PHASE: a Software Package for **Ph**ylogenetics **A**nd **S**equence **E**volution

Vivek Gowri-Shankar and Howsun Jow,
with amendments by James Allen[1]

Version 3.0 - June 2013

[1]Please send bug reports, suggestions, and requests for help to simon.whelan.evolution@gmail.com

Why is PHASE different from other phylogenetic programs?

Though this sofware package can handle various types of molecular sequences, it is designed specifically for use with RNA sequences that have a conserved secondary structure, e.g. rRNAs and tRNAs. It is well known that compensatory substitutions occur in the paired regions of RNA helices. This means that substitutions occurring on one side of a pair are correlated with substitutions on the other side. Most phylogenetic programs assume that each site in a molecule evolves independently of the others and this assumption is not valid for RNA genes. Since current methods are based on the probability of nucleotide replacements over evolutionary time, neglecting the selection mechanisms which act for the maintenance of RNA stems can strongly affect the estimation of likelihood of the plausible evolutionary scenarios in competition.

Substitution models of sequence evolution that consider pairs of sites rather than single sites are implemented in this package, along with the standard nucleotide, codon and amino-acid substitution models in use nowadays. When a RNA molecule with a secondary structure is used in conjunction with a RNA substitution model, **PHASE** requires a structure-based alignment of the sequences. It is assumed that you can provide a consensus secondary structure in bracket and dot notation at the top of your alignment.

PHASE uses a Markov-Chain Monte Carlo sampler to generate large numbers of possible phylogenetic trees with probability proportional to their likelihood. This is a Bayesian statistical method that allows posterior probabilities to be generated for alternative trees and alternative clades. These posterior probabilities provide a sound statistical measure of support of alternative phylogenetic hypotheses, and they (arguably) remove the need for bootstrapping. Where many alternative arrangements of a given set of species exist, it is possible to calculate posterior probabilities for all the alternative arrangements of these species in a convenient way.

Some standard Maximum-Likelihood techniques for inferring the optimal tree with any of the DNA or RNA evolution models are also implemented. **PHASE** can also compute a matrix of pairwise distances between sequences in your alignment which can be used as input for other phylogenetic software.

To increase reliability, it is now commonplace to perform combined analyses of heterogeneous sequence data when genes with different patterns of evolution are sequenced for a set of studied species. It is possible to use several substitution models simultaneously with **PHASE** when analysing protein coding genes or when stems and loops of RNA genes are used. We also recently implemented some heterogeneous models to account for the variation of the process over time and across sites frequently observed with real sequences. Though they are still under investigation and testing, these methods are made available.

The program's features include:

- Bayesian estimation of phylogenies and substitution model parameters
- RNA models with 6, 7 and 16 states, standard 4 state DNA models, RY model, AGY model
- Invariant and discrete gamma models for substitution rate heterogeneity between sites
- Mixing of molecular data types in a single analysis
- Models to account for heterogeneity in space and in time
- Flexible priors and hyperpriors on the model parameters

Journal publications

- A. Gibson, V. Gowri-Shankar, P.G. Higgs and M. Rattray. "A comprehensive analysis of mammalian mitochondrial genome base composition and improved phylogenetic methods". Molecular Biology and Evolution, 22(2):251-264 (2005).
- M.J. Telford, M.J. Wise, V. Gowri-Shankar. "Consideration of RNA secondary structure significantly improves likelihood-based estimates of phylogeny: examples from the bilateria". Molecular Biology and Evolution, 22(4):1129-1136 (2005).

- P. Higgs, D. Jameson, H. Jow, M. Rattray. "The Evolution of tRNA-Leu Genes in Animal Mitochondrial Genomes". Journal of Molecular Evolution, 57(4):435-445 (2003).
- C. Hudelot, V. Gowri-Shankar, H. Jow, M. Rattray and P. Higgs. "RNA-based phylogenetic methods: application to mammalian mitochondrial RNA sequences". Molecular Phylogenetics and Evolution, 28(2):241-252 (2003).
- H. Jow, C. Hudelot, M. Rattray and P. Higgs. "Bayesian phylogenetics using an RNA substitution model applied to early mammalian evolution". Molecular Biology and Evolution, 19(9):1591-1601 (2002).

