

IQ-TREE version 1.5.5: Tutorials and Manual  
Phylogenomic software by maximum likelihood

<http://www.iqtree.org>

Bui Quang Minh, Jana Trifinopoulos, Dominik Schrempf, Heiko A. Schmidt

October 25, 2017

**Preface**

# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction</b>                     | <b>9</b>  |
| 1.1      | Why IQ-TREE? . . . . .                  | 9         |
| 1.2      | Key features . . . . .                  | 10        |
| 1.3      | Free web server . . . . .               | 11        |
| 1.4      | User support . . . . .                  | 11        |
| 1.5      | Documentation . . . . .                 | 11        |
| 1.6      | How to cite IQ-TREE? . . . . .          | 12        |
| 1.7      | Development team . . . . .              | 13        |
| 1.8      | Credits and acknowledgements . . . . .  | 13        |
| <b>2</b> | <b>Getting started</b>                  | <b>15</b> |
| 2.1      | IQ-TREE web server . . . . .            | 15        |
| 2.2      | Installation . . . . .                  | 15        |
| 2.2.1    | Packages and bundles . . . . .          | 15        |
| 2.2.2    | Manual download . . . . .               | 16        |
| 2.3      | For Windows users . . . . .             | 16        |
| 2.4      | For Mac OS X users . . . . .            | 17        |
| 2.5      | Minimal command-line examples . . . . . | 19        |
| 2.6      | Where to go from here? . . . . .        | 21        |
| <b>3</b> | <b>Web server tutorial</b>              | <b>23</b> |
| 3.1      | Tree Inference . . . . .                | 23        |
| 3.2      | Model Selection . . . . .               | 25        |
| 3.3      | Analysis Results . . . . .              | 26        |
| <b>4</b> | <b>Beginner's tutorial</b>              | <b>29</b> |
| 4.1      | Input data . . . . .                    | 29        |
| 4.2      | First running example . . . . .         | 30        |

|          |  |           |
|----------|--|-----------|
| 4.3      | Choosing the right substitution model . . . . .                            | 33        |
| 4.4      | Using codon models . . . . .   | 35        |
| 4.5      | Binary, morphological and SNP data . . . . .                               | 36        |
| 4.6      | Assessing branch supports with ultrafast bootstrap approximation . . . . . | 37        |
| 4.7      | Reducing impact of severe model violations with UFBoot . . . . .           | 38        |
| 4.8      | Assessing branch supports with standard nonparametric bootstrap . . . . .  | 39        |
| 4.9      | Assessing branch supports with single branch tests . . . . .               | 39        |
| 4.10     | Utilizing multi-core CPUs . . . . .  | 41        |
| 4.11     | Where to go from here? . . . . .   | 41        |
| <b>5</b> | <b>Advanced tutorial</b>   | <b>43</b> |
| 5.1      | Partitioned analysis for multi-gene alignments . . . . .                   | 43        |
| 5.2      | Partitioned analysis with mixed data . . . . .                             | 45        |
| 5.3      | Choosing the right partitioning scheme . . . . .                           | 46        |
| 5.4      | Ultrafast bootstrapping with partition model . . . . .                     | 47        |
| 5.5      | Constrained tree search . . . . .  | 48        |
| 5.6      | Tree topology tests . . . . .  | 49        |
| 5.7      | Testing constrained tree . . . . .   | 51        |
| 5.8      | Consensus construction and bootstrap value assignment . . . . .            | 53        |
| 5.9      | User-defined substitution models . . . . .                                 | 54        |
| 5.10     | Inferring site-specific rates . . . . .                                    | 55        |
| 5.11     | Where to go from here? . . . . .   | 56        |
| <b>6</b> | <b>Command reference</b>   | <b>57</b> |
| 6.1      | General options . . . . .  | 57        |
| 6.1.1    | Example usages: . . . . .  | 59        |
| 6.2      | Checkpointing to resume stopped run . . . . .                              | 59        |
| 6.3      | Likelihood mapping analysis . . . . .                                      | 60        |
| 6.3.1    | Example usages: . . . . .  | 61        |
| 6.4      | Automatic model selection . . . . .  | 61        |
| 6.4.1    | Example usages: . . . . .  | 64        |
| 6.5      | Specifying substitution models . . . . .                                   | 64        |
| 6.5.1    | Example usages: . . . . .  | 65        |
| 6.6      | Rate heterogeneity . . . . .   | 66        |
| 6.7      | Partition model options . . . . .  | 67        |
| 6.8      | Site-specific frequency model options . . . . .                            | 67        |
| 6.9      | Tree search parameters . . . . .   | 68        |
| 6.9.1    | Example usages: . . . . .  | 69        |

|          |  |           |
|----------|--|-----------|
| 6.10     | Ultrafast bootstrap parameters . . . . .                           | 69        |
| 6.10.1   | Example usages: . . . . .  | 70        |
| 6.11     | Nonparametric bootstrap . . . . .                                  | 70        |
| 6.12     | Single branch tests . . . . .                                      | 71        |
| 6.12.1   | Example usages: . . . . .  | 71        |
| 6.13     | Tree topology tests . . . . .                                      | 72        |
| 6.13.1   | Example usages: . . . . .  | 73        |
| 6.14     | Constructing consensus tree . . . . .                              | 73        |
| 6.15     | Computing Robinson-Foulds distance . . . . .                       | 74        |
| 6.15.1   | Example usages: . . . . .  | 74        |
| 6.16     | Generating random trees . . . . .                                  | 74        |
| 6.16.1   | Example usages: . . . . .  | 75        |
| 6.17     | Miscellaneous options . . . . .                                    | 75        |
| <b>7</b> | <b>Substitution models</b>   | <b>77</b> |
| 7.1      | DNA models . . . . .   | 77        |
| 7.1.1    | Base substitution rates . . . . .                                  | 77        |
| 7.1.2    | Base frequencies . . . . .   | 79        |
| 7.1.3    | Lie Markov models . . . . .  | 79        |
| 7.2      | Protein models . . . . .   | 81        |
| 7.2.1    | Amino-acid exchange rate matrices . . . . .                        | 81        |
| 7.2.2    | Protein mixture models . . . . .                                   | 82        |
| 7.2.3    | User-defined empirical protein models . . . . .                    | 83        |
| 7.2.4    | Amino-acid frequencies . . . . .                                   | 84        |
| 7.3      | Codon models . . . . .   | 85        |
| 7.3.1    | Codon substitution rates . . . . .                                 | 86        |
| 7.3.2    | Codon frequencies . . . . .  | 87        |
| 7.4      | Binary and morphological models . . . . .                          | 87        |
| 7.5      | Ascertainment bias correction . . . . .                            | 88        |
| 7.6      | Rate heterogeneity across sites . . . . .                          | 88        |
| <b>8</b> | <b>Complex models</b>  | <b>91</b> |
| 8.1      | Partition models . . . . .   | 91        |
| 8.1.1    | Partition file format . . . . .                                    | 92        |
| 8.1.2    | Partitioned analysis . . . . .                                     | 94        |
| 8.2      | Mixture models . . . . .   | 94        |
| 8.2.1    | What is the difference between partition and mixture models? . . . | 94        |
| 8.2.2    | Defining mixture models . . . . .                                  | 94        |

|           |  |            |
|-----------|--|------------|
| 8.2.3     | Profile mixture models . . . . .   | 95         |
| 8.2.4     | NEXUS model file . . . . .   | 96         |
| 8.3       | Site-specific frequency models . . . . .   | 97         |
| 8.3.1     | Example usages . . . . .   | 98         |
| 8.4       | Heterotachy models . . . . .   | 99         |
| 8.4.1     | Download . . . . .   | 99         |
| 8.4.2     | Quick usages . . . . .   | 100        |
| <b>9</b>  | <b>Polymorphism-aware models</b>   | <b>101</b> |
| 9.1       | Counts files . . . . .   | 102        |
| 9.1.1     | Conversion scripts . . . . .   | 103        |
| 9.2       | First running example . . . . .  | 103        |
| 9.3       | Substitution models . . . . .  | 104        |
| 9.4       | Virtual population size . . . . .  | 104        |
| 9.5       | Level of polymorphism . . . . .  | 104        |
| 9.6       | Sampling method . . . . .  | 105        |
| 9.7       | State frequency type . . . . .   | 106        |
| 9.8       | Rate heterogeneity . . . . .   | 106        |
| 9.9       | Bootstrap branch support . . . . .   | 106        |
| 9.10      | Interpretation of branch lengths . . . . .   | 107        |
| 9.11      | Schrempf et al., 2016: <a href="http://dx.doi.org/10.1016/j.jtbi.2016.07.042">http://dx.doi.org/10.1016/j.jtbi.2016.07.042</a> . . . . . | 107        |
| <b>10</b> | <b>Compilation guide</b>   | <b>109</b> |
| 10.1      | General requirements . . . . .   | 109        |
| 10.2      | Downloading source code . . . . .  | 109        |
| 10.3      | Compiling under Linux . . . . .  | 110        |
| 10.4      | Compiling under Mac OS X . . . . .   | 111        |
| 10.5      | Compiling under Windows . . . . .  | 112        |
| 10.6      | Compiling 32-bit version . . . . .   | 113        |
| 10.7      | Compiling MPI version . . . . .  | 114        |
| 10.8      | Compiling Xeon Phi Knights Landing version . . . . .   | 115        |
| 10.9      | About precompiled binaries . . . . .   | 115        |
| <b>11</b> | <b>Frequently asked questions</b>  | <b>117</b> |
| 11.1      | How do I get help? . . . . .   | 117        |
| 11.2      | How do I report bug? . . . . .   | 117        |
| 11.3      | How do I interpret ultrafast bootstrap (UFBoot) support values? . . . . .  | 118        |
| 11.4      | How does IQ-TREE treat gap/missing/ambiguous characters? . . . . .   | 118        |
| 11.5      | Can I mix DNA and protein data in a partitioned analysis? . . . . .  | 119        |

|       |  |     |
|-------|--|-----|
| 11.6  | What is the interpretation of branch lengths when mixing codon and DNA data? . . . . . | 120 |
| 11.7  | What is the purpose of composition test? . . . . .                                     | 120 |
| 11.8  | What is the good number of CPU cores to use? . . . . .                                 | 121 |
| 11.9  | How do I save time for standard bootstrap? . . . . .                                   | 122 |
| 11.10 | Why does IQ-TREE complain about the use of +ASC model? . . . . .                       | 123 |





