PhyML – Manual

Version 3.0
July 8, 2013

http://code.google.com/p/phyml

http://www.atgc-montpellier.fr/phyml

Contents

1 Availability

- Binaries: http://www.atgc-montpellier.fr/phyml
- Sources: http://code.google.com/p/phyml
- Discussion forum: http://groups.google.com/group/phyml-forum

2 Authors

- Stéphane Guindon and Olivier Gascuel conceived the original PhyML algorithm.
- Stéphane Guindon conceived the PhyTime method.
- Stéphane Guindon, Wim Hordjik and Olivier Gascuel conceived the SPR-based tree search algorithm.
- Maria Anisimova and Olivier Gascuel conceived the aLRT method for branch support.
- Stéphane Guindon, Franck Lethiec, Jean-Francois Dufayard and Vincent Lefort implemented PhyML.
- Jean-Francois Dufayard created the benchmark and implemented the tools that are used to check PhyML accuracy and performances.
- Vincent Lefort, Stéphane Guindon, Patrice Duroux and Olivier Gascuel conceived and implemented PhyML web server.
- Stéphane Guindon wrote this document.

3 Overview

PhyML [1] is a software package which primary task that is to estimate maximum likelihood phylogenies from alignments of nucleotide or amino-acid sequences. It provides a wide range of options that were designed to facilitate standard phylogenetic analyses. The main strength of PhyML lies in the large number of substitution models coupled to various options to search the space of phylogenetic tree topologies, going from very fast and efficient methods to slower but generally more accurate approaches. It also implements two methods to evaluate branch supports in a sound statistical framework (the non-parametric bootstrap and the approximate likelihood ratio test).

PhyML was designed to process moderate to large data sets. In theory, alignments with up to 4,000 sequences 2,000,000 character-long can analyzed. In practice however, the amount of memory required to process a data set is proportional of the product of the number of sequences by their length. Hence, a large number of sequences can only be processed provided that they are short. Also, PhyML can handle long sequences provided that they are not numerous. With most standard personal computers, the "comfort zone" for PhyML generally lies around 100-500 sequences less than 10,000 character long. For larger data sets, we recommend using other softwares such as RAxML [2] or GARLI [3] or Treefinder (http://www.treefinder.de).

4 Bug report

While PhyML is, of course, bug-free (!) (please read the disclaimer carefuly...), if you ever come across an issue, please feel free to report it using the discuss group web site at the following address: `https://groups.google.com/forum/?fromgroups#!forum/phyml-forum`. Alternatively, you can send an email to `s.guindon@auckland.ac.nz`. Do not forget to mention the version of PhyML and program options you are using.

5 Installing PhyML

5.1 Sources and compilation

The sources of the program are available free of charge by sending an e-mail to Stéphane Guindon at `guindon@lirmm.fr` or `s.guindon@auckland.ac.nz`. The compilation on UNIX-like systems is fairly standard. It is described in the 'INSTALL' file that comes with the sources. In a command-line window, go to the directory that contains the sources and type:

```
./configure;
make clean;
make V=0;
```

By default, PhyML will be compiled with optimization flags turned on. It is possible to generate a version of PhyML that can run through a debugging tool (such as `ddd`) or a profiling tool (such as `gprof`) using the following instructions:

```
./configure --enable-debug;
make clean;
make V=0;
```

5.2 Installing PhyML on UNIX-like systems (including Mac OS)

Copy PhyML binary file in the directory you like. For the operating system to be able to locate the program, this directory must be specified in the global variable `PATH`. In order to achieve this, you will have to add `export PATH="/your_path/:$PATH"` to the `.bashrc` or the `.bash_profile` located in your home directory (`your_path` is the path to the directory that contains PhyML binary).

5.3 Installing PhyML on Microsoft Windows

Copy the files `phyml.exe` and `phyml.bat` in the same directory. To launch PhyML, click on the icon corresponding to `phyml.bat`. Clicking on the icon for `phyml.exe` works too but the dimensions of the window will not fit PhyML PHYLIP-like interface.

5.4 Installing the parallel version of PhyML

Bootstrap analysis can run on multiple processors. Each processor analyses one bootstraped dataset. Therefore, the computing time needed to perform R bootstrap replicates is divided by the number of processors available.

This feature of PhyML relies on the MPI (Message Passing Interface) library. To use it, your computer must have MPI installed on it. In case MPI is not installed, you can dowload it from http://www.mcs.anl.gov/research/projects/mpich2/. Once MPI is installed, it is necessary to launch the MPI daemon. This can be done by entering the following instruction: `mpd &`. Note however that in most cases, the MPI daemon will already be running on your server so that you most likely do not need to worry about this. You can then just go in the `phyml/` directory (the directory that contains the `src/`, `examples/` and `doc/` folders) and enter the commands below:

```
./configure --enable-mpi;
make clean;
make;
```

A binary file named `phyml-mpi` has now been created in the `src/` directory and is ready to use with MPI. A typical MPI command-line which uses 4 CPUs is given below:

```
mpirun -n 4 ./phyml-mpi -i myseq -b 100
```

Please read section 6.4 of this document for more information.

6 Program usage.

PhyML has three distinct user-interfaces. The first corresponds to a PHYLIP-like text interface that makes the choice of the options self-explanatory. The command-line interface is well-suited for people that are familiar with PhyML options or for running PhyML in batch mode. The XML interface is more sophisticated. It allows the user to analyse partitionned data using flexible mixture models of evolution.

6.1 PHYLIP-like interface

The default is to use the PHYLIP-like text interface by simply typing '`phyml`' in a command-line window or by clicking on the PhyML icon (see Section 5.3). After entering the name of the input sequence file, a list of sub-menus helps the users set up the analysis. There are currently four distinct sub-menus:

1. *Input Data*: specify whether the input file contains amino-acid or nucleotide sequences. What the sequence format is (see Section 7) and how many data sets should be analysed.

2. *Substitution Model*: selection of the Markov model of substitution.

3. *Tree Searching*: selection of the tree topology searching algorithm.

4. *Branch Support*: selection of the method that is used to measure branch support.

'`+`' and '`-`' keys are used to move forward and backward in the sub-menu list. Once the model parameters have been defined, typing '`Y`' (or '`y`') launches the calculations. The meaning of some options may not be obvious to users that are not familiar with phylogenetics. In such situation, we strongly recommend to use the default options. As long as the format of the input sequence file is correctly specified (sub-menu *Input data*), the safest option for non-expert users is to use the default settings. The different options provided within each sub-menu are described in what follows.

6.1.1 Input Data sub-menu

```
[D] ............................... Data type (DNA/AA)
```

Type of data in the input file. It can be either DNA or amino-acid sequences in PHYLIP format (see Section 7). Type `D` to change settings.

```
[I] ...... Input sequences interleaved (or sequential)
```

PHYLIP format comes in two flavours: interleaved or sequential (see Section 7). Type `I` to selected among the two formats.

```
[M] ....................... Analyze multiple data sets
```

If the input sequence file contains more than one data sets, PhyML can analyse each of them in a single run of the program. Type `M` to change settings.

```
[R] ............................................ Run ID
```

This option allows you to append a string that identifies the current PhyML run. Say for instance that you want to analyse the same data set with two models. You can then 'tag' the first PhyML run with the name of the first model while the second run is tagged with the name of the second model.

6.1.2 Substitution model sub-menu

```
[M] ................ Model of nucleotide substitution
```

```
[M] ............... Model of amino-acids substitution
```

PhyML implements a wide range of substitution models: JC69 [4], K80 [5], F81 [6], F84 [7], HKY85 [8], TN93 [9] GTR [10, 11] and custom for nucleotides; LG [12], WAG [13], Dayhoff [14], JTT [15], Blosum62 [16], mtREV [17], rtREV [18], cpREV [19], DCMut [20], VT [21] and mt-MAM [22] and custom for amino acids. Cycle through the list of nucleotide or amino-acids substitution models by typing `M`. Both nucleotide and amino-acid lists include a 'custom' model. The custom option provides the most flexible way to specify the nucleotide substitution model. The model is defined by a string made of six digits. The default string is '`000000`', which means that the six relative rates of nucleotide changes: $A \leftrightarrow C$, $A \leftrightarrow G$, $A \leftrightarrow T$, $C \leftrightarrow G$, $C \leftrightarrow T$ and $G \leftrightarrow T$, are equal. The string '`010010`' indicates that the rates $A \leftrightarrow G$ and $C \leftrightarrow T$ are equal and distinct from $A \leftrightarrow C = A \leftrightarrow T = C \leftrightarrow G = G \leftrightarrow T$. This model corresponds to HKY85 (default) or K80 if the nucleotide frequencies are all set to 0.25. '`010020`' and '`012345`' correspond to TN93 and GTR models respectively. The digit string therefore defines groups of relative substitution rates. The initial rate within each group is set to 1.0, which corresponds to F81 (JC69 if the base frequencies are equal). Users also have the opportunity to define their own initial rate values. These rates are then optimised afterwards (option '`O`') or fixed to their initial values. The custom option can be used to implement all substitution models that are special cases of GTR. Table 1 on page 15 gives the correspondence between the 'standard' name of the model (see http://mbe.oxfordjournals.org/content/18/6/897/F2.large.jpg) and the custom model code. The custom model also exists for protein sequences. It is useful when one wants to use an amino-acid substitution model that is not hard-coded in PhyML. The symmetric part of the rate matrix, as well as the equilibrium amino-acid frequencies, are given in a file which name is given as input of the program. The format of this file is described in the section 7.4.

```
[F] ................ Optimise equilibrium frequencies
```

```
[E] ........ Equilibrium frequencies (empirical/user)
```

`[F] . Amino acid frequencies (empirical/model defined)`

For nucleotide sequences, optimising nucleotide frequencies means that the values of these parameters are estimated in the maximum likelihood framework. When the custom model option is selected, it is also possible to give the program a user-defined nucleotide frequency distribution at equilibrium (option `E`). For protein sequences, the stationary amino-acid frequencies are either those defined by the substitution model or those estimated by counting the number of different amino-acids observed in the data. Hence, the meaning of the `F` option depends on the type of the data to be processed.

`[T] .................... Ts/tv ratio (fixed/estimated)`

Fix or estimate the transition/transversion ratio in the maximum likelihood framework. This option is only available when DNA sequences are to be analysed under K80, HKY85 or TN93 models. The definition given to this parameter by PhyML is the same as PAML's one. Therefore, the value of this parameter does *not* correspond to the ratio between the expected number of transitions and the expected number of transversions during a unit of time. This last definition is the one used in PHYLIP. PAML's manual gives more detail about the distinction between the two definitions (`http://abacus.gene.ucl.ac.uk/software/paml.html`).

`[V] . Proportion of invariable sites (fixed/estimated)`

The proportion of invariable sites, i.e., the expected frequency of sites that do not evolve, can be fixed or estimated. The default is to fix this proportion to 0.0. By doing so, we consider that each site in the sequence may accumulate substitutions at some point during its evolution, even if no differences across sequences are actually observed at that site. Users can also fix this parameter to any value in the $[0.0, 1.0]$ range or estimate it from the data in the maximum-likelihood framework.

`[R] ....... One category of substitution rate (yes/no)`

`[C] ........... Number of substitution rate categories`

`[A] ... Gamma distribution parameter (fixed/estimated)`

```
[G] .........‘Middle’ of each rate class (mean/median)
```

Rates of evolution often vary from site to site. This heterogeneity can be modelled using a discrete gamma distribution. Type `R` to switch this option on or off. The different categories of this discrete distribution correspond to different (relative) rates of evolution. The number of categories of this distribution is set to 4 by default. It is probably not wise to go below this number. Larger values are generally preferred. However, the computational burden involved is proportional to the number of categories (i.e., an analysis with 8 categories will generally take twice the time of the same analysis with only 4 categories). Note that the likelihood will not necessarily increase as the number of categories increases. Hence, the number of categories should be kept below a "reasonable" number, say 20. The default number of categories can be changed by typing `C`.

The middle of each discretized substitution rate class can be determined using the mean or the median. PAML, MrBayes and RAxML use the mean. However, the median is generally associated with greater likelihoods than the mean. This conclusion is based on our analysis of several real-world data sets extracted from TreeBase. Despite this, the default option in PhyML is to use the mean in order to make PhyML likelihoods comparable to those of other phylogenetic software. One must bare in mind that likelihoods calculated with the mean approximation are not directly comparable to the likelihoods calculated using the median approximation.

The shape of the gamma distribution determines the range of rate variation across sites. Small values, typically in the $[0.1, 1.0]$ range, correspond to large variability. Larger values correspond to moderate to low heterogeneity. The gamma shape parameter can be fixed by the user or estimated via maximum-likelihood. Type `A` to select one or the other option.

6.1.3 Tree searching sub-menu

```
[O] ...........................  Optimise tree topology
```

By default the tree topology is optimised in order to maximise the likelihood. However, it is also possible to avoid any topological alteration. This option is useful when one wants to compute the likelihood of a tree given as input (see below). Type `O` to select among these two options.

```
[S] ..................  Tree topology search operations
```

PhyML proposes three different methods to estimate tree topologies. The default approach is to use simultaneous NNI. This option corresponds to the original PhyML algorithm [1]. The second approach relies on subtree pruning and regrafting (SPR). It generally finds better tree topologies compared to NNI but is also significantly slower. The third approach, termed BEST, simply estimates the phylogeny using both methods and returns the best solution among the two. Type `S` to choose among these three choices.

`[R] ......................... Use random starting tree`

`[N] .................. Number of random starting trees`

When the SPR or the BEST options are selected, is is possible to use random trees rather than BioNJ or a user-defined tree, as starting tree. If this option is turned on (type `R` to change), five trees, corresponding to five random starts, will be estimated. The output tree file will contain the best tree found among those five. The number of random starts can be modified by typing `N`. Setting the number of random starting trees to N means that the analysis will take (slightly more than) N times the time required for a standard analysis where only one (BioNJ) starting tree is used. However, the analysis of real data sets shows that the best trees estimated using the random start option almost systematically have higher likelihoods than those inferred using a single starting tree.

