PhyloBayes 3.1

a Bayesian software for phylogenetic reconstruction and molecular dating using mixture models

Nicolas Lartillot nicolas.lartillot@lirmm.fr
Samuel Blanquart samuel.blanquart@lirmm.fr
Thomas Lepage thomas.lepage@mcgill.ca

1. Introduction

PhyloBayes is a Bayesian Monte Carlo Markov Chain (MCMC) sampler for phylogenetic reconstruction and molecular dating using protein and nucleic acid alignments. Compared to other phylogenetic MCMC samplers, the main distinguishing feature of PhyloBayes is the use of non-parametric methods for modelling site-specific features of sequence evolution.

PhyoBayes3.1, implements a wide spectrum of probabilistic models, among which:
- non-parametric models of site-specific rates, of equilibrium frequency profiles, or even, of the entire GTR matrix. All this using a combination of independent Dirichlet processes.
- Tuffley and Steel's covarion model, as well as the mixture of branch length (mbl) models.
- empirical mixtures of profiles, or of matrices.
- auto-correlated and non auto-correlated models of clock relaxation, meant for molecular dating

Modelling site-specific effects using non-parametric methods

In general, each position of a gene is under a very specific selective constraint. This fact has fundamental consequences for phylogenetic reconstruction. For instance, more constrained sites are slower, and less constrained sites are faster. Similarly, only a subset of the possible states (4 nucleic acids or 20 amino-acids) is likely to be accepted at a given position during evolutionary times.

This suggests that the overall rate of substitution, the equilibrium frequency profile (more briefly called ***profile*** in the following) of the substitution process, or even the entire substitution matrix, should be modeled as site-specific random variables. As has been shown in several previous works, accounting for such variations across sites is indeed crucial, both to obtain a better statistical fit and to alleviate phylogenetic artifacts, due to long branch attraction phenomena.

There are two ways site-specific evolutionary 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 (Yang, 1994).

Alternatively, one can use *non-parametric* models: the general idea of non-parametric model is that the overall shape of the distribution across sites is not specified a priori, but is itself directly inferred from the data. Thus, it is much more general than the parametric method. On the other hand, it requires more data (since more is learnt from them).

There are many different ways to make non-parametric models. A practical approach, often adopted in Bayesian inference, is the use of Dirichlet process mixtures. These are infinite mixtures (in practice, mixture models whose number of categories is not directly specified, but integrated over a prior distribution).

PhyloBayes uses a combination of two independent Dirichlet processes: one for modeling the site-specific rates (Huelsenbeck and Suchard, 2007), and another one aimed at describing the sites-specific profiles (Lartillot and Philippe, 2004). Through this combination, each site is given a rate and 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 flat values (the CAT-Poisson, or more simply, CAT, settings), or inferred from the data (CAT-GTR settings).

In the QMM ('Q-Matrix Mixture') model, the second Dirichlet process is not defined only on the equilibrium frequency profile, but on the entire rate matrix (Q-matrix): in that case, and contrarily to what happens in the CAT-GTR model, sites do not share the same exchange rates. However, the QMM model seems in fact to be less fit than the simpler CAT-GTR version, with global exchange rates shared by all sites.

Empirical mixture models

The non-parametric models introduced above are very flexible. They automatically estimate the distribution of site-specific effects underlying each dataset. But on the other hand, they may require a fairly large amount of data for correct 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 the more recent LG (Le and Gascuel, 2007) are a specific case of empirical model, with only one component. Recently, empirical mixtures have been proposed: either mixtures of profiles (Le et al 2008, Wang et al 2008), or mixtures of 2 or 3 matrices (Le et al, 2008b). All these models are implemented in the current version of PhyloBayes. The user can also specify its own custom set of exchange rates, mixture of profiles, or mixture of matrices.

Phylogenetic reconstruction and molecular dating

Once a sample is obtained, it can be marginalized over the parameters of interest. Most often, people are interested in the phylogenetic tree itself: here, PhyloBayes works like usual Bayesian phylogenetic reconstruction programs, and outputs a majority rule posterior consensus tree. But conversely, you might be interested in the site-specific biochemical specificities that have been captured by CAT, or some other site-specific or mixture model. In that case, you can read out the mean posterior site-specific rates and profiles, which are also directly available from the MCMC output. Note that, in contrast to usual sequence profiles, these site-specific features are corrected for the phylogenetic correlations between the sequences. You may also be interested in the goodness-of-fit of the model, which can be assessed by Bayes factors, cross-validations, or posterior predictive resampling procedures.

Finally, PhyloBayes can be used to perform ***Bayesian molecular dating*** using relaxed molecular clocks. Several alternative clock relaxation models are available, including the popular log-normal auto-correlated model of Thorne et al. (1998), the CIR process of Lepage et al (2007). Fossil calibrations can be provided as hard (as in Kishino et al 2001) or as soft (as in Yang and Rannala 2006) constraints.

Choosing the right model

It is still difficult to give 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.

Amino-acid replacement models

- For large datasets (more than 1,000 aligned positions), CAT-GTR is virtually always the model with highest fit among all models implemented in PhyloBayes, whether non-parametric or empirical.

- CAT is less fit than CAT-GTR, but generally better than one-matrix or empirical models on large datasets.

- Empirical mixture models (UL3, C60 or WLSR5) may be ideal for single gene datasets.

- Concerning the posterior consensus tree, both CAT and CAT-GTR seem to be significantly more robust against long branch attraction artifacts, compared to all other models (although the empirical mixtures still need to be investigated in more details for their phylogenetic accuracy). In addition, CAT seems to yield more 'skeptical' outputs (lower posterior probabilities) compared to CAT-GTR.

- Concerning computational efficiency, CAT is by far the most efficient settings. Thus, for large datasets, CAT probably remains the best option, in particular when there is a high level of saturation. However, comparing the results obtained under CAT, CAT-GTR and GTR (or returned by other programs), whenever possible, is important.

Nucleotide substitution models

Very few options are implemented for nucleic acid data in Phylobayes, which is primarily devoted to amino-acid data. However, the CAT-GTR model is probably a very good model for DNA or RNA data.

Rates across sites

The Dirichlet process over the rates (as in Huelsenbeck and Suchard, 2007) is probably better than the more classical discrete gamma distribution with 4 categories (although this question would require more detailed investigations). Thanks to parameter expansion, this model also requires less RAM, and is faster on a per-cycle basis, than the gamma. On the other hand, parameter expansion may sometimes result in MCMC convergence and mixing difficulties.

2. Downloading and installing PhyloBayes:

PhyloBayes is freely available. It can be downloaded from www.phylobayes.org. It is provided as a zipped file **phylobayes3.1.tar.gz**. There are several pre-compiled versions, for mac, linux 64 bits and windows.

Once unzipped, the downloaded file will release a directory, **phylobayes3.1**, containing several subdirectories: **sources** containing the source files; **data**, contaning the data files; several **exe_** directories containing the binary files for mac, linux and windows, and **aux**, containing auxiliary files (necessary for optional postscript outputs). If you need or wish to recompile, simply go into the **sources** directory, and use the following command:

```
make
```

This will put new binaries directly into the **data** directory. Since the CIR clock relaxation model (Lepage et al, 2007) requires functions from the **gsl** library, to make compiling as simple as possible for everyone, the CIR has been deactivated by default. Thus, to compile as described above, you should not need anything particular pre-installed on your computer. In particular, you do not need to install the **gsl** library. But on the other hand, you will not be able to use CIR. Conversely, if you want to use the CIR model, you need to activate a compilation in the Makefile (at the very beginning of the file, remove the '#' in front of **USE_GSL=-DGSL**). Of course, you then need **gsl** to be installed.

