PhyloBayes-MPI

A Bayesian software for phylogenetic reconstruction using mixture models

MPI version

Nicolas Lartillot, Nicolas Rodrigue, Daniel Stubbs, Jacques Richer

nicolas.lartillot@univ-lyon1.fr

Version 1.9, April, 2022

Contents

1 Introduction **3**

1.1 Modeling pattern-heterogeneity across sites 3

1.2 Phylogenetic reconstruction 4

1.3 Choosing a model 4

1.4 Detailed model specification 5

2 Input data format **6**

2.1 Sequences 6

2.2 Trees 6

3 General presentation of the programs **8**

3.1 Running a chain (`pb_mpi`) 8

3.2 Checking convergence and mixing (`bpcomp` and `tracecomp`) 10

3.3 Obtaining posterior consensus trees and parameter estimates 12

4 Model comparison and selection **13**

4.1 Leave-one-out cross-validation and the wAIC 14

4.2 Marginal likelihoods and Bayes factors 16

5 Posterior predictive model checking **18**

5.1 Practical posterior predictive checks 18

5.2 Some issues about site-specific variables and missing data 20

6 Detailed options **22**

6.1 pb_mpi 22

6.2 bpcomp 25

6.3 readpb_mpi 26

7 Benchmark **30**

7.1 Inference of the tree tree topology 30

7.2 Posterior predictive checks 30

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 based on Dirichlet processes (infinite mixture models) for modeling among-site variation in nucleotide or amino-acid propensities.

1.1 Modeling pattern-heterogeneity across sites

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 (rate-heterogeneity across sites), and does not account for the fact that different sites might show, not just different rates, but also different preferences for nucleotides or amino-acids (pattern-heterogeneity across sites).

As a way of modeling pattern-heterogeneity, 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. Accounting for such generalized among-site variation results in a better statistical fit and a greater phylogenetic accuracy – in particular, a greater robustness to systematic errors such as long-branch attraction.

PhyloBayes uses Dirichlet processes (which are infinite mixtures) for modeling sites-specific profiles (Lartillot and Philippe, 2004). Each site draws a frequency vector profile over the 20 amino-acids or the 4 bases from the mixture. 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). A codon-version of the CAT model, called mutation-selection model (Rodrigue et al., 2010) is also implemented in the current version of PhyloBayes.

An alternative to the non-parametric models introduced above, which is less flexible and less accurate but computationally more efficient, 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. Over the years, several empirical mixtures of profiles have been proposed (e.g.

Quang et al., 2008; Wang et al., 2008). Some of them are implemented in the current version of PhyloBayes. The user can also specify its own custom set of exchange rates or its own mixture of profiles.

1.2 Phylogenetic reconstruction

Samples approximately from the posterior distribution are obtained by Markov chain Monte Carlo (MCMC), using the `pb_mpi` command, and are then used to estimate expectations over the parameters of interest (using `readpb_mpi` and `bpcomp`). Concerning the tree topology, PhyloBayes outputs, in a separate file, the list of trees (topology + branch-lengths) sampled during the MCMC. This tree list can then be processed using a post-analysis program given in the package (`bpcomp`) to get a majority-rule posterior consensus tree. Alternatively, one may be interested in other aspects of the model, such as the site-specific biochemical specificities that have been captured by the infinite mixture model, or the site-specific rates of substitution, or the relative exchange rates across nucleotide or amino-acid pairs, all of which can be estimated from the MCMC output using `readpb_mpi`.

1.3 Choosing a model

The current version of PhyloBayes proposes several approaches for computing the relative fit of alternative models, essentially, by leave-one-out cross-validation (LOO-CV), using a numerical approach previously introduced (Gelfand et al., 1992; Lewis et al., 2014), or using the widely applicable information criterion, or wAIC (Watanabe, 2009; Vehtari et al., 2016). Absolute model fit can also be assessed using posterior predictive checks. In all cases, this is implemented as post-analysis routines using samples from the posterior distribution obtained by first running `pb_mpi` under the model(s) of interest and then running `readpb_mpi` with the relevant options. These approaches to model selection and assessment could in principle be used for each new analysis being conducted, on each new dataset. In practice, however, general trends are observed, which can be used for establishing guidelines as to which model to use by default.

Except for very 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 is less fit than CAT-GTR but generally more fit than single-matrix models for data sets larger than 1000 positions and 30 taxa, particularly when mutational saturation (multiple substitutions) is prevalent. Concerning tree estimation, both CAT-GTR and CAT-

Poisson are significantly more robust against long-branch attraction compared to all other models. This matters particularly for deep phylogenies (e.g. at the scale of metazoans, plants, fungi, eukaryotes, archaea or eubacteria) but is perhaps less important when working over shallower evolutionary scales (e.g. mammals), in which case site-homogeneous models are probably robust enough. Although differences can sometimes be observed between the topologies obtained under CAT or CAT-GTR, in general, these two models will give similar results, although with generally weaker posterior probability supports under CAT-Poisson than under CAT-GTR. Empirical mixture models are generally better than site-homogeneous models but less fit than the CAT models. Empirical mixtures may be appropriate for single gene datasets, although even in that case, CAT-Poisson or CAT-GTR will often show a higher fit.