# 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 collaborators.

The name IQ-TREE comes from the fact that it is the successor of **IQPNNI** 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., in press).
- **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., in press) 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 ultrafast bootstrap (UFBoot) please cite:

- **D.T. Hoang, O. Chernomor, A. von Haeseler, B.Q. Minh, and L.S. Vinh** (2017) UFBoot2: Improving the ultrafast bootstrap approximation. *Mol. Biol. Evol.*, in press. DOI: [10.1093/molbev/msx281](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** (2017) Modeling site heterogeneity with posterior mean site frequency profiles accelerates accurate phylogenomic estimation. *Syst. Biol.*, in press. DOI: [10.1093/sysbio/syx068](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. DOI: [10.1038/nmeth.4285](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. DOI: [10.1093/molbev/msu300](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. DOI: [10.1093/sysbio/syw037](https://doi.org/10.1093/sysbio/syw037)

If you used the polymorphism-aware models please cite:

- **D. Schrempf, B.Q. Minh, N. De Maio, A. von Haeseler, and C. Kosiol** (2016) Reversible polymorphism-aware phylogenetic models and their application to

tree inference. *J. Theor. Biol.*, 407:362–370. DOI: [10.1016/j.jtbi.2016.07.042](https://doi.org/10.1016/j.jtbi.2016.07.042)

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. DOI: [10.1093/nar/gkw256](https://doi.org/10.1093/nar/gkw256)

## 1.7 Development team

IQ-TREE is actively developed by:

**Bui Quang Minh**, *Team leader*, Designs IQ-TREE software core, ultrafast bootstrap, model selection.

**Lam Tung Nguyen**, *Developer*, Designs and parallelizes IQ-TREE search algorithm.

**Olga Chernomor**, *Developer*, Designs partition models and phylogenomic tree search.

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

**Jana Trifinopoulos**, *Developer*, Designs and maintains IQ-TREE web service.

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

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

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

**Arndt von Haeseler**, *Advisor*, Provides advice, inspiring ideas and financial support.

## 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 Library](#), [Eigen library](#), [SPRNG library](#), [Zlib library](#), [gzstream library](#), [vectorclass library](#), [GNU scientific library](#).

IQ-TREE was partially funded by the [Austrian Science Fund - FWF](#) (grant no. I760-B17 from 2012-2015 and I 2508-B29 from 2016-2019) and the [University of Vienna](#) (Initiativkolleg I059-N).



# 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 iqtrees):

- [Debian Linux](#): `sudo apt-get install iqtrees`
- [Arch Linux \(AUR\)](#)
- [Anaconda](#): `conda install -c bioconda iqtrees`
- [Homebrew](#): `brew install homebrew/science/iqtrees`

- [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` or `iqtree-omp-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` or `iqtree-omp` 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.



```

C:\ Command Prompt
Estimate model parameters (epsilon = 0.010)
1. Initial log-likelihood: -23117.037
Optimal log-likelihood: -23117.034
Rate parameters:  A-C: 1.000  A-G: 2.440  A-T: 1.000  C-G: 1.000  C-T: 2.440  G-
T: 1.000
Base frequencies:  A: 0.355  C: 0.228  G: 0.192  T: 0.225
Parameters optimization took 1 rounds (0.062 sec)
BEST SCORE FOUND : -23117.034
Total tree length: 2.805

Total number of iterations: 101
CPU time used for tree search: 2.188 sec (0h:0m:2s)
Wall-clock time used for tree search: 2.578 sec (0h:0m:2s)
Total CPU time used: 2.203 sec (0h:0m:2s)
Total wall-clock time used: 3.125 sec (0h:0m:3s)

Analysis results written to:
  IQ-TREE report:          example.phy.iqtree
  Maximum-likelihood tree: example.phy.treefile
  Likelihood distances:    example.phy.mldist
  Screen log file:         example.phy.log

Date and Time: Mon Oct 24 12:57:42 2016
C:\Users\minh\Downloads\iqtree-1.5.0-Windows>

```

Figure 2.1: Windows command prompt

## 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”.
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).

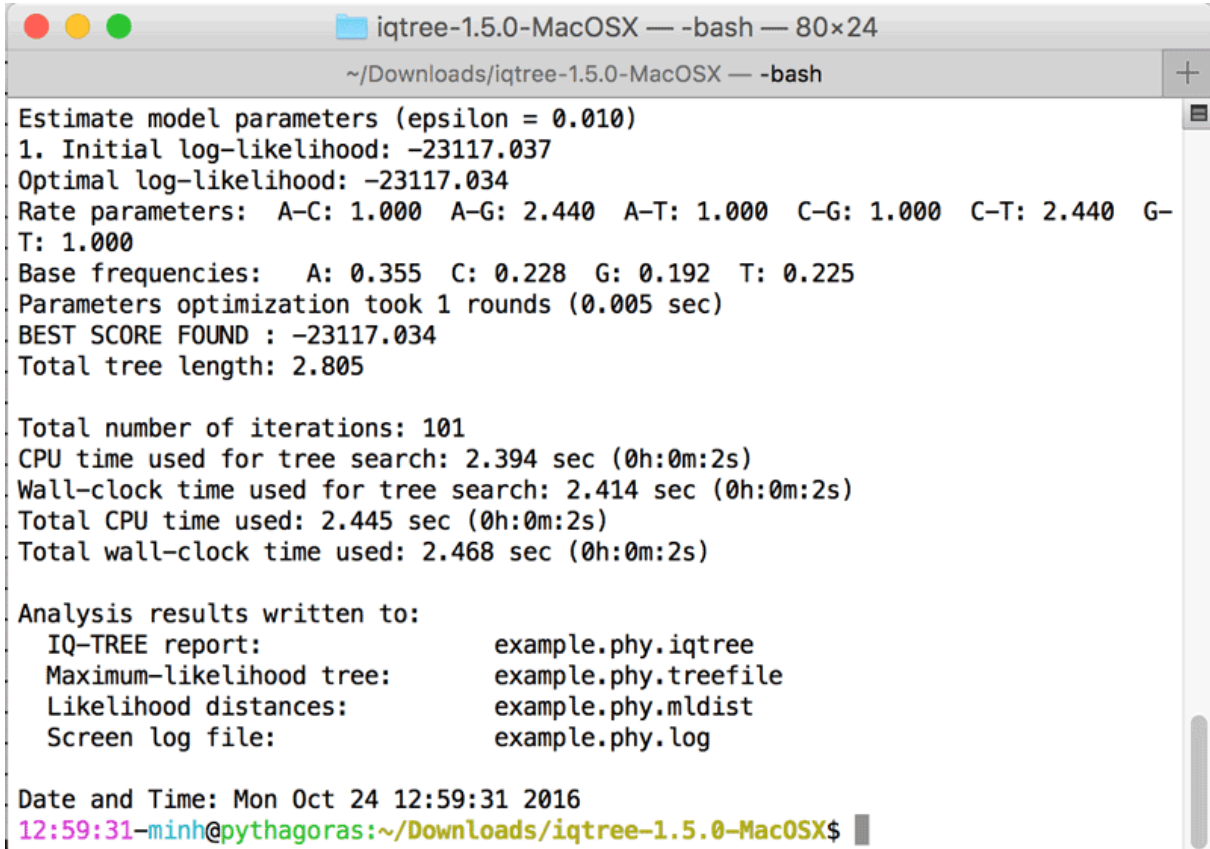
3. Now you can try an example run by entering

```
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.

A screenshot of a Mac terminal window titled 'iqtree-1.5.0-MacOSX — -bash — 80x24'. The window shows the output of the IQ-TREE software. The output includes model parameter estimation results, log-likelihood values, rate parameters, base frequencies, and timing information for the tree search. It also lists the files where the analysis results were written and the date and time of the execution.