`[U] ........ Starting tree (BioNJ/parsimony/user tree)`

When the tree topology optimisation option is turned on, PhyML proceeds by refining an input tree. By default, this input tree is estimated using BioNJ [23]. The alternative option is to use a parsimony tree. We found this option specially useful when analysing large data sets with NNI moves as it generally leads to greater likelihoods than those obtained when starting from a BioNJ trees. The user can also to input her/his own tree. This tree should be in Newick format (see Section 7). This option is useful when one wants to evaluate the likelihood of a given tree with a fixed topology, using PhyML. Type `U` to choose among these two options.

6.1.4 Branch support sub-menu

`[B] ................ Non parametric bootstrap analysis`

The support of the data for each internal branch of the phylogeny can be estimated using non-parametric bootstrap. By default, this option is switched off. Typing `B` switches on the bootstrap analysis. The user is then prompted for a number of bootstrap replicates. The largest this number the more precise the bootstrap support estimates are. However, for each bootstrap replicate a phylogeny is estimated. Hence, the time needed to analyse N bootstrap replicates corresponds to N-times the time spent on the analysis of the original data set. $N = 100$ is generally considered as a reasonable number of replicates.

```
[A] ................ Approximate likelihood ratio test
```

When the bootstrap option is switched off (see above), approximate likelihood branch supports are estimated. This approach is considerably faster than the bootstrap one. However, both methods intend to estimate different quantities and conducting a fair comparison between both criteria is not straightforward. The estimation of approximate likelihood branch support comes in multiple flavours. The default is set to aBayes, corresponding to the approximate Bayes method described in [24]. The approximate likelihood ratio test (aLRT) [25], ShimodairaHasegawa aLRT (SH-aLRT) statistics are the other available options.

6.2 Command-line interface

An alternative to the PHYLIP-like interface is the command-line interface. Users that do not need to modify the default parameters can launch the program with the '`phyml -i seq_file_name`' command. The list of all command line arguments and how to use them is given in the 'Help' section which is displayed when entering the '`phyml --help`' command. The available command-line options are described in what follows.

- `-i` (or `--input`) `seq_file_name`
  `seq_file_name` is the name of the nucleotide or amino-acid sequence file in PHYLIP format.

- `-d` (or `--datatype`) `data_type`
  `data_type` is `nt` for nucleotide (default) and `aa` for amino-acid sequences.

- `-q` (or `--sequential`)
  Changes interleaved format (default) to sequential format.

- `-n` (or `--multiple`) `nb_data_sets`
  `nb_data_sets` is an integer giving the number of data sets to analyse.

- `-p` (or `--pars`)
  Use a minimum parsimony starting tree. This option is taken into account when the '-u' option is absent and when tree topology modifications are to be done.

- `-b` (or `--bootstrap`) `int`

  - `int` > 0: `int` is the number of bootstrap replicates.
  - `int` $= 0$: neither approximate likelihood ratio test nor bootstrap values are computed.
  - `int` = -1: approximate likelihood ratio test returning aLRT statistics.
  - `int` = -2: approximate likelihood ratio test returning Chi2-based parametric branch supports.
  - `int` = -4: SH-like branch supports alone.
  - `int` = -5: (default) approximate Bayes branch supports.

- `-m` (or `--model`) `model_name`
  `model_name` : substitution model name.

  - *Nucleotide-based models*: `HKY85` (default) `| JC69 | K80 | F81 | F84 | TN93 | GTR | custom`
    The `custom` option can be used to define a new substitution model. A string of six digits identifies the model. For instance, 000000 corresponds to F81 (or JC69 provided the distribution of nucleotide frequencies is uniform). 012345 corresponds to GTR. This option can be used for encoding any model that is a nested within GTR. See Section 6.1.2 and Table 1. *NOTE:* the substitution parameters of the custom model will be optimised so as to maximise the likelihood. It is possible to specify and fix (i.e., avoid optimisation) the values of the substitution rates only through the PHYLIP-like interface.
  - *Amino-acid based models*: `LG` (default) `| WAG | JTT | MtREV | Dayhoff | DCMut | RtREV | CpREV | VT | Blosum62 | MtMam | MtArt | HIVw | HIVb | custom`
    The `custom` option is useful when one wants to use an amino-acid substitution model that is not available by default in PhyML.

Name	Command-line option
JC69	`-m 000000 -f 0.25,0.25,0.25,0.25`
F81	`-m 000000`
K80	`-m 010010 -f 0.25,0.25,0.25,0.25`
HKY85	`-m 010010`
TrNef	`-m 010020 -f 0.25,0.25,0.25,0.25`
TrN	`-m 010020`
K81	`-m 123321 -f 0.25,0.25,0.25,0.25`
K81uf	`-m 123321`
TIMef	`-m 132241 -f 0.25,0.25,0.25,0.25`
TIM	`-m 132241`
TVMef	`-m 102304 -f 0.25,0.25,0.25,0.25`
TVM	`-m 102304`
SYM	`-m 123456 -f 0.25,0.25,0.25,0.25`
GTR	`-m 123456`

Table 1. Nucleotide substitution model names (as defined in [26]) and the corresponding custom model code used in PhyML.

The symmetric part of the rate matrix, as well as the equilibrium amino-acid frequencies, are given in a file which name is asked for by the program. The format of this file is described in section 7.4.

- `--aa_rate_file file_name`
  This option is compulsory when analysing amino-acid sequences under a 'custom' model (see above). `file_name` should provide a rate matrix and equilibrium amino acid in PAML format (see Section 7.4).

- `-f e`, `m`, or "`fA,fC,fG,fT`"
  Nucleotide or amino-acid frequencies.
  - `e` : the character frequencies are determined as follows :
    * *Nucleotide sequences*: (Empirical) the equilibrium base frequencies are estimated by counting the occurence of the different bases in the alignment.
    * *Amino-acid sequences*: (Empirical) the equilibrium amino-acid frequencies are estimated by counting the occurence of the different amino-acids in the alignment.
  - `m` : the character frequencies are determined as follows :

    * *Nucleotide sequences*: (ML) the equilibrium base frequencies are estimated using maximum likelihood.
    * *Amino-acid sequences*: (Model) the equilibrium amino-acid frequencies are estimated using the frequencies defined by the substitution model.
  - "`fA,fC,fG,fT`": only valid for nucleotide-based models. `fA`, `fC`, `fG` and `fT` are floating numbers that correspond to the frequencies of A, C, G and T respectively.

- `-t` (or `--ts/tv`) `ts/tv_ratio`
  `ts/tv_ratio`: transition/transversion ratio. DNA sequences only. Can be a fixed positive value (e.g., 4.0) or type `e` to get the maximum likelihood estimate.

- `-v` (or `--pinv`) `prop_invar`
  `prop_invar`: proportion of invariable sites. Can be a fixed value in the [0,1] range or type `e` to get the maximum likelihood estimate.

- `-c` (or `--nclasses`) `nb_subst_cat`
  `nb_subst_cat`: number of relative substitution rate categories. Default: `nb_subst_cat=4`. Must be a positive integer.

- `-a` (or `--alpha`) `gamma`
  `gamma`: value of the gamma shape parameter. Can be a fixed positive value or e to get the maximum likelihood estimate. The value of this parameter is estimated in the maximum likelihood framework by default.

- `--use_median`
  The middle of each substitution rate class in the discrete gamma distribution is taken as the median. The mean is used by default.

- `--free_rates`
  As an alternative to the discrete gamma model, it is possible to estimate the (relative) rate in each class of the (mixture) model and the corresponding frequencies directly from the data. This model, called the FreeRate model, has more parameters than the discrete gamma one but usually provides a significantly better fit to the data. See [27] for more information about this model and an illustration of its use.

- `--codpos 1,2 or 3`
  When analysing an alignment of coding sequences, use this option to

consider only the first, second or the third coding position for the estimation.

- `-s` (or `--search`) `move`
  Tree topology search operation option. Can be either `NNI` (default, fast) or `SPR` (usually slower than `NNI` but more accurate) or `BEST` (best of NNI and SPR search).

- `-u` (or `--inputtree`) `user_tree_file`
  `user_tree_file`: starting tree filename. The tree must be in Newick format.

- `-o params`
  This option focuses on specific parameter optimisation.

  - `params=tlr`: tree topology (`t`), branch length (`l`) and substitution rate parameters (`r`) are optimised.
  - `params=tl`: tree topology and branch lengths are optimised.
  - `params=lr`: branch lengths and substitution rate parameters are optimised.
  - `params=l`: branch lengths are optimised.
  - `params=r`: substitution rate parameters are optimised.
  - `params=n`: no parameter is optimised.

- `--rand_start`
  This option sets the initial tree to random. It is only valid if SPR searches are to be performed.

- `--n_rand_starts num`
  `num` is the number of initial random trees to be used. It is only valid if SPR searches are to be performed.

- `--r_seed num`
  `num` is the seed used to initiate the random number generator. Must be an integer.

- `--print_site_lnl`
  Print the likelihood for each site in file *_phyml_lk.txt. For Γ or Γ+I or FreeRate models, this option returns the posterior probability of each relative rate class at each site. Such information can then be used to identify fast- and slow-evolving regions of the alignment.

- `--print_trace`
  Print each phylogeny explored during the tree search process in file *_phyml_trace.txt. This option can be useful for monitoring the progress of the analysis for very large data sets and have an approximate idea of what the final phylogeny will look like.

- `--run_id ID_string`
  Append the string ID_string at the end of each PhyML output file. This option may be useful when running simulations involving PhyML. It can also be used to 'tag' multiple analysis of the same data set with various program settings.

- `--no_memory_check`
  By default, when processing a large data set, PhyML will pause and ask the user to confirm that she/he wants to continue with the execution of the analysis despite the large amount of memory required. The `--no_memory_check` skips this question. It is especially useful when running PhyML in batch mode.

- `--no_colalias`
  By default, PhyML preprocesses each alignment by putting together (or aliasing) the columns that are identical. Use this option to skip this step but be aware that the analysis might then take more time to complete.

- `--constrained_lens`
  When an input tree with branch lengths is provided, this option will find the branch multiplier that maximises the likelihood (i.e., the relative branch lengths remain constant)

- `--constraint_file file_name`
  `file_name` lists the topological constraints under which the tree topology search is conducted. This option should be used in conjunction with `-u file_name`. See Section 7.5 for more information.

- `--quiet`
  Runs PhyML in quiet mode. The program will not pause if the memory required to run the analysis exceeds 256MB and will not output the log-likelihood score to the output.

6.3 XML interface

- `--xml=xml_file_name`
  `xml_file_name` is the name of the XML file containing the information required to run the analysis. More details about this type of file is given in the section 8.

6.4 Parallel bootstrap

Bootstrapping is a highly parallelizable task. Indeed, bootstrap replicates are independent from one another. Each bootstrap replicate can then be analysed separately. Modern computers often have more than one CPU. Each CPU can therefore be used to process a bootstrap sample. Using this parallel strategy, performing R bootstrap replicates on C CPUs 'costs' the same amount of computation time as processing $R \times C$ bootstrap replicates on a single CPU. In other words, for a given number of replicates, the computation time is divided by R compared to the non-parallel approach.

PhyML sources must be compiled with specific options to turn on the parallel option (see Section 5.4). Once the binary file (`phyml`) has been generated, running a bootstrap analysis with, say 100 replicates on 2 CPUs, can be done by typing the following command-line:

```
mpd &;
mpirun -np 2 ./phyml -i seqfile -b 100;
```

The first command launches the mpi daemon while the second launches the analysis. Note that launching the daemon needs to be done only once. The output files are similar to the ones generated using the standard, non-parallel, analysis (see Section 7). Note that running the program in batch mode, i.e.:

```
mpirun -np 2 ./phyml -i seqfile -b 100 &
```

will probably NOT work. I do not know how to run a mpi process in batch mode yet. Suggestions welcome... Also, at the moment, the number of bootstrap replicates must be a multiple of the number of CPUs required in the mpirun command.

7 Inputs & outputs for command-line and PHYLIP interface

PhyML reads data from standard text files, without the need for any particular file name extension.

```
PHYLIP interleaved

5 80
seq1   CCATCTCACGGTCGGTACGATACACCKGCTTTTGGCAGGAAATGGTCAATATTACAAGGT
seq2   CCATCTCACGGTCAG---GATACACCKGCTTTTGGCGGGAAATGGTCAACATTAAAAGAT
seq3   RCATCTCCCGCTCAG---GATACCCCKGCTGTTG????????????????ATTAAAAGGT
seq4   RCATCTCATGGTCAA---GATACTCCTGCTTTTGGCGGGAAATGGTCAATCTTAAAAGGT
seq5   RCATCTCACGGTCGGTAAGATACACCTGCTTTTGGCGGGAAATGGTCAAT????????GT

ATCKGCTTTTGGCAGGAAAT
ATCKGCTTTTGGCGGGAAAT
AGCKGCTGTTG?????????
ATCTGCTTTTGGCGGGAAAT
ATCTGCTTTTGGCGGGAAAT
```

```
PHYLIP sequential

5 40
seq1   CCATCTCANNNNNNNNNACGATACACCKGCTTTTGGCAGG
seq2   CCATCTCANNNNNNNNNGGGATACACCKGCTTTTGGCGGG
seq3   RCATCTCCCGCTCAGTGAGATACCCCKGCTGTTGXXXXX
seq4   RCATCTCATGGTCAATG-AATACTCCTGCTTTTGXXXXX
seq5   RCATCTCACGGTCGGTAAGATACACCTGCTTTTGxxxxx
```

Figure 1. **PHYLIP interleaved and sequential formats.**

7.1 Sequence formats

Alignments of DNA or protein sequences must be in PHYLIP or NEXUS [28] sequential or interleaved format (Figures 7.1 and 2). For PHYLIP formated sequence alignments, the first line of the input file contains the number of species and the number of characters, in free format, separated by blank characters. One slight difference with PHYLIP format deals with sequence name lengths. While PHYLIP format limits this length to ten characters, PhyML can read up to hundred character long sequence names. Blanks and the symbols "(),:" are not allowed within sequence names because the Newick tree format makes special use of these symbols. Another slight difference with PHYLIP format is that actual sequences must be separated from their names by at least one blank character.