The CAT-GTR model is a very good generic model also for DNA or RNA data, 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.

1.3.1 Data size

In practice, the Dirichlet process mixture works best for alignments in the range of 1 000 to 50 000 aligned positions. For very large alignments, convergence and mixing of the Monte Carlo can be more challenging. As for the number of taxa, the MCMC sampler seems able to deal with 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. An example of gene-jackknife using PhyloBayes is described in Delsuc et al. (2008).

1.4 Detailed 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 (Rodrigue et al., 2010), 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 as a C++ source code. To compile the code, a 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.) This is the simplest version of the command. Many options are available for modulating the model or the MCMC.

Most clusters propose a batch-queuing system, such as Slurm or Sun Grid Engine. In that case, you will need to write a script containing the command specified above, together with options specifying the requests for time and memory, and then send it to the queue.

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 $n = 16$ or $n = 32$. 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 Geyer (2006). The guidelines are:

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

3.3 Obtaining posterior consensus trees and parameter estimates

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 Model comparison and selection

Several approaches have been proposed for model comparison and selection in the applied Bayesian statistics literature: essentially, marginal likelihoods (Bayes factors), cross-validation and information criteria. All of them are now available in PhyloBayes, although they differ both in terms of their statistical meaning and in terms of the computational investment that they require. For a more detailed introduction, see Lartillot (2022), `https://www.biorxiv.org/content/10.1101/2022.04.22.489153v1`.

Practically, estimating the relative fit of alternative models in PhyloBayes is most easily conducted using leave-one-out cross-validation (LOO-CV) and, for large datasets, the widely applicable information criterion, or wAIC (Watanabe, 2009). For the LOO-CV score, the cross-predictive ordinate approach (Gelfand et al., 1992), first recruited in a phylogenetic context by Lewis et al. (2014), was re-implemented. For the wAIC, the Monte Carlo numerical evaluation is straightforward (Vehtari et al., 2016).

The LOO-CV score and the wAIC can be obtained simultaneously based on a post analysis (using `reapb_mpi` with the `-sitelogl` command, see below), based on two independent runs under the posterior distribution under the model of interest (a pair of runs is needed to get a measure of the Monte Carlo bias and error on these two estimates of model fit). Numerically, both scores behave relatively well: provided that the MCMC has a good convergence and good effective sample size (such as indicated by tracecomp), they both should give reliable estimates. The wAIC gives a very good approximation for large datasets (> 4000 positions). It also tends to be more stable than LOO-CV, and thus requires less computational investment. More details are given in the next subsection.

Of note, the k-fold cross-validation approach that was implemented in the earlier versions of PhyloBayes (v1.8 and before) turns out to be numerically inaccurate (Lartillot, 2022)). In practice, it tends to under-estimate the fit of the more flexible models, such as CAT-Poisson or CAT-GTR. This older implementation of the k-fold approach is still available under the current implementation, although its use is not recommended. As discussed in Lartillot (2022), a site-wise variant of k-fold cross-validation, implemented in the current version (starting with v1.9), is well-behaved numerically, although computationally intensive. Finally, a sequential importance sampling approach is also available for computing the marginal likelihood. and thus the Bayes factors between pairs of models. However, it is computationally impractical for large datasets.

4.1 Leave-one-out cross-validation and the wAIC

In this subsection, the method for computing the leave-one-out cross validation (LOO-CV) score and the widely applicable information criterion (wAIC) are explained on an empirical example. The data are available in the `pbmpi/data/model_fit_examples` folder. The CAT-Poisson model is taken as the model for which to compute the fit, on the dataset `ef2.ali` (elongation factor 2, 627 amino-acid positions for 30 taxa). Here, the fit is computed under a fixed tree topology (`ef2.tree`). Of note, fixing the tree topology is not mandatory for computing the fit, but this will be computationally less expensive.

The first step is to run 2 independent plain mcmc chains under the CAT model (here, saving every 10 cycles, for a total of 11000 cycles, thus 1100 MCMC points saved):

```
mpirun -np 8 pb_mpi -d ef2.ali -T ef2.tree -poisson -dp -dgam 4 -x 10 1100 catef2_plain1
mpirun -np 8 pb_mpi -d ef2.ali -T ef2.tree -poisson -dp -dgam 4 -x 10 1100 catef2_plain2
```

The quality of the runs can be checked visually and then with `tracecomp` (see section 3.2). Here, using a burnin of 100:

```
tracecomp -x 100 catef2_plain?.trace
```

returns:

```
name                effsize rel_diff


loglik              840     0.125997
length              943     0.0569384
alpha               991     0.138605
Nmode               915     0.0921728
statent             819     0.109174
statalpha           870     0.0628273
```

Thus, we have good effective sample sizes and reasonably good relative discrepancies.