```
iqtree-1.5.0-MacOSX — -bash — 80x24
~/Downloads/iqtree-1.5.0-MacOSX — -bash
Estimate model parameters (epsilon = 0.010)
1. Initial log-likelihood: -23117.037
Optimal log-likelihood: -23117.034
Rate parameters: A-C: 1.000 A-G: 2.440 A-T: 1.000 C-G: 1.000 C-T: 2.440 G-
T: 1.000
Base frequencies: A: 0.355 C: 0.228 G: 0.192 T: 0.225
Parameters optimization took 1 rounds (0.005 sec)
BEST SCORE FOUND : -23117.034
Total tree length: 2.805

Total number of iterations: 101
CPU time used for tree search: 2.394 sec (0h:0m:2s)
Wall-clock time used for tree search: 2.414 sec (0h:0m:2s)
Total CPU time used: 2.445 sec (0h:0m:2s)
Total wall-clock time used: 2.468 sec (0h:0m:2s)

Analysis results written to:
  IQ-TREE report:          example.phy.iqtree
  Maximum-likelihood tree: example.phy.treefile
  Likelihood distances:    example.phy.mldist
  Screen log file:         example.phy.log

Date and Time: Mon Oct 24 12:59:31 2016
12:59:31-minh@pythagoras:~/Downloads/iqtree-1.5.0-MacOSX$
```

Figure 2.2: Mac terminal

## 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:

```
# for version >= 1.5.4
iqtree -s example.phy

# for version <= 1.5.3
iqtree -s example.phy -m TESTNEW

(use '-m TEST' to resemble jModelTest/ProtTest)
```

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

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

- Perform ModelFinder without subsequent tree inference:

```
# for version >= 1.5.4
iqtree -s example.phy -m MF

# for version <= 1.5.3
iqtree -s example.phy -m TESTNEWONLY

(use '-m TESTONLY' to resemble jModelTest/ProtTest)
```

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

```
# for version >= 1.5.4
iqtree -s example.phy -alrt 1000 -bb 1000

# for version <= 1.5.3
iqtree -s example.phy -m TESTNEW -alrt 1000 -bb 1000
```

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

```
iqtree -s example.phy -spp example.nex

(replace '-spp' by '-sp' for edge-unlinked model)
```

- Find best partition scheme by possibly merging partitions:

```
# for version >= 1.5.4
iqtree -s example.phy -sp example.nex -m MF+MERGE

# for version <= 1.5.3
iqtree -s example.phy -sp example.nex -m TESTNEWMERGEONLY

(use '-m TESTMERGEONLY' to resemble PartitionFinder)
```

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

```
# for version >= 1.5.4
iqtree -s example.phy -spp example.nex -m MFP+MERGE -bb 1000

# for version <= 1.5.3
iqtree -s example.phy -spp example.nex -m TESTNEWMERGE -bb 1000

(use '-m TESTMERGE' to resemble PartitionFinder)
```

- Use 4 CPU cores to speed up computation:

```
iqtree-omp -s example.phy -nt 4
```

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

```
iqtree-omp -s example.phy -m GTR+R4 -nt AUTO
```

- Show all available options:

```
iqtree -h
```

**WARNING:** All these 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. We plan to implement the edge-linked partition finding in version 1.6.

## 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](http://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., in press) 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.

**IQ-TREE web server: fast and accurate phylogenetic trees under maximum likelihood**

Server load: 9%

Nguyen LT, Schmidt HA, von Haeseler A, Minh BQ (2015) *Mol. Biol. Evol.*, 32:268-274  
 Minh BQ, Nguyen MAT, and von Haeseler A (2013) *Mol. Biol. Evol.* 30:1188-1195

Tree Inference | Model Selection | Analysis Results

### Input Data

**Alignment file :**  [Browse...](#) [Show example >](#)

**Use example alignment:** ☒ Yes [Phylip, Fasta, Nexus, Clustal or MSF format](#)

**Sequence type:** ☒ Auto-detect ☐ DNA ☐ Protein ☐ Codon  
☐ DNA->AA ☐ Binary ☐ Morphology

**Partition file:**  This field is optional. [Browse...](#) [Show example >](#)

**Partition type:** ☒ Edge-linked ☐ Edge-unlinked ?

### Substitution Model Options

**Substitution model:** Auto ?

**FreeRate heterogeneity:** ☐ Yes [+R] ?

**Rate heterogeneity:** ☐ Gamma [+G] ☐ Invar. sites [+I] ?

**#rate categories:** 4 ?

**State frequency:** ☒ Empirical (from data) ☐ AA model (from matrix) ☐ ML-optimized  
☐ Codon F1x4 ☐ Codon F3x4

**Ascertainment bias correction:** ☐ Yes [+ASC] ?

### Branch Support Analysis

**Bootstrap analysis:** ☐ None ☒ Ultrafast ☐ Standard #replicates: 1000 ?

**Create .ufboot file:** ☐ Yes (write bootstrap trees to .ufboot file)

**Maximum iterations:** 1000 ?

**Minimum correlation coefficient:** 0.99 ?

**Single branch tests:** ?

**SH-aLRT branch test:** ☐ No ☒ Yes #replicates: 1000 ?

**Approximate Bayes test:** ☐ Yes

### IQ-TREE Search Parameters

**Perturbation strength:** 0.5 ?

**Stopping rule:** 100 ?

**Email (optional, to retrieve results):**  [SUBMIT JOB](#)

☒ Tree Topology Evaluation and Tests

Please visit the [IQ-TREE homepage](#) for more information or if you want to download the main software!

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.

IQ-TREE web server: fast and accurate phylogenetic trees under maximum likelihood

Server load: 9%

Nguyen LT, Schmidt HA, von Haeseler A, Minh BQ (2015) *Mol. Biol. Evol.*, **32**:268-274  
Minh BQ, Nguyen MAT, and von Haeseler A (2013) *Mol. Biol. Evol.* **30**:1188-1195

Tree Inference **Model Selection** Analysis Results

Input Data

Alignment file:  Browse... Show example >

Use example alignment: ☐ Yes ?

Sequence type: ☒ Auto-detect ☐ DNA ☐ Protein ☐ Codon  
☐ DNA->AA ☐ Binary ☐ Morphology

Partition file:  Browse... Show example >

User tree file:  Browse... ?

Options

Selection criterion: ☐ Akaike (AIC) ☐ Corrected AIC ☒ Bayesian (BIC)

New model selection procedure: ☐ Yes [+R] Maximum no. of categories: 10 ?

Mixture models:  ?

Partition merging: ☐ Yes

Relaxed clustering %:  ?

Email (optional, to retrieve results):  SUBMIT JOB

More Options

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.

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   |
|------------------|---|
| <b>.iqtree</b>   | Full result of the run, this is the main report file  |
| <b>.log</b>      | Run log   |
| <b>.treefile</b> | Maximum likelihood tree in NEWICK format, can be visualized with treeviewer programs  |
| <b>.svg</b>      | Graphical tree representation in SVG format, done with ete view   |
| <b>.pdf</b>      | Graphical tree representation in PDF format, done with ete view   |
| <b>.contree</b>  | Consensus tree with assigned branch supports where branch lengths are optimized on the original alignment; printed if Ultrafast Bootstrap is selected |
| <b>.ckp.gz</b>   | 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](#).

**IQ-TREE web server: fast and accurate phylogenetic trees under maximum likelihood**

Server load: 6%    **Nguyen LT, Schmidt HA, von Haeseler A, Minh BQ (2015) *Mol. Biol. Evol.*, 32:268-274**  
**Minh BQ, Nguyen MAT, and von Haeseler A (2013) *Mol. Biol. Evol.* 30:1188-1195**

Tree Inference   Model Selection   **Analysis Results**

User name or Email: jana.trifinopoulos@gmail.com   QUERY STATUS

| <input type="checkbox"/>            | No. | Submission Time  | Status  |
|-------------------------------------|-----|------------------|---------|
| <input checked="" type="checkbox"/> | 1   | 2016-01-07 15:20 | Waiting |
| <input type="checkbox"/>            | 2   | 2016-01-07 15:05 | Success |
| <input type="checkbox"/>            | 3   | 2015-12-22 10:53 | Success |
| <input type="checkbox"/>            | 4   | 2015-12-21 16:29 | Success |

**Summary**   Run Log   Full Result

Please bookmark the following link to later monitor/retrieve results:  
<http://iqtree.cibiv.univie.ac.at/?user=jana.trifinopoulos@gmail.com&jobid=160107152018>

An email will be sent to jana.trifinopoulos@gmail.com once the job is finished  
Alternatively, you can [download IQ-TREE](#) and run it locally with the command-line:  
`path_to_iqtree -s protein_example.phy -m TEST`

**Note:** The CPU time limit is 24 hours and RAM limit is 1GB. Your job will be stopped if it exceeds these limits.  
In that case, please run yourself a local IQ-TREE as explained above.

**DOWNLOAD SELECTED JOBS**

Figure 3.3: Analysis Results



# 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 environment 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. 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
```

**NOTE:** If you have raw sequences, you need to first apply alignment programs like [MAFFT](#) or [ClustalW](#) to align the sequences, before feeding them into IQ-TREE.

## 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.

**NOTE:** Starting with version 1.5.4, with this simple command IQ-TREE will by default perform ModelFinder (see [choosing the right substitution model](#) below) to find the best-fit substitution model and then infer a phylogenetic tree using the selected model.

