treefile -p ALN_DIR --scfl 100 --prefix
   concord2

# simple command, with concatenated alignments
# compute site concordance factor using likelihood with v2.2.2
iqtree2 -te concat.treefile -s ALN_FILE --scfl 100 --prefix
   concord2
```

To one of these, where we add the two extra commands via `-blfix` and `-m`, to fix all the parameters we already calculated. A reminder - do NOT use the exact commandlines above. You have to replace everything after the `-m` with what you found in your own `concat.log` file:

```
# faster analysis, using pre-computed model parameters, with per-
   locus alignments
# compute site concordance factor using likelihood with v2.2.2
iqtree2 -te concat.treefile -p ALN_DIR --scfl 100 --prefix
   concord2 -blfix -m "Q.plant+I{0.177536}+R8
   {0.147295,0.0935335,0.114418,0.190578,0.108376,0.538389,0.113777,0.804005,0.
   "

# faster analysis, using pre-computed model parameters, with
   concatenated alignments
# compute site concordance factor using likelihood with v2.2.2
iqtree2 -te concat.treefile -s ALN_FILE --scfl 100 --prefix
   concord2 -blfix -m "Q.plant+I{0.177536}+R8
   {0.147295,0.0935335,0.114418,0.190578,0.108376,0.538389,0.113777,0.804005,0.
   "
```

All this does is tells IQ-TREE to use the model parameters and branch lengths you already calculated. On large datasets this can save a lot of analysis time.

layout: userdoc title: “Phylogenetic Dating” author: Minh Bui, Rob Lanfear date: 2021-03-11 docid: 7 icon: info-circle doctype: tutorial tags: - tutorial description: “Building time tree with node dates on phylogenetic trees.” sections: - name: Inferring time tree with tip dates url: inferring-time-tree-with-tip-dates - name: Calibrating tree using ancestral dates url: calibrating-tree-using-ancestral-dates - name: Dating an existing tree url: dating-an-existing-tree - name: Obtaining confidence intervals url: obtaining-confidence-intervals - name: Excluding outlier taxa/nodes url: excluding-outlier-taxanodes - name: Full list of LSD2 options url: full-list-of-lsd2-options —

Chapter 8

Phylogenetic Dating

Since IQ-TREE 2.0.3, we integrate the least square dating (LSD2) method to build a time tree when you have date information for tips or ancestral nodes. So if you use this feature please cite:

Thu-Hien To, Matthieu Jung, Samantha Lycett, Olivier Gascuel (2016) Fast dating using least-squares criteria and algorithms. *Syst. Biol.* 65:82-97. https://doi.org/10.1093/sysbio/syv068

We will now walk through examples but the full options are:

```
TIME TREE RECONSTRUCTION:
  --date FILE           Dates of tips or ancestral nodes
  --date TAXNAME        Extract dates from taxon names after last
     '|'
  --date-tip STRING     Tip dates as a real number or YYYY-MM-DD
  --date-root STRING    Root date as a real number or YYYY-MM-DD
  --date-ci NUM         Number of replicates to compute confidence
      interval
  --clock-sd NUM        Std-dev for lognormal relaxed clock (
     default: 0.2)
  --date-outlier NUM    Z-score cutoff to exclude outlier nodes (e
     .g. 3)
  --date-options ".."   Extra options passing directly to LSD2
```

DISCLAIMER: Please download version 2.0.6 with new options like `--date-ci`.
This feature is new and might still have bugs. So suggestions and bug reports are much welcome.

8.1 Inferring time tree with tip dates

This is a common scenario e.g. in virus datasets where you have sampling time for many sequences. You need first to prepare a *date file*, which comprises several lines, each with a taxon name (from your sequence alignment) and its date separated by spaces, tabs or blanks. Note that it is not required to have dates for all tips. For example, this date file is part of the new corona virus dataset:

```
hCoV-19/Wuhan-Hu-1          2019-12-31
hCoV-19/China/WF0028        2020-02
hCoV-19/USA/WA-S88          2020-03-01
hCoV-19/USA/CA-CDPH-UC1     2020
hCoV-19/Italy/SPL1          2020-01-29
hCoV-19/Spain/Valencia5     2020-02-27
hCoV-19/Australia/QLD01     2020-01-28
hCoV-19/Vietnam/CM295       2020-03-06
hCoV-19/bat/Yunnan          2013-07-24
hCoV-19/pangolin/Guangdong 2019-02-01:2019-12-31
```

The date information here can be uncertain. For example, `hCoV-19/China/WF0028` was sampled in Feb 2020, `hCoV-19/USA/CA-CDPH-UC1` was sampled in 2020, and `hCoV-19/pangolin/Guangdong` was sample between 1st Feb 2019 and 31st Dec 2019. For such data range you can use "NA" to mean that the lower or upper bound is missing, e.g.:

```
TaxonA  2018-02-01:NA
TaxonB  NA:2018-03-31
```

which means that `TaxonA` was sampled after 1st Feb 2018 and TaxonB was sampled before 31st Mar 2018.

Now run IQ-TREE with:

```
iqtree -s ALN_FILE --date DATE_FILE
```

where `ALN_FILE` is the sequence alignment and `DATE_FILE` is the date file. This single command line will perform three steps: (1) find the best-fit model using ModelFinder,

(2) find the maximum likelihood (ML) tree with branch lengths in number of substitutions per site, and (3) rescale the branch lengths of the ML tree to build a time tree with dated ancestral node. As output IQ-TREE will additional print three files:

- `ALN_FILE.timetree.lsd`: The report of LSD.
- `ALN_FILE.timetree.nex`: Time tree file in NEXUS format, that can be viewed nicely in FigTree (Click on "Node Labels" on the left tab and choose "Display" as "date" in FigTree, see figure below).
- `ALN_FILE.timetree.nwk`: Time tree file in NEWICK format.

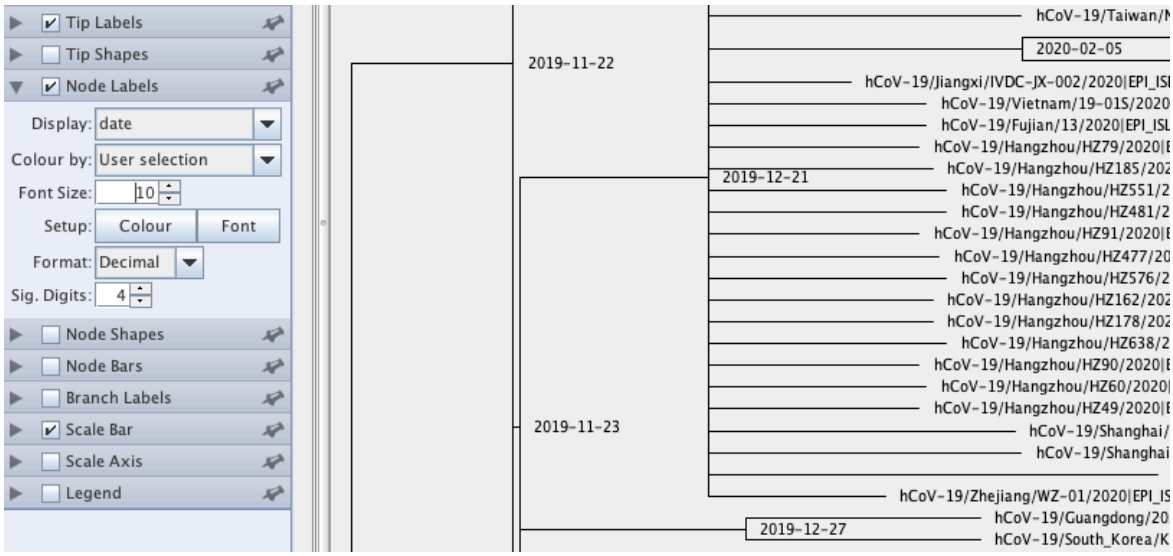


Figure 8.1: Node dates in FigTree

This command will automatically detect the best root position (according to LSD criterion). However, if the root is incorrectly inferred, it may produce wrong dates. Therefore, it is advisable to provide outgroup taxa if possible. In this example, we have this information, so you can use `-o` option:

```
iqtree -s ALN_FILE --date DATE_FILE -o "hCoV-19/bat/Yunnan,hCoV
   -19/pangolin/Guangdong"
```

to instruct IQ-TREE that the root is on the branch separating `bat` and `pangolin` sequences from the rest.

Alternatively you can also append the dates into the sequence names of the alignment file using the `|` separator, such as (assuming a FASTA file here):

```
>hCoV-19/Wuhan-Hu-1|2019-12-31
......
```

```
>hCoV-19/China/WF0028|2020-02
......
>hCoV-19/USA/WA-S88|2020-03-01
......
>hCoV-19/USA/CA-CDPH-UC1|2020
......
>hCoV-19/Italy/SPL1|2020-01-29
......
>hCoV-19/Spain/Valencia5|2020-02-27
......
>hCoV-19/Australia/QLD01|2020-01-28
......
>hCoV-19/Vietnam/CM295|2020-03-06
......
>hCoV-19/bat/Yunnan|2013-07-24
......
>hCoV-19/pangolin/Guangdong|2019
......
```

Then run IQ-TREE:

```
iqtree -s ALN_FILE --date TAXNAME -o "hCoV-19/bat/Yunnan,hCoV-19/
   pangolin/Guangdong"
```

The special keyword `TAXNAME` for the `--date` option instructs IQ-TREE to automatically extract the dates from the taxon names.

8.2 Calibrating tree using ancestral dates

Another scenario is that we have sequences from present day and want to calibrate the dates of the ancestral nodes. This will only work if you have fossil date record of at least one ancestral node in the tree. Then you again need to prepare a date file which looks like:

```
taxon1,taxon2          -50
taxon3,taxon4,taxon5   -100
taxon6                 -10
```

which, for example, mean that the most recent common ancestor (MRCA) of `taxon1` and `taxon2` was 50 mya (million year ago) and the MRCA of `taxon3`, `taxon4`, `taxon5` was 100 mya. Note that **no empty space** should be added to the comma-separated list of taxa, as empty space is used as a separator between taxon list and dates.

Now run IQ-TREE:

```
iqtree -s ALN_FILE --date DATE_FILE --date-tip 0
```

This means that except for `taxon6`, all other taxa have the date of 0 for presence.

If you know the root date, then you can set it via `--date-root` option.

8.3 Dating an existing tree

If you already have a tree, you can use option `-te TREE_FILE` to ask IQ-TREE to load and fix this tree topology:

```
iqtree -s ALN_FILE --date DATE_FILE -te TREE_FILE
```

This will work with the scenarios above, i.e., IQ-TREE will date the user-defined tree instead of the ML tree. To further speed up the process: If you know the model already, you set can it via `-m` option; or in a partitioned analysis, you can provide a partition file with specified models.

8.4 Obtaining confidence intervals

To infer the confidence interval of the estimated dates, use `--date-ci` option:

```
iqtree -s ALN_FILE --date DATE_FILE --date-ci 100
```

which will resample branch lengths 100 times to infer the confidence intervals. Note that this is not bootstrap and the method is much faster but unpublished. Roughly speaking, it is based on a mixture of Poisson and lognormal distributions for a relaxed clock model. You can control the standard deviation of the lognormal distribution via `--clock-sd` option. The default is 0.2. If you set a higher value, the confidence interval will become wider.

8.5 Excluding outlier taxa/nodes

Long branches may cause biased date estimates. To detect and exclude outlier taxa or nodes prior to dating, use `--date-outlier` option:

```
iqtree -s ALN_FILE --date DATE_FILE --date-outlier 3
```

that specifies a z-score threshold to detect outliers. The higher this value is, the more outliers will be removed from the resulting time tree.

8.6 Full list of LSD2 options

The main options in IQ-TREE provide easy access to the key LSD2 functions. If you would like more control of what LSD2 is doing, you can use the `--date-options "..."` command to pass any valid options to LSD2. For example, to control the way that LSD2 treats outliers, you can do this:

```
iqtree -s ALN_FILE --date DATE_FILE --date-options "-e 2"
```

A full list of the options for LSD2 can be obtained by downloading LSD2 and running `lsd2 -h`, the output of that command is reproduced here for convenience:

```
LSD: LEAST-SQUARES METHODS TO ESTIMATE RATES AND DATES - v.1.8

DESCRIPTION
    This program estimates the rate and the dates of the input
       phylogenies given some temporal constraints.
    It minimizes the square errors of the branch lengths under
       normal distribution model.

SYNOPSIS
    ./lsd [-i inputFile] [-d inputDateFile] [-o outputFile] [-s
       sequenceLength] [-g outgroupFile] [-f nbSamplings]
OPTIONS
    -a rootDate
       To specify the root date if there's any. If the root date
          is not a number, but a string (ex: 2020-01-10, or b
          (2019,2020)) then it should
       be put between the quotes.
    -b varianceParameter
       The parameter (between 0 and 1) to compute the variances
          in option -v. It is the pseudo positive constant to add
           to the branch lengths
       when calculating variances, to adjust the dependency of
          variances to branch lengths. By default b is the
          maximum between median branch length
       and 10/seqlength; but it should be adjusted  based on how/
          whether the input tree is relaxed or strict. The
          smaller it is the more variances
       would be linear to branch lengths, which is relevant for
          strict clock. The bigger it is the less effect of
          branch lengths on variances,
       which might be better for relaxed clock.
    -d inputDateFile
```

```
      This options is used to read the name of the input date
         file which contains temporal constraints of internal
         nodes
      or tips. An internal node can be defined either by its
         label (given in the input tree) or by a subset of tips
         that have it as
      the most recent common ancestor (mrca). A date could be a
         real or a string or format year-month-day.
      The first line of this file is the number of temporal
         constraints. A temporal constraint can be fixed date,
         or a
      lower bound l(value), or an upper bound u(value), or an
         interval b(v1,v2)
      For example, if the input tree has 4 taxa a,b,c,d, and an
         internal node named n, then following is a possible
         date file:
       6
       a l(2003.12)
       b u(2007.07)
       c 2005
       d b(2001.2,2007.11)
       mrca(a,b,c,d) b(2000,2001)
       n l(2004.3)
      If this option is omitted, and option -a, -z are also
         omitted, the program will estimate relative dates by
         giving T[root]=0 and T[tips]=1.
   -D outDateFormat
       Specify output date format: 1 for real, 2 for year-month-
          day. By default the program will guess the format of
          input dates and uses it for
       output dates.
   -e ZscoreOutlier
      This option is used to estimate and exclude outlier nodes
         before dating process.
      LSD2 normalize the branch residus and decide a node is
         outlier if its related residus is great than the
         ZscoreOutlier.
      A normal value of ZscoreOutliercould be 3, but you can
         adjust it bigger/smaller depending if you want to have
      less/more outliers. Note that for now, some
         functionalities could not be combined with outliers
         estimation, for example
      estimating multiple rates, imprecise date constraints.
```

```
-f samplingNumberCI
   This option calculates the confidence intervals of the
      estimated rate and dates. The branch lengths of the
      esimated
   tree are sampled samplingNumberCI times to generate a set
      of simulated trees. To generate simulated lengths
   for each branch, we use a Poisson distribution whose mean
      equals to the estimated one multiplied by the sequence
      length, which is
   1000 by default if nothing was specified via option -s.
      Long sequence length tends to give small confidence
      intervals. To avoid
   over-estimate the confidence intervals in the case of very
       long sequence length but not necessarily strict
      molecular clock, you
   could use a smaller sequence length than the actual ones.
      Confidence intervals are written in the nexus tree with
       label CI_height,
   and can be visualzed with Figtree under Node bar feature.
-g outgroupFile
   If your data contain outgroups, then specify the name of
      the outgroup file here. The program will use the
      outgroups to root the trees.
   If you use this combined with options -G, then the
      outgroups will be removed. The format of this file
      should be:
        n
        OUTGROUP1
        OUTGROUP2
        ...
        OUTGROUPn
-F
   By default without this option, we impose the constraints
      that the date of every node is equal or smaller then
      the
   dates of its descendants, so the running time is quasi-
      linear. Using this option we ignore this temporal
      constraints, and
   the the running time becomes linear, much faster.
-h help
   Print this message.
-i inputTreesFile
   The name of the input trees file. It contains tree(s) in
```

```
        newick format, each tree on one line. Note that the
        taxa sets of all
    trees must be the same.
-j
    Verbose mode for output messages.
-G
    Use this option to remove the outgroups (given in option -
        g) in the estimated tree. If this option is not used,
        the outgroups
    will be kept and the root position in estimated on the
        branch defined by the outgroups.
-l nullBlen
    A branch in the input tree is considered informative if
        its length is greater this value. By default it is 0.5/
        seq_length. Only
    informative branches are forced to be bigger than a
        minimum branch length (see option -u for more
        information about this).
-m samplingNumberOutlier
    The number of dated nodes to be sampled when detecting
        outlier nodes. This should be smaller than the number
        of dated nodes,
    and is 10 by default.
-n datasetNumber
    The number of trees that you want to read and analyse.
-o outputFile
    The base name of the output files to write the results and
         the time-scale trees.
-p partitionFile
    The file that defines the partition of branches into
        multiple subsets in the case that you know each subset
        has a different rate.
    In the partition file, each line contains the name of the
        group, the prior proportion of the group rate compared
        to the main rate
    (selecting an appropriate value for this helps to converge
         faster), and a list of subtrees whose branches are
        supposed to have the
    same substitution rate. All branches that are not assigned
         to any subtree form a group having another rate.
    A subtree is defined between {}: its first node
        corresponds to the root of the subtree, and the
        following nodes (if there any)
```

```
      correspond to the tips of the subtree. If the first node
         is a tip label then it takes the mrca of all tips as
         the root of the subtree.
      If the tips of the subtree are not defined (so there's
         only the defined root), then by
      default this subtree is extended down to the tips of the
         full tree. For example the input tree is
      ((A:0.12,D:0.12)n1:0.3,((B:0.3,C:0.5)n2:0.4,(E:0.5,(F:0.2,
         G:0.3)n3:0.33)n4:0.22)n5:0.2)root;
      and you have the following partition file:
            group1 1 {n1} {n5 n4}
            group2 1 {n3}
      then there are 3 rates: the first one includes the
         branches (n1,A), (n1,D), (n5,n4), (n5,n2), (n2,B), (n2,
         C); the second one
      includes the branches (n3,F), (n3,G), and the last one
         includes all the remaining branches. If the internal
         nodes don't have labels,
      then they can be defined by mrca of at least two tips, for
          example n1 is mrca(A,D)
   -q standardDeviationRelaxedClock
      This value is involved in calculating confidence intervals
          to simulate a lognormal relaxed clock. We multiply the
          simulated branch lengths
      with a lognormal distribution with mean 1, and standard
         deviation q. By default q is 0.2. The bigger q is, the
         more your tree is relaxed
      and give you bigger confidence intervals.
   -r rootingMethod
      This option is used to specify the rooting method to
         estimate the position of the root for unrooted trees,
         or
      re-estimate the root for rooted trees. The principle is to
          search for the position of the root that minimizes
      the objective function.
      Use -r l if your tree is rooted, and you want to re-
         estimate the root locally around the given root.
      Use -r a if you want to estimate the root on all branches
         (ignoring the given root if the tree is rooted).
          In this case, if the constrained mode is chosen (
             option -c), method "a" first estimates the root
             without using the constraints.
          After that, it uses the constrained mode to improve
```

```
            locally the position of the root around this pre-
            estimated root.
      Use -r as if you want to estimate to root using
         constrained mode on all branches.
      Use -r k if you want to re-estimate the root position on
         the same branche of the given root.
          If combined with option -g, the root will be estimated
              on the branche defined by the outgroups.
   -R round_time
      This value is used to round the minimum branch length of
         the time scaled tree. The purpose of this is to make
         the minimum branch length
      a meaningful time unit, such as day, week, year ... By
         default this value is 365, so if the input dates are
         year, the minimum branch
      length is rounded to day. The rounding formula is round(R*
         minblen)/R.
   -s sequenceLength
      This option is used to specify the sequence length when
         estimating confidence intervals (option -f). It is used
          to generate
      integer branch lengths (number of substitutions) by
         multiplying this with the estimated branch lengths. By
         default it is 1000.
   -S minSupport
      Together with collapsing internal short branches (see
         option -l), users can also collapse internal branches
         having weak support values (if
      provided in the input tree) by using this option. The
         program will collapse all internal branches having
         support <= the specifed value.
   -t rateLowerBound
      This option corresponds to the lower bound for the
         estimating rate. It is 1e-10 by default.
   -u minBlen
      By default without this option, lsd2 forces every branch
         of the time scaled tree to be greater than 1/(
         seq_length*rate) where rate is
      an pre-estimated median rate. This value is rounded to the
          number of days or weeks or years, depending on the
         rounding parameter -R.
      By using option -u, the program will not estimate the
         minimum branch length but use the specified value
```

```
      instead.
-U minExBlen
   Similar to option -u but applies for external branches if
      specified. If it's not specified then the minimum
      branch length of external
   branches is set the same as the one of internal branch.
-v variance
   Use this option to specify the way you want to apply
      variances for the branch lengths. Variances are used to
       recompense big errors on
   long estimated branch lengths. The variance of the branch
      Bi is Vi = (Bi+b) where b is specified by option -b.
   If variance=0, then we don't use variance. If variance=1,
      then LSD uses the input branch lengths to calculate
      variances.
   If variance=2, then LSD runs twice where the second time
      it calculates the variances based on the estimated
      branch
   lengths of the first run. By default variance=1.
-V
   Get the actual version.
-w givenRte
   This option is used to specify the name of the file
      containing the substitution rates.
   In this case, the program will use the given rates to
      estimate the dates of the nodes.
   This file should have the following format
        RATE1
        RATE2
        ...
  where RATEi is the rate of the tree i in the inputTreesFile
     .
-z tipsDate
   To specify the tips date if they are all equal. If the
      tips date is not a number, but a string (ex:
      2020-01-10, or b(2019,2020))
   then it should be put between the quotes.
```

Chapter 9

Rooting phylogenetic trees

Using time-reversible Markov models is a very common practice in phylogenetic analysis, because they provide high computational efficiency. However, these models infer *unrooted* trees hence lack the ability to infer the root placement of the estimated phylogeny. In order to compensate for the inability of these models to root the tree, many researchers use external information such as using outgroup taxa or additional assumptions such as molecular-clocks.

This guide provides the outgroup approach and another rooting approach using *non-reversible* models (Naser-Khdour et al., 2021), which will be useful when an outgroup is lacking. Please make sure that you use IQ-TREE **version 2.1.3** or later for full features below and cite this manuscript:

> S. Naser-Khdour, B.Q. Minh, R. Lanfear (2021) Assessing Confidence in Root Placement on Phylogenies: An Empirical Study Using Non-Reversible Models. https://doi.org/10.1101/2020.07.31.230144

9.1 Inferring unrooted tree with outgroup

We first demonstrate the outgroup approach to root the Bovidae family of five sampled species (Yak, Cow, Goat, Sheep and Tibetan antelope) using two outgroup species (Pig and Whale). Please download:

- An input DNA alignment file for these 7 species.
- An input partition file that defines 52 genes in this alignment. This is a subset of the mammal dataset (Wu et al., 2018).

Choosing a "good" outgroup is an entire topic on its own. In generally, the outgroup must contain taxa that do not belong to the ingroup but are evolutionarily close enough to the ingroup taxa.

To infer an unrooted tree, run:

```
iqtree2 -s bovidae_outgroup.phy -p bovidae.nex -B 1000 -T AUTO --
   prefix rev_dna_outg
```

that will invoke the ultrafast bootstrap with 1000 replicates (`-B 1000`), detect the optimal number of threads (`-T AUTO`) and write all output files with the prefix `rev_dna_outg` .

The input alignment contains protein-coding genes. We can ask IQ-TREE to translate the alignment into protein sequences using the standard genetic code (`-st NT2AA`) and perform an amino-acid analysis on the translated alignment with:

```
iqtree2 -s bovidae_outgroup.phy -p bovidae.nex -B 1000 -T AUTO -
   st NT2AA --prefix rev_aa_outg
```

where setting the prefix to `rev_aa_outg` avoids file overwriting with the previous run. The resulting tree may now look like (extracted from `rev_aa_outg.iqtree`):

```
NOTE: Tree is UNROOTED although outgroup taxon 'Yak' is drawn at
   root
Numbers in parentheses are  ultrafast bootstrap support (%)

+--Yak
|
+--Cow
|
|          +--Goat
|       +--| (100)
|       |  +--Sheep
|   +---| (100)
|   |   +---Tibetan_antelope
+---| (100)
    |                       +-------------------------------
       Wild_pig
    +----------------------| (100)
                           +--------------------Minke_whale
```

You can open `rev_aa_outg.treefile` in a tree viewer software (e.g. FigTree) and re-root the tree on the branch separating the outgroup (`Wild_pig` and `Minke_whale`) from the remaining ingroup to obtain an outgroup-rooted tree.

Finally, if you want you can also perform a non-partition analysis by removing the option `-p`.

9.2 Inferring rooted trees without outgroup

We will now infer a rooted tree using non-reversible models. Please download:

- An input DNA alignment file for 5 ingroup species (Yak, Cow, Goat, Sheep and Tibetan antelope). This is a sub-alignment of the alignment above. We can re-use the same partition file.

To speed up the analysis, we will perform two steps. The first step is the same as the run above to infer an unrooted tree using reversible models:

```
iqtree2 -s bovidae.phy -p bovidae.nex -B 1000 -T AUTO --prefix
   rev_dna
```

This run will also write the best partitioning scheme to `rev_dna.best_scheme.nex` file. In the second step, we will re-use this best scheme but replace the substitution model with the most general non-reversible DNA model, 12.12 or UNREST (see this doc) to obtain a rooted tree:

```
iqtree2 -s bovidae.phy -p rev_dna.best_scheme.nex --model-joint
   12.12 -B 1000 -T AUTO --prefix nonrev_dna
```

The option `--model-joint 12.12` tells IQ-TREE use a linked substitution model 12.12 across all partitions. This is to avoid potential over-parameterization as this is very parameter-rich model with 12 parameters.

The resulting tree extracted from .iqtree file might look like this:

```
NOTE: Tree is ROOTED at virtual root '__root__'
Numbers in parentheses are  ultrafast bootstrap support (%)

       +---Yak
+------| (72)
|      |                                        +----------Goat
|      |                                   +---| (100)
|      |                                   |   +----------Sheep
|      +-----------------------------------| (95)
```

```
|                                                    +-------------
   Tibetan_antelope
|
+**Cow
|
+**__root__
```

(You can better visualize the .treefile in a tree viewer software).

This run will write an additional tree file `nonrev_dna.rootstrap.nex` with *rootstrap* support values (see box below for definition) annotated on every branch of the tree. If you open this file in FigTree it may look like this (click on "Branch Labels" and choose `rootstrap` for "Display" as shown in the figure):

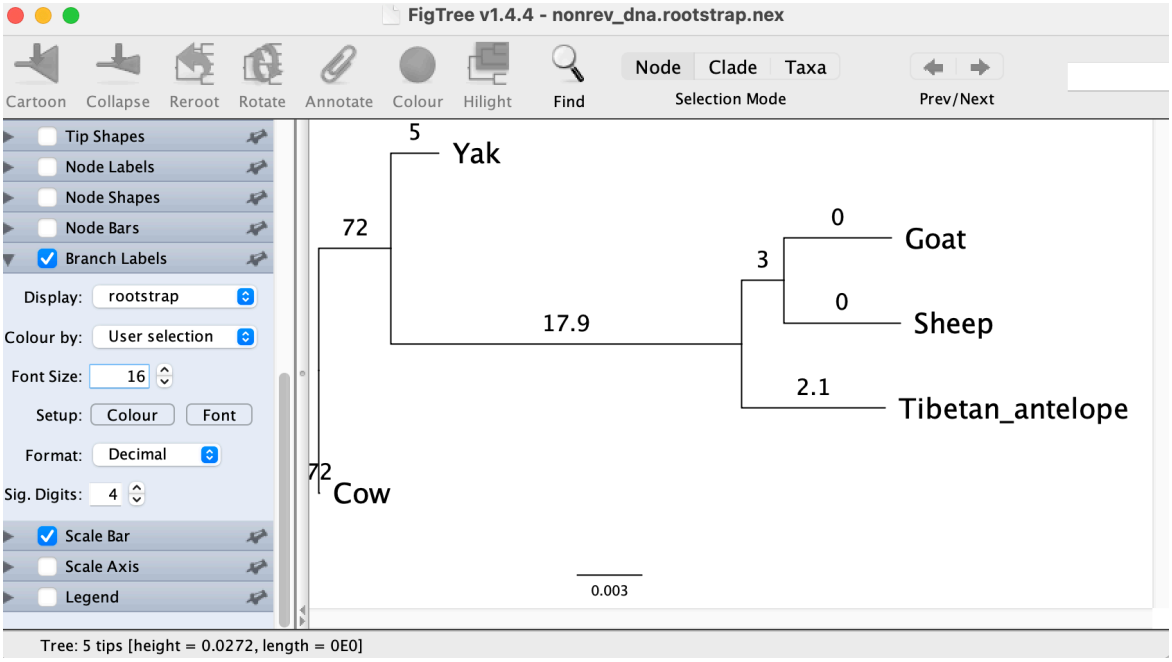


Figure 9.1: Rooted tree with rootstrap supports for DNA

It shows that the tree might be rooted in the branch leading to `Cow` with a rootstrap support of 72%, which is rather low. The 2nd best branch separating Cow and Yak from the rest has a rootstrap support of 17.9%. So with this dataset the DNA model cannot reliably tell where the root position is, but at least provides some candidates.

Rootstrap: To compute rootstrap supports, we conduct a bootstrap analysis to obtain a number of rooted bootstrap trees using non-reversible models. We define the rootstrap support for each branch in the maximum likelihood (ML) tree, as the proportion of rooted bootstrap trees that have the root on that branch. The rootstrap support values are computed for all the branches including external branches. The sum of the rootstrap support values along the tree are always smaller than or equal to one. A sum that is smaller than one can occur when one or more bootstrap replicates are rooted on a branch that does not occur in the ML tree.

We will now try the amino-acid model to see if that helps. We again use `-st NT2AA` option to conveniently perform this analysis:

```
# step 1: infer unrooted tree with reversible models
iqtree2 -s bovidae.phy -p bovidae.nex -B 1000 -T AUTO -st NT2AA
   --prefix rev_aa

# step 2: infer rooted tree with linked non-reversible models
iqtree2 -s bovidae.phy -p rev_aa.best_scheme.nex --model-joint
   NONREV -B 1000 -T AUTO -st NT2AA --prefix nonrev_aa
```

The option `--model-joint NONREV` tells IQ-TREE to use the most general amino-acid model NONREV and to link the NONREV model parameters across all partitions: NONREV has 379 parameters and linking them across partitions will avoid over-parameterization. The tree extracted from `nonrev_aa.iqtree` file now may look like:

```
NOTE: Tree is ROOTED at virtual root '__root__'
Numbers in parentheses are  ultrafast bootstrap support (%)

                                  +------------Yak
+---------------------------------| (100)
|                                 +----------------Cow
|
|                                    +--------------------Goat
|                           +--------| (100)
|                           |        +----------------------Sheep
+---------------------------| (100)
|                           +------------------------------
   Tibetan_antelope
|
+**__root__
```

Interestingly, the amino-acid model suggests a different root position compared with the DNA model. But this position agrees with the outgroup rooting approach. And the tree `nonrev_aa.rootstrap.nex` with rootstrap supports look like:

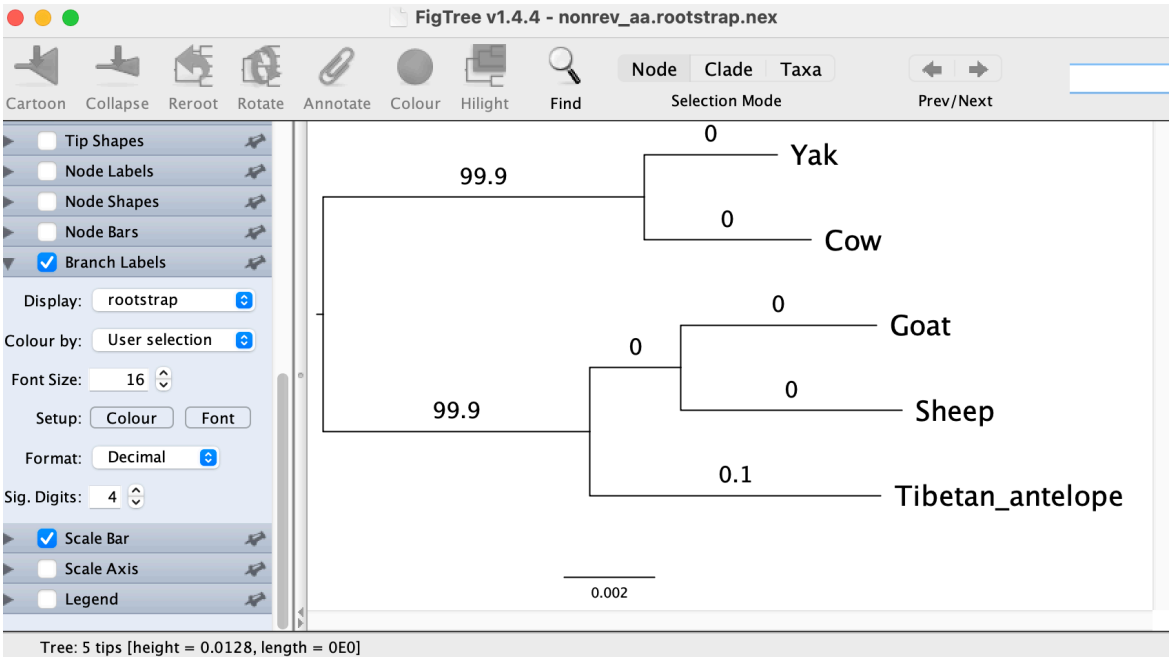


Figure 9.2: Rooted tree with rootstrap supports using amino-acid nonreversible model

That means, the branch separating Yak and Cow from the rest receives a very high rootstrap support of 99.9%. Therefore, the amino-acid model seems to have a much higher power to detect the root, compared with the DNA model.

9.3 Testing root positions

The rootstrap introduced above is one way to measure our confidence in the root placement, but it is not a statistical test. Alternatively, we can apply the tree topology tests to compare the log-likelihoods of the trees being rooted on every branch of the ML tree. IQ-TREE v2.1.3 provides a convenient option `--root-test` that will re-root the tree on every branch and perform the test for you. So you can run:

```
iqtree2 -s bovidae.phy -p rev_aa.best_scheme.nex --model-joint
   NONREV -st NT2AA --root-test -zb 1000 -au -te nonrev_aa.
   treefile --prefix nonrev_aa_test
```

`-zb 1000 -au` is to perform several tree topology tests including the approximately-unbiased (AU) test for the tree found above (`-te nonrev_aa.treefile`). This run will write a file `nonrev_aa_test.roottest.csv` which might look like:

```
# Test results for rooting positions on every branch
# This file can be read in MS Excel or in R with command:
#    dat=read.csv('nonrev_aa_test.roottest.csv',comment.char='#')
# Columns are comma-separated with following meanings:
#    ID:      Branch ID
#    logL:    Log-likelihood of the tree rooted at this branch
#    deltaL:  logL difference from the maximal logl
#    bp-RELL: bootstrap proportion using RELL method (Kishino et
   al. 1990)
#    p-KH:    p-value of one sided Kishino-Hasegawa test (1989)
#    p-SH:    p-value of Shimodaira-Hasegawa test (2000)
#    c-ELW:   Expected Likelihood Weight (Strimmer & Rambaut
   2002)
#    p-AU:    p-value of approximately unbiased (AU) test (
   Shimodaira, 2002)
ID,logL,deltaL,bp-RELL,p-KH,p-SH,c-ELW,p-AU
1,-90388.66044,0,0.983,0.96,1,0.9695602131,0.9975595105
8,-90401.6833,13.02286164,0.005,0.04,0.19,0.01262108065,0.00558089101

5,-90401.68371,13.0232665,0.01,0.04,0.19,0.01262245766,0.006374455939

3,-90410.10499,21.44455589,0.002,0.016,0.104,0.002397842346,0.001014359781

2,-90410.1084,21.44796542,0,0.016,0.104,0.002389013519,0.0008999725939

6,-90413.04441,24.38397245,0,0.005,0.059,0.0002047272296,0.0004975092439

7,-90413.04797,24.38753181,0,0.005,0.059,0.0002046654974,0.0005061888325
```

The branches are sorted by log-likelihoods in descending order. The last column (p-AU) shows the p-values of the AU test. The branch ID 1 has an AU p-value of 0.9975595105, whereas all other branches has p-values < 0.01. To associate branch ID you can return to the FigTree window for `nonrev_aa.rootstrap.nex` file and select "Display" to "id" in the "Branch Labels" tab.

The conclusion from this analysis: we can reject all rooting positions on branches other than branch ID 1, which agrees with the rootstrap measure.

TIP: These options `--root-test -zb 1000 -au` can be combined with the rootstrap

run in the previous section to calculate the rootstrap support values and the rooting test p-values in one single analysis.

Chapter 10

Simulating sequence alignments

Sequence simulators play an important role in phylogenetics. Simulated data has many applications, such as evaluating the performance of different methods, hypothesis testing with parametric bootstraps, and, more recently, generating data for training machine-learning applications. Many sequence simulation programs exist, but the most feature-rich programs tend to be rather slow, and the fastest programs tend to be feature-poor. Here, we introduce AliSim, a new tool that can efficiently simulate biologically realistic alignments under a large range of complex evolutionary models. To achieve high performance across a wide range of simulation conditions, AliSim implements an adaptive approach that combines the commonly-used rate matrix and probability matrix approach. AliSim takes 1.3 hours and 1.3 GB RAM to simulate alignments with one million sequences or sites, while popular software Seq-Gen, Dawg, and INDELible require two to five hours and 50 to 500 GB of RAM.

To use AliSim please make sure that you download the IQ-TREE version 2.2.0 or later.

If you use AliSim please cite the following paper(s):

- Nhan Ly-Trong, Suha Naser-Khdour, Robert Lanfear, Bui Quang Minh, AliSim: A Fast and Versatile Phylogenetic Sequence Simulator for the Genomic Era, Molecular Biology and Evolution, Volume 39, Issue 5, May 2022, msac092, https://doi.org/10.1093/molbev/msac092

- Nhan Ly-Trong, Giuseppe M.J. Barca, Bui Quang Minh (2023) AliSim-HPC: parallel sequence simulator for phylogenetics. https://doi.org/10.1101/2023.01.15.524158 (*for the parallel version*)

10.1 Simulating an alignment from a tree and model

Similar to other software, AliSim can simulate a multiple sequence alignment from a given tree with branch lengths and a model with:

```
iqtree2 --alisim <OUTPUT_PREFIX> -m <MODEL> -t <TREEFILE>
```

The `-m` option specifies a model, and `-t` option specifies a tree file in the standard Newick format. This will print the output alignment into `OUTPUT_PREFIX.phy` file in Phylip format.

For example, if you want to simulate a DNA alignment under the Jukes-Cantor model for the following tree `tree.nwk`:

```
(A:0.3544,(B:0.1905,C:0.1328):0.0998,D:0.0898);
```

You can run IQ-TREE with:

```
iqtree2 --alisim alignment -m JC -t tree.nwk
```

this will print the simulated alignment to `alignment.phy`.

The output MSA should contain 4 sequences of 1000 sites, each, for example:

```
4 1000
A        AAATTTGGTCCTGTGATTCAGCAGTGAT...
B        CTTCCACACCCCAGGACTCAGCAGTGAT...
C        CTACCACACCCCAGGACTCAGCAGTAAT...
D        CTACCACACCCCAGGAAACAGCAGTGAT...
```

Importantly, we note that AliSim uses a random number seed corresponding to the current CPU clock of the running computer. If you run two AliSim commands at the same time, it may generate two identical alignments, which may not be the desired outcome. In that case, you can use -seed option to specify the random number seed:

```
iqtree2 --alisim alignment_123 -m JC -t tree.nwk -seed 123
```

`-seed` option has another advantage of reproducing the same alignment when rerunning IQ-TREE.

NOTE: AliSim fully supports multifurcating input trees, e.g., `(A:0.3544,(B:0.1905,C:0.1328,D:0.0898):0.05,E:0.1);`

10.2 Simulating other datatypes

Apart from the DNA data, AliSim can also simulate other types of data under amino-acid, codon, binary, and multi-state morphological models.

10.2.1 Amino-acid models

AliSim supports all common empirical amino-acid models. For example, to simulate an alignment under the LG model:

```
iqtree2 --alisim alignment_aa -m LG -t tree.nwk
```

10.2.2 Codon models

AliSim offers several codon models. For example:

```
iqtree2 --alisim alignment_codon -m MG{2.0}+F1X4{0.2/0.3/0.4/0.1}
    -t tree.nwk
```

This simulates an alignment under MG model with Nonsynonymous/synonymous (dn/ds) rate ratio of 2.0 and unequal nucleotide frequencies (0.2,0.3,0.4,0.1 for nucleotide A, C, G, T, respectively) but equal nucleotide frequencies over three codon positions.

10.2.3 Binary and morphological models

AliSim supports some binary and morphological models. For example:

```
iqtree2 --alisim alignment_bin -m JC2 -t tree.nwk
```

will simulate a binary alignment under Jukes-Cantor-type binary model.

To simulate morphological alignments, users should specify the number of states with `-st MORPH{<NUM_STATES>}` option:

```
iqtree2 --alisim alignment_morph -m MK -t tree.nwk -st MORPH{20}
```

This simulates a morphological alignment (with 20 states) under MK model.

AliSim also supports An ascertainment bias correction (`+ASC`) model (Lewis, 2001) to simulate sequences without constant sites, for example:

```
iqtree2 --alisim alignment_morph_asc -m MK+ASC -t tree.nwk -st
   MORPH{20}
```

10.3 Non-reversible models

Apart from the standard reversible models, AliSim also provides non-reversible models such as Lie Markov DNA models, for DNA and NONREV for amino-acid.

As an example, to simulate an alignment under the 12.12 model (equivalent to UNREST (unrestricted model)):

```
iqtree2 --alisim alignment_lie_markov -m
   12.12{0.5/0.6/0.9/0.2/0.1/0.4/0.7/0.8/0.3/0.15/0.65}+F
   {0.1/0.2/0.4/0.3} -t tree.nwk
```

NOTE: Users can specify base frequencies with `+F{...}`. Without this, AliSim randomly generates the state frequencies from empirical distributions (See Specifying model parameters).

10.4 Rate heterogeneity across sites

AliSim supports all common rate heterogeneity across sites models, such as allowing for a proportion of invariable sites, continuous/discrete Gamma distribution rates, Distribution-free (Yang, 1995; Soubrier et al., 2012) rates, for example:

To simulate an alignment with a proportion of invariable sites, users can use `+I{<invar_proportion>}` as follows.

```
iqtree2 --alisim alignment_I -t tree.nwk -m JC+I{0.2}
```

This simulates a new alignment under the Jukes-Cantor model with 20% of sites being invariant.

To simulate a new alignment with rate heterogeneity across sites under continuous Gamma distribution, users can specify `+GC{<shape>}` where `<shape>` is the Gamma shape parameter like the following example.

```
iqtree2 --alisim alignment_GC -t tree.nwk -m JC+GC{0.5}
```

Similarly, to apply a discrete Gamma distribution for rate heterogeneity across sites, users can employ `+Gk{<shape>}` where `k` is the number of rate categories, for example:

```
iqtree2 --alisim alignment_G4 -t tree.nwk -m JC+G4{0.5}
```

This simulates a new alignment with 4 discrete rates based on a Gamma distribution with a Gamma shape of 0.5.

Users can specify the Free-rate model for rate heterogeneity via `+Rk{w1/r1/.../doc/wk/rk}` where `k` is the number of rate categories, `w1, ..., wk` are the weights, and `r1, ..., rk` the rates for each category, for example:

```
iqtree2 --alisim alignment_R3 -t tree.nwk -m JC+R3
   {0.5,1.5,0.2,0.7,0.3,2.0}
```

This specifies three rates of 1.5, 0.7, and 2.0 with the weights of 0.5, 0.2, and 0.3, respectively, for rate heterogeneity.

Note that users can combine the Gamma or Free-rate distribution with the proportion of invariant sites, for example:

```
iqtree2 --alisim alignment_G4 -t tree.nwk -m JC+G4{0.5}+I{0.2}
```

to simulate a new alignment with 20% sites are constant while the other sites evolve under 4 discrete Gamma rates (with a Gamma shape of 0.5).

10.5 Customizing output alignments

AliSim provides a number of options to customize the output such as setting the alignment format, length, compression, and simulating more than one alignment.

Users can use `--length` option to change the length of the root sequence:

```
iqtree2 --alisim alignment_5000 -m JC -t tree.nwk --length 5000
```

will simulate an alignment with 5000 sites. Users could also output the alignment in FASTA format with `--out-format` option:

```
iqtree2 --alisim alignment -m JC -t tree.nwk --out-format fasta
```

will print the alignment to `alignment.fa` file.

To generate multiple alignments, users could use `--num-alignments` option:

```
iqtree2 --alisim alignment -m JC -t tree.nwk --num-alignments 3
```

This will output three alignments into `alignment_1.phy`, `alignment_2.phy`, and `alignment_3.phy`, respectively.

If users want to compress the output file, they could try `-gz` option:

```
iqtree2 --alisim alignment -m JC -t tree.nwk -gz
```

This will compress the output file, but it could take a longer running time.

10.6 Insertion and deletion models

AliSim can also simulate insertions and deletions, for example:

```
iqtree2 --alisim alignment_indel -m JC -t tree.nwk --indel
   0.03,0.1
```

`--indel` option specifies the insertion and deletion rates (separated by a comma) relative to the substitution rates. Here, it means that, on average, we have 3 insertion and 10 deletion events per every 100 substitution events. Apart from the normal output file `alignment_indel.phy`, AliSim also exports an additional file `alignment_indel_withoutgaps.fa` containing sequences without gaps. If not needing the additional output file, one could disable that feature by `--no-export-sequence-wo-gaps`. By default, AliSim assumes that the size of indels follows a Zipfian distribution (as defined in INDELible) with an empirical exponent of 1.7 and a maximum indel sizes of 100 . If wanting to change this distribution, one can use `--indel-size` option:

```
iqtree2 --alisim alignment_indel_size -m JC -t tree.nwk --indel
   0.1,0.05 --indel-size GEO{5},GEO{4}
```

It means that the insertion size follows a Geometric distribution with mean of 5 and variance of 20, whereas deletion size also follows the Geometric distribution but with mean of 4 and variance of 12. *Note that the variance is computed from mean.* Apart from this distribution, AliSim also supports Negative Binomial distribution, Zipfian distribution, and Lavalette distribution as following examples:

```
iqtree2 --alisim alignment_indel_size -m JC -t tree.nwk --indel
   0.1,0.05 --indel-size NB{5/20},POW{1.5/10}
```

To specify a Negative Binomial distribution (with mean of 5 and variance of 20) and a Zipfian distribution (with exponent `a` of 1.5 and `max` of 10) for the insertion size, and deletion size, respectively. Or to specify Lavalette distribution (with parameter `a` of 1.5 and `max` of 10) for both insertion and deletion size, users could use:

```
iqtree2 --alisim alignment_indel_size -m JC -t tree.nwk --indel
   0.1,0.05 --indel-size LAV{1.5/10},LAV{1.5/10}
```

NOTE: When using `--length` option with indel models, the output alignment can be longer due to gaps inserted into the root sequence.

10.7 Specifying model parameters

Apart from the simple Juke-Cantor models with no parameters, AliSim also supports all other more complex models available in IQ-TREE. For example:

```
iqtree2 --alisim alignment_HKY -t tree.nwk -m HKY{2.0}+F
   {0.2/0.3/0.1/0.4}
```

This simulates a new alignment under the HKY model with a transition/transversion ratio of 2 and nucleotide frequencies of 0.2, 0.3, 0.1, 0.4 for A, C, G, T, respectively.

By default, if nucleotide frequencies are neither specified nor possible to be inferred from a user-provided alignment, AliSim will randomly generate these frequencies from empirical distributions as the following example.

```
iqtree2 --alisim alignment_HKY -t tree.nwk -m HKY{2.0}
```

In this case, AliSim would simulate an alignment from the HKY model. The frequencies of base A, C, G, and T, will be randomly generated from empirical distributions, namely, Generalized-logistic, Exponential-normal, Power-log-normal, Exponential-Weibull. These distributions and their parameters were estimated from a large collection of empirical datasets (Naser-Khdour et al. 2021).

Besides, AliSim allows users to simulate alignnments with DNA error model by adding `+E{<Error_Probability>}` into the `<model>` when specifying the model with `-m <model>`. For example:

```
iqtree2 --alisim alignment_HKY_error -t tree.nwk -m HKY{2.0}+F
   {0.2/0.3/0.1/0.4}+E{0.01}
```

This simulates a new alignment under the HKY model as the above example, but with a sequencing error probability of 0.01. That means the nucleotide of 1% sites of the simulated sequences is randomly changed to another nucleotide.

10.7.1 Using user-defined parameter distributions

In addition to five built-in distributions, namely *uniform, Generalized_logistic, Exponential_normal, Power_log_normal, and Exponential_Weibull*, users could define their own lists of numbers, then generate other model parameters from these lists by following these steps. Note that user-defined lists of numbers could be generated from different distributions.

Firstly, generating a set of random numbers for each list, then defining the new lists in a new file (e.g.,`custom_distributions.txt`) as the following example.

```
F_A 0.363799 0.313203 0.277533 0.24235 0.260252
F_B 0.321134 0.299891 0.315519 0.269172 0.258165
F_C 0.287641 0.309442 0.264017 0.23103
F_D 0.200087 0.336534 0.337547 0.325379 0.335034
F_E 0.306336 0.359459 0.249315 0.388073
F_F 0.345694 0.338733 0.305404 0.294181
I_A 0.257679 0.417313 0.290922 0.301826 0.292324 0.33887
I_B 0.179902 0.122071 0.348381 0.33887 0.228999
I_C 0.377297 0.296036 0.044523 0.262098 0.295087
R_A 10.363799 20.313203 10.277533 5.24235 3.26025
R_B 6.321134 0.299891 10.315519 0.269172 04.258165
R_C 10.287641 8.309442 20.264017 03.23103 04.178778
R_D 9.200087 10.336534 30.337547 03.325379 0.335034
R_E 2.306336 4.359459 0.249315 0.388073 04.296979
R_F 4.345694 06.338733 02.305404 02.294181 04.303477
R_G 3.257679 07.417313 03.290922 04.301826 03.292324
N_A -0.363799 -0.313203 -0.277533 0.24235 -0.260252
N_B 0.321134 -0.299891 -01.315519 -0.269172 -0.258165
N_C 0.287641 -0.309442 0.264017 -0.23103 0.178778
```

Each list should be defined in a single line, starting with the list name, followed by random numbers. These numbers should be separated by space. The given file `custom_distributions.txt` defines 8 new lists. Each list could have a different number of random elements.

Secondly, loading these lists and generating a new alignment with random parameters with

```
iqtree2 --alisim alignment_GTR_custom -t tree.nwk -m GTR{1.5/R_A
   /R_B/0.5/R_C}+F{Generalized_logistic/0.3/F_A/0.2}+I{F_D}+G{
   F_C} --distribution custom_distributions.txt
```

In this example, 3 substitution rates of GTR models are randomly drawn from the `R_A,R_B,R_C` lists while the user specifies other rates. Similarly, the frequencies of base A and G are generated from `Generalized_logistic` distribution and the list `F_A` whereas the relative frequencies of base C and T are 0.3 and 0.2. These state frequencies are automatically normalized so that they sum to 1. Furthermore, the Invariant Proportion and the Gamma Shape are drawn from the appropriate lists named `F_D`, and `F_C`, respectively.

Users can also use user-defined lists to randomly generate other parameters (e.g., substitution rates, state frequencies, nonsynonymous/synonymous rate ratio, transition

rate, transversion rate, category weight/proportion) for other kinds of models/data (e.g., Protein, Codon, Binary, Morph, Lie Markov, Heterotachy, and Mixture).

10.8 Mimicking a real alignment

AliSim allows users to simulate alignments that mimic the evolutionary history of a given alignment as the below example:

```
iqtree2 --alisim alignment_mimic -s example.phy
```

- `-s example.phy` is the option to supply the input alignment.

In this example, AliSim internally runs IQ-TREE to infer a phylogenetic tree and the best-fit substitution model (using ModelFinder) with its parameters from the input alignment `example.phy`. After that, AliSim simulates a new alignment with the same length as the original alignment based on the inferred tree and model and copies the gaps from the input alignment `example.phy` to the output alignment `alignment_mimic.phy`. To disable this feature, use `--no-copy-gaps` option.

Additionally, for simulations under a mixture models and/or discrete rate heterogeneity (under Gamma/Free-rate distributions), e.g.

