IQ-TREE version 1.0 (July 2014)

Fast phylogenetic inference and ultrafast bootstrap analysis by maximum likelihood.

User Manual and Tutorial

Please read carefully before using IQ-TREE the first time!

Project managers:

Bui Quang Minh - `minh.bui(at)mfpl.ac.at`

Arndt von Haeseler - `arndt.von.haeseler(at)mfpl.ac.at`

Core developers:

Lam-Tung Nguyen - `tung.nguyen(at)mfpl.ac.at`

Olga Chernomor - `olga.chernomor(at)mfpl.ac.at`

Diep Thi Hoang - `diep.thi.hoang(at)gmail.com`

Support:

Heiko A. Schmidt - `heiko.schmidt(at)mfpl.ac.at`

Contact address:

Center for Integrative Bioinformatics Vienna (CIBIV)
Max F. Perutz Laboratories, University of Vienna, Medical University of Vienna
Dr. Bohr-Gasse 9, A-1030 Vienna, Austria

Contents

1 Introduction

IQ-TREE is an efficient program for reconstructing large maximum likelihood trees and assessing branch supports with the ultrafast bootstrap approximation. IQ-TREE extends the IQPNNI algorithm with many enhancements. IQ-TREE is open-source and available free of charge from

`http://www.cibiv.at/software/iqtree/`

IQ-TREE has been tested on Unix, Mac OS X, and Windows. The code of IQ-TREE has been written in standard C/C++, which is possibly compilable on other platforms. Please read the *Installation* section 2 for more details. We suggest that this documentation should be read before using IQ-TREE the first time!

For impatient users we established a very user-friendly web server:

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

Its intuitive web interface allows users to perform online tree reconstruction within a few clicks. Note that this online service only allows max. 12 CPU hours and 1 GB memory per job. In case your job exceeds these limits, you can copy and paste the command-line displayed to run the analysis at your local machine.

To cite IQ-TREE please use the following paper:

Bui Quang Minh, Minh Anh Thi Nguyen, and Arndt von Haeseler (2013) Ultrafast approximation for phylogenetic bootstrap. *Mol. Biol. Evol.*, 30:1188-1195.

Further readings on the methods developed:

- **Heiko A. Schmidt and Arndt von Haeseler** (2009) Phylogenetic Inference Using Maximum Likelihood Methods. In P. Lemey, M. Salemi, A.M. Vandamme (eds.) *The Phylogenetic Handbook: a Practical Approach to Phylogenetic Analysis and Hypothesis Testing.*, 2nd Edition, 181-209, Cambridge University Press, Cambridge.
- **Bui Quang Minh, Le Sy Vinh, Arndt von Haeseler and Heiko A. Schmidt** (2005) pIQPNNI: Parallel reconstruction of large maximum likelihood phylogenies. *Bioinformatics*, 21(19):3794-6.
- **Le Sy Vinh and Arndt von Haeseler** (2004) IQPNNI: Moving fast through tree space and stopping in time. *Mol. Biol. Evol.*, 21(8):1565-1571.

If you encounter bugs please send the `.log` file of the run and possibly the alignment to: `tung.nguyen(AT)univie.ac.at` and `minh.bui(AT)univie.ac.at`.

1.1 What's new in version 1.0?

Version 1.0 is the major release of the IQ-TREE software. We are happy to announce the following new features:

- Integration of the phylogenetic likelihood library (PLL; Flouri *et al.*, 2014) for fast likelihood computation. This is enabled via `-pll` option and gives a speedup of 2X to 8X.
- A novel fast and effective stochastic algorithm for estimating maximum likelihood phylogenies. It outperforms RAxML and PhyML in terms of log-likelihoods while requiring similar amount of computing time. A manuscript describing the new method was submitted. See Section 3.4 for more details.
- Codon models: GY (Goldman & Yang 1994), MG (Muse & Gaut 1994), and ECM (Kosiol et al. 2007)
- Morphological models: MK and ORDERED (Lewis 2001)
- Ascertainment bias correction model (+ASC) for e.g., morphological or SNP data (Lewis 2001)
- Nearest neighbor interchange with five (instead of one) branch optimization (`-nni5`) is now the default option because of its higher accuracy
- SH-aLRT branch test also works now for partition models.