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
|      |      |      |      |      |      +--|

```





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](https://doi.org/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:

```
#for IQ-TREE version >= 1.5.4:
iqtree -s example.phy -m MFP

#for IQ-TREE version <= 1.5.3:
iqtree -s example.phy -m TESTNEW
```

`-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.model1`: 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:

```
#for IQ-TREE version >= 1.5.4:
iqtree -s example.phy -m MF

#for IQ-TREE version <= 1.5.3:
iqtree -s example.phy -m TESTNEWONLY
```

### Why ModelFinder?

- ModelFinder is up to 100 times faster than jModelTest/ProtTest.
- jModelTest/ProtTest provides the invariable (+I) and Gamma rate (+G) heterogeneity across sites, but there is no reason to believe that evolution follows a Gamma distribution. ModelFinder additionally considers the [FreeRate heterogeneity model \(+R\)](#), which relaxes the assumption of Gamma distribution, where the site rates and proportions are *free-to-vary* and inferred independently from the data. Moreover, +R allows to automatically determine the number of rate categories, which is impossible with +G. This can be important especially for phylogenomic data, where the default 4 rate categories may “underfit” the data.
- ModelFinder works transparently with tree inference in IQ-TREE, thus combining both steps in just one single run! This eliminates the need for a separate software for DNA (jModelTest) and another for protein sequences (ProtTest).
- Apart from DNA and protein sequences, ModelFinder also works with codon, binary and morphological sequences.
- ModelFinder can also find best partitioning scheme just like PartitionFinder ([Lanfear et al., 2012](#)). See [advanced tutorial for more details](#).

If you still want to resembles jModelTest/ProtTest, then use option `-m TEST` or `-m TESTONLY` instead.

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

```
#for IQ-TREE version >= 1.5.4:
```

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

#for IQ-TREE version <= 1.5.3:
iqtree -s example.phy -m TESTNEWONLY -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:

```
#for IQ-TREE version >= 1.5.4:
iqtree -s example.phy -m MF -mtree

#for IQ-TREE version <= 1.5.3:
iqtree -s example.phy -m TESTNEWONLY -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
```

Note that ModelFinder works for codon alignments. IQ-TREE version  $\geq 1.5.4$  will automatically invokes ModelFinder to find the best-fit codon model. For version  $\leq 1.5.3$ , use option `-m TESTNEW` (ModelFinder and tree inference) or `-m TESTNEWONLY` (ModelFinder only).

## 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 character 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 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:

```
#for IQ-TREE version >= 1.5.4:
iqtree -s morphology.phy -st MORPH

#for IQ-TREE version <= 1.5.3:
iqtree -s morphology.phy -st MORPH -m TESTNEW
```

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:

```
#for IQ-TREE version >= 1.5.4:
iqtree -s SNP_data.phy -m MFP+ASC

#for IQ-TREE version <= 1.5.3:
iqtree -s SNP_data.phy -m TESTNEW+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., in press) 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**  
(2017) UFBoot2: Improving the ultrafast bootstrap approximation. *Mol. Biol. Evol.*, in press. <http://dx.doi.org/10.1093/molbev/msx281>

To run UFBoot, use the option `-bb`:

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

`-bb` 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`: support values in percentage for all splits (bipartitions), computed as the occurrence frequencies in the bootstrap trees. This file is in “star-dot” format.
- `example.phy.splits.nex`: has the same information as `example.phy.splits` but in NEXUS format, which 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.

**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 -bb 1000 -bnni
```

## 4.8 Assessing branch supports with standard non-parametric 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, 1996](#)). See [single branch tests](#) for more details.

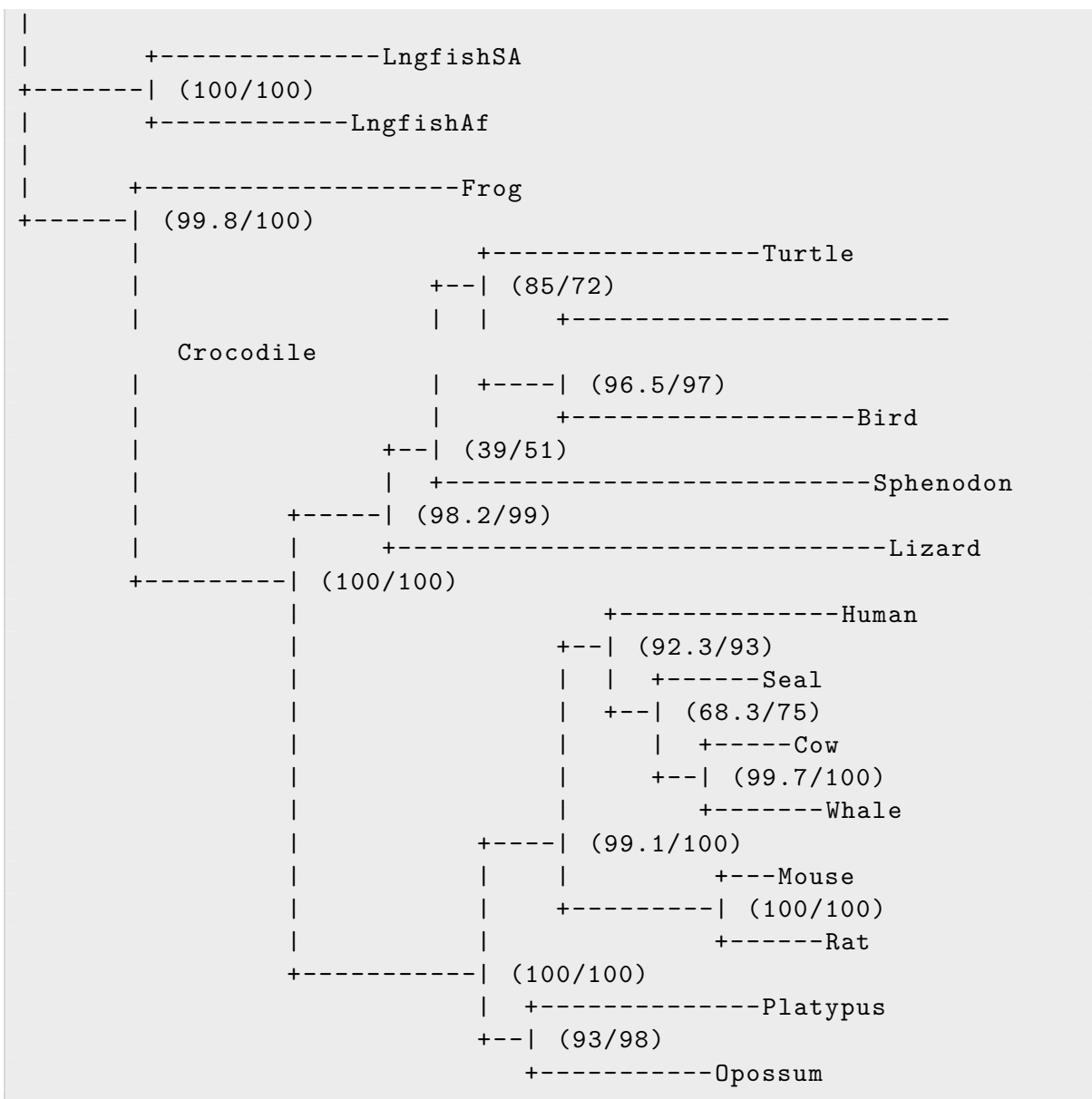
You can also perform both SH-aLRT and the ultrafast bootstrap within one single run:

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

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
```



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

A specialized version of IQ-TREE (`iqtree-omp`) can utilize multiple CPU cores to speed up the analysis. To obtain this version please refer to the [quick starting guide](#). A complement option `-nt` allows specifying the number of CPU cores to use. For example:

```
iqtree-omp -s example.phy -m TIM2+I+G -nt 2
```

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 `-nt AUTO` to determine the best number of cores:

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

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.

## 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).
- `-spp partition_file`: like above but allowing each partition to have its own evolution rate.
- `-sp partition_file`: 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:** `-spp` is recommended for typical analysis. `-q` is unrealistic and `-sp` 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 -q example.partitions -m GTR+I+G
```

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 `-spp` to input the file as following:

```
iqtree -s example.phy -spp example.nex
```

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 substitution 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 -sp example.nex
```

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 subsequently merges two genes until the model fit does not increase any further:

```
# for IQ-TREE version >= 1.5.4:
iqtree -s example.phy -spp example.nex -m MFP+MERGE

# for IQ-TREE version <= 1.5.3:
iqtree -s example.phy -spp example.nex -m TESTNEWMERGE
```

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 -spp example.nex -m TESTMERGE
```

**WARNING:** All these 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. We plan to implement the edge-linked partition finding in version 1.6.

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:

```
# for IQ-TREE version >= 1.5.4:
```

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

# for IQ-TREE version <= 1.5.3:
iqtree -s example.phy -spp example.nex -m TESTNEWMERGEONLY

# to resemble PartitionFinder and save time:
iqtree -s example.phy -spp example.nex -m TESTMERGEONLY
```

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

```
# for IQ-TREE version >= 1.5.4:
iqtree -s example.phy -spp example.nex -m MF+MERGE -rcluster 10

# for IQ-TREE version <= 1.5.3:
iqtree -s example.phy -spp example.nex -m TESTNEWMERGEONLY -
  rcluster 10
```

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 -spp example.nex -bb 1000
```

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 -spp example.nex -bb 1000 -bspec GENE
```

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). This may help to reduce false positives (i.e. wrong branch receiving 100% support):

```
iqtree -s example.phy -spp example.nex -bb 1000 -bspec GENESITE
```

## 5.5 Constrained tree search

Starting with version 1.5.0, 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 -pre example.constr0
```

(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 -pre example.constr1
```

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.

**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 version 1.6.

**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 -pre example.unconstr
```

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 -pre
example.constr1
```

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 -pre
example.constr2
```

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 -pre
example.constr3
```

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 separated 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  $p\text{-inv}=0.15$ .