Acknowledgments

The **PHASE** software was developed as part of a BBSRC funded research project into RNA-based phylogenetic methods (investigators: Paul Higgs and Magnus Rattray). Under the supervision of Magnus Rattray, Howsun Jow and Vivek Gowri-Shankar carried out the programming as PhD students at Manchester University. Bastien Guillard and Antoine Buxerolles implemented codon and amino-acid models during their MSc projects. We gratefully acknowledge contributions to the design, documentation and testing from Paul Higgs and Cendrine Hudelot and we thank all the users of the 1.x versions for sharing their experience with RNA models, reporting bugs and giving valuable comments. After a hiatus in the development of **PHASE**, James Allen made some revisions to enable comparison of models across state-space, and brought some of the code up to date, during his PhD with Simon Whelan at the University of Manchester.

Contents

Introduction

What's new in PHASE version 3.0

Version 3.0 of **PHASE** consists of improvements to the stability of the programs when using relatively short alignments and/or highly parametrised models, and some new functionality in the maximum likelihood programs (`mlphase` and `optimizer`).

Empirical frequencies

The parameter optimization in version 2.0 of **PHASE**'s maximum likelihood (ML) programs (`mlphase` and `optimizer`) estimates (di)nucleotide frequency parameters via ML rather than using empirical frequencies. (Amino acid and codon models have an *Empirical Values* option, for specifying a file containing empirical frequencies, such as the JTT or WAG matrices.) ML estimates are potentially useful in evolutionary models in general, and in RNA models in particular, where the original authors of **PHASE** found that the empirical and ML frequencies of non-canonical base pairs differed significantly [??]. Nonetheless, empirical frequencies are widely used, and can provide sufficiently good estimates of the values to be useful. Having the option to use empirical frequencies allows one to compare the effect on likelihood values of empirical versus ML frequencies, and makes it possible to compare models of evolution with different numbers of states.

The default in **PHASE** 3.0 is to use ML estimates of frequency parameters; to use frequencies calculated from the alignment, the following line should be added to the **MODEL** block of the control file:

```
Empirical frequencies = yes
```

If a mixed model is used, the setting applies to all of the models, and thus must appear in the **MODEL** block, rather than any of the **MODEL***i* blocks (where *i* is the model number). ML estimation of frequencies can be explicitly indicated by setting the value of *Empirical frequencies* to 'no'. Note that, as with all options in PHASE control files, *Empirical frequencies* is case-sensitive.

Mapping 7-state likelihoods to 16-state space

Until now, there was no easy way to compare the likelihoods between DNA and RNA models with different state spaces. The likelihoods of 4-state nucleotide and 16-state dinucleotide models are, in fact, directly comparable; and a likelihood correction is easily calculated to make 7-state models equivalent to a 16-state model. The `optimizer` program in **PHASE** 3.0 reports the likelihood correction value in its output if a 7-state model is used, for both the empirical and equal frequency variations.

Additional models

DNA model: `F81`

Earlier versions of **PHASE** omitted the `F81` model [?], which allows unequal equilibrium frequencies, and assumes a single rate of exchangeability. More complex models are often used now, but it has been added for completeness.

RNA model: `RNA7G`

The `RNA7G` model takes the process of parameter generalisation to its natural conclusion within the set of 7-state models. It combines the `7E` and `7F` models, and thus has base pair symmetry in the frequency parameters, and a mismatch rate and a single transition rate; double substitutions are not permitted. With 4 free parameters, it is the simplest available 7-state model.

