PhyloBayes-MPI

A Bayesian software for phylogenetic reconstruction using mixture models

MPI version

Nicolas Lartillot, Nicolas Rodrigue, Daniel Stubbs, Jacques Richer

nicolas.lartillot@umontreal.ca

Version 1.8, January, 2018

Contents

1 Introduction

PhyloBayes-MPI is a Bayesian Markov chain Monte Carlo (MCMC) sampler for phylogenetic inference exploiting a message-passing-interface system for multi-core computing. The program will perform phylogenetic reconstruction using either nucleotide, protein, or codon sequence alignments. Compared to other phylogenetic MCMC samplers, the main distinguishing feature of PhyloBayes is the use of non-parametric methods for modeling among-site variation in nucleotide or amino-acid propensities.

1.1 Modeling site-specific effects using non-parametric methods

Among-site variation in the rate of substitution is often modeled using a gamma distribution. Doing so, however, only models variation in the rate of evolution, and does not account for the fact that the different nucleotides or amino-acids might have uneven propensities across positions.

As a way of modeling state propensity variation along sequences, in PhyloBayes, the rate and also a *profile*—controlling nucleotide or amino acid equilibrium frequencies associated with the substitution process—are modeled as site-specific random variables. As has been shown in several previous works, accounting for such generalized among-site variation results in a better statistical fit and greater phylogenetic accuracy. Of particular interest to phylogenetic reconstruction, such models show a greater robustness to systematic errors such as long-branch attraction problems.

There are two ways site-specific propensities can be modeled. First, one can use a parametric model. In such models, the parametric form of the law describing the distribution of the site-specific feature under investigation is assumed known, up to a few parameters that will be estimated from the data. The best example is the use of a gamma distribution for modeling the distribution of relative rates of substitution across sites (**?**).

Alternatively, one can use non-parametric models: the overall shape of the distribution across sites is not specified a priori but is directly inferred from the data. It is thus more general than the parametric method. There are different non-parametric methods. A practical approach, often adopted in Bayesian inference, is the use of Dirichlet process mixtures (**?**). Formally, these are *infinite mixtures*, i.e. mixtures with a countably infinite number of components.

PhyloBayes uses Dirichlet processes for modeling sites-specific profiles (**?**). Each site is thus given a frequency vector profile over the 20 amino-acids or the 4 bases. These are

combined with a globally defined set of exchange rates, so as to yield site-specific substitution processes. The global exchange rates can be fixed to uniform values (the CAT-Poisson, or more simply CAT, settings), to empirical estimates (e.g. JTT, WAG or LG) or inferred from the data (CAT-GTR settings).

The CAT-like models proposed thus far are phenomenological: they account for among site variation in amino-acid propensities but without any reference to the underlying molecular evolutionary mechanisms. Recently, a mechanistic reformulation of these models has been proposed (**?**). The idea is to work at the codon level and express substitution rates between codons as the product of mutation rates (such as specified by an underlying nucleotide-level mutation process) and fixation probabilities. Mutation rates are assumed to be homogeneous across sites. Fixation probabilities, on the other hand, are site-specific and are essentially determined by selection operating at the amino-acid level. The model assumes that selection acts independently at each site (no epistasis) and is constant across the phylogeny (no fluctuating selection). As a result, the model can be parameterized in terms of site-specific vectors of fitness parameters (for the 20 amino-acids). The distribution of the 20-dimensional fitness profiles across sites is emulated by a Dirichlet process.

1.2 Empirical mixture models

The non-parametric models introduced above are flexible. They automatically estimate the distribution of site-specific effects underlying each dataset. But on the other hand, they may require a relatively large amount of data for reliable estimation.

An alternative that would be suitable for smaller alignments is offered by the so-called empirical mixture models. Unlike non-parametric mixtures, empirical models have a fixed, pre-determined set of components, which have been estimated on a large database of multiple sequence alignments. Classical empirical matrices, such as JTT, WAG and LG (**?**) are a specific case of empirical model, with only one component. A few years ago, empirical mixtures of profiles were proposed (**??**), which are implemented in the current version of PhyloBayes. The user can also specify its own custom set of exchange rates or mixture of profiles.