1.2 Key features

IQ-TREE provides a lot of options for phylogenetic reconstruction. The main features include:

- Reconstruction of the maximum likelihood tree from sequence alignments (Vinh and von Haeseler, 2004; Minh *et al.*, 2005).
- Ultrafast bootstrap approximation for assessing branch supports (Minh *et al.*, 2013).
- Various substitution models for binary, nucleotide, amino-acid with/without rate heterogeneity.
- Partition models for phylogenomic data
- Automatic selection of best-fit models similarly to ModelTest (Posada and Crandall, 1998).
- Standard non-parametric bootstrap (Felsenstein, 1985).

- Single branch tests (LBP, SH-aLRT; Adachi and Hasegawa, 1996; Guindon *et al.*, 2010).
- Test of model homogeneity assumption along the tree (Weiss and von Haeseler, 2003).
- Site-specific rate model (Meyer and von Haeseler, 2003).
- Fast consensus tree reconstruction, Robinson-Foulds distance computation.

2 Installation

See below for information how to install/build the different versions of the IQ-TREE software. Executable versions of the sequential, that is, non-parallel program are intended for a number of operating systems.

2.1 Binary release

1. Download the executable version of IQ-TREE for your operating system if it is available (`iqtree-XXX-OS.tar.gz` or `iqtree-XXX-OS.zip`, where `XXX` is the current version number and OS the operating system) from
`http://www.cibiv.at/software/iqtree`
2. Extract the files (e.g., with `tar xvzf iqtree-XXX-OS.tar.gz` under Unix). This should create a directory `iqtree-XXX-OS`.
3. You will find the executable in `iqtree-XXX-OS/`. This executable you should rename to `iqtree` (or `iqtree.exe` on Windows systems) and copy it to your system search path such that it is found by your system.

Note on multi–ompcore version: The executable is named `iqtree-omp` (or `iqtree-omp.exe` on Windows). Please also copy other files needed for OpenMP (e.g., `*.dll` on Windows) to the same folder that you copied `iqtree-omp` to. Finally, for Mac OS X you have to install MacPorts and the associated gcc47 to run `iqtree-omp` properly (see a how-to in section 2.3).

If you encounter problems, please ask your local administrator for help.

2.2 Building source package

To build IQ-TREE from the sources you need a C++ compiler (e.g., gcc) and the CMake tool installed (This is usually the case on UNIX/Linux systems. For Windows you might want to obtain CygWin/MinWG/MS Visual C++ or XCode for MacOSX). Then you can follow the procedure below:

1. Download the current version of the software (`iqtree-XXX-Source.tar.gz` or `iqtree-XXX-Source.zip`, where `XXX` is the current version number) from `http://www.cibiv.at/software/iqtree`
2. Extract the files (e.g., with `tar xvzf iqtree-XXX-Source.tar.gz` under Unix). This should create a directory `iqtree-XXX-Source`.
3. Change into this directory.
4. Create a sub-directory `build` and go into this sub-directory by entering:

   ```
   mkdir build
   cd build
   ```

5. Configure the source codes using CMake:

   ```
   cmake ..
   ```

6. Compile and build the source codes:

   ```
   make
   ```

   This creates an executable `iqtree` (or `iqtree.exe` on Windows systems). This executable can copied to your system search path such that it is found by your system.

If you encounter problems, please ask your local administrator for help.

2.3 Building multi-core parallel version (Update!)

To build the multi-core version you need a compiler that supports the OpenMP standard (e.g., gcc). For Linux and Windows the gcc and MinGW compilers work just fine. However, in our test on Mac OS X, IQ-TREE was successfully compiled with the default the XCode gcc but the example run crashed for unknown reason. Therefore, we employed MacPorts (with gcc47 or later) and successfully ran IQ-TREE compiled with MacPorts gcc. To this end, please first install MacPorts, gcc in MacPorts (`sudo port install gcc47`) and configure gcc to point to the MacPorts' gcc version (`sudo port select --set gcc mp-gcc47`).