```
iqtree2 --alisim alignment_mimic -s example.phy -m "MIX{GTR
   {2/3/4/5/6}+F{0.2/0.3/0.4/0.1},HKY{2}+F{0.3/0.2/0.1/0.4},JC
   }+G{0.5}"
```

The above mixture consists of three model components. AliSim randomly assigns a model component of the mixture to each site according to the site posterior probability distribution of the mixture. For site-frequency mixture models, AliSim assigns site frequency as the mean frequency of the posterior distribution (Wang et al. 2018) (default). Or the user can use `--site-freq SAMPLING` to sample site-frequencies from the posterior probability distribution of the mixture, or use `--site-freq MODEL` to employ the frequencies specified for each model component.

Similarly, for discrete rate heterogeneity (based on Gamma/Free-rate distributions), AliSim assigns site rate as the mean rate of the posterior distribution (by default). Or the user can use `--site-rate SAMPLING` to sample site-specific rate from the posterior probability distribution of rate categories, or `--site-rate MODEL` to sample site-specific rate from the weight (prior distribution) of rate categories.

NOTE:

- When mimicking an alignment and specifying `--length` option without an insertion-deletion model, the output alignment might be shorter or longer than

the original alignment. If shorter, AliSim will copy the gap patterns from the original alignment from site 1 to the last site index in the output alignment. If longer, AliSim can only copy gaps from site 1 to the last site of the original alignment. All remaining sites until the end of the output alignment won't contain any gaps.

- When mimicking an alignment with an insertion-deletion model, AliSim will set the root sequence length to the original alignment length if `--length` is not specified (otherwise it is equal to `--length` option). Moreover, AliSim will ignore the gaps from the original alignment and generate gaps according to the indel model.

10.9 Simulating along a random tree

AliSim can simulate alignments along a random tree under biologically plausible processes such as Yule-Harding, and Birth-Death with the option `-t RANDOM{<MODEL>/<NUM_TAXA>}` as the following example:

```
# simulate 1000-taxon alignment under Yule-Harding random tree
   model
iqtree2 --alisim alignment_yh -t RANDOM{yh/1000}

# simulate 1000-taxon alignment under Birth-Death model with
   birth rate of 0.1 and death rate of 0.05
iqtree2 --alisim alignment_bd -t RANDOM{bd{0.1/0.05}/1000}
```

- `-t RANDOM{yh/1000}` tells AliSim to generate a random tree with 1000 taxa under the Yule-Harding model, with branch lengths following a exponential distribution with a mean of 0.1.
- `-t RANDOM{bd{0.1/0.05}/1000}`: tells AliSim to generate a random tree with 1000 taxa under the Birth-Death model (with birth rate of 0.1 and death rate of 0.05), with branch lengths following a exponential distribution with a mean of 0.1.

For other model, uses can specify `u`, `cat`, or `bal` for Uniform, Caterpillar, or Balanced model, respectively).

`<NUM_TAXA>` can be a fixed number, or a list `{<NUM_1>/<NUM_2>/.../doc/<NUM_N>}`, or a Uniform distribution `U{<LOWER_BOUND>/<UPPER_BOUND>}` where the number of taxa is randomly generated from the given list or distribution.

In the above examples, AliSim generates `alignment_yh.phy` or `alignment_bd.phy` under the Jukes-Cantor DNA model. If you want to change the model, use -m option as

described above.

For the distribution of branch lengths, users could adjust the minimum, mean and maximum of the exponential distribution via the option `-rlen <MIN_LEN> <MEAN_LEN> <MAX_LEN>`.

Furthermore, users can also randomly generate branch lengths of the phylogenetic tree from a user-defined list (or a built-in distribution, such as *uniform, Generalized_logistic, Exponential_normal, Power_log_normal, and Exponential_Weibull*) with `--branch-distribution` option:

```
iqtree2 --alisim alignment_yh_custom_branch -t RANDOM{yh/1000} --
   branch-distribution F_A --distribution custom_distributions.
   txt
```

In this example, the branch lengths of the random tree are randomly drawn from the user-defined list `F_A`. Besides, if the user supplies a tree file (instead of a random tree), the branch lengths of the user-provided tree will be overridden by the random lengths from the list `F_A`.

10.10 Branch-specific models

AliSim supports branch-specific models, which assign different evolutionary models to individual branches of a tree.

To use branch-specific models, users should specify the models for individual branches with the syntax `[&model=<model>]` in the input tree file, say for example, `input_tree.nwk`:

```
(A:0.1,(B:0.1,C:0.2[&model=HKY]),(D:0.3,E:0.1[&model=GTR
   {0.5/1.7/3.4/2.3/1.9}+F{0.2/0.3/0.4/0.1}+I{0.2}+G{0.5}]):0.2);
```

Then, simulate an alignment by

```
  iqtree2 --alisim alignment_example_1 -t input_tree.nwk -m JC
```

Here, AliSim uses the Juke-Cantor model to simulate an alignment along the input tree. However, the `HKY` with random parameters is used to simulate the sequence of taxon C. Similarly, the `GTR` model with the specified parameters is used to generate the sequence of taxon E.

To mimic heterotachy (rate heterogeneity across branches), users can supply a set of branch-lengths containing `n` lengths corresponding to the `n` categories of the model via `lengths=<length_1>,...,<length_n>`, for example:

```
(A:0.1,(B:0.1,C:0.2[&model=HKY{2.0}*H4,lengths=0.1/0.2/0.15/0.3])
   ,(D:0.3,E:0.1):0.2);
```

Then, simulate an alignment by

```
  iqtree2 --alisim alignment_example_2 -t input_tree.nwk -m JC
```

Here, AliSim simulates a new alignment using the Juke-Cantor model. However, at the branch connecting taxon C to its ancestral node, the GHOST model with 4 categories is used with 4 branch lengths 0.1, 0.2, 0.15, and 0.3 to generate the sequence of taxon C.

Additionally, in a rooted tree, users may want to generate the root sequence with particular state frequencies and then simulate new sequences from that root sequence based on a specific model. To do so, one should supply a rooted tree, then specify a model and state frequencies (with `[&model=<model>,freqs=<freq_0,...,<freq_n>]`) for example:

```
(A:0.1,(B:0.1,C:0.2),(D:0.3,E:0.1):0.2):0.3[&model=GTR,freqs
   =0.2/0.3/0.1/0.4];
```

Then, simulate an alignment by

```
  iqtree2 --alisim alignment_example_3 -t input_tree.nwk -m JC
```

Here, AliSim first generates a random sequence at the root based on the user-specified frequencies (`0.2, 0.3, 0.1, 0.4` for A, C, G, T, respectively). Then, it uses the `GTR` model with random parameters to simulate a sequence for the child node of the root. For the remaining branches AliSim applies the Juke-Cantor model.

10.11 Partition models

AliSim allows users to simulate a multi-locus alignment using a partition model specified in a NEXUS file as described in the partition model tutorial. An example partition file may look like:

```
#nexus
begin sets;
    charset gene_1 = DNA, 1-846;
    charset gene_2 = DNA, 847-1368;
    charset gene_3 = DNA, 1369-2040;
    charset gene_4 = DNA, 2041-2772;
```

```
    charset gene_5 = DNA, 2773-3738;
    charpartition mine = HKY{2}+F{0.2/0.3/0.1/0.4}:gene_1,
        GTR{1.2/0.8/0.5/1.7/1.5}+F{0.1/0.2/0.3/0.4}+G{0.5}:gene_2
           ,
        JC:gene_3,
        HKY{1.5}+I{0.2}:gene_4,
        K80{2.5}:gene_5;
end;
```

This means that we define an alignment with 5 genes (partitions). The gene positions are described in `charset` command and the models for each gene are specified in `charpartition` command. Moreover, we use the HKY model for gene_1 with transition-transversion ratio of 2 and nucleotide frequencies of 0.2, 0.3, 0.1, 0.4 for A, C, G and T, respectively. See also the custom model section for how to specify model parameters.

Assuming we name this partition file `multi_genes.nex`. Then, you can simulate an alignment consisting of these five genes by

```
iqtree2 --alisim partition_multi_genes  -q multi_genes.nex -t
   tree.nwk
```

That simulation outputs the new alignment into a single file named `partition_multi_genes.phy`.

In the following we will describe scenarios for more complex partition models.

10.11.1 Edge-proportional partition model

The above example simulates a concatenated alignment under edge-equal partition model, i.e., all partitions share the same tree with the same branch lengths. This is not realistic as different genes might have different evolutionary rates. Therefore, users can specify gene-specific tree lengths directly in the `charpartition` command as follows:

```
#nexus
begin sets;
    charset gene_1 = DNA, 1-846;
    charset gene_2 = DNA, 847-1368;
    charset gene_3 = DNA, 1369-2040;
    charset gene_4 = DNA, 2041-2772;
    charset gene_5 = DNA, 2773-3738;
    charpartition mine = HKY{2}+F{0.2/0.3/0.1/0.4}:gene_1
       {0.26019},
```

```
        GTR{1.2/0.8/0.5/1.7/1.5}+F{0.1/0.2/0.3/0.4}+G{0.5}:gene_2
           {1.51542},
        JC:gene_3{1.03066},
        HKY{1.5}+I{0.2}:gene_4{0.489315},
        K80{2.5}:gene_5{0.680204};
end;
```

Meaning that gene_1 will rescale the branch lengths such that the total tree length becomes 0.26019. Note that `<tree_length>` of a partition is equal to the length of the input tree times the `partition_rate` of that partition. For example, assuming the length of the input tree `tree.nwk` is 0.8673, then the rate of gene_1 is 0.26019/0.8673 = 0.3.

After changing the `multi_genes.nex` file, one could start the simulation by:

```
iqtree2 --alisim partition -p multi_genes.nex -t tree.nwk
```

Note that we use `-p` option here instead of `-q` option like above. If users still used `-q` option, the partition-specific rates will be ignored, i.e., AliSim will use edge-equal partition model.

10.11.2 Topology-unlinked partition model

AliSim supports topology-unlinked partition models, which allow each partition to have its own tree topology and branch lengths. The partition trees can have non-overlapping taxon sets. To do so, users need to prepare a tree file containing multiple NEWICK strings, one for each partition, for example:

```
(A:0.1,(B:0.26,C:0.15):0.1,D:0.05);
(A:0.2,B:0.1,C:0.3);
(A:0.05,B:0.03,(C:0.21,D:0.22):0.21);
((A:0.24,B:0.14):0.19,C:0.07,D:0.1);
((A:0.1,C:0.1):0.04,B:0.4,D:0.5);
```

Note that the 2nd tree does not contain all taxa.

Assuming that this file is named `multi_trees.nwk`, you can simulate an alignment consisting of these five genes from multiple gene trees by

```
iqtree2 --alisim multi_alignment  -Q multi_genes.nex -t
   multi_trees.nwk
```

That simulation outputs the new alignment containing all four taxa A, B, C, D into a single file named `multi_alignment.phy`. AliSim will add a stretch of gaps corresponding to the missing taxon D in partition `gene_2`.

NOTE: We use `-Q` option here to specify topology-unlinked model. If users specify `-q` option, the behaviour will be completely different: AliSim will only load the first tree in `multi_trees.nwk` and simulate an alignment under this one tree.

10.11.3 Mixing different datatypes

AliSim allows users to simulate mixed data (e.g., DNA, Protein, and MORPH) in a single simulation, in which each kind of data is exported into a different alignment file. Here is an example for mixing DNA, protein, and morphological data. Firstly, users need to specify partitions in an input partition file as following.

```
#nexus
begin sets;
    charset part1 = DNA, 1-200\2 201-300;
    charset part2 = DNA, 2-200\2;
    charset part3 = MORPH{6}, 1-300;
    charset part4 = AA, 1-200;
    charset part5 = MORPH{30}, 1-200\2;
    charset part6 = MORPH{30}, 2-200\2;
    charset part7 = DNA, 301-500;
    charpartition mine = HKY{2.0}:part1, JC+G{0.5}:part2, MK:
       part3, Dayhoff:part4, MK:part5, ORDERED:part6, F81+F
       {0.1/0.2/0.3/0.4}:part7;
end;
```

Here, `part1`, `part2`, and `part7` contain three DNA sub-alignments, whereas `part3` , `part5`, and `part6` contain sub-alignments for morphological data. Besides, `part4` contains an amino-acid alignment with 200 sites.

Assuming that the above partition file is named `example_mix.nex` and one would like to simulate alignments from a single tree in `tree.nwk`, one could start the simulation with the following command:

```
iqtree2 --alisim partition_mix -q example_mix.nex -t tree.nwk
```

At the end of the run, AliSim writes out the simulated alignments into four output files. The first file named `partition_mix_DNA.phy` stores the merged 400-site DNA alignment from `part1`, `part2`, and `part7`. Although `part3`, `part5`, and `part6` contain morphological data, `part3` simulates a morphological alignment with 6 states while

part5 and part6 have 30 states. Thus, AliSim outputs the alignment of part3 into partition_mix_MORPH6.phy, whereas partition_mix_MORPH30.phy stores the alignment merging part5 and part6. Lastly, partition_mix_AA.phy stores the simulated amino-acid alignment of part4.

10.12 Mixture models

AliSim allows users to simulate alignments under protein mixture models for example:

```
iqtree2 --alisim alignment_mix_C10 -m C10 -t tree.nwk
```

to simulate a new alignment under the C10 mixture model.

Besides, users can simulate alignments from user-defined mixture model via MIX{<model_1>,...,<model_n>} as described in Mixture models. The following example simulates an alignment under a mixture model contains 2 model components (JC, and HKY) with rate heterogeneity across sites based on discrete Gamma distribution.

```
iqtree2 --alisim alignment_mix_JC_HKY_G -m "MIX{JC,HKY{2.0}+F
   {0.2/0.4/0.1/0.3}}+G4" -t tree.nwk
```

To simulate alignments with more complex mixture models, users can define a new mixture via a model definition file and supply it to AliSim via -mdef <model_file>. For more detail about how to define a mixture model, please have a look at Profile Mixture Models.

10.13 Heterotachy GHOST model

If one wants to simulate sequences based on a GHOST model with 4 categories in conjunction with the GTR model of DNA evolution, one should first specify a multi-length tree as follows.

```
(A[0.067/0.151/0.562/1.269],(B[0.001/0.078/0.319/1.724],C
   [0.076/0.101/0.002/1.061])[0.043/0.086/0.003/0.002],D
   [0.002/0.136/0.002/0.001]);
```

In the above file, each branch should have 4 lengths (corresponding to 4 categories of the GHOST model), which are specified in a pair of square brackets [...], and separated by a slash /.

Assuming that the above tree file is named ghost_tree.nwk, one can simulate an alignment under GHOST model with:

```
iqtree2 --alisim alignment_ghost -m "GTR{2/3/4/5/6}+F
   {0.2/0.3/0.1/0.4}+H4{0.15/0.2/0.35/0.3}" -t ghost_tree.nwk
```

Here, AliSim applies the GHOST model with the weights of 0.15, 0.2, 0.35, 0.3 for the four categories, respectively. If the weights are ignored, AliSim will assume uniform weight distribution.

If you want to unlink GTR parameters so that AliSim could use a GTR model (with specific substitution rates) for each category, you can use `MIX{...}*H4` and specify the model parameters for each categories inside `MIX{...}` as follow:

```
iqtree2 --alisim alignment_ghost_unlink -m "MIX{GTR{2/3/4/5/6},
   GTR{3/4/5/6/7},GTR{4/5/6/7/8},GTR{5/6/7/8/9}}+F
   {0.2/0.3/0.1/0.4}*H4{0.15/0.2/0.35/0.3}" -t ghost_tree.nwk
```

You can also specify a different set of state frequencies for each model component as follow:

```
iqtree2 --alisim alignment_ghost_unlink_freqs -m "MIX{GTR
   {2/3/4/5/6}+F{0.2/0.3/0.4/0.1},GTR{3/4/5/6/7}+F
   {0.3/0.2/0.4/0.1},GTR{4/5/6/7/8}+F{0.4/0.2/0.3/0.1},GTR
   {5/6/7/8/9}+F{0.1/0.2/0.4/0.3}}*H4{0.15/0.2/0.35/0.3}" -t
   ghost_tree.nwk
```

Besides, assuming that we have an input alignment `example.phy` evolving under the GHOST model with 4 categories in conjunction with the `GTR` model. If one wants to simulate an alignment that mimics that input alignment, one should use the following command:

```
iqtree2 --alisim alignment_ghost_mimick -m GTR+H4 -s example.phy
```

or using `GTR*H4` instead of `GTR+H4`, if you want to unlink GTR parameters:

```
iqtree2 --alisim alignment_ghost_unlink_mimick -m GTR*H4 -s
   example.phy
```

or using `GTR+FO*H4` to unlink GTR parameters and state frequencies:

```
iqtree2 --alisim alignment_ghost_unlink_freqs_mimick -m GTR+FO*H4
    -s example.phy
```

10.14 Functional divergence model

AliSim supports the FunDi model, which allows a proportion number of sites (`<RHO>`) in the sequence of each taxon in a given list (`<TAXON_1>,...,<TAXON_N>`), could be permuted with each other. To simulate new alignments under the FunDi model, one could use `--fundi` option:

```
iqtree2 --alisim alignment_fundi -t tree.nwk -m JC --fundi A,C
   ,0.1
```

This example simulates a new alignment under the Juke-Cantor model from the input tree `tree.nwk` with the default sequence length of 1,000 sites. Since the user specifies FunDi model with `<RHO>` = 0.1, thus, in the sequences of Taxon A, and C, 100 random sites (sequence length * `<RHO>` = 1,000 * 0.1) are permuted with each other.

10.15 Pre-define mutations

Starting from IQ-TREE v2.2.3, AliSim allows users to enfore pre-defined mutations that must occur at some specific nodes of the tree. Those mutations could be, for example, generated by running VGsim with the option `-writeMutations`.

Given a tree file `tree_example.nwk`

```
(T1:0.2,(T2:0.3,T4:0.1)Node5:0.4,T3:0.1);
```

and a mutations file `mutations.txt` like below:

```
Node5    C39G,T17A,G25C
T2       C25A,A5G
```

Each line starts with a taxon name or an internal node name, followed by whitespace(s), and a comma-separated list of mutations. Each mutation is denoted by a character state at the parent node, followed by a position number **starting from index 0** and the character state to be fixed at the current node. For the above example,

- AliSim will enforce the 40th, 18th, and 26th positions of the sequence at internal node `Node5` to be `G`, `A` and `C`.
- The 26th and 6th positions of the sequence at the taxon `T2` to be `A` and `G`, respectively.

NOTE: Site index starts from 0 to make AliSim compatible with VGsim's output). If you want to start from 1, use the option `--index-from-one`.

The following command

```
iqtree2 --alisim example_mutations  -t tree_example.nwk -m JC --
   mutation mutations.txt
```

will simulate an alignment with 4 sequences (i.e., T1, T2, T3, and T4) under the Jukes-Cantor model and the above mutation rule.

10.16 Parallel sequence simulations

AliSim supports simulating many large alignments in parallel with OpenMP and/or MPI. To simulate large alignment(s) with OpenMP, one can use `-nt` option to specify the number of threads:

```
iqtree2 --alisim large_alignment -t tree.nwk --length 1000000 -m
   JC -nt 4
```

This example simulates a new alignment under the Juke-Cantor model from the input tree `tree.nwk` with the sequence length of 1,000,0000 sites using 4 threads. For multi-threading simulations, AliSim supports two algorithms AliSim-OpenMP-IM (default) and AliSim-OpenMP-EM (please see AliSim-HPC). Users can specify `--openmp-alg EM` if they want to employ the AliSim-OpenMP-EM algorithm.

NOTES:

- The performance of AliSim-OpenMP-IM is affected by a memory limit factor (=0.2 (by default) and can be set in the range (0 to 1]): a small factor will potentially increase the runtime; a large factor will increase the memory consumption. To specify this memory limit factor, one can use `--mem-limit <FACTOR>` option.
- If using AliSim-OpenMP-EM algorithm, the simulated sequences will be written in an arbitrary order to the alignment (which is not a matter in most phylogenetic software). However, if users want to maintain the sequence order (based on the preorder traversal of the tree), they can use `--keep-seq-order` option, but it will sacrifice a certain runtime.
- If using AliSim-OpenMP-EM algorithm, one can use `--no-merge` to skip the concatenation step to save the runtime. Note that, when simulating an alignment of length L with K threads, AliSim will output the alignment as K sub-alignment files of L/K sites.

To simulate many alignments, one can use the MPI version of AliSim:

```
mpirun -np 10 iqtree2-mpi --alisim many_alignment -t tree.nwk -m
   JC --num-alignments 100
```

This example uses 10 MPI processes to simulate 100 alignments under the Juke-Cantor model from the input tree `tree.nwk` with the default sequence length of 1,000 sites. Note that AliSim-MPI version can run on a distributed-memory system with many nodes and multiple CPUs per node. To maximize the parallel efficiency, we recommend users specify the number of processes as a divisor of the number of alignments.

To simulate many large alignments, users can employ both MPI and OpenMP on a high-performance computing system:

```
mpirun -np 10 --map-by node:PE=4 --rank-by core iqtree2-mpi --
   alisim many_large_alignment -t tree.nwk --length 1000000 -m JC
    --num-alignments 100 -nt 4
```

This example uses 10 MPI processes, each having 4 threads (i.e. a total of 40 threads will be run) to simulate 100 large alignments under the Juke-Cantor model from the input tree `tree.nwk` with the sequence length of 1,000,000 sites.

NOTES: Our MPI implementation supports Indels as the original version of AliSim, while the OpenMP algorithm does not. Therefore, one can employ only MPI to simulate many alignments with Indels.

10.17 Command reference

All the options available in AliSim are shown below:

Option	Usage and meaning
`--alisim <OUTPUT_ALIGNMENT>`	Activate AliSim and specify the output alignment filename.
`-t <TREE_FILE>`	Set the input tree file name.
`--seqtype <SEQUENCE_TYPE>`	Specify the sequence type (BIN, DNA, AA, CODON, MORPH{`NUMBER_STATES`}, where `NUMBER_STATES` is the number of states for morphological model). *By default, Alisim automatically detects the sequence type from the model name.*
`-m <MODEL>`	Specify the model name. See Substitution Models and Complex Models for the list of supported models.
`-mdef <MODEL_FILE>`	Name of a NEXUS model file to define new models, which can be used with `-m` option.

Option	Usage and meaning
`--fundi <TAXON_1>,...,<TAXON_N>,<RHO>`	Specify the FunDi model (Gaston et al. 2011). The last number `RHO` in this list is the proportion of sites, that will be randomly permuted in the sequences of the given taxa. The same permutation is applied to the sequences.
`--indel <INS>,<DEL>`	Set the insertion and deletion rate of the indel model, relative to the substitution rate.
`--indel-size <INS_DIS>,<DEL_DIS>`	Set the insertion and deletion size distributions. By default, AliSim uses `POW{1.7/100}` for a power-law (Zipfian) distribution with parameter `a` of 1.7 and maximum indel size of 100.
`--no-unaligned`	Do not output a file of unaligned sequences when using indel models. Default: a file `.unaligned.fa` containing unaligned sequences is written.
`-q <PARTITION>` or `-p <PARTITION>` or `-Q <PARTITION>`	Specify different types of Partition models
`--distribution <FILE>`	Supply a definition file of distributions, which could be used to generate random model parameters (see Using user-defined parameter distributions).
`--branch-distribution <DISTRIBUTION>`	Specify a distribution, from which branch lengths of the input trees are randomly generated and overridden.
`--branch-scale <SCALE>`	Specify a value to scale all branch lengths of the input tree.
`--length <LENGTH>`	Set the root sequence length.*Default: 1,000*
`--num-alignments <NUMBER>`	Set the number of output datasets.*Default: 1*

Option	Usage and meaning
`--root-seq <ALN_FILE>,<SEQ_NAME>`	Specify the root sequence from an alignment.AliSim automatically sets the output sequence length (`--length`) equally to the length of the root sequence.
`-t RANDOM{<MODEL>/<NUM_TAXA>}`	Specify the model and the number of taxa to generate a random tree (see Simulating along a random tree.
`-rlen <MIN_LEN> <MEAN_LEN> <MAX_LEN>`	Specify three numbers: minimum, mean and maximum branch lengths when generating a random tree with `-t RANDOM{<MODEL>/<NUM_TAXA>}`. *Default: -rlen 0.001 0.1 0.999.*
`-s <ALIGNMENT>`	Specify an input alignment file in PHYLIP, FASTA, NEXUS, CLUSTAL or MSF format.
`--no-copy-gaps`	Disable copying gaps from the input alignment.
`--site-freq <OPTION>`	Specify the option (`MEAN` *(default)*, or `SAMPLING`, or `MODEL`) to mimic the site-frequencies for mixture models from the input alignment (see Mimicking a real alignment).
`--site-rate <OPTION>`	Specify the option (`MEAN` *(default)*, or `SAMPLING`, or `MODEL`) to mimic the discrete rate heterogeneity from the input alignment (see Mimicking a real alignment).
`-nt <NUM_THREADS>`	Specify the number of threads for simulating large alignment(s) with OpenMP.
`--openmp-alg <ALG>`	Specify the multithreading algorithm (`IM` or `EM` for AliSim-OpenMP-IM or AliSim-OpenMP-EM, respectively).*Default: IM*
`--mem-limit <FACTOR>`	Specify the memory limit factor for the AliSim-OpenMP-IM algorithm: 0 < `<FACTOR>` <= 1.*Default: 0.2*
`--keep-seq-order`	Output the sequences (simulated by the AliSim-OpenMP-EM algorithm) following the visiting order of tips (based on the preorder traversal).
`--no-merge`	Skip the concatenation step in the AliSim-OpenMP-EM algorithm, output alignment in multiple sub-alignment files.
`--single-output`	Output all alignments into a single file.
`--write-all`	Enable outputting internal sequences.

Option	Usage and meaning
`-seed <NUMBER>`	Specify the seed number. *Default: the clock of the PC*. Be careful! To make the AliSim reproducible, users should specify the seed number.
`-gz`	Enable output compression. It may take a longer running time.*By default, output compression is disabled.*
`--out-format <FORMAT>`	Set the output format (`fasta`, `phy`, or `maple` for FASTA, PHYLIP, or MAPLE format, respectively).*Default: phy*

Chapter 11

Command reference

Commprehensive documentation of command-line options.

IQ-TREE is invoked from the command-line with e.g.:

```
iqtree -s <alignment> [OPTIONS]
```

assuming that IQ-TREE can be run by simply entering `iqtree`. If not, please change `iqtree` to the actual path of the executable or read the Quick start guide.

11.1 General options

General options are mainly intended for specifying input and output files:

Option	Usage and meaning
`-h` or `-?`	Print help usage.
`-s`	Specify input alignment file in PHYLIP, FASTA, NEXUS, CLUSTAL or MSF format.

Option	Usage and meaning
`-st`	Specify sequence type as either of `DNA`, `AA`, `BIN`, `MORPH`, `CODON` or `NT2AA` for DNA, amino-acid, binary, morphological, codon or DNA-to-AA-translated sequences. This is only necessary if IQ-TREE did not detect the sequence type correctly. Note that `-st CODON` is always necessary when using codon models (otherwise, IQ-TREE applies DNA models) and you also need to specify a genetic code like this if differed from the standard genetic code. `-st NT2AA` tells IQ-TREE to translate protein-coding DNA into AA sequences and then subsequent analysis will work on the AA sequences. You can also use a genetic code like `-st NT2AA5` for the Invertebrate Mitochondrial Code (see genetic code table).
`-t`	Specify a file containing starting tree for tree search. The special option `-t BIONJ` starts tree search from BIONJ tree and `-t RANDOM` starts tree search from completely random tree. *DEFAULT: 100 parsimony trees + BIONJ tree*
`-te`	Like `-t` but fixing user tree. That means, no tree search is performed and IQ-TREE computes the log-likelihood of the fixed user tree.
`-o`	Specify an outgroup taxon name to root the tree. The output tree in `.treefile` will be rooted accordingly. *DEFAULT: first taxon in alignment*
`-pre`	Specify a prefix for all output files. *DEFAULT: either alignment file name (`-s`) or partition file name (`-q`, `-spp` or `-sp`)*
`-nt`	Specify the number of CPU cores for the multicore version. A special option `-nt AUTO` will tell IQ-TREE to automatically determine the best number of cores given the current data and computer.
`-ntmax`	Specify the maximal number of CPU cores `-nt AUTO` is allowed to allocate *DEFAULT: #CPU cores on the current machine*
`-seed`	Specify a random number seed to reproduce a previous run. This is normally used for debugging purposes. *DEFAULT: based on current machine clock*
`-v`	Turn on verbose mode for printing more messages to screen. This is normally used for debugging purposes. *DEFAULT: OFF*
`-quiet`	Silent mode, suppress printing to the screen. Note that `.log` file is still written.
`-keep-ident`	Keep identical sequences in the alignment. Bu default: IQ-TREE will remove them during the analysis and add them in the end.

Option	Usage and meaning
`-safe`	Turn on safe numerical mode to avoid numerical underflow for large data sets with many sequences (typically in the order of thousands). This mode is automatically turned on when having more than 2000 sequences.
`-mem`	Specify maximal RAM usage, for example, `-mem 64G` to use at most 64 GB of RAM. By default, IQ-TREE will try to not to exceed the computer RAM size.

11.1.1 Example usages:

- Reconstruct a maximum-likelihood tree given a sequence alignment file `example.phy`:

```
iqtree -s example.phy
```

- Reconstruct a maximum-likelihood tree using at most 8 GB RAM and automatically detect the number of cores to use:

```
# For version <= 1.5.X
iqtree-omp -s example.phy -nt AUTO -mem 8G

# For version >= 1.6.0, change iqtree-omp to just iqtree
iqtree -s example.phy -nt AUTO -mem 8G
```

- Like above but write all output to `myrun.*` files:

```
# For version <= 1.5.X
iqtree-omp -s example.phy -nt AUTO -mem 8G -pre myrun

# For version <= 1.6.0
iqtree -s example.phy -nt AUTO -mem 8G -pre myrun
```

11.2 Checkpointing to resume stopped run

Starting with version 1.4.0 IQ-TREE supports checkpointing: If an IQ-TREE run was interrupted for some reason, rerunning it with the same command line and input data will automatically resume the analysis from the last stopped point. The previous log file will then be appended. If a run successfully finished, IQ-TREE will deny to rerun and issue an error message. A few options that control checkpointing behavior:

Option	Usage and meaning
`-redo`	Redo the entire analysis no matter if it was stopped or successful. WARNING: This option will overwrite all existing output files.
`-cptime`	Specify the minimum checkpoint time interval in seconds (default: 20s)

NOTE: IQ-TREE writes a checkpoint file with name suffix `.ckp.gz` in gzip format. Do not modify this file. If you delete this file, all checkpointing information will be lost!

11.3 Likelihood mapping analysis

Starting with version 1.4.0, IQ-TREE implements the likelihood mapping approach (Strimmer and von Haeseler, 1997) to assess the phylogenetic information of an input alignment. The detailed results will be printed to `.iqtree` report file. The likelihood mapping plots will be printed to `.lmap.svg` and `.lmap.eps` files.

Compared with the original implementation in TREE-PUZZLE, IQ-TREE is much faster and supports many more substitution models (including partition and mixture models).

Option	Usage and meaning
`-lmap`	Specify the number of quartets to be randomly drawn. If you specify `-lmap ALL`, all unique quartets will be drawn, instead.
`-lmclust`	Specify a NEXUS file containing taxon clusters (see below for example) for quartet mapping analysis.
`-n 0`	Skip subsequent tree search, useful when you only want to assess the phylogenetic information of the alignment.
`-wql`	Write quartet log-likelihoods into `.lmap.quartetlh` file (typically not needed).

TIP: The number of quartets specified via `-lmap` is recommended to be at least 25 times the number of sequences in the alignment, such that each sequence is covered ~100 times in the set of quartets drawn.

An example NEXUS cluster file (where A, B, C, etc. are sequence names):

```
#NEXUS
begin sets;
    taxset Cluster1 = A B C;
    taxset Cluster2 = D E;
    taxset Cluster3 = F G H I;
    taxset Cluster4 = J;
    taxset IGNORED = X;
end;
```

Here, `Cluster1` to `Cluster4` are four user-defined clusters of sequences. Note that users can give any names to the clusters instead of `Cluster1`, etc. If a cluster is named `ignored` or `IGNORED` the sequences included will be ignored in the likelihood mapping analysis.

If you provide a cluster file it has to contain between 1 and 4 clusters (plus an optional `IGNORED` or `ignored` cluster), which will tell IQ-TREE to perform an unclustered (default without a cluster file) or a clustered analysis with 2, 3 or 4 groups, respectively.

The names given to the clusters in the cluster file will be used to label the corners of the triangle diagrams.

11.3.1 Example usages:

- Perform solely likelihood mapping analysis (ignoring tree search) with 2000 quartets for an alignment `example.phy` with model being automatically selected:

  ```
  iqtree -s example.phy -lmap 2000 -n 0 -m TEST
  ```

You can now view the likelihood mapping plot file `example.phy.lmap.svg`, which looks like this:

It shows phylogenetic information of the alignment `example.phy`. On the top: distribution of quartets depicted by dots on the likelihood mapping plot. On the left: the three areas show support for one of the different groupings like (a,b)-(c,d). On the right: quartets falling into the three corners are informative. Those in three rectangles are partly informative and those in the center are uninformative. A good data set should have high number of informative quartets and low number of uninformative quartets. The meanings can also be found in the `LIKELIHOOD MAPPING STATISTICS` section of the report file `example.phy.iqtree`:

```
LIKELIHOOD MAPPING STATISTICS
-----------------------------
```

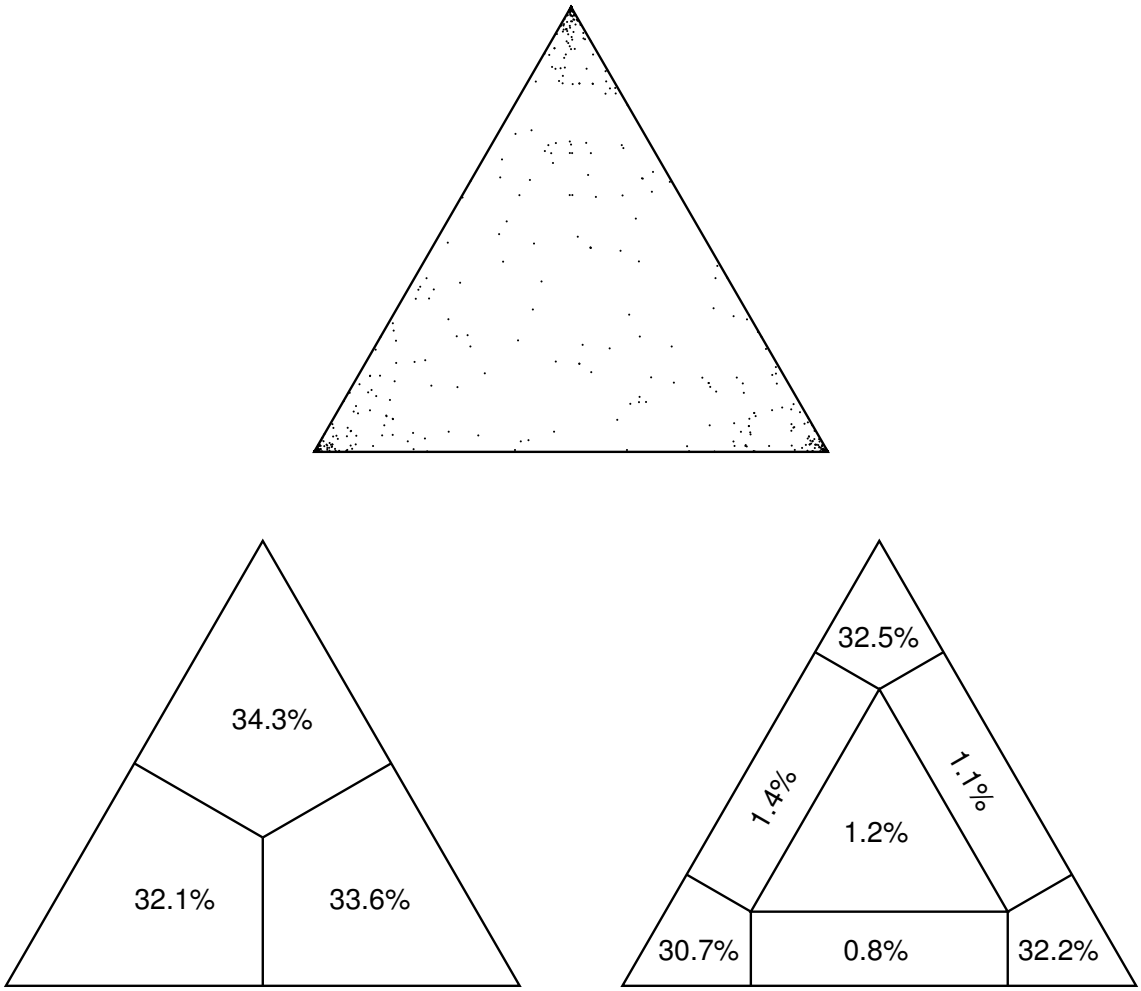
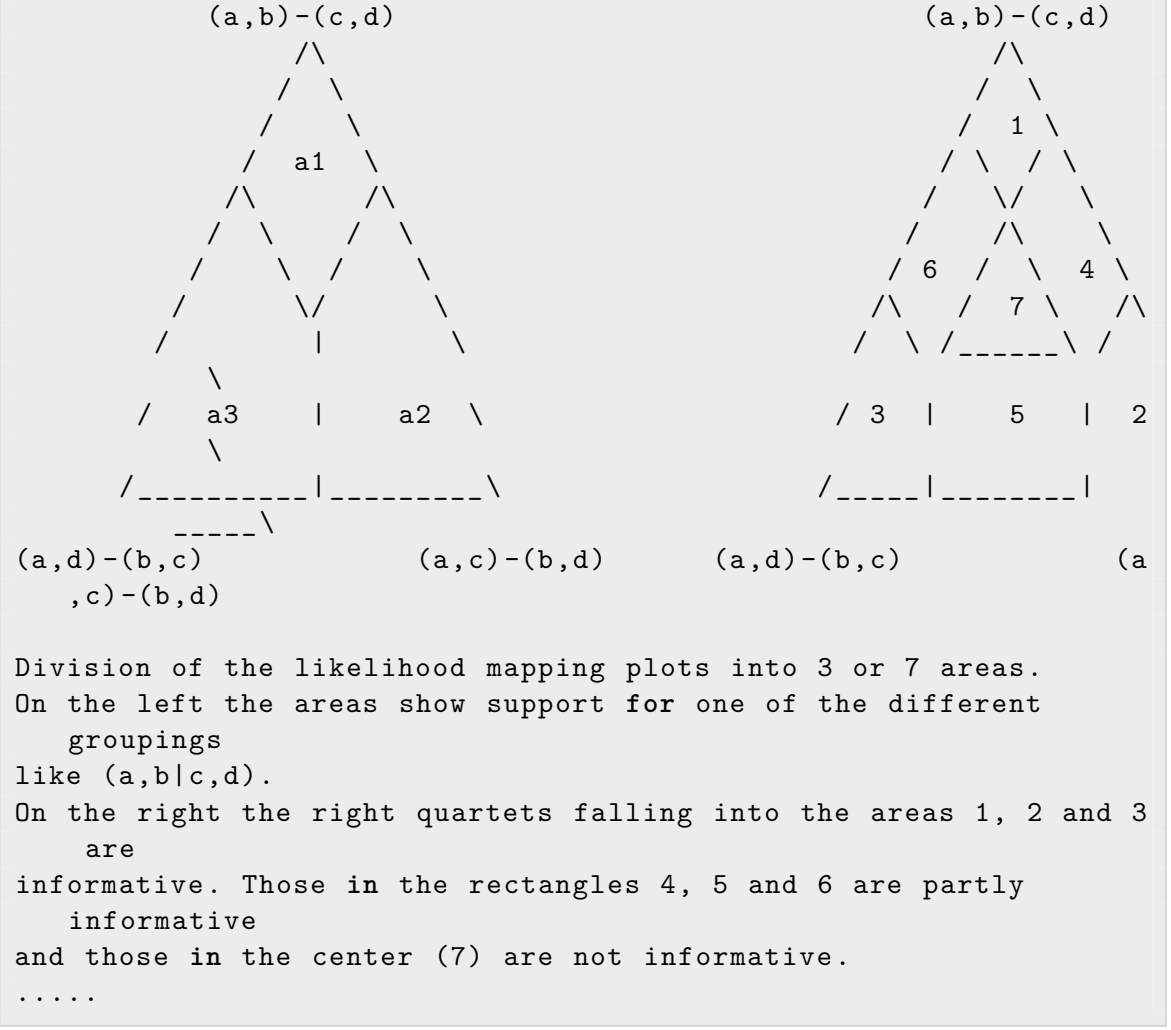


Figure 11.1: Likelihood mapping plot.




```
Division of the likelihood mapping plots into 3 or 7 areas.
On the left the areas show support for one of the different
   groupings
like (a,b|c,d).
On the right the right quartets falling into the areas 1, 2 and 3
    are
informative. Those in the rectangles 4, 5 and 6 are partly
   informative
and those in the center (7) are not informative.
.....
```

11.4 Automatic model selection

The default model (e.g., `HKY+F` for DNA, `LG` for protein data) may not fit well to the data. Therefore, IQ-TREE allows to automatically determine the best-fit model via a series of `-m TEST...` option:

Option	Usage and meaning
`-m TESTONLY`	Perform standard model selection like jModelTest (for DNA) and ProtTest (for protein). Moreover, IQ-TREE also works for codon, binary and morphogical data. If a partition file is specified, IQ-TREE will find the best-fit model for each partition.

Option	Usage and meaning
`-m TEST`	Like `-m TESTONLY` but immediately followed by tree reconstruction using the best-fit model found. So this performs both model selection and tree inference within a single run.
`-m TESTNEWONLY` or `-m MF`	Perform an extended model selection that additionally includes FreeRate model compared with `-m TESTONLY`. *Recommended as replacement for `-m TESTONLY`*. Note that `LG4X` is a FreeRate model, but by default is not included because it is also a protein mixture model. To include it, use `-madd` option (see table below).
`-m TESTNEW` or `-m MFP`	Like `-m MF` but immediately followed by tree reconstruction using the best-fit model found.

TIP: During model section, IQ-TREE will write a model checkpoint file (suffix `.model` in version <= 1.5.X or `.model.gz` in version >= 1.6.X) that stores information of all models tested so far. Thus, if IQ-TREE is interrupted for whatever reason, restarting the run will load this file to reuse the computation.

IQ-TREE version 1.6 or later allows to additionally test Lie Markov DNA models by adding the following keyword to `-m` option:

Option	Usage and meaning
`+LM`	Additionally consider all Lie Markov models
`+LMRY`	Additionally consider all Lie Markov models with RY symmetry
`+LMWS`	Additionally consider all Lie Markov models with WS symmetry
`+LMMK`	Additionally consider all Lie Markov models with MK symmetry
`+LMSS`	Additionally consider all strand-symmetric Lie Markov models

When a partition file is specified then you can append `MERGE` keyword into `-m` option to find the best-fit partitioning scheme like PartitionFinder (Lanfear et al., 2012). More specifically,

Option	Usage and meaning
`-m TESTMERGEONLY`	Select best-fit partitioning scheme by possibly merging partitions to reduce over-parameterization and increase model fit. It implements the greedy algorithm of PartitionFinder.

Option	Usage and meaning
`-m TESTMERGE`	Like `-m TESTMERGEONLY` but immediately followed by tree reconstruction using the best partitioning scheme found.
`-m TESTNEWMERGEONLY` or `-m MF+MERGE`	Like `-m TESTMERGEONLY` but additionally includes FreeRate model.
`-m TESTNEWMERGE` or `-m MFP+MERGE`	Like `-m MF+MERGE` but immediately followed by tree reconstruction using the best partitioning scheme found.
`-rcluster`	Specify the percentage for the relaxed clustering algorithm (Lanfear et al., 2014) to speed up the computation instead of the default slow greedy algorithm. This is similar to `--rcluster-percent` option of PartitionFinder. For example, with `-rcluster 10` only the top 10% partition schemes are considered to save computations.
`-rclusterf`	Similar to `-rcluster` but using the **fast** relaxed clustering algorithm (Lanfear et al., 2017) of PartitionFinder2. Introduced in version 1.6.
`-rcluster-max`	Specify the absolute maximum number of partition pairs in the paritition merging phase. Default: the larger of 1000 and 10 times the number of partitions. This option is similar to `--rcluster-max` option of PartitionFinder2.

WARNING: For versions <= 1.5.X, all commands with `-m ...MERGE...` will always perform an edge-unlinked partition scheme finding even if `-spp` option is used. Only in the next phase of tree reconstruction, then an edge-linked partition model is used. However, for versions 1.6.X onwards, the edge-linked partition finding is performed by `-spp` option.

Several parameters can be set to e.g. reduce computations:

Option	Usage and meaning
`-mset`	Specify the name of a program (`raxml`, `phyml` or `mrbayes`) to restrict to only those models supported by the specified program. Alternatively, one can specify a comma-separated list of base models. For example, `-mset WAG,LG,JTT` will restrict model selection to WAG, LG, and JTT instead of all 18 AA models to save computations.
`-msub`	Specify either `nuclear`, `mitochondrial`, `chloroplast` or `viral` to restrict to those AA models designed for specified source.

Option	Usage and meaning
`-mfreq`	Specify a comma-separated list of frequency types for model selection. *DEFAULT:* `-mfreq FU,F` *for protein models (FU = AA frequencies given by the protein matrix, F = empirical AA frequencies from the data),* `-mfreq ,F1x4,F3x4,F` *for codon models*
`-mrate`	Specify a comma-separated list of rate heterogeneity types for model selection. *DEFAULT:* `-mrate E,I,G,I+G` *for standard procedure,* `-mrate E,I,G,I+G,R` *for new selection procedure.* (E means Equal/homogeneous rate model).
`-cmin`	Specify minimum number of categories for FreeRate model. *DEFAULT: 2*
`-cmax`	Specify maximum number of categories for FreeRate model. It is recommended to increase if alignment is long enough. *DEFAULT: 10*
`-merit`	Specify either `AIC`, `AICc` or `BIC` for the optimality criterion to apply for new procedure. *DEFAULT: all three criteria are considered*
`-mtree`	Turn on full tree search for each model considered, to obtain more accurate result. Only recommended if enough computational resources are available. *DEFAULT: fixed starting tree*
`-mredo`	Ignore model checkpoint file computed earlier. *DEFAULT: model checkpoint file (if exists) is loaded to reuse previous computations*
`-madd`	Specify a comma-separated list of mixture models to additionally consider for model selection. For example, `-madd LG4M,LG4X` to additionally include these two protein mixture models.
`-mdef`	Specify a NEXUS model file to define new models.

NOTE: Some of the above options require a comma-separated list, which should not contain any empty space!

11.4.1 Example usages:

- Select best-fit model for alignment `data.phy` based on Bayesian information criterion (BIC):

```
iqtree -s data.phy -m TESTONLY
```

- Select best-fit model for a protein alignment `prot.phy` using the new testing procedure and only consider WAG, LG and JTT matrix to save time:

```
iqtree -s prot.phy -m MF -mset WAG,LG,JTT
```

- Find the best partitioning scheme for alignment `data.phy` and partition file `partition.nex` with a relaxed clustering at 10% to save time:

```
iqtree -s data.phy -spp partition.nex -m TESTMERGEONLY -
   rcluster 10
```

11.5 Specifying substitution models

`-m` is a powerful option to specify substitution models, state frequency and rate heterogeneity type. The general syntax is:

```
-m MODEL+FreqType+RateType
```

where `MODEL` is a model name, `+FreqType` (optional) is the frequency type and `+RateType` (optional) is the rate heterogeneity type.

The following `MODEL`s are available:

DataType	Model names
DNA	JC/JC69, F81, K2P/K80, HKY/HKY85, TN/TrN/TN93, TNe, K3P/K81, K81u, TPM2, TPM2u, TPM3, TPM3u, TIM, TIMe, TIM2, TIM2e, TIM3, TIM3e, TVM, TVMe, SYM, GTR and 6-digit specification. See DNA models for more details.
Protein	BLOSUM62, cpREV, Dayhoff, DCMut, FLU, HIVb, HIVw, JTT, JTTDCMut, LG, mtART, mtMAM, mtREV, mtZOA, mtMet, mtVer, mtInv, Poisson, PMB, rtREV, VT, WAG.
Protein	Mixture models: C10, ..., C60 (CAT model) (Lartillot and Philippe, 2004), EX2, EX3, EHO, UL2, UL3, EX_EHO, LG4M, LG4X, CF4. See Protein models for more details.
Codon	MG, MGK, MG1KTS, MG1KTV, MG2K, GY, GY1KTS, GY1KTV, GY2K, ECMK07/KOSI07, ECMrest, ECMS05/SCHN05 and combined empirical-mechanistic models. See Codon models for more details.
Binary	JC2, GTR2. See Binary and morphological models for more details.
Morphology	MK, ORDERED. See Binary and morphological models for more details.

The following `FreqType`s are supported:

FreqType	Meaning
`+F`	Empirical state frequency observed from the data.
`+FO`	State frequency optimized by maximum-likelihood from the data. Note that this is with letter-O and not digit-0.
`+FQ`	Equal state frequency.
`+F1x4`	See Codon frequencies.
`+F3x4`	See Codon frequencies.

Further options:

Option	Usage and meaning
`-mwopt`	Turn on optimizing weights of mixture models. Note that for models like `LG+C20+F+G` this mode is automatically turned on, but not for `LG+C20+G`.

11.5.1 Example usages:

- Infer an ML tree for a DNA alignment `dna.phy` under GTR+I+G model:

```
iqtree -s dna.phy -m GTR+I+G
```

- Infer an ML tree for a protein alignment `prot.phy` under LG+F+G model:

```
iqtree -s prot.phy -m LG+F+G
```

- Infer an ML tree for a protein alignment `prot.phy` under profile mixture model LG+C10+F+G:

```
iqtree -s prot.phy -m LG+C10+F+G
```

11.6 Rate heterogeneity

The following `RateTypes` are supported:

RateType	Meaning
`+I`	Allowing for a proportion of invariable sites.
`+G`	Discrete Gamma model (Yang, 1994) with default 4 rate categories. The number of categories can be changed with e.g. `+G8`.

RateType	Meaning
`+I+G`	Invariable site plus discrete Gamma model (Gu et al., 1995).
`+R`	FreeRate model (Yang, 1995; Soubrier et al., 2012) that generalizes `+G` by relaxing the assumption of Gamma-distributed rates. The number of categories can be specified with e.g. `+R6`. *DEFAULT: 4 categories*
`+I+R`	invariable site plus FreeRate model.

See Rate heterogeneity across sites for more details.

Further options:

Option	Usage and meaning
`-a`	Specify the Gamma shape parameter (default: estimate)
`-gmedian`	Perform the *median* approximation for Gamma rate heterogeneity instead of the default *mean* approximation (Yang, 1994)
`-i`	Specify the proportion of invariable sites (default: estimate)
`--opt-gamma-inv`	Perform more thorough estimation for `+I+G` model parameters
`-wsr`	Write per-site rates to `.rate` file

Optionally, one can specify Ascertainment bias correction by appending `+ASC` to the model string. Advanced mixture models can also be specified via `MIX{...}` and `FMIX{...}` syntax. Option `-mwopt` can be used to turn on optimizing weights of mixture models.

11.7 Partition model options

Partition models are used for phylogenomic data with multiple genes. You first have to prepare a partition file in NEXUS or RAxML-style format. Then use the following options to input the partition file:

Option	Usage and meaning
`-q`	Specify partition file for edge-equal partition model. That means, all partitions share the same set of branch lengths (like `-q` option of RAxML).
`-spp`	Like `-q` but allowing partitions to have different evolutionary speeds (edge-proportional partition model).

Option	Usage and meaning
`-sp`	Specify partition file for edge-unlinked partition model. That means, each partition has its own set of branch lengths (like `-M` option of RAxML). This is the most parameter-rich partition model to accomodate *heterotachy*.

11.8 Site-specific frequency model options

The site-specific frequency model is used to substantially reduce the time and memory requirement compared with full profile mixture models `C10` to `C60`. For full details see site-specific frequency model. To use this model you have to specify a profile mixture model with e.g. `-m LG+C20+F+G` together with a guide tree or a site frequency file:

Option	Usage and meaning
`-ft`	Specify a guide tree (in Newick format) to infer site frequency profiles.
`-fs`	Specify a site frequency file, e.g. the `.sitefreq` file obtained from `-ft` run. This will save memory used for the first phase of the analysis.
`-fmax`	Switch to posterior maximum mode for obtaining site-specific profiles. Default: posterior mean.

With `-fs` option you can input a file containing your own site frequency profiles. The format of this file is that each line contains the site ID (starting from 1) and the state frequencies (20 for amino-acid) separated by white space. So it has as many lines as the number of sites in the alignment. The order of amino-acids is:

```
A   R   N   D   C   Q   E   G   H   I   L   K   M   F   P   S
    T   W   Y   V
```

11.9 Tree search parameters

The new IQ-TREE search algorithm (Nguyen et al., 2015) has several parameters that can be changed with:

Option	Usage and meaning
`-allnni`	Turn on more thorough and slower NNI search. It means that IQ-TREE will consider all possible NNIs instead of only those in the vicinity of previously applied NNIs. *DEFAULT: OFF*

Option	Usage and meaning
`-djc`	Avoid computing ML pairwise distances and BIONJ tree.
`-fast`	Turn on the fast tree search mode, where IQ-TREE will just construct two starting trees: maximum parsimony and BIONJ, which are then optimized by nearest neighbor interchange (NNI). Introduced in version 1.6.
`-g`	Specify a topological constraint tree file in NEWICK format. The constraint tree can be a multifurcating tree and need not to include all taxa.
`-ninit`	Specify number of initial parsimony trees. *DEFAULT: 100*. Here the PLL library (Flouri et al., 2015) is used, which implements the randomized stepwise addition and parsimony subtree pruning and regafting (SPR).
`-n`	Specify number of iterations to stop. This option overrides `-nstop` criterion.
`-ntop`	Specify number of top initial parsimony trees to optimize with ML nearest neighbor interchange (NNI) search to initialize the candidate set. *DEFAULT: 20*
`-nbest`	Specify number of trees in the candidate set to maintain during ML tree search. *DEFAULT: 5*
`-nstop`	Specify number of unsuccessful iterations to stop. *DEFAULT: 100*
`-pers`	Specify perturbation strength (between 0 and 1) for randomized NNI. *DEFAULT: 0.5*
`-sprrad`	Specify SPR radius for the initial parsimony tree search. *DEFAULT: 6*

NOTE: While the default parameters were empirically determined to work well under our extensive benchmark (Nguyen et al., 2015), it might not hold true for all data sets. If in doubt that tree search is still stuck in local optima, one should repeat analysis with at least 10 IQ-TREE runs. Moreover, our experience showed that `-pers` and `-nstop` are the most relevant options to change in such case. For example, data sets with many short sequences should be analyzed with smaller perturbation strength (e.g. `-pers 0.2`) and larger number of stop iterations (e.g. `-nstop 500`).

11.9.1 Example usages:

- Infer an ML tree for an alignment `data.phy` with increased stopping iteration of 500 and reduced perturbation strength of 0.2:

```
iqtree -s data.phy -m TEST -nstop 500 -pers 0.2
```

- Infer an ML tree for an alignment `data.phy` obeying a topological constraint tree `constraint.tree`:

```
iqtree -s data.phy -m TEST -g constraint.tree
```

11.10 Ultrafast bootstrap parameters

The ultrafast bootstrap (UFBoot) approximation (Minh et al., 2013; Hoang et al., in press) has several parameters that can be changed with:

Option	Usage and meaning
`-bb`	Specify number of bootstrap replicates (>=1000).
`-bcor`	Specify minimum correlation coefficient for UFBoot convergence criterion. *DEFAULT: 0.99*
`-beps`	Specify a small epsilon to break tie in RELL evaluation for bootstrap trees. *DEFAULT: 0.5*
`-bnni`	Perform an additional step to further optimize UFBoot trees by nearest neighbor interchange (NNI) based directly on bootstrap alignments. This option is recommended in the presence of **severe model violations**. It increases computing time by 2-fold but reduces the risk of overestimating branch supports due to severe model violations. Introduced in IQ-TREE 1.6.
`-bsam`	Specify the resampling strategies for partitioned analysis. By default, IQ-TREE resamples alignment sites within partitions. With `-bsam GENE` IQ-TREE will resample partitions. With `-bsam GENESITE` IQ-TREE will resample partitions and then resample sites within resampled partitions (Gadagkar et al., 2005; Seo et al., 2005).
`-nm`	Specify maximum number of iterations to stop. *DEFAULT: 1000*
`-nstep`	Specify iteration interval checking for UFBoot convergence. *DEFAULT: every 100 iterations*
`-wbt`	Turn on writing bootstrap trees to `.ufboot` file. *DEFAULT: OFF*
`-wbtl`	Like `-wbt` but bootstrap trees written with branch lengths. *DEFAULT: OFF*

11.10.1 Example usages:

- Select best-fit model, infer an ML tree and perform ultrafast bootstrap with 1000 replicates:

```
iqtree -s data.phy -m TEST -bb 1000
```

- Reconstruct ML and perform ultrafast bootstrap (5000 replicates) under a partition model file `partition.nex`:

```
iqtree -s data.phy -spp partition.nex -m TEST -bb 5000
```

11.11 Nonparametric bootstrap

The slow standard nonparametric bootstrap (Felsenstein, 1985) can be run with:

Option	Usage and meaning
`-b`	Specify number of bootstrap replicates (recommended >=100). This will perform both bootstrap and analysis on original alignment and provide a consensus tree.
`-bc`	Like `-b` but omit analysis on original alignment.
`-bo`	Like `-b` but only perform bootstrap analysis (no analysis on original alignment and no consensus tree).

11.12 Single branch tests

The following single branch tests are faster than all bootstrap analysis and recommended for extremely large data sets (e.g., >10,000 taxa):

Option	Usage and meaning
`-alrt`	Specify number of replicates (>=1000) to perform SH-like approximate likelihood ratio test (SH-aLRT) (Guindon et al., 2010). If number of replicates is set to 0 (`-alrt 0`), then the parametric aLRT test (Anisimova and Gascuel 2006) is performed, instead of SH-aLRT.
`-abayes`	Perform approximate Bayes test (Anisimova et al., 2011).
`-lbp`	Specify number of replicates (>=1000) to perform fast local bootstrap probability method (Adachi and Hasegawa, 1996b).

TIP: One can combine all these tests (also including UFBoot `-bb` option) within a single IQ-TREE run. Each branch in the resulting tree will be assigned several support values separated by slash (`/`), where the order of support values is stated in the `.iqtree` report file.

11.12.1 Example usages:

- Infer an ML tree and perform SH-aLRT test as well as ultrafast bootstrap with 1000 replicates:

```
iqtree -s data.phy -m TEST -alrt 1000 -bb 1000
```

11.13 Ancestral sequence reconstruction

This feature is newly introduced in version 1.6. You can combine this feature with `-te` option to determine ancestral sequences along a user-defined tree (Otherwise, IQ-TREE computes ancestral sequences of the ML tree).

Option	Usage and meaning
`-asr`	Write ancestral sequences (by empirical Bayesian method) for all nodes of the tree to `.state` file.
`-asr-min`	Specify the minimum threshold of posterior probability to determine the best ancestral state. Default: observed state frequency from the alignment.
`-te`	Specify a user-defined tree to determine ancestral sequences along this tree. You can assign each node of this tree with a node name, and IQ-TREE will report the ancestral sequences of the corresponding nodes. If nodes do not have names, IQ-TREE will automatically assign node names as Node1, Node2, etc.

11.13.1 Example usages:

```
iqtree -s example.phy -m JC+G -asr
```

The first few lines of the output file `example.phy.state` may look like this:

```
# Ancestral state reconstruction for all nodes in example.phy.
   treefile
```

```
# This file can be read in MS Excel or in R with command:
#   tab=read.table('example.phy.state',header=TRUE)
# Columns are tab-separated with following meaning:
#   Node:  Node name in the tree
#   Site:  Alignment site ID
#   State: Most likely state assignment
#   p_X:   Posterior probability for state X (empirical Bayesian
   method)
Node    Site    State   p_A     p_C     p_G     p_T
Node2   1       C       0.00004 0.99992 0.00002 0.00002
Node2   2       A       0.92378 0.00578 0.00577 0.06468
Node2   3       A       0.95469 0.02634 0.00675 0.01222
Node2   4       C       0.00002 0.99992 0.00002 0.00004
...
```

11.14 Tree topology tests

IQ-TREE provides a number of tests for significant topological differences between trees. The AU test implementation in IQ-TREE is much more efficient than the original CONSEL by supporting SSE, AVX and multicore parallelization. Moreover, it is more appropriate than CONSEL for partition analysis by bootstrap resampling sites *within* partitions, whereas CONSEL is not partition-aware.

NOTE: There is a discrepancy between IQ-TREE and CONSEL for the AU test: IQ-TREE implements the least-square estimate for p-values whereas CONSEL provides the maximum-likelihood estimate (MLE) for p-values. Hence, the AU p-values might be slightly different. We plan to implement MLE for AU p-values in IQ-TREE soon.

Option	Usage and meaning
`-z`	Specify a file containing a set of trees. IQ-TREE will compute the log-likelihoods of all trees.
`-zb`	Specify the number of RELL (Kishino et al., 1990) replicates (>=1000) to perform several tree topology tests for all trees passed via `-z`. The tests include bootstrap proportion (BP), KH test (Kishino and Hasegawa, 1989), SH test (Shimodaira and Hasegawa, 1999) and expected likelihood weights (ELW) (Strimmer and Rambaut, 2002).
`-zw`	Used together with `-zb` to additionally perform the weighted-KH and weighted-SH tests.

Option	Usage and meaning
`-au`	Used together with `-zb` to additionally perform the approximately unbiased (AU) test (Shimodaira, 2002). Note that you have to specify the number of replicates for the AU test via `-zb`.
`-n 0`	Only estimate model parameters on an initial parsimony tree and ignore a full tree search to save time.
`-te`	Specify a fixed user tree to estimate model parameters. Thus it behaves like `-n 0` but uses a user-defined tree instead of parsimony tree.

11.14.1 Example usages:

- Given alignment `data.phy`, test a set of trees in `data.trees` using AU test with 10,000 replicates:

```
iqtree -s data.phy -m GTR+G -n 0 -z data.trees -zb 10000 -
   au
```

- Same above but for a partitioned data `partition.nex` and additionally performing weighted test:

```
iqtree -s data.phy -spp partition.nex -n 0 -z data.trees -
   zb 10000 -au -zw
```

11.15 Constructing consensus tree

IQ-TREE provides a fast implementation of consensus tree construction for post analysis:

Option	Usage and meaning
`-t`	Specify a file containing a set of trees.
`-con`	Compute consensus tree of the trees passed via `-t`. Resulting consensus tree is written to `.contree` file.
`-net`	Compute consensus network of the trees passed via `-t`. Resulting consensus network is written to `.nex` file.
`-minsup`	Specify a minimum threshold (between 0 and 1) to keep branches in the consensus tree. `-minsup 0.5` means to compute majority-rule consensus tree. *DEFAULT: 0 to compute extended majority-rule consensus.*

Option	Usage and meaning
`-bi`	Specify a burn-in, which is the number of beginning trees passed via `-t` to discard before consensus construction. This is useful e.g. when summarizing trees from MrBayes analysis.
`-sup`	Specify an input "target" tree file. That means, support values are first extracted from the trees passed via `-t`, and then mapped onto the target tree. Resulting tree with assigned support values is written to `.suptree` file. This option is useful to map and compare support values from different approaches onto a single tree.
`-suptag`	Specify name of a node in `-sup` target tree. The corresponding node of `.suptree` will then be assigned with IDs of trees where this node appears. Special option `-suptag ALL` will assign such IDs for all nodes of the target tree.
`-scale`	Set the scaling factor of support values for `-sup` option (default: 100, i.e. percentages)
`-prec`	Set the precision of support values for `-sup` option (default: 0)

11.16 Computing Robinson-Foulds distance

IQ-TREE provides a fast implementation of Robinson-Foulds (RF) distance computation between trees:

Option	Usage and meaning
`-t`	Specify a file containing a set of trees.
`-rf`	Specify a second set of trees. IQ-TREE computes all pairwise RF distances between two tree sets passed via `-t` and `-rf`
`-rf_all`	Compute all-to-all RF distances between all trees passed via `-t`
`-rf_adj`	Compute RF distances between adjacent trees passed via `-t`

11.16.1 Example usages:

- Compute the pairwise RF distances between 2 sets of trees:

```
iqtree -rf tree_set1 tree_set2
```

- Compute the all-to-all RF distances within a set of trees:

```
iqtree -rf_all tree_set
```

11.17 Generating random trees

IQ-TREE provides several random tree generation models:

Option	Usage and meaning
`-r`	Specify number of taxa. IQ-TREE will create a random tree under Yule-Harding model with specified number of taxa
`-ru`	Like `-r`, but a random tree is created under uniform model.
`-rcat`	Like `-r`, but a random caterpillar tree is created.
`-rbal`	Like `-r`, but a random balanced tree is created.
`-rcsg`	Like `-r`, bur a random circular split network is created.
`-rlen`	Specify three numbers: minimum, mean and maximum branch lengths of the random tree. *DEFAULT:* `-rlen 0.001 0.1 0.999`

11.17.1 Example usages:

- Generate a 100-taxon random tree into the file `100.tree` under the Yule Harding model:

```
iqtree -r 100 100.tree
```

- Generate 100-taxon random tree with mean branch lengths of 0.2 and branch lengths are between 0.05 and 0.3:

```
iqtree -r 100 -rlen 0.05 0.2 0.3 100.tree
```

- Generate a random tree under uniform model:

```
iqtree -ru 100 100.tree
```

- Generate a random tree where taxon labels follow an alignment:

```
iqtree -s example.phy -r 17 example.random.tree
```

Note that, you still need to specify the `-r` option being equal to the number of taxa in the alignment.

11.18 Miscellaneous options

Option	Usage and meaning
`-alninfo`	Print alignment site statistics to `.alninfo` file.
`-blfix`	Fix branch lengths of tree passed via `-t` or `-te`. This is useful to evaluate the log-likelihood of an input tree with fixed tolopogy and branch lengths. *DEFAULT: OFF*
`-blmin`	Specify minimum branch length. Default: the smaller of 0.000001 and 0.1/alignment_length.
`-blmax`	Specify the maximum branch length. Default: 10
`-czb`	Collapse near zero branches, so that the final tree may be multifurcating. This is useful for bootstrapping in the presence of polytomy to reduce bootstrap supports of short branches.
`-me`	Specify the log-likelihood epsilon for final model parameter estimation (Default: 0.01). With `-fast` option, the epsilon is raised to 0.05.
`-wpl`	Write partition log-likelihoods to `.partlh` file. Only effective with partition model.
`-wspr`	Write site posterior probabilities per rate category to `.siteprob` file.
`-wspm`	Write site posterior probabilities per mixture class to `.siteprob` file.
`-wspmr`	Write site posterior probabilities per mixture class and rate category to `.siteprob` file.
`-wsl`	Write site log-likelihoods to `.sitelh` file in TREE-PUZZLE format. Such file can then be passed on to CONSEL for further tree tests.
`-wslr`	Write site log-likelihoods per rate category to `.sitelh` file.
`-wslm`	Write site log-likelihoods per mixture class to `.sitelh` file.
`-wslmr`	Write site log-likelihoods per mixture class and rate category to `.sitelh` file.
`-wt`	Turn on writing all locally optimal trees into `.treels` file.
`-fconst`	Specify a list of comma-separated integer numbers. The number of entries should be equal to the number of states in the model (e.g. 4 for DNA and 20 for protein). IQ-TREE will then add a number of constant sites accordingly. For example, `-fconst 10,20,15,40` will add 10 constant sites of all A, 20 constant sites of all C, 15 constant sites of all G and 40 constant sites of all T into the alignment.

11.18.1 Example usages:

- Print alignment information about parsimony informative sites:

```
iqtree -s example.phy -m JC -n 0 -alninfo
```

The first few lines of the output file `example.phy.alninfo` may look like this:

```
# Alignment site statistics
# This file can be read in MS Excel or in R with command:
#   tab=read.table('example.phy.alninfo',header=TRUE)
# Columns are tab-separated with following meaning:
#   Site:   Site ID
#   Stat:   Statistic, I=informative, C=constant, c=constant+
   ambiguous,
#           U=Uninformative but not constant, -=all-gaps
Site    Stat
1       U
2       I
3       I
4       U
5       U
```

- Print site log-likelihood and posterior probability for each Gamma rate category:

```
iqtree -s example.phy -m JC+G -wspr -wslr  -n 0
```

The first few lines of the output file example.phy.siteprob (printed by `-wspr` option) may look like this:

```
Site    p1      p2      p3      p4
1       0.180497        0.534405        0.281   0.00409816
2       4.73239e-05     0.0373409       0.557705        0.404907
3       1.23186e-08     0.000426294     0.0672021       0.932372
4       0.180497        0.534405        0.281   0.00409816
5       0.180497        0.534405        0.281   0.00409816
```

where `pX` is the probability of the site falling into rate category `X`.

The first few lines of the output file example.phy.sitelh (printed by `-wslr` option) may look like this:

```
# Site likelihood per rate/mixture category
# This file can be read in MS Excel or in R with command:
#   tab=read.table('example.phy.sitelh',header=TRUE,fill=TRUE)
# Columns are tab-separated with following meaning:
#   Site:   Alignment site ID
#   LnL:    Logarithm of site likelihood
#           Thus, sum of LnL is equal to tree log-likelihood
#   LnLW_k: Logarithm of (category-k site likelihood times
   category-k weight)
```