A PHYLIP input sequence file may also display more than a single data set. Each of these data sets must be in PHYLIP format and two successive alignments must be separated by an empty line. Processing multiple data sets requires to toggle the '`M`' option in the *Input Data* sub-menu or use the '`-n`' command line option and enter the number of data sets to analyse. The multiple data set option can be used to process re-sampled data that were generated using a non-parametric procedure such as cross-validation or jackknife (a bootstrap option is already included in PhyML). This option is also useful in multiple gene studies, even if fitting the same substitution

Nexus nucleotides

```
[ This is a comment ]
#NEXUS
BEGIN DATA;
DIMENSIONS NTAX=10 NCHAR=20;
FORMAT DATATYPE=DNA;
MATRIX
tax1       ?ATGATTTCCTTAGTAGCGG
tax2       CAGGATTTCCTTAGTAGCGG
tax3       ?AGGATTTCCTTAGTAGCGG
tax4       ?????????????GTAGCGG
tax5       CAGGATTTCCTTAGTAGCGG
tax6       CAGGATTTCCTTAGTAGCGG
tax7       ???GATTTCCTTAGTAGCGG
tax8       ????????????????????
tax9       ???GGATTTCTTCGTAGCGG
tax10      ???????????????AGCGG;
END;
```

Nexus digits

```
[ This is a comment ]
#NEXUS
BEGIN DATA;
DIMENSIONS NTAX=10 NCHAR=20;
FORMAT DATATYPE=STANDARD SYMBOLS="0 1 2 3";
MATRIX
tax1       ?0320333113302302122
tax2       10220333113302302122
tax3       ?0220333113302302122
tax4       ?????????????2302122
tax5       10220333113302302122
tax6       10220333113302302122
tax7       ???20333113302302122
tax8       ????????????????????
tax9       ???22033313312302122
tax10      ???????????????02122;
END;
```

Nexus digits

```
[ This is a comment ]
#NEXUS
BEGIN DATA;
DIMENSIONS NTAX=10 NCHAR=20;
FORMAT DATATYPE=STANDARD SYMBOLS="00 01 02 03";
MATRIX
tax1       ??00030200030303010103030002030002010202
tax2       0100020200030303010103030002030002010202
tax3       ??00020200030303010103030002030002010202
tax4       ??????????????????????????02030002010202
tax5       0100020200030303010103030002030002010202
tax6       0100020200030303010103030002030002010202
tax7       ??????0200030303010103030002030002010202
tax8       ????????????????????????????????????????
tax9       ??????0202000303030103030102030002010202
tax10      ??????????????????????????????0002010202;
END;
```

Figure 2. **NEXUS formats.**

model to all data sets may not be suitable.

PhyML can also process alignments in NEXUS format. Although not all the options provided by this format are supported by PhyML, a few specific features are exploited. Of course, this format can handle nucleotide and protein sequence alignments in sequential or interleaved format. It is also possible to use custom alphabets, replacing the standard 4-state and 20-state alphabets for nucleotides and amino-acids respectively. Examples of a 4-state custom alphabet are given in Figure 2. Each state must here correspond to one digit or more. The set of states must be a list of consecutive digits starting from 0. For instance, the list "0, 1, 3, 4" is not a valid alphabet. Each state in the symbol list must be separated from the next one by a space. Hence, alphabets with large number of states can be easily defined by using two-digit number (starting with 00 up to 19 for a 20 state alphabet). Most importantly, this feature gives the opportunity to analyse data sets made of presence/absence character states (use the `symbols=''0 1''` option for such data). Alignments made of custom-defined states will be processed using the Jukes and Cantor model. Other options of the program (e.g., number of rate classes, tree topology search algorithm) are freely configurable. Note that, at the moment, the maximum number of different states is set to 22 in order to save memory space. It is however possible to lift this threshold by modifiying the value of the variable `T_MAX_ALPHABET` in the file '`utilities.h`'. The program will then have to be re-compiled.

7.1.1 Gaps and ambiguous characters

Gaps correspond to the '`-`' symbol. They are systematically treated as unknown characters "on the grounds that we don't know what would be there if something were there" (J. Felsenstein, PHYLIP main documentation). The likelihood at these sites is summed over all the possible states (i.e., nucleotides or amino acids) that could actually be observed at these particular positions. Note however that columns of the alignment that display only gaps or unknown characters are simply discarded because they do not carry any phylogenetic information (they are equally well explained by any model). PhyML also handles ambiguous characters such as R for A or G (purines) and Y for C or T (pyrimidines). Tables 2 and 3 give the list of valid characters/symbols and the corresponding nucleotides or amino acids.

7.1.2 Specifying outgroup sequences

PhyML can return rooted trees provided outgroup taxa are identified from the sequence file. In order to do so, sequence names that display a '*' char-

Character	Nucleotide	Character	Nucleotide
A	Adenosine	Y	C or T
G	Guanine	K	G or T
C	Cytosine	B	C or G or T
T	Thymine	D	A or G or T
U	Uracil (=T)	H	A or C or T
M	A or C	V	A or C or G
R	A or G	$-$ or N or X or ?	unknown
W	A or T		(=A or C or G or T)
S	C or G		

Table 2. **List of valid characters in DNA sequences and the corresponding nucleotides.**

Character	Amino-Acid	Character	Amino-Acid
A	Alanine	L	Leucine
R	Arginine	K	Lysine
N or B	Asparagine	M	Methionine
D	Aspartic acid	F	Phenylalanine
C	Cysteine	P	Proline
Q or Z	Glutamine	S	Serine
E	Glutamic acid	T	Threonine
G	Glycine	W	Tryptophan
H	Histidine	Y	Tyrosine
I	Isoleucine	V	Valine
L	Leucine	$-$ or X or ?	unknown
K	Lysine		(can be any amino acid)

Table 3. **List of valid characters in protein sequences and the corresponding amino acids.**

acter will be automatically considered as belonging to the outgroup.

The topology of the rooted tree is exactly the same as the unrooted version of the same tree. In other words, PhyML first ignores the distinction between ingroup and outgroup sequences, builds a maximum likelihood unrooted tree and then tries to add the root. If the outgroup has more than one sequence, the position of the root might be ambiguous. In such situation, PhyML tries to identify the most relevant position of the root by considering which edge provides the best separation between ingroup and outgroup taxa (i.e., we are trying to make the outgroup "as monophyletic as possible").

7.2 Tree format

PhyML can read one or several phylogenetic trees from an input file. This option is accessible through the *Tree Searching* sub menu or the '`-u`' argument from the command line. Input trees are generally used as initial maximum likelihood estimates to be subsequently adjusted by the tree searching algorithm. Trees can be either rooted or unrooted and multifurcations are allowed. Taxa names must, of course, match the corresponding sequence names.

```
((seq1:0.03,seq2:0.01):0.04,(seq3:0.01,(seq4:0.2,seq5:0.05):0.2):0.01);
((seq3,seq2),seq1,(seq4,seq5));
```

Figure 3. **Input trees**. The first tree (top) is rooted and has branch lengths. The second tree (bottom) is unrooted and does not have branch lengths.

7.3 Multiple alignments and trees

Single or multiple sequence data sets may be used in combination with single or multiple input trees. When the number of data sets is one ($n_D = 1$) and there is only one input tree ($n_T = 1$), then this tree is simply used as input for the single data set analysis. When $n_D = 1$ and $n_T > 1$, each input tree is used successively for the analysis of the single alignment. PhyML then outputs the tree with the highest likelihood. If $n_D > 1$ and $n_T = 1$, the same input tree is used for the analysis of each data set. The last combination is $n_D > 1$ and $n_T > 1$. In this situation, the i-th tree in the input tree file is used to analyse the i-th data set. Hence, n_D and n_T must be equal here.

7.4 Custom amino-acid rate model

The custom amino-acid model of substitutions can be used to implement a model that is not hard-coded in PhyML. This model must be time-reversible. Hence, the matrix of substitution rates is symmetrical. The format of the rate matrix with the associated stationary frequencies is identical to the one used in PAML. An example is given below:

```
0.55
0.51 0.64
0.74 0.15 5.43
1.03 0.53 0.27 0.03
0.91 3.04 1.54 0.62 0.10
1.58 0.44 0.95 6.17 0.02 5.47
1.42 0.58 1.13 0.87 0.31 0.33 0.57
0.32 2.14 3.96 0.93 0.25 4.29 0.57 0.25
0.19 0.19 0.55 0.04 0.17 0.11 0.13 0.03 0.14
0.40 0.50 0.13 0.08 0.38 0.87 0.15 0.06 0.50 3.17
0.91 5.35 3.01 0.48 0.07 3.89 2.58 0.37 0.89 0.32 0.26
0.89 0.68 0.20 0.10 0.39 1.55 0.32 0.17 0.40 4.26 4.85 0.93
0.21 0.10 0.10 0.05 0.40 0.10 0.08 0.05 0.68 1.06 2.12 0.09 1.19
1.44 0.68 0.20 0.42 0.11 0.93 0.68 0.24 0.70 0.10 0.42 0.56 0.17 0.16
3.37 1.22 3.97 1.07 1.41 1.03 0.70 1.34 0.74 0.32 0.34 0.97 0.49 0.55 1.61
2.12 0.55 2.03 0.37 0.51 0.86 0.82 0.23 0.47 1.46 0.33 1.39 1.52 0.17 0.80 4.38
0.11 1.16 0.07 0.13 0.72 0.22 0.16 0.34 0.26 0.21 0.67 0.14 0.52 1.53 0.14 0.52 0.11
0.24 0.38 1.09 0.33 0.54 0.23 0.20 0.10 3.87 0.42 0.40 0.13 0.43 6.45 0.22 0.79 0.29 2.49
2.01 0.25 0.20 0.15 1.00 0.30 0.59 0.19 0.12 7.82 1.80 0.31 2.06 0.65 0.31 0.23 1.39 0.37 0.31

8.66 4.40 3.91 5.70 1.93 3.67 5.81 8.33 2.44 4.85 8.62 6.20 1.95 3.84 4.58 6.95 6.10 1.44 3.53 7.09
```

The entry on the i-th row and j-th column of this matrix corresponds to the rate of substitutions between amino-acids i and j. The last line in the file gives the stationary frequencies and must be separated from the rate matrix by one line. The ordering of the amino-acids is alphabetical, i.e, Ala, Arg, Asn, Asp, Cys, Gln, Glu, Gly, His, Ile, Leu, Lys, Met, Phe, Pro, Ser, Thr, Trp, Tyr and Val.

7.5 Topological constraint file

PhyML can perform phylogenetic tree estimation under user-specified topological constraints. In order to do so, one should use the `--constraint_file file_name` command-line option where `file_name` lists the topological constraints. Such constraints are straightforward to define. For instance, the following constraints:

```
((A,B,C),(D,E,F));
```

indicate that taxa A, B and C belong to the same clade. D, E and F also belong to the same clade and the two clades hence defined should not overlap. Under these two constraints, the tree ((A,B),D,((E,F),C)) is not valid. From the example above, you will notice that the constraints are defined using a multifurcating tree in NEWICK format. Note that this tree does not need

Sequence file name : 'seq'

Output file name	Content
seq_phyml_tree.txt	ML tree
seq_phyml_stats.txt	ML model parameters
seq_phyml_boot_trees.txt	ML trees – bootstrap replicates
seq_phyml_boot_stats.txt	ML model parameters – bootstrap replicates
seq_phyml_rand_trees.txt	ML trees – multiple random starts

Table 4. **Standard output files**

to display the whole list of taxa. For instance, while the only taxa involved in specifying topological constraints above are A, B, C, D, E & F, the actual data set could include more than these six taxa only.

PhyML tree topology search algorithms all rely on improving a starting tree. By default, BioNJ is the method of choice for building this tree. However, there is no guarantee that the phylogeny estimated with PhyML does comply with the topological constraints. While it is probably possible to implement BioNJ with topological constraints, we have not done so yet. Instead, the same multifurcating tree that defines the topological constraints should also be used as starting tree using the `-u` (`--inputtree`) option. Altogether, the command line should look like the following: `-u=file_name --constraint_file=file_name`. It is not possible to use as input tree a non-binary phylogeny that is distinct from that provided in the constraint tree file. However, any binary tree compatible with the constraint one can be used as input tree.

7.6 Output files

Table 4 presents the list of files resulting from an analysis. Basically, each output file name can be divided into three parts. The first part is the sequence file name, the second part corresponds to the extension '`_phyml_`' and the third part is related to the file content. When launched with the default options, PhyML only generates two files: the tree file and the model parameter file. The estimated maximum likelihood tree is in standard Newick format (see Figure 3). The model parameters file, or statistics file, displays the maximum likelihood estimates of the substitution model parameters, the likelihood of the maximum likelihood phylogenetic model, and other important information concerning the settings of the analysis (e.g., type of data, name of the substitution model, starting tree, etc.). Two additional output files are created if bootstrap supports were evaluated. These files simply contain the maximum likelihood trees and the substitution model parameters estimated from each bootstrap replicate. Such information can be used to estimate sampling errors around each parameter of the phylogenetic model. When the random tree option is turned on, the maximum likelihood trees estimated from each random starting trees are printed in a separate tree file (see last row of Table 4).