Model selection with Perl library and script

The 3.0 distribution of **PHASE** has a `scripts` directory, which contains a Perl module, `PHASEUtils.pm`, with a range of functions for executing PHASE programs, and for sequence manipulation and file IO. Documentation is provided in POD format within the file. These functions are used by the `model_selection.pl` script, which performs maximum likelihood inference of model parameters and branch lengths for a fixed topology, using a range of DNA and RNA models. AIC values are calculated in order to choose the model which best fits the data. The script can then perform MCMC tree search using this best model.

By default, model selection uses two 4-state nucleotide models (`HKY85` and `REV`), seven 7-state dinucleotide models (`RNA7A`, `RNA7B`, `RNA7C`, `RNA7D`, `RNA7E`, `RNA7F`, `RNA7G`), and nine 16-state dinucleotide models (`RNA16A`, `RNA16B`, `RNA16C`, `RNA16D`, `RNA16E`, `RNA16F`, `RNA16I`, `RNA16J`, `RNA16K`). The PHASE parameters are controlled by configuration files in the templates subdirectory.

The models to be executed have one of four possible patterns:

1. a single nucleotide+G model for stems and loop
2. different nucleotide+G models for stems and loops
3. a nucleotide+G model for loops and a dinucleotide model for stems
4. a nucleotide+G model for loops and a dinucleotide+G model for stems

where +G indicates gamma-distributed rates-across-sites. Each pattern has a corresponding template file, which contains most of the parameters necessary to execute **PHASE** (the model name and filenames are inserted by the `model_selection.pl` script). Two lists of nucleotide and dinucleotide models are specified, and are combined with the four patterns to generate RNA models for all combinations of nucleotide and dinucleotide model.

Mandatory parameters:

```
--a[lignment_file] <file>
--s[tructure_file] <file>
--t[ree_file] <file>
```

Three input files are required, an alignment in Fasta format, a structure in dot-bracket notation, and a tree in Newick format. Readseq is included in this distribution of **PHASE** (`../readseq`), so please use that to convert to Fasta format, if necessary.

Optional parameters:

```
--o[ut_dir] <dir>
```

If an output directory is not specified, results are saved in the current directory. The PHASE input and output files are saved in subdirectories, "`control`" and "`results`", respectively. The output is summarised in a text file in the output directory.

```
--phase_o[ptimizer] <file>
--phase_m[cmc] <file>
```

The script will first look for the required **PHASE** executable files in the directory "`../bin`". If not found, the executables are assumed to be in your path. If they are in neither of these locations, paths must be provided as parameters.

`--phase_t[emplate_dir] <dir>`

The script uses templates to generate the control files that are required by **PHASE**; these are stored in the "`./templates`" directory. If this directory cannot be found, a path must be provided as a parameter.

`--c[alculate_sample_size] ('characters'|'sites')`

Calculation of corrected AIC (AICc) values requires a sample size. It is not clear what this means in the context of sequence alignments, but two commonly used approximations are the number of characters and the number of sites.

Optional flags:

`--v[erbose]`

Display informational messages about what the script is doing.

`--r[eplace]`

Existing results will not be overwritten unless this option is used.

`--m[cmc_tree_search]`

After model selection, do MCMC tree search with the best model. The ML estimates are used as a starting point for the model parameters. The MCMC search has 150,000 burn-in iterations, 300,000 sampling iterations, and a sampling period of 100. See the `*.mcmc` files in the "`./templates`" directory for further details. The majority-rule consensus tree (nodes labelled with posterior probabilities) is saved in the output directory. For comparison, the MCMC tree search is repeated for the simplest nucleotide+G model (HKY+G, unless you edit the script).

`--e[xhaustive_mcmc]`

After model selection, do MCMC tree search with all models (parameters the same as the `--m` option). Note that this could be computationally intensive, so is not generally recommended.

Acquiring and installing the software

PHASE can be downloaded from http://code.google.com/p/rna-phase-3; the source code is available, as well as pre-compiled executable files for Windows, Linux and Mac OS X platforms.