The second step is to get the site-specific log likelihood statistics. Here we will again use a burn-in of 100, and a thinning of one every 10 points, thus targeting a total of 100 MCMC points. Thinning is used here because computing site log likelihoods is expensive under the CAT model. We will see below how to get a refined estimate. As for now, the following command:

```
mpirun -np 8 readpb_mpi -x 100 10 -sitelogl catef2_plain1
mpirun -np 8 readpb_mpi -x 100 10 -sitelogl catef2_plain2
```

produces two files: `catef2_plain1.sitelogl` and `catef2_plain2.sitelogl`, containing the site-specific log-likelihood statistics.

The third step is to collect the site-specific scores, compute the mean score (per site), the bias and the error on this estimate, using the script named `read_loocv_waic.py`, which can be found in the `pbmpi/scripts` folder:

```
python3 scripts/read_loocv_waic.py gtref2_plain?.sitelogl
```

In the present case, this should return (up to numerical stochastic error), something like:

```
           debiased score    bias      stdev      CI95min    CI95max      ess     %(ess<10)  f(ess<10)
LOO-CV        -27.6375      0.0346     0.0094   -27.7573   -27.5177    58.3814      0.091      0.200
waic          -27.6574     -0.0138     0.0087   -27.7680   -27.5468    61.0064      0.041      0.086
```

the scores of interest are the debiased LOO-CV and wAIC scores.

Doing the same procedure for the GTR model (running two chains, `gtref2_plain1` and 2, reading the site log likelihoods and running the python script) gives:

```
           debiased score    bias      stdev      CI95min    CI95max      ess     %(ess<10)  f(ess<10)
LOO-CV        -28.3051      0.0053     0.0087   -28.4150   -28.1952    69.6953      0.003      0.007
waic          -28.3016     -0.0028     0.0069   -28.3887   -28.2144    72.9226      0.002      0.003
```

Thus, the CAT model is better fitting than GTR on this dataset, both according to LOO-CV ($\Delta\texttt{cv} = -27.64 + 28.31 = 0.67$) and according to wAIC ($\Delta\texttt{wAIC} = -27.66 + 28.30 = 0.64$). This also indicates that wAIC is already a good approximation of LOO-CV even for datasets of relatively small size (627 aligned positions). Of note, the score is in log units per site of the dataset.

The quality of the estimation can be evaluated based on ESS statistics:

- `%(ess<10)` : propotion of sites for which the effective sample size (ESS) is less than 10 (here, in the case of the CAT model, 9%)
- `f(ess<10)` : fraction of the score itself contributed by those sites (here, for the CAT model, 20%)

These two fractions should be low (ideally, less than 0.1, i.e. less than 10%, although in practice, for models that are very different in their fit, up to 0.3 should still give reasonable estimates of the fit and qualitatively reliable model selection). If there are concerns or if a more accurate/reliable estimate is needed (in particular, when comparing models that have a similar fit), then the site log likelihood statistics should be computed with a larger number of MCMC points, which can be done here by not thinning at step 2, thus using 1000 points:

```
readpb_mpi -x 100 1 -sitelogl catef2_plain1
readpb_mpi -x 100 1 -sitelogl catef2_plain2
```

and then again collecting the results and summarizing with the python script:

```
python3 scripts/read_loocv_waic.py gtref2_plain?.sitelogl
```

which finally returns more accurate estimates than with thinning, also showing good ESS statistics:

```
          debiased score    bias      stdev     CI95min    CI95max   ess     %(ess<10)  f(ess<10)
LOO-CV      -27.6689      0.0107     0.0066   -27.7526   -27.5851   550.4161   0.022   0.055
waic        -27.6800     -0.0016     0.0015   -27.6990   -27.6609   589.8704   0.000   0.000
```

Based on this new estimate for CAT, the scores relative to GTR are thus more accurately estimated as $\Delta \texttt{cv} = 0.64$ and $\Delta \texttt{wAIC} = 0.62$.

4.2 Marginal likelihoods and Bayes factors

For computing the marginal likelihood under a model, the GTR model is taken as an example, on `ef2.ali`, under a fixed tree topology `ef2.tree`. The first step is to run a plain mcmc under the GTR model:

```
mpirun -np 8 pb_mpi -d ef2.ali -T ef2.tree -gtr -ncat 1 -dgam 4 -x 1 1100 gtref2_plain
```

This chain does not need to be of very high quality: it is just to get reasonable posterior mean estimates of the model parameters, using the following command:

```
mpirun -np 8 readpb_mpi -x 100 1 -posthyper gtref2_plain
```