The general philosophy of PhyloBayes is a bit different from other programs: rather than having one single program that does everything, through standard input and output streams (i.e. keyboard and screen), several small programs have been written, working more or less like unix commands using input and output files. The detailed model and Monte Carlo settings can be controlled through options, starting with the usual hyphen '-'.

The program files can be put anywhere. For instance, you can copy the relevant binaries (depending on your operating system) into the **data** directory. Then you can create subdirectories in **data**, which will contain all your analyses using PhyloBayes (for instance, one distinct sub-directory for each phylogenetic session made of related analyses on the same dataset). Alternatively you can put the programs in **/usr/bin/**, if you want to have access to them from anywhere in your file system.

Finally, for the postscript options to work, first, LateX and dvips have to be installed, and second, a global variable needs to be specified, e.g. by writing the following two lines at the end of the **.bashrc** file which is in your home directory:

```
set PBPATH = .../aux_ps/
export PBPATH
```

The "**.../**" in front of **aux/** should be replaced by the path leading to where exactly you have put the **aux** directory. Note, however, that all programs output only plain text files by default. Therefore, as long as you do not explicitely ask for postscript outputs (using the **-ps** option), all of them will work even if this path has not been specified.

3. Input Data Format

Sequences

Two main formats are recognised: NEXUS, and a generalisation of the PHYLIP format, formatted as follows :

```
<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. Be sure that the lengths of the sequences are all the same, and identical to the lengths indicated in the header. Sequences can be interleaved, in which case the taxon names may or may not be repeated in each block.

The following characters will be considered all 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 recognised, but the case matters for taxon names.

Trees

In some cases, an initial tree can be provided, or the program can be constrained to sample the posterior distribution of parameters under a specified tree topology, which will the remained fixed throughout the analysis. Trees should be provided in NEWICK format, e.g.:

```
(taxon1,(taxon2,taxon3),(taxon4,taxon5));
```

or

```
((taxon1,(taxon2,taxon3)),(taxon4,taxon5));
```

The first tree is unrooted, the second is rooted. Both are recognised. Branch lengths can be specified, but will be ignored. Trees should always be followed by a ';' **and should not contain spaces or tabs.**

Taxon names should correspond to the names specified in the data matrix (case sensitive). Alternatively, you can provide integer labels, in which case the order of the taxa provided by the datafile is considered. In the case where names are used: 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, this will generate an error message.

Outgroups

For molecular dating, a tree needs to be specified, and its rooting matters. If a rooted tree is given as an input, it is used as such. If an unrooted tree is given, an outgroup has to be specified to the program. For this, the outgroup has to be written into a file according to the following format:

```
<number of taxa>
<taxon1>
<taxon2>
...
```

All the taxa making the outgroup should be specified. They should correspond to a split of the tree.

Calibrations

Molecular dating requires fossil calibrations to be specified. The format is as follows:

```
<ncalib>
<taxon1a> <taxon1b> <upper_limit> <lower_limit>
<taxon2a> <taxon2b> <upper_limit> <lower_limit>
...
```

The calibration given on each line will be imposed onto the last common ancestor of the two specified taxa. Upper limits are older, and thus, should always specify larger numbers than lower limits. The only exception is that upper or lower limits can be set equal to -1, in which case no limit is enforced. For example ;

```
taxon1 taxon2 -1 50
```

means that the node of the last commmon ancestor of taxon1 and taxon2 should be older than 50 Million years (My), but otherwise, does not have an upper constraint. Likewise:

```
taxon1 taxon2 70 -1
```

only specifies an upper constraint of 70 My, and no lower constraint. And finally:

```
taxon1 taxon2 70 50
```

specifies an upper and a lower constraint, thus the interval [50,70] My.

4. General presentation of the programs:

There are 11 different programs in the package (for details about any of these programs type its name without any argument to get an online help):

pb/stoppb : runs and stops the MCMC sampling proper.
readpb : post-analysis program, that returns the posterior consensus tree, as well as 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 files.
ppred : performs posterior predictive analyses.
readdiv : post-analysis program for relaxed clock analyses.
bf : for comparing relaxed clock models by Bayes factor numerical evaluation.
cvrep/readcv/sumcv : for comparing models by cross-validation.
tree2ps : converts a tree in newick format into a postscript file.

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

Running a chain (pb / stoppb)

A run of the main program of PhyloBayes (**pb**) will produce a series of points drawn from the posterior distribution over the parameters of the model. Each point defines a value for all these parameters (which include the tree topology, the branch lengths, the biochemical profiles of the mixture, etc.). The series of points defines a Monte Carlo Markov chain, or more simply, a *chain*.

Each time you create a new chain, you give it a name. An example is given in the package: the **phylobayes/data/brpo/** directory contains a file named **brpo.ali**, which is an alignment of eukaryotic RNA polymerases. To conduct an inference on this data set, first go into the **brpo** directory. From there, you can create a new chain, of name **rnapol1**:

```
../pb -d brpo.ali rnapol1
```

The "**../**" pathway may be modified depending on where exactly you have put the program. It will be omitted in the following. The **-d** option allows you to specify the dataset. There are many other options for specifying the model (see below), but the default options (CAT + Gamma) are fine (at least, for proteins). Before starting, the chain will output a summary of the settings on the screen.

It is often more convenient to run the chain in background. You can also make two independent chains in parallel:

```
pb -d brpo.ali rnapol1 &
pb -d brpo.ali rnapol2 &
```

A series of files will be produced, whose names all start with the name of the chain, combined with a variety of extensions. The most important are:
<name>.treelist: list of sampled trees.
<name>.trace: the trace file, containing a few relevant statistics (e.g. number of generations, log-likelihood, total length of the tree, number of components in the mixture), which can help you checking how fast the chain equilibrates, and how long you will need to run it. The best is to plot the evolution of a few of these statistics, as a function of time, to check for the absence of trend in the long run.

The chains will run as long as you let them run. You can stop, interrupt, or even kill them at any time, and restart them afterwards. In that case, they will resume from the last point that was saved before the interruption. To soft-stop a chain, just type **stoppb <name>**, e.g. :

```
stoppb rnapol1
```

The chain will finish the current cycle before exiting. To restart an already existing chain : **pb <name>** (be careful not to restart an already running chain). In our example :

```
pb rnapol1 &
```

Checking for convergence (bpcomp and tracecomp)

The program saves one point after each cycle. A cycle itself is made of a varying number of generations: specifically, the number of generations (second column of the trace file) is defined in PhyloBayes as the number of elementary topological updates tried during a cycle. This number is not constant across cycles, because PhyloBayes implements recursive 'waves' of topological updates along the tree, whose number of steps might depend on the current topology.

In any case, the absolute number of generations performed by a MCMC sampler 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. We therefore need other ways of estimating when to stop the run, and how much burn-in to discard.

First, one can monitor the evolution of the log-likelihood and other statistics as a function of time. This will be a good indication of how well the profile-mixture has reached the stationary state. But on the other hand, it will not be sufficient, and furthermore, it lacks an objective basis.

An alternative is to run two (or more) chains in parallel, and compare the posterior distributions obtained under these several independent runs. Two main classes of variables can be considered: the summary variables that are displaed in the trace files, and the bipartition frequencies. The first is done using the **tracecomp** program, and the second using the **bpcomp** program. Both use the same syntax:

```
bpcomp -x 100 2 rnapol1 rnapol2
```

Here, with a burn-in of 100, and taking every 2 trees (**-x 100 2**), the **bpcomp** program will output the largest (**maxdiff**) and mean (**meandiff**) discrepancy observed across all bipartitions (**maxdiff**). If in addition you specify an output file :

```
bpcomp -x 100 2 -o rnapol rnapol1 rnapol2
```

it will give you a detailed comparison of the frequency of all the bipartitions observed in the chains being compared, and will output a consensus tree made by pooling all the trees of all the chains.

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 big yet enough, but it seems to be on the right track.
- if **maxdiff** = 1, even after 10,000 points, this is really bad : it indicates that at least one of the runs is stuck in a local maximum.

Similarly:

```
tracecomp -x 100 2 rnapol1 rnapol2
```

will produce an output summarizing the discrepancies and the effective sizes estimated for each column of the trace file. The discrepancy is defined as $D=2|m1 - m2| / (s1 + s2)$, where m_i is the mean and s_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 (1992). The guidelines are:

- maximum discrepancy < 0.1 and minimum effective size > 100 : good run,
- maximum discrepancy < 0.3 and minimum effective size > 50 : acceptable run.

Automatic stopping rule (pb)

The program now has an option to run several chains in parallel, regularly probing the convergence, and stopping automatically as soon as
- the bipartition (**maxdiff**) and summary variable discrepancies all become lower than a specified threshold c.
- the effective sizes associated to all summary variables are all larger than a specified minimum size.
The burnin is chosen as 1/5 of the chain's length. Thus:

```
pb -d brpo.ali -nchain 2 100 0.3 50 rnapol
```

will run 2 chains in parallel, for a minimum of 500 cycles. Then, every 100 cycles, a **bpcomp** and a **tracecomp** between the two chains are automatically done (with a burnin equal to one fifth of the total length of the chain) and the run stops once all the discrepancies are lower or equal to 0.3 and all effective sizes are larger than 50.

Note however that the two chains are run on the same processor, which is different from the situation above, where the two chains could potentially be run on two different processors. Thus, it will take about twice as much time to accomplish the same number of cycles. On the other hand, the program is devised so that the two chains do not run simultaneously, but are just 'intertwined': that is, chain 1 runs for one cycle, then chain 2 runs for 1 cycle, and then both are saved, and so on. In this way, the two chains can share most the RAM they need, and do not 'overcrowd' the processor on which they run.

Post-analysis (readpb)

Once a chain has been obtained, one usually removes the first few hundreds of points (the so-called burn-in), and use the remaining points to compute averages (posterior consensus, mean site-specific profiles, etc.). All this is done with the **readbp** program. The series of points that is used by readpb to compute averages is called a *sample* (understand : a (sub-)sample of points taken from a chain).

Following our example :

```
readpb -x 100 10 rnapol1
```

will take one tree every ten, discarding the first 10 trees. It will produce a few additional files, among which :

rnapol1_sample.general, containing a few general statistical averages (mean posterior log likelihood, tree length, alpha parameter, number of modes, etc.)
rnapol1_sample.con.tre : the majority-rule posterior consensus tree
rnapol1_sample.bplist : the list of weighted bipartitions.

Various options allow you to compute several other posterior averages, such as site-specific rates and equilibrium frequency profiles.

Estimating divergence times (pb / readdiv).

PhyloBayes includes a variety of clock relaxation models, including the log-normal auto-correlated model, as in Thorne et al. (1998), and the CIR process (Lepage et al., 2007). It can work with or without the normal approximation, and in the latter case, using any of the substitution models implemented in the program. In particular, thanks to a better estimation of multiple substitutions (see Lartillot et al, 2007), CAT could turn out to be a better model for estimating branch lengths, and therefore also divergence times.

Molecular dating can be done using **pb** program. It only works under a fixed topology, that has to be specified using the **-T** option. The molecular dating option of **pb** is activated by specifying a model of clock relaxation. An example is proposed in the package, in the **moc** directory:

```
pb -d moc.ali -T moc.tree -ln mocln1
```

Here, log-normal autocorrelated rates are assumed (**-ln** option), but this can be changed using the relevant options. Unless specified otherwise, **pb** assumes that the tree is correctly rooted. Note that if no relaxed clock model were specified, the program would simply sample parameters assuming the usual deconstrained model, which is non time-aware and directly infers the branch lengths in expected number of substitutions.

Any substitution model can be used for estimating divergence times. You can also run chains under the normal approximation :

```
pb -cov <oesfile> -ln <name>
```

where **<oesfile>** is a "estbranches" file, containing a tree,with ML branch lengths, and a variance covariance matrix. Any file obtained using estbranches will be accepted here, after the **-cov** option. One such file (**moc10.oes**) is provided in the package (from Douzery et al, 2004):

```
pb -cov moc10.oes mocoesln1
```

If you do not provide any calibration, you will get relative age estimates: by definition, the root will have age 100, and the leaves age 0. Alternatively, you can specify a set of calibrations in a file (here **calib**), having the relevant format (see above, format section):

```
pb -d moc.ali -T moc.tree -r bikont.outgroup -ln -cal calib name
```

Once a chain has been obtained, you can read out the posterior mean node ages using **readdiv**:

```
../readdiv -x 100 1 <chainname>
```

this will output a chronogram (**<chainname>_sample.chronogram**), as well as a file containing the list of divergence times, with standard errors (**<chainname>_sample.dates**). The labels of the nodes are indicated in a specific tree, in the file with the ".labels" extension.

Model evaluation.

Apart from computing averages from the posterior distribution, Bayesian inference also allows for model comparison and goodness-of-fit analyses. Model comparison often consists in computing Bayes factors. However, numerical evaluation of Bayes factor is a very difficult task. Simple estimators often fail. For instance, the popular harmonic mean estimator is not reliable (Lartillot and Philippe, 2006), and is thus not implemented here. Path sampling, or thermodynamic integration, is more reliable, and we have used it previously for comparing phylogenetic models (Lartillot and Philippe 2006). But it is time-consuming, and is not implemented here for general substitution models. In the present version, we only provide thermodynamic integration for comparing clock models under the normal-approximation: this can be done using the program called **bf**.

For example:

```
bf -cov moc10.oes -ln moc10ln
```

will compute the Bayes factor between the log-normal and the deconstrained model. You can do this for several models, and then choose the model returning the largest Bayes factor.

Apart from Bayes factors, another general model comparison method in Bayesian inference is cross-validation. It is implemented in the PhyloBayes package. The exact procedure is a bit involved, and is explained at the end of this manual.

Finally, models can also be assessed by posterior predictive resampling, which is the Bayesian equivalent of the parametric boostrap often used in Maximum Likelihood. The idea is to compare the observed value of a given test statistic *s* on the true data, with the distribution of *s* on data replicates simulated under the reference model. If the observed value of *s* deviates significanlty, this means that the reference model is rejected.

According to parametric boostrap, replicates should be simulated under the ML value of the parameter. The posterior predictive formalism is slightly different: simulations should instead be performed using parameter values drawn from the posterior distribution. This has the advantage of 'smoothing out' your null distribution with respect to the uncertainty over your parameter estimates. On the other hand, the test may then be sensitive to the prior distribution.

Posterior predictive resampling can be done using the **ppred** program. By default, **ppred** takes a chain as an argument (like **readpb**), and for each point of the sample, simulates a series of replicates of the dataset using the topology and the parameters defined by the current point. For instance, say you have run a chain of length 1100 points, under the wag model:

```
pb -d brpo.ali -x 1 1100 -wag brpowag1
```

Once this chain is done, you can then use it to obtain a series of 100 replicates using **ppred**:

```
ppred -x 100 10 brpowag1
```

Here, the options tell **ppred** to take one point every 10, after a burn-in of 100, and use the parameters defined by each of these points to simulate replicates. Once replicates have been obtained, any test statistic which can be computed on the true dataset (e.g. compositional deviation of a given taxon) can also be computed on each of the replicates, which give a null distribution, with which to compare the true value obtained on the original dataset. The implementation of the test statistisc is left to the user, except for 3 pre-defined statistics: saturation, site-specificity, and compositional homogeneity.

5. Detailed program options.

pb

General options:

`-d <datafile>`

allows one to specify a file containing aligned sequences. For the exact specifications of the formats accepted by the program, see above (see: Input Data Format).

`-dc`

constants sites are removed.

`-cov <varcovar>`

specifies a variance-covariance matrix. This matrix can be a file produced by the **`estbranches`** program of Thorne et al.

`-t <treefile>`

forces the chain to start from specified tree.

`-T <treeefile>`

constrains the chain to run under the topology specified in **`<treefile>`**. It will thus only samples from the posterior distribution over all other parameters (branch lengths, alpha parameter, etc.), conditional on the specified topology.

`-r <outgroup>`

re-roots the tree, using the specified outgroup, before starting the chain. Current substitution models implemented in PhyloBayes are reversible, so this rerooting is useful only for clock models.

`-s`

in order to save disk space, by default the current version only saves the trees explored by the chain during MCMC. This is enough if what you want to do is only computing a consensus tree. However, if you wish to do a more detailed analysis, such as looking at the site-specific biochemical profiles, or perform posterior predictive tests, then, you need the program to save the whole parameter vector for each visited point, and not only the current tree. The **`-s`** option is meant for this.

Fixed topology analyses (where you have used the **`-T`** option to constrain the tree topology) will automatically activate the **`-s`** option.

-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 (a cycle is not a well defined concept in MCMC programs: it is very model AND implementation dependent). In some cases, you may realise that your samples are very correlated, and that they are taking too much space on your hard-drive. In that case, it would make sense to save points less frequently, say 10 times less often: then, you should use the **-x 10** option.

-nchain <nchain> [<step> <cutoff> <eff_size>]

runs **nchain** chains in parallel, for a minimum of **burnin_factor x step** cycles, where **burnin_factor** is a parameter equal to 5 by default; then, every **step** cycles, checks (using **bpcomp** and **tracecomp**) the discrepancy between the two chains and the effective sample sizes, and stops if this discrepancies are below the specified **cutoff**, and the effective sample sizes are larger than the specified **eff_size.**

-b <burnin_factor>

tunes the burnin-factor parameter. This parameter is is used to define the burnin when checking for the convergence of independent chains (see **-nchain** option). It is equal to 5 by default (thus, by default, the burnin is 1/5 of the total length of the chains).

Evolutionary models

Prior on branch lengths:

-lexp : prior on branch lengths is a product of i.i.d. exponential distributions.

-lgam : prior on branch lengths is a product of i.i.d. gamma distributions (this is more flexible than the exponential, as it allows to independently control the mean and the variance of the branch length distribution).

-meanlength <meanlength> : fixes the mean of the exponential (or gamma) priors to the specified value. Without this command, this mean is considered as a free parameter, endowed with an exponential prior of mean 0.1. In the case of the gamma prior, the variance/mean ratio is always considered as a free parameter, with en exponential prior of mean 1.

Rates across sites:

-ratecat : the distribution of rates across sites is emulated by a Dirichlet process (as in Huelsenbeck and Suchard, 2007).

-uni : uniform-rate model.

-cgam : continuous gamma distribution.

-dgam [<n>] : discrete gamma distribution with **<n>** categories.

The default model is the Dirichlet process.

Relative exchangeabilities (exchange rates):

-poi (or **-poisson**): exchange rates all equal to 1. The model is then a mixture of Poisson (Felsenstein81) processes.

-wag, **-jtt**, **-mtrev**: empirical exchange rates (Jones and Taylor, 1992, Whelan and Goldman, 2002, Adachi and Hasegawa 1996).

-lg: empirical exchange rates obtained by Le and Gascuel 2008.

-gtr: general time reversible matrix: exchange rates are free parameters, with prior distribution a product of independent exponential distributions of mean 1.

-rr -gtr <filename>: exchange rates are fixed, according to the specifications provided by the specified file. The file should read as follows:

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

You may specify the order in which amino acids should be considered on the first line (**[<ALPHABET>]**). Letters should be separated by spaces, tabs or returns. By default, if you do not specify it, this will be the alphabetical order of the one letter code (A C D E F G H I K L M N P Q R S T V W Y), in which case the order in which the file would have to specify the relative rates should be as follows: between A and C, then between A and D ... between A and Y on the first line; then on the second line, between C and D, etc. Tabulations, spaces and returns are all considered equivalent: only the order in which all these real numbers are specified is important.

Profile mixture:

-cat : Dirichlet process (number of components, weights and profiles are all inferred from the data).

-ncat <n> : mixture of **<n>** profiles. (number of components is fixed, profiles and associated weights are inferred).

By default, the Dirichlet process is activated. Fixing the number of components of the mixture (**-ncat <n>**) yields a poor mixing.

Empirical profile mixture models:

-catfix <predef> : mixture of a set of pre-defined profiles (only the weights are inferred). **<predef>** can be either one of the following keywords: C20, C30, C40, C50, C60, which correspond to empirical profile mixture models, learnt on the HSSP database (Le 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> : where **<filename>** is the name of a file containing a set of profiles specified as follows:

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

where **<ncat>** is the number of profiles, and each line following this number should be a set of 20 real numbers, defining a profile of equilibrium frequencies. You may specify the order in which amino acids should be considered on the first line (**[<ALPHABET>]**). Letters should be separated by spaces, tabs or returns. By default, if you do not specify it, this will be the alphabetical order of the one letter code. Tabulations, spaces and returns are all considered equivalent.

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. To have different sets of exchange rates, see below (matrix mixture models)

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 -poi**, that is, a mixture of Poisson (Felsenstein1981) processes. The program works best under those conditions.

Note also that, if you activate non-Poisson exchange rates, the mixture model is automatically desactivated by default: you only have one such matrix, conditioning all the sites. Thus, the **-gtr** option is equivalent to **-gtr -ncat 1**. Thus if you want a Dirichlet process combined with a GTR matrix of exchange rates, you have to EXPLICITELY call it **-gtr -cat**.

Matrix mixture models:

-qmm: activates the QMM model, which is a mixture of matrices (thus, allowing for DISTINCT sets of exchange rates and equilibrium frequencies for each component of the mixture).

-qmmfix ul2 or **-qmmfix ul3:** defines the empirical mixture with 2 or 3 components (Le et al, 2007b).

-qmmfix <filename>; where filename is the name of a file containing a set of matrices specified as follows:

```
[<ALPHABET>]

<ncat>

<freq1> <freq2> ... <freq20>
<rr1_2> <rr1_3>    ...   <rr1_20>
<rr2_3> <rr2_4> ...  <rr2_20>
...
<rr19_20>