7.7 Treatment of invariable sites with fixed branch lengths

PhyML allows users to give an input tree with fixed topology and branch lengths and find the proportion of invariable sites that maximise the likelihood (option `-o r`). These two options can be considered as conflicting since branch lengths depend on the proportion of invariants. Hence, changing the proportion of invariants implies that branch lengths are changing too. More formally, let l denote the length of a branch, i.e., the expected number of substitutions per site, and p be the proportion of invariants. We have $l = (1 - p)l'$, where l' is the expected number of substitutions at variable sites. When asked to optimize p but leave l unchanged, PhyML does the following:

1. Calculate $l' = l/(1 - p)$ and leave l' unchanged throughout the optimization.

2. Find the value of the proportion of invariants that maximises the likelihood. Let p^* denote this value.

3. Set $l^* = (1 - p^*)l'$ and print out the tree with l^* (instead of l).

PhyML therefore assumes that the users wants to fix the branch lengths measured at variable sites only (i.e., l^* is fixed). This is the reason why the branch lengths in the input and output trees do differ despite the use of the the `-o r` option. While we believe that this approach relies on a sound rationale, it is not perfect. In particular, the original transformation of branch lengths ($l' = l/(1 - p)$) relies on a default value for p with is set to 0.2 in practice. It is difficult to justify the use of this value rather than another one. One suggestion proposed by Bart Hazes is to avoid fixing the branch lengths altogether and rather estimate the value of a scaling factor applied to each branch length in the input tree (option `--contrained_lens`). We agree that this solution probably matches very well most users expectation, i.e., "find the best value of p while constraining the ratio of branch lengths to be that given in the input tree". Please feel free to send us your suggestions regarding this problem by posting on forum (`http://groups.google.com/group/phyml-forum`).

8 Inputs & outputs for the XML interface

8.1 Mixture models in PhyML

PhyML implements a wide range of mixture models. The discrete gamma model [29] is arguably the most popular of these models in phylogenetics. However, in theory, mixture models are not restricted to the description of the variation of substitution rates across sites. For instance, if there are good reasons to believe that the relative rates of substitution between nucleotides vary along the sequence alignments, it makes sense to use a mixture of GTR models. Consider the case where substitutions between A and C occur at high rate in some regions of the alignment and low rate elsewhere, a mixture with two classes, each class having its own GTR rate matrix, would be suitable. The likelihood at any site of the alignment is then obtained by averaging the likelihoods obtained for each GTR rate matrix, with the same weight given to each of these matrices.

PhyML implements a generic framework that allows users to define mixtures on substitution rates, rate matrices and nucleotide or amino-acid equilibrium frequencies. Each class of the mixture model is built by assembling a substitution rate, a rate matrix[1] and a vector of equilibrium frequencies. For instance, let $\{R_1, R_2, R_3\}$ be a set of substitution rates, $\{M_1, M_2\}$ a set of rate matrices and $\{F_1, F_2\}$ a set of vectors of equilibrium frequencies. One could then define the first class of the mixture model as $\mathcal{C}_1 = \{R_1, M_1, F_1\}$, a second class as $\mathcal{C}_2 = \{R_2, M_1, F_1\}$, and a third class as $\mathcal{C}_3 = \{R_3, M_2, F_2\}$. If R_1, R_2 and R_3 correspond to slow, medium and fast substitution rates, then this mixture model allows the fast evolving rates to have their own vector of equilibrium frequencies and rate matrix, distinct from that found at the medium or slow evolving sites. The likelihood at any given site D_s of the alignment is then:

$$\Pr(D_s) = \sum_{c=1}^{3} \Pr(D_s|\mathcal{C}_s = c)\Pr(\mathcal{C}_s = c), \tag{1}$$

[1] the rate matrix corresponds here the symmetrical matrix giving the so-called "echangeability rates"

where $\Pr(\mathcal{C}_s = c)$ is obtained by multiplying the probability (density) of the three components (i.e., rate, matrix, frequencies). For instance, $\Pr(\mathcal{C}_1 = \{R_1, M_1, F_1\}) = \Pr(R_1) \times \Pr(M_1) \times \Pr(F_1)$. We therefore assume here that substitution rates, rate matrices and equilibrium frequencies are independent from one another.

Note that, using the same substitution rates, rate matrices and vector of equilibrium frequencies, it is possible to construct many other mixture models. For instance, the mixture model with the largest number of classes can be created by considering all the combinations of these three components. We would then get a mixture of $3 \times 2 \times 2 = 12$ classes, corresponding to all the possible combinations of 3 rates, 2 matrices and 2 vectors of frequencies.

8.2 Partitions

We first introduce some terms of vocabulary that have not been presented before. A partitionned data set, also referred to as partition, is a set of partition elements. Typically, a partitionned data set will be made of a set of distinct gene alignments. A partition element will then correspond to one (or several) of these gene alignments. Note that the biology litterature often uses the term partition to refer to an element of a partitionned data. We thus use here instead the mathematical definition of the terms 'partition' and 'partition element'.

Phylogenetics models usually assume individual columns of an alignment to evolve independently from one another. Codon-based models (e.g., [30–33]) are exceptions to this rule since the substitution process applies here to triplets of consecutive sites of coding sequences. The non-independence of the substitution process at the three coding positions (due to the specificities of the genetic code), can therefore be accounted for. Assuming that sites evolve independently does not mean that a distinct model is fitted to each site of the alignment. Estimating the parameters of these models would not make much sense in practice due to the very limited amount of phylogenetic signal conveyed by individual sites. Site independence means instead that the columns of the observed alignment were sampled randomly from the same "population of columns". The stochasticity of the substitution process running along the tree is deemed responsible to the variability of site patterns.

Some parameters of the phylogenetic model are considered to be common to all the sites in the alignment. The tree topology is typically one such parameter. The transition/transversion ratio is also generally assumed to be the same for all columns. Other parameters can vary from site to site. The rate at which substitutions accumulate is one of these parameters. Hence, different sites can have distinct rates. However, such rates are all "drawn" from the same probabilitic distribution (generally a discrete Gamma density). Hence, while different sites may have distinct rates of evolution, they all share the same *distribution* of rates.

This reasonning also applies on a larger scale. When analysing multiple genes, one can indeed assume that the same mechanism generated the different site patterns observed for every gene. Here again, we can assume that all the genes share the same underlying tree topology (commonly refered to as the "species tree"). Other parameters of the phylogenetic model, such as branch lengths for instance, might be shared across genes. However, due to the specificities of the gene evolution processes, some model parameters need to be adjusted for each gene separately. To sum up, the phylogenetic analysis of partitionned data requires flexible models with parameters, or distribution of parameters, shared across several partition elements and other parameters estimated separately for each element of the partition.

The likelihood of a data set made of the concatenation of n sequence alignments noted $D^{(1)}$, $D^{(2)}, \ldots, D^{(n)}$ is then obtained as follow:

$$\begin{aligned} \Pr(D^{(1)}, D^{(2)}, \ldots, D^{(n)}) &= \prod_{i=1}^{n} \Pr(D^{(i)}) \\ &= \prod_{i=1}^{n} \prod_{s=1}^{L_i} \Pr(D_s^{(i)}), \end{aligned} \tag{2}$$

where L_i is the number of site columns in partition element i. $\Pr(D_s^{(i)})$ is then obtained using Equation 1, i.e., by summing over the different classes of the mixture model that applies to site s for partition element i. Hence, the joint probability of all the partition elements is here broken down into the product of likelihood of every site for each partition element. As noted just above, any given component of the mixture model at a given particular site is shared by the other sites that belong to the same partition element and, for some of them, by sites in other partition elements (e.g., the same tree topology is shared by all the sites, throughout all the partition elements).

PhyML implements a wide variety of partition models. The only parameter that is constrained to be shared by all the partition elements is the tree topology. This constraint makes sense when considering distantly related taxa, typically inter-species data. For closely related taxa, i.e., when analysing intra-species or population-level data, not all the genes might have the same evolutionary history. Recombination events combined to the incomplete lineage sorting phenomenon can generate discrepancies between the gene trees and the underlying species tree (see [34] for a review). The phylogenetic softwares BEST [35], STEM [36] and *BEAST [37] are dedicated to the estimation of species tree phylogenies from the analysis of multi-gene data and allow gene-tree topologies to vary across genes.

Aside from the tree topology that is common to all the sites and all the partition elements, other parameters of the phylogenetic model can be either shared across partition elements or estimated separately for each of these. When analysing three partition elements, A, B and C for instance, PhyML can fit a model where the same set of branch lengths applies to A and B while C has its own estimated lengths. The same goes for the substitution model: the same GTR model, with identical parameter values, can be fitted to A and C and JC69 for instance can be used for B. The sections below give more detailed information on the range of models available and how to set up the corresponding XML configuration files to implement them.

8.3 Combining mixture and partitions in PhyML: the theory

The rationale behind mixture models as implemented in PhyML lies in (1) the definition of suitable rate matrices, equilibrium frequency vectors and relative rates of substitution and (2) the assembly of these components so as to create the classes of a mixture. The main idea behind partitionned analysis in PhyML lies in (1) the hypothesis of statistical independance of the different data partition elements and (2) distinct data partition can share model components such as rate matrices, equilibrium frequencies or distribution of rates across sites. More formally, the likelihood of a data set made of n partition elements is written as follows:

$$\begin{aligned}
\Pr(D^{(1)}, D^{(2)}, \ldots, D^{(n)}) &= \prod_{i=1}^{n} \prod_{s=1}^{L_i} \Pr(D_s^{(i)}) \\
&= \prod_{i=1}^{n} \prod_{s=1}^{L_i} \sum_{c=1}^{K_i} \Pr(D_s^{(i)} | \mathcal{C} = c) \Pr(\mathcal{C} = c),
\end{aligned}$$

where L_i is the number of sites in partition element i and K_i is the number of classes in the mixture model that applies to this same partition element. Each class of a mixture is made of a rate matrix M, a vector of equilibrium frequencies F and a relative rate of substitution R. Branch lengths, L and tree topology τ are also required for the calculation of the likelihood. Hence we have:

$$
\begin{aligned}
&\Pr(D^{(1)}, D^{(2)}, \ldots, D^{(n)}) \\
&= \prod_{i=1}^{n} \prod_{s=1}^{L_i} \sum_{c=1}^{K_i} \Pr(D_s^{(i)} | \mathcal{C} = c) \Pr(\mathcal{C} = c) \\
&= \prod_{i=1}^{n} \prod_{s=1}^{L_i} \sum_{m}^{\mathcal{M}_i} \sum_{f}^{\mathcal{F}_i} \sum_{r}^{\mathcal{R}_i} \Pr(D_s^{(i)} | M_m^{(i)}, F_f^{(i)}, R_r^{(i)}, L^{(i)}, \tau) \Pr(M_m^{(i)}, F_f^{(i)}, R_r^{(i)}) \mathcal{I}(m, f, r, i)
\end{aligned}
$$

where $\mathcal{M}_i$, $\mathcal{F}_i$ and $\mathcal{R}_i$ are the number of rate matrices, vector of equilibrium frequencies and relative rates that apply to partition element i respectively. $\mathcal{I}(m, f, r, i)$ is an indicator function that takes value 1 if the combination M_m, F_f and R_r is acually defined in the model for this particular partition element i. Its value is 0 otherwise. In the example given in section 8.1 $\{R_1, R_2, R_3\}$ is the set of substitution rates, $\{M_1, M_2\}$ the set of rate matrices and $\{F_1, F_2\}$ the set of vectors of equilibrium frequencies. We then define the first class of the mixture model as $\mathcal{C}_1 = \{R_1, M_1, F_1\}$, a second class as $\mathcal{C}_2 = \{R_2, M_1, F_1\}$ and the third as $\mathcal{C}_3 = \{R_3, M_2, F_2\}$. Hence, we have $\mathcal{I}(1, 1, 1, i)$, $\mathcal{I}(1, 1, 2, i)$ and $\mathcal{I}(2, 2, 3, i)$ equal to one while the nine other values that this indicator function takes, corresponding to the possible combinations of two vectors of frequencies, two matrices and three rates, are all zero.

As stated before, our implementation assumes that the different components of a mixture are independant. In other words, we have $\Pr(M_m^{(i)}, F_f^{(i)}, R_r^{(i)}) = \Pr(M_m^{(i)}) \times \Pr(F_f^{(i)}) \times \Pr(R_r^{(i)})$. In practice, the joint probability $\Pr(M_m^{(i)}, F_f^{(i)}, R_r^{(i)})$ is obtained as follows:

$$
\Pr(M_m^{(i)}, F_f^{(i)}, R_r^{(i)}) = \frac{\Pr(M_m^{(i)}) \Pr(F_f^{(i)}) \Pr(R_r^{(i)})}{\sum_{m,f,r} \Pr(M_m^{(i)}) \Pr(F_f^{(i)}) \Pr(R_r^{(i)}) \mathcal{I}(m, f, r, i)} \tag{3}
$$

The probabilities $\Pr(M_m^{(i)})$, $\Pr(F_f^{(i)})$ and $\Pr(R_r^{(i)})$, also called 'weights', can be fixed or estimated from the data.

8.4 The XML format and its use in PhyML

The few paragraphs below are largely inspired from the Wikipedia page that describes the XML format (http://en.wikipedia.org/wiki/XML). XML (eXtensible Markup Language) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. An XML document is divided into *markup* and *content*, which may be distinguished by the application of simple syntactic rules.

Generally, strings that constitute markup either begin with the character '<' and end with a '>'. Strings of characters that are not markup are content:

XML markup and content example

```
<markup>
 content
</markup>
```

A markup construct that begins with '<' and ends with '>' is called a *tag*. Tags come in three flavors: (1) start-tags (e.g, <section>), end-tags (e.g., </section>) and empty-element tags (e.g., <line-break />). A *component* either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag. The characters between the start- and end-tags, if any, are the element's content, and may contain markup, including other elements, which are called child elements. In the following example, the element img has two *attributes*, src and alt: . Another example would be <step number="3">Connect A to B.</step> where the name of the attribute is "number" and the value is "3".