Pre-compiled executables

The quickest and easiest way to get started with **PHASE** is to use the pre-compiled executables in the `bin_*` directories:

- `bin_windows`: Windows (cygwin)
- `bin_x86_32`: 32-bit Linux
- `bin_x86_64`: 64-bit Linux

- `bin_macosx`: Mac OS X

We recommend that you add the executables to your path, for ease of use. You can either copy them to somewhere that's already in your path, or (probably easier) add the relevant `bin_*` directory to your path. For example, in a bash environment, assuming that you've placed **PHASE** in your home directory:

```
export PATH=$PATH:$HOME/rna-phase-3/bin_x86_32
```

Add this to `$HOME/.bashrc` to execute every time that you log in.

Compiling PHASE

To compile the program yourself, clone the git repository from http://code.google.com/p/rna-phase-3/source/checkout. Then compile, from the top-level `rna-phase-3` directory.

Linux (including cygwin on Windows)

1. To compile, type `make` on the command line. A recent-ish g++ version (at least gcc 3.x) is required. The latest gcc version that has been successfully tested is 4.5.3.
2. **PHASE** uses BLAS and LAPACK libraries, which are probably already installed on your system. By default, the compilation will use those, and you do not need to edit the make files.
3. If you get a message about missing BLAS and LAPACK libraries, you can use the versions that are included with PHASE; this requires `gfortran` to be installed. You need to change the OPTLIBS/LIBS options in the file `makefile` to compile BLAS and LAPACK libraries:

   ```
   OPTLIBS = false
   LIBS = -llapack -lblas -lgfortran -lm
   ```

Mac OS X

1. To compile, type `make` on the command line. A recent-ish g++ version (at least gcc 3.x) is required. The latest gcc version that has been successfully tested is 4.5.3.
2. PHASE uses BLAS and LAPACK libraries, which are probably already installed on your system. By default, the compilation will use Linux settings, so you need to change the OPTLIBS/LIBS options in the file `makefile`:

   ```
   OPTLIBS = true
   LIBS = -framework vecLib
   ```

3. If the compilation fails during the linking stage, you might have to use:

   ```
   LIBS = -framework vecLib -bind_at_load
   ```

4. **PHASE** might not compile on newer systems because of a previous patch that was added for older Mac OS X. If the compilation fails with the error: `"error: parse error before 'sizeof'"` in `"include/configfix.h"`, then you have to edit this file and remove the 2 lines:

   ```
   extern "C" int isnan (double);
   extern "C" int isinf (double);
   ```

Solaris

On Solaris, you need to use the GNU make instead of the default make; type `gmake` instead of `make` to compile.

Installation

There is no installation procedure (i.e. no `make install`).

The programs of the package are created in the `bin` directory. See the instructions in the **Precompiled executables** section for adding this directory to your execution path.

Description of programs in the PHASE package

The **PHASE** (**PH**ylogenetics **A**nd **S**equence **E**volution) package consists of two main programs, `mcmcphase` and `mlphase`.

- `mcmcphase` is a Bayesian phylogenetic inference program
- `mlphase` performs maximum-likelihood inference with (limited) tree-search capabilities

There are six other smaller programs in the package:

- `mcmcsummarize` is used with `mcmcphase` to output and summarize the results of a MCMC run
- `optimizer` is a smaller version of `mlphase` designed to optimize user-defined trees
- `likelihood` computes the likelihood of the data given an evolutionary model (tree + substitution model)
- `distphase` computes pairwise-ML distance estimates between species
- `analyzer` checks the content of the molecular sequences
- `simulate` generates sequences according to a specified evolutionary model

Below we summarize the behaviour of these programs; details are given in subsequent sections.

`mcmcphase` and `mcmcsummarize`

The mcmcphase program performs Bayesian phylogenetic inference. It uses a Markov Chain Monte Carlo algorithm to sample from the posterior probability distribution of phylogenetic tree topology, branch lengths and sequence evolution model parameters [Jow et al., 2002].