1.3 Phylogenetic reconstruction

Samples approximately from the posterior distribution are obtained by Markov chain Monte Carlo (MCMC) and are then used to estimate marginal distributions, or expectations, over

the parameters of interest. Concerning the topology of the phylogenetic tree, PhyloBayes works like usual Bayesian phylogenetic reconstruction programs and outputs a majority-rule posterior consensus tree. Conversely, one may be interested in the site-specific biochemical specificities that have been captured by the infinite mixture model, in which case mean posterior site-specific profiles will be estimated from the MCMC output.

1.4 Choosing a model

It is still difficult to get a general idea of the relative merits of the models, as it depends on the dataset and on the criterion used to measure the fit. But a few trends are observed, which can be used for establishing general guidelines.

1.4.1 Amino acid replacement models

Except for small datasets (less than 400 aligned positions), CAT-GTR is virtually always the model with highest fit among all models implemented in PhyloBayes. The CAT-Poisson model, which was the model initially developed in **?**), is less fit than CAT-GTR but generally more fit than single-matrix models on large and datasets, particularly when mutational saturation (multiple substitutions) is prevalent.

Concerning the posterior consensus tree, both CAT-GTR and CAT-Poisson are significantly more robust against long-branch attraction (LBA) artifacts, compared to all other models. However, in some cases, there are differences between the topologies obtained under CAT or CAT-GTR. The flat exchange rates of CAT (which were introduced mostly for the sake of computational efficiency) are not biologically realistic, suggesting that CAT-GTR would generally be a better option.

Empirical mixture models (C60 or WLSR5) may be appropriate for single gene datasets.

1.4.2 Nucleotide substitution models

The CAT-GTR model is a very good generic model also for DNA or RNA data, again probably better than CAT (which does not correctly handle differences in transition and transversion rates) but also, in most cases, better than classical nucleotide substitution models.

1.4.3 Codon substitution models

The mutation-selection model described in **?**) is implemented in the present version, and can be used for estimating distributions of selection coefficients and for phylogenetic inference. Other applications of this model, and other codon models implemented in this version, will be discussed at greater length in future works.

1.4.4 Rate across sites

For the amino acid and nucleotide models, the present version only implements the discretized gamma distribution of rates across sites.

1.4.5 Dataset size

In practice, the Dirichlet process mixture works best for alignments in the range of 1 000 to 20 000 aligned positions. Beyond 20 000 positions, convergence and mixing of the Monte Carlo progressively become challenging. As for the number of taxa, the MCMC sampler seems able to deal alignments with up to 100 taxa reasonably well and has already been used on datasets with up to 250 taxa.

A possible approach for analyzing very large multi-gene datasets (large in terms of the number of aligned positions) is to perform *jackknife* resampling procedures (although resampling procedures are not so easily justified from a Bayesian philosophical standpoint.). An example of gene-jackknife using PhyloBayes is described in **?**).

1.5 Detailed prior and model specification

The exact mathematical structure of the model and the algorithms are described in a separate document (pbmpi_suppmat) available on the website (www.phylobayes.org).

2 Input data format

2.1 Sequences

The main format recognized by PhyloBayes is a generalization of the PHYLIP format:

```
<number_of_taxa> <number_of_sites>
taxon1 sequence1...
taxon2 sequence2...
...
```

Taxon names may contain more than 10 characters. Sequences can be interrupted by space and tab, but not by return characters. They can be interleaved, in which case the taxon names may or may not be repeated in each block.

PhyloBayes automatically recognizes DNA, RNA, or protein alphabets. The following characters will all be considered equivalent to missing data: "-", "?", "$", ".", "*", "X", "x", as well as the degenerate bases of nucleic acid sequences ("B", "D", "H", "K", "M", "N", "R", "S", "V", "W", "Y"), and the "B" and "Z" characters for protein sequences. Upper or lower case sequences are both recognized, but the case matters for taxon names.

For running analyses under mutation-selection models (**?**), the alignment should be a protein coding nucleotide alignment.

PhyloBayes can also be applied to datasets with arbitrary alphabets. The file should then be formatted as follows:

```
#SPECIALALPHABET
<number_of_taxa> <number_of_sites> <ALPHABET>
taxon1 sequence1...
taxon2 sequence2...
...
```

where the alphabet is specified by concatenating all characters, in one single word, e.g.:

```
#SPECIALALPHABET
2 4 ABCDE
taxon1 ACABEDE
taxon2 ACEBBDE
```

2.2 Trees

An initial tree can be provided or, alternatively, the program can be constrained to sample the posterior distribution of parameters under a specified tree topology, which will remained

fixed throughout the analysis. Trees should be provided in NEWICK format. Branch lengths can be specified but will be ignored.