The compilation then follows the same route with slightly changed command line for cmake:

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

All other commands remain the same. It is recommended to copy the executable file `iqtree-omp` (or `iqtree-omp.exe` on Windows) to the system search path such that one can simply run `iqtree-omp` from the command-line.

3 Tutorial

This section gives users a quick starting guide. You can either download the binary for your platform from the IQ-TREE website or the source code. In the later case, you will need to compile the source code (see *Installation* section 2). For the next steps, the `iqtree` executable should be then copied into the `bin` folder such that IQ-TREE can be invoked by simply entering `'iqtree'` at the command-line. You can run `'iqtree -h'` to see a list of options available in IQ-TREE.

3.1 First running example(Update!)

From the download there is an example alignment called `example.phy` in PHYLIP format (IQ-TREE also supports FASTA and NEXUS files). You can now start to reconstruct a maximum-likelihood tree from this alignment by typing (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:

- `example.phy.iqtree`: the main report file that is self readable for users. You should look at this file to see the results.
- `example.phy.treefile`: the ML tree in NEWICK format, which can be visualized by tree viewer tools such as FigTree, iTOL. Note that this newick tree is also embedded in `example.phy.iqtree`.
- `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 that all output files have the default prefix as the alignment file name. You can always change the prefix using the `'-pre'` option, e.g.:

```
iqtree -s example.phy -pre myprefix
```

Then IQ-TREE will write output files `myprefix.iqtree, myprefix.treefile`, etc. This is helpful when you do several runs for the same input.

******** NEW IN VERSION 1.0 ********

Since version 1.0 IQ-TREE by default offers a more accurate tree search and bootstrap by optimizing five branches around the nearest neighbor interchanges (NNIs). This comes with a trade-off of approximately 2X longer running time than 0.9.X version. To switch back to old behaviour of optimizing one branch around NNIs, simply use the `-nni1` option:

```
iqtree -s example.phy -nni1
```

3.2 Choosing the substitution model

IQ-TREE supports numerous substitution models for binary, DNA, and protein data and Gamma rate heterogeneity model. If you do not specify, IQ-TREE will use the default HKY, WAG, and JC models for DNA, protein, and binary alignments, respectively. For most data sets these models are too simplified. If you have no idea about which model is appropriate for your data, let IQ-TREE automatically determine the best-fit model for your alignment with:

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

`'-m'` is the option to specify the model name to use during the analysis. `'TEST'` is a key word telling IQ-TREE to first select the best-fit model. The remaining analysis will be done using the selected model. More specifically, IQ-TREE computes the log-likelihoods of the initial BIONJ tree for many different models and the Akaike information criterion (AIC), corrected Akaike information criterion (AICc), and the Bayesian information criterion (BIC). Then IQ-TREE chooses the model that minimizes the BIC score (you can also change to AIC or AICc by adding the option ”-AIC” or ”-AICc”, respectively). Moreover, IQ-TREE will write an additional file:

- `example.phy.model`: log-likelihoods for all models tested.

If you now look at `example.phy.iqtree` you will see that IQ-TREE selected the model `'TIM'` with `'Invar+Gamma'` rate heterogeneity. So `'TIM+I+G'` is the best-fit model for this example data. From now on you can run with e.g.:

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

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

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

Here, IQ-TREE will stop after finishing the model selection. The name of the best-fit model will be printed on the screen. Finally, note that IQ-TREE will check if the file `'*.model'` exists and is correct. If so, it will automatically reuse the log-likelihoods computed to speed up the model selection procedure.

********NEW********

Since version 0.9.6 IQ-TREE offers the partition model selection for multi-gene analysis. See Section 3.8.1 for more details.

3.3 Support for phylogenetic likelihood library (New in version 1.0!)

In the major release 1.0, we added the support for PLL (Flouri *et al.*, 2014) which helps to speed up IQ-TREE by a factor of 2X to 8X. To test the new feature simply use option '-pll', for example:

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

Here, we also specifies model GTR+G as it is currently supported by PLL. Note that this option does not entirely work with other options yet (such as '-m TEST'). In such cases, an error message will be displayed.