<freq1> <freq2> ... <freq20>
<rr1_2> <rr1_3>    ...   <rr1_20>
<rr2_3> <rr2_4> ...  <rr2_20>
...
<rr19_20>

...
```

As for profile mixtures, you may specify the order in which amino acids should be considered on the first line (**[<ALPHABET>]**). Letters should be separated by spaces, tabs or returns. By default, if you do not specify it, this will be the alphabetical order of the one letter code. Tabulations, spaces and returns are all considered equivalent: only the order in which all this real numbers are specified is important.

Weights of mixture components:

By default, the weights are free parameters (endowed with a uniform Dirichlet prior).

-fxw : In case you use one of the pre-specified mixtures C10 to C60, the weights are fixed to the values estimated on the training HSSP database. In all other cases (other pre-specified profile and matrix mixtures, or custom mixtures), the weights are fixed, and set all equal. Thus, this option is interesting only when using empirical profile mixtures on very small datasets.

Heterotachy:

-covarion: Tuffley and Steel's covarion model. This can be combined with any of the substitution or amino-acid replacement models defined above. However, it does not work in combination with the **-ratecat** or **-cgam** options. Thus, you should either use a more classical discrete gamma distribution

with the **-cgam** option (or no variations or rates across sites, with the **-uni** option).

-covext: a slightly different version of Tuffley and Steel's covarion model, in which the rate of the site is applied both to the substitution and to the switching processes. That is: under the classical covarion model (-covarion), fast sites make more substitutions when in the on state, compared to slow sites. But all sites switch between the on and off states at the same rate. In contrast, under the covext model, fast sites also switch faster between on and off. In practice, this means that the rate is simply applied as a multiplier in front of the tensor product matrix. The advantage of doing this is that it can be combined with any model of rate variation across sites (although you may encounter identifiability problems in practice).

-mmbl: This is a Bayesian non-parametric version of the mixture of branch lengths (MBL) model (Kolaczkowzki and Thornton, 2008, Zhou et al 2008). Basically a Dirichlet process of vectors of branch length multipliers. These multipliers are drawn from a gamma distribution, whose inverse variance (ϕ) is itself considered as a free parameter. Convergence is challenging under this model. You should always check carefully for convergence of the topology and of the continuous parameters.

-gbl <partition>: separate model, with each gene having its own set of branch lengths, that are obtained by multiplying the global branch lengths by gene-specific branch length multipliers. These multipliers are drawn from a gamma distribution, whose inverse variance (φ) is itself considered as a free parameter. Convergence in the space of topology is challenging under this model. The partition file should be specified as follows:

<G>
nsite_1 nsite_2 ... nsite_G
where **G** is the total number of genes, and **nsite_g** is the number of aligned positions for gene *g*.

Relaxed clock models:

-cl : strict molecular clock
-ln : log normal model (Thorne et al, 1998)
-cir: CIR process (Lepage et al, 2007)
-wn : white noise process (Lepage et al, 2007)
-ugam : uncorrelated gamma multpliers (Drummond et al, 2006).

prior on divergence times (by default, a uniform prior):

-bd [<p1> <p2>] : birth death prior. The two parameters p1 and p2 are the hyperparameters of the birth death process (p1 = birth rate - death rate, and p2 = birth rate * sampling rate, see Lepage et al 2007 for details). If two real numbers are given, p1 and p2 will be fixed to the specified values. Otherwise, they will be inferred from the data.

-bdhyperprior [<mean1> <stdev1> <mean2> <stdev2>] : birth death hyperprior. The two parameters p1 and p2 will be endowed with gamma distributions of means and priors specified by the command. Default values are 1, 1, 1, 1 (exponential distributions of mean 1 for both p1 and p2).

-dir [<chi>] : Dirichlet prior. chi>0 is the hyperparameter (with smaller value resulting in a larger variance of the waiting times between successive speciations). chi can be fixed or be inferred from the data, depending on whether a real number is specified in the command. Note that in the specific case of the Dirichlet prior, analyses with calibrations require chi to be specified.

-cal <calibrations> : impose a set of calibrations see Format section for the exact format. the calibrations work only under the uniform prior on divergence times.

-sb [<cutoff>] : activates the soft constraint option (as described in Yang and Rannala, 2006). This option is available only under the birth death prior. The cutoff determines how much of the probability mass is allowed outside of the proposed boundaries. By default, **cutoff** = 0.05.

-lb [<p> <c>] : fine tuning of the parameters of the Cauchy lower bound (as the L(t,p,c) in paml4.2, see Inoue et al, 2009 for details).

-rp <age> : impose a prior mean age for root. If not specified, then, under the uniform and the Dirichlet priors, the root age is uniformly distributed (this is an improper prior, which can lead to problems, in particular when no upper bound is specified (which is in itself poor practice). Under the birth death prior, on the other hand, if no prior mean age is specified for the root, then the conditional distribution implied by the birth death prior is used (it is a proper, but apparently unstable, prior). Thus, in all cases, we advise the explicit use of this option. In addition, we strongly suggest to conduct runs under the prior (see below) to check the effective prior distribution of the age of the root.

Running the models under their prior:

-prior: desactivates all likelihood computations. As a result, the program samples from the prior of the model, such as specified by the command. This option works only for molecular dating studies, but it is an important one in this context. In general, the prior proposed by the user on the age of the root (using the -rp option) is the prior without calibrations, and can sometimes be very different from the prior conditional on the calibrations. Therefore, one should always run a MCMC under the prior, and check that the distribution of the root's age thus sampled by the MCMC is sufficiently wide (i.e. non informative), before proceeding with a posterior sampling.

Data recoding

Data recoding is the generalisation of the RY coding sometimes used for nucleotide sequences. It is meant for two things : first, recoding several biochemically similar amino-acids (or bases) as one single letter eliminates the potentially saturated signal brought about by repeated substitutions among those amino-acids (or bases). Second, it can also be used to remove possible compositional biases (RY coding, or for proteins, the case of Arginine and Lysine, whose relative frequency is very sensitive to the GC content).

Data recoding has not yet been used very often in practice. Theoretically speaking, it is not totally satisfactory: in principle, one would prefer a model that automatically accounts for higher substitution rates between equivalent amino-acids, or accomodates variations of the amino-acid composition along the lineages. In addition, there is no objective way to compare the "fit" of alternative recoding schemes.

For instance, comparing cross-validated likelihoods, or computing Bayes factors, between alternative recoding schemes would not make sense. But recoding can still be used as an exploratory device. In practice, it can sometimes yield interesting results (REF), albeit most often at the cost of a significant loss of phylogenetic signal.

You can recode the data by using the **-recode** command :

-recode <recoding_scheme>
or

-rec <recoding_scheme>

Three recoding schemes are available: dayhoff6, and hp. They are characterised as follows :

dayhoff6 (A,G,P,S,T) (D,E,N,Q) (H,K,R) (F,Y,W) (I,L,M,V) (C)
dayhoff4 (A,G,P,S,T) (D,E,N,Q) (H,K,R) (F,Y,W,I,L,M,V) (C= ?)
hp (A,C,F,G,I,L,M,V,W) (D,E,H,K,N,P,Q,R,S,T,Y)

You can also define your own recoding, by setting **<recoding_scheme>** equal to the name of a file containing a translation table formatted as follows:

<AA1> <Letter1>
<AA2> <Letter2>
..
<AA20> <Letter20>

You can propose '?' for a given amino-acid, in which case, it will be simply considered as equivalent to "no data" (as the cysteine, in the case of the dayhoff4 recoding).

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 100

defines a burn-in of 100

-x 100 10

a burn-in of 100, taking one every 10 trees, up to the end of each chain, and

-x 100 10 1000