```
#            Thus, sum of exp(LnLW_k) is equal to exp(LnL)
Site    LnL      LnLW_1  LnLW_2  LnLW_3  LnLW_4
1       -7.0432 -8.7552 -7.6698 -8.3126 -12.5404
2       -17.5900         -27.5485         -20.8776         -18.1739
          -18.4941
3       -20.3313         -38.5435         -28.0917         -23.0314
          -20.4014
4       -7.0432 -8.7552 -7.6698 -8.3126 -12.5404
5       -7.0432 -8.7552 -7.6698 -8.3126 -12.5404
```

Chapter 12

Substitution models

All common substitution models and usages.

IQ-TREE supports a wide range of substitution models, including advanced partition and mixture models. This guide gives a detailed information of all available models.

TIP: If you do not know which model to use, simply run IQ-TREE with the standard model selection (`-m TEST` option) or the new ModelFinder (`-m MFP`). It automatically determines best-fit model for your data.

12.1 DNA models

12.1.1 Base substitution rates

IQ-TREE includes all common DNA models (ordered by complexity):

Model	df	Explanation	Code
JC or JC69	0	Equal substitution rates and equal base frequencies (Jukes and Cantor, 1969).	000000
F81	3	Equal rates but unequal base freq. (Felsenstein, 1981).	000000
K80 or K2P	1	Unequal transition/transversion rates and equal base freq. (Kimura, 1980).	010010
HKY or HKY85	4	Unequal transition/transversion rates and unequal base freq. (Hasegawa, Kishino and Yano, 1985).	010010
TN or TN93	5	Like `HKY` but unequal purine/pyrimidine rates (Tamura and Nei, 1993).	010020

Model	df	Explanation	Code
TNe	2	Like `TN` but equal base freq.	010020
K81 or K3P	2	Three substitution types model and equal base freq. (Kimura, 1981).	012210
K81u	5	Like `K81` but unequal base freq.	012210
TPM2	2	AC=AT, AG=CT, CG=GT and equal base freq.	010212
TPM2u	5	Like `TPM2` but unequal base freq.	010212
TPM3	2	AC=CG, AG=CT, AT=GT and equal base freq.	012012
TPM3u	5	Like `TPM3` but unequal base freq.	012012
TIM	6	Transition model, AC=GT, AT=CG and unequal base freq.	012230
TIMe	3	Like `TIM` but equal base freq.	012230
TIM2	6	AC=AT, CG=GT and unequal base freq.	010232
TIM2e	3	Like `TIM2` but equal base freq.	010232
TIM3	6	AC=CG, AT=GT and unequal base freq.	012032
TIM3e	3	Like `TIM3` but equal base freq.	012032
TVM	7	Transversion model, AG=CT and unequal base freq.	012314
TVMe	4	Like `TVM` but equal base freq.	012314
SYM	5	Symmetric model with unequal rates but equal base freq. (Zharkikh, 1994).	012345
GTR	8	General time reversible model with unequal rates and unequal base freq. (Tavare, 1986).	012345

The last column `Code` is a 6-digit code defining the equality constraints for 6 *relative* substitution rates: A-C, A-G, A-T, C-G, C-T and G-T. `010010` means that A-G rate is equal to C-T rate (corresponding to `1` in the code) and the remaining four substitution rates are equal (corresponding to `0` in the code). Thus, `010010` is equivalent to K80 or HKY model (depending on whether base frequencies are equal or not). `012345` is equivalent to GTR or SYM model as there is no restriction defined by such 6-digit code.

Moreover, IQ-TREE supports arbitrarily restricted DNA model via a 6-digit code, e.g. with option `-m 012012+G`.

NOTE: The digits in the codes do not necessarily have to have the same order as above. That means '101101' describes the same matrix as '010010'. The last rate, which corresponds to G-T, (and all rates with the same digit) is always set equal to 1.0 for convenience because the rates are relative.

If users want to fix model parameters, append the model name with a curly bracket `{`, followed by the comma-separated rate parameters, and a closing curly bracket `}`. For example, `GTR{1.0,2.0,1.5,3.7,2.8}` specifies 6 substitution rates A-C=1.0, A-G=2.0, A-T=1.5, C-G=3.7, C-T=2.8 and G-T=1.0.

Another example is for model `TIM2` that has the 6-digit code `010232`. Thus, `TIM2{4.39,5.30,12.1}` means that A-C=A-T=4.39 (coded `0`), A-G=5.30 (coded `1`), C-T=12.1 (coded `3`) and C-G=G-T=1.0 (coded `2`). This is, in turn, equivalent to specifying `GTR{4.39,5.30,4.39,1.0,12.1}`.

12.1.2 Base frequencies

Users can specify three different kinds of base frequencies:

FreqType	Explanation
+F	Empirical base frequencies. This is the default if the model has unequal base freq. In AliSim, if users neither specify base frequencies nor supply an input alignment, AliSim will generate base frequencies from empirical distributions.
+FQ	Equal base frequencies.
+FO	Optimized base frequencies by maximum-likelihood.

For example, `GTR+FO` optimizes base frequencies by ML whereas `GTR+F` (default) counts base frequencies directly from the alignment.

Finally, users can fix base frequencies with e.g. `GTR+F{0.1,0.2,0.3,0.4}` to fix the corresponding frequencies of A, C, G and T (must sum up to 1.0).

12.1.3 Lie Markov models

Starting with version 1.6, IQ-TREE supports a series of Lie Markov models (Woodhams et al., 2015), many of which are non-reversible models. Lie Markov models have a consistent property, which is lacking in other common models such as GTR. The following table shows the list of all Lie Markov models (the number before . in the name shows the number of parameters of the model):

Model	Rev?	Freq	Note
1.1	Yes	0	equiv. to JC
2.2b	Yes	0	equiv. to K2P
3.3a	Yes	0	equiv. to K3P

Model	Rev?	Freq	Note
3.3b	No	0	
3.3c	Yes	0	equiv. to TNe
3.4	Yes	1	
4.4a	Yes	3	equiv. to F81
4.4b	Yes	1	
4.5a	No	1	
4.5b	No	1	
5.6a	No	0	
5.6b	No	3	
5.7a	No	2	
5.7b	No	0	
5.7c	No	0	
5.11a	No	2	
5.11b	No	0	
5.11c	No	0	
5.16	No	1	
6.6	No	1	equiv. to STRSYM for strand-symmetric model (Bielawski and Gold, 2002)
6.7a	No	3	F81+K3P
6.7b	No	3	
6.8a	No	3	
6.8b	No	1	
6.17a	No	1	
6.17b	No	1	
8.8	No	3	
8.10a	No	3	
8.10b	No	1	
8.16	No	3	
8.17	No	3	
8.18	No	3	
9.20a	No	2	
9.20b	No	0	Doubly stochastic
10.12	No	3	
10.34	No	3	
12.12	No	3	equiv. to UNREST (unrestricted model)

Column **Rev?** shows whether the model is reversible or not. Column **Freq** shows the number of free base frequencies. 0 means equal base frequency; 1 means f(A)=f(G) and

f(C)=f(T); 2 means f(A)+f(G)=0.5=f(C)+f(T); 3 means unconstrained frequencies.

All Lie Markov models can have one of the following prefices:

Prefix	Meaning
RY	purine-pyrimidine pairing (default)
WS	weak-strong pairing
MK	aMino-Keto pairing

12.2 Protein models

12.2.1 Amino-acid exchange rate matrices

IQ-TREE supports all common empirical amino-acid exchange rate matrices (alphabetical order):

Model	Region	Explanation
Blosum62	nuclear	BLOcks SUbstitution Matrix (Henikoff and Henikoff, 1992). Note that `BLOSUM62` is not recommended for phylogenetic analysis as it was designed mainly for sequence alignments.
cpREV	chloroplast	chloroplast matrix (Adachi et al., 2000).
Dayhoff	nuclear	General matrix (Dayhoff et al., 1978).
DCMut	nuclear	Revised `Dayhoff` matrix (Kosiol and Goldman, 2005).
FLAVI	viral	Flavivirus (Le and Vinh, 2020).
FLU	viral	Influenza virus (Dang et al., 2010).
GTR20	general	General time reversible models with 190 rate parameters. *WARNING: Be careful when using this parameter-rich model as parameter estimates might not be stable, especially when not having enough phylogenetic information (e.g. not long enough alignments).*
HIVb	viral	HIV between-patient matrix HIV-Bm (Nickle et al., 2007).
HIVw	viral	HIV within-patient matrix HIV-Wm (Nickle et al., 2007).
JTT	nuclear	General matrix (Jones et al., 1992).
JTTDCMut	nuclear	Revised `JTT` matrix (Kosiol and Goldman, 2005).
LG	nuclear	General matrix (Le and Gascuel, 2008).
mtART	mitochondrial	Mitochondrial Arthropoda (Abascal et al., 2007).
mtMAM	mitochondrial	Mitochondrial Mammalia (Yang et al., 1998).
mtREV	mitochondrial	Mitochondrial Vertebrate (Adachi and Hasegawa, 1996).
mtZOA	mitochondrial	Mitochondrial Metazoa (Animals) (Rota-Stabelli et al., 2009).
mtMet	mitochondrial	Mitochondrial Metazoa (Vinh et al., 2017).

Model	Region	Explanation
mtVer	mitochondrial	Mitochondrial Vertebrate (Vinh et al., 2017).
mtInv	mitochondrial	Mitochondrial Inverterbrate (Vinh et al., 2017).
NQ.bird	nuclear	Non-reversible Q matrix (Dang et al., 2022) estimated for birds (Jarvis et al., 2015).
NQ.insect	nuclear	Non-reversible Q matrix (Dang et al., 2022) estimated for insects (Misof et al., 2014).
NQ.mammal	nuclear	Non-reversible Q matrix (Dang et al., 2022) estimated for mammals (Wu et al., 2018).
NQ.pfam	nuclear	General non-reversible Q matrix (Dang et al., 2022) estimated from Pfam version 31 database (El-Gebali et al., 2018).
NQ.plant	nuclear	Non-reversible Q matrix (Dang et al., 2022) estimated for plants (Ran et al., 2018).
NQ.yeast	nuclear	Non-reversible Q matrix (Dang et al., 2022) estimated for yeasts (Shen et al., 2018).
Poisson	none	Equal amino-acid exchange rates and frequencies.
PMB	nuclear	Probability Matrix from Blocks, revised `BLOSUM` matrix (Veerassamy et al., 2004).
Q.bird	nuclear	Q matrix (Minh et al., 2021) estimated for birds (Jarvis et al., 2015).
Q.insect	nuclear	Q matrix (Minh et al., 2021) estimated for insects (Misof et al., 2014).
Q.mammal	nuclear	Q matrix (Minh et al., 2021) estimated for mammals (Wu et al., 2018).
Q.pfam	nuclear	General Q matrix (Minh et al., 2021) estimated from Pfam version 31 database (El-Gebali et al., 2018).
Q.plant	nuclear	Q matrix (Minh et al., 2021) estimated for plants (Ran et al., 2018).
Q.yeast	nuclear	Q matrix (Minh et al., 2021) estimated for yeasts (Shen et al., 2018).
rtREV	viral	Retrovirus (Dimmic et al., 2002).
VT	nuclear	General 'Variable Time' matrix (Mueller and Vingron, 2000).
WAG	nuclear	General matrix (Whelan and Goldman, 2001).

12.2.2 Protein mixture models

IQ-TREE also supports a series of protein mixture models:

Model	Explanation
C10 to C60	10, 20, 30, 40, 50, 60-profile mixture models (Le et al., 2008a) as variants of the CAT model (Lartillot and Philippe, 2004) for ML. Note that these models assume `Poisson` AA replacement and implicitly include a Gamma rate heterogeneity among sites.
EX2	Two-matrix model for exposed/buried AA sites (Le et al., 2008b).
EX3	Three-matrix model for highly exposed/intermediate/buried AA sites (Le et al., 2008b).
EHO	Three-matrix model for extended/helix/other sites (Le et al., 2008b).
UL2, UL3	Unsupervised-learning variants of `EX2` and `EX3`, respectively.
EX_EHO	Six-matrix model combining `EX2` and `EHO` (Le and Gascuel, 2010).
LG4M	Four-matrix model fused with Gamma rate heterogeneity (Le et al., 2012).
LG4X	Four-matrix model fused with FreeRate heterogeneity (Le et al., 2012).
CF4	Five-profile mixture model (Wang et al., 2008).

One can even combine a protein matrix with a profile mixture model like:

- `LG+C20`: Applying `LG` matrix instead of `Poisson` for all 20 classes of AA profiles and a Gamma rate heterogeneity.
- `LG+C20+F`: Applying `LG` matrix for 20 classes plus the 21th class of empirical AA profile (counted from the current data) and Gamma rate heterogeneity.
- `JTT+CF4+G`: Applying `JTT` matrix for all 5 classes of AA profiles and Gamma rate heteorogeneity.

Moreover, one can override the Gamma rate by FreeRate heterogeneity:

- `LG+C20+R4`: Like `LG+C20` but replace Gamma by FreeRate heterogeneity.

12.2.3 User-defined empirical protein models

If the matrix name does not match any of the above listed models, IQ-TREE assumes that it is a file containing AA exchange rates and frequencies in PAML format. It contains the lower diagonal part of the matrix and 20 AA frequencies, e.g.:

```
0.425093
0.276818 0.751878
0.395144 0.123954 5.076149
2.489084 0.534551 0.528768 0.062556
0.969894 2.807908 1.695752 0.523386 0.084808
1.038545 0.363970 0.541712 5.243870 0.003499 4.128591
```

```
2.066040 0.390192 1.437645 0.844926 0.569265 0.267959 0.348847
0.358858 2.426601 4.509238 0.927114 0.640543 4.813505 0.423881
   0.311484
0.149830 0.126991 0.191503 0.010690 0.320627 0.072854 0.044265
   0.008705 0.108882
0.395337 0.301848 0.068427 0.015076 0.594007 0.582457 0.069673
   0.044261 0.366317 4.145067
0.536518 6.326067 2.145078 0.282959 0.013266 3.234294 1.807177
   0.296636 0.697264 0.159069 0.137500
1.124035 0.484133 0.371004 0.025548 0.893680 1.672569 0.173735
   0.139538 0.442472 4.273607 6.312358 0.656604
0.253701 0.052722 0.089525 0.017416 1.105251 0.035855 0.018811
   0.089586 0.682139 1.112727 2.592692 0.023918 1.798853
1.177651 0.332533 0.161787 0.394456 0.075382 0.624294 0.419409
   0.196961 0.508851 0.078281 0.249060 0.390322 0.099849 0.094464
4.727182 0.858151 4.008358 1.240275 2.784478 1.223828 0.611973
   1.739990 0.990012 0.064105 0.182287 0.748683 0.346960 0.361819
    1.338132
2.139501 0.578987 2.000679 0.425860 1.143480 1.080136 0.604545
   0.129836 0.584262 1.033739 0.302936 1.136863 2.020366 0.165001
    0.571468 6.472279
0.180717 0.593607 0.045376 0.029890 0.670128 0.236199 0.077852
   0.268491 0.597054 0.111660 0.619632 0.049906 0.696175 2.457121
    0.095131 0.248862 0.140825
0.218959 0.314440 0.612025 0.135107 1.165532 0.257336 0.120037
   0.054679 5.306834 0.232523 0.299648 0.131932 0.481306 7.803902
    0.089613 0.400547 0.245841 3.151815
2.547870 0.170887 0.083688 0.037967 1.959291 0.210332 0.245034
   0.076701 0.119013 10.649107 1.702745 0.185202 1.898718
   0.654683 0.296501 0.098369 2.188158 0.189510 0.249313

0.079066 0.055941 0.041977 0.053052 0.012937 0.040767 0.071586
   0.057337 0.022355 0.062157 0.099081 0.064600 0.022951 0.042302
    0.044040 0.061197 0.053287 0.012066 0.034155 0.069147
```

(This is an example of an LG matrix taken from PAML package). Note that the amino-acid order in this file is:

```
 A   R   N   D   C   Q   E   G   H   I   L   K   M   F   P   S
    T   W   Y   V
Ala Arg Asn Asp Cys Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser
   Thr Trp Tyr Val
```

12.2.4 Amino-acid frequencies

By default, AA frequencies are given by the model. Users can change this with:

FreqType	Explanation
+F	empirical AA frequencies from the data. In AliSim, if users neither specify the base frequencies nor supply an input alignment, AliSim will randomly generate the base frequencies from Uniform distribution.
+FO	ML optimized AA frequencies from the data.
+FQ	Equal AA frequencies.

Users can also specify AA frequencies with, e.g.:

```
+F
   {0.079066,0.055941,0.041977,0.053052,0.012937,0.040767,0.071586,0.057337,0.0
```

(Example corresponds to the AA frequencies of the LG matrix).

12.3 Codon models

To apply a codon model one should use the option `-st CODON` to tell IQ-TREE that the alignment contains protein coding sequences (otherwise, IQ-TREE thinks that it contains DNA sequences and will apply DNA models). This implicitly applies the standard genetic code. You can change to an other genetic code by appending the appropriate ID to the `CODON` keyword:

Code	Genetic code meaning
CODON1	The Standard Code (same as `-st CODON`)
CODON2	The Vertebrate Mitochondrial Code
CODON3	The Yeast Mitochondrial Code
CODON4	The Mold, Protozoan, and Coelenterate Mitochondrial Code and the Mycoplasma/Spiroplasma Code
CODON5	The Invertebrate Mitochondrial Code
CODON6	The Ciliate, Dasycladacean and Hexamita Nuclear Code
CODON9	The Echinoderm and Flatworm Mitochondrial Code
CODON10	The Euplotid Nuclear Code

Code	Genetic code meaning
CODON11	The Bacterial, Archaeal and Plant Plastid Code
CODON12	The Alternative Yeast Nuclear Code
CODON13	The Ascidian Mitochondrial Code
CODON14	The Alternative Flatworm Mitochondrial Code
CODON16	Chlorophycean Mitochondrial Code
CODON21	Trematode Mitochondrial Code
CODON22	Scenedesmus obliquus Mitochondrial Code
CODON23	Thraustochytrium Mitochondrial Code
CODON24	Pterobranchia Mitochondrial Code
CODON25	Candidate Division SR1 and Gracilibacteria Code

(The IDs follow the specification at http://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi).

12.3.1 Codon substitution rates

IQ-TREE supports several codon models:

Model	Explanation
MG	Nonsynonymous/synonymous (dn/ds) rate ratio (Muse and Gaut, 1994).
MGK	Like `MG` with additional transition/transversion (ts/tv) rate ratio.
MG1KTS or MGKAP2	Like `MG` with a transition rate (Kosiol et al., 2007).
MG1KTV or MGKAP3	Like `MG` with a transversion rate (Kosiol et al., 2007).
MG2K or MGKAP4	Like `MG` with a transition rate and a transversion rate (Kosiol et al., 2007).
GY	Nonsynonymous/synonymous and transition/transversion rate ratios (Goldman and Yang, 1994).
GY1KTS or GYKAP2	Like `GY` with a transition rate (Kosiol et al., 2007).
GY1KTV or GYKAP3	Like `GY` with a transversion rate (Kosiol et al., 2007).
GY2K or GYKAP4	Like `GY` with a transition rate and a transversion rate (Kosiol et al., 2007).

Model	Explanation
ECMK07 or KOSI07	Empirical codon model (Kosiol et al., 2007).
ECMrest	Restricted version of `ECMK07` that allows only one nucleotide exchange.
ECMS05 or SCHN05	Empirical codon model (Schneider et al., 2005).

Users could specify the model parameters (e.g., Nonsynonymous/synonymous (dn/ds) rate ratio, and/or transition/transversion (ts/tv) rate ratio, and/or transition rate, and/or a transversion rate) by `<Model_Name>{<omega>,[<kappa>],[<kappa2>]}`. For example, `MG2K{1.0,0.3,0.5}` specifies the nonsynonymous/synonymous (dn/ds) rate ratio, the transition rate, and the transversion rate are 1.0, 0.3, 0.5, respectively. The number of input parameters depends on the definition of each model.

The last three models (`ECMK07`, `ECMrest` or `ECMS05`) are called *empirical* codon models, whereas the others are called *mechanistic* codon models.

Moreover, IQ-TREE supports combined empirical-mechanistic codon models using an underscore separator (_). For example:

- `ECMK07_GY2K`: The combined `ECMK07` and `GY2K` model, with the rate entries being multiplication of the two corresponding rate matrices.

Thus, there can be many such combinations.

Starting with version 1.5.6: If the model name does not match any of the above listed models, IQ-TREE assumes that it is a file containing lower diagonal part of non-stop codon exchange rate matrix, non-stop codon frequencies and a list of non-stop codons. The rest of the file will be ignored. Example files (ECMrest.dat and ECMunrest.dat) can be downloaded from the supplementary material (Kosiol et al., 2007).

NOTE: Branch lengths under codon models are interpreted as number of nucleotide substitutions per codon site. Thus, they are typically 3 times longer than under DNA models.

12.3.2 Codon frequencies

IQ-TREE supports the following codon frequencies:

FreqType	Explanation
+F	Empirical codon frequencies counted from the data. In AliSim, if users neither specify base frequencies nor supply an input alignment, AliSim will generate base frequencies from empirical distributions.
+FQ	Equal codon frequencies.
+F1X4	Unequal nucleotide frequencies but equal nt frequencies over three codon positions. In AliSim, if users don't supply an input alignment, the base frequencies are randomly generated based on empirical distributions, or users could specify the frequencies via `+F1X4{<freq_0>,...,<freq_4>}`.
+F3X4	Unequal nucleotide frequencies and unequal nt frequencies over three codon positions. In AliSim, if users don't supply an input alignment, the base frequencies are randomly generated based on empirical distributions, or users could specify the frequencies via `+F3X4{<freq_0>,...,<freq_11>}`

If not specified, the default codon frequency will be `+F3X4` for `MG`-type models, `+F` for `GY`-type models and given by the model for empirical codon models.

12.4 Binary and morphological models

The binary alignments should contain state `0` and `1`, whereas for morphological data, the valid states are `0` to `9` and `A` to `Z`.

Model	Explanation
JC2	Jukes-Cantor type model for binary data.
GTR2	General time reversible model for binary data.
MK	Jukes-Cantor type model for morphological data.
ORDERED	Allowing exchange of neighboring states only.

Except for `GTR2` that has unequal state frequencies, all other models have equal state frequencies.

TIP: If morphological alignments do not contain constant sites (typically the case), then an ascertainment bias correction model (`+ASC`) should be applied to correct the branch lengths for the absence of constant sites.

12.5 Ascertainment bias correction

An ascertainment bias correction (`+ASC`) model (Lewis, 2001) should be applied if the alignment does not contain constant sites (such as morphological or SNPs data). For example:

- `MK+ASC`: For morphological data.
- `GTR+ASC`: For SNPs data.

`+ASC` will correct the likelihood conditioned on variable sites. Without `+ASC`, the branch lengths might be overestimated.

12.6 Rate heterogeneity across sites

IQ-TREE supports all common rate heterogeneity across sites models:

RateType	Explanation
+I	allowing for a proportion of invariable sites.
+G	discrete Gamma model (Yang, 1994) with default 4 rate categories. The number of categories can be changed with e.g. `+G8`.
+GC	continuous Gamma model (Yang, 1994) (for AliSim only).
+I+G	invariable site plus discrete Gamma model (Gu et al., 1995).
+R	FreeRate model (Yang, 1995; Soubrier et al., 2012) that generalizes the `+G` model by relaxing the assumption of Gamma-distributed rates. The number of categories can be specified with e.g. `+R6` (default 4 categories if not specified). The FreeRate model typically fits data better than the `+G` model and is recommended for analysis of large data sets.
+I+R	invariable site plus FreeRate model.

TIP: The new ModelFinder (`-m MFP` option) tests the FreeRate model, whereas the standard procedure (`-m TEST`) does not.

Users can fix the parameters of the model. For example, `+I{0.2}` will fix the proportion of invariable sites (pinvar) to 0.2; `+G{0.9}` will fix the Gamma shape parameter (alpha) to 0.9; `+I{0.2}+G{0.9}` will fix both pinvar and alpha. To fix the FreeRate model parameters, use the syntax `+Rk{w1,r1,...,wk,rk}` (replacing `k` with the number of categories). Here, `w1, ..., wk` are the weights and `r1, ..., rk` the rates for each category.

NOTE: For the `+G` model IQ-TREE implements the *mean* approximation approach (Yang, 1994). The same is done in RAxML and PhyML. However, some programs like TREE-PUZZLE implement the *median* approximation approach, which makes the resulting log-likelihood not comparable. IQ-TREE can change to this approach via the `-gmedian` option.

Chapter 13

Complex models

Complex models such as partition and mixture models.

This document gives detailed descriptions of complex maximum-likelihood models available in IQ-TREE. It is assumed that you know the basic substitution models already.

13.1 Partition models

Partition models are intended for phylogenomic (e.g., multi-gene) alignments, which allow each partition to have its own substitution models and evolutionary rates. IQ-TREE supports three types of partition models:

1. *Edge-equal* partition model with equal branch lengths: All partitions share the same set of branch lengths.
2. *Edge-proportional* partition model with proportional branch lengths: Like above but each partition has its own partition specific rate, that rescales all its branch lengths. This model accomodates different evolutionary rates between partitions (e.g. between 1st, 2nd, and 3rd codon positions).
3. *Edge-unlinked* partition model: Each partition has its own set of branch lengths. This is the most parameter-rich partition model, that accounts for e.g., *heterotachy* (Lopez et al., 2002).

TIP: The edge-equal partition model is typically unrealistic as it does not account for different evolutionary speeds between partitions, whereas the edge-unlinked partition model can be overfitting if there are many short partitions. Therefore, the edge-proportional partition model is recommended for a typical analysis.

13.1.1 Partition file format

To apply partition models users must first prepare a partition file in RAxML-style or NEXUS format. The RAxML-style is defined by the RAxML software and may look like:

```
DNA, part1 = 1-100
DNA, part2 = 101-384
```

This means two DNA partitions of an alignment, where one groups aligment sites `1-100` into `part1` and `101-384` into `part2`.

The NEXUS format is more complex but more powerful. For example, the above partition scheme may look like:

```
#nexus
begin sets;
    charset part1 = 1-100;
    charset part2 = 101-384;
    charpartition mine = HKY+G:part1, GTR+I+G:part2;
end;
```

The first line contains the keyword `#nexus` to indicate a NEXUS file. It has a `sets` block, which contains two character sets (`charset` command) named `part1` and `part2`. Furthermore, with the `charpartition` command we set the model `HKY+G` for `part1` and `GTR+I+G` for `part2`. This is not possible with the RAxML-style format (i.e., one cannot specify `+G` rate model for one partition and `+I+G` rate model for the other partition).

TIP: IQ-TREE fully supports mixed rate heterogeneity types types between partitions (see above example).

One can also specify non-consecutive sites of a partition, e.g. under RAxML-style format:

```
DNA, part1 = 1-100, 250-384
```

```
DNA, part2 = 101-249\3, 102-249\3
DNA, part3 = 103-249\3
```

or under NEXUS format:

```
#nexus
begin sets;
    charset part1 = 1-100 250-384;
    charset part2 = 101-249\3 102-249\3;
    charset part3 = 103-249\3;
end;
```

This means, `part2` contains sites 101, 102, 104, 105, 107, ..., 246, 248, 249; whereas `part3` contains sites 103, 106, ..., 247. This is useful to specify partitions corresponding to 1st, 2nd and 3rd codon positions.

Moreover, the NEXUS file allows each partition to come from a separate alignment file (not possible under RAxML-style format) with e.g.:

```
#nexus
begin sets;
    charset part1 = aln1.phy: 1-100\3 201-300;
    charset part2 = aln1.phy: 101-200;
    charset part3 = aln2.phy: *;
    charpartition mine = HKY:part1, GTR+G:part2, WAG+I+G:part3;
end;
```

Here, `part1` and `part2` correspond to sub-alignments of `aln1.phy` file and `part3` is the entire alignment file `aln2.phy`. Note that `aln2.phy` is a protein alignment in this example.

TIP: IQ-TREE fully supports mixed data types between partitions.

If you want to specify codon model for a partition, use the `CODON` keyword (otherwise, the partition may be detected as DNA):

```
#nexus
begin sets;
    charset part1 = aln1.phy:CODON, 1-300;
    charset part2 = aln1.phy: 301-400;
    charset part3 = aln2.phy: *;
    charpartition mine = GY:part1, GTR+G:part2, WAG+I+G:part3;
end;
```

Note that this assumes `part1` has standard genetic code. If not, append `CODON` with the right genetic code ID.

13.1.2 Partitioned analysis

Having prepared a partition file, one is ready to start a partitioned analysis with `-q` (edge-equal), `-spp` (edge-proportional) or `-sp` (edge-unlinked) option. See this tutorial for more details.

13.2 Mixture models

13.2.1 What is the difference between partition and mixture models?

Mixture models, like partition models, allow more than one substitution model along the sequences. However, while a partition model assigns each alignment site a given specific model, mixture models do not need this information. A mixture model will compute for each site its probability (or weight) of belonging to each of the mixture classes (also called categories or components). Since the site-to-class assignment is unknown, the site likelihood under mixture models is the weighted sum of site likelihoods per mixture class.

For example, the discrete Gamma rate heterogeneity is a simple mixture model type. It has several rate categories with equal weight. IQ-TREE also supports a number of predefined protein mixture models such as the profile mixture models `C10` to `C60` (The ML variants of Bayesian `CAT` models).

Here, we discuss several possibilities to define new mixture models in IQ-TREE.

13.2.2 Defining mixture models

To start with, the following command:

```
iqtree -s example.phy -m "MIX{JC,HKY}"
```

specifies a mixture model (via the `MIX` keyword in the model string) with two components. The components (1) `JC` model, and (2) `HKY` model, are given in curly brackets and separated with a comma. IQ-TREE will then estimate the parameters of both mixture components as well as their weights: the proportion of sites belonging to each component.

NOTE: Do not forget the double-quotes around model string! They prevent interpretation of the curly brackets by the command line shell, i.e., `MIX{JC,HKY}` would otherwise be interpreted as `MIXJC MIXHKY`.

Mixture models can be combined with rate heterogeneity, e.g.:

```
iqtree -s example.phy -m "MIX{JC,HKY}+G4"
```

Here, we specify two mixture components and four Gamma rate categories. Effectively, this means that there are eight mixture components. Each site has a probability belonging to either `JC` or `HKY` and to one of the four rate categories.

13.2.3 Profile mixture models

Sometimes one only wants to model the changes in nucleotide or amino-acid frequencies along the sequences while keeping the substitution rate matrix the same. This can be specified in IQ-TREE via `FMIX{...}` model syntax. For convenience the mixture components can be defined in a NEXUS file like this (example corresponds to the CF4 model of (Wang et al., 2008)):

```
#nexus
begin models;
    frequency Fclass1 = 0.02549352 0.01296012 0.005545202
       0.006005566 0.01002193 0.01112289 0.008811948 0.001796161
       0.004312188 0.2108274 0.2730413 0.01335451 0.07862202
       0.03859909 0.005058205 0.008209453 0.03210019 0.002668138
       0.01379098 0.2376598;
    frequency Fclass2 = 0.09596966 0.008786096 0.02805857
       0.01880183 0.005026264 0.006454635 0.01582725 0.7215719
       0.003379354 0.002257725 0.003013483 0.01343441 0.001511657
        0.002107865 0.006751404 0.04798539 0.01141559 0.000523736
        0.002188483 0.004934972;
    frequency Fclass3 = 0.01726065 0.005467988 0.01092937
       0.3627871 0.001046402 0.01984758 0.5149206 0.004145081
       0.002563289 0.002955213 0.005286931 0.01558693 0.002693098
        0.002075771 0.003006167 0.01263069 0.01082144 0.000253451
        0.001144787 0.004573568;
    frequency Fclass4 = 0.1263139 0.09564027 0.07050061
       0.03316681 0.02095119 0.05473468 0.02790523 0.009007538
       0.03441334 0.005855319 0.008061884 0.1078084 0.009019514
       0.05018693 0.07948 0.09447839 0.09258897 0.01390669
       0.05367769 0.01230413;
```