Taxon names should correspond to the names specified in the data matrix (case sensitive). If some names are present in the tree but not in the matrix, the corresponding taxa will be pruned out of the tree. That is, the spanning subtree containing all the taxa mentioned in the data matrix will be considered as the input tree. Conversely, if some taxa are present in the data matrix but not in the input tree, the program will exit with an error message.

3 General presentation of the programs

First, a quick note on system requirements: PhyloBayes MPI is provided both as executable files (for linux x86-64) and as a C++ source code. Depending on the operating system running on your cluster, you may need to recompile the code. To this end, a simple Makefile is provided in the sources directory, and compiling with the make command should then work in most simple situations, assuming that a version of MPI compatible with C++ compilation (with a mpicxx/mpic++ executable) is already installed on your cluster. Note that, in some machines, OpenMPI needs to be specifically uploaded by the user before compiling and/or running a MPI program. You may need to check all these details with your system administrator. The present code can run in principle on MacOSX or Windows operating systems, however, it is primarily intended for (and has been exclusively tested on) high performance computing facilities operating under linux or Unix.

The following programs can be found in the package (for details about any of these programs, type the name without arguments):

- `pb_mpi` : the MCMC sampler.
- `readpb_mpi` : post-analysis program, for computing a variety of posterior averages.
- `bpcomp` : evaluates the discrepancy of bipartition frequencies between two or more independent runs and computes a consensus by pooling the trees of all the runs being compared.
- `tracecomp` : evaluates the discrepancy between two or more independent runs based on the summary variables provided in the trace files. `bpcomp` and `tracecomp` were directly obtained from the non-mpi version of phylobayes (thus, if some recompiling is needed, these 2 programs should be recompiled from the non-mpi phylobayes suite).

In this section, a rapid tour of the programs is proposed. A more detailed explanation of all available options for each program is given in the next section.

3.1 Running a chain (`pb_mpi`)

A run of the `pb_mpi` program will produce a series of points drawn from the posterior distribution over the parameters of the model. Each point defines a detailed model configuration

(tree topology, branch lengths, nucleotide or amino-acid profiles of the mixture, etc.). The series of points defines a chain.

To run the program:

```
mpirun -np <n> pb_mpi -d <dataset> <chainname>
```

Here, `<n>` is the number of processes running in parallel. You could also use `mpiexec` instead of `mpirun`. You cannot run `pb_mpi` with less than 2 processes (the parallelization scheme involves a master and $n - 1$ slaves.)

Most clusters use Sun Grid Engine. In that case, you might need to write a script that would probably look like the following:

```
#!/bin/bash
#
#PBS -l walltime=120:00:00
#PBS -l nodes=1:ppn=8
#PBS -o out
#PBS -e err
#PBS -j oe
#PBS -W umask=022
#PBS -r n
mpirun -n 8 pb_mpi -d datafile -cat -gtr chainname
```

and then send the script to the queue using `qsub`.

As for choosing the right parallelization scheme, the most straightforward approach is to set n equal to the number of cores of a given node. On the other hand, for large datasets, and if your cluster has efficient communication between nodes (e.g., Infiniband), then parallelizing over more than one node could be efficient. Thus, for instance, if you have 8 cores per node, then you could run with $n = 16$ (in which case `nodes=2:ppn=8`) or even $n = 32$ (`nodes=4:ppn=8`). You can try several degrees of parallelization and look at the `trace` file: the first and second columns give the total time spent by the chain thus far (in seconds), and the time per saved point, thus allowing you to compare the efficiency of various parallelization schemes. Finally, the third column gives the percentage of time spent in topological updates (as opposed to updates of the mixture or of other continuous parameters). As a general rule, this percentage decreases as the degree of parallelization increases (this is because topological updates are the most efficiently parallelized part of the program). If this percentage is more than 50%, this generally means that a further increase in the degree of parallelization should normally result in close to linear gains (twice as many cores, twice as fast the program will

run). In contrast, if the percentage of time spent in topological updates is 40% or less, then parallelization gains start to be less than linear, so that a further increase in the degree of parallelization, although still resulting in a faster run, is globally a waste of computational resources.