a burn-in of 100, taking one every 10 trees, up to the 1000th point of the chains (or less, if the chains are shorter). If the chain is long enough, this implies a sample size of 90.

-o <basename>

outputs the results of the comparison in files with name **<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.

-ps

activates the postscript output (this requires LateX to be installed on your computer).

-v

make a 'verbose' account of the bipartitions in the **.bplist** file. This allows one to more easily spot which groups of taxa display discrepancies in their supports obtained across the runs.

-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, although strictly speaking, standard significance thresholding would imply considering only those supports above a standard value of 0.95 as significant.

readpb

-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 chain. The syntax is the same as for **bpcomp**.

-c <cutoff>

tunes the cutoff for the majority rule consensus. By default, the cutoff is equal to 0.5 (see **bpcomp**)

-r outputs the mean posterior rates at each site.

-rr outputs the mean posterior exchange rates.

-ss outputs the mean posterior profile at each site. with **-ps** : output the profiles as graphical logos.

-m outputs a histogram approximating the posterior distribution of the number of components.

-cl clustering of the profiles sampled during the MCMC: this is a single-link clustering, that has too parameters:

-md <dist> : the cutoff distance between profiles, that is, if the distance between two profiles is less than **<dist>**, they will end up in the same cluster, and

-ms <size> : the minimum size of the clusters to be displayed in the final output, given that all clusters of size smaller than **<size>** will be merged together into a "garbage cluster". Defaults are **<dist>** = 0.03 and **<size>** = 10.

-s save the subsample into a file. This can be useful if a chain is very big, and you want to subsample it to save disk space. Once a subsample has been saved, you can call **readpb** directly on it. For instance:

readpb -x 1000 100 11000 <chainname>

will save a sample of 100 points regularly spaced, taken from the chain **<chainname>**. This sample will be saved into a file called **<chainname>_sample.sample**. Then:

readpb -cl <chainname>_sample.sample

will perform a hierarchical clustering on this subsample.

-ps

activates the postscript output. If **readpb** is used for computing a consensus, a postscript tree will be produced (this requires LateX to be installed on your computer). If **readpb** is used for clustering modes, it will result in a postscript file displaying the logos of the profiles of the clusters.

readdiv

readdiv -x <burn-in> [<every> <until>] [other options] chainname

This program works like **readpb**. It will output several files:
<chainname>_sample.dates
<chainname>_sample.chronogram
<chainname>_sample.labelss

-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 chain.

-ps activates the postscript output.

How to use birth death and dirichlet priors on divergence times for calibrated analyses.

As mentioned earlier, if the hyperparameters of the birth death and the Dirichlet priors can be considered as free parameters in an analysis without calibration, they have to be fixed in a calibrated analysis (because of normalization problems). However, to which value to fix them is not a priori evident. A solution to this problem is to proceed in 3 steps:
(1) run an analysis without the calibration, and with the required prior, and with free hyperparameters;
(2) use this analysis to estimate these hyperparameters, and
(3) re-run an analysis, this time with the calibrations, and with the hyperparameters fixed to the empirical values just obtained.

To facilitate this procedure, **readdiv** outputs the posterior mean values of the hyperparameters of the prior (thus, of p1 and p2 for the birth death, or of chi for the Dirichlet prior), which are the natural Bayesian estimates that we need to obtain (step 2), in order to proceed with step 3.

bf

bf -cov <covname> [other options] <chainname>

This program computes Bayes factor using thermodynamic integration. It works only under the normal approximation. The integration consists in running a quasi-static chain from one model to the other, and then back (see Lartillot and Philippe, 2006). The program stops automatically, and a confidence interval for the logarithm of the Bayes factor is proposed in the **<chainname>.log** file. The confidence interval accounts for the thermic lag and the discretization error, but not for the sampling error, which however, should be negligible, at least for the long integrations.

By default, the integration is between the specified relaxed clock model and the deconstrained model. The relaxed clock model can be specified using the same option as for **pb** (it is the log normal model by default). For instance:

bf -cov moc10.oes -ln moclnbf

computes the Bayes factor between the log-normal and the deconstrained model, for the **moc10.oes** dataset.

Two other types of integration can be performed:

-prior

between the specified prior on divergence time (options to specify them are as in **pb**) and the uniform prior on divergence times.

-auto

between the two auto-correlated models (CIR and log-normal). A positive value means a support in favor of CIR over the log-normal model.

To control the precision (and thus, the duration) of the integration:

-short

burn-in of 100, integration over 1000 points (takes a few hours)

-long

burning of 1,000, integration over 10,000 points (takes about one day)

-vlong

burning of 1,000, integration over 10,000 points, but saving every 10 (takes more than one week)

tree2ps

tree2ps [options] <treefile>

This program converts a tree in newick format (**<treefile>**) into a postscript tree.

-o <output>

will output the postscript tree into **<output>.ps.**

-r <outgroup>

reroot the tree.

+l (-l)

with (without) branchlengths.

+p (-p)

with (without) posterior probabilities at nodes.

ppred

ppred -x <burn-in> [<every> <until>] [additional options] name

For each point of the sample, **ppred** simulates a replicate using the parameter values defined by this point. All replicates are written as standard datafiles. You can then evaluate any statistic of interest on them, as well as on the true original dataset, so as to compute the deviation between the observed and the mean value of *s* under the null distribution, the p-value (number of times the simulated value is more extreme than the observed value), the z-score, etc.

There are three pre-defined posterior predictive tests. They are called using the **-div**, **-comp** and **-sat** options, and measure the biochemical site-specificity, the compositional homogeneity, and the saturation, respectively. When using one of these options of **ppred**, the replicates are not written into files. Instead, the program directly computes the test statistics on them, and only outputs the z-score and p-values. In addition, a special posterior predictive testing framework is available for clock models (**-clockrate** option)

Biochemical specificity (option -div)

Under this option, the test statistic is the mean number of distinct amino acids observed at each column (the mean is taken over all the columns). This test statistic is meant as a way of measuring how well the site-specific biochemical patterns are accounted for by a model (see Lartillot et al, 2007). Often, standards models such as WAG, JTT or GTR are rejected by this statistic, except CAT, which should normally not be rejected, or very weakly.

Compositional homogeneity (option -comp)

This test statistic measures the compositional deviation of each taxon. The deviation is measured as the sum over the 20 amino-acids of the absolute differences between the taxon-specific and global empirical frequencies. The program also uses a global statistic, which is the maximum deviation over the taxa. This alleviates multiple testing, but on the other hand, only gives you a global answer (i.e. whether there are significant compositional deviations, but not who is significantly deviating).

Neither CAT, nor the standard WAG, JTT or GTR models, can account for the non-stationarity of the equilibrium frequencies. Therefore, and as you will probably see by yourself, in most cases, all these models are rejected by true data. For the moment, there is not yet any efficient model against this potential problem. But two workable solutions may be explored :
- removing the most deviating genes from your concatenation : this requires running the test on each gene separately, and sorting the genes, by decreasing z-score.
- alternatively, you may try to remove the most deviating taxa. Actually, there is an option to do this in **ppred** :