In practice, building a mixture model in a XML file readable by PhyML is relatively straightforward. The first step is to define the different components of each class of the mixture. Consider for instance that the fitted model will have a Gamma distribution with four classes plus a proportion of invariants. The rate component of the mixture can then be specified using the following XML code:

Γ4+I rates

```

<siterates id="SiteRates1">
  <weights  id="Distrib" family="gamma+inv" alpha=".1" \
  optimise.alpha="yes" pinv="0.4" optimise.pinv="yes">
  </weights>
  <instance id="R1" init.value="1.0"/>
  <instance id="R2" init.value="1.0"/>
  <instance id="R5" init.value="0.0"/>
  <instance id="R3" init.value="1.0"/>
  <instance id="R4" init.value="1.0"/>
</siterates>

```

In the example above, the <siterates> component completely defines a model of substitution rate variation across sites. This component has a particular identity, i.e., a name associated to it ("SiteRates1" here), which is not mandatory. This <siterates> component has six sub-components. The first is the <weights> component, followed by five <instance> components. The <weights> component defines the type of distribution that characterizes the variation of rates across sites. A discrete Gamma plus invariants is used here. Two parameters specify this distribution: the gamma shape and the proportion of invariant parameters. Their initial values are set by using the corresponding attributes and attribute values (`alpha="0.1"` and `pinv="0.4"`). Also, PhyML can optimise these parameters so as to maximise the likelihood of the whole phylogenetic model (`optimise.pinv="yes"` and `optimise.alpha="yes"`). The following five `<instance>` components define the rate classes themselves. The `id` attribute is here mandatory and must be unique to each class. Note that one of the initial (relative) rate (`init.value` attribute) is set to zero. The corresponding rate class (the third in this example) will then correspond to the invariant site category.

Having specified the part of the phylogenetic model that describes the variation of rates across sites, we can now move on to build the rest of the model. The component below defines two substitution models:

Rate matrices

```
<ratematrices id="RateMatrices">
  <instance id="M1" model="HKY85" tstv="4.0" optimise.tstv="no"/>
  <instance id="M2" model="GTR" optimise.rr="yes"/>
</ratematrices>
```

This `<ratematrices>` component sets out a list of substitution models (HKY85 and GTR here). Here again, the different elements in this list correspond to the `<instance>` sub-components. Each instance must have a unique `id` attribute for a reason that will become obvious shortly. The remaining attributes and their functions are described in Section 8.6.3.

The next "ingredient" in our phylogenetic model are vectors of nucleotide frequencies. The `<equfreqs>` component below specifies two of such vectors:

Equilibrium frequencies

```
<equfreqs id="EquFreq">
  <instance id="F1"/>
  <instance id="F2"/>
</equfreqs>
```

Now, we need to assemble these three components (rate variation across sites, rate matrices and vectors of equilibrium frequencies) into a mixture model. The `<partitionelem>` component below defines one such model:

Mixture model

```
<partitionelem id="Part1" file.name="./nucleic.txt" data.type="nt">
  <mixtureelem list="R1, R2, R3, R4, R5"/>
  <mixtureelem list="M1, M1, M1, M2, M2"/>
  <mixtureelem list="F1, F2, F1, F2, F2"/>
</partitionelem>
```

The `<partitionelem>` component defines a particular partition element. In this example, the partition element corresponds to the sequence file called `nucleic.txt`, which is an alignment of nucleotide sequences (see the `data.type` attribute value). The `<mixtureelem>` are sub-components of the `<partitionelem>` component. Each `<mixtureelem>` has a `list` atrribute. Each such `list` gives the ID of components that have been defined before. For instance, the first `<mixtureelem>` refers to the five classes of the `<siterates>` component. The ordering of the different term in these list matters a lot since it is directly related to the elements in each class of the mixture model. Hence, the first element in the `<list>` attribute of the first `<mixtureelem>` added to the first element in the `<list>` attribute of the second `<mixtureelem>` plus the the first element in `<list>` attribute of the third `<mixtureelem>` defines the first class of the mixture model. Therefore, the mixture model defined above has five classes: $\mathcal{C}_1 = \{R_1, M_1, F_1\}$, $\mathcal{C}_2 = \{R_2, M_1, F_2\}$, $\mathcal{C}_3 = \{R_3, M_1, F_1\}$, $\mathcal{C}_4 = \{R_4, M_2, F_2\}$ and $\mathcal{C}_5 = \{R_5, M_2, F_2\}$.

8.5 Setting up mixture and partition models in PhyML: the basics

Mixture models are particularly relevant to the analysis of partitionned data. Indeed, some features of evolution are gene-specific (e.g., substitution rates vary across genes). Models that can accomodate for such variation, as mixture models do, are therefore relevant in this context. However, other evolutionary features are shared across loci (e.g., genes located in the same genomic region usually have similar GC contents). As a consequence, some components of mixture models need to be estimated separately for each partition element while others should be shared by different partition elements.

Below is a simple example with a partitionned data set made of two elements:

```
Two sets of branch lengths (one per partition element)

<branchlengths id="BranchLens">
  <instance id="L1"/>
  <instance id="L2"/>
</branchlengths>

<partitionelem id="Part1" file.name="./nucleic1.txt" data.type="nt">
  <mixtureelem list="R1, R2, R3, R4, R5"/>
  <mixtureelem list="L1, L1, L1, L1, L1"/>
</partitionelem>

<partitionelem id="Part2" file.name="./nucleic2.txt" data.type="nt">
  <mixtureelem list="R1, R2, R3, R4, R5"/>
  <mixtureelem list="L2, L2, L2, L2, L2"/>
</partitionelem>
```

Mixture elements with names R1,. . ., R5 refer to the Γ4+I model defined previsouly (see Section 8.4). The <branchlengths> XML component defines a mixture element that had not been introduced before. It defined vectors of branch lengths that apply to the estimated phylogeny. Two instances of such vectors are defined: L1 and L2. When examining the two partition elements (<partitionelem> component), it appears that L1 is associated with Part1 while L2 is associated with Part2. Hence, branch lengths will be estimated separately for these two partition elements.

Note that a given partition element can only have one branchlengths instance associated to it. For instance, the example given below is not valid:

```
Invalid mixture

<partitionelem id="Part1" file.name="./nucleic1.txt" data.type="nt">
  <mixtureelem list="R1, R2, R3, R4, R5"/>
  <mixtureelem list="L1, L1, L1, L2, L2"/>
</partitionelem>
```

In other words, mixture of branch lengths are forbidden. One reason for this restriction is that mixture of edge lengths sometimes lead to non-identifiable models (i.e., models with distinct sets of branch lengths have the same likelihood) [38]. But mostly, combining mixture of branch lengths with mixture of rates appears like a deadly combination. Consider for instance the following model:

```
Invalid mixture

<partitionelem id="Part1" file.name="./nucleic1.txt" data.type="nt">
  <mixtureelem list="R1, R2, R3"/>
  <mixtureelem list="L1, L2, L3"/>
</partitionelem>
```

It is here impossible to tell apart branch lengths and substitution rates. Such model is strongly non-identifiable and therefore not relevant.

In the example given above, the same Γ4+I model (i.e. the same gamma shape parameter and proportion of invariant) applies to the two partition elements. It is possible to use two distinct Γ4+I models instead using the following XML code:

Two distinct Γ4+I models

```

<siterates id="SiteRates1">
  <weights  id="Distrib1" family="gamma+inv" alpha=".1" \
  optimise.alpha="yes" pinv="0.4" optimise.pinv="yes">
  </weights>
  <instance id="R1" init.value="1.0"/>
  <instance id="R2" init.value="1.0"/>
  <instance id="R5" init.value="0.0"/>
  <instance id="R3" init.value="1.0"/>
  <instance id="R4" init.value="1.0"/>
</siterates>

<siterates id="SiteRates2">
  <weights  id="Distrib2" family="gamma+inv" alpha=".1" \
  optimise.alpha="yes" pinv="0.4" optimise.pinv="yes">
  </weights>
  <instance id="R6"  init.value="1.0"/>
  <instance id="R7"  init.value="1.0"/>
  <instance id="R8"  init.value="0.0"/>
  <instance id="R9"  init.value="1.0"/>
  <instance id="R10" init.value="1.0"/>
</siterates>

<partitionelem id="Part1" file.name="./nucleic1.txt" data.type="nt">
  <mixtureelem list="R1, R2, R3, R4, R5"/>
  <mixtureelem list="L1, L1, L1, L1, L1"/>
</partitionelem>

<partitionelem id="Part2" file.name="./nucleic2.txt" data.type="nt">
  <mixtureelem list="R6, R7, R8, R9, R10"/>
  <mixtureelem list="L2, L2, L2, L2, L2"/>
</partitionelem>

```

`SiteRates1` and `SiteRates2` here define two distinct Γ4+I models. Each of these models apply to one of the two partition elements (`nucleic1.txt` and `nucleic2.txt`), allowing them to display different patterns of rate variation across sites.

8.6 XML options

8.6.1 `phyml` component

Options:

- `output.file="filename"`. The main output files of PhyML analysis will be named `filename_phyml_tree` and `filename_phyml_stats`.

- `bootstrap="nreplicates"`. Run `nreplicates` replicates for the non-parametric bootstrap analysis.

- `run.id="idstring"`. PhyML will append the string `idstring` to each output file.

- `print.trace="yes|true|no|false"`. PhyML will print the estimated trees (and the corresponding loglikelihoods) at multiple stages of the estimation process. This option is useful for monitoring the progress of the analysis when processing large data sets.

- `branch.test="aBayes|aLRT|SH|no"`. Calculate fast branch support using the aBayes method [24], aLRT [25] or SH [39] tests. These branch statistics are much faster to estimate than the bootrap proportions and usually provide good estimates of the probabilities that the corresponding edges are correctly inferred (see Anisimova et al. 2011 for more precision). By default and if no bootstrap analysis is performed, branch supports are estimated using the aBayes approach.

8.6.2 `topology` component

Options:

- `init.tree="bionj"|"user"|"random"`. Starting tree. Default is `bionj`.

- `file.name="name_of_tree_file"`. In case `init.tree="user"`, this attribute is mandatory. `name_of_tree_file` is a text file containing a tree in NEWICK format.

- `optimise.tree="yes"|"true"|"no"|"false"`. The starting tree topology as defined by `init.tree` is to be optimised (or not) so as to maximise the likelihood function.

- `search="nni"|"spr"|"none"`. Tree topology search is conducted using NNI (fast), SPR (a bit slower but more accurate) or no moves.

Example of ‘topology’ component

```

<topology>
  <instance id="T1" init.tree="bionj" optimise.tree="true" \
   search="spr"/>
</topology>

```

8.6.3 `ratematrices` component

Options:

- `model="JC69"|"K80"|"F81"|"F84"|"HKY85"|"TN93"|"GTR"|"custom"` for nucleotide data. The default is `"HKY85"`.
  `model="LG"|"WAG"|"JTT"|"MtREV"|"Dayhoff"|"DCMut"|"RtREV"|"CpREV"|"VT"|"Blosum62"|"MtMam"|"MtArt"|"HIVw"|"HIVb"|"customaa"` for amino-acid sequences. The default is `"LG"`.

- `model.code="012345"`. For `custom` model applied to nucleotide sequences: set the string of digits that define a custom substitution model. See Table 1 on page 15 for more information about the model codes.

- `ratematrix.code="filename"`. When used in conjunction with `model="customaa"`, `filename` is the name of the file that gives the rates of substitution between amino-acids as well as their frequences at equilibrium using PAML rate matrix format. An example of such file is provided in phyml/examples/X1.mat.

- `optimise.rr="yes"|"true"|"no"|"false"`. For `custom` and `GTR` nucleotide models only: optimise the substitution rate model parameters.

- `optimise.tstv="yes"|"true"|"no"|"false"`. For `K80`, `F84`, `HKY85` and `TN93` models only: optimise the transition/transversion rate ratio.

- `tstv="value"`. For `K80`, `HKY85` and `TN93` models only: set the transition/transversion to a given value.

The `ratematrices` component has the attribute `optimise.weights=yes/no` (default is `no`). If `optimise.weights=yes`, then the probabilities (or weights) or each matrix in the set of matrices defined by this component (see Equation 3), will be estimated from the data.

Example of 'ratematrices' component

```

<ratematrices id="RM1" optimise.weights="yes">
  <instance id="M1" model="custom" model.code="000000"/>
  <instance id="M2" model="GTR" optimise.rr="yes"/>
  <instance id="M3" model="WAG"/>
</ratematrices>

```

8.6.4 `equfreqs` component

Options:

- `base.freqs="a,b,c,d"` where `a-d` are nucleotide frequencies. Make sure that these frequencies are separated by comas and no space character is inserted.

- `aa.freqs="empirical|model"`. Amino-acid frequencies are derived from counting the number of occurence of each amino-acid in the alignment (`aa.freqs="empirical"`) or given by the substitution model (`aa.freqs="model"`).

- `optimise.freqs="true|yes|false|no"`. Nucleotide frequencies can be optimised so as to maximise the likelihood (`optimise.freqs="yes|true"`).

The `equfreqs` component has the attribute `optimise.weights=yes/no` (default is `no`). If `optimise.weights=yes`, then the probabilities (or weights) or each vector of equilibrium frequencies in the set of vectors defined by this component (see Equation 3), will be estimated from the data.

Example of 'equfreqs' component

```
<equfreqs id="EF1" optimise.weights="yes">
  <instance id="F1" base.freqs="0.25,0.25,0.25,0.25"/>
  <instance id="F2" aa.freqs="empirical"/>
  <instance id="F3" optimise.freq="yes"/>
</equfreqs>
```

8.6.5 `branchlengths` component

Options:

- `optimise.lens="yes"|"true"|"no"|"false"`: branch lengths are optimised or not. The default is set to `"yes"`.

Example of 'branchlengths' component

```
<branchlengths id="BL1">
  <instance id="L1" optimise.lens="yes"/>
  <instance id="L2"/>
  <instance id="L3" optimise.lens="false"/>
</branchlengths>
```

8.6.6 `siterates` component

Options:

- `value="val"`, where `"val"` is the relative substitution rate for the corresponding class.

A `siterate` component generally includes a `weights` element that specifies the probabilitic distribution of the relative rates. The available options for such element are:

- `family="gamma|gamma+inv|freerates"`. `gamma` indicates that the distribution of the relative rates is set to be a discrete Gamma density. `gamma+inv` indicates that the relative rate model is a mixture of Gamma and invariant sites (this is the common Γ+I model). FreeRate is a model that does not use any parametric function to describe the distribution of the relative rates (see [27]). Under this option, relative rates and the corresponding frequencies of these classes are directly estimated from the data. While such approach is slightly more computationally demanding than the Γ (or Γ+I) model, it often provides a significantly better fit to the data.

- `alpha="value|optimised"`, where `value` is a real positive number. Use this option to set the gamma shape parameter to the selected value. `optimised`: the parameter is estimated from the data (see also next option).

