BAli-Phy User's Guide v3.0-beta4

Benjamin Redelings

Table of Contents

1. Introduction

BAli-Phy is a Unix command line program that is developed primarily on Linux. *BAli-Phy* also runs on Windows and Mac OS X, but it is not a GUI program and so you must run it in a terminal. Therefore, you might want to keep a Unix tutorial [http://www.ee.surrey.ac.uk/Teaching/Unix] or Unix cheat sheet [http://www.rain.org/~mkummel/unix.html] handy while you work.

In addition to the main **bali-phy** executable, *BAli-Phy* comes with a collection of small command-line utilities such as **alignment-cat**, **trees-consensus**, etc. These utilities can be used to process alignments, assemble data sets, and summarize the results of MCMC.

2. Installation

2.1. Hardware requirements

We typically run *BAli-Phy* on workstations with at least 8Gb of RAM and 2 cores. More cores will allow you to run more MCMC chains at once, and more RAM will allow you to run larger data sets. However, it is often easier and faster to run BAli-Phy on a (Linux) computing cluster, if you have one available.

2.2. MS Windows requirements

2.2.1. Installing Msys2

Before you can use *BAli-Phy* on Windows you need to install a Unix command-line environment. We recommand installing the 64-bit version of MSYS2 [http://www.msys2.org]. You may then access the Unix command line environment by running the MSYS2 shell (not the normal windows command line). The MSYS2 shell mounts the `C:` drive on `/c/`, so you can access the directory `C:/Users/` as `/c/Users/`, for example.

2.2.2. Install BAli-Phy executables from website

First, download and extract the executables:

```
% mkdir -p ~/Applications
% cd ~/Applications
% wget http://www.bali-phy.org/files/bali-phy-3.0-beta4-win64.tar.gz
% tar -zxf bali-phy-3.0-beta4-win64.tar.gz
```

Second, check that the **bali-phy** executable runs:

```
% ~/Applications/bali-phy-3.0-beta4/bin/bali-phy --version
```

Third, add the executables to your PATH, so that the shell knows where to find them:

```
% export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH
% test -r ~/.bash_profile && echo 'export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH
% echo 'export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH' >> ~/.profile
```

2.3. Mac OS X

2.3.1. Install BAli-Phy using homebrew (recommended)

First install XCode [https://developer.apple.com/xcode/] version 6 or higher. Then install homebrew [http://brew.sh/]:

```
% /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master
```

Then use homebrew to compile and install **bali-phy**:

```
% brew tap bredelings/bioinformatics
% brew install bali-phy
```

Check that the executable runs:

```
% bali-phy --version
```

2.3.2. Install BAli-Phy using executables from website (alternative)

First download and extract the executables:

```
% mkdir -p ~/Applications
% cd ~/Applications
% curl -O http://www.bali-phy.org/files/bali-phy-3.0-beta4-mac64.tar.gz
% tar -zxf bali-phy-3.0-beta4-mac64.tar.gz
```

Check that the executable runs:

```
% ~/Applications/bali-phy-3.0-beta4/bin/bali-phy --version
```

Then, add it to your PATH, so that the shell knows where to find it:

```
% export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH
% test -r ~/.bash_profile && echo 'export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH
% echo 'export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH' >> ~/.profile
```

2.3.3. Install programs used by bp-analyze

- You can install **gnuplot** via homebrew:

```
%  brew install gnuplot
```

- You can install R from MRAN [https://mran.microsoft.com/open/].

2.4. Linux

2.4.1. Install BAli-Phy using executables from website (alternative)

First install **wget**. If you have Debian or Ubuntu Linux, type:

```
% sudo apt-get install wget
```

Then download and extract the executables:

```
% mkdir -p ~/Applications
% cd ~/Applications
% wget http://www.bali-phy.org/files/bali-phy-3.0-beta4-linux64.tar.gz
% tar -zxf bali-phy-3.0-beta4-linux64.tar.gz
```

Second, check that the executable runs:

```
% ~/Applications/bali-phy-3.0-beta4/bin/bali-phy --version
```

Third, add it to your PATH, so that the shell knows where to find it:

```
% export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH
% test -r ~/.bash_profile && echo 'export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH' >
% echo 'export PATH=~/Applications/bali-phy-3.0-beta4/bin:$PATH' >> ~/.profile
```

2.4.2. Install programs used by bp-analyze

If you have Debian or Ubuntu linux, you can install other recommended programs by typing:

```
% sudo apt-get install gnuplot
% sudo apt-get install r-base
% sudo apt-get install seaview
```

2.5. Adding BAli-Phy to your `PATH`

2.5.1. I have a path?

If you installed *BAli-Phy* to the directory `~/Applications`, then you can run bali-phy by typing ~/**Applications/bali-phy-3.0-beta4/bin/bali-phy**. However, it would be much nicer to simply type **bali-phy** and let the computer find the executable for you. This can be achieved by putting the directory that contains the *BAli-Phy* executables into your "path". The "path" is a colon-separated list of directories that is searched to find program names that you type. It is stored in an environment variable called `PATH`.

Setting your `PATH` is also a pre-requisite for running the **bp-analyze.pl** script to summarize your MCMC runs.

2.5.2. Examining your `PATH`

You can examine the current value of this environment variable by typing:

```
% echo $PATH
```

We will assume that you extracted the bali-phy archive in `~/Applications` and so you want to add `$HOME/Applications/bali-phy-3.0-beta4/bin` to your `PATH`. (If you installed to another directory, replace `$HOME/Applications/bali-phy-3.0-beta4/` with that directory.)

2.5.3. Adding BAli-Phy to your `PATH`

The commands for doing this depend on what "shell" you are using. Type **echo $SHELL** to find out. If your shell is **sh** or **bash** then the command looks like this:

```
% PATH=$HOME/Applications/bali-phy-3.0-beta4/bin:$PATH
```

If your shell is **csh** or **tcsh**, then the command looks like this:

```
% setenv PATH $HOME/Applications/bali-phy-3.0-beta4/bin:$PATH
```

Note that these commands will only affect the window you are typing in, and will vanish when you reboot.

2.5.4. Making the change stick

To make this change survives when you logout or reboot, open your shell configuration file in a text editor, and add the command on a line by itself. This will ensure that it is run every time you log in.

To find the right configuration file, look in your $HOME directory for `.profile` (for the Bourne shell **sh**), `.bash_profile` (for BASH), or `.login` (for tcsh). You may have to create the file if it is not present. On Cygwin, you should put the change in the file `.bashrc`.

If you do not know which directory is your home directory, you can find its full name by typing:

```
% echo $HOME
```

2.6. Additional required software

The following software is important to install:

- The graphical MCMC diagnostic program Tracer [http://tree.bio.ed.ac.uk/software/tracer/].

The following software is recommended to install:

- The plotting program gnuplot.
- The phylogeny-viewer FigTree [http://tree.bio.ed.ac.uk/software/figtree/].
- The alignment-viewer Seaview [http://pbil.univ-lyon1.fr/software/seaview.html].

GNUplot can be installed using the Cygwin installer on Windows systems. It can also be installed using macports [http://www.macports.org] or homebrew [http://http://mxcl.github.com/homebrew/] on Macintosh systems, but installing these package managers requires first installing Xcode Developer Tools.

2.7. Testing the installed software

In order to determine that the software has been correctly installed, and the `PATH` has been correctly set, run the following commands:

```
% bali-phy ~/Applications/bali-phy-3.0-beta4/share/bali-phy/examples/sequences/5S-rRNA/5d.
% bp-analyze.pl 5d-1/
```

Furthermore, the directory `5d-1` should contain a file called `C1.log`. You should be able to load this file in Tracer, although the chain will not really have converged yet.

3. Compiling *BAli-Phy*

Compiling *BAli-Phy* is intended to be a relatively painless process. However, most people will want to download a compiled version [http://bali-phy.org/download.php].