The `mcmcsummarize` program is used to exploit the results of a MCMC run. This program produces a PHYLIP-style consensus tree (extended majority-rule) and attempts to add some branch lengths to it. It also produces two consensus models (using mean and median of the parameters Bayesian posterior probabilities computed from the sample).

`mlphase`, `optimizer` and `distphase`

The `mlphase` program is a maximum-likelihood phylogenetic inference program. It is designed to find the evolutionary model (tree topology, branch lengths and, optionally, substitution model parameters) that yields the highest probability of having generated the observed sequences. Parameters are optimized for each tree visited and different strategies are implemented to search for the ML phylogeny in the discrete topology space. The user can choose the search algorithm to be used among the four available:

- Simple exhaustive search: all the possible phylogenies are considered
- Branch and bound search: non-optimal phylogenies are rejected before evaluation
- Heuristic search via stepwise addition: greedy search for the best topology
- Star-decomposition: greedy search for the best topology

The two first algorithms are theorically guaranted to find the best tree but are computationaly too intensive to be used with a large number of species. Constraints can be placed on the phylogenetic tree topologies that are considered during ML inference in order to reduce the search space and the computation time.

The `optimizer` program is similar to `mlphase` but has no topology search capacity. It is designed to compute ML estimates of evolutionary parameters (branch lenghts and substitution model parameters) for a set of topologies provided by the user (e.g. competing evolutionary hypotheses, the consensus tree found with MCMC techniques, topologies found with distance-based methods).

The `distphase` program should prove useful when analysing a large number of species in the ML framework. The program computes evolutionary distances between pairs of taxa for a given substitution model (base-pair models can be used). It produces a distance matrix that can be used as an input for a tree reconstruction algorithm (e.g. UGPMA or NJ).

`likelihood`

The `likelihood` program computes the likelihood of a phylogeny with any implemented substitution model. Branch lengths and substitution parameters are provided by the user. This program can be used for marginal ancestral sequence reconstruction and to estimate site-specific substitution rate.

`simulate`

The program `simulate` generates molecular sequences according to an user-specified tree (topology and branch lengths) and substitution model (type and parameters). It can also generate "random" topologies with birth-death processes and sampling effects.

`analyzer` and `splitdataset`

The program `analyzer` outputs basic statistics about a sequence data file. It can also be used to locate sites with many gaps, in case you decide to remove them. The `analyzer` program can be quite useful to validate your secondary structure alignment and to set a maximum limit for the mismatch frequency at each site.

The `splitdataset` program splits a given datafile into smaller parts. For instance, it can be used to separate RNA stems and loops in two different alignments or it can extract the two first codon positions from a protein-coding genes alignment.

Running the programs

Programs in the **PHASE** package are run through the command line under both Unix-like and Windows systems. For Windows operating systems, you have to open an MS-DOS command window to use them. Click on "*Run...*" in the "*Start*" menu and type **cmd** in the newly opened dialog box. You might have to type **command** instead of **cmd** depending on your Windows version.

Run the programs by typing their name followed by the arguments they require. In most cases, the programs take one argument which is the name of a control file (see later sections for details of these files). You can type, for example:

```
mcmcphase rna-phase-3/example/control/mcmcphase/hiv6-HKY85I.ctl
```
[1]

If you have not added the **PHASE** `bin` directory to your path, as recommended in the **Installation** section, you must move to the `bin` directory to execute the program. On Unix systems you may need to prefix the program name with "`./`".

[1] Note that the use of the \ or / characters is dependent on your operating system.

Using Programs in PHASE

Inputs and outputs in PHASE

Data file format

All molecular sequence data used by the programs in **PHASE** are stored in a common format. The data file format is similar to the **PHYLIP** data file format but with a few minor modifications.

A data file is divided into four sections where two of them are optional in some cases. Comments can be included by starting the line with a hash (#) symbol. The example data files in the package (`*.dna` and `*.rna` in the `example/equence-data` directory) will make the following explanations easier to understand.