- `optimise.alpha="yes|true|no|false"`. Optimise the shape of the Gamma distribution of relative rates (or not).

- `pinv="value|optimised"`, where `value` is in $[0, 1]$. Use this option to set the proportion of invariants to the selected value. `optimised`: the parameter is estimated from the data (see also next option).

- `optimise.pinv="yes|true|no|false"`. Optimise the proportion of invariable sites (or not).

- `optimise.freerates="yes|true|no|false"`. Optimise the parameters of the FreeRate model, i.e., the relative rates and the corresponding frequencies.

```
                    Example of 'siterates' component

<siterates id="SR1">
  <instance id="R1" init.value="1.0"/>
  <instance id="R2" init.value="1.0"/>
  <instance id="R3" init.value="1.0"/>
  <instance id="R4" init.value="1.0"/>
  <weights  id="D1" family="gamma" optimise.alpha="yes" \
  optimise.pinv="no">
  </weights>
</siterates>
```

8.6.7 `partitionelem` and `mixtureelem` components

Options:

- `file.name="inputfilename"`, where `inputfilename` is the name of the input sequence file (in PHYLIP format) to be analysed.
- `data.type="nt|aa"`. Specify the type of sequences to be processed (nucleotide of amino-acid sequences).
- `interleaved="yes|true|no|false"`. Interleaved (`yes|true`) or sequential format (`no|false`) for the sequence alignment.

Each `partitionelem` element should include exactly four `mixtureelem` elements, corresponding to branch lengths, equilibrium frequencies, substitution rate model and tree topology. The ordering of in which the `mixtureelem` elements are given does not matter, though exceptions apply for the $\Gamma + I$ model (see below). The n-th element in the `list` attribute of each `mixtureelem` defines the n-th class of the mixture model. In the example given below, the first class of the mixture is made of the following elements: `T1`, `F1`, `R1` and `L1`, the second class is made of `T1`, `F1`, `R2` and `L1`, etc.

```
                 Example of 'partitionelem' component

<partitionelem id="partition1" file.name="./small_p1.nxs" \
 data.type="nt" interleaved="yes">
  <mixtureelem list="T1, T1, T1, T1"/>
  <mixtureelem list="F1, F1, F1, F1"/>
  <mixtureelem list="R1, R2, R3, R4"/>
  <mixtureelem list="L1, L1, L1, L1"/>
</partitionelem>
```

In general, the ordering of the `mixtureelem` elements does not matter. However, when the model has invariable sites, then the corresponding class should be first in the list of classes provided by `mixtureelem`. For instance, in the example above, if the rates are defined as follows:

Example of ‘siterates’ component

```

<siterates id="SR1">
  <instance id="R1" init.value="0.0"/>
  <instance id="R2" init.value="1.0"/>
  <instance id="R3" init.value="1.0"/>
  <instance id="R4" init.value="1.0"/>
  <weights  id="D1" family="gamma+inv" optimise.alpha="yes" \
  optimise.pinv="no">
  </weights>
</siterates>
```

then `R1` corresponds to the invariable rate class (as `init.value="0.0"`). As `R1` is first in the `mixtureelem` (see line 6 in the example of `‘partionelem’` given above), PhyML will print out an explicit error message and bail out. One way to avoid this shortcoming is to define `mixtureelem` as `R4, R2, R3, R1` instead.

8.7 A simple example: GTR + Γ4 + I

The example below provides all the required options to fit a Γ4+I model to a single alignment of nucleotide sequences under the GTR model of substitution using a SPR search for the best tree topology. The `phyml` component sets the name for the analysis to `simple.example`, meaning that each output file will display this particular string of characters. Also, the tree and statistics file names will begin with `p1.output`. The tree topology will be estimated so as to maximise the likelihood and the topology search algorithm used here is SPR, as indicated by the value of the corresponding attribute (i.e., `search="spr"`). Only one vector of branch lengths will be used here since only one partition element will be processed. Hence, the `<branchlengths>` component only has one `<instance>` sub-component. Also, a single GTR model will apply to all the classes for the mixture model – the `<ratematrices>` component has only one `<instance>` sub-component, corresponding to this particular substitution model. The next component, `<equfreqs>`, indicates that a single vector of equilibrium frequencies will apply here. Next, the `<siterates>` component has five `<instance>` sub-components. Four of these correspond to the non-zero relative rates of evolution a defined by a discrete Gamma distribution. The last one (`<instance id="R5" value="0.0"/>`) defines the class of the mixture corresponding to invariable sites. The `<weight>` component indicates that a Γ+I model will be fitted here. The shape parameter of the Gamma distribution and the proportion of invariants will be estimated from the data. The `<partitionelem>` gives information about the sequence alignment (the corresponding file name, the type of data and the alignment format). The `<mixtureelem>` components next define the mixture model. Each class of the fitted model corresponds to one column, with the first column made of the following elements: `T1, M1, F1, R1` and `L1`. The second class of the mixture is made of `T1, M1, F1, R2, L1` and so forth.

Simple PhyML XML example

```
<phyml runid="simple.example" output.file="p1.output">

  <topology>
    <instance id="T1" init.tree="bionj" optimise.tree="yes" \
    search="spr"/>
  </topology>

  <branchlengths id="BL1">
    <instance id="L1" optimise.lens="yes"/>
  </branchlengths>

  <ratematrices id="RM1">
    <instance id="M1" model="GTR"/>
  </ratematrices>

  <equfreqs id="EF1">
    <instance id="F1"/>
  </equfreqs>

  <siterates id="SR1">
    <instance id="R1" value="1.0"/>
    <instance id="R2" value="1.0"/>
    <instance id="R3" value="1.0"/>
    <instance id="R4" value="1.0"/>
    <instance id="R5" value="0.0"/>
    <weights  id="D1" family="gamma+inv" optimise.alpha="yes" \
    optimise.pinv="yes">
    </weights>
  </siterates>

  <partitionelem id="partition_elem1" file.name=\
  "./p1.seq" data.type="nt" interleaved="yes">
    <mixtureelem list="T1, T1, T1, T1, T1"/>
    <mixtureelem list="M1, M1, M1, M1, M1"/>
    <mixtureelem list="F1, F1, F1, F1, F1"/>
    <mixtureelem list="R1, R2, R3, R4, R5"/>
    <mixtureelem list="L1, L1, L1, L1, L1"/>
  </partitionelem>

</phyml>
```

8.8 A second example: LG4X

The example below shows how to fit the LG4X model [40] to a given alignment of amino-acid sequences (file `M587.nex.Phy`). LG4X is a mixture model with four classes. Each class has its own rate and corresponding frequencies (hence the use of the FreeRate model below, see the <siterates> component). In the particular example given here, the rate values and frequencies are set by the users. These parameters will then be optimized by PhyML (`optimise.freerates="yes"`). Each class also has its own rate matrix and vector of equilibrium frequencies, which need to be provided by the user (Note that these matrices can be downloaded from the following web address: http://www.atgc-montpellier.fr/download/datasets/models/lg4x/LG4X_4M.txt. They are also provided in the PhyML package `example/lg4x/` directory.)

LG4X

```
<phyml run.id="lg4x" output.file="M587.tests" branch.test="no">

  <!-- Tree topology: start with BioNJ and then SPRs -->
  <topology>
    <instance id="T1" init.tree="user" file.name="user_tree.txt" \
    search="spr" optimise.tree="no"/>
  </topology>


  <!-- Four rate matrices, read from files -->
  <ratematrices id="RM1">
    <instance id="M1" model="customaa" ratematrix.file="X1.mat"/>
    <instance id="M2" model="customaa" ratematrix.file="X2.mat"/>
    <instance id="M3" model="customaa" ratematrix.file="X3.mat"/>
    <instance id="M4" model="customaa" ratematrix.file="X4.mat"/>
  </ratematrices>

  <!-- Freerate model of variation of rates across sites -->
  <siterates id="SR1">
    <instance id="R1" init.value="0.197063"/>
    <instance id="R2" init.value="0.750275"/>
    <instance id="R3" init.value="1.951569"/>
    <instance id="R4" init.value="5.161586"/>
    <weights  id="D1" family="freerates" optimise.freerates="yes">
      <instance appliesto="R1" value="0.422481"/>
      <instance appliesto="R2" value="0.336848"/>
      <instance appliesto="R3" value="0.180132"/>
      <instance appliesto="R4" value="0.060539"/>
    </weights>
  </siterates>

  <!-- Amino-acid equilibrium freqs. are given by the models -->
  <equfreqs id="EF1">
    <instance id="F1" aa.freqs="model"/>
    <instance id="F2" aa.freqs="model"/>
    <instance id="F3" aa.freqs="model"/>
    <instance id="F4" aa.freqs="model"/>
  </equfreqs>


  <!-- One vector of branch lengths -->
  <branchlengths id="BL1" >
    <instance id="L1" optimise.lens="yes"/>
  </branchlengths>


  <!-- Mixture model assemblage -->
  <partitionelem id="partition1" file.name="M587.nex.Phy" \
  data.type="aa" interleaved="yes">
    <mixtureelem list="T1, T1, T1, T1"/>
    <mixtureelem list="M1, M2, M3, M4"/>
    <mixtureelem list="F1, F2, F3, F4"/>
    <mixtureelem list="R1, R2, R3, R4"/>
    <mixtureelem list="L1, L1, L1, L1"/>
  </partitionelem>

</phyml>
```

In order to fit the LG4X model to the `proteic` sequence file provided in the `examples/` directory, simply type `./phyml --xml=../examples/lg4x/lg4x.xml` (assuming the PhyML binary is installed in the `src/` directory). You can of course slightly tweak the file `../examples/lg4x/lg4x.xml` and use it as a template to fit this model to another data set.

8.9 An example with multiple partition elements

The example below gives the complete XML file to specify the analysis of three partition elements, corresponding to the nucleotide sequence files `small_p1_pos1.seq`, `small_p1_pos2.seq` and `small_p1_pos3.seq` in interleaved PHYLIP format. `small_p1_pos1.seq` is fitted with the HKY85 model of substitution (with the transition/transversion ratio being estimated from the data), combined to a $\Gamma 4$ model of rate variation across sites (with the gamma shape parameter being estimated from the data). `small_p1_pos2.seq` is fitted to a custom substitution model with the constraint $A \leftrightarrow G{=}C \leftrightarrow T$. The nucleotide frequencies are set to $\frac{1}{4}$ here. The model does not allow substitution rates to vary across sites. `small_p1_pos3.seq` is fitted using a GTR model conbined to a $\Gamma 4$+I model of rate variation across sites. Note that the equilibrium nucleotide frequencies for the fourth and fifth class of the mixture are set to be equal to that estimated from the first partition element (i.e., `F1`) . The initial phylogeny is built using BioNJ and the tree topology is to be estimated using a NNI search algorithm.

Example of PhyML XML file

```
<phyml runid="nnisearch" output.file="small_p1_output">

  <topology>
    <instance id="T1" init.tree="bionj" optimise.tree="yes" \
    search="nni"/>
  </topology>

  <branchlengths id="BL1">
    <instance id="L1" optimise.lens="yes"/>
    <instance id="L2"/>
    <instance id="L3"/>
  </branchlengths>

  <ratematrices id="RM1">
    <instance id="M1" model="HKY85" optimise.tstv="yes"/>
    <instance id="M2" model="custom" model.code="102304" \
    optimise.rr="yes"/>
    <instance id="M3" model="GTR"/>
  </ratematrices>

  <equfreqs id="EF1">
    <instance id="F1"/>
    <instance id="F2" base.freqs="0.25,0.25,0.25,0.25"/>
    <instance id="F3"/>
  </equfreqs>

  <siterates id="SR1">
    <instance id="R1" value="1.0"/>
    <instance id="R2" value="1.0"/>
    <instance id="R3" value="1.0"/>
    <instance id="R4" value="1.0"/>
    <weights  id="D1" family="gamma" optimise.alpha="yes" \
    optimise.pinv="no">
    </weights>
  </siterates>

  <siterates id="SR2">
    <instance id="R8" value="1.0"/>
    <weights  id="D2" family="gamma" optimise.alpha="yes" \
    optimise.pinv="yes">
    </weights>
  </siterates>

  <siterates id="SR3">
    <instance id="R10" value="1.0"/>
    <instance id="R11" value="1.0"/>
    <instance id="R12" value="1.0"/>
    <instance id="R13" value="1.0"/>
    <instance id="R14" value="1.0"/>
    <weights  id="D3" family="gamma" optimise.alpha="yes" \
    optimise.pinv="yes">
    </weights>
  </siterates>

```

Example of PhyML XML file (ctnd)

```

  <partitionelem id="partition_elem1" file.name=\
  "./small_p1_pos1.seq" data.type="nt" interleaved="yes">
    <mixtureelem list="T1, T1, T1, T1"/>
    <mixtureelem list="M1, M1, M1, M1"/>
    <mixtureelem list="F1, F1, F1, F1"/>
    <mixtureelem list="R1, R2, R3, R4"/>
    <mixtureelem list="L1, L1, L1, L1"/>
  </partitionelem>

  <partitionelem id="partition_elem2" file.name=\
  "./small_p1_pos2.seq" data.type="nt" interleaved="yes">
    <mixtureelem list="T1"/>
    <mixtureelem list="M2"/>
    <mixtureelem list="R8"/>
    <mixtureelem list="F2"/>
    <mixtureelem list="L2"/>
  </partitionelem>

  <partitionelem id="partition_elem3" file.name=\
  "./small_p1_pos3.seq" data.type="nt" interleaved="yes">
    <mixtureelem list="T1, T1, T1, T1, T1"/>
    <mixtureelem list="M3, M3, M3, M3, M3"/>
    <mixtureelem list="R10, R11, R12, R13, R14"/>
    <mixtureelem list="F3, F3, F3, F1, F1"/>
    <mixtureelem list="L3, L3, L3, L3, L3"/>
  </partitionelem>

</phyml>

```

8.10 Branch lengths with invariants and partionned data

Accommodating for models with invariable sites applying to some elements of a partitioned data, with these elements sharing the same set of edge lengths can lead to inconsistencies. Consider for instance a partitioned data set with two elements. Assume that these two elements share the same set of edge lengths. Also, consider that GTR+I applies to the first element and HKY applies to the second. Now, the expected number of substitutions per site for the first element of the partition is equal to $(1-p)l$, where p is the estimated proportion of invariants and l is the maximum-likelihood estimate for the length of that specific edge. For the second element of the partition, the expected number of substitutions per site is equal to l, rather than $(1-p)l$. While l are common to the two elements, matching the specification of the input model, the actual edge lengths do differ across the two partition elements. Please be aware that, due to the programming structure implemented in PhyML, the program will only return one value here, which will be equal to $(1-p)l$.