```
    frequency CF4model = FMIX{empirical,Fclass1,Fclass2,Fclass3,
       Fclass4};
end;
```

NOTE: The amino-acid order in this file is: A R N D C Q E G H I L K M F P S T W Y V.

Here, the NEXUS file contains a `models` block to define new models. More explicitly, we define four AA profiles `Fclass1` to `Fclass4`, each containing 20 AA frequencies. Then, the frequency mixture is defined with

```
FMIX{empirical,Fclass1,Fclass2,Fclass3,Fclass4}
```

This means, we have five components: the first corresponds to empirical AA frequencies to be inferred from the data and the remaining four components are specified in this NEXUS file. Please save this to a file, say, `mymodels.nex`. One can now start the analysis with:

```
iqtree -s some_protein.aln -mdef mymodels.nex -m JTT+CF4model+G
```

The `-mdef` option specifies the NEXUS file containing user-defined models. Here, the `JTT` matrix is applied for all alignment sites and one varies the AA profiles along the alignment. One can use the NEXUS syntax to define all other profile mixture models such as `C10` to `C60`.

13.2.4 NEXUS model file

In fact, IQ-TREE uses this NEXUS model file internally to define all protein mixture models. In addition to defining state frequencies, one can specify the entire model with rate matrix and state frequencies together. For example, the LG4M model (Le et al., 2012) can be defined by:

```
#nexus
begin models;
    model LG4M1 =
        0.269343
        0.254612 0.150988
        0.236821 0.031863 0.659648
        ....;
    ....
    model LG4M4 = ....;
```

```
    model LG4M = MIX{LG4M1,LG4M2,LG4M3,LG4M4}*G4;
end;
```

Here, we first define the four matrices `LG4M1`, `LG4M2`, `LG4M3` and `LG4M4` in PAML format (see protein models). Then `LG4M` is defined as mixture model with these four components *fused* with Gamma rate heterogeneity (via `*G4` syntax instead of `+G4`). This means that, in total, we have 4 mixture components instead of 16. The first component `LG4M1` is rescaled by the rate of the lowest Gamma rate category. The fourth component `LG4M4` corresponds to the highest rate.

Note that both `frequency` and `model` commands can be embedded into a single model file.

13.3 Site-specific frequency models

Starting with version 1.5.0, IQ-TREE provides a new posterior mean site frequency (PMSF) model as a rapid approximation to the time and memory consuming profile mixture models `C10` to `C60` (Le et al., 2008a; a variant of PhyloBayes' `CAT` model). The PMSF are the amino-acid profiles for each alignment site computed from an input mixture model and a guide tree. The PMSF model is much faster and requires much less RAM than `C10` to `C60` (see table below), regardless of the number of mixture classes. Our extensive simulations and empirical phylogenomic data analyses demonstrate that the PMSF models can effectively ameliorate long branch attraction artefacts.

If you use this model in a publication please cite:

> **H.C. Wang, B.Q. Minh, S. Susko and A.J. Roger** (2018) Modeling site heterogeneity with posterior mean site frequency profiles accelerates accurate phylogenomic estimation. *Syst. Biol.*, 67:216-235. https://doi.org/10.1093/sysbio/syx068

Here is an example of computation time and RAM usage for an Obazoa data set (68 sequences, 43615 amino-acid sites) from Brown et al. (2013) using 16 CPU cores:

Models	CPU time	Wall-clock time	RAM usage
`LG+F+G`	43h:38m:23s	3h:37m:23s	1.8 GB
`LG+C20+F+G`	584h:25m:29s	46h:39m:06s	38.8 GB
`LG+C60+F+G`	1502h:25m:31s	125h:15m:29s	112.8 GB
`LG+PMSF+G`	73h:30m:37s	5h:7m:27s	2.2 GB

13.3.1 Example usages

To use the PMSF model you have to provide a *guide tree*, which, for example, can be obtained by a quicker analysis under the simpler `LG+F+G` model. The guide tree can then be specified via `-ft` option, for example:

```
iqtree -s <alignment> -m LG+C20+F+G -ft <guide_tree>
```

Here, IQ-TREE will perform two phases. In the first phase, IQ-TREE estimates mixture model parameters given the guide tree and then infers the site-specific frequency profile (printed to `.sitefreq` file). In the second phase, IQ-TREE will conduct typical analysis using the inferred frequency model instead of the mixture model to save RAM and running time. Note that without `-ft` option, IQ-TREE will conduct the analysis under the specified mixture model.

The PMSF model allows one, for the first time, to conduct nonparametric bootstrap under such complex models, for example (with 100 bootstrap replicates):

```
iqtree -s <alignment> -m LG+C20+F+G -ft <guide_tree> -b 100
```

Please note that the first phase still consumes as much RAM as the mixture model. To overcome this, you can perform the first phase in a high-memory server and the second phase in a normal PC as follows:

```
iqtree -s <alignment> -m LG+C20+F+G -ft <guide_tree> -n 0
```

This will stop the analysis after the first phase and also write a `.sitefreq` file. You can now copy this `.sitefreq` file to another low-memory machine and run with the same alignment:

```
iqtree -s <alignment> -m LG+C20+F+G -fs <file.sitefreq> -b 100
```

This will omit the first phase and thus need much less RAM.

Finally, note that for long (phylogenomic) alignments you can utilize the multicore IQ-TREE version to further save the computing times with, say, 24 cores by:

```
# For IQ-TREE version <= 1.5.X
iqtree-omp -nt 24 -s <alignment> -m LG+C20+F+G -fs <file.sitefreq
   >

# For IQ-TREE version >= 1.6.0
iqtree -nt 24 -s <alignment> -m LG+C20+F+G -fs <file.sitefreq>
```

See also the list of relevant command line options.

13.4 Heterotachy models

Sequence data that have evolved under *heterotachy*, i.e., rate variation across sites and lineages (Lopez, Casane, and Philippe, 2002), are known to mislead phylogenetic inference (Kolaczkowski and Thornton, 2004). To address this issue we introduce the General Heterogeneous evolution On a Single Topology (GHOST) model. More specifically, GHOST is an *edge-unlinked mixture model* consisting of several site classes, each having a separate set of model parameters and edge lengths on the same tree topology. Thus, GHOST naturally accounts for heterotachous evolution. In contrast to an edge-unlinked partition model, the GHOST model does not require the *a priori* data partitioning, a possible source of model misspecification.

Extensive simulations show that the GHOST model can accurately recover the tree topology, branch lengths, substitution rate and base frequency parameters from heterotachously-evolved sequences. Moreover, we compare the GHOST model to the partition model and show that, owing to the minimization of model constraints, the GHOST model is able to offer unique biological insights when applied to empirical data.

If you use this model in a publication please cite:

S.M. Crotty, B.Q. Minh, N.G. Bean, B.R. Holland, J. Tuke, L.S. Jermiin and A. von Haeseler (2019) GHOST: Recovering historical signal from heterotachously-evolved sequence alignments. *Syst. Biol.*, in press. https://doi.org/10.1093/sysbio/syz051

13.4.1 Quick usages

Make sure that you have IQ-TREE version 1.6.0 or later. The GHOST model with `k` mixture classes is executed by adding `+Hk` to the model option (`-m`). For example if one wants to fit a GHOST model with 4 classes in conjunction with the `GTR` model of DNA evolution to sequences contained in `data.fst`, one would use the following command:

```
iqtree -s data.fst -m GTR+H4
```

By default the above command will link GTR parameters across all classes. If you want to unlink GTR parameters, so that IQ-TREE estimates them separately for each class, replace `+H4` by `*H4`:

```
iqtree -s data.fst -m GTR*H4
```

Note that this infers one set of empirical base frequencies and apply those to all classes. If one wishes to infer separate base frequencies for each class then the `+FO` option is required:

```
iqtree -s data.fst -m GTR+FO*H4
```

The `-wspm` option will generate a `.siteprob` output file. This contains the probability of each site belonging to each class:

```
iqtree -s data.fst -m GTR+FO*H4 -wspm
```

13.5 Multitree models

Hundreds or thousands of loci are now routinely used in modern phylogenomic studies. Concatenation approaches to tree inference assume that there is a single topology for the entire dataset, but different loci may have different evolutionary histories due to incomplete lineage sorting, introgression, and/or horizontal gene transfer; even single loci may not be treelike due to recombination. To overcome this shortcoming, we introduce the mixture across sites and trees (MAST) model, which uses a mixture of bifurcating trees to represent multiple histories in a single concatenated alignment. The MAST model allows each tree to have its own topology, branch lengths, substitution model, nucleotide or amino acid frequencies, and model of rate heterogeneity across sites.

We applied the MAST model to multiple primate datasets and found that it can recover the signal of incomplete lineage sorting in the Great Apes, as well as the asymmetry in minor trees caused by introgression among several macaque species. When applied to a dataset of four Platyrrhine species for which standard concatenated maximum likelihood and gene tree approaches disagree, we find that MAST gives the highest weight to the tree favored by gene tree approaches. These results suggest that the MAST model is able to analyse a concatenated alignment using maximum likelihood, while avoiding some of the biases that come with assuming there is only a single tree. The MAST model can therefore offer unique biological insights when applied to datasets with multiple evolutionary histories.

Meanwhile the manuscript is under review. If you use this model in a publication please cite:

T.K.F. Wong, C. Cherryh, A.G. Rodrigo, M.W. Hahn, B.Q. Minh and R. Lanfear (2022) MAST: Phylogenetic Inference with Mixtures Across Sites and Trees. *bioRxiv*. https://doi.org/10.1101/2022.10.06.511210

13.5.1 Quick usage

** WARNING: Always check that your models make sense before you interpret the things you are interested in Of course, you should *always* do this anyway, but we put this warning here because multitree mixture models are new, somewhat complex, and may be easy to over-parameterise. So, if you are using these models for your research, please keep your biological head screwed on, and before interpreting any output (e.g. the weights of the classes in the mixture) check that the branch lengths of the trees look sensible, that the model parameters (e.g. base frequencies, transition rates, rates across sites) look sensible. Remember that if you are going to interpret any part of the model, you are also putting your faith in all of the other parameters.

Make sure that you have IQ-TREE version 2.2.0.7.mix. The MAST model is executed by adding `+T` to the model option (`-m`) and providing a newick file with multiple trees by the option (`-te`). For example if one wants to fit a MAST model with different topologies contained in `trees.nwk` in conjunction with the `GTR` model to sequences in `data.fst`, one would use the following command:

```
iqtree2 -s data.fst -m "GTR+T" -te trees.nwk
```

The above command will *link* GTR parameters across all the trees. That means all trees will have the same GTR model. IQ-TREE will check the number of trees inside the newick file, and then estimate the model parameters and the weights of each tree: the proportion of sites belonging to each tree.

An example of the newick file with 3 topologies:

```
((A,B),(C,D));
((A,C),(B,D));
((A,D),(B,C));
```

You can also link the GTR parameters, frequency array, and the rate-heterogeneity-across-site (RHAS) model across all the trees by including the frequency and the RHAS model in the model option (`-m`). For example:

```
iqtree2 -s data.fst -m "GTR+FO+G+T" -te trees.nwk
```

If one would like to have *unlink* components across the trees (for example, each tree has its own substitution model, frequency array and RHAS model), one can specify the unlinked components via the `TMIX` keyword in the model string. For example:

```
iqtree2 -s data.fst -m "TMIX{GTR+FO+G,F81+FO+R3,HKY+FO+I}+T" -te
   trees.nwk
```

The above command specifies the `GTR+FO+G` model for the first topology (inside the newick file), the `F81+FO+R3` model for the second topology, and the `HKY+FO+I` model for the third topology. These components are given in curly brackets and separated with a comma. Note that the number of components has to match with the number of topologies in the newick file.

There is a flexibility to set substitution model, frequencies or RHAS model *linked* or *unlinked* separately. The followings show some examples of different situations, assuming there are 2 topologies in the newick file:

	Model option	Linked parameters	Description
1	`"TMIX{GTR+FO+G,GTR+FO+G}+T"`	subst freq RHAS	Each tree has its own GTR model, DNA frequencies and gamma model
2	`"TMIX{GTR+FO,GTR+FO}+G+T"`	subst freq RHAS	Each tree has its own GTR model and DNA frequencies but all share the same gamma model
3	`"TMIX{GTR+F+G,GTR+F+G}+T"`	subst freq RHAS	Each tree has its own GTR model and gamma model, but all DNA frequencies are set to the frequencies of A,C,G,T in the alignment
4	`"TMIX{GTR+F,GTR+F}+G+T"`	subst freq RHAS	Each tree has its own GTR model, but all share the same gamma model and all DNA frequencies are set to the frequencies of A,C,G,T in the alignment
5	`"GTR+FO+TMIX{G,G}+T"`	subst freq RHAS	Each tree has its own gamma model, but all share the same GTR model and DNA frequencies

	Model option	Linked parameters	Description
6	`"GTR+FO+G+T"`	subst freq RHAS	All trees share the same GTR model, DNA frequencies and gamma model

Note: subst - substitution model; freq - DNA/AA frequency array; RHAS - rate heterogeneity across site model

13.5.2 More usages

Branch-length-restricted MAST model

One can use `+TR` instead of `+T` to represent the branch-length-Restricted MAST model. In this model, the length of branch `x` of a tree Ti is constrained to be equal to the length of branch `y` of a tree Tj if the branches `x` and `y` split the trees Ti and Tj into the same two sets of taxa. For example:

```
iqtree2 -s data.fst -m "GTR+FO+G+TR" -te trees.nwk
```

In the above command, all trees share the same GTR model, DNA frequencies and gamma model, and the lengths of the branches across the trees which split the taxa set into the same partition are restricted the same.

Weight-constrained MAST model

One can define a constraint array following `+T` to restrict the tree weights. The constraint array can be defined as [s1,s2,...,sn] where si can be any string. The weight of tree Ti and that of tree Tj are restricted the same value if si = sj. For example, assuming there are 3 topologies in the newick file:

```
iqtree2 -s data.fst -m "GTR+FO+G+T[x,x,y]" -te trees.nwk
```

In the above command, all trees share the same GTR model, DNA frequencies and gamma model, and the weight of the first tree is constrained as the same as the weight of the second tree.

13.5.3 More explanations on the results

File	Description
`.treefile`	By using the MAST model, IQ-TREE will report multiple trees inside this file. Their topologies should match the input topologies in the newick file.
`.iqtree`	All the estimated model parameters for each tree and the tree weights (i.e. proportions of the sites belonging to the tree and the model) are shown in this file. The order of the tree weights follows the order of the input topologies in the newick file.

Please note that, in any MAST model with more than one substitution model (i.e. models 1 - 5 in the previous table), the weights can only be interpreted as the linked weight of the model and the tree. So the weights are not unique to the tree. In other words, IQ-TREE will report the weights pertaining only to the trees for the model 6 in the previous table.

Chapter 14

Polymorphism-aware models

Use population data to improve inferences.

Polymorphism-aware phylogenetic **Mo**dels (PoMo) improve phylogenetic inference using population data (site frequency data). Thereby they builds on top of DNA substitution models and naturally account for incomplete lineage sorting. In order to run PoMo, you need more sequences per species/population (ideally five or more chromosomes per species/population) so that information about the site frequency spectrum is available.

Currently this model is only available in the beta version 1.6. Please download it from here:

http://www.iqtree.org/#variant

TIP: For a quick overview of all PoMo related options in IQ-TREE, run the command `iqtree -h` and scroll to the heading `POLYMORPHISM AWARE MODELS (PoMo)`.

If you use PoMo, please cite Schrempf et al., 2016:

```
Dominik Schrempf, Bui Quang Minh, Nicola De Maio, Arndt von
Haeseler, and Carolin Kosiol (2016) Reversible polymorphism-aware
phylogenetic models and their application to tree inference.
J. Theor. Biol., 407, -362370.
http://doi.org/10.1016/j.jtbi.2016.07.042.
```

14.1 Counts files

The input of PoMo is allele frequency data. Especially, when populations have many individuals it is preferable to count the number of bases at each position compared to providing data for each chromosome in a FASTA file. Thereby file size is decreased and parsed faster.

Counts files contain:

- One headerline that specifies the file as counts file and states the number of populations (leaves on the tree) as well as the number of sites (separated by white space).
- A second headerline with white space separated headings: CRHOM (chromosome), POS (position) and sequence names.
- Many lines with counts of A, C, G and T bases and their respective positions.

Comments:

- Lines before the first headerline starting with # are treated as comments.

Example:

```
COUNTSFILE  NPOP 5   NSITES N
CHROM  POS  Sheep    BlackSheep  RedSheep  Wolf     RedWolf
1      1    0,0,1,0  0,0,1,0     0,0,1,0   0,0,5,0  0,0,0,1
1      2    0,0,0,1  0,0,0,1     0,0,0,1   0,0,0,5  0,0,0,1
.
.
.
9      8373 0,0,0,1  1,0,0,0     0,1,0,0   0,1,4,0  0,0,1,0
.
.
.
Y      9999 0,0,0,1  0,1,0,0     0,1,0,0   0,5,0,0  0,0,1,0
```

The download also includes an example counts file called `example.cf`. This file is a subset of the great ape data that has been analyzed in one of our publications. It includes twelve great ape population with one to 23 inividuals each (two to 56 chromosomes).

14.1.1 Conversion scripts

If you do not want to create counts files with your own scripts, you can use the python script that we provide. For detailed instructions, please refer to the GitHub repository

of the counts file library `cflib`.

14.2 First running example

You can now start to reconstruct a maximum-likelihood tree from this alignment by entering (assuming that `example.cf` is in the same folder):

```
iqtree -s example.cf -m HKY+P
```

or, e.g.,

```
iqtree -nt 4 -s example.cf -m HKY+P
```

if you use the multicore (OMP) version. `-s` specifies of the alignment file and `-m` the model (HKY substitution model with polymorphisms; PoMo), similar to the standard IQ-TREE usage. At the end of the run IQ-TREE writes the same output files as in the standard version (see tutorial).

- `example.cf.iqtree`: the main report file that is self-readable. You should look at this file to see the computational results. It also contains a textual representation of the final tree.
- `example.cf.treefile`: the ML tree in NEWICK format, which can be visualized by any supported tree viewer programs like FigTree or iTOL.
- `example.cf.log`: log file of the entire run (also printed on the screen). To report bugs, please send this log file and the original alignment file to the authors.

The default prefix of all output files is the alignment file name. However, you can always change the prefix using the `-pre` option, e.g.:

```
iqtree -s example.cf -pre myprefix
```

This prevents output files to be overwritten when you perform multiple analyses on the same alignment within the same folder.

14.3 Substitution models

Different DNA substitution models can be selected with the `-m` option. E.g., to select the GTR model, run IQ-TREE with:

```
iqtree -s example.cf -m GTR+P
```

TIP: For a quick overview of all available models in IQ-TREE, run the command `iqtree -h` and scroll to the heading `POLYMORPHISM AWARE MODELS (PoMo)`.

14.4 Virtual population size

PoMo describes the evolution of populations along a phylogeny by means of a virtual population of constant size `N`, which defaults to nine (for details, see Schrempf et al., 2016). This is a good and stable default value. If only very few chromosomes have been sequenced per population (e.g., two to four), `N` should be lowered to the average number of chromosomes per population. If enough data is available and calculations are not too time consuming, we advise to increase N up to a maximum of 19. You can choose odd values from three to 19 as well as 2 and 10. E.g., to set N to 19:

```
iqtree -s example.cf -m HKY+P+N19
```

14.5 Level of polymorphism

As of version `1.6`, IQ-TREE with PoMo also allows fixation of the level of heterozygosity, which is also called Watterson's theta or `4Nu`. When analyzing population data, the amount of polymorphism is inferred during maximization of the likelihood. However, in some situations it may be useful to set the level of polymorphism to the observed value in the data (empirical value):

```
iqtree -s example.cf -m HKY+P{EMP}
```

or to set the level of polymorphism by hand, e.g.,:

```
iqtree -s example.cf -m HKY+P{0.0025}
```

Together with the ability to set model parameters, the model can be fully specified, e.g.:

```
iqtree -s example.cf -m HKY{6.0}+P{0.0025}
```

This sets the transition to transversion ratio to a value of `6.0` and the level of polymorphism to a value of `0.0025`. In this case, IQ-TREE only performs a tree search because the model is fully specified.

14.6 Sampling method

For advanced users. PoMo offers different methods to read in the data (Schrempf et al., 2016). Briefly, each population and site are treated as follows

1. *Weighted binomial* (default, `+WB`): assign the likelihood of each PoMo state to its probability of leading to the observed data, assuming it is **binomially** sampled. Example:

   ```
   iqtree -s example.cf -m HKY+P+WB
   ```

2. *Weighted hypergeometric* (`+WH`): assign the likelihood of each PoMo state to its probability of leading to the observed data, assuming it is **hypergeometrically** sampled. Example:

   ```
   iqtree -s example.cf -m HKY+P+WH
   ```

3. *Sampled*: randomly draw N samples with replacement from the given data. The N picked samples constitute a PoMo state which will be assigned a likelihood of 1. All other PoMo states have likelihood 0. Example:

   ```
   iqtree -s example.cf -m HKY+P+S
   ```

We expect a slight overestimation of the heterozygosity for *weighted binomial* sampling. This is because monomorphic (fixed) states can be reached from polymorphic states during the sampling step, while polymorphic states cannot be reached from monomorphic states (sampling does not involve mutation). I.e., only when the level of heterozygosity at the leaves is overestimated, the sampling step leads to the correct amount of heterozygosity observed in the data.

If you wish to avoid this effect, use *weighted hypergeometric* sampling. However, we have observed that *weighted binomial* sampling is more stable.

14.7 State frequency type

Similar to standard models, the state frequency type can be selected with `+F` model string modifiers. The default is to set the state frequencies (i.e., the frequencies of the nucleotides A, C, G and T) to the observed values in the data (empirical value). To estimate the allele frequencies together with the rate parameters during maximization of the likelihood, use:

```
iqtree -s example.cf -m GTR+P+FO
```

14.8 Rate heterogeneity

Recently, PoMo allows inference with different rate categories. As of version `1.6`, only discrete Gamma rate heterogeneity is supported. Please be aware, that for biological and mathematical reasons (we cannot simply scale the full transition matrix but have to separate the mutational component from genetic drift), the run time scales linearly with the number of rate categories. In the future, we plan to work on decreasing run time as well as implement more rate heterogeneity types. To use a discrete Gamma model with 4 rate categories, use:

```
iqtree -s example.cf -m HKY+P+G4
```

14.9 Bootstrap branch support

Bootstrapping works as expected with PoMo. The standard non-parametric bootstrap is invoked by the `-b` option, e.g., for 100 replicates:

```
iqtree -s example.cf -m HKY+P -b 100
```

To overcome the computational burden required by the non-parametric bootstrap, IQ-TREE introduces an ultra fast bootstrap approximation (UFBoot) that is orders of magnitude faster than the standard procedure and provides relatively unbiased branch support values. To run UFBoot, use the option `-bb`, e.g., for 1000 replicates:

```
iqtree -s example.cf -m HKY+P -bb 1000
```

For a detailed description, please refer to the bootstrap tutorial.

14.10 Interpretation of branch lengths

PoMo estimates the branch length in number of mutations and frequency shifts (drift) per site. The number of drift events compared to the number of mutations becomes higher if the virtual population size is increased. To get the branch length measured in number of substitutions per site which enables a comparison to the branch length estimated by standard DNA substitution models, it has to be divided by `N^2`. PoMo also outputs the total tree length measured in number of substitutions per site in `example.cf.iqtree`. An example of the relevant section:

```
NOTE: The branch lengths of PoMo measure mutations and frequency
   shifts.
```

```
To compare PoMo branch lengths to DNA substitution models use the
    tree length
measured in substitutions per site.