Header row

The first non-comment section of the data file must be a single line containing:

1. the number of species
2. the length of the alignment
3. a code which can be either:

- **DNA** or **PROT** for molecular sequences with amino acids or nucleotides that are assumed to be independent
- **CODON** for protein-coding nucleotide sequences with triplet of nucleotides that will be analyzed with a codon or an amino acid substitution model (after translation)
- **STRUCT** for the rest (e.g. for base-paired nucleotides or for combination of the three mentioned types)

The purpose of **STRUCT** is to indicate that a structure mask (see below) is present in the data file. When the **STRUCT** token is used the parser will read and use the structure provided in your datafile to process your molecular sequences. With **DNA** or **PROT**, all the nucleotides/amino-acids are considered as independent entities. With **CODON**, nucleotides are processed triplet by triplet and consequently your alignment length must be a multiple of three.

The line,

```
5 100 DNA
```

at the beginning of a data file indicates that there are five non-base-paired nucleotide sequences of length 100 in the file. With,

```
10 105 CODON
```

PHASE will expect 10 sequences of $105/3 = 35$ codons.

Structure mask

The second section of the data file is the structure mask. A structure mask is required whenever sites cannot be all considered independently (or all as triplets). This mask is directly associated with the **STRUCT** code (i.e. you must not use one without the other). The information provided by the structure-line is used to group related nucleotides together (i.e. by pairs and triplets). Consequently, you should be using a structure when your alignment contains base-paired nucleotides and/or when it contains codons mixed with unpaired nucleotides.

The structure mask is a sequence of special characters whose length is equal to the length of the alignement (i.e. the common length of each sequence). You can put the entire structure line without label between the first line described previously and your first sequence. If you are using the interleaved format (see below), you can also interleave the structure with the sequences.

The structure mask consists of dots (unpaired nucleotides), parentheses (pairs) or "123" (codons). Unpaired sites (nucleotides or amino acids) are indicated with a dot "." or a hyphen "-". Parentheses "()" indicate that the bases at those positions in the sequence form a base-pair in the RNA secondary structure. **123** is used to group three successive nucleotides together and form a codon.

Here is an example:

```
2 10 STRUCT
       (((.(.))))
Mouse  ACCAGAUGGU
Rat    CCCAGUUGGG
```

The mouse sequence ACCAGAUGGU with a structure mask (((.(.)))) indicates that the sequence is made of the base-pairs A:U,C:G,C:G,G:U and two unpaired sites. The structure mask is shared among sequences.

PHASE can handle pseudoknots that are indicated with different pairs of brackets "[]", "{}", "<>", and pairs of upper and lower case letters "Aa", "Bb",... For example, GGGACUCCGU with the structure ((.[[.))]] or equivalently ((.AA.))aa will correctly produce the pairs G:C,G:C,A:U,C:G. Please note the "uppercase then lowercase" order used by **PHASE** when letters are included in this structure. The "lowercase then uppercase" order has a different meaning and you probably should not use it: if a is encountered first then it is paired with the first following A, the second a is paired with the second A, and so on.

Please note that the structure mask contains no information about which model should be used for each position. Obviously, the two elements of a pair or the three nucleotides of a codon must be assigned to a unique model able to handle doublets or codons, but it is possible to use different substitution models for different pairs (or codons), e.g. if they are not from the same gene.

Molecular sequences

The third section of the data file contains the molecular sequences. The standard single-letter code for nucleotide bases is recognized by the software; gaps (-) and ambiguities (e.g. purine (R), pyrimidine (Y), unknown (N, X or ?)) are allowed. At the moment, gaps are treated as ambiguities.

Sequences can be written in one of two formats. The first is the *non-interleaved* format described here. The *interleaved* format is described later. The *non-interleaved* format consists of an identifying label for each sequence followed by the whole sequence. For example:

```
2 16 DNA
Mouse  ACCGUGGU
       UCCAUAAA
Rat    ACUGUGGC
       UCGAUAUA
```

There can be **no spaces in the label**, though the sequence itself can be formatted into blocks using multiple lines and spaces. There is a limit of 50 characters in species labels, and it is important to leave at least one space (or end-of-line character) between the label and the sequence.