The `-d` option is for specifying the dataset. There are many other options for specifying the model (see below). The default options are the CAT-GTR model with discrete gamma (4 categories). Before starting, the chain will output a summary of the settings.

A series of files will be produced with a variety of extensions. The most important are:

- `<name>.treelist`: list of sampled trees;
- `<name>.trace`: the trace file, containing a few relevant summary statistics (log-likelihood, total length of the tree, number of components in the mixture, etc).
- `<name>.chain`: this file contains the detailed parameter configurations visited during the run and is used by `readpb_mpi` for computing posterior averages.

The chains will run as long as allowed. They can be interrupted at any time and then restarted, in which case they will resume from the last check-point (last point saved before the interruption). To soft-stop a chain, just open the `<name>.run` file and replace the 1 by a 0. Under linux, this can be done with the simple following command:

```
echo 0 > <chainname>.run
```

The chain will finish the current cycle before exiting. To restart an already existing chain:

```
mpirun -np <n> pb_mpi <chainname>
```

Be careful not to restart an already running chain. You can stop a chain and restart it under a different degree of parallelization.

3.2 Checking convergence and mixing (`bpcomp` and `tracecomp`)

It is difficult to know how long a chain should run beforehand. Different datasets, or different models, may not require the same number of cycles before reaching convergence and may display very different mixing behaviors. In general, for larger datasets, each cycle will take more time, but also, more cycles will be needed before reaching convergence. Note also that the absolute number of cycles is not really a relevant measure of the quality of the resulting sample: update mechanisms for the mixture, the topology or the hyperparameters are not

really comparable, and their mixing efficiency depends very much on the model, the data and the implementation. In the case of phylobayes, the MCMC sampler saves one point after each cycle. A cycle itself is made of a set of complex and integrated series of updates of the topology, the branch length or the substitution model (including the mixture), which are not easily compared with the number of generations realized by other phylogenetic samplers. Generally, a run under phylobayes provides good results for a total number of points of the order of 10 000 to 30 000, although again, this really depends on the datasets.

The best is therefore to rely on more objective measures, such as effective sample size and reproducibility of the results across independent runs started from different initial conditions.

In the case of the relatively complex infinite mixture models CAT and CAT-GTR, convergence and mixing should be carefully assessed both for the phylogenetic and for the mixture aspects of the model. Thus, one should make sure that posterior consensus trees are reproducible across independent runs, but also, that the trace plots of the summary statistics recorded in the trace file capturing various sub-components of the model (tree length, alpha parameter, number of occupied components of the infinite mixture, entropy of the mixture, entropy of exchangeabilities) appear to be at stationarity and to be reproducible across runs.

Convergence can first be visually assessed by plotting the summary statistics recorded in the trace file as a function of number of iterations. This can be done using simple linux utilities, such as `gnuplot`. Alternatively, the trace file of phylobayes is compatible with the `Tracer` program of the Beast software. Thus, you can use `Tracer` to check convergence and estimate effective sample size (`tracecomp`, introduced below, does similar things, albeit with a more primitive interface).

It is also good practice to run at least two chains in parallel and compare the samples obtained under these several independent runs. The can be done using the `tracecomp` program (for checking convergence of the continuous parameters of the model) and the `bpcomp` program (for assessing convergence in tree space). Both use a similar syntax:

```
bpcomp -x 1000 10 <chain1> <chain2>
```

Here, using a burn-in of 1000, and sub-sampling every 10 trees, the `bpcomp` program will output the largest (`maxdiff`) and mean (`meandiff`) discrepancy observed across all bipartitions. It will also produce a file (`bpcomp.con.tre`) with the consensus obtained by pooling all the trees of the chains given as arguments.

Some guidelines:

- maxdiff < 0.1: good run.

- maxdiff < 0.3: acceptable: gives a good qualitative picture of the posterior consensus.
- $0.3 <$ maxdiff < 1: the sample is not yet sufficiently large, and the chains have not converged, but this is on the right track.
- if maxdiff $= 1$ even after 10,000 points, this indicates that at least one of the runs is stuck in a local maximum.

Similarly,

```
tracecomp -x 1000  <chain1> <chain2>
```

will produce an output summarizing the discrepancies and the effective sizes estimated for each column of the trace file. The discrepancy d is defined as $d = 2|\mu_1 - \mu_2|/(\sigma_1 + \sigma_2)$, where μ_i is the mean and σ_i the standard deviation associated with a particular column and i runs over the chains. The effective size is evaluated using the method of **?**). The guidelines are:

- maxdiff < 0.1 and minimum effective size > 300: good run;
- maxdiff < 0.3 and minimum effective size > 50: acceptable run.