This produces a file, `gtref2_plain.posthyper`, containing the posterior estimates that will be used to make a reference distribution Then, the sequential importance sampling method (sIS) can be called, with 2 independent runs:

```
mpirun -np 8 pb_mpi -f -d ef2.ali -T ef2.tree -gtr -ncat 1 -dgam 4 -self_tuned_sis 1 10 30 0.1 1000
-emp_ref gtref2_plain.posthyper gtref2_sis1
mpirun -np 8 pb_mpi -f -d ef2.ali -T ef2.tree -gtr -ncat 1 -dgam 4 -self_tuned_sis 1 10 30 0.1 1000
-emp_ref gtref2_plain.posthyper gtref2_sis2
```

This will produce two files: `gtref2_sis1.stepping` and `gtref2_sis2.stepping`. Finally, we can collect the site-specific scores, compute mean score, bias and error, using the following python script:

```
python3 scripts/read_marglikelihood.py gtref2_sis?.stepping
```

This should return (up to numerical stochastic error), something like:

```
n     score     debiased     CI95min    CI95max     bias      stdev     mean ess   #ess<10
627   -28.6594   -28.6480   -28.7022   -28.5939    -0.0114     0.0043    31.7643 (0/627)
```

Some words about the options for tuning the sIS run:
`-selftuned_sis 1 10 30 0.1 1000`. The arguments are:

- 1 : one site at a time
- 10 : small burn-in done after a new site has been added
- 30 : min number of cycles (K_{min})
- 0.1 : target variance (v_0)
- 1000: max number of cycles (K_{max})

Thus, for each site: a burnin of 10 cycles is done, followed 30 cycles for computing an estimate of the site log likelihood variance v, which is used to compute the number of cycles $K_i = \min(K_{min} e^{v-v0}, K_{max})$; finally K_i cycles are run, giving K_i log likelihood values that are used to compute the site-specific importance sampling estimate (saved in the `.stepping` file).
`-emp_ref <posthyper file>`: uses the `.posthyper` file given as an argument to compute the reference distribution

5 Posterior predictive model checking

Checking for model adequacy is an important step of Bayesian data analysis. 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. In particular, 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 modeled. Therefore, it is 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 can be 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 (multiple 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 calibrated against empirical data.

5.1 Practical posterior predictive checks

Posterior predictive simulations and checks can be done based on the output of a MCMC sampler – for each of a series of points sampled at stationarity, re-simulate new data based on the corresponding parameter configuration. Here the `ef2.ali` dataset is taken as an example, under a fixed tree topology given in the file `ef2.tree`. Two models will be compared, GTR and CAT-Poisson, for the diversity statistic (which measures the mean number of distinct amino-acids observed at each site).

The first step is to run an mcmc chain under the GTR and the CAT models:

```
pb_mpi -d ef2.ali -T ef2.tree -dp -poisson -x 3 1100 catef2
pb_mpi -d ef2.ali -T ef2.tree -gtr -ncat 1 -x 1 1100 gtref2
```

Then, once the run have completed, we can run the posterior predictive test on these two chains using the following commands:

```
readpb_mpi -x 100 1 -ppred -div catef2
readpb_mpi -x 100 1 -ppred -div gtref2
```

The results will be written into files with the `.div` extension: `catef2.div` and `gtref2.div`. In the present case, this should give, under the GTR model:

```
diversity test
obs div : 3.6571
mean div: 4.46541 +/- 0.0788579
z-score : 10.2503
pp      : 0
```

and under the CAT model:

```
diversity test
obs div : 3.6571
mean div: 3.75094 +/- 0.0579987
z-score : 1.61798
pp      : 0.053
```

which can be interpreted as follows:

- the original empirical sequence alignment has on average 3.66 amino-acids per column;
- alignments simulated under the GTR model, and under parameter values estimated on this EF2 alignment, have on average a higher number amino-acids per column (4.47); the difference with the original dataset is highly significant (z-score of 10.25, pp $\simeq 0$). The absolute difference may look small, but there is an underlying heterogeneity: sites evolving at a higher rate tend to show larger differences between empirical and simulated, compared to more slowly evolving sites.
- in contrast, alignments simulated under the CAT model have on average 3.75 amino-acids per site, which is thus much closer to what is seen on the empirical sequence alignment. The difference between the diversity induced by CAT and the observed diversity is not significant ($pp = 0.053$).

5.2 Some issues about site-specific variables and missing data

In the present context, 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. 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. This default behavior can be overriden by using a specific option (see next section).

Of note, there has been some instability in the implementation of the posterior predictive check procedures in the early versions (prior to June 2016) of the program (for more information, see Benchmark).

6 Detailed options

6.1 pb_mpi

6.1.1 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. Use of this option is therefore problematic and is not recommended.

`-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.

6.1.2 Evolutionary models

6.1.2.1 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.

6.1.2.2 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).

6.1.2.3 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 may result in poor MCMC mixing. 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 (Quang et al., 2008); or WLSR5, which correspond to the model of Wang et al. (2008). 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.

6.1.2.4 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 Wang et al. (2008) 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 Wang et al. (2008), plus one vector of empirical frequencies.

The default model is -cat -gtr.