Class section

The fourth section is not compulsory and is used when performing a combined analysis of heterogeneous data sets (e.g. loops and stems of a RNA molecule, concatenated data of different genes with different evolutionary patterns or three codon positions). This section is relevant when the **MIXED** model is used, and is not needed when using only one nucleotide, base-pair, codon or amino acid substitution model.

The aim of the class section is to assign each nucleotide/pair/triplet to a class. Each class is expected to have a different pattern of evolution. Classes are subsequently used to determine the model of sequence evolution **PHASE** will use with each site: each class in the data file is treated by its own model of substitution during the phylogenetic inference.

The class section consists of a sequence of integers which correspond to the class of each nucleotide. It is found after the sequences at the end of your datafile but it is possible to interleave the line with the sequences when using the *interleaved* format. If you intend to study protein-coding nucleotide sequences with two distinct 4-state models for the first and second codon positions and a RY-model for the third codon position, this class line should look like this:

```
1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 ...
```

The first and second models used in the **MIXED** substitution model will be standard 4-state models and the third model will be the RY-model (**TWOSTATE**).

When concatenating LSU and SSU rRNA genes, one can treat them as a single gene but it is also possible to use different substitution models for them. In such a case, the class line is compulsory and the full datafile may look like this:

```
59 3247 STRUCT
     .(((....)))... ... ..((((()))))..((.))
sp1  AGGGAAAACCCAAA ... CCAAAAUUUUCCUUCAA
sp2  AGAGAGGACUCAAG ... CCGGAAUUCUCCUUCAA
...                 ...
sp59 AGGGAAAAUCUAAA ... CCAGAAUUCUCCUUCAA
     1 3 3 3 1 1 1 1 3 3 3 1 1 1 ... 2 2 4 4 4 4 4 4 4 4 4 2 2 4 4 2 4 4
```

The first and second model used in the **MIXED** substitution model will be standard 4-state models (for unpaired nucleotides) and the third and fourth model will be base-paired models.

When the data file contains a class section, programs in the **PHASE** package expect it to comply to the following set of rules:

- classes are labelled from 1 to K, where K is the number of distinct classes
- the corresponding **MIXED** substitution model is made with K models
- numbers are separated by a space
- the number of labels equals the length of the sequences
- when there are pairs in the structure, the two nucleotides of a pair must be in the same class (if applicable, the three nucleotides of a codon must be in the same class too)

Class section (automatic)

Since **PHASE** is specifically designed for the analysis of RNA sequences with secondary structure, the most common use of the class section should be the obvious separation of unpaired and base-paired sites into two distinct classes. For such simple cases, you do not have to produce a class section, please refer to the **DATAFILE** block section, below, to learn how sites can be assigned to a substitution model of the **MIXED** model automatically. When the automatic assignment is used, unpaired sites are attributed to class 1, paired-sites are attributed to class 2 and triplets are attributed to class 3 (or class 2 if you do not have any paired sites). You have to make sure the substitution models in the **MIXED** model will match this automatic assignment (e.g. the **MODEL1** and **MODEL2** of your **MIXED** model must be a DNA and a RNA substitution model, respectively, if your structure mask contains unpaired nucleotides and pairs).

Interleaved format

An alternative way of specifying the sequences is to use the *interleaved* format. This enables the sequences to be split into homologous blocks. The non-interleaved example given above could equivalently be written:

```
2 16 DNA
Mouse  ACCG
Rat    ACUG
Mouse  UGGUUCCAUAAA
Rat    UGGCUCGAUAUA
```

When using the *interleaved* format, you are still allowed to arrange your sequences by inserting spaces, but a new line is interpreted as the next species. Within a block, all sequences must have the same length. Constraints on species label are the same