3.3 Obtaining posterior consensus trees and parameter estimates (`readppb_mpi`)

The consensus of all trees sampled at equilibrium by the MCMC sampler (which is a MCMC estimate of the posterior consensus tree), is usually taken as the point estimate of the phylogenetic tree. Such a (majority-rule) consensus trees is automatically produced by the `bpcomp` program (see above). Note that `bpcomp` can be run on a single chain (in which case it will simply produce the consensus of all trees after burn-in), and not necessarily on multiple chains (in which case, as explained above, it will make the consensus of all trees pooled across all chains, and compute a discrepancy measure across chains). Using bpcomp on multiple chains usually results in more stable MCMC estimates of the posterior consensus tree.

The `readpb_mpi` program is meant for estimating some key parameters of the model, for performing posterior predictive analyses and for computing posterior mean likelihoods and cross-validation scores.

By default, `readpb_mpi` only computes a simple estimate (mean and 95 % credibility interval) for the total length of the tree (total number of substitutions per site across the entire phylogeny) and for the α parameter of the discrete gamma distribution of rates across

sites. All other tasks performed by `readpb_mpi` are accessible via specific options (see detailed options of `readpb_mpi` in the next section).

4 Posterior predictive checks

Checking for model adequacy is a particularly important step in Bayesian inference. An adequate model should perform well in predicting the patterns that are typically observed in real data. In the present context, the model should correctly predict those patterns which are suspected to be of particular relevance for phylogenetic reconstruction. Thus for instance, if we suspect that site-specific restrictions in acceptable nucleotides or amino-acids, or compositional variation among species, are important factors potentially resulting in systematic errors in tree reconstruction if not properly modelled, then it is particularly important to check that our models correctly predict the typical patterns of variation of composition across sites and across taxa.

In Bayesian inference, this type of model checking is done using posterior predictive simulations. Posterior predictive checks can be seen as the Bayesian analogue of the parametric bootstrap: once the model has been conditioned on empirical data, the parameter configuration thus estimated is used to re-simulate data (many times). Then, the value of some summary statistic of interest (meant to capture the key patterns which the model should correctly capture and reproduce) is computed on the simulated replicates, thus yielding a null distribution for the statistic under the model. The value of the statistic computed on the original data is then compared to this null distribution. Typically, a p-value is computed, defined as the fraction of simulated replicates for which the value for the test statistic is at least as extreme as the observed value. As a good complement to the posterior predictive p-value, a z-score can also be considered: the deviation between the observed value and the mean of the null distribution, relative to the standard deviation of this null distribution. This z-score is useful in those cases where the MCMC estimate of the p-value is identically 0. Apart from model checking, posterior predictive simulations can also be considered as a principled approach to do simulations that are correctly calibrated against empirical data.

Posterior predictive simulations and checks can be done easily based on the output of a MCMC sampler – it is just a matter of reading a chain and, for each of a series of points sampled at stationarity, re-simulate new data based on the corresponding parameter configuration. In the present context, however, posterior predictive checks raise specific issues: how to deal with missing data and with site-specific random variables.

Missing data – Many data matrices, in particular in phylogenomics, are characterized by a sometimes fairly large fraction of missing entries. In addition, the distribution of those missing entries is most often highly non-uniform across the data matrix. Missing entries

can create systematic biases in the posterior predictive null distribution, if not properly accounted for. In particular, if missing data are simply ignored at the simulation step (such that all simulation replicates have 0% missing data), then some of the statistics (e.g. the mean diversity across sites) will be systematically higher across simulations than on the original empirical data, merely because of a differential amount of missing information. In PhyloBayes, the method for controlling for the potential impact of missing data is to first simulate a replicate and then mask all entries that were missing in the original data matrix. This way, the fraction and non-uniform pattern of missing entries is preserved in the null distribution of the test.