3.1. Setup

In order to compile BAli-Phy, you need a C++ compiler than understands the C++14 standard. We recommend the GNU C ++ Compiler (GCC [http://gcc.gnu.org]) version 5.0 (or higher) or the Clang [http://clang.llvm.org] compiler version 3.5.0 or higher. You will also need recent versions of autoconf, automake, and libtool to generate the configure script. The Cairo [http://www.cairographics.org/] graphics library is optional, but if it is missing, the **drawtree** tool that is used to draw consensus trees won't be built. See also Section 2.6, “Additional required software” [5].

3.1.1. Linux

On Debian and Ubuntu, you can type:

```
% sudo apt-get install g++
% sudo apt-get install libtool
% sudo apt-get install autoconf
% sudo apt-get install automake
% sudo apt-get install git
% sudo apt-get install libcairo2-dev
```

3.1.2. Mac

On Mac OS X, the simplest way to get a compiler is to install XCode [https://developer.apple.com/xcode/] (version 6 or newer), which comes with Clang [http://clang.llvm.org]. To get the other tools, first install homebrew [http://brew.sh/], and then type:

```
% brew install libtool
% brew install autoconf
% brew install automake
% brew install git
% brew install cairo
```

3.1.3. Windows

We recommend compiling windows executables in the MINGW format instead of the Cygwin [http://www.cygwin.com/] format. MINGW executables are native windows executables, and do not need `cygwin.dll` to run.

However, the easiest way to build MINGW executables is to use either Linux or Cygwin as the host environment. This is called using a "cross-compiler". You can obtain cross compilers for MINGW on both Linux and Cygwin. To inform the `configure.sh` script that you wish to use a cross compiler, add the **`--host`** flag. You should add the flag **`--host=x86_64-w64-mingw32`** to build 64-bit windows, and **`--host=i686-w64-mingw32`** to build 32-bit executables.

3.2. Clone, Configure, Compile

First check out the code using git, and generate the configure script:

```
% git clone https://github.com/bredelings/BAli-Phy.git
% (cd BAli-Phy ; ./bootstrap.sh)
```

Then create a *separate* build directory, enter it, and run the configure command:

```
% mkdir build
% cd build
% ../BAli-Phy/configure --prefix=$HOME/Applications/bali-phy-3.0-beta4/
```

Finally, build and install the software:

```
% make
% make install
```

The command **bali-phy** and its associated tools should then be located in `~/Applications/bali-phy-3.0-beta4/bin/`. To install to another directory *`dir`*, specify --prefix=*`dir`* to **configure**.

3.3. Options: compiler and linker flags

You can select the C++ compiler by setting the CXX variable. A useful example of this is to use **g++-5** on systems where **g++** invokes a compiler that is too old:

```
% ../BAli-Phy/configure --prefix=$HOME/Applications/bali-phy-3.0-beta4 CXX=g++-5
```

You may also set compiler and linker options using the CXXFLAGS and LDFLAGS variables. A useful example of this is to produce an OS X executuable on that can run on older versions of OS X:

```
% ../BAli-Phy/configure --prefix=$HOME/Applications/bali-phy-3.0-beta4 CXXFLAGS="-mmacosx-
```

3.4. Options: where to find libraries and header files (e.g. Cairo)

You can instruct the compiler to look for include files in directory *`dir`* by passing *`--with-extra-includes=dir`* to the **configure** script.

You can instruct the compiler to look for libraries files in directory *`dir`* by passing *`--with-extra-libs=dir`* to the **configure** script.

For example, if your system has Cairo installed in `/usr/local/`, then you might need to add "**`--with-extra-includes=/usr/local/include --with-extra-libs=/usr/local/lib`**" to the configure script arguments so that the compiler can find the Cairo include files and libraries.

4. Running the program

Here are some examples and explanations of how to run **bali-phy**. You can get an overview of command line options by running **bali-phy --help**.

We recommend running multiple chains in parallel for each command, because

1. You can combine the samples, leading to faster run times.
2. You can compare the runs to determine if the chains have converged.

This can be done simply by starting several instances of the program, and does not require using MPI or special command-line options.

4.1. Quick Start

The simplest way to run **BAli-Phy** is to type all the arguments on the command line:

```
% bali-phy sequence-file
```

Here *`sequence-file`* is a FastA or PHYLIP file containing the sequences you wish to analyze. The filename should end in **`.fasta`** or **`.phy`** to indicate which format it is using.

In this simple example, **bali-phy** automatically detects whether *`sequence-file`* contains DNA, RNA, or Amino-Acids and uses default values for several command line options. Thus, if *`sequence-file`* contains DNA, then this is equivalent to the more verbose command line

```
% bali-phy sequence-file --alphabet DNA --smodel TN --imodel RS07 --iterations=100000
```

Here the substitution model is Tamura-Nei, the insertion/deletion model is RS07, and the number of iterations is 100,000. If *sequence-file* contains amino acids, then the defaults will be:

```
% bali-phy sequence-file --alphabet Amino-Acids --smodel LG --imodel RS07 --iterations=100000
```

4.2. Command line options

You can specify a more complex substitution model as follows (See Section 7.2, “Basic CTMC models” [16]):

```
% bali-phy sequence-file --smodel LG+Rates.Gamma+INV
```

You may specify an indel model of **none** to fix the alignment to its initial value, and ignore information in shared insertions or deletions.

```
% bali-phy sequence-file --imodel none
```

4.3. Multiple genes or data partitions

You may analyze multiple genes by putting each one it its own data partition:

```
% bali-phy sequence-file1 sequence-file2
```

You should put the data from the first gene in *sequence-file1* and the second gene in *sequence-file2*. The sequence names in both files should be the same. In this scenario, both genes share the same tree, but their alignments vary independently. Furthermore, the branch lengths for each gene are scaled by an independent factor. By default, each partition will have its own default alphabet, substitution model, insertion/deletion model, and tree length.

By default, each partition will recieve an independent copy of the model, and will not share parameter values:

```
% bali-phy sequence-file1 sequence-file2 --smodel TN --imodel RS07
```

However, you can select partition-specific values for 5 options: **--smodel**, **--imodel**, **--alphabet**, and **--scale**. For example, to specify different substitution models but the same alphabet:

```
% bali-phy sequence-file1 sequence-file2 --smodel 1:TN --smodel 2:GTR --alphabet 1,2:DNA
```

You can fix the alignment and ignore insertion/deletion information in one partition, while allowing the alignment to vary and using insertion/deletion information in another partition:

```
% bali-phy sequence-file1 sequence-file2 --imodel 1:RS07 --imodel 2:none
```

You can also specify that two partitions share a single copy of a single substitution model or indel model. This reduces the number of parameters and also pools information between the partitions:

```
% bali-phy sequence-file1 sequence-file2 --smodel 1,2:TN --imodel 1,2:RS07
```

By default each partition has a separate scale, but you can force groups of partitions to share a scale. If you leave the value of the scale blank, the default distribution on scales will be used:

```
% bali-phy sequence-file1 sequence-file2 --smodel 1:TN --smodel 2:GTR --scale 1,2:
```

Finally, you may specify **-Inone** or **--imodel none**, which affects all partitions:

```
% bali-phy sequence-file1 sequence-file2 --smodel 1:TN --smodel 2:GTR -t
```

4.4. Running on computing clusters

Running **bali-phy** on a computing cluster is not necessary, but can speed up the analysis dramatically. This is because a cluster allows you to run several *independent* MCMC chains simultaneously and pool the resulting samples. You can run

multiple chains simultaneously simply by starting several different instances of **bali-phy**. Each instance of bali-phy runs only one chain and does not require using MPI or special command-line options.

This approach to parallel computation is sometimes more efficient than MCMCMC-based parallelism involving heated chains. It is equivalent to running MCMCMC with no temperature difference between chains, with the exception that it allows results from *all* chains to be used, instead of just results from the single "cold" chain. Thus, if you run 10 independent chains in parallel, then you may gather samples 10 times faster that a single chain.