3.4 Novel tree search algorithm (New in version 1.0!)

IQ-TREE 1.0 implemented a new tree search algorithm, which explore the tree space much more efficiently than version 0.9.X. Here, IQ-TREE combines parsimony analysis to provide better starting trees, new stochastic algorithm to escape local optima, and new stopping rule. The new search strategies come with a few parameters where the default values were tested to work well for many different data sets. Moreover, you can change the default parameters with options:

```
  -numpars <number>    Number of initial parsimony trees (default: 100)
  -toppars <number>    Number of top initial parsimony trees (dfault: 20)
  -numcand <number>    Number of candidate trees during search (defaut: 5)
  -pers <perturbation> Perturbation strength for stochastic NNI (default: 0.5)
  -numstop <number>    Number of unsuccessful iterations to stop (default: 100)
```

Finally, you can still switch back to the old algorithm of 0.9.X by options:

```
  -iqp                 Use IQP tree perturbation (default: sNNI)
  -iqpnni              Switch entirely to old IQPNNI algorithm
```

3.5 Codon models (New in version 1.0!)

IQ-TREE 1.0 supports basic codon models (GY, MG, and ECM). 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, an error message will occur. IQ-TREE will group sites 1,2,3 into codon site 1; site 4,5,6 to codon site 2; etc. Moreover, any codon, which has at least one gap/unknown/ambiguous nucleotide, will be treated unknown codon character.

If you are not sure which model to use, simply add `'-m TEST'`, which also works for codon alignments:

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

By default IQ-TREE uses the standard genetic code. You can change to other genetic code (see `http://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi`) with following options:

Option	Genetic code
`-st CODON1`	The Standard Code (same as `-st CODON`)
`-st CODON2`	The Vertebrate Mitochondrial Code
`-st CODON3`	The Yeast Mitochondrial Code
`-st CODON4`	The Mold, Protozoan, and Coelenterate Mitochondrial Code and the Mycoplasma/Spiroplasma Code
`-st CODON5`	The Invertebrate Mitochondrial Code
`-st CODON6`	The Ciliate, Dasycladacean and Hexamita Nuclear Code
`-st CODON9`	The Echinoderm and Flatworm Mitochondrial Code
`-st CODON10`	The Euplotid Nuclear Code
`-st CODON11`	The Bacterial, Archaeal and Plant Plastid Code
`-st CODON12`	The Alternative Yeast Nuclear Code
`-st CODON13`	The Ascidian Mitochondrial Code
`-st CODON14`	The Alternative Flatworm Mitochondrial Code
`-st CODON16`	Chlorophycean Mitochondrial Code
`-st CODON21`	Trematode Mitochondrial Code
`-st CODON22`	Scenedesmus obliquus Mitochondrial Code
`-st CODON23`	Thraustochytrium Mitochondrial Code
`-st CODON24`	Pterobranchia Mitochondrial Code
`-st CODON25`	Candidate Division SR1 and Gracilibacteria Code

3.6 Morphological and SNP data (New in version 1.0!)

IQ-TREE 1.0 supports discrete morphological alignment by `'-st MORPH'` option:

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

IQ-TREE implements to two morphological ML models (MK and ORDERED; see Lewis 2001), where MK is the default model. MK is a Juke-Cantor-like model. ORDERED model considers only transitions between states $i \rightarrow i-1$, $i \rightarrow i$, and $i \rightarrow i+1$. Morphological data typically do not have constant (uninformative) sites. In such case, you should apply ascertainment bias correction model by e.g.:

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

You can again select best-fit model with `'-m TEST'` (which also consider +G):

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

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

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

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

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

3.7 Assessing branch supports with ultrafast bootstrap approximation

The ultrafast bootstrap approximation is the most value-added feature available in IQ-TREE. Simply run:

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

`'-bb'` specifies the number of bootstrap replicates where 1000 is the minimal number recommended. When you now look at the section `'MAXIMUM LIKELIHOOD TREE'` in `example.phy.iqtree`, you will see that every internal node of the tree figure will be associated a support value in percentage. Branch supports are assigned onto the ML tree and printed in `example.phy.treefile` that can be viewed again in FigTree. 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 occurence 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 SplitsTree program.