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 specified. 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 -m GTR+G -wsr -n 0
```

`-wsr` option stands for writing site rates and the number of search iterations is set to 0, such that model parameters are quickly estimated from an initial parsimony tree. 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).

For better site-rate estimates it is recommended to use more than the default 4 rate categories ([Mayrose et al., 2004](#)). Moreover, one should use a more reasonable tree rather than the parsimony tree. For example:

```
iqtree -s example.phy -m GTR+G16 -te ml.treefile -wsr
```

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

Moreover, we recommend to apply the FreeRate model whenever it fits the data better than the Gamma rate model. This is because the Gamma model constrains the rates to come from a Gamma distribution and thus the highest rate may not be *high enough* to accomodate the most fast-evolving sites in the alignment. On the contrary, the FreeRate model allows the rates to freely vary. Moreover, FreeRate allows to automatically determine the best number of rate categories, a feature missing in the Gamma model. The following command:

```
# for IQ-TREE version >= 1.5.4:
iqtree -s example.phy -te ml.treefile -wsr

# for IQ-TREE version <= 1.5.3:
iqtree -s example.phy -m TESTNEW -te ml.treefile -wsr
```

will apply ModelFinder to find the best fit model and then infer the site-specific rates based on a given tree file within a single run! If you omit `-te` option, then IQ-TREE will reconstruct an ML tree and use it to infer site-specific rates.

## 5.11 Where to go from here?

See [Command Reference](#) for a complete list of all options available in IQ-TREE.



# Chapter 6

## Command reference

Comprehensive 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](#).

### 6.1 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   |
|--------------------|---|
| <b>-st</b>         | 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 <b>-st CODON</b> is always necessary when using codon models (otherwise, IQ-TREE applies DNA models) and you also need to specify a <a href="#">genetic code like this</a> if differed from the standard genetic code. <b>-st NT2AA</b> 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 <b>-st NT2AA5</b> for the Invertebrate Mitochondrial Code (see <a href="#">genetic code table</a> ). |
| <b>-t</b>          | Specify a file containing starting tree for tree search. The special option <b>-t BIONJ</b> starts tree search from BIONJ tree and <b>-t RANDOM</b> starts tree search from completely random tree. <i>DEFAULT: 100 parsimony trees + BIONJ tree</i>  |
| <b>-te</b>         | Like <b>-t</b> but fixing user tree. That means, no tree search is performed and IQ-TREE computes the log-likelihood of the fixed user tree.  |
| <b>-o</b>          | Specify an outgroup taxon name to root the tree. The output tree in <b>.treedata</b> will be rooted accordingly. <i>DEFAULT: first taxon in alignment</i>   |
| <b>-pre</b>        | Specify a prefix for all output files. <i>DEFAULT: either alignment file name (-s) or partition file name (-q, -spp or -sp)</i>   |
| <b>-nt</b>         | Specify the number of CPU cores for <b>iqtree-omp</b> version. A special option <b>-nt AUTO</b> will tell IQ-TREE to automatically determine the best number of cores given the current data and computer.  |
| <b>-seed</b>       | Specify a random number seed to reproduce a previous run. This is normally used for debugging purposes. <i>DEFAULT: based on current machine clock</i>  |
| <b>-v</b>          | Turn on verbose mode for printing more messages to screen. This is normally used for debugging purposes. <i>DEFAULT: OFF</i>  |
| <b>-quiet</b>      | Silent mode, suppress printing to the screen. Note that <b>.log</b> file is still written.  |
| <b>-keep-ident</b> | Keep identical sequences in the alignment. By default: IQ-TREE will remove them during the analysis and add them in the end.  |

| Option             | Usage and meaning  |
|--------------------|--|
| <code>-safe</code> | 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. |
| <code>-mem</code>  | Specify maximal RAM usage, for example, <code>-mem 64G</code> to use at most 64 GB of RAM. By default, IQ-TREE will try to not to exceed the computer RAM size.  |

### 6.1.1 Example usages:

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

```
iqtree -s data.phy
```

- Reconstruct a maximum-likelihood tree using 4 CPU cores and at most 8 GB RAM:

```
iqtree-omp -s data.phy -nt 4 -mem 8G
```

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

```
iqtree-omp -s data.phy -nt 4 -mem 8G -pre myrun
```

## 6.2 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   |
|----------------------|---|
| <code>-redo</code>   | Redo the entire analysis no matter if it was stopped or successful.<br>WARNING: This option will overwrite all existing output files. |
| <code>-cptime</code> | 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!

## 6.3 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  |
|-----------------------|--|
| <code>-lmap</code>    | Specify the number of quartets to be randomly drawn. If you specify <code>-lmap ALL</code> , all unique quartets will be drawn, instead. |
| <code>-lmc1ust</code> | Specify a NEXUS file containing taxon clusters (see below for example) for quartet mapping analysis.                                     |
| <code>-wql</code>     | Write quartet log-likelihoods into <code>.lmap.quartet1h</code> file (typically not needed).   |
| <code>-n 0</code>     | Skip subsequent tree search, useful when you only want to assess the phylogenetic information of the alignment.                          |

**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.

### 6.3.1 Example usages:

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

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

## 6.4 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   |
|--------------------------|---|
| <code>-m TESTONLY</code> | Perform standard model selection like jModelTest (for DNA) and ProtTest (for protein). Moreover, IQ-TREE also works for codon, binary and morphological data.                               |
| <code>-m TEST</code>     | Like <code>-m TESTONLY</code> 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. |

| Option  | Usage and meaning   |
|---|---|
| <code>-m TESTNEWONLY</code> or<br><code>-m MF</code>            | Perform an extended model selection that additionally includes FreeRate model compared with <code>-m TESTONLY</code> . <i>Recommended as replacement for <code>-m TESTONLY</code></i> . Note that <code>LG4X</code> is a FreeRate model, but by default is not included because it is also a protein mixture model. To include it, use <code>-madd</code> option (see table below). |
| <code>-m TESTNEW</code> or<br><code>-m MFP</code>               | Like <code>-m MF</code> but immediately followed by tree reconstruction using the best-fit model found.   |
| <code>-m TESTMERGEONLY</code><br><code>-m TESTMERGE</code>      | Select best-fit partitioning scheme like PartitionFinder. Like <code>-m TESTMERGEONLY</code> but immediately followed by tree reconstruction using the best partitioning scheme found.  |
| <code>-m TESTNEWMERGEONLY</code><br>or <code>-m MF+MERGE</code> | Like <code>-m TESTMERGEONLY</code> but additionally includes FreeRate model.  |
| <code>-m TESTNEWMERGE</code><br>or <code>-m MFP+MERGE</code>    | Like <code>-m MF+MERGE</code> but immediately followed by tree reconstruction using the best partitioning scheme found.   |

**WARNING:** 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. We plan to implement the edge-linked partition finding in version 1.6.

**TIP:** During model section run, IQ-TREE will write a file with suffix `.model` 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. Thus, this file acts like a checkpoint to resume the model selection.

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

| Option                 | Usage and meaning   |
|------------------------|---|
| <code>-rcluster</code> | Specify the percentage for the relaxed clustering algorithm ( <a href="#">Lanfear et al., 2014</a> ). This is similar to <code>--rcluster-percent</code> option of PartitionFinder. For example, with <code>-rcluster 10</code> only the top 10% partition schemes are considered to save computations. |

| Option              | Usage and meaning   |
|---------------------|---|
| <code>-mset</code>  | Specify the name of a program ( <code>raxml</code> , <code>phym1</code> or <code>mrBayes</code> ) to restrict to only those models supported by the specified program. Alternatively, one can specify a comma-separated list of base models. For example, <code>-mset WAG,LG,JTT</code> will restrict model selection to WAG, LG, and JTT instead of all 18 AA models to save computations. |
| <code>-msub</code>  | Specify either <code>nuclear</code> , <code>mitochondrial</code> , <code>chloroplast</code> or <code>viral</code> to restrict to those AA models designed for specified source.   |
| <code>-mfreq</code> | Specify a comma-separated list of frequency types for model selection. <i>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</i>   |
| <code>-mrate</code> | Specify a comma-separated list of rate heterogeneity types for model selection. <i>DEFAULT: -mrate E,I,G,I+G for standard procedure, -mrate E,I,G,I+G,R for new selection procedure.</i> (E means Equal/homogeneous rate model).  |
| <code>-cmin</code>  | Specify minimum number of categories for FreeRate model. <i>DEFAULT: 2</i>  |
| <code>-cmax</code>  | Specify maximum number of categories for FreeRate model. It is recommended to increase if alignment is long enough. <i>DEFAULT: 10</i>  |
| <code>-merit</code> | Specify either AIC, AICc or BIC for the optimality criterion to apply for new procedure. <i>DEFAULT: all three criteria are considered</i>  |
| <code>-mtree</code> | Turn on full tree search for each model considered, to obtain more accurate result. Only recommended if enough computational resources are available. <i>DEFAULT: fixed starting tree</i>   |
| <code>-mredo</code> | Ignore <code>.model</code> file computed earlier. <i>DEFAULT: .model file (if exists) is loaded to reuse previous computations</i>  |
| <code>-madd</code>  | Specify a comma-separated list of mixture models to additionally consider for model selection. For example, <code>-madd LG4M,LG4X</code> to additionally include these two <a href="#">protein mixture models</a> .   |
| <code>-mdef</code>  | Specify a <a href="#">NEXUS model file</a> to define new models.  |

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

### 6.4.1 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
```

## 6.5 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 `MODELS` 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 <a href="#">DNA models</a> for more details. |
| Protein  | BLOSUM62, cpREV, Dayhoff, DCMut, FLU, HIVb, HIVw, JTT, JTTDCMut, LG, mtART, mtMAM, mtREV, mtZOA, Poisson, PMB, rtREV, VT, WAG.  |



| DataType   | Model names  |
|------------|--|
| Protein    | Mixture models: C10, ..., C60 (CAT model) ( <a href="#">Lartillot and Philippe, 2004</a> ), EX2, EX3, EHO, UL2, UL3, EX_EHO, LG4M, LG4X, CF4. See <a href="#">Protein models</a> for more details. |
| Codon      | MG, MGK, MG1KTS, MG1KTV, MG2K, GY, GY1KTS, GY1KTV, GY2K, ECMK07/KOSI07, ECMrest, ECMS05/SCHN05 and combined empirical-mechanistic models. See <a href="#">Codon models</a> for more details.       |
| Binary     | JC2, GTR2. See <a href="#">Binary and morphological models</a> for more details.   |
| Morphology | MK, ORDERED. See <a href="#">Binary and morphological models</a> for more details.   |