4.5. Option files (Scripts)

In addition to using the command line, you may also specify options in a file. Using an option file can be more convenient if you are going to run the same analysis many times, or if the number of options is large. Furthermore, the option file may contain comments and blank lines. Option files are a good to record what options you used in an analysis, and why.

An option file is specified with the command line option **`--config`** ***`file`*** or **`-c`**`file`. If values for an option are given both on the command line and in an option file, then the command line value overrides the value in the option file.

4.5.1. Syntax

Option files use the same option names as the command line. However, the syntax is different: each option is given on its own line using the syntax "**`option = value`**" instead of the syntax "**`--option value`**". If the option has no value then it is given using the syntax "**`option = option`**".

4.5.2. Example

For example, consider the following option file:

```
#select a data set to analyze
align = examples/sequences/EF-Tu/5d.fasta

#select an substitution model
smodel = LG+Rates.logNormal+INV

#fix the alignment and do not model indels
imodel = none
```

The first option, **`align`**, is the name of the sequence file, which has no name on the command line. Lines that begin with # are comments, and blank lines are ignored.

```
% bali-phy examples/sequences/EF-Tu/5d.fasta --smodel LG+Rates.logNormal+INV --imodel none
```

4.5.3. The configuration file

The file `~/.bali-phy` is a special option file called the *configuration file*. If it exists, it is always loaded. Options given on the command line or an option file override values given in `~/.bali-phy`.

4.6. Examples

Here are some examples which demonstrate how to run *BAli-Phy*. In order to run these examples, you must find the `examples/sequences/` directory which contains the example files. If you downloaded executables and extracted them in the `~/Applications` directory, then the `examples/sequences/` directory will be found at `~/Applications/bali-phy-3.0-beta4/share/bali-phy/examples/sequences/`.

Also note that **bali-phy** does *not* run until it is "finished", but continues to gather samples until the user determines that enough samples have been gathered, and stops it. Thus, it is useful to continually examine the output files while the program is running.

Example 1. No frills

Here we analyze the EF-Tu 5-taxon data set provided with the software.

```
% bali-phy ~/Applications/bali-phy-3.0-beta4/share/bali-phy/examples/sequences/EF-Tu/5d.fast
```

Example 2. Multiple-Rate Substitution Model

We now modify the previous example by changing the substitution model to allow log-normal-distributed rate variation and invariant sites. The amount of rate variation and the fraction of invariant sites are estimated

```
% bali-phy ~/Applications/bali-phy-3.0-beta4/share/bali-phy/examples/sequences/EF-Tu/5d.fast
```

Example 3. Fixed alignment

Here we use the 5S rRNA 25-taxon data set provided with the software. The **`-Inone`** option is used, fixing the alignment and making indels non-informative.

```
% bali-phy ~/Applications/bali-phy-3.0-beta4/share/bali-phy/examples/sequences/5S-rRNA/25-mu
```

5. Input

5.1. Sequence formats

BAli-Phy can read in sequences and alignments in both FastA and PHYLIP formats. Filenames for FastA files should end in **`.fasta`**, **`.mpfa`**, **`.fna`**, **`.fas`**, **`.fsa`**, or **`.fa`**. Filenames for PHYLIP files should end in **`.phy`**. If one of these extensions is not used, then *BAli-Phy* will attempt to guess which format is being used.

5.2. Is my data set too large?

Large data sets run more slowly than small data sets. We recommend a conservative starting point with few taxa and short sequence lengths. You can then increase the size of your data set until a balance between speed and size is reached. The tool *alignment-thin* described in Section 11, “Alignment utilities” [26] can be used to construct a smaller data set.

The number of samples that you need depends on whether you are primarily interested in obtaining a point estimate or in obtaining detailed measures of confidence and uncertainty. For detailed measures of confidence and uncertainty you should obtain a minimum of 10,000 samples after the Markov chain converges. For an estimate, you don't need very many samples after convergence. (But you may need many samples to be sure that you've converged!)

See also Section 4.4, “Running on computing clusters” [8].

5.2.1. Too many taxa?

BAli-Phy is quite CPU intensive, and so we recommend using 50 or fewer taxa in order to limit the time required to accumulate enough MCMC samples. (Despite this recommendation, data sets with more than 100 taxa have occasionally been known to converge.) We recommend initially pruning as many taxa as possible from your data set, then adding some back if the MCMC is not too slow.

5.2.2. Sequences too long?

Aligning just a pair of sequences takes time and memory, where represents the sequence length. Therefore sequences longer than (say) 1000 letters become increasingly impractical. However, you might try to see how long you can make your sequences before you run out of memory, or the program becomes too slow.

For multi-gene analyses, two separate data partitions (i.e. genes) of 500 letters will be twice as fast to align as one data partition of 1000 letters. So, it may be possible to analyze several genes as long as each gene individually is not too long.

You can speed up alignment for long genes by specifying alignment constraints (See Section 9, “Alignment constraints” [23]). Ideally, 10 evenly spaced constraints should reduce the cost of re-aligning a sequence by a factor of 10.

Also, note that you can sometimes speed up the analysis of protein sequences by coding them as amino acids or codons, rather than nucleotides. This is because it decreases the sequence length.

6. Output

6.1. Output directory

BAli-Phy creates a new directory to store its output files each time it is run. By default, the directory name is the name of the sequence file, with a number added on the end to make it unique. *BAli-Phy* first checks if there is already a directory called `file-1/`, and then moves on to `file-2/`, etc. until it finds an unused directory name.

You can specify a different name to use instead of the sequence-file name by using the **`--name`** option.

6.2. Output files

BAli-Phy writes the following output files inside the directory that it creates:

C1.out

Iteration numbers, probabilities, success probabilities for transition kernels, etc..

C1.P*p*.fastas

Sampled alignments for partition *p*

C1.err

Log file for hopefully irrelevant error messages.

C1.MAP

Successive estimates of the MAP alignment, tree and parameters.

C1.log

Numeric parameters: indel and substitution rates, etc.

C1.trees

Tree samples: one sample per line, in Newick format.

For the last two files, each line in these files corresponds to one iteration.

6.2.1. Field names in `C1.log`

This section explains the meaning of the various field names in the file `C1.log`.

6.2.1.1. Computed parameter names

prior

The log prior probability. This includes the probability of the alignment, since the alignment is not observed.

prior_A*n*

The log of the probability of the alignment of the th partition, given the topology , the branch lengths , and insertion-deletion process parameters . This log probability is the probabilistic equivalent of a gap penalty on the alignment given the scoring parameters .

likelihood

The log of the likelihood. Conditional on the alignment, this is determined entirely by the substitution model, and ignores insertions and deletions. This is the probabilistic equivalent of the mismatch penalty.

logp

The log of the probability. The probability is the product of the prior and the likelihood.

|A|

The total number of alignment columns across all partitions.

#indels*n*

The number of indel events in partition *n*, if we group adjacent indels that occur on the same branch.

#indels

The total number of indel events across all partitions, if we group adjacent indels that occur on the same branch.

|indels*n*|

The length of indel events in partition *n*, if we group adjacent indels that occur on the same branch.

|indels|

The total length of indel events across all partitions, if we group adjacent indels that occur on the same branch.

#substs*n*

The unweighted parsimony score for substitutions in partition *n*.

#substs

The total unweighted parsimony score for substitutions across all partitions.

Scale*n* * |T|

The branch lengths for partition group *n*.

6.2.1.2. Model parameter names

The prefixes "S*n*::" and "I*n*::" will be dropped if not necessary to disambiguate parameters with the same name in different sub-models.

Scale*n*

The average number of substitutions per branch in partition group *n*. This ordinarily applies to the *n*th partition, unless multiple partitions are forced to have the same branch length scale using **`--scale`**.

S*n*::*name*

Parameter *name* in the *n*th substitution model.

I*n*::*name*

Parameter *name* in the *n*th insertion/deletion model.

6.3. Summarizing the output

This section is primarily oriented to extracting estimates from output files. See Section 10, “Convergence and Mixing: Is it done yet?” [24] for methods of determine effective sample sizes, and for checking mixing and convergence.