9 Citing PhyML

The "default citation" for PhyML is:

- "New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0". Guindon S., Dufayard J.F., Lefort V., Anisimova M., Hordijk W., Gascuel O. 2010, *Systematic Biology*, 59(3):307-321

The "historic citation" for PhyML is:

- "A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood" Guindon S., Gascuel O. 2003, *Systematic Biology*, 52(5):696-704

10 Other programs in the PhyML package

PhyML is software package that provides tools to tackle problems other than estimating maximum likelihood phylogenies. Installing these tools and processing data sets is explained is the following sections.

10.1 PhyTime

PhyTime is a program that estimates node ages and substitution rates using a Bayesian approach. The performance and main features of this software are described in two article (see Section 10.1.9).

It relies on a Gibbs sampler which outperforms the "standard" Metropolis-Hastings algorithm implemented in a number of phylogenetic softwares. The details and performance of this approach are described in the following article:

10.1.1 Installing PhyTime

Compiling PhyTime is straightforward on Unix-like machines (i.e., linux and MacOS systems). PhyTime is not readily available for Windows machines but compilation should be easy on this system too. In the 'phyml' directory, where the 'src/' and 'doc/' directories stand, enter the following commands:

```
./configure --enable-phytime;
make clean;
make;
```

This set of commands generates a binary file called `phytime` which can be found in the 'src/' directory.

10.1.2 Running PhyTime

Passing options and running PhyTime on your data set is quite similar to running PhyML in commmand-line mode. The main differences between the two programs are explained below:

- PhyTime takes as mandatory input a *rooted* phylogenetic tree. Hence, the '`-u`' option must be used. Also, unlike PhyML, PhyTime does not modify the tree topology. Hence, the options that go with the '`-s`' command do not alter the input tree topology.

- PhyTime needs an input file giving information about calibration nodes. The command '`--calibration=`' followed by the name of the file containing the calibration node information is mandatory. The content of that file should look as follows:

Calibration node file
```
Dugong_dugon Procavia_capensis Elephantidae | -65 -54
Equus_sp. Ceratomorpha | -58 -54
Cercopithecus_solatus Macaca_mulatta Hylobates_lar Homo_sapiens | -35 -25
Lepus_crawshayi Oryctolagus_cuniculus Ochotona_princeps | -90 -37
Marmota_monax Aplodontia_rufa | -120 -37
Dryomys_nitedula Glis_glis | -120 -28.5
@root@ | -100 -120
```

  Every row in this file lists a set of taxa that belong to the same subtree (i.e., a clade). This list of taxa is followed by the character '`|`' and two real numbers corresponding to the lower and upper bounds of the calibration interval for the node at the root of the clade. In the example given here, the clade grouping the three taxa "Dugong_dugon", "Procavia_capensis" and "Elephantida" has -65 as lower bound and -54 as upper bound. Node ages (or node heights) are relative to the most recent tip node in the phylogeny, which age is set to 0. It is also possible to define a clade using only two taxon names. PhyTime will then search for the most recent common ancestor of these two taxa in the user-defined phylogeny and assign time boundaries to the corresponding node. For serially-sampled data, the calibration nodes correspond to tips in the tree. A calibration file will then look as follows:

Calibration node file (serially-sampled sequences)
```
taxaA | -65 -65
taxaB | -65 -65
taxaC | -20 -20
taxaD | -30 -30
taxaE | -30 -30
taxaF | -60 -60
taxaG | -61 -51
@root@ | -100 -120
```

Note that the node corresponding to the root of the whole tree has a specific label: '`@root@`'. It is important to specify upper and lower bounds for the root node in order to ensure convergence of the Gibbs sampler. If the prior interval for the root height is not specified, the upper bound will be set to the upper bound of the oldest calibration node and the lower bound will be set to twice this age. As a consequence, leaving the prior on root height interval unspecified may produce inaccurate estimates of node ages, especially if there are only few otherwise calibration nodes available.

A notable exception to this rule comes from the analysis of serial sample data, i.e., alignments in which sequences were not sampled at the same time point. For such data, the estimated number of substitutions accumulated between successive time points is used to estimate the substitution rate averaged over lineages. Because the time of collection of the sequences is generally known without ambiguity, this extra piece of data is translated into very informative calibration intervals for the tip nodes (i.e., calibration interval of zero width), which in turn results in substitution rate estimates with descreased variances. Posterior distribution of substitution rates with small variances then allows one to get good estimates of the root age.

A typical PhyTime command-line should look like the following:

```
./phytime -i seqname -u treename --calibration=calibration_file -m GTR -c 8
```

Assuming the file '`seqname`' contains DNA sequences in PHYLIP or NEXUS format, '`treename`' is the rooted input tree in Newick format and '`calibration_file`' is a set of calibration nodes, PhyTime will estimate the posterior distribution of node times and substitution rates under the assumption that the substitution process follows a GTR model with 8 classes of rates in the Gamma distribution of rates across sites. The model parameter values are estimated by a Gibbs sampling technique. This algorithm tries diferent values of the model parameters and record the most probable ones. By default, 10^6 values for each parameter are collected. These values are recorded every 10^3 sample. These settings can be modified using the appropriate command-line options (see below).

10.1.3 Upper bounds of model parameters

The maximum expected number of substitutions per along a given branch is set to 1.0. Since calibration times provide prior information about the time scale considered, it is possible to use that information to define an upper bound for the substitution rate. This upper bound is equal to the ratio of the maximum value for a branch length (1.0) by the amount of time elapsed since the oldest calibration point (i.e., the minimum of the lower bounds taken over the whole set of calibration points)[2]. It is important to keep in mind that the upper bound of the average substitution rate depends on the time unit used in the calibration priors. The value of the upper bound is printed on screen at the start of the execution.

PhyTime implements two models that authorize rates to be autocorrelated. The strength of autocorrelation is governed by a parameter which value is estimated from the data. However, it is necessary to set an appropriate upper bound for this parameter prior running the analysis. The maximum value is set such that the correlation between the rate at the beginning and at the end of a branch of length 1.0 calendar time unit is not different from 0. Here again the upper bound for the model parameter depends on the time unit. It is important to choose this unit so that a branch of length 1.0 calendar unit can be considered as short. For this reason, we recommend to select a time unit so that the calibration times take values between -10 and -1000.

10.1.4 PhyTime specific options

Beside the `--calibration` option, there are other command line options that are specific to PhyTime:

- `--chain_len=num`
  `num` is the number of iterations required to estimate the joint posterior density of all the model parameters, i.e., the length of the MCMC chain. Its default is set to 1E+6.

- `--sample_freq=num`
  `num` is the number of generations between successive collection of the model parameter values throughout the MCMC algorithm. For instance, the `--sample_freq=1E+2` option will make PhyTime sample the model parameter every 100th iteration of the MCMC algorithm. Its default is set to 1E+3.

- `--fastlk=yes (no)` [Default: no]
  The option is used to turn on (off) the approximation of the likelihood function using a multivariate normal density. By default, the exact likelihood is used. Using the normal approximation considerably speeds up

[2]The actual formula involves an extra parameter which does not need to be introduced here

the calculation. However, it is necessary to ensure that this approximation is appropriate by looking at the correlation between the exact and approximated likelihood values that are sampled. Please read Section 11.2 for a description of the appropriate steps to take.

- `--no_sequences`
  Use this option to run the sampler without sequence data. This option can be useful when one wants to compare the marginal posterior density of model parameters to those derived when ignoring the information conveyed by the sequences. Such comparison should be conducted on a systematic basis so as to determine whether the parameters estimates are mostly determined by the prior of driven by the sequence data.

- `--rate_model=gbs/gbd/gamma/clock`
  This option is to select the model of evolution of the rate of evolution. `gbs` (default) stands for Geometric Brownian + Stochastic. This model considers that rates evolve along the tree according to a geometric Brownian process [41] and the average rate of substitution along a branch is a gamma distributed random variable. This model is described in [42]. The `gbd` model (Geometric Browninan with Deterministic calculation of edge lengths) assumes the same Geometric Brownian model of rates. However, as opposed to `gbs`, this model uses a deterministic approximation to calculate the average rates of evolution along edges. This model corresponds to the one described in [41] and implemented in the program Multidivtime. `gamma` is a less sophisticated model that assumes that average rates along edges are distributed a priori according to a gamma distribution. It is analogous to the uncorrelated clock model implemented in BEAST with a gamma distribution replacing the exponential one. The `clock` option corresponds to the strict clock model where all the lineages in the tree evolve at the same pace.

10.1.5 PhyTime output

The program PhyTime generates two output files. The file called '`your_seqfile_phytime.XXXX.stats`', where XXXX is a randomly generated integer, lists the node times and branch relative rates sampled during the estimation process. It also gives the sampled values for other parameters, such as the autocorrelation of rates (parameter 'Nu'), and the rate of evolution (parameter 'EvolRate') amongst others. This output file can be analysed with the program Tracer from the BEAST package (`http://beast.bio.ed.ac.uk/Main_Page`). The second file is called 'your_seqfile_phytime.XXXX.trees'. It is the list of trees that were collected during the estimation process, i.e., phylogenies sampled from the posterior density of trees. This file can be processed using the software TreeAnnotator, also part of the BEAST package (see http://beast.bio.ed.ac.uk/Main_Page) in order to generate confidence sets for the node time estimates.

Important information is also displayed on the standard output of PhyTime (the standard output generally corresponds to the terminal window from which PhyTime was launched). The first column of this output gives the current generation, or run, of the chain. It starts at 1 and goes up to 1E+6 by default (use `--chain_len` to change this value, see above). The second column gives the time elapsed in seconds since the sampling began. The third column gives the log likelihood of the phylogenetic model (i.e., 'Felsenstein's likelihood'). The fourth column gives the logarithm of the joint prior probability of substitution rates along the tree and node heights. The fifth column gives the current sampled value of the EvolRate parameter along with the corresponding Effective Sample Size (ESS) (see Section 10.1.7) for this parameter. The sixth column gives the tree height and the corresponding ESS. The seventh column gives the value of the autocorrelation parameter followed by the corresponding ESS. The eightth column gives the values of the birth rate parameter that governs the birth-rate model of species divergence dates. The last column of the standard output gives the minimum of the ESS values taken over the whole set of node height estimates. It provides useful information when one has to decide whether or not the sample size is large enough to draw valid conclusion, i.e., decide whether the chain was run for long enough (see Section 11.2 for more detail about adequate chain length).

10.1.6 ClockRate vs. EvolRate

The average rate of evolution along a branch is broken into two components. One is called ClockRate and is the same throughout the tree. The other is called EvolRate and corresponds to a weighted average of branch-specific rates. The model of rate evolution implemented in PhyTime forces the branch-specific rate values to be greater than one. As a consequence, ClockRate is usually smaller EvolRate.

In more mathematical terms, let μ be the value of ClockRate, r_i be the value of the relative rate along branch i and Δ_i the time elapsed along branch

i. The value of EvolRate is then given by:

$$\text{EvolRate} = \mu \frac{\sum_i^{2n-3} r_i \Delta_i}{\sum_i^{2n-3} \Delta_i}.$$

It is clear from this equation that multiplying each r_i by a constant and dividing μ by the same constant does not change the value of EvolRate. The r_is and μ are then confounded, or non-identifiable, and only the value of EvolRate can be estimated from the data. Please make sure that you use the value of EvolRate rather than that of ClockRate when referring to the estimate of the substitution rate.

10.1.7 Effective sample size

The MCMC technique generates samples from a target distribution (in our case, the joint posterior density of parameters). Due to the Markovian nature of the method, these samples are not independent. The ESS is the estimated number of independent measurements obtained from a set of (usually dependent) measurements. It is calculated using the following formula:

$$\text{ESS} = N \left(\frac{1-r}{1+r} \right),$$

where N is the length of the chain (i.e., the 'raw' or 'correlated' sample size) and r is the autocorrelation value, which is obtained using the following formula:

$$r = \frac{1}{(N-k)\sigma_x^2} \sum_{i=1}^{N-k} (X_i - \mu_x)(X_{i+k} - \mu_x),$$

where μ_x and σ_x are the mean and standard deviation of the X_i values respectively and k is the lag. The value of r that is used in PhyTime corresponds to the case where $k = 1$, which therefore gives a first order approximation of the 'average' autocorrelation value (i.e., the autocorrelation averaged over the set of possible values of the lag).

10.1.8 Prior distributions of model parameters

Any Bayesian analysis requires specifying a prior distribution of model parameters. The outcome of the data analysis, i.e., the posterior distribution, is influenced by the priors. It is especially true if the signal conveyed by the data is weak. While some have argued that the specification of priors relies more on arbitrary decisions than sound scientific reasoning, choosing relevant prior distributions is in fact fully integrated in the process of building model that generates the observed data. In particular, the problem of estimating divergence times naturally lends itself to hierarchical Bayesian modelling. Based on the hypothesis that rates of evolution are conserved (to some extant) throughout the course of evolution, the hierarchical Bayesian approach provides an adequate framework for inferring substitution rates and divergence dates separately. Hence, in this situation, it makes good sense to use what is known about a relatively well-defined feature of the evolution of genetic sequences (the "molecular clock" hypothesis combined to stochastic variations of rates across lineages) to build a prior distribution on rates along edges.

10.1.9 Citing PhyTime

The "default citation" is:

- Guindon S. "From trajectories to averages: an improved description of the heterogeneity of substitution rates along lineages", *Systematic Biology*, 2012. doi: 10.1093/sysbio/sys063.