The following `FreqTypes` are supported:

| FreqType | Meaning   |
|----------|---|
| +F       | Empirical state frequency observed from the data.   |
| +F0      | 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 <a href="#">Codon frequencies</a> .   |
| +F3x4    | See <a href="#">Codon frequencies</a> .   |

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. |

### 6.5.1 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
```

## 6.6 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. <code>+G8</code> .  |
| +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. <code>+R6</code> . <i>DEFAULT: 4 categories</i> |
| +I+R     | invariable site plus FreeRate model.  |

See [Rate heterogeneity across sites](#) for more details.

Further options:

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

## 6.7 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  |
|-------------|--|
| <b>-q</b>   | Specify partition file for edge-equal <a href="#">partition model</a> . That means, all partitions share the same set of branch lengths (like <b>-q</b> option of RAxML).  |
| <b>-spp</b> | Like <b>-q</b> but allowing partitions to have different evolutionary speeds ( <a href="#">edge-proportional partition model</a> ).  |
| <b>-sp</b>  | Specify partition file for <a href="#">edge-unlinked partition model</a> . That means, each partition has its own set of branch lengths (like <b>-M</b> option of RAxML). This is the most parameter-rich partition model to accomodate <i>heterotachy</i> . |

## 6.8 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   |
|--------------|---|
| <b>-ft</b>   | Specify a guide tree (in Newick format) to infer site frequency profiles.   |
| <b>-fs</b>   | Specify a site frequency file, e.g. the <b>.sitefreq</b> file obtained from <b>-ft</b> run. This will save memory used for the first phase of the analysis. |
| <b>-fmax</b> | 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 |   |   |   |   |   |   |   |   |   |   |   |   |   |

## 6.9 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  |
|----------------|--|
| <b>-ninit</b>  | Specify number of initial parsimony trees. <i>DEFAULT: 100</i> . Here <a href="#">the PLL library</a> (Flouri et al., 2015) is used, which implements the randomized stepwise addition and parsimony subtree pruning and regrafting (SPR). |
| <b>-ntop</b>   | Specify number of top initial parsimony trees to optimize with ML nearest neighbor interchange (NNI) search to initialize the candidate set. <i>DEFAULT: 20</i>  |
| <b>-nbest</b>  | Specify number of trees in the candidate set to maintain during ML tree search. <i>DEFAULT: 5</i>  |
| <b>-nstop</b>  | Specify number of unsuccessful iterations to stop. <i>DEFAULT: 100</i>   |
| <b>-n</b>      | Specify number of iterations to stop. This option overrides <b>-nstop</b> criterion.   |
| <b>-sprrad</b> | Specify SPR radius for the initial parsimony tree search. <i>DEFAULT: 6</i>  |
| <b>-pers</b>   | Specify perturbation strength (between 0 and 1) for randomized NNI. <i>DEFAULT: 0.5</i>  |
| <b>-allnni</b> | 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. <i>DEFAULT: OFF</i>   |
| <b>-djc</b>    | Avoid computing ML pairwise distances and BIONJ tree.  |
| <b>-g</b>      | Specify a topological constraint tree file in NEWICK format. The constraint tree can be a multifurcating tree and need not to include all taxa.  |

**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`).

### 6.9.1 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
```

## 6.10 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  |
|--------------------|--|
| <code>-bb</code>   | Specify number of bootstrap replicates ( $\geq 1000$ ).  |
| <code>-wbt</code>  | Turn on writing bootstrap trees to <code>.ufboot</code> file. <i>DEFAULT: OFF</i>              |
| <code>-wbtl</code> | Like <code>-wbt</code> but bootstrap trees written with branch lengths. <i>DEFAULT: OFF</i>    |
| <code>-nm</code>   | Specify maximum number of iterations to stop. <i>DEFAULT: 1000</i>                             |
| <code>-bcor</code> | Specify minimum correlation coefficient for UFBoot convergence criterion. <i>DEFAULT: 0.99</i> |

| Option              | Usage and meaning  |
|---------------------|--|
| <code>-nstep</code> | Specify iteration interval checking for UFBoot convergence. <i>DEFAULT: every 100 iterations</i>   |
| <code>-beps</code>  | Specify a small epsilon to break tie in RELL evaluation for bootstrap trees. <i>DEFAULT: 0.5</i>   |
| <code>-bspec</code> | Specify the resampling strategies for partitioned analysis. By default, IQ-TREE resamples alignment sites within partitions. With <code>-bspec GENE</code> IQ-TREE will resample partitions. With <code>-bspec GENESITE</code> IQ-TREE will resample partitions and then resample sites within resampled partitions ( <a href="#">Gadagkar et al., 2005</a> ).                   |
| <code>-bnni</code>  | 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 <b>severe model violations</b> . 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. |

### 6.10.1 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
```

## 6.11 Nonparametric bootstrap

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

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

## 6.12 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  |
|----------------|--|
| <b>-alrt</b>   | Specify number of replicates ( $\geq 1000$ ) to perform SH-like approximate likelihood ratio test (SH-aLRT) ( <a href="#">Guindon et al., 2010</a> ). If number of replicates is set to 0 ( <b>-alrt 0</b> ), then the parametric aLRT test ( <a href="#">Anisimova and Gascuel 2006</a> ) is performed, instead of SH-aLRT. |
| <b>-abayes</b> | Perform approximate Bayes test ( <a href="#">Anisimova et al., 2011</a> ).   |
| <b>-lbp</b>    | Specify number of replicates ( $\geq 1000$ ) to perform fast local bootstrap probability method ( <a href="#">Adachi and Hasegawa, 1996</a> ).   |

**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.

### 6.12.1 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
```

## 6.13 Tree topology tests

IQ-TREE provides a number of tests for significant topological differences between trees:

| Option            | Usage and meaning   |
|-------------------|---|
| <code>-z</code>   | Specify a file containing a set of trees. IQ-TREE will compute the log-likelihoods of all trees.  |
| <code>-zb</code>  | Specify the number of RELL (Kishino et al., 1990) replicates ( $\geq 1000$ ) to perform several tree topology tests for all trees passed via <code>-z</code> . 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). |
| <code>-zw</code>  | Used together with <code>-zb</code> to additionally perform the weighted-KH and weighted-SH tests.  |
| <code>-au</code>  | Used together with <code>-zb</code> 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 <code>-zb</code> .  |
| <code>-n 0</code> | Only estimate model parameters on an initial parsimony tree and ignore a full tree search to save time.   |
| <code>-te</code>  | Specify a fixed user tree to estimate model parameters. Thus it behaves like <code>-n 0</code> but uses a user-defined tree instead of parsimony tree.  |

**NOTE1:** 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 version 1.6.

**NOTE2:** 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.



### 6.13.1 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
```

## 6.14 Constructing consensus tree

IQ-TREE provides a fast implementation of consensus tree construction for post analysis:

| Option               | Usage and meaning   |
|----------------------|---|
| <code>-t</code>      | Specify a file containing a set of trees.   |
| <code>-con</code>    | Compute consensus tree of the trees passed via <code>-t</code> . Resulting consensus tree is written to <code>.contree</code> file.   |
| <code>-net</code>    | Compute consensus network of the trees passed via <code>-t</code> . Resulting consensus network is written to <code>.nex</code> file.   |
| <code>-minsup</code> | Specify a minimum threshold (between 0 and 1) to keep branches in the consensus tree. <code>-minsup 0.5</code> means to compute majority-rule consensus tree. <i>DEFAULT: 0 to compute extended majority-rule consensus.</i>  |
| <code>-bi</code>     | Specify a burn-in, which is the number of beginning trees passed via <code>-t</code> to discard before consensus construction. This is useful e.g. when summarizing trees from MrBayes analysis.  |
| <code>-sup</code>    | Specify an input “target” tree file. That means, support values are first extracted from the trees passed via <code>-t</code> , and then mapped onto the target tree. Resulting tree with assigned support values is written to <code>.suptree</code> file. This option is useful to map and compare support values from different approaches onto a single tree. |
| <code>-suptag</code> | Specify name of a node in <code>-sup</code> target tree. The corresponding node of <code>.suptree</code> will then be assigned with IDs of trees where this node appears. Special option <code>-suptag ALL</code> will assign such IDs for all nodes of the target tree.  |

| Option        | Usage and meaning  |
|---------------|--|
| <b>-scale</b> | Set the scaling factor of support values for <b>-sup</b> option (default: 100, i.e. percentages) |
| <b>-prec</b>  | Set the precision of support values for <b>-sup</b> option (default: 0)                          |

## 6.15 Computing Robinson-Foulds distance

IQ-TREE provides a fast implementation of Robinson-Foulds (RF) distance computation between trees:

| Option         | Usage and meaning   |
|----------------|---|
| <b>-t</b>      | Specify a file containing a set of trees.   |
| <b>-rf</b>     | Specify a second set of trees. IQ-TREE computes all pairwise RF distances between two tree sets passed via <b>-t</b> and <b>-rf</b> |
| <b>-rf_all</b> | Compute all-to-all RF distances between all trees passed via <b>-t</b>  |
| <b>-rf_adj</b> | Compute RF distances between adjacent trees passed via <b>-t</b>  |

### 6.15.1 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
```