3.7.1 Assessing branch supports with standard nonparametric bootstrap

The standard nonparametric bootstrap can be invoked by:

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

`'-b'` specifies the number of bootstrap replicates where 100 is the minimal number recommended. IQ-TREE will additionally writes the following files:

- `example.phy.boottrees`: the set of bootstrap trees reconstructed.
- `example.phy.contree`: the bootstrap consensus tree with assigned branch supports where branch lengths are optimized on the original alignment.

3.7.2 Assessing branch supports with single branch tests

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

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

`'-alrt'` specifies the number of bootstrap replicates for SH-aLRT where 1000 is the minimal number recommended. IQ-TREE will perform SH-aLRT at the end of the tree reconstruction process and assign support values onto the ML tree. The support values will be reflected in the tree file `example.phy.treefile`.

IQ-TREE also provides a fast implementation of the local bootstrap probabilities method (Adachi and Hasegawa, 1996), which we call Fast-LBP. Fast-LBP computes the branch support by comparing the tree log-likelihood with the log-likelihoods of the two alternative nearest-neighbor-interchange (NNI) trees around the branch of interest. However, Fast-LBP is different from LBP where we compute the log-likelihoods of the two alternative NNI trees by only reoptimizing five branches around the branch of interest (Similar idea is used in the SH-aLRT test). To perform Fast-LBP, simply run:

```
iqtree -s example.phy -m TIM+I+G -lbp 1000
```

You can also perform both tests:

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

The branches of the resulting ML tree will be assigned with both SH-aLRT and Fast-LBP support values. Finally, you can also combine the ultrafast bootstrap approximation with single branch tests within one single run:

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

3.8 Partitioned analysis for multi-gene alignments

In the partition model, you can specify a substitution model for each gene/character set individually. IQ-TREE will then estimate the model parameters and branch lengths separately for every partition. To this end, you have to first prepare a NEXUS file including 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;
```

Now if you save this into a file `example.nex` and run:

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

This means that IQ-TREE will partition the alignment `example.phy` into 2 subsets named `part1` and `part2` containing sites (columns) 1-100 and 101-384, respectively. Moreover, IQ-TREE applies the subtitution models `HKY+G` and `GTR+I+G` to `part1` and `part2`, respectively. After the run has finished, the `example.nex.iqtree` file will contain substitution model parameters, trees with branch lengths for all subsets in the partition.

Moreover, the `CharSet` command allows to specify non-consecutive sites using comma-separated list of ranges 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.

Moreover, IQ-TREE allows a more advanced feature compared to other programs: IQ-TREE allows different subsets coming from different alignments. For example:

```
#nexus
begin sets;
        charset part1 = part1.phy: 1-100\3 201-300\3;
        charset part2 = part2.phy: 101-300;
        charpartition mine = HKY:part1, GTR+G:part2;
end;
```

Here, `part1` and `part2` are read from alignment files `part1.phy` and `part2.phy`, respectively (a ':' is needed to separate the alignment file name and site specification). Because the alignment file names were embedded in this NEXUS file, you can simply run:

```
  iqtree -sp example.nex
```

Note that `part1.phy` and `part2.phy` need not contain the same set of sequence names. That means, if some sequence occurs in `part1.phy` but not in `part2.phy`, IQ-TREE will treat corresponding part of sequence in `part2.phy` as missing data. For your convenience IQ-TREE writes the concatenated alignment into the file `example.nex.conaln`.

********NEW EXPERIMENTAL FEATURE********

Since version 0.9.6 IQ-TREE supports partition models with joint and proportional branch lengths between genes. This is to reduce the number of parameters in case of model overfitting for the full partition model. For example:

```
  iqtree -spp example.nex
```

applies a proportional partition model. That means, we have only one set of branch lengths for species tree but allow each gene to evolve under a specific rate (scaling factor) normalized to the average of 1.

A partition model with joint branch lengths is specified by:

```
  iqtree -spj example.nex
```

(i.e., all gene-specific rates are equal to 1).

3.8.1 Choosing the right partitioning scheme

Since version 0.9.6 IQ-TREE implements a greedy strategy (Lanfear *et al.*, 2012) that starts with the full partition model and sequentially merges two genes until the model fit does not increase any further:

```
  iqtree -sp example.nex -m TESTLINK
```

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

```
iqtree -sp example.nex -m TESTONLYLINK
```

3.8.2 Bootstrapping with partition model

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

```
iqtree -sp example.nex -bb 1000
```

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

********NEW********

Since version 0.9.6 IQ-TREE supports the gene-resampling strategy:

```
iqtree -sp example.nex -bb 1000 -bspec GENE
```

is to resample genes instead of sites. Moreover, IQ-TREE allows an even more complicated strategy: resampling genes and sites within resampled genes:

```
iqtree -sp example.nex -bb 1000 -bspec GENESITE
```

3.9 Utilizing multi-core CPUs

A specialized version of IQ-TREE allows users to perform the analysis that utilizes multiple cores during the run (made possible by the OpenMP library). You can download the binary from the software website or compile the source code yourself (see *Installation* section 2). For the following please copy the binary `iqtree-omp` and other files in the package bin folder into the system `bin` folder such that it can be invoked from the command-line by simply running the command `iqtree-omp`.

If you now run with e.g.:

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

Then IQ-TREE will use all the available cores of your CPU. This might not be a good practice because our parallelization technique only works well on long alignments. If you have a very short alignment, it is not recommended to use this IQ-TREE version. Because the speedup gain depends on the alignment length, a good practice is to run this version with increasing number of cores by e.g.:

```
iqtree-omp -s example.phy -omp 2
```

Here, `-omp` is the option to specify the number of cores that IQ-TREE will use. If you see that the wall-clock time reduction is substantial compared with the sequential IQ-TREE version, then you can try:

```
iqtree-omp -s example.phy -omp 3
```

and so on, until no substantial reduction of running time is observed. The remaining analysis can then be carried out with that number of cores.

For example, on my computer (Linux, Intel Core i5-2500K, 3.3 GHz, quad cores) I observed the following wall-clock running time for this example alignment:

No. cores	Wall-clock time
1	21.465 sec.
2	13.627 sec.
3	11.119 sec.
4	10.807 sec.

Therefore, I would only use 2 cores for this specific alignment (`"-omp 2"` option).

4 Advanced tutorial

This section gives an advanced tutorial for more experienced users. It includes several advanced features like tree topology test, user-defined substitution models.

4.1 Tree topology tests

IQ-TREE can compute log-likelihoods of user-defined trees passed via `-z` option:

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

assuming that `example.treels` contains the trees in NEWICK format. At the end of the usual run, IQ-TREE will additionally evaluate all trees in there using the estimated model parameters. When you look into `example.phy.iqtree` there will be a section `USER TREES` that lists the tree IDs and the corresponding log-likelihoods. Moreover, IQ-TREE will additionally write a file:

- `example.phy.treels.trees`: the trees with optimized branch lengths.

If you only want to evaluate the trees without reconstructing the ML tree, you can run:

```
iqtree -s example.phy -z example.treels -n 1
```

Here, IQ-TREE will only reconstruct the BIONJ+NNI tree and use that tree to estimate the model parameters, which are normally accurate enough for our purpose.

IQ-TREE also supports several tree topology tests using the RELL approximation (Kishino *et al.*, 1990) including: bootstrap proportion (BP), Kishino-Hasegawa test (KH; Kishino and Hasegawa, 1989), Shimodaira-Hasegawa test (SH; Shimodaira and Hasegawa, 1999), expected likelihood weights (ELW; Strimmer and Rambaut, 2002), weighted-KH (WKH), and weighted-SH (WSH) tests. The trees are passed via `-z` option, thus you can run:

```
iqtree -s example.phy -z example.treels -n 1 -zb 1000
```

Here, `-zb` specifies the number of RELL replicates, where 1000 is the minimum number recommended. The `USER TREES` section of `example.phy.iqtree` will list the results of BP, KH, SH, and ELW methods. If you want to also perform the WKH and WSH, simply add `-zw` option:

```
iqtree -s example.phy -z example.treels -n 1 -zb 1000 -zw
```

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

4.2 User-defined substitution models

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

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

Moreover, with -m option one can input a file name 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), e.g.:

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

where `mymodel` is a file containing the 10 entries described above. 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 an 8-category Gamma-distributed site rates with the shape parameter (alpha) of 0.5 and a proportion of invariable sites p-inv=0.15.

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

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

For amino-acid alignments, if one wants to use the frequencies of the empirical protein model, then use `+Fu`, for example:

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

Finally, note that all model specifications above can be used in the partition model NEXUS file.

4.3 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 -t 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 -t 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, an 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) by:

```
  iqtree -sup input_tree set_of_trees
```

4.4 Computing Robinson-Foulds distance between trees

IQ-TREE implements a very fast Robinson-Foulds (RF) distance computation using hash table, which is a lot faster than PHYLIP package. For example, you can run:

```
  iqtree -rf tree_set1 tree_set2
```

to compute the pairwise RF distances between 2 sets of trees. If you want to compute the all-to-all RF distances of a set of trees, use:

```
  iqtree -rf_all tree_set
```

4.5 Generating random trees

IQ-TREE provides several random tree generation models. For example,

```
    iqtree -r 100 100.tree
```

is to generate a 100-taxon random tree into the file `100.tree` under the Yule Harding model, where the branch lengths follow an exponential distribution with mean of 0.1. If you want to change the branch length distribution, run e.g:

```
    iqtree -r 100 -rlen 0.05 0.2 0.3 100.tree
```

to set the minimum, mean, and maximum branch lengths as 0.05, 0.2, and 0.3, respectively. If you want to generate trees under uniform model instead, use '-ru' option:

```
    iqtree -ru 100 100.tree
```

5 Frequently asked questions (FAQ)

5.1 How does IQ-TREE treat gap/missing 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 (RAxML, PhyML, etc.). Technically in the Felsenstein's pruning algorithm we fill a partial likelihood vector of all 1's for all character states. This is the same as follows. For a site (column) of an alignment containing AC-AG-A (i.e. A for sequence 1, C for sequence 2, - for sequence 3,...), 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:

$$\ell(T|AC-AG-A) = \ell(T_{sub}|ACAGA)$$

6 Version History

Version 0.9.6: October 2013

- Ultrafast model selection and partitioning for phylogenomic alignments.
- Introduction of nearest neighbor interchange (NNI) with five branch optimization to evaluate candidate NNIs. This will bring higher accuracy for tree reconstruction and bootstrap with a tradeoff of c.a. 2X longer running time.
- Introduction of joint and proportional partition models to reduce the number of parameters in case of model overfitting (experimental).
- Introduction of gene-resampling and gene-and-site resampling for the bootstrap on multi-gene alignments.

Version 0.9.5: May 2013

- Introduction of bootstrap epsilon to select equally good bootstrap trees at random to deal with polytomies

Version 0.9.4: Easter 2013

- Tree topology tests

Version 0.9.3: March 2013

- New implementation of model selection that works on all data types.
- A tutorial about using partition models.
- Parallel OpenMP support to utilize multi-core CPUs.

Version 0.9.0: September 2012 - First beta release.

Credits and Acknowledgement

Some parts of the code were taken from the following packages/libraries: Phylogenetic likelihood library (Flouri *et al.*, 2014), TREE-PUZZLE (Schmidt *et al.*, 2002), BIONJ (Gascuel, 1997), Nexus Class Libary (Lewis, 2003), Eigen library (Guennebaud *et al.*, 2010), SPRNG library (Mascagni and Srinivasan, 2000), Zlib library (`http://www.zlib.net`).

Financial supports from the Austrian Science Fund (FWF), the Vienna Science and Technology Fund (WWTF), and the University of Vienna are greatly appreciated.

References

Adachi, J. and Hasegawa, M. (1996) *MOLPHY Version 2.3 – Programs for Molecular Phylogenetics Based on Maximum Likelihood*, volume 28 of *Computer Science Monographs*. Institute of Statistical Mathematics, Minato-ku, Tokyo.

Felsenstein, J. (1985) Confidence limits on phylogenies: An approach using the bootstrap. *Evolution*, **39**, 783–791.

Flouri, T., Izquierdo-Carrasco, F., Darriba, D., Aberer, A. J., Nguyen, L.-T., Minh, B. Q., von Haeseler, A. and Stamatakis, A. (2014) The Phylogenetic Likelihood Library. *Systematic Biology*, **63**, accepted.

Gascuel, O. (1997) BIONJ: An improved version of the NJ algorithm based on a simple model of sequence data. *Mol. Biol. Evol.*, **14**, 685–695.

Guennebaud, G., Jacob, B. *et al.* (2010) Eigen v3. http://eigen.tuxfamily.org.

Guindon, S., Dufayard, J.-F., Lefort, V., Anisimova, M., Hordijk, W. and Gascuel, O. (2010) New algorithms and methods to estimate maximum-likelihood phylogenies: Assessing the performance of phyml 3.0. *Syst. Biol.*, **59**, 307–321.

Kishino, H. and Hasegawa, M. (1989) Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in Hominoidea. *J. Mol. Evol.*, **29**, 170–179.

Kishino, H., Miyata, T. and Hasegawa, M. (1990) Maximum likelihood inference of protein phylogeny and the origin of chloroplasts. *J. Mol. Evol.*, **31**, 151–160.

Lanfear, R., Calcott, B., Ho, S. Y. W. and Guindon, S. (2012) PartitionFinder: Combined selection of partitioning schemes and substitution models for phylogenetic analyses. *Mol. Biol. Evol.*, **29**, 1695–1701.

Lewis, P. O. (2003) NCL: a C++ class library for interpreting data files in NEXUS format. *Bioinformatics*, **19**, 2330–2331.

Mascagni, M. and Srinivasan, A. (2000) Algorithm 806: SPRNG: A scalable library for pseudorandom number generation. *ACM Transactions on Mathematical Software*, **26**, 436–461.

Meyer, S. and von Haeseler, A. (2003) Identifying site-specific substitution rates. *Mol. Biol. Evol.*, **20**, 182–189.

Minh, B. Q., Nguyen, M. A. T. and von Haeseler, A. (2013) Ultrafast approximation for phylogenetic bootstrap. *Mol. Biol. Evol.*, **in press**.

Minh, B. Q., Vinh, L. S., von Haeseler, A. and Schmidt, H. A. (2005) pIQPNNI – parallel reconstruction of large maximum likelihood phylogenies. *Bioinformatics*, **21**, 3794–3796.

Posada, D. and Crandall, K. A. (1998) MODELTEST: Testing the model of DNA substitution. *Bioinformatics*, **14**, 817–818.

Schmidt, H. A., Strimmer, K., Vingron, M. and von Haeseler, A. (2002) TREE-PUZZLE: Maximum likelihood phylogenetic analysis using quartets and parallel computing. *Bioinformatics*, **18**, 502–504.

Shimodaira, H. and Hasegawa, M. (1999) Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Mol. Biol. Evol.*, **16**, 1114–1116.

Strimmer, K. and Rambaut, A. (2002) Inferring confidence sets of possibly misspecified gene trees. *Proc. R. Soc. Lond. B*, **269**, 137–142.

Vinh, L. S. and von Haeseler, A. (2004) IQPNNI: Moving fast through tree space and stopping in time. *Mol. Biol. Evol.*, **21**, 1565–1571.

Weiss, G. and von Haeseler, A. (2003) Testing substitution models within a phylogenetic tree. *Mol. Biol. Evol.*, **20**, 572–578.