```
../ppred -comp -t 2 -x 100 10 rnapol1
```

will output a dataset where all taxa whose compositional z-score is greater than 2 have been removed. This dataset can then be used to make a new run.

Saturation test (option -sat)

In phylogenetics, saturation refers to the problem of multiple substitutions at a given site. Such multiple substitutions blur the phylogenetic signal. In addition, they create spurious convergences between unrelated taxa. Under challenging conditions (poor taxonomic sampling, or fast evolving species), these convergences can be mistaken for true phylogenetic signal, thereby creating systematic errors

A good statistical test, to see whether a model is likely to produce artefacts, is therefore to measure how well the model anticipates sequence saturation. If the model does not anticipate saturation correctly, then, it will be more 'naïve', i.e. more prone to mistakingly interpreting spurious convergences as true phylogenetic signal, and as a consequence, will more likely create artefacts (Lartillot et al., 2007).

The saturation index is defined as the number of homoplasies (convergences and reversions) per sites. However, even assuming the topology is known, convergences and reversions cannot be assessed just by looking at a dataset, but require the full substitution history to be reconstructed. To do this, one uses stochastic substitution mapping. Technically, the test thus proceeds as follows:

```
../ppred -sat -x 100 10 rnapol1
```

for each parameter value sampled from the posterior distribution, (i.e. each point of the MCMC run), two stochastic mappings are performed: one conditional on the true data at the leaves ('observed' mapping), and one unconstrained ('predicted'). The saturation index is then measured for both, and the procedure is repeated for each point of the run (here, 1 every 10). Finally, the observed and the predicted distributions of the saturation index are summarised by their means and variances. which are output by the program. Note that, since the 'observed' value of the statistic is itself a distribution, it is not possible to define a sensible p-value for that test. But by simply comparing the two confidence intervals offered by the two means, and their corresponding variances, you can get a good idea of how well the model handles saturation.The ppred program also makes two files containing the histograms of the two distributions. These histograms can then be jointly plotted using **gnuplot**, or some other program.

For instance, in the case of brpo.ali, the test was conducted under WAG:

```
observed homoplasy....:. 3.74290 +/- 0.0858103
posterior predictive..:. 3.32902 +/- 0.025057
```

and CAT:

```
observed homoplasy....:. 4.92162 +/- 0.219208
posterior predictive..:. 4.86584 +/- 0.44926
```

In the present case, it is clear that (1) WAG infers a much lower saturation of the data than CAT, and that (2) it predicts significantly less saturation than it observes. Both facts indicate that WAG tends to underestimates saturation, which in turn may cause systematic errors, or at least distorsions of the posterior probabilities. On the other hand, CAT seems to correctly account for the observed saturation.

Testing branch lengths induced by clock relaxation (option -clockrate)

Under this option, for each parameter value sampled from the posterior distribution the program produces two trees: one with the branch lengths specified by that specific parameter value (thus, branch lengths taken from the posterior distribution), and saved in the .postlength file, and one with posterior predictive branch lengths: that is, conditional on the current times and parameters, a complete rate history is simulated along the tree, and the branch lengths induced by this rate history are computed, and written in the .predlength file. In thi way, patterns of branch lengths induced by the model can be compared with the patterns directly inferred from the data.

6. Cross-Validation

Cross-validation (CV) is a very general and reliable method for comparing models. The rationale is as follows: the dataset is randomly split into two (unequal) parts, the learning set and the test set. The parameters of the model are estimated on the learning set (i.e. the model is 'trained' on the learning set), and these parameter values are then used to compute the likelihood of the test set (which measures how well the test set is 'predicted' by the model). The overall procedure has to be repeated (and the resulting log likelihood scores averaged) over several random splits.

CV is computationally heavy. Note also that it does not work if the continuous gamma or the Dirichlet process on the site-specific rates is activated: you have to be under the discrete gamma model, or the uniform rate model. On the other hand, CV can be easily parallelized, as each replicated random split can be processed independently. To achieve an optimal parallel processing, a series of distinct programs are proposed in PhyloBayes, corresponding to each step of the method:

- prepare the replicates, using **cvrep**;
- run each model under each replicated learning set, using **pb**;
- compute the cross-validated log likelihoods scores (i.e. the likelihood of each test set, averaged over the parameter values estimated by pb on the corresponding learning set), using **readcv**;
- pool all the cv-scores, and combine them into a global scoring of the models, using **sumcv**.

First, to prepare the replicated splits:

```
cvrep -nrep 10 -nfold 10 -d brpo.ali cvb
```

Here, **-nrep 10** means that we are asking for 10 replicates, and **-nfold 10** that we want to perform 10-fold cross validation: that is, for each replicate, the learning set will amount to 9/10, and the test set for 1/10, of the initial dataset. The replicates are saved under files using the specified prefix (here, **cvb**):

```
cvb0_learn.ali, cvb0_test.ali,
cvb1_learn.ali, cvb1_test.ali,
...
```

Next, you run a MCMC under each learning set, and for each model you want to compare, using **pb**. It is highly recommended to do these runs under a fixed topology (you can use the topology estimated by the model itself on the full dataset): this will be much faster. For instance, if you want to compare WAG and CAT:

```
pb -d cvb0_learn.ali -T brpo.tree -x 1 1100 CATcvb0_learn.ali
pb -d cvb0_learn.ali -T brpo.tree -x 1 1100 -wag WAGcvb0_learn.ali
```

and of course, the same should be done for the other 9 learning sets.

IMPORTANT: the name of the chains should be defined as here above, by prefixing the name of the dataset with some prefix standing for the model (e.g. **CATcvb0_learn.ali**). This is necessary for further processing of the resulting chains.

Once the chains have proceeded for all the replicates and all the models, you can compute, for each of

them, the cross-validated likelihoods, using **readcv**.

```
readcv -nrep 10 -x 100 1 CAT cvb
readcv -nrep 10 -x 100 1 WAG cvb
```

The cross-validation score, for each replicate, is the likelihood under the test set, averaged over the posterior distribution of the learning set. Here, it is approximated by averaging over the parameter values visited by the chain run on the learning set, discarding a burnin of 100 points, and taking every point thereafter. The logarithm of the resulting average is stored into a file with a **.cv** extension. There is one such file for each replicate (thus, **cvb0.cv**, **cvb1.cv**, ..., **cvb9.cv**).

Note that **readcv** will process each replicate successively, which may take a very long time. If you want to get a first rough idea of the score, do not hesitate to subsample (e.g. **-x 100 10**). Also, if you are working on a multiprocessor, **readcv** allows you to parallelize the likelihood computation procedure, in several ways. First, you can explicitely call **readcv** on only one replicate:

```
readcv -rep 2 -x 100 10 CAT cvb
```

This command will process only the replicate number 2. This allows you to split the work across several computers, according to what is most convenient for you. Second, you have the **-bg** (background) option, which will directly send the 10 jobs in parallel, all in the background mode.