6.3.1. Finding the consensus tree (`C1.trees`)

To compute the majority consensus tree, do the following. (The program FigTree [http://tree.bio.ed.ac.uk/software/figtree/] allows you to view the resulting tree file graphically.)

```
% trees-consensus C1.trees > c50.PP.tree
```

You can (and should) pool results from different MCMC runs by adding multiple tree sample files on the command line. The different MCMC runs should have the same input files and parameters.

```
% trees-consensus dir-1/C1.trees dir-2/C1.trees > c50.PP.tree
```

By default, the first 10% of tree samples are skipped as burn-in. You can specify the number of samples (e.g. 1000) to skip by adding the options **`-s1000`** or **`--skip=1000`**. You may also specify a percentage of all samples:

```
% trees-consensus -s20% C1.trees > c50.PP.tree
```

To discard some samples, keeping (say) every 10th sample, you may add the options **`-x10`** or **`--subsample=10`**. This can make the program a lot faster, at the possible expense of some loss in accuracy.

```
% trees-consensus -s20% -x10 C1.trees > c50.PP.tree
```

By default, splits are included in the consensus tree if they have a PP greater than 0.5. You can specify a more stringent level (e.g. 0.66) by adding the option **`--consensus-PP=0.66`** as follows:

```
% trees-consensus -s20% -x10 --consensus-PP=0.66 C1.trees > c66.PP.tree
```

You may also make the program write directly to the output file (e.g. `c66.PP.tree`) by using the more general form **`--consensus-PP=0.66:c66.PP.tree`**. Leaving off the "**`:c66.PP.tree`**" part (as we did above) or specifying "**`:-`**" sends the output to the standard output (e.g. the terminal, if not redirected).

```
% trees-consensus -s20% -x10 C1.trees --consensus-PP=0.66:c66.PP.tree
```

You can supply multiple levels and filenames separated by commas. This is faster than running the program multiple times with different consensus levels.

```
% trees-consensus -s20% -x10 C1.trees --consensus-PP=0.5:c50.PP.tree,0.66:c66.PP.tree
```

Finally, you may use the option **`--consensus=`** instead of the option **`--consensus-PP=`** if you do not wish the resulting tree to contain embedded posterior probabilities on branches, as well as branch lengths.

```
% trees-consensus -s20% -x10 C1.trees --consensus=0.5:c50.PP.tree,0.66:c66.PP.tree
```

Both the **`--consensus=`** and **`--consensus-PP=`** options may be given simultaneously.

See **`trees-consensus --help`** for a complete list of options.

6.3.2. Finding the M.A.P. topology (`C1.trees`)

To compute the *maximum a posteriori* tree topology do:

```
% trees-consensus --skip=burnin C1.trees --map-tree=MAP.tree
```

The MAP topology may be used instead of a consensus tree when a fully resolved (e.g. bifurcating) tree is required. However, when the topology has many tips, each topology may be sampled only once, leading to low quality estimates of the MAP topology.

The program FigTree [http://tree.bio.ed.ac.uk/software/figtree/] allows you to view the consensus tree graphically.

6.3.3. Checking topology convergence (`C1.trees`)

```
% trees-bootstrap dir-1/C1.trees dir-2/C1.trees
```

This command computes the effective sample size for the posterior probability of each split. It also computes the Average Standard Deviation of Split Frequencies (ASDSF) between two or more independent runs.

See Section 10, “Convergence and Mixing: Is it done yet?” [24] for more information.

6.3.4. Summarizing numerical parameters (`C1.log`)

This command gives a median and confidence interval, ESS, and a stabilization time:

```
% statreport C1.log > Report
```

This command compares multiple runs to give PSRF and joint ESS values as well:

```
% statreport dir-1/C1.log dir-2/C1.log > Report
```

The program Tracer [http://tree.bio.ed.ac.uk/software/tracer/] allows you to view the same summaries graphically.

See Section 10, “Convergence and Mixing: Is it done yet?” [24] for more information.

6.3.5. Computing an alignment using Posterior Decoding (C1.P*p*.fastas)

```
% cut-range --skip=burn-in < C1.Pp.fastas | alignment-max > Pp-max.fasta
```

You can use the program seaview [http://pbil.univ-lyon1.fr/software/seaview.html] to view the alignment graphically.

6.3.6. Find the alignment from the maximum a posterior (MAP) point (C1.MAP)

```
% alignment-find < C1.MAP > P1-MAP.fasta
```

6.3.7. Create an Au (Alignment Uncertainty) plot (C1.P*p*.fastas)

To annotate a specific alignment *alignment*.fasta, choose a fully resolved tree estimate *tree*:

```
% cut-range --skip=burn-in < C1.Pp.fastas | alignment-gild alignment.fasta tree  > alignment-
% alignment-draw alignment.fasta --AU alignment-AU.prob > alignment-AU.html
```

The majority consensus tree is usually not fully resolved, so we recommend using the MAP topology instead.

6.4. Summarizing the output - scripted

Instead of manually running each of the steps to analyze the output files, you may instead run the PERL script **bp-analyze.pl** to execute these commands. The script will create an HTML page `Results/index.html` that summarizes the posterior distribution.

You may run **bp-analyze.pl** inside the output directory, like this:

```
% bp-analyze.pl --burnin=iterations
```

You may also run it with one or more output directories as arguments, like this:

```
% bp-analyze.pl --burnin=iterations directory-1/ directory-2/
```

In this case, output from multiple runs will be used to assess convergence and mixing, as well as to increase the precision of the estimates.

All the commands that are executed by **bp-analyze.pl** will be logged to `Results/bp-analyze.log`. You can also see these commands as they are executed by supplying the **--verbose** option:

```
% bp-analyze.pl --burnin=iterations --verbose
```

6.4.1. Meaning of generated files

The `Results/` directory will contain the following useful files:

Report

A summary of numerical parameters: credible intervals and mixing.

consensus

A summary of supported splits (clades).

c-levels.plot

The number of splits (clades) supported at each LOD level.

c50.tree

The majority consensus topology + branch lengths (Newick format)

c50.PP.tree

The majority consensus topology + branch lengths + Posterior Probabilities (Newick format)

MAP.tree

An estimate of the MAP topology + branch lengths (Newick format)

The following files will be generated to summarize alignment uncertainty, unless the analysis uses a fixed alignment.

MAP.fasta

An estimate of the MAP alignment.

P*p*-max.fasta

An estimate of the alignment for partition *p* using maximum posterior decoding.

MAP-AU.html

An AU plot of the MAP alignment (AA/DNA color-cheme).

P*p*-max-AU.html

An AU plot of the maximum posterior decoding alignment for partition *p* (AA/DNA color-cheme).

The following files describe convergence and mixing:

partitions.bs

Confidence intervals on the support for partitions, generated using a block bootstrap.

partitions.SRQ

A collection of SRQ plots for the supported partitions.

c50.SRQ

An SRQ plot for the majority consensus tree.

The SRQ plots can be viewed by typing "**`plot 'file' with lines`**" in *gnuplot*.

6.4.2. `Mixing/partitions.bs`: partition mixing

This file reports the quality of estimates of support for each partition in terms of the posterior probability (PP) and log-10 odds (LOD). It also reports the auto-correlation time (ACT), the effective sample size (Ne), the number of samples that support (1) or do not support (0) the partition, and the number of regenerations. Only partitions with PP > 0.1 are shown by default.

7. Substitution models

Substitution models in *BAli-Phy* are specified using a stack, as follows: **`Model[arg]+Model[arg]+...+Model[arg]`** where each model uses the previous models as input. For example, **`LG+Rates.Gamma[4]+INV`**. Some arguments are optional.

> **Note**
>
> If you are using the C-shell command line shell (**csh** or **tcsh**), then it will try to interpret each argument as an array reference, giving the error message "bali-phy: No match." To avoid this, put quotes around the substitution model, like this: **`--smodel="Model[arg]+Model[arg]+...+Model[arg]"`**.