An earlier article also described some of the methods implemented in PhyTime:

- Guindon S. "Bayesian estimation of divergence times from large data sets", *Molecular Biology and Evolution*, 2010, 27(8):1768:81.

11 Recommendations on program usage

11.1 PhyML

The choice of the tree searching algorithm among those provided by PhyML is generally a tough one. The fastest option relies on local and simultaneous modifications of the phylogeny using NNI moves. More thorough explorations of the space of topologies are also available through the SPR options. As these two classes of tree topology moves involve different computational burdens, it is important to determine which option is the most suitable for the type of data set or analysis one wants to perform. Below is a list of recommendations for typical phylogenetic analyses.

1. *Single data set, unlimited computing time.* The best option here is probably to use a SPR search (i.e., straight SPR of best of SPR and

NNI). If the focus is on estimating the relationships between species, it is a good idea to use more than one starting tree to decrease the chance of getting stuck in a local maximum of the likelihood function. Using NNIs is appropriate if the analysis does not mainly focus on estimating the evolutionary relationships between species (e.g. a tree is needed to estimate the parameters of codon-based models later on). Branch supports can be estimated using bootstrap and approximate likelihood ratios.

2. *Single data set, restricted computing time.* The three tree searching options can be used depending on the computing time available and the size of the data set. For small data sets (i.e., < 50 sequences), NNI will generally perform well provided that the phylogenetic signal is strong. It is relevant to estimate a first tree using NNI moves and examine the reconstructed phylogeny in order to have a rough idea of the strength of the phylogenetic signal (the presence of small internal branch lengths is generally considered as a sign of a weak phylogenetic signal, specially when sequences are short). For larger data sets (> 50 sequences), a SPR search is recommended if there are good evidence of a lack of phylogenetic signal. Bootstrap analysis will generally involve large computational burdens. Estimating branch supports using approximate likelihood ratios therefore provides an interesting alternative here.

3. *Multiple data sets, unlimited computing time.* Comparative genomic analyses sometimes rely on building phylogenies from the analysis of a large number of gene families. Here again, the NNI option is the most relevant if the focus is not on recovering the most accurate picture of the evolutionary relationships between species. Slower SPR-based heuristics should be used when the topology of the tree is an important parameter of the analysis (e.g., identification of horizontally transferred genes using phylogenetic tree comparisons). Internal branch support is generally not a crucial parameter of the multiple data set analyses. Using approximate likelihood ratio is probably the best choice here.

4. *Multiple data sets, limited computing time.* The large amount of data to be processed in a limited time generally requires the use of the fastest tree searching and branch support estimation methods Hence, NNI and approximate likelihood ratios rather than SPR and non-parametric bootstrap are generally the most appropriate here.

Another important point is the choice of the substitution model. While default options generally provide acceptable results, it is often warranted to perform a pre-analysis in order to identify the best-fit substitution model. This pre-analysis can be done using popular software such as Modeltest [43] or ProtTest [44] for instance. These programs generally recommend the use of a discrete gamma distribution to model the substitution process as variability of rates among sites is a common feature of molecular evolution. The choice of the number of rate classes to use for this distribution is also an important one. While the default is set to four categories in PhyML, it is recommended to use larger number of classes if possible in order to best approximate the patterns of rate variation across sites [45]. Note however that run times are directly proportional to the number of classes of the discrete gamma distribution. Here again, a pre-analysis with the simplest model should help the user to determine the number of rate classes that represents the best trade-off between computing time and fit of the model to the data.

11.2 PhyTime

Analysing a data set using PhyTime should involve three steps based on the following questions: (1) do the priors seem to be adequate (2) can I use the fast approximation of the likelihood and (3) how long shall I run the program for? I explain below how to provide answers to these questions.

- *Are the priors adequate?* Bayesian analysis relies on specifiying the joint prior density of model parameters. In the case of node age estimation, these priors essentially describe how rates of substitution vary across lineages and the probabilistic distribution that node ages have when ignoring the information provided by the genetic sequences. These priors vary from tree to tree. It is therefore essential to check the adequacy of priors for each user-defined input tree. In order to do so, PhyTime needs to be run with the `--no_data` option. When this option is required, the sequence data provided as input will be ignored and the rest of the analysis will proceed normally. The prior distribution of model parameters, essentially edge rates and node heights, can then be checked using the program Tracer as one would do for the standard 'posterior' analysis.

- *Can I use the fast approximation to the likelihood?* The suface of the log-likelihood function can be approximated using a multivariate normal density. This technique is saving very substantial amounts of computation time. However, like most approximations, there are situations where it does not provide a good fit to the actual function. This usually happens when the phylogeny displays a lot of short branches, i.e., the signal conveyed by the sequences is weak. It is therefore important to first check whether using the approximate likelihood is reasonable. In order to do so, it is recommended to first run the program without the approximation, i.e., using the default settings. Once the minimum value of the ESS of node ages (the last column on the right of the standard output) has reached 40-50, open the `phytime.XXXX` output file with Tracer and examine the correlation between the exact and approximate likelihood values. If the correlation is deemed to be good enough, PhyTime can be re-run using the `--fast_lk` option, which uses the fast normal approximation to the likelihood function.

- *How long shall I run the program for?* PhyTime should be run long enough such that the ESS of each parameter is 'large enough'. The last column on the right handside of the standard output gives the minimum ESS across all internal node heights. It is recommended to run the program so that this number reaches at least 100.

12 Frequently asked questions

1. *PhyML crashes before reading the sequences. What's wrong ?*

   - The format of your sequence file is not recognized by PhyML. See Section 7
   - The carriage return characters in your sequence files are not recognized by PhyML. You must make sure that your sequence file is a plain text file, with standard carriage return characters (i.e., corresponding to "`\n`", or "`\r`")

2. *The program crashes after reading the sequences. What's wrong ?*

   - You analyse protein sequences and did not enter the `-d aa` option in the command-line.
   - The format of your sequence file is not recognized by PhyML. See Section 7

3. *Does PhyML handle outgroup sequences ?*

   - Yes, it does. Outgroup taxa are identified by adding the '*' sign at the end of each corresponding sequence name (see Section 7.1.2)

4. *Does PhyML estimate clock-constrained trees ?*

   - No, the PhyML program does not estimate clock-contrained trees. One can however use the program PhyTime to perform such analysis but the tree topology will not be estimated.

5. *Can PhyML analyse partitioned data, such as multiple gene sequences ?*

   - We are currently working on this topic. Future releases of the program will provide options to estimate trees from phylogenomic data sets, with the opportunity to use different substitution models on the different data partitions (e.g., different genes). PhyML will also include specific algorithms to search the space of tree topologies for this type of data.

13 Acknowledgements

The development of PhyML since 2000 has been supported by the Centre National de la Recherche Scientifique (CNRS) and the Ministère de l'Éducation Nationale.

References

[1] Guindon, S. & Gascuel, O. A simple, fast and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology* **52**, 696–704 (2003).

[2] Stamatakis, A. RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics* **22**, 2688–2690 (2006).

[3] Zwickl, D. *Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion.* Ph.D. thesis, The University of Texas at Austin (2006).

[4] Jukes, T. & Cantor, C. Evolution of protein molecules. In Munro, H. (ed.) *Mammalian Protein Metabolism*, vol. III, chap. 24, 21–132 (Academic Press, New York, 1969).

[5] Kimura, M. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution* **16**, 111–120 (1980).

[6] Felsenstein, J. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution* **17**, 368–376 (1981).

[7] Felsenstein, J. *PHYLIP (PHYLogeny Inference Package) version 3.6a2* (Distributed by the author, Department of Genetics, University of Washington, Seattle, 1993).

[8] Hasegawa, M., Kishino, H. & Yano, T. Dating of the Human-Ape splitting by a molecular clock of mitochondrial-DNA. *Journal of Molecular Evolution* **22**, 160–174 (1985).

[9] Tamura, K. & Nei, M. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Molecular Biology and Evolution* **10**, 512–526 (1993).

[10] Lanave, C., Preparata, G., Saccone, C. & Serio, G. A new method for calculating evolutionary substitution rates. *Journal of Molecular Evolution* **20**, 86–93 (1984).

[11] Tavaré, S. Some probabilistic and statistical problems on the analysis of DNA sequences. *Lectures on Mathematics in the Life Sciences* **17**, 57–86 (1986).

[12] Le, S. & Gascuel, O. An improved general amino-acid replacement matrix. *Mol. Biol. Evol.* **25**, 1307–1320 (2008).

[13] Whelan, S. & Goldman, N. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Molecular Biology and Evolution* **18**, 691–699 (2001).

[14] Dayhoff, M., Schwartz, R. & Orcutt, B. A model of evolutionary change in proteins. In Dayhoff, M. (ed.) *Atlas of Protein Sequence and Structure*, vol. 5, 345–352 (National Biomedical Research Foundation, Washington, D. C., 1978).

[15] Jones, D., Taylor, W. & Thornton, J. The rapid generation of mutation data matrices from protein sequences. *Computer Applications in the Biosciences (CABIOS)* **8**, 275–282 (1992).

[16] Henikoff, S. & Henikoff, J. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)* **89**, 10915–10919 (1992).

[17] Adachi, J. & Hasegawa, M. MOLPHY version 2.3. programs for molecular phylogenetics based on maximum likelihood. In Ishiguro, M. *et al.* (eds.) *Computer Science Monographs*, 28 (The Institute of Statistical Mathematics, Tokyo, 1996).

[18] Dimmic, M., Rest, J., Mindell, D. & Goldstein, D. rtREV: an amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny. *Journal of Molecular Evolution* **55**, 65–73 (2002).

[19] Adachi, J., P., W., Martin, W. & Hasegawa, M. Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA. *Journal of Molecular Evolution* **50**, 348–358 (2000).

[20] Kosiol, C. & Goldman, N. Different versions of the Dayhoff rate matrix. *Molecular Biology and Evolution* **22**, 193–199 (2004).

[21] Muller, T. & Vingron, M. Modeling amino acid replacement. *Journal of Computational Biology* **7**, 761–776 (2000).

[22] Cao, Y. *et al.* Conflict among individual mitochondrial proteins in resolving the phylogeny of eutherian orders. *Journal of Molecular Evolution* **47**, 307–322 (1998).

[23] Gascuel, O. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Molecular Biology and Evolution* **14**, 685–695 (1997).

[24] Anisimova, M., Gil, M., Dufayard, J., Dessimoz, C. & Gascuel, O. Survey of branch support methods demonstrates accuracy, power, and robustness of fast likelihood-based approximation schemes. *Systematic biology* **60**, 685–699 (2011).

[25] Anisimova, M. & Gascuel, O. Approximate likelihood-ratio test for branches: a fast, accurate, and powerful alternative. *Syst. Biol.* **55**, 539–552 (2006).

[26] Posada, D. & Crandall, K. Selecting models of nucleotide substitution: an application to human immunodeficiency virus 1 (hiv-1). *Molecular Biology and Evolution* **18**, 897–906 (2001).

[27] Soubrier, J. *et al.* The influence of rate heterogeneity among sites on the time dependence of molecular rates. *Molecular Biology and Evolution* (2012).

[28] Maddison, D., Swofford, D. & Maddison, W. NEXUS: an extensible file format for systematic information. *Systematic Biology* **46**, 590–621 (1997).

[29] Yang, Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *Journal of Molecular Evolution* **39**, 306–314 (1994).

[30] Yang, Z. Likelihood ratio tests for detecting positive selection and application to primate lysozyme evolution. *Molecular Biology and Evolution* **15**, 568–573 (1998).

[31] Yang, Z. & Nielsen, R. Estimating synonymous and nonsynonymous substitution rates under realistic evolutionary models. *Molecular Biology and Evolution* **17**, 32–43 (2000).

[32] Yang, Z. & Nielsen, R. Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages. *Molecular Biology and Evolution* **19**, 908–917 (2002).

[33] Guindon, S., Rodrigo, A., Dyer, K. & Huelsenbeck, J. Modeling the site-specific variation of selection patterns along lineages. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)* **101**, 12957–12962 (2004).

[34] Degnan, J. & Rosenberg, N. Gene tree discordance, phylogenetic inference and the multispecies coalescent. *Trends in Ecology & Evolution* **24**, 332–340 (2009).

[35] Liu, L. Best: Bayesian estimation of species trees under the coalescent model. *Bioinformatics* **24**, 2542–2543 (2008).

[36] Kubatko, L., Carstens, B. & Knowles, L. Stem: species tree estimation using maximum likelihood for gene trees under coalescence. *Bioinformatics* **25**, 971–973 (2009).

[37] Heled, J. & Drummond, A. Bayesian inference of species trees from multilocus data. *Molecular Biology and Evolution* **27**, 570–580 (2010).

[38] Matsen, F. & Steel, M. Phylogenetic mixtures on a single tree can mimic a tree of another topology. *Systematic biology* **56**, 767–775 (2007).

[39] Shimodaira, H. & Hasegawa, M. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Molecular Biology and Evolution* **16**, 1114–1116 (1999).

[40] Le, S. Q., Dang, C. C. & Gascuel, O. Modeling protein evolution with several amino acid replacement matrices depending on site rates. *Molecular Biology and Evolution* **29**, 2921–2936 (2012).

[41] Kishino, H., Thorne, J. & Bruno, W. Performance of a divergence time estimation method under a probabilistic model of rate evolution. *Mol. Biol. Evol.* **18**, 352–361 (2001).

[42] Guindon, S. From trajectories to averages: an improved description of the heterogeneity of substitution rates along lineages. *Systematic Biology* (2012).

[43] Posada, D. & Crandall, K. Modeltest: testing the model of DNA substitution. *Bioinformatics* **14**, 817–918 (1998).

[44] Abascal, F., Zardoya, R. & Posada, D. Prottest: selection of best-fit models of protein evolution. *Bioinformatics* **21**, 2104–2105 (2005).

[45] Galtier, N. & Jean-Marie, A. Markov-modulated Markov chains and the covarion process of molecular evolution. *Journal of Computational Biology* **11**, 727–733 (2004).

Index