6.1.2.5 Mutation-selection models

```
-mutsel
```

activates the mutation-selection model as described in Rodrigue et al. (2010) (codon alignments only).

```
-mtvert
```

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

6.2 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.

6.3 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 Rodrigue et al. (2010), 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). Along with the site log-likelihoods, this routine also outputs site-specific quantities which are useful for computing the leave-one-out cross-validation score and the wAIC (see above, section 4, Model comparison and selection).

`-sitecv [test_dataset]`

computes an estimate of the site-wise k-fold cross-validation score using naive importance sampling (Lartillot, 2022). This approach to cross-validation is computationally accurate but less convenient than leave-one-out cross validation or the wAIC (see above, section 4, Model comparison and selection). It requires randomly splitting the original dataset into a training set and a validation set (the latter representing a fraction $f = 1/k$ of the total dataset, with k typically equal to 5 or 10), running a standard MCMC run under the training set, and then running `readpb_mpi`, with this option and giving the name of the validation set as the argument. This procedure should be repeated multiple times (typically, 10 replicates).

`-jointcv [test_dataset]`

this option is deprecated. It corresponds to the `-cv` option of older versions of the program (up to v1.8), which computes an estimate of the joint k-fold cross-validation score using naive importance sampling. This approach, however, it numerically inaccurate (Lartillot, 2022), and its use is not recommended (see above, section 4, Model comparison and selection). It requires randomly splitting the original dataset into a training set and a validation set multiple times, as in the case of the `-sitecv` option.

`-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 Lartillot et al., 2007).

`-sitecomp`

performs a posterior predictive test of compositional heterogeneity across sites (PPA-VAR). The test statistic is the variance in site-specific empirical frequencies across the alignment. That is, for each state (i.e. each of the 4 nucleotides, or the 20 amino-acids, depending on the nature of the alignment), the variance across sites in the frequency at which this state is observed is first computed. Then, this variance is averaged over all possible states (all possible nucleotides or amino-acids). The observed value of this statistic is computed on the true data, and compared with its null (posterior predictive) distribution.

`-siteconvprob`

This posterior predictive diversity test uses as a statistic the mean squared empirical frequency, over all states and across all sites (PPA-CONV). For a given site, this can be interpreted as the mean probability of drawing twice the same state from the equilibrium frequency profile at that site. Hence, averaged over all sites, this statistic is meant to estimate the long-term probability of converging toward the same state in two independent species at a randomly chosen aligned position. The observed value of this statistic is computed on the true data, and compared with its null (posterior predictive) distribution.

`-comp`

performs a posterior predictive test of compositional homogeneity across taxa: 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 Blanquart and Lartillot, 2006)

`-allppred`

performs all posterior predictive tests mentioned above (`-div, -sitecomp, -siteconvprob, -comp`) at once.

`-ppredrate [prior / posterior]`

in the case where among site rate variation is modeled, this option will specify whether the site-specific rate should be drawn from the conditional prior or the conditional posterior distribution. By default, it will be from the conditional posterior (see above, section 5, Posterior predictive checks).

`-ppredprofile [prior / posterior]`

in the case of profile mixture models (such as CAT), this option will specify whether the site-specific profile should be drawn from the conditional prior or the conditional posterior distribution. By default, it will be from the conditional posterior (see above, section 5, Posterior predictive checks).

7 Benchmark

This section presents the results of a benchmark of the program across the history of its successive versions.

7.1 Inference of the tree tree topology

A series of 8 versions, from November 2012 to November 2019 are tested, on two datasets (an alignment from TreeBase, reference M1487, and a dataset made of mitochondrial coding sequences for 42 mammalian species, 3507 aligned positions) and under the CAT-GTR model. Small datasets are considered here, for computational reasons, but also because small alignments tend to give intermediate bipartition posterior probabilities, which are more sensitive to potential bugs in the underlying MCMC algorithmics. Chains were run for 11000 cycles. Bipartition posterior probability estimates were obtained using `bpcomp`, using a burn-in of 1000 cycles.

As can be seen from figures 1 and 2, tree inference appears to be very stable across the history of the code: all versions return very similar posterior distributions over bipartitions. Of note, earlier versions show slightly more dispersed plots against subsequent versions, which reflects their less efficient mixing in tree space.

7.2 Posterior predictive checks

In contrast to tree topology inference, posterior predictive checks have been somewhat problematic until June 2016. Since then, the code has remained essentially unchanged. To illustrate this point, here, a series of 6 versions, from December 2015 to November 2019 are tested, by applying them to three datasets, under the CAT-GTR model. The results are shown in table 1. Two points are noteworthy.

First, there was a bug in the simulation algorithm in the earliest versions, before April 9, 2016. This bug was leading to simulated data that were looking too similar to the true data. As a result, the earliest versions of the program were giving an underestimate of the extent to which the model is rejected by the data (tables, December 09, 2015). Of note, this bug was specific to the MPI version and was not present in the original non-MPI version.

Starting from April 2016 (beginning of version 1.7), the routine for simulation is correct, except that there was still a conceptual problem concerning how the root state was sampled. In the earlier versions, the root state was drawn from the conditional posterior (thus informed by the original data), instead of being drawn from the equilibrium frequencies of the

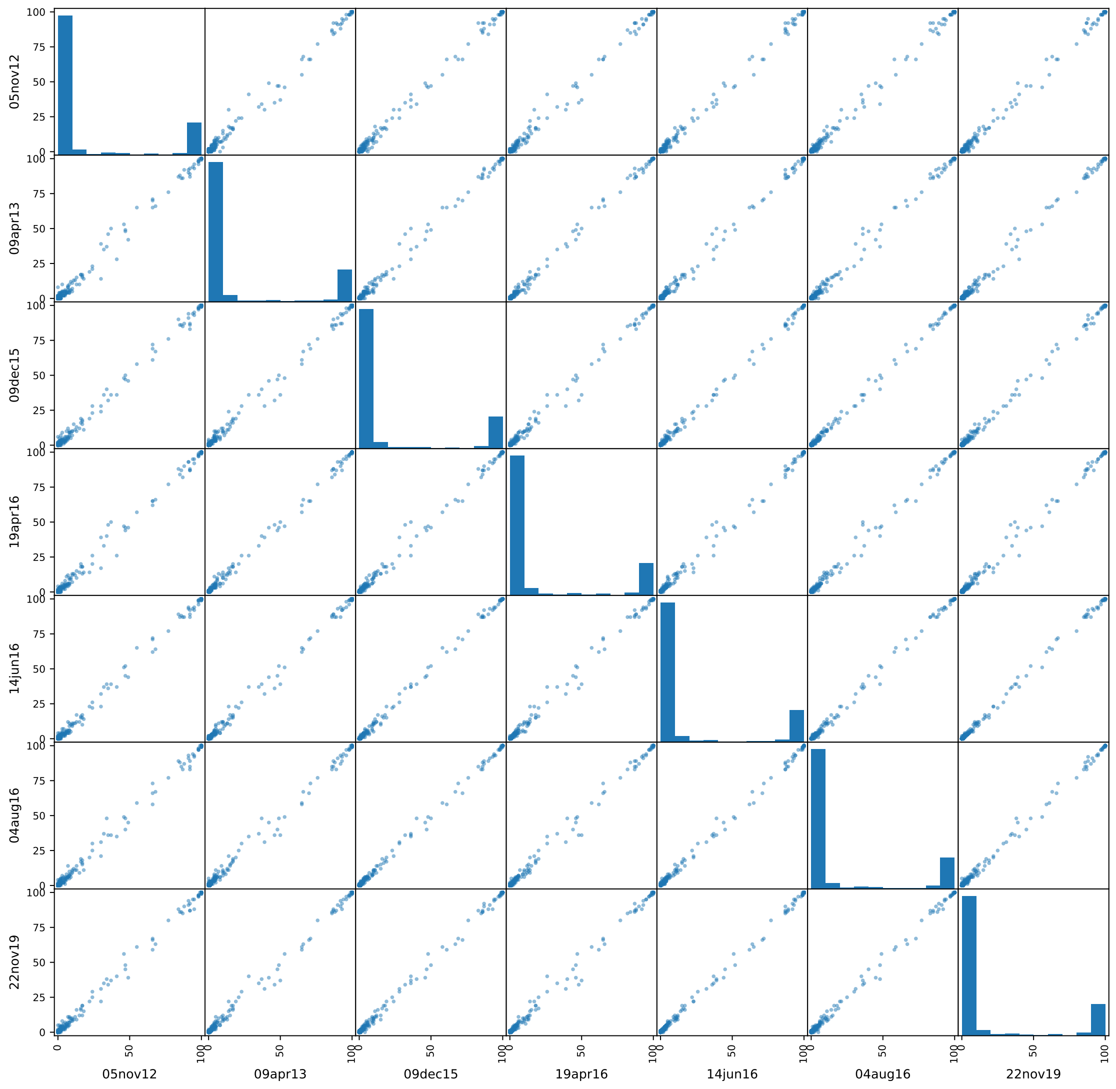

Figure 1. Bipartition frequencies across pb_mpi versions: TreeBase M1487 dataset

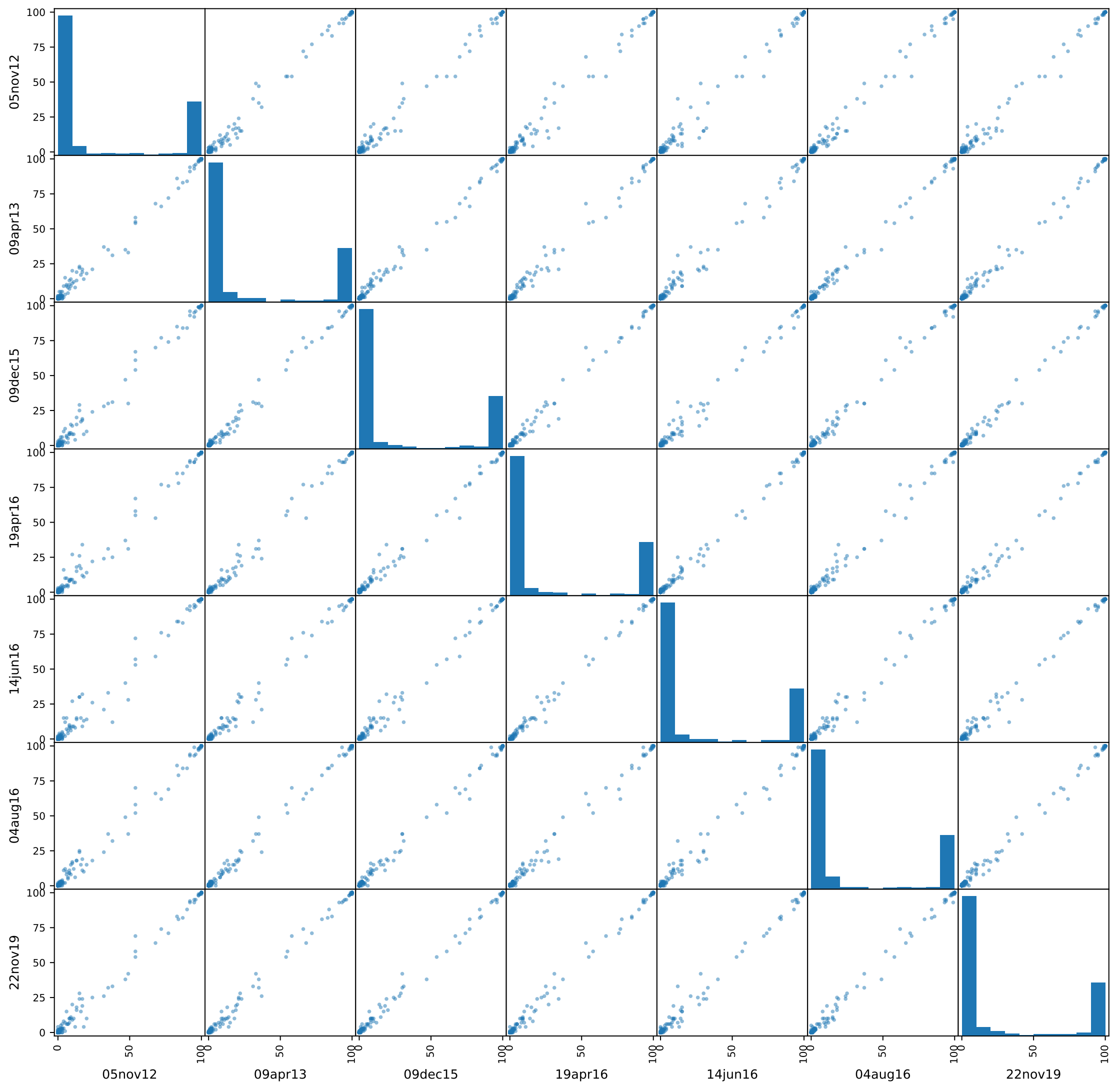

Figure 2. Bipartition frequencies across pb_mpi versions: mammalian mitochondrial dataset

process. This approach turns out to be quite problematic. First, it makes the posterior predictive sampling routine not root-invariant. Another more fundamental consequence is that the simulation process is then not at equilibrium. In particular, in the context of the compositional tests, this means that the program was actually testing only for time-homogeneity, but not for stationarity, of the process.

Sampling the root state from the equilibrium distribution of the process was implemented in April 22, 2016 and then set up as the default behavior of the program in the version of June 06, 2016. Thus, the posterior predictive test is correct since June 06, 2016 and has not changed since then. Of note, the previous approach (sampling from the conditional posterior at the root) can still be activated in the current version of the program (by using the `-ppredroot posterior` option). However, it is there only for the record, and its use is not recommended.

In practice, the change from conditional posterior sampling to conditional prior sampling (i.e. sampling from equilibrium frequencies) of the root state is dataset dependent. For data that are compositionally homogeneous across taxa (simulated data, first table), the bug has virtually no impact. In cases showing moderate compositional heterogeneity (second table, mitochondrial data), the z-scores are moderately impacted. This makes sense: if the data have relatively minor deviations from the compositional homogeneity assumption, then, root states sampled from the conditional posterior are close to equilibrium under the inferred substitution process. Finally, for data showing strong deviation from compositional homogeneity (arthropod mitochondrial data, third table), the z-scores can be substantially different between the two versions of the test, although the qualitative outcome is not fundamentally different, showing strong rejection in all cases. As for the diversity test, it appears to be relatively insensitive to the choice between the two alternative approaches for sampling the states at the root.

We would like to apologize for the confusion and the lack of reproducibility of the posterior predictive checks that these problems might have caused.

code version	comp (max)		comp (mean)		div	
	rep a	rep b	rep a	rep b	rep a	rep b
09dec15	-0.14	-0.14	-	-	0.57	0.54
19apr16	0.05	0.04	-	-	0.83	0.80
22apr16	0.05	0.06	2.15	2.24	0.88	0.90
06jun16	0.02	0.06	2.13	2.22	0.85	0.87
14jun16	0.04	0.06	2.20	2.26	0.80	0.82
04aug16	-0.01	0.04	2.16	2.15	0.87	0.86
22nov19	0.03	-0.00	2.19	2.16	0.88	0.87

code version	comp (max)		comp (mean)		div	
	rep a	rep b	rep a	rep b	rep a	rep b
09dec15	11.45	11.36	-	-	1.98	1.90
19apr16	13.89	14.51	-	-	2.65	2.69
22apr16	14.70	14.36	18.41	20.11	2.82	2.57
06jun16	14.54	14.24	21.45	21.46	1.94	1.96
14jun16	14.21	14.65	21.36	20.90	1.96	2.10
04aug16	15.17	14.68	22.16	21.08	1.92	1.89
22nov19	14.27	13.91	21.60	19.83	1.72	1.86

code version	comp (max)		comp (mean)		div	
	rep a	rep b	rep a	rep b	rep a	rep b
09dec15	11.03	10.93	-	-	0.96	0.86
19apr16	23.08	24.66	-	-	1.25	1.43
22apr16	23.29	24.93	23.09	24.44	1.36	1.36
06jun16	42.50	43.53	49.62	51.00	1.23	1.30
14jun16	42.62	43.47	50.99	50.71	1.37	1.24
04aug16	42.57	43.01	49.90	49.82	1.29	1.25
22nov19	41.50	42.92	50.89	50.75	1.38	1.26

Table 1. posterior predictive z-scores; top: data simulated under a time-homogeneous and stationary model (cat-gtr, based on arthropod mitochondrial dataset, 20 taxa, 1243 aligned amino-acid positions); middle: mammalian mitochondrial dataset (42 taxa, 3507 aligned amino-acid positions), bottom: arthropod dataset (20 taxa, 1243 aligned amino-acid positions). Two replicates per version are shown.

References

Blanquart, Samuel, and Nicolas Lartillot. 2006. A Bayesian compound stochastic process for modeling nonstationary and nonhomogeneous sequence evolution. *Mol. Biol. Evol.* 23:2058–2071.

Delsuc, Frédéric, Georgia Tsagkogeorga, Nicolas Lartillot, and Hervé Philippe. 2008. Additional molecular support for the new chordate phylogeny. *Genesis* 46:592–604.

Gelfand, A E, Dipak K Dey, and Hong Chang. 1992. Model determination using predictive distributions with implementation via sampling-based methods. In *Bayesian statistic, 4th edn*, ed. J M Bernardo, J O Berger, A P Dawid, and A F M Smith, 147–167. Oxford: Oxford University Press.

Geyer, C. 2006. Practical Markov Chain Monte Carlo. *Stat. Sci.* 7:473–483.

Lartillot, Nicolas. 2022. Identifying the best approximating model in Bayesian phylogenetics: Bayes factors, cross-validation or wAIC? *bioRxiv* 1–46.

Lartillot, Nicolas, Henner Brinkmann, and Hervé Philippe. 2007. Suppression of long-branch attraction artefacts in the animal phylogeny using a site-heterogeneous model. *BMC Evol. Biol.* 7 Suppl 1:S4.

Lartillot, Nicolas, and Hervé Philippe. 2004. A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process. *Mol. Biol. Evol.* 21:1095–1109.

Lewis, Paul O, Wangang Xie, Ming-Hui Chen, Yu Fan, and Lynn Kuo. 2014. Posterior predictive Bayesian phylogenetic model selection. *Syst. Biol.* 63:309–321.

Quang, Le Si, Olivier Gascuel, and Nicolas Lartillot. 2008. Empirical profile mixture models for phylogenetic reconstruction. *Bioinformatics* 24:2317–2323.

Rodrigue, Nicolas, Hervé Philippe, and Nicolas Lartillot. 2010. Mutation-selection models of coding sequence evolution with site-heterogeneous amino acid fitness profiles. *Proc. Natl. Acad. Sci. USA* 107:4629–4634.

Vehtari, Aki, Andrew Gelman, and Jonah Gabry. 2016. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Stat Comput* 27:1413–1432.

Wang, Huai-Chun, Karen Li, Edward Susko, and Andrew J Roger. 2008. A class frequency mixture model that adjusts for site-specific amino acid frequencies and improves inference of protein phylogeny. *BMC Evol. Biol.* 8:331.

Watanabe, S. 2009. *Algebraic Geometry and Statistical Learning Theory*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.