```
readcv -bg -nrep 10 -x 100 10 CAT cvb
```

And if you are working on a 'qsub' grid, the following command will send the 10 jobs in the queue:

```
readcv -qsub -nrep 10 -x 100 10 CAT cvb
```

Finally, if you just want to prepare the 10 scripts, without launching them:

```
readcv -qprep -nrep 10 -x 100 10 CAT cvb
```

Once all the cross-likelihood scores, for all the models and all the replicates, have been obtained, you need to gather them, and compute summary statistics:

```
sumcv -nrep 10 WAG CAT cvb
```

You can use this command for more than 2 models (provided that the mean cv-likelihoods are available, and are all stored into the relevant files, with the .cv extension):

```
sumcv -nrep 10 WAG CAT GTR cvb
```

This last program will average the cv-log-likelihood scores over replicates, using the leftmost model (here WAG) as the reference.

7. Priors

Branch lengths

`-lexp`
bl ~ iid Exponential of mean μ

`-lgam`
bl ~ iid Gamma of mean μ and variance $\varepsilon\mu^2$ (reduces to the exponential distribution when $\varepsilon = 1$).

μ ~ Exponential of mean 0.1
ε ~ Exponential of mean 1

Rates across sites

`-cgam`
r ~ iid Gamma of mean 1 and variance $1/\alpha$

`-dgam <ncat>`
r ~ iid discretized Gamma of mean 1 and variance $1/\alpha$

`-ratecat`
r ~ Dirichlet process of granularity ζ and Base distribution Gamma of mean 1 and variance $1/\alpha$

α ~ Exponential of mean 1
ζ ~ Exponential of mean 1

Profiles

`-ncat 1`
π ~ Uniform

`-cat`
π ~ Dirichlet process of granularity η and base distribution G0

`-statflat`
G_0 = Uniform
`-statfree`
G_0 = generalized Dirichlet, of center π_0 and concentration δ

π_0 ~ Uniform
δ ~ Exponential of mean S (where S is the number of states, 4 for nucleotides, 20 for amino acids)
η ~ Exponential of mean 10.

Relative exchange rates

rr ~ iid Exponential of mean 1

Relaxed clock parameters

Rates are relative (i.e. dimensionless). They are multiplied by a global scaling factor, μ (expressed in number of substitutions per site and per unit of relative time). See Lepage et al for more details.

`-ln`
ρ ~ lognormal autocorrelated process of variance parameter σ

`-cir`
ρ ~ CIR process of variance parameter σ and autocorrelation parameter θ

`-ugam`
ρ ~ iid gamma of variance parameter σ

μ ~ Exponential of mean 1
σ ~ Exponential of mean 1
under CIR:
θ ~ Exponential of mean 1 with the additional constraint that $\theta > \sigma/2$

Divergence dates

Divergence dates are relative (dimensionless): root has divergence 1, and leaves divergence 0. These relative dates are then multiplied by a global scale factor *Scale* (expressed in Million of years) which is tune by the **`-rp`** (root prior) option.

`-uni`
t ~ uniform

`-dir`
t ~ Dirichlet prior of concentration parameter χ (when χ =1, prior is uniform).
χ ~ Exponential of mean 1

`-bd`
t ~ birth death process of parameters λ (birth rate), μ (death rate) and ω (sampling fraction). To avoid identifiability, we set $p_1=\lambda-\mu$, and $p_2=\lambda\omega$
p_1 ~ Exponential of mean 1
p_2 ~ Exponential of mean 1

Scale ~ Uniform over R^+ by default: this is a pseudo prior. Override if possible.

`-rp <meanrootage>`
Scale ~ Exponential of mean **`meanrootage`**.

Covarion model

The covarion parameters can be expressed as
rate from OFF to ON: $s_{01} = \xi . (1-p_0)$
rate from ON to OFF: $s_{10} = \xi . p_0$
where ξ is the rate of the ON-OFF process, and p0 the stationary probability of the OFF state.

$\xi \sim$ Exponential of mean 1
$p_0 \sim$ Uniform over [0,1]

Mixture of branch lengths

The mixture of branch length is emulated by using branch length multipliers λ_{ij}, for site i and branch j.
if we denote by Λ_i the vector of multipliers at site i (the λ_{ij}, for j=1..2P-3), then
$\Lambda \sim$ Dirichlet process of base distribution H_0, and granularity β
H_0 = product of independent gamma of mean 1 and variance $1/\phi$
$\phi \sim$ Exponential of mean 1
$\beta \sim$ Exponential of mean 1

Gene-specific branch length multipliers

If we denote by Δ_g the vector of multipliers for gene g
$\Delta \sim$ product of independent gamma of mean 1 and variance $1/\varphi$
$\varphi \sim$ Exponential of mean 1

References

Drummond A.J, Ho S.Y.W., Philipps M.J. and Rambaut, A. (2006). Relaxed phylogenetics and dating with confidence. ***PLOS Biology***, 4, 699-710.

Huelsenbeck, J.P. and Ronquist, F. (2001). MrBayes: Bayesian inference of phylogenetic trees. ***Bioinformatics***, 17, 754-755.

Huelsenbeck, J.P. and Suchard, M.A. (2007). A nonparametric method for accomodating and testing across-site rate variation. ***Systematic Biology***, 56, 975-987.

Inoue, J., Donoghue, P.C.J. and Yang, Z. (2009). The impact of the representation of fossil calibration on Bayesian estimation of species divergence times. ***Systematic Biology***, in press.

Kolaczkowski, B. and Thornton, J.W. (2008). A mixed branch length model of heterotachy improves phylogenetic accuracy. ***Molecular Biology and Evolution***, 25, 1054-1066.

Lartillot, N., Brinkmann, H. and Philippe, H. (2007). Suppression of long-branch attraction artefacts in the animal phylogeny using a site-heterogeneous model. ***BMC Evolutionary Biology***, 8, Supplement 1:S4.

Lartillot, N. (2007). Conjugate sampling for phylogenetic models. ***Journal of computational biology,*** 13, 43-63.

Lartillot, N. and Philippe, H. (2006). Computing Bayes factors using thermodynamic integration. ***Systematic Biology,*** 55, 195-207.

Lartillot, N. and Philippe, H. (2004). A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process. ***Molecular Biology and Evolution*** 21, 1095-1109.

Le, QS. and Gascuel, O. (2008). An improved general amino acid replacement matrix. ***Molecular Biology and Evolution,*** 25, 1307-1320.

Le, QS., Gascuel, O. and Lartillot, N.. (2008). Empirical profile mixture models for phylogenetic reconstruction. ***Bioinformatics,*** 24, 2316-2323.

Le, QS., Lartillot, N. and Gascuel, O. (2008b). Phylogenetic mixture models for proteins. ***Philosophical transactions of the Royal Society B*** (in press).

Lepage, T., Philippe, H., Bryant, D. and Lartillot, N. (2004). A general relaxed molecular clock model comparison. ***Molecular Biology and Evolution*** doi: 10.1093/molbev/msm193.

Rannala, B. and Yang, Z. (2007). Inferring speciation times under an episodic molecular clock. ***Systematic Biology***, 56, 453-466.

Thorne, J.L., Kishino, H., and Painter, I.S. (1998). Estimating the rate of evolution of the rate of molecular evolution. ***Molecular Biology and Evolution.***, 15, 1647-1657.

Wang, H.C., Li, K.M., Susko, E. and Roger, A.J. (2008). A class frequency mixture model that adjusts for site-specific amino-acid frequencies...***BMC Evolutionary Biology*** 8, 331.

Yang, Z. (2002). Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. ***Journal of Molecular Evolution***, 39, 306-314.

Yang, Z. (2006). Bayesian estimation of species divergence times under a molecular clock using multiple fossil calibrations with soft bounds. ***Molecular Biology Evolution***, 23, 212-226.

Zhou, Y., Rodrigue, N., Lartillot, N. and Philippe, H. (2007). Evaluation of models handling heterotachy in phylogenetic inference. ***BMC Evolutionary Biology***, 7, 206.