Site-specific random variables – In the presence of random effects across observations (here across sites), the posterior predictive formalism raises some subtelties: random effects across sites can be either explicited summed over (as is classically done with the discretized gamma distribution of rates across sites), or explicitly sampled during the MCMC (as is done by PhyloBayes in the case of mixture models of site-specific amino-acid or nucleotide equilibrium frequency profiles). In this context, if we use a random parameter configuration at equiibrium of the MCMC, then which site-specific rate and which site-specific profile should we use to simulate under the posterior-predictive distribution for that site? The question is not totally trivial. At first sight, the most natural procedure would be to use the site-specific profile such as specified by the current parameter configuration for that site. As for the site-specific rate, one would choose a rate category uniformly at random for each site. One would then simulate the column pattern for that site under those site-specific rate and profile (this is what is done by most phylogenetic software programs).

However, technically, this would mean that we are drawing the site-specific rate from the conditional *prior* rate distribution (the gamma distribution with the current value of the alpha parameter), whereas the site-specific profile is effectively drawn from the conditional *posterior* profile distribution. This can be seen from the fact that the information contained by the particular column pattern being present at that site in the original empirical data is not used to choose the rate, but has been used to define the profile that is eventually chosen to conduct the simulation. Yet, both are site-specific random effects, so why should we behave differently?

One possible approach is to draw both of them, rate and profile, from their respective *conditional prior* distributions. In that case, one would re-draw the component of the profile mixture to which the focal site is allocated, based on the relative weights of the mixture. The site-specific rate is from the conditional prior if it is drawn uniformly at random for

each site.

An alternative is to draw both of them, rate and profile, from their respective *conditional posterior* distributions. To do so, one would use the current profile at the focal site, such as specified by the current parameter configuration. As for the site-specific rate, one should draw it, not uniformly at random, but proportionally to the rate-specific likelihoods at that site.

There are arguments in favor of both of them. The conditional prior is closer in spirit to what would be done in a frequentist context (using the parametric bootstrap). On the other hand, if there are correlations between patterns of missing data and site-specific rates or profiles, then these correlations will be lost in the simulated data. Yet, it could be useful to control the posterior predictive null distribution for such correlations.

As far as I can tell, most phylogenetic software draw site-specific rates from the conditional prior. The default option in PhyloBayes, in contrast, is to simulate under the conditional posterior, for both rates and profiles. In earlier versions, the nucleotide or amino-acid state at the root was also drawn from the conditional posterior (thus informed by the original data). However, this procedure is potentially problematic in a context where the tree topology is not specified a priori. In the current version (as of 1.7), the root state is drawn from the equilibrium distribution of the substitution process at the focal site. Thus, in summary, conditional posterior for rate and for profile, and prior for the root, represent the default behavior of PhyloBayes – this behavior can be controlled through specific options (see detailed options, below).

5 Detailed options

5.1 pb_mpi

5.1.1 General options

`-d <datafile>`

option for specifying the multiple sequence alignment to be analyzed (see: Input Data Format section).

`-dc`

constant sites are removed. Note that the likelihood will not be properly conditioned on this removal, as it normally should be (as explained by **?**). Using this option is therefore problematic.

`-t <treefile>`

forces the chain to start from the specified tree.

`-T <treefile>`

forces the chain to run under a fixed topology (as specified in the given file). In other words, the chain only samples from the posterior distribution over all other parameters (branch lengths, alpha parameter, etc.), conditional on the specified topology. This should be a bifurcating tree (see Input Data Format section).

`-S`

only saves the trees explored during MCMC in the `treelist` file (and the summary statistics in the `trace` file). Saving only the trees, and not the detailed parameter configurations visited during the MCMC, has the advantage of producing smaller files. This is enough for computing the consensus tree but insufficient for estimating the continuous parameters of the model (e.g. site-specific equilibrium frequency profiles) or for conducting posterior predictive tests or cross-validation analyses. For this, you should save the detailed model configuration for each point visited during the run (which is done by default by the program). Note that this is a different behavior, compared to the old serial version of PhyloBayes (in which the `-s` option was explicitly required to activate the "save all" mode). In the present version, the `-s` option does not do anything.

`-f`

forces the program to overwrite an already existing chain with same name.

`-x <every> [<until>]`

specifies the saving frequency and (optional) the number of points after which the chain should stop. If this number is not specified, the chain runs "forever". By definition, -x 1 corresponds to the default saving frequency. In some cases, samples may be strongly correlated, in which case, if disk space or access is limiting, it would make sense to save points less frequently, say 10 times less often: to do this, you can use the `-x 10` option.