## 6.16 Generating random trees

IQ-TREE provides several random tree generation models:

| Option    | Usage and meaning  |
|-----------|--|
| <b>-r</b> | Specify number of taxa. IQ-TREE will create a random tree under Yule-Harding model with specified number of taxa |

| Option             | Usage and meaning  |
|--------------------|--|
| <code>-ru</code>   | Like <code>-r</code> , but a random tree is created under uniform model.   |
| <code>-rcat</code> | Like <code>-r</code> , but a random caterpillar tree is created.   |
| <code>-rbal</code> | Like <code>-r</code> , but a random balanced tree is created.  |
| <code>-rcsg</code> | Like <code>-r</code> , but a random circular split network is created.   |
| <code>-rlen</code> | Specify three numbers: minimum, mean and maximum branch lengths of the random tree. <i>DEFAULT: <code>-rlen 0.001 0.1 0.999</code></i> |

### 6.16.1 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.

## 6.17 Miscellaneous options

| Option           | Usage and meaning  |
|------------------|--|
| <code>-wt</code> | Turn on writing all locally optimal trees into <code>.treels</code> file. <i>DEFAULT: <code>OFF</code></i> |

| Option         | Usage and meaning   |
|----------------|---|
| <b>-fixbr</b>  | Turn on fixing branch lengths of tree passed via <b>-t</b> or <b>-te</b> . This is useful to evaluate the log-likelihood of an input tree with fixed topology and branch lengths. <i>DEFAULT: OFF</i>   |
| <b>-wsl</b>    | Turn on writing site log-likelihoods to <b>.site1h</b> file in <a href="#">TREE-PUZZLE</a> format. Such file can then be passed on to <a href="#">CONSEL</a> for further tree tests. <i>DEFAULT: OFF</i>  |
| <b>-wslg</b>   | Turn on writing site log-likelihoods per rate category. <i>DEFAULT: OFF</i>   |
| <b>-fconst</b> | 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, <b>-fconst 10,20,15,40</b> 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. |

# Chapter 7

## 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.

### 7.1 DNA models

#### 7.1.1 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 ( <a href="#">Jukes and Cantor, 1969</a> ). | 000000 |
| F81        | 3  | Equal rates but unequal base freq. ( <a href="#">Felsenstein, 1981</a> ).                       | 000000 |
| K80 or K2P | 1  | Unequal transition/transversion rates and equal base freq. ( <a href="#">Kimura, 1980</a> ).    | 010010 |

| Model        | df | Explanation  | Code   |
|--------------|----|--|--------|
| HKY or HKY85 | 4  | Unequal transition/transversion rates and unequal base freq. ( <a href="#">Hasegawa, Kishino and Yano, 1985</a> ). | 010010 |
| TN or TN93   | 5  | Like HKY but unequal purine/pyrimidine rates ( <a href="#">Tamura and Nei, 1993</a> ).                             | 010020 |
| TNe          | 2  | Like TN but equal base freq.   | 010020 |
| K81 or K3P   | 2  | Three substitution types model and equal base freq. ( <a href="#">Kimura, 1981</a> ).                              | 012210 |
| K81u         | 5  | Like K81 but unequal base freq.  | 012210 |
| TPM2         | 2  | AC=AT, AG=CT, CG=GT and equal base freq.   | 121020 |
| TPM2u        | 5  | Like TPM2 but unequal base freq.   | 121020 |
| TPM3         | 2  | AC=CG, AG=CT, AT=GT and equal base freq.   | 120120 |
| TPM3u        | 5  | Like TPM3 but unequal base freq.   | 120120 |
| 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.  | 121030 |
| TIM2e        | 3  | Like TIM2 but equal base freq.   | 121030 |
| TIM3         | 6  | AC=CG, AT=GT and unequal base freq.  | 120130 |
| TIM3e        | 3  | Like TIM3 but equal base freq.   | 120130 |
| TVM          | 7  | Transversion model, AG=CT and unequal base freq.   | 412310 |
| TVMe         | 4  | Like TVM but equal base freq.  | 412310 |
| SYM          | 5  | Symmetric model with unequal rates but equal base freq. ( <a href="#">Zharkikh, 1994</a> ).                        | 123450 |
| GTR          | 8  | General time reversible model with unequal rates and unequal base freq. ( <a href="#">Tavare, 1986</a> ).          | 123450 |

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). 123450 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 120120+G`.

**NOTE:** The last digit in this code must always be 0. It corresponds to G-T rate which is always 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 121030. Thus, `TIM2{4.39,5.30,12.1}` means that A-C=A-T=4.39 (coded 1), A-G=5.30 (coded 2), C-T=12.1 (coded 3) and C-G=G-T=1.0 (coded 0). This is, in turn, equivalent to specifying `GTR{4.39,5.30,4.39,1.0,12.1}`.

### 7.1.2 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. |
| +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).

### 7.1.3 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                             |
| 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 (strand symmetric model) |
| 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)     |



| Model | Rev? | Freq | Note |
|-------|------|------|------|
|-------|------|------|------|

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 prefixes:

| Prefix | Meaning                             |
|--------|-------------------------------------|
| RY     | purine-pyrimidine pairing (default) |
| WS     | weak-strong pairing                 |
| MK     | aMino-Keto pairing                  |

## 7.2 Protein models

### 7.2.1 Amino-acid exchange rate matrices

IQ-TREE supports all common empirical amino-acid exchange rate matrices (alphabetical order):

| Model    | Explanation  |
|----------|--|
| BLOSUM62 | BLOcks SUBstitution Matrix ( <a href="#">Henikoff and Henikoff, 1992</a> ). Note that BLOSUM62 is not recommended as it was designed mainly for sequence alignments. |
| cpREV    | chloroplast matrix ( <a href="#">Adachi et al., 2000</a> ).  |
| Dayhoff  | General matrix ( <a href="#">Dayhoff et al., 1978</a> ).   |
| DCMut    | Revised Dayhoff matrix ( <a href="#">Kosiol and Goldman, 2005</a> ).   |
| FLU      | Influenza virus ( <a href="#">Dang et al., 2010</a> ).   |
| HIVb     | HIV between-patient matrix HIV-Bm ( <a href="#">Nickle et al., 2007</a> ).   |
| HIVw     | HIV within-patient matrix HIV-Wm ( <a href="#">Nickle et al., 2007</a> ).  |
| JTT      | General matrix ( <a href="#">Jones et al., 1992</a> ).   |
| JTTDCMut | Revised JTT matrix ( <a href="#">Kosiol and Goldman, 2005</a> ).   |
| LG       | General matrix ( <a href="#">Le and Gascuel, 2008</a> ).   |
| mtART    | Mitochondrial Arthropoda ( <a href="#">Abascal et al., 2007</a> ).   |
| mtMAM    | Mitochondrial Mammalia ( <a href="#">Yang et al., 1998</a> ).  |

| Model   | Explanation  |
|---------|--|
| mtREV   | Mitochondrial Vertebrate ( <a href="#">Adachi and Hasegawa, 1996</a> ).  |
| mtZOA   | Mitochondrial Metazoa (Animals) ( <a href="#">Rota-Stabelli et al., 2009</a> ).  |
| Poisson | Equal amino-acid exchange rates and frequencies.   |
| PMB     | Probability Matrix from Blocks, revised BLOSUM matrix ( <a href="#">Veerassamy et al., 2004</a> ).   |
| rtREV   | Retrovirus ( <a href="#">Dimmic et al., 2002</a> ).  |
| VT      | General matrix ( <a href="#">Mueller and Vingron, 2000</a> ).  |
| WAG     | General matrix ( <a href="#">Whelan and Goldman, 2001</a> ).   |
| GTR20   | General time reversible models with 190 rate parameters. <i>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).</i> |

## 7.2.2 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 ( <a href="#">Le et al., 2008a</a> ) as variants of the CAT model ( <a href="#">Lartillot and Philippe, 2004</a> ) for ML. Note that these models assume <b>Poisson</b> AA replacement and implicitly include a <b>Gamma rate heterogeneity among sites</b> . |
| EX2        | Two-matrix model for exposed/buried AA sites ( <a href="#">Le et al., 2008b</a> ).  |
| EX3        | Three-matrix model for highly exposed/intermediate/buried AA sites ( <a href="#">Le et al., 2008b</a> ).  |
| EHO        | Three-matrix model for extended/helix/other sites ( <a href="#">Le et al., 2008b</a> ).   |
| UL2, UL3   | Unsupervised-learning variants of <b>EX2</b> and <b>EX3</b> , respectively.   |
| EX_EHO     | Six-matrix model combining <b>EX2</b> and <b>EHO</b> ( <a href="#">Le and Gascuel, 2010</a> ).  |
| LG4M       | Four-matrix model fused with <b>Gamma rate heterogeneity</b> ( <a href="#">Le et al., 2012</a> ).   |
| LG4X       | Four-matrix model fused with <b>FreeRate heterogeneity</b> ( <a href="#">Le et al., 2012</a> ).   |
| CF4        | Five-profile mixture model ( <a href="#">Wang et al., 2008</a> ).   |

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 heterogeneity.

Moreover, one can override the Gamma rate by FreeRate heterogeneity:

- **LG+C20+R4**: Like LG+C20 but replace Gamma by FreeRate heterogeneity.