Total tree length (sum of branch lengths)
 - measured in number of mutations and frequency shifts per site:
     0.71200751
 - measured in number of substitutions per site (divided by N^2):
     0.00879022
Sum of internal branch lengths
- measured in mutations and frequency shifts per site: 0.01767814
    (2.48285810% of tree length)
- measured in substitutions per site: 0.01767814 (2.48285810% of
   tree length)
```

Chapter 15

Compilation guide

For advanced users to compile IQ-TREE source code.

15.1 General requirements

- A C++ compiler such as GCC (version >= 4.8), Clang, MS Visual Studio and Intel C++ compiler.
- CMake version >= 2.8.10.
- Boost library for IQ-TREE version 2. Boost library is typically available under Linux. Under MacOS you use Homebrew and run `brew install boost` to install the Boost library. By default IQ-TREE will detect the path to the installed Boost library.
- Eigen3 library (for IQ-TREE version >= 1.6). Under MacOS you use Homebrew and run `brew install eigen` to install the Boost library. By default IQ-TREE will detect the path to the installed Eigen3 library. If this failed, you have to manually specify `-DEIGEN3_INCLUDE_DIR=<installed_eigen3_dir>` to the `cmake` command (see below).
- OpenMP library, which is used to compile the multicore version. This should typically be the case with `gcc` under Linux. Under MacOS you use Homebrew and run `brew install libomp` to install the OpenMP library.
- (*Optional*) Install git if you want to clone source code from IQ-TREE GitHub repository.

15.2 Downloading source code

Choose the source code (`zip` or `tar.gz`) of the IQ-TREE release you want to use from:

https://github.com/iqtree/iqtree2/releases

For IQ-TREE version 1 please use:

https://github.com/Cibiv/IQ-TREE/releases/

Alternatively, if you have `git` installed, you can also clone the source code from GitHub with:

```
git clone https://github.com/iqtree/iqtree2.git
```

For IQ-TREE version 1 please clone:

```
git clone https://github.com/Cibiv/IQ-TREE.git
```

Please find below separate compilation guide fors Linux, Mac OS X, and Windows as well as for 32-bit version or for MPI version.

15.3 Compiling under Linux

TIP: Ready made IQ-TREE packages are provided for Debian and Arch Linux (AUR).

1. Open a Terminal.
2. Change to the source code folder:

   ```
   cd PATH_TO_EXTRACTED_SOURCE_CODE
   ```

3. Create a subfolder, say, `build` and go into this subfolder:

   ```
   mkdir build
   cd build
   ```

4. Configure source code with CMake:

   ```
   cmake ..
   ```

If `cmake` failed with message about `Eigen3 not found`, then install Eigen3 library and run `cmake` again. If this still failed, you have to manually specify the downloaded directory of Eigen3 with:

```
cmake -DEIGEN3_INCLUDE_DIR=<eigen3_dir> ..
```

5. Compile source code with `make`:

```
make -j
```

`j` option tells it to use all CPU cores to speed up the compilation. Without this option, `make` uses only one core, which might be slow.

This creates an executable `iqtree2` (`iqtree` for version 1). It can be copied to your system search path so that IQ-TREE can be called from the Terminal simply with the command line `iqtree2`.

TIP: The above guide typically compiles IQ-TREE with `gcc`. If you have Clang installed and want to compile with Clang, the compilation will be similar to Mac OS X like below.

15.4 Compiling under Mac OS X

TIP: A ready made IQ-TREE package is provided by * Homebrew by simply running `brew install homebrew/science/iqtree2`.

- Make sure that Clang compiler is installed, which is typically the case if you installed Xcode and the associated command line tools.
- If you installed cmake with Homebrew
- Find the path to the CMake executable, which is typically `/Applications/CMake .app/Contents/bin/cmake`. For later convenience, please create a symbolic link `cmake` to this cmake executable, so that cmake can be invoked from the Terminal by simply entering `cmake`.

The steps to compile IQ-TREE are similar to Linux (see above), except that you need to specify `clang` as compiler when configuring source code with CMake (step 4):

```
cmake -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ ..
```

(please change `cmake` to absolute path like `/Applications/CMake.app/Contents/bin/cmake`).

To compile the multicore version, the default installed Clang unfortunately does not support OpenMP (which might change in the near future). However, the latest Clang 3.7 supports OpenMP, which can be downloaded from http://clang.llvm.org. After that you can run CMake with:

```
cmake -DIQTREE_FLAGS=omp -DCMAKE_C_COMPILER=clang-3.7 -
   DCMAKE_CXX_COMPILER=clang++-3.7 ..
```

(assuming that `clang-3.7` and `clang++-3.7` points to the installed Clang 3.7).

15.5 Compiling under Windows

- Please first install TDM-GCC (a GCC version for Windows) from http://tdm-gcc.tdragon.net.
- Then install Clang for Windows from http://clang.llvm.org.

WARNING: Although IQ-TREE can also be built with TDM-GCC, the executable does not run properly due to stack alignment issue and the `libgomp` library causes downgraded performance for the OpenMP version. Thus, it is recommended to compile IQ-TREE with Clang.

1. Open Command Prompt.
2. Change to the source code folder:

   ```
   cd PATH_TO_EXTRACTED_SOURCE_CODE
   ```

   Please note that Windows uses back-slash (\) instead of slash (/) as path name separator.
3. Create a subfolder, say, `build` and go into this subfolder:

   ```
   mkdir build
   cd build
   ```

4. Configure source code with CMake:

```
cmake -G "MinGW Makefiles" -DCMAKE_C_COMPILER=clang -
   DCMAKE_CXX_COMPILER=clang++ -DCMAKE_C_FLAGS=--target=
   x86_64-pc-windows-gnu -DCMAKE_CXX_FLAGS=--target=x86_64-
   pc-windows-gnu -DCMAKE_MAKE_PROGRAM=mingw32-make ..
```

To build the multicore version please add `-DIQTREE_FLAGS=omp` to the cmake command. Note that the make program shipped with TDM-GCC is called `mingw32-make`, thus needed to specify like above. You can also copy `mingw32-make` to `make` to simplify this step.

5. Compile source code with:

```
mingw32-make
```

or

```
mingw32-make -j4
```

to use 4 cores for compilation instead of only 1.

15.6 Compiling 32-bit version

NOTE: Typically a 64-bit IQ-TREE version is built and recommended! The 32-bit version has several restriction like maximal RAM usage of 2GB and no AVX support, thus not suitable to analyze large data sets.

To compile the 32-bit version instead, simply add `m32` into `IQTREE_FLAGS` of the cmake command:

```
cmake -DIQTREE_FLAGS=m32 ..
```

To build the 32-bit multicore version, run:

```
cmake -DIQTREE_FLAGS=omp-m32 ..
```

For Windows you need to change Clang target with:

```
cmake -G "MinGW Makefiles" -DCMAKE_C_FLAGS=--target=i686-pc-
   windows-gnu -DCMAKE_CXX_FLAGS=--target=i686-pc-windows-gnu -
   DCMAKE_MAKE_PROGRAM=mingw32-make ..
```

15.7 Compiling MPI version

Requirements:

- Download source code of IQ-TREE version 1.5.1 or later.
- Install an MPI library (e.g., OpenMPI) if not available in your system. For Mac OS X, the easiest is to install Homebrew package manager, and then install OpenMPI library from the command line with:

```
brew install openmpi
```

Then simply run `CMake` and `make` by:

```
cmake -DIQTREE_FLAGS=mpi ..
make -j4
```

IQ-TREE will automatically detect and setup the MPI paths and library. Alternatively, you can also use the MPI C/C++ compiler wrappers (typically named `mpicc` and `mpicxx`), for example:

```
cmake -DCMAKE_C_COMPILER=mpicc -DCMAKE_CXX_COMPILER=mpicxx ..
make -j4
```

The executable is named `iqtree-mpi`. One can then run `mpirun` to start the MPI version with e.g. 2 processes:

```
mpirun -np 2 iqtree-mpi -s alignment ...
```

If you want to compile the hybrid MPI/OpenMP version, simply run:

```
cmake -DIQTREE_FLAGS=omp-mpi ..
make -j4
```

The resulting executable is then named `iqtree-mpi` (`iqtree-omp-mpi` for IQ-TREE versions <= 1.5.X). This can be used to start an MPI run with e.g. 4 processes and 2 cores each (i.e., a total of 8 cores will be used):

```
# For IQ-TREE version <= 1.5.X
mpirun -np 4 iqtree-omp-mpi -nt 2 -s alignment ...

# For IQ-TREE version >= 1.6.0
mpirun -np 4 iqtree-mpi -nt 2 -s alignment ...
```

NOTE: Please be aware that OpenMP and OpenMPI are different! OpenMP is the standard to implement shared-memory multithreading program, that we use to provide the multicore IQ-TREE version. Whereas OpenMPI is a message passing interface (MPI) library for distributed memory parallel system, that is used to compile `iqtree-mpi`. Thus, **one cannot run `iqtree` with `mpirun`!**

15.8 Compiling Xeon Phi Knights Landing version

Starting with version 1.6, IQ-TREE supports Xeon Phi Knights Landing (AVX-512 instruction set). To build this version the following requirements must be met:

- A C++ compiler, which supports AVX-512 instruction set: GCC 5.1, Clang 3.7, or Intel compiler 14.0.

The compilation steps are the same except that you need to add `-DIQTREE_FLAGS=KNL` to the cmake command:

```
cmake -DIQTREE_FLAGS=KNL ..
make -j4
```

The compiled `iqtree` binary will automatically choose the proper computational kernel for the running computer. Thus, it works as normal and will speed up on Knights Landing CPUs. Run `./iqtree` to make sure that the binary was compiled correctly:

```
IQ-TREE multicore Xeon Phi KNL version 1.6.beta for Linux 64-bit
   built May  7 2017
```

15.9 Compling with deep learning kernel for ModelFinder 2

IQ-TREE version 2.2.x supports deep learning to speed up ModelFinder 2. To compile you will need to install the onnxruntime library. On a MacOS, the fastest way is via homebrew package manager:

```
brew install onnxruntime
```

This will install the necessary header files in

```
/usr/local/Cellar/onnxruntime/1.11.0/include/onnxruntime/core/
   session/
```

and the library file in:

```
/usr/local/Cellar//onnxruntime/1.11.0/lib/
```

where 1.11.0 is the version of onnxruntime at the time of writing this document. You may need the version number which can be found by:

```
brew info onnxruntime
```

Now you will need to run cmake by additional options:

```
cmake -Donnxruntime_INCLUDE_DIRS=/usr/local/Cellar//onnxruntime
   /1.11.0/include/onnxruntime/core/session/ -
   Donnxruntime_LIBRARIES=/usr/local/Cellar//onnxruntime/1.11.0/
   lib/libonnxruntime.dylib ..
```

15.10 About precompiled binaries

To provide the pre-compiled IQ-TREE binaries at http://www.iqtree.org, we used Clang 3.9.0 for Windows and Clang 4.0 for Linux and macOS. We recommend to use Clang instead of GCC as Clang-compiled binaries run about 5-10% faster than GCC-compiled ones.

Linux binaries were statically compiled with Ubuntu 16.4 using libc++ library. The static-linked binaries will thus run on most Linux distributions. The CMake command is (assuming that clang-4 and clang++-4 point to the installed Clang):

```
# 64-bit version
cmake -DIQTREE_FLAGS=static-libcxx -DCMAKE_C_COMPILER=clang-4 -
   DCMAKE_CXX_COMPILER=clang++-4 <source_dir>

# 32-bit version
cmake -DIQTREE_FLAGS=static-m32 -DCMAKE_C_COMPILER=clang-4 -
   DCMAKE_CXX_COMPILER=clang++-4 <source_dir>
```

macOS binaries were compiled under macOS Sierra, but the binaries are backward compatible with Mac OS X 10.7 Lion:

```
cmake -DCMAKE_C_COMPILER=clang-4 -DCMAKE_CXX_COMPILER=clang++-4 <
   source_dir>
```

Windows binaries were statically compiled under Windows 7 using Clang 3.9.0 in combination with TDM-GCC 5.1.0, which provides the neccessary libraries for Clang.

```
# 64-bit version
cmake -G "MinGW Makefiles" -DIQTREE_FLAGS=static -
   DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ -
   DCMAKE_C_FLAGS=--target=x86_64-pc-windows-gnu -
   DCMAKE_CXX_FLAGS=--target=x86_64-pc-windows-gnu -
   DCMAKE_MAKE_PROGRAM=mingw32-make ..

#32-bit version
cmake -G "MinGW Makefiles" -DIQTREE_FLAGS=static -
   DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ -
   DCMAKE_C_FLAGS=--target=i686-pc-windows-gnu -DCMAKE_CXX_FLAGS
   =--target=i686-pc-windows-gnu -DCMAKE_MAKE_PROGRAM=mingw32-
   make ..
```

15.11 Setup an Xcode project in MacOS

Many developers in MacOS use Xcode to develop the code. To generate an XCode project for IQ-TREE, you need to run:

```
mkdir build-xcode
cd build-xcode
cmake -G Xcode <IQTREE_SOURCE_DIR>
```

15.12 This will generate a a subfolder `build-xcode/iqtree.xco` which you can open in Xcode now.

layout: userdoc title: “Frequently Asked Questions” author: Heiko Schmidt, Jana Trifinopoulos, Minh Bui, Rob Lanfear date: 2023-09-06 docid: 9 icon: question-circle doctype: manual tags: - manual description: For common questions and answers. sections: - name: How do I get help? url: how-do-i-get-help - name: How do I report bug? url: how-do-i-report-bug - name: How to interpret ultrafast bootstrap (UFBoot) supports? url: how-do-i-interpret-ultrafast-bootstrap-ufboot-support-values - name: How does IQ-TREE treat gap/missing/ambiguous characters? url: how-does-iq-tree-treat-gapmissingambiguous-characters - name: Can I mix DNA and protein data in a partitioned analysis? url: can-i-mix-dna-and-protein-data-in-a-partitioned-analysis - name: What is the interpretation of branch lengths when mixing codon and DNA data? url: what-is-the-interpretation-of-branch-lengths-when-mixing-codon-and-dna-data - name: What is the purpose of composition test? url: what-is-the-purpose-of-composition-test - name: What is the good number of CPU cores to use url:

what-is-the-good-number-of-cpu-cores-to-use - name: How do I save time for standard bootstrap? url: how-do-i-save-time-for-standard-bootstrap - name: Why does IQ-TREE complain about the use of +ASC model? url: why-does-iq-tree-complain-about-the-use-of-asc-model - name: How does IQ-TREE treat identical sequences? url: how-does-iq-tree-treat-identical-sequences —

Chapter 16

Frequently asked questions

For common questions and answers.

16.1 How do I get help?

If you have questions please follow the steps below:

1. Continue to read the FAQ below, which may answer your questions already.
2. If not, read the documentation http://www.iqtree.org/doc.
3. If you still could not find the answer, search the IQ-TREE Google group. There is a "Search for topics" box at the top of the Google group web page.
4. Finally, if no answer is found, post a question to the IQ-TREE group. The average response time is one to two working days.

For other feedback and feature requests, please post a topic to the IQ-TREE Google group. We welcome all suggestions to further improve IQ-TREE! For feature request, please also explain why you think such a new feature would be useful or how can it help for your work.

16.2 How do I report a bug?

For bug report, please send the following information to the IQ-TREE Google group:

1. A description of the behaviour, which you think might be unexpected or caused by a bug.
2. The first 10 lines and last 10 lines of the `.log` file.

3. (If possible) the assertion message printed on the screen, which may look like this:

```
iqtree: ....cpp:140: ...: Assertion '...' failed.
```

The development team will get back to you and may ask for the full `.log` file and input data files for debugging purpose, if necessary. In such case please **only send your data files directly to the developers for confidential reason**! Keep in mind that everyone can see all emails sent to the group!

16.3 How do I interpret ultrafast bootstrap (UFBoot) support values?

The ultrafast bootstrap (UFBoot) feature (`-bb` option) was published in (Minh et al., 2013). One of the main conclusions is, that UFBoot support values are more unbiased: 95% support correspond roughly to a probability of 95% that a clade is true. So this has a different meaning than the normal bootstrap supports (where you start to believe in the clade if it has >80% BS support). For UFBoot, you should only start to believe in a clade if its support is >= 95%. Thus, the interpretations are different and you should not compare BS% with UFBoot% directly.

Moreover, it is recommended to also perform the SH-aLRT test (Guindon et al., 2010) by adding `-alrt 1000` into the IQ-TREE command line. Each branch will then be assigned with SH-aLRT and UFBoot supports. One would typically start to rely on the clade if its SH-aLRT >= 80% and UFboot >= 95%.

16.4 How does IQ-TREE treat gap/missing/ambiguous characters?

Gaps (`-`) and missing characters (`?` or `N` for DNA alignments) are treated in the same way as `unknown` characters, which represent no information. The same treatment holds for many other ML software (e.g., RAxML, PhyML). More explicitly, for a site (column) of an alignment containing `AC-AG-A` (i.e. A for sequence 1, C for sequence 2, - for sequence 3, and so on), the site-likelihood of a tree T is equal to the site-likelihood of the subtree of T restricted to those sequences containing non-gap characters (`ACAGA`).

Ambiguous characters that represent more than one character are also supported: each represented character will have equal likelihood. For DNA the following ambigous nucleotides are supported according to IUPAC nomenclature:

Nucleotide	Meaning
R	A or G (purine)
Y	C or T (pyrimidine)
W	A or T (weak)
S	G or C (strong)
M	A or C (amino)
K	G or T (keto)
B	C, G or T (next letter after A)
H	A, C or T (next letter after G)
D	A, G or T (next letter after C)
V	A, G or C (next letter after T)
?, -, ., ~, !, O, N, X	A, G, C or T (unknown; all 4 nucleotides are equally likely)

For protein sequences the following ambiguous amino-acids are supported:

Amino-acid	Meaning
B	N or D
Z	Q or E
J	I or L
U	unknown AA (although it is the 21st AA)
?, -, ., ~, *, ! or X	unknown AA (all 20 AAs are equally likely)

The letters `*` and `!` may found in alignments of protein and/or coding DNA sequences. Stop codon is typically translated to `*`. Some alignment programs also mark frameshift mutations (cf. Ranwez et al., 2011), that means since frameshift mutations in a codon alignment cause incomplete codons that cannot be unambiguously translated the resulting position in the translated protein sequence and padding positions in the respective codon are marked using `!`.

16.5 Can I mix DNA and protein data in a partitioned analysis?

Yes! You can specify this via a NEXUS partition file. In fact, you can mix any data types supported in IQ-TREE, including also codon, binary and morphological data. To do so, each data type should be stored in a separate alignment file (see also Partitioned analysis with mixed data). As an example, assuming `dna.phy` is a DNA

alignment and and `prot.phy` is a protein alignment. Then a partition file mixing two types of data can be specified as follows:

```
#nexus
begin sets;
    charset part1 = dna.phy: 1-100 201-300;
    charset part2 = dna.phy: 101-200;
    charset part3 = prot.phy: 1-150;
    charset part4 = prot.phy: 151-400;
    charpartition mine = HKY:part1, GTR+G:part2, WAG+I+G:part3,
       LG+G:part4;
end;
```

NOTE: The site count for each alignment should start from 1, and **not** continue from the last position of a previous alignment (e.g., see `part3` and `part4` declared above).

16.6 What is the interpretation of branch lengths when mixing codon and DNA data?

When mixing codon and DNA data in a partitioned analysis, the branch lengths are interpreted as the number of nucleotide substitutions per nucleotide site! This is different from having only codon data, where branch lengths are the number of nucleotide substitutions per codon site (thus typically 3 times longer than under DNA models).

Note that if you mix codon, DNA and protein data, the branch lengths are then the number of character substitutions per site, where character is either nucleotide or amino-acid.

16.7 What is the purpose of composition test?

At the beginning of each run, IQ-TREE performs a composition chi-square test for every sequence in the alignment. The purpose is to test for homogeneity of character composition (e.g., nucleotide for DNA, amino-acid for protein sequences). A sequence is denoted `failed` if its character composition significantly deviates from the average composition of the alignment.

More specifically, for each sequence, compute:

```
chi2 = \sum_{i=1}^k (O_i - E_i)^2 / E_i
```

where k is the size of the alphabet (e.g. 4 for DNA, 20 for amino acids) and the values 1 to k correspond uniquely to one of the characters. O_i is the character frequency in the sequence tested. E_i is the overall character frequency from the entire alignment.

Whether the character composition deviates significantly from the overall composition is done by testing the chi2 value using the chi2-distribution with k-1 degrees of freedom (df=3 for DNA or df=19 for amino acids). By and large it is a normal Chi^2 test.

This test should be regarded as an *explorative tool* which might help to nail down problems in a dataset. One would typically not remove failing sequences by default. But if the tree shows unexpected topology the test might point in direction of the origin of the problem.

Furthermore, please keep in mind, this test is performed at the very beginning, where IQ-TREE does not know anything about the models yet. That means:

- If you have partitioned (multi-gene) data, it might be more reasonable to test this separately for each partition in a partition analysis. Here, one might want to be able to decide whether some partitions should better be discarded if it is hard to find a composition representing the sequences in the partition. Or on the other hand if a sequence fails for many partitions and show very unexpected phylogenetic topologies, try without it.
- If you have (phylogenomic) protein data, you can also try several protein mixture models, which account for different amino-acid compositions along the sequences, for example, the `C10` to `C60` profile mixture models.
- Finally, it is recommended to always check the alignment (something one should always do anyway), especially if they have been collected and produced automatically.

16.8 What is the good number of CPU cores to use?

Starting with version 1.5.1, you can use option `-nt AUTO` to automatically determine the best number of threads for your current data and computer.

If you want to know more details: IQ-TREE can utilize multicore machines to speed up the analysis via `-nt` option. However, it does not mean that using more cores will always result in less running time: if your alignment is short, using too many cores may even slow down the analysis. This is because IQ-TREE parallelizes the likelihood computation along the alignment. Thus, the parallel efficiency is only increased with longer alignments.

If you want to restrict the number of CPU cores allocated by `-nt AUTO` use `-ntmax` to specify the maximal number of CPU cores allowed (DEFAULT: #CPU cores on the current machine).

16.9 How do I save time for standard bootstrap?

The standard bootstrap is rather slow and may take weeks/months for large data sets. One way to speed up is to use the multicore version. However, this only works well for long alignments (see What is the good number of CPU cores to use?). Another way is to use many machines or a computing cluster and split the computation among the machines. To illustrate, you want to perform 100 bootstrap replicates and have 5 PCs, each has 4 CPU cores. Then you can:

1. Perform 5 independent bootstrap runs (each with 20 replicates) on the 5 machines with 5 prefix outputs (such that output files are not overwritten). For example:

```
# For old IQ-TREE versions <= 1.5.X, change iqtree to
   iqtree-omp
iqtree -nt 4 -s input_alignment -bo 20 ... -pre boot1
iqtree -nt 4 -s input_alignment -bo 20 ... -pre boot2
iqtree -nt 4 -s input_alignment -bo 20 ... -pre boot3
iqtree -nt 4 -s input_alignment -bo 20 ... -pre boot4
iqtree -nt 4 -s input_alignment -bo 20 ... -pre boot5
```

   Note that if you have access to a computing cluster, you may want to submit these jobs onto the cluster queue in parallel and with even more fined grained parallelization (e.g. one replicate per job).

2. Once all 5 runs finished, combine the 5 `.boottrees` file into one file (e.g. by `cat` command under Linux):

```
cat boot*.boottrees > alltrees
```

3. Construct a consensus tree from the combined bootstrap trees:

```
iqtree -con -t alltrees
```

   The consensus tree is then written to `.contree` file.

4. Estimate branch lengths of the consensus tree using the original alignment:

```
iqtree -s input_alignment -te alltrees.contree -pre
   alltrees.contree
```

5. You can also perform the analysis on the original alignment:

```
# For old IQ-TREE versions <= 1.5.X, change iqtree to
   iqtree-omp
iqtree -nt 4 -s input_alignment ...
```

and map the support values onto the obtained ML tree:

```
iqtree -sup input_alignment.treefile -t alltrees
```

The ML tree with assigned bootstrap supports is written to `.suptree` file.

16.10 Why does IQ-TREE complain about the use of +ASC model?

When using ascertainment bias correction (ASC) model, sometimes you may get an error message:

```
ERROR: Invaid use of +ASC because of ... invariant sites in the alignment
```

or when performing model testing:

```
Skipped since +ASC is not applicable
```

This is because your alignment contains *invariant* sites (columns), which violate the mathematical condition of the model. The invariant sites can be:

- Constant sites: containing a single character state over all sequences. For example, all sequences show an `A` (Adenine) at a particular site in a DNA alignment.
- Partially constant sites: containing a single character, gap or unknown character. For example, at a particular site some sequences show a `G` (Guanine), some sequences have `-` (gap) and the other have `N`.
- Ambiguously constant sites: For example, some sequences show a `C` (Cytosine), some show a `Y` (meaning `C` or `T`) and some show a `-` (gap).

All these sites must be removed from the alignment before a +ASC model can be applied.

TIP: Starting with IQ-TREE version 1.5.0, an output alignment file with suffix `.varsites` is written in such cases, which contain only variable sites from the input alignment. The `.varsites` alignment can then be used with the +ASC model.

16.11 How does IQ-TREE treat identical sequences?

Among a group of identical sequences, IQ-TREE will keep the first two and ignore the rest. If the sequence is the 2nd one, it will be "kept for subsequent analysis". If it is the 3rd or more, it will be "ignored but added at the end". The rationale for this is to still be able to calculate the bootstrap support for this group of identical sequences: it is not always 100%. Because by bootstrap resampling, on average only two third of the sites will be present in a bootstrap alignment (due to sampling with replacement), and suddenly another sequence not in this group may actually become identical to this group of sequences. In that case, the bootstrap value will be $< 100\%$.

Therefore, the `.uniqueseq.phy` printed by IQ-TREE may still contain the identical sequences, but no more than two of each identical group.

16.12 What are the differences between alignment columns/sites and patterns?

Columns are the columns/sites in the alignment and the number of columns is the length of the alignment. In the alignment there might be the same columns. Different columns are called patterns. While (parsimony) informative sites are patterns that have at least two different characters (nucleotides or amino acids) and each character should occur in at least two species. Essentially, informative sites have information for the grouping of species. These patterns are mainly important in the context of parsimony, where no evolutionary model is used. In maximum likelihood inference all patterns containing different characters are important for the estimation of tree topology and branch lengths, while constant/invariant sites (containing only the same character: only A's or only G's etc) are important for the correct estimation of the branch lengths. Therefore, should not be excluded from the alignment.

Example:

```
            123456789
species_1   AACGTACGT
species_2   AACGATCGT
species_3   AACCGTCCT
species_4   AACCTACCT
```

- sites/columns 1 and 2 are identical and contain only A's - invariant site pattern, uninformative

- sites/columns 3 and 7 are identical and contain only C's - invariant site pattern, uninformative
- sites/columns 4 and 8 are identical and contain 2 G'c and 2 C's - informative site pattern
- sites/columns 5, 6 and 9 occur only once, each site is a different pattern. 5th is uninformative, 6th is informative, 9th is invariant and uninformative

Summing up, the alignment has 9 columns,

6 patterns

```
A C G T A T
A C G A T T
A C C G T T
A C C T A T
```

(2 informative)

```
G A
G T
C T
C A
```