5.1.2 Evolutionary models

5.1.2.1 Rates across sites

`-dgam <n>`

specifies n categories for the discrete gamma distribution. Setting $n = 1$ amounts to a model without across-site variation in substitution rate.

5.1.2.2 Relative exchangeabilities (exchange rates)

`-poisson`

exchange rates are all equal to 1. The model is then a mixture of Poisson (F81) processes.

`-lg, -wag, -jtt, -mtrev, -mtzoa, -mtart`

specifies empirical exchangeabilities.

`-gtr`

specifies a general time reversible matrix: exchangeabilities are free parameters, with prior distribution a product of independent exponential distributions of mean 1.

`-rr <filename>`

exchangeabilities are fixed to the values given in the specified file. The file should be formatted as follows:

```
[<ALPHABET>]
<rr1_2> <rr1_3>   ...    <rr1_20>
<rr2_3> <rr2_4>  ... <rr2_20>
...
<rr18_19> <rr18_20>
<rr19_20>
```

You have to specify the order in which amino acids should be considered on the first line (`[<ALPHABET>]`), with letters separated by spaces or tabs. This header should then be followed by the exchangeabilities in the order specified (spaces, tabs or returns are equivalent: only the order matters).

5.1.2.3 Profile mixture

`-dp (or -cat)`

activates the Dirichlet process.

`-ncat <n>`

specifies a mixture of n components; the number of components is fixed whereas the weights and profiles are treated as random variables. Fixing the number of components of the mixture most often results in a poor mixing of the MCMC. The Dirichlet process usually has a much better mixing behavior.

`-catfix <predef>`

specifies a mixture of a set of pre-defined profiles (the weights are re-estimated). <predef> can be either one of the following keywords: C20, C30, C40, C50, C60, which correspond to empirical profile mixture models (**?**); or WLSR5, which correspond to the model of **?**). Note that this latter model actually defines 4 empirical profiles, which are then combined with a fifth component made of the empirical frequencies of the dataset.

`-catfix <filename>`

specifies a mixture of a set of user-pre-defined profiles, where `<filename>` is the name of a file containing a set of profiles specified as follows:

```
[<ALPHABET>]
<ncat>
<weight> <freq1> <freq2> ... <freq20>
<weight> <freq1> <freq2> ... <freq20>
...
```

where `<ncat>` is the number of profiles, and each line following this number should be a set of 21 real numbers, defining a weight, and then a profile of equilibrium frequencies (separated by spaces or tabs). You should specify the order in which amino acids should be considered on the first line (`[<ALPHABET>]`), with letters separated by spaces or tabs. Note that the weights are there only for historical reasons – they are re-estimated anyway.

5.1.2.4 Combining profiles and exchange rates Any set of exchange rates can be combined with any of the three settings for the mixture. But the *same* set of exchange rates will be used by all components of the mixture.

For instance, -cat -gtr makes an infinite mixture model whose components differ by their equilibrium frequencies but otherwise share the same set of relative exchange rates (themselves considered as free parameters). As another example, -catfix WLSR5 -jtt defines the **?**) model: a model with 5 components, each of which is a matrix made from the relative exchange rates of the JTT matrix, combined with one of the 4 vectors of equilibrium frequencies defined by **?**), plus one vector of empirical frequencies.

The default model is -cat -gtr.

5.1.2.5 Mutation-selection models

`-mutsel`

activates the mutation-selection model as described in **?**) (codon alignments only).

`-mtvert`

specifies the vertebrate mitochondrial code (the universal genetic code is the default).

5.2 bpcomp

`-x <burn-in> [<every> <until>]`

Defines the burn-in, the sub-sampling frequency, and the size of the samples of trees to be taken from the chains under comparison. By default, <burn-in> = 0, <every> = 1 and <until> is equal to the size of the chain. Thus, for instance:

```
-x 1000
```

defines a burn-in of 1000,

```
-x 1000 10
```

a burn-in of 1000, taking one every 10 trees, up to the end of each chain, and

```
-x 1000 10 11000
```

a burn-in of 1000, taking one every 10 trees, up to the 11 000th point of the chains (or less, if the chains are shorter). If the chain is long enough, this implies a sample size of 1000.

```
-o <basename>
```

outputs the results of the comparison in files with the specified basename combined with several extensions:

- `<basename>.bpcomp`: summary of the comparison;
- `<basename>.bplist`: tabulated list of bipartitions (splits) sorted by decreasing discrepancy between the chains;
- `<basename>.con.tre`: consensus tree based on the merged bipartition list.

```
-c <cutoff>
```

tunes the cutoff for the majority rule consensus (posterior probability support under which nodes are collapsed in the final consensus tree). By default, the cutoff is equal to 0.5.