### 7.2.3 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

```

## 7.2.4 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.    |
| +FO      | ML optimized AA frequencies from the data. |
| +FQ      | Equal AA frequencies.                      |

Users can also specify AA frequencies with, e.g.:

```

+F
{0.079066,0.055941,0.041977,0.053052,0.012937,0.040767,0.071586,0.057337,0.02

```

(Example corresponds to the AA frequencies of the LG matrix).

## 7.3 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 <code>-st CODON</code> )  |
| 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  |
| 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>).

### 7.3.1 Codon substitution rates

IQ-TREE supports several codon models:

| Model               | Explanation  |
|---------------------|--|
| MG                  | Nonsynonymous/synonymous (dn/ds) rate ratio ( <a href="#">Muse and Gaut, 1994</a> ).                         |
| MGK                 | Like MG with additional transition/transversion (ts/tv) rate ratio.  |
| MG1KTS or<br>MGKAP2 | Like MG with a transition rate ( <a href="#">Kosiol et al., 2007</a> ).                                      |
| MG1KTV or<br>MGKAP3 | Like MG with a transversion rate ( <a href="#">Kosiol et al., 2007</a> ).                                    |
| MG2K or<br>MGKAP4   | Like MG with a transition rate and a transversion rate ( <a href="#">Kosiol et al., 2007</a> ).              |
| GY                  | Nonsynonymous/synonymous and transition/transversion rate ratios ( <a href="#">Goldman and Yang, 1994</a> ). |
| GY1KTS or<br>GYKAP2 | Like GY with a transition rate ( <a href="#">Kosiol et al., 2007</a> ).                                      |
| GY1KTV or<br>GYKAP3 | Like GY with a transversion rate ( <a href="#">Kosiol et al., 2007</a> ).                                    |
| GY2K or<br>GYKAP4   | Like GY with a transition rate and a transversion rate ( <a href="#">Kosiol et al., 2007</a> ).              |
| ECMK07 or<br>KOSI07 | Empirical codon model ( <a href="#">Kosiol et al., 2007</a> ).   |
| ECMrest             | Restricted version of ECMK07 that allows only one nucleotide exchange.                                       |
| ECMS05 or<br>SCHN05 | Empirical codon model ( <a href="#">Schneider et al., 2005</a> ).  |

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.

### 7.3.2 Codon frequencies

IQ-TREE supports the following codon frequencies:

| FreqType | Explanation   |
|----------|---|
| +F       | Empirical codon frequencies counted from the data.                                    |
| +FQ      | Equal codon frequencies.  |
| +F1X4    | Unequal nucleotide frequencies but equal nt frequencies over three codon positions.   |
| +F3X4    | Unequal nucleotide frequencies and unequal nt frequencies over three codon positions. |

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.

## 7.4 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.

## 7.5 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.

## 7.6 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 ( <a href="#">Yang, 1994</a> ) with default 4 rate categories. The number of categories can be changed with e.g. <b>+G8</b> .   |
| +I+G     | invariable site plus discrete Gamma model ( <a href="#">Gu et al., 1995</a> ).   |
| +R       | FreeRate model ( <a href="#">Yang, 1995</a> ; <a href="#">Soubrier et al., 2012</a> ) that generalizes the <b>+G</b> model by relaxing the assumption of Gamma-distributed rates. The number of categories can be specified with e.g. <b>+R6</b> (default 4 categories if not specified). The FreeRate model typically fits data better than the <b>+G</b> 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 8

## 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.

### 8.1 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.

### 8.1.1 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 alignment 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 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](#).

### 8.1.2 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.

## 8.2 Mixture models

### 8.2.1 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: it will compute for each site its probability of belonging to each of the mixture classes (also called categories or components). Since the site-to-class assignment is not known, 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 type of mixture model, which have several rate categories with equal probability. 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.

### 8.2.2 Defining mixture models

To start with, the following command:

```
iqtree -s example.phy -m "MIX{JC,HKY}"
```

is a valid analysis. Here, we specify a mixture model (via `MIX` keyword in the model string) with two components (`JC` and `HKY` model) given in curly bracket and comma separator.

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! Otherwise, the Terminal/Console might not recognize the model string properly.

Mixture models can be combined with rate heterogeneity, e.g.:

```
iqtree -s example.phy -m "MIX{JC,HKY}+G4"
```

Here, we specify two models and four Gamma rate categories. Effectively it means that there are 8 mixture components! Each site has a probability belonging to either JC or HKY and to one of the four rate categories.

### 8.2.3 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;

```

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}
```

That 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`.

## 8.2.4 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.

## 8.3 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** (2017) Modeling site heterogeneity with posterior mean site frequency profiles accelerates accurate phylogenomic estimation. *Syst. Biol.*, in press. <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 |
|------------|---------------|-----------------|-----------|
| 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    |

### 8.3.1 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:

```
iqtree-omp -nt 24 -s <alignment> -m LG+C20+F+G -fs <file.sitefreq>
```

See also [the list of relevant command line options](#).

## 8.4 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. Jermini and A. von Haeseler** (2017) GHOST: Recovering historical signal from heterotachously-evolved sequence alignments.

<https://doi.org/10.1101/174789>

### 8.4.1 Download

Currently this model is only available in the beta version 1.6. Please download it from [here](#):

<http://www.iqtree.org/#variant>

### 8.4.2 Quick usages

The GHOST model with  $k$  mixture classes is executed by adding **+H $k$**  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 **+F0** option is required:

```
iqtree -s data.fst -m GTR+F0*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+F0*H4 -wspm
```

# Chapter 9

## Polymorphism-aware models

Use population data to improve inferences.

**Polymorphism-aware phylogenetic Models (PoMo)** improve phylogenetic inference using population data (site frequency data). Thereby they build 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 **POLYMRPHISM 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.
```

## 9.1 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 individuals each (two to 56 chromosomes).

### 9.1.1 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](#).

## 9.2 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.

## 9.3 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).

## 9.4 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
```

## 9.5 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  $4N\mu$ . 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.

## 9.6 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.

## 9.7 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+F0
```

## 9.8 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
```

## 9.9 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](#).

## 9.10 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)
```

## 9.11 [Schrempf et al., 2016](http://dx.doi.org/10.1016/j.jtbi.2016.05.001): <http://dx.doi.org/10.1016/j.jtbi.2016.05.001>

layout: userdoc title: "Compilation Guide" author: Dominik Schrempf, Jana Trifinopoulos, Minh Bui date: 2017-05-10 docid: 20 icon: book doctype: manual tags: - manual description: For advanced users to compile IQ-TREE source code. sections: - name: General requirements url: general-requirements - name: Downloading source code url: downloading-source-code - name: Compiling under Linux url: compiling-under-linux - name: Compiling under Mac OS X url: compiling-under-mac-os-x - name: Compiling under Windows url: compiling-under-windows - name: Compiling 32-bit version url: compiling-32-bit-version - name: Compiling MPI version url: compiling-mpi-version -

name: Compiling Xeon Phi Knights Landing version url: [compiling-xeon-phi-knights-landing-version](#) —

# Chapter 10

## Compilation guide

For advanced users to compile IQ-TREE source code.

### 10.1 General requirements

- A C++ compiler such as GCC (version  $\geq 4.8$ ), Clang, MS Visual Studio and Intel C++ compiler.
- [CMake](#) version  $\geq 2.8.10$ .
- [Eigen3 library](#) (for IQ-TREE version  $\geq 1.6$ ). 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).
- (*Optional*) If you want to compile the multicore version, make sure that the OpenMP library was installed. This should typically be the case with `gcc` under Linux.
- (*Optional*) Install [git](#) if you want to clone source code from [IQ-TREE GitHub repository](#).

### 10.2 Downloading source code

Choose the source code (`zip` or `tar.gz`) of the IQ-TREE release you want to use from:

<https://github.com/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/Cibiv/IQ-TREE.git
```

Please find below separate compilation guide for [Linux](#), [Mac OS X](#), and [Windows](#) as well as for [32-bit version](#) or for [MPI version](#).

## 10.3 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 ..
```

To build the multicore version please add `-DIQTREE_FLAGS=omp` to the `cmake` command:

```
cmake -DIQTREE_FLAGS=omp ..
```

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
```

You can speed up the compilation by specifying the number of CPU cores to use with `make` by e.g.:

```
make -j4
```

to use 4 cores instead of the default 1 core.

This creates an executable `iqtree` or `iqtree-omp`. It can be copied to your system search path so that IQ-TREE can be called from the Terminal simply with the command line `iqtree`.

**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.

## 10.4 Compiling under Mac OS X

**TIP:** A ready made IQ-TREE package is provided by \* [Homebrew](#).

- Make sure that Clang compiler is installed, which is typically the case if you installed Xcode and the associated command line tools.
- 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).

## 10.5 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.

## 10.6 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 `DIQTREE_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 ..
```

## 10.7 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-omp-mpi`. 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):

```
mpirun -np 4 iqtree-omp-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 **iqtree-omp**. 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-omp with mpirun!**

## 10.8 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
```

## 10.9 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 necessary 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 ..
```

# Chapter 11

## Frequently asked questions

For common questions and answers.

### 11.1 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 request, 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.

### 11.2 How do I report 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-omp: ....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!

### 11.3 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  $\geq 95\%$ . 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  $\geq 80\%$  and UFboot  $\geq 95\%$ .

### 11.4 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 ambiguous 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,<br>N, X | A, G, C or T (unknown; all 4 nucleotides are equally likely) |

For protein 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) |

## 11.5 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 `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).

## 11.6 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.

## 11.7 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 composi-



tion (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:

$$\text{chi2} = \sum_{i=1}^k (O_i - E_i)^2 / E_i$$

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.

## 11.8 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.

## 11.9 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:

```
iqtree-omp -nt 4 -s input_alignment -bo 20 ... -pre boot1
iqtree-omp -nt 4 -s input_alignment -bo 20 ... -pre boot2
iqtree-omp -nt 4 -s input_alignment -bo 20 ... -pre boot3
iqtree-omp -nt 4 -s input_alignment -bo 20 ... -pre boot4
iqtree-omp -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. You can also perform the analysis on the original alignment:

### 11.10. WHY DOES IQ-TREE COMPLAIN ABOUT THE USE OF +ASC MODEL?123

```
iqtree-omp -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.

## 11.10 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: Invaidd 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.