7.1. Default substitution models

If the substitution model is not specified, then the default model for the alphabet is used. For DNA or RNA, the default model is TN. For Triplets, the default is TNx3. For Codons, the default model is M0. For Amino-Acids, the default model is LG.

7.2. Basic CTMC models

The basic substitution models in *BAli-Phy* are continuous-time Markov chains (CTMC). CTMC models can be characterized by transition rates from letter to letter . After a given time the probability for transition from state to state is given by using a matrix exponential. Becase the CTMC models used in *BAli-Phy* are all reversible, the rate matrix for these reversible models can be decomposed into a symmetric matrix and equilibrium frequencies as follows: The matrix is called the exchangability matrix, and represents how exchangeable letters and are, independent of their frequencies.

The basic CTMC models are EQU, HKY, TN, GTR, HKYx3, TNx3, GTRx3, JTT, WAG, LG, and M0. Each of these models is a way of specifying the exchangeability matrix .

Table 1. Substitution Models

Model	Alphabet	Parameters
`EQU`	any	none
`HKY` Hasegawa, Kishino, Yano (1985)	DNA or RNA	kappa: the ts/tv ratio.
`TN` Tamura, Nei (1993)	DNA or RNA	kappaPur: the purine ts/tv ratio. kappaPyr: the pyrimidine ts/tv ratio.
`GTR` General Time-Reversible Tavare (1986)	DNA or RNA	
`JTT` Jones, Taylor, Thornton (1992)	Amino-Acids	none.
`WAG` Whelan and Goldman (2001)	Amino-Acids	none.
`LG` Le and Gascuel (2008)	Amino-Acids	none.
`Empirical[`*`file`*`]`	Amino-Acids	none.
`HKYx3` **`TNx3`** **`GTRx3`**	Triplets	*`nuc-model`* parameters.
`M0`	Codons	omega: the dN/dS ratio

Model	Alphabet	Parameters
`M0[HKY]` (default) **`M0[GTR]`** Nielsen and Yang (1998)		

7.3. Substitution Frequency models

The rate matrix can be more generally expressed as where ranges from to . Here the parameter specifies the relative importance of unequal conservation () and unequal replacement () in maintaining the equilibrium frequencies .

In fact, this can be generalized even further to where

These models can therefore be expressed as a combination of an "exchange model" (for) and a "frequency model" (for).

Table 2. Frequency Models

Model	Alphabet	Parameters
`F` Whelan and Goldman (2001)	any	pi$l$: frequency of letter l.
`F61` Synonym of F for codon models	any	pi$l$: frequency of letter l.
`gwF` Goldman and Whelan (2002)	any	f: determines cause of high-frequency letters. pi$l$: frequency of letter l.
`F1x4` Single nucleotide frequency model	Triplets	pi$l$: frequency of nucleotide l in first codon position.
`F3x4` Independent nucleotide frequency model	Triplets	Site1.pi$l$: frequency of letter l in first codon position. Site2.pi$l$: frequency of letter l in first codon position. Site3.pi$l$: frequency of letter l in first codon position.
`MG94` Muse and Gaut (1994) Single nucleotide frequency model	Triplets	pi$l$: frequency of nucleotide l in first codon position.
`MG94w9` Muse and Gaut (1994) Independent nucleotide frequency model	Triplets	Site1.pi$l$: frequency of letter l in first codon position. Site2.pi$l$: frequency of letter l in first codon position. Site3.pi$l$: frequency of letter l in first codon position.

7.4. Substitution Mixture Models

Complex substitution models in *BAli-Phy* are constructed as mixtures of reversible CTMC models (see Section 7.2, “Basic CTMC models” [16]) that run at different rates (e.g.) or have different parameters (e.g. an M2a codon model).

Table 3. CTMC Mixture Models

Model	Alphabet	Parameters
model **+ Rates.Gamma** ***model*** **+ Rates.Gamma[4]** (default) Yang (1994)	*model* alphabet	**sigmaOverMu** noise to signal ratio for Gamma(a,b). Also equal to 1/sqrt(a). **a** parameter for Gamma(a,b) **b** parameter for Gamma(a,b)
model **+ Rates.logNormal** ***model*** **+ Rates.logNormal[4]** (default)	*model* alphabet	**sigmaOverMu** noise to signal ratio for . **pInv** Fraction of invariant sites.
model **+ INV**	*model* alphabet	**pInv** Fraction of invariant sites.
M1a **M1a[HKY,F61]** (default) **M1a[GTR,F3x4]** Wong, et. al. (2004)	Codons	**w1** the conserved dN/dS ratio **f1** the fraction of conserved sites among non-positively selected sites **f2** the fraction of neutral sites among non-positively selected sites
M2a **M2a[HKY,F61]** (default) **M2a[GTR,F3x4]** Wong, et. al. (2004)	Codons	**w1** the conserved dN/dS ratio **f1** the fraction of conserved sites among non-positively selected sites **f2** the fraction of neutral sites among non-positively selected sites **posW** the positively selected dN/dS ratio **posP** the fraction of positively selected sites

Model	Alphabet	Parameters
`M2a_Test` **`M2a_Test[HKY,F61]`** (default) **`M2a_Test[GTR,F3x4]`** Wong, et. al. (2004)	Codons	**w1** the conserved dN/dS ratio **f1** the fraction of conserved sites among non-positively selected sites **f2** the fraction of neutral sites among non-positively selected sites **posW** the positively selected dN/dS ratio **posP** the fraction of positively selected sites. **posSelection** model selector (1 if positive selection, 0 if not).
`M3` **`M3[HKY,F61]`** (default) **`M3[GTR,MG94,n=4]`** Yang, et. al. (2000)	Codons	**M3.omegas[i]** the conserved dN/dS ratio for category i **M3.ps[i]** the fraction of sites in category i
`M3_Test` **`M3_Test[HKY,F61]`** (default) **`M3_Test[GTR,MG94]`** Yang, et. al. (2000)	Codons	**M3.omegas[i]** the conserved dN/dS ratio for category i **M3.ps[i]** the fraction of sites in category i **posW** the positively selected dN/dS ratio **posP** the fraction of positively selected sites. **posSelection** model selector (1 if positive selection, 0 if not).
`M7` **`M7[4,HKY,F61]`** (default) **`M7[6,GTR,F3x4]`**	Codons	**mu** mean of the Beta(a,b) distribution.

Model	Alphabet	Parameters
Yang, et. al. (2000)		**a** parameter of Beta(a,b) distribution **b** parameter of Beta(a,b) distribution
M8a **M8a[4,HKY,F61]** (default) **M8a[6,GTR,F3x4]** Yang, et. al. (2000)	Codons	**mu** mean of the Beta(a,b) distribution. **a** parameter of Beta(a,b) distribution **b** parameter of Beta(a,b) distribution **posW** the positively selected dN/dS ratio **posP** the fraction of neutral sites.
M8 **M8[4,HKY,F61]** (default) **M8[6,GTR,F3x4]** Yang, et. al. (2000)	Codons	**mu** mean of the Beta(a,b) distribution. **a** parameter of Beta(a,b) distribution **b** parameter of Beta(a,b) distribution **posW** the positively selected dN/dS ratio **posP** the fraction of positively selected sites.
M8a_Test **M8a_Test[4,HKY,F61]** (default) **M8a_Test[6,GTR,F3x4]** Yang, et. al. (2000)	Codons	**mu** mean of the Beta(a,b) distribution. **a** parameter of Beta(a,b) distribution **b** parameter of Beta(a,b) distribution **posW** the positively selected dN/dS ratio **posP** the fraction of positively selected sites.

Model	Alphabet	Parameters
		posSelection model selector (1 if positive selection, 0 if not).
`branch-site` **`branch-site[HKY,F61]`** (default) **`branch-site[GTR,F3x4]`** Zhang, et. al. (2005)	Codons	**fs[1]** the fraction of conserved sites on background branches. **fs[2]** the fraction of neutral sites on background branches. **omegas[1]** the conserved dN/dS ratio. **posW** the positively selected dN/dS ratio. **posP** the fraction of sites (neutral or conserved) that switch to positive selection on foreground branches. **posSelection** model selector (1 if positive selection, 0 if not).

7.5. The branch-site substitution model

In order to use the branch-site substitution model, the user needs to

- Specify an unrooted tree topology in Newick format.
- Label foreground branches on the tree using NHX attributes.
- Disable topology changes in order to fix the tree topology (branch lengths are estimated).

Here is an example tree file:

Example 4. An initial tree file with branch lengths

(((A1:0.1, B1:0.1):0.1,(C1:0.1, D1:0.1):0.1):0.1,((E1:0.1, F1:0.1):0.1,(G1:0.1, H1:0.1):0.1):0.1,(((A2:0.1, B2:0.1):0.1, (C2:0.1, D2:0.1):0.1):0.1,((E2:0.1, F2:0.1):0.1,(G2:0.1, H2:0.1):0.1):0.1):[&&NHX:foreground=1]0.2);

Any branch lengths provided will be used as initial values in the MCMC analysis. However, it is not necessary to provide them:

Example 5. An initial tree file without branch lengths

(((A1, B1),(C1, D1)),((E1, F1),(G1, H1)),(((A2, B2),(C2, D2)),((E2, F2),(G2, H2))):[&&NHX:foreground=1]);

The NHX attribute must be applied to the branch, not the node. Therefore it must occur after a colon. Multiple branches may be marked as foreground branches.

An example command line is as follows:

```
% bali-phy alignment.fasta --smodel branch-site[HKY,F3x4] --disable=topology --tree=tree.tree
```

The posterior probability of positive selection is the posterior mean of the posSelection parameter. This may be computed using the statreport program with the **--mean** option.

In case this probability is extremely close to 1 or 0, you may wish to add the option **--Rao-Blackwellize S1.BranchSiteTest.posSelection**. This will report the log-probability of positive selection each iteration. The user may exponentiate the reported values and then average them (using R, for example) in order to compute a more accurate estimate of the posterior probability of positive selection.

7.6. Substitution model examples

Example: --smodel WAG+F+Rates.logNormal+INV

Example: --smodel WAG+Rates.logNormal+INV (same as above)

Example: --smodel LG+gwF+Rates.logNormal+INV

Example: --smodel EQU --alphabet Triplets

Example: --smodel HKY

Example: --smodel M0

Example: --smodel M0+F1x4

Example: --smodel M2a

Example: --smodel M2a[HKY] (same as above)

7.7. Genetic Codes

When using a codon-based substitution model like **M0**, you may select the genetic code by specifying **--alphabet Codons[,*genetic-code*]**. Available genetic codes are **standard**, **mt-vert**, **mt-invert**, **mt-yeast**, **mt-protozoan**.

If the genetic code is not specified, then the standard code is used:

```
% bali-phy sequence-file --smodel M0 --alphabet Codons
% bali-phy sequence-file --smodel M0 --alphabet Codons[RNA]
```

These examples specify the vertebrate mitochondrial code:

```
% bali-phy sequence-file --smodel M0 --alphabet Codons[DNA,mt-vert]
% bali-phy sequence-file --smodel M0 --alphabet Codons[,mt-vert]
```

8. Insertion/deletion models

The current models are RS05, RS07, and **none**. The default is RS07. Each of these models is a probability distribution on pairwise alignments. The probability distribution on multiple sequence alignments is constructed by factoring the multiple sequence alignment into pairwise alignments along each branch of the tree, as described in Redelings and Suchard (2005).

Table 4. Substitution Models

Model	Parameters	Description
RS05 Redelings and Suchard (2005)	: the gap-opening probability : the gap-extension probability	Gap lengths are geometrically distributed with extension probability .

Model	Parameters	Description
		This indel model is independent of the branch length connecting the ancestor and descent sequences.
RS07 Redelings and Suchard (2007)	: the insertion and deletion rate : the gap-extension probability	Gap lengths are geometrically distributed with extension probability . The probability of an indel event depends on the branch length in this model.
none		Indicates the lack of a model.

Specifying an indel model of **none** for a given partition results in fixing the alignment for that partition to its initial value, and ignoring information in shared insertions or deletions.

9. Alignment constraints

To fix specific columns of the alignment, you may specify alignment constraints in a file as follows:

1. Use the argument **`--align-constraint`** ***`filename`***
2. For multiple partitions, list multiple filenames separated by colons. If a partition doesn't have a constraint, then use an empty filename. For example, to specify constraints for partitions 1 and 3, write: **`--align-constraint`** ***`filename1`*****`::`*****`filename3`***
3. Each filename refers to a file in which each line represents a constraint.

9.1. Syntax

The first line of the file is a header consisting of an ordered list of sequence names separated by spaces. Each subsequent line consists of a space-separated list of sequence positions, with the first position corresponding to the first leaf sequence, the second position corresponding to the second leaf sequence, etc. Thus, if there are n leaf taxa, then each line corresponds to a space-separated list of n integers.

9.2. Examples

For example, the file

```
A B C
1 2 2
```

implies that position 1 of leaf sequence A is aligned to position 2 of leaf sequences B and C. Note that the first position in a sequence is position 0.

Optionally, one may use a '-' instead of an integer, which denotes a lack of constraint for that sequence. This can be useful as follows:

```
A B C D
2 2 - -
- - 2 2
```

The above constraints force alignment between position 2 of sequences A and B, and between position 2 of sequence C and D.

9.3. Computing the constraints

The program **alignment-indices** may be used to aid in computing a constraint file from an input alignment. See Q: 13.7.3 [31].

10. Convergence and Mixing: Is it done yet?

When using Markov chain Monte Carlo (MCMC) programs like *MrBayes*, *BEAST* or *BAli-Phy*, it is hard to determine in advance how many iterations are required to give a good estimate. The number depends on the specific data set that is being examined. As a result, *BAli-Phy* relies on the user to analyze the output of a running chain periodically in order to determine when enough samples have been obtained. This section describes a number of techniques to diagnose when more samples must be taken.

Some of the better diagnostics for lack of convergence rely on running at least 4 independent copies of the Markov chain (preferably 10) from different random starting points to see if the sampled posterior distributions for each chain are the same. Unfortunately, when the distributions all seem to be this same, this doesn't *prove* that they have all converged to the equilibrium distribution. However, if the distributions are different then you can reject either convergence or good mixing.

10.1. Definition of Convergence

Convergence refers to the the tendency of a Markov chain to to "forget" its starting value and become typical of its equilibrium distribution. Note that convergence is a property of the Markov chain itself, not of individual runs of the Markov chain. Ideally a number of individual runs should be examined in order to determine how many initial iterations to discard as "burnin".

10.2. Definition of Mixing

In MCMC, each sample is not fully independent of previous samples. In fact, even after a Markov chain has converged, it can get "stuck" in one part of the parameter space for a long time, before jumping to an equally important part. When this happens, each new sample contributes very little new information, and we need to obtain many more samples to get good precision on our parameter estimates. In such a case, we say that the chain isn't "mixing" well.

10.3. Diagnostics: Variation in split frequencies across runs (ASDSF/MSDSF)

10.3.1. ASDSF and MSDSF

To calculate the ASDSF and MSDSF run:

```
% trees-bootstrap dir-1/C1.trees dir-2/C1.trees ... dir-n/C1.trees > partitions.bs
```

For each split, the SDSF value is just the standard deviation across runs of the Posterior Probabilities for that split. By averaging the resulting SDSF values across splits, we may obtain the ASDSF value (Huelsenbeck and Ronquist 2001). This is commonly considered acceptable if it is < 0.01.

However, it is also useful to consider the maximum of the SDSF values (MSDSF). This represents the range of variation in PP across the runs for the split with the most variation.

10.3.2. Split-frequency comparison plot

To generate the split-frequency comparison plot, you must have R installed. Locate the script `compare-runs.R`. Then run:

```
% trees-bootstrap dir-1/C1.trees dir-2/C1.trees ... dir-n/C1.trees --LOD-table=LOD-table > pa
% R --slave --vanilla --args LOD-table compare-SF.pdf < compare-runs.R
```

Following Beiko et al (2006) [http://dx.doi.org/10.1080/10635150600812544], this displays the variation in estimates of split frequencies across runs. Splits are arranged on the x-axis in increasing order of Posterior Probability (PP), which is obtained by averaging over runs. We then plot a vertical bar from the minimum PP to the maximum PP.

10.4. Diagnostics: Potential Scale Reduction Factors (PSRF)

Potential Scale Reduction Factors check that different runs have similar posterior distributions. Only numerical variables may have a PSRF. To calculate the PSRF for each numerical parameter, you may run:

```
% statreport dir-1/C1.log dir-2/C2.p ... dir-n/C1.log > Report
```

The PSRF is a ratio of the width of the pooled distribution to the average width of each distribution, and should ideally be 1. The PSRF is customarily considered to be small enough if it is less than 1.01.

We compare the PSRF based on the length of 80% credible intervals (Brooks and Gelman 1998) and report the result as PSRF-80%CI. For integer-valued parameters, we avoid excessively large PSRF values by subtracting 1 from the width of the pooled CI.

We also report a new PSRF that is more sensitive for integer distributions. For each individual distribution, we find the 80% credible interval. We divide the probability of that interval (which may be more than 80%) by the probability of the same interval under the pooled distribution. The average of this measure over all distributions gives us a PSRF that we report as PSRF-RCF.

This convergence diagnostic gives a criterion for detecting when a parameter value has stabilized at different values in several independent runs, indicating a lack of convergence. This situation might occur if different runs of the Markov chain were trapped in different modes and failed to adequately mix between modes.

10.5. Diagnostics: Effective sample sizes (ESS)

10.5.1. ESS for numerical values

To calculate the split ESS values, run:

```
% statreport dir-1/C1.log dir-2/C1.log ... dir-n/C1.log > Report
```

We calculate effective sample sizes based on integrated autocorrelation times. This method has the nice property that simply duplicating every sample does not increase the ESS.

The program Tracer [http://evolve.zoo.ox.ac.uk/software/tracer/] also computes ESS values.

10.5.2. ESS for split frequencies

To calculate the split ESS values, run:

```
% trees-bootstrap dir-1/C1.trees dir-2/C1.trees ... dir-n/C1.trees > partitions.bs
```

To compute the ESS for a split, we consider the presence or absence of a split in each iteration as a series of binary values. We compute the integrated autocorrelation time for this binary sequence, which leads to an ESS. This approach is similar to dividing the iterations into blocks and computing the ESS on the PP estimates in the blocks. It is also similar to estimating the variance reduction under a block bootstrap.

10.6. Diagnostics: Stabilization

10.6.1. Stabilization of numerical values

To obtain estimates of the stabilization time for each numerical parameter, you may run:

```
% statreport C1.log > Report
```

Each series of values is counted as having stabilized after the series crosses its upper and then lower 95% confidence bounds twice (if the initial value is below the median) or crosses its lower and then upper confidence bounds twice (if the initial value is above the median). The confidence bounds are those based on its equilibrium distribution as calculated from the last third of the values in the sequence.

10.6.2. Stabilization of tree topologies and tree distances

In addition to examining convergence diagnostics for continuous parameters, it is important to examine convergence diagnostics for the topology as well (Beiko et al 2006 [http://dx.doi.org/10.1080/10635150600812544]). In theory, we recommend the web tool Are We There Yet (AWTY) [http://ceb.csit.fsu.edu/awty/] (Wilgenbush et al, 2004). However, AWTY gives incorrect results if you upload plain NEWICK tree samples -- which is what BAli-Phy outputs. Therefore, if you wish to use AWTY, you must convert the tree samples files to NEXUS before you upload them to AWTY in order to get correct results.

It is also be possible to assess stabilization of tree topologies using tools distributed with *bali-phy* by using commands like the following. Here, sub-sampling and burnin does not apply to the equilibrium tree files. Also, note that you need to manually construct the equilibrium samples, which we recommend to contain at least 500 trees; you might do this by sub-sampling using the *BAli-Phy* tool **sub-sample**.

1. To report the average distances within and between two tree samples:

   ```
   % trees-distances --skip=burnin --subsample=factor compare dir-1/C1.trees dir-2/C1.trees
   ```

2. To compute the distance from each tree in C1.trees to all trees equilibrium.trees, as a time series:

   ```
   % trees-distances --skip=burnin --subsample=factor convergence C1.trees equilibrium.trees
   ```

3. To assess when the above time series stabilizes:

   ```
   % trees-distances --skip=burnin --subsample=factor converged C1.trees equilibrium.trees
   ```

   The stabilization criterion is the same one described above for numerical values.

Note that the running time is the product of the number of trees in the two files. Therefore, comparing two complete tree samples without sub-sampling will take too long.

11. Alignment utilities

Most of these tools will describe their options if given the "**`--help`**" argument on the command line.

11.1. alignment-info

Show basic information about the alignment:

```
% alignment-info file.fasta
% alignment-info file.fasta file.tree
```

11.2. alignment-cat

To select columns from an alignment:

```
% alignment-cat -c1-10,50-100,600- file.fasta > result.fasta
% alignment-cat -c5-250/3 file.fasta > first_codon_position.fasta
% alignment-cat -c6-250/3 file.fasta > second_codon_position.fasta
```

To concatenate two or more alignments:

```
% alignment-cat file1.fasta file2.fasta > all.fasta
```

11.3. alignment-thin

Remove columns without a minimum number of letters:

```
% alignment-thin --min-letters=5 file.fasta > file-thinned.fasta
```

Remove sequences:

```
% alignment-thin --remove=seq1,seq2 file.fasta > file2.fasta
```

Remove short sequences:

```
% alignment-thin --longer-than=250 file.fasta > file-long.fasta
```

Remove sequences while preserving sequence diversity:

```
% alignment-thin --down-to=30 file.fasta > file-30taxa.fasta
% alignment-thin --down-to=30 file.fasta --keep=seq1,seq2 > file-30taxa.fasta
```

Remove sequences that are missing conserved columns:

```
% alignment-thin --remove-crazy=10 file.fasta > file2.fasta
```

11.4. alignment-draw

Draw an alignment to HTML, optionally coloring residues by AU.

```
%  alignment-draw file.fasta --show-ruler --color-scheme=DNA+contrast > file.html
%  alignment-draw file.fasta --show-ruler --AU=file-AU.prob --color-scheme=DNA+contrast+fa
```

11.5. alignment-find

Find the last (or first) FastA alignment in a file.

```
% alignment-find --first < file.fastas > first.fasta
% alignment-find < file.fastas > last.fasta
```

11.6. alignment-indices

Turn columns from a template alignment into alignment constraints:

```
% alignment-indices template.fasta > constraints.txt
% alignment-indices -c100-110,200,300- template.fasta > constraints.txt
```

Each line in this file corresponds to one alignment column.

11.7. alignment-chop-internal

Remove internal-node ancestral sequences from an alignment. (This probably only works for alignments output by bali-phy.)

```
% alignment-chop-internal file.fasta > file-chopped.fasta
```

12. Tree utilities

12.1. trees-consensus

This program analyzes the tree sample contained in *`file`*. It reports the MAP topology, the supported taxa partitions (including partial partitions), and the majority consensus topology.

12.2. trees-bootstrap

Usage: trees-bootstrap *`file1`* [*`file2`* ...] --predicates *`predicate-file`* [OPTIONS]

This program analyzes the tree samples contained in *`file1`*, *`file2`*, etc. It gives the support of each tree sample for each predicate in *`predicate-file`*, and reports a confidence interval based on the block bootstrap.

Each predicate is the intersection of a set of partitions, and is specified as a list of partitions or (multifurcating) trees, one per line. Predicates are separated by blank lines.

12.3. trees-to-SRQ

Usage: trees-to-SRQ *`predicate-file`* [OPTIONS] *`trees-file`*

This program analyzes the tree samples contained in *`trees-file`*. It uses them to produce an SRQ plot for each predicate in *`predicate-file`*. Plots are produced in *gnuplot* format, with one point per line and with plots separated by a blank line.

If **`--mode sum`** is specified, then a "sum" plot is produced instead of an SRQ plot. In this plot, the slope of the curve corresponds to the posterior probability of the event. If the **`--invert`** option is used then the slope of the curve correspond to the probability of the inverse event. This is recommended if the probability of the event is near 1.0, because the sum plot does not distinguish variation in probabilities near 1.0 well.

13. Frequently Asked Questions (FAQ)

13.1. Input files

13.1.1. Does BAli-Phy accept the wildcard characters "N" or "X"? How does it treat them?

Yes, BAli-Phy accepts the wildcard characters "N" (for DNA) and "X" (for proteins). These characters indicate that some letter is present (as opposed to a gap), but that you don't know *which* letter it is.

13.1.2. Does BAli-Phy accept "?" characters?

No. "?" characters are often used to indicate *either* letter presence (e.g. "N", "X") *or* absence (e.g. "-"). BAli-phy will insist that you replace each "?" with either "N"/"X" or "-" to indicate which one you mean.

(Most programs ignore indels and consider only substitutions, and in that case "N" and "-" have the same effect on the likelihood or parsimony score. However, since BAli-Phy takes indels into account, these two alternatives are quite different.)

13.1.3. Does BAli-Phy accept the characters "R" and "Y", etc.?

Yes. BAli-Phy accepts the characters Y, R, W, S, K, M, B, D, H, and V for DNA, RNA, and Codon alphabets. BAli-Phy also accepts the characters B, Z, and J for amino acids. These characters indicate partial knowledge about a letter. For example, R indicates that a nucleotide is present, and is a puRine (A or G). J indicates that an amino acid is present and is either I or L.

(Note that sequences sometimes contain such ambiguity codes because the DNA that was sequenced contains *both* values. This might occur when sequencing a heterozygote or when sequencing pooled DNA from several individ-

uals. However, the model in BAli-Phy (and other phylogeny inference programs) is that only one letter is correct, but we do not know which one it is. This is probably not problematic when dealing with pooled sequences, but should be considered.)

13.2. Running bali-phy.

13.2.1. Can I fix the alignment and ignore indel information, like MrBayes, BEAST, PhyloBayes and other MCMC programs?

Yes. Add **`-Inone`** or **`--imodel=none`** on the command line.

13.2.2. Can I fix the tree topology, while allowing the alignment to vary?

Yes. Add **`--disable=topology --tree=`*`treefile`*** on the command line.

13.2.3. Can I fix the tree topology and *relative* branch lengths, while allowing the alignment to vary?

Yes. Add **`--disable=tree --tree=`*`treefile`*** on the command line.

13.2.4. Can I fix the tree topology and *absolute* branch lengths *in all data partitions*, while allowing the alignment to vary?

Yes. Add **`--disable=tree --tree=`*`treefile`*` --scale=1`** on the command line.

13.3. Run-time error messages

13.3.1. I tried to use **`--smodel LG+Rates.Gamma[6]`** and I got an error message "bali-phy: No match." What gives?

You are probably using the C-shell as your command line shell. It is trying to interpret **`LG+Rates.Gamma[6]`** as an array before running the command, and it is not succeeding. Therefore, it doesn't even run **bali-phy**.

To avoid this, put quotes around the substitution model, like this: **`--smodel "LG+Rates.Gamma[6]"`**. This will keep the C-shell from interfering with your command.

13.4. Stopping bali-phy.

13.4.1. Why is **bali-phy** still running? How long will it take?

It runs until you stop it. Stop it when its done.

13.4.2. How do I stop a **bali-phy** run on my personal computer?

Simply kill the process -- there is no special command to stop **bali-phy**. If you are running it on your personal workstation, then you can use the command **kill**. To do that, you need to find the PID (process ID) of the running program. You can find this by examining the beginning of the file `C1.out`. For example:

```
% less 5d-1/C1.out
command: bali-phy 5d.fasta
start time: Wed Aug  9 13:44:39 2017

VERSION: 3.0-beta2  [type-constraints commit b792c30b+]  (Aug 04 2017 13:40:40)
BUILD: Aug  7 2017 21:01:44
ARCH: x86_64-pc-linux-gnu
COMPILER: GCC 7.1.0
FLAGS: -isystem $(top_srcdir)/boost/include -isystem $(top_srcdir)/include -ffast-mat
directory: /home/bredelings/Work
subdirectory: 5d-257
hostname: telomere
PID: 18838
...
```

Here the PID is 18838. Therefore you can type:

```
% kill 18838
```

On some operating systems you can also type:

```
% killall bali-phy
```

However, be aware that this will terminate *all* of your **bali-phy** runs on that computer.

13.4.3. How do I stop a **bali-phy** run on a computing cluster?

Simply terminate the submitted job. The specific command to terminate a job will depend on the queue manager that is installed on your cluster. Examine the documentation for your cluster, or ask your cluster support staff how to delete running jobs on your cluster.

As an example, if the SGE software is used to submit jobs, then the command **qstat** should list your jobs and their job ID numbers (which is different than the process ID number). You can then use the command **qdel** to delete jobs by ID number. The SGE documentation describes how to use these commands.

13.4.4. So, how can I know when to stop it?

You can stop when it has both converged and also run for long enough to give you >1000 effectively independent samples.

13.4.5. How can I tell when the chain has converged?

See section Section 10, “Convergence and Mixing: Is it done yet?” [24].

13.4.6. How can I check how many iterations the chain has finished?

Run **wc -l C1.log** inside the output directory, and subtract 2.

13.5. Running bp-analyze.pl.

13.5.1. Why does **bp-analyze.pl** say "Program 'draw-tree' not found. Tree pictures will not be generated"?

The program **draw-tree** was not distributed on this platform (Windows, Mac). This is not a fatal error message, it just means that a pretty picture of the tree will not be generated automatically. You can still view the tree with *FigTree*, for example.

13.5.2. Why does **bp-analyze.pl** say "Program 'gnuplot' not found. Trace plots will not be generated"?

This is because you have not installed *gnuplot*. This is not a fatal error message, it just means that pictures of traces, partition support, and SRQ plots will not be generated automatically. You can still view MCMC traces with *Tracer*.

13.5.3. Why does **bp-analyze.pl** say "Program 'R' not found. Some mixing graphs will not be generated"?

This is because you have not installed *R*. This is not a fatal error message, it just means that a plot showing differences in clade probabilities between runs will not be generated.

13.5.4. Why is **bp-analyze.pl** stopping early, or failing to generate some files?

Look in the file `Results/bp-analyze.log`. This should contain the actual commands that were run, along with error message from these commands. These error message should give you a hint as to what the problem might be.

13.6. Interpreting the results.

13.6.1. How do I compute the clade support?

Actually, BAli-Phy uses unrooted trees, so it only estimates bi-partition support. A bi-partition is a division of taxa into two groups, but it does not specify which group contains the root.

13.6.2. How do I compute the split/bi-partition support?

After you analyze the output (Section 6.4, “Summarizing the output - scripted” [14]), the partition support is indicated in `Results/consensus` and in `Results/c50.PP.tree`.

13.7. How do I...

13.7.1. How do I concatenate alignments?

```
% alignment-cat filename1.fasta filename2.fasta > result.fasta
```

The alignments must have the same sequence names, but the names need not be in the same order.

13.7.2. How do I select columns from an alignment?

```
% alignment-cat -c1-10,50-100,600- filename.fasta > result.fasta
```

The resulting alignment will contain the selected columns in the order you specified.

13.7.3. How do I create an alignment-constraint file from an alignment?

To constrain the alignment to match some alignment file *`filename.fasta`* in columns 100, 200-250, and 300, run:

```
% alignment-indices -c100,200-250,300 filename.fasta > filename.constraint
```