5.3 readpb_mpi

```
-x <burn-in> [<every> <until>]
```

Defines the burn-in, the sub-sampling frequency, and the size of the samples of trees to be taken from the chains under comparison. By default, <burn-in> = 0, <every> = 1 and <until> is equal to the size of the chain (see `bpcomp`).

Note that under the mutation-selection model described in **?**), these sub-sampling options are the only ones currently available with the readpb_mpi command. In this context, the command will produce files that allow one to plot the distribution of scaled selection coefficients, both globally and in a site-specific manner. Under nucleotide and amino acid models, several other options are available, as described below.

`-rr`

computes the mean posterior relative exchangeabilities (only if those are free parameters of the model).

`-ss`

computes the mean posterior site-specific state equilibrium frequencies (only under infinite mixture models).

`-r`

computes the mean posterior rates across sites

`-sitelogl`

computes the site-specific marginal log likelihoods: these likelihoods are summed over all site-specific random variables (rates and profiles).

`-ppred`

for each point of the chain (after burn-in), produces a data replicate simulated from the posterior predictive distribution.

`-div`

performs a posterior predictive diversity test: the test statistic is the mean diversity per site (mean number of distinct amino-acid per sites); the observed value of this statistic is computed on the true data, and compared with its null (posterior predictive) distribution (see **?**)

`-comp`

performs a posterior predictive test of compositional homogeneity: the test statistic is the maximum square deviation between global and taxon-specific empirical frequencies; the observed value of this statistic is computed on the true data, and compared with its null (posterior predictive) distribution (see **?**)

`-ppredrate [prior / posterior]`

in the case where among site rate variation is modelled, this option will specify whether the site-specific rate should be drawn from the conditional prior or the conditional posterior distribution (see above, section 4, Posterior predictive checks).

`-ppredprofile [prior / posterior]`

in the case of profile mixture models (such as cat or cat-gtr), this option will specify whether the site-specific profile should be drawn from the conditional prior or the conditional posterior distribution (see above, section 4, Posterior predictive checks).

`-ppredroot [prior / posterior]`

this option will specify whether the state at the root should be drawn from the conditional prior or the conditional posterior distribution (see above, section 4, Posterior predictive checks).

`-cv [test_dataset]`

computes a posterior mean cross-validation score. If D_1 is the dataset used for running the chain (the training set), D_2 is the dataset specified after `-cv` (the test set) and M is the model under which the chain was run, then what the program outputs is a Monte Carlo estimate of

$$\ln p(D_2 \mid D_1, M) \quad = \quad \int p(D_2 \mid \theta, M)\, p(\theta \mid D_1) d\theta$$

where θ is the set of (global) parameters of the model. The cross-validation likelihood is a measure of how well the model 'predicts' site patterns of D_2 after it has 'learnt' its parameters on D_1. This measure can be computed for alternative models M_i, $i = 1..K$, and models with higher score should in principle be preferred.

Cross validation needs to be replicated (cross-validation scores typically have a large variability, depending on the exact columns that have been included in D_1 and D_2). If your original data set is D, then you should produce random pairs D_1 and D_2, by randomly sampling columns of D (without replacement), running `pb_mpi` separately on each replicate of D_1, and then running `readpb_mpi` with the `-cv` option (followed by the name of the corresponding D_2 test set) on each resulting chain (and all this for each model M_i).

The cross-validation score can then be averaged over the replicates for a given model. Then, supposing that a model M_1 has a higher average score than M_2, the number of replicates for which the score of M_1 is indeed higher than the score of M_2 can be considered as a measure of the 'significance' of this preference for M_1 over M_2.

Typically, 10-fold cross-validation (such that D_2 represents 10% and D_1 90% of the original dataset) has been used (e.g. `?`), and ten replicates have been run (although ideally, 100 replicates would certainly be more adequate). However, alternative schemes are possible. In particular, for faster computation in the case of very large datasets, cross-validation schemes in which the size of D_1 and D_2 combined together is smaller than the size of D could be useful (as long as D_1 is large enough for the parameters to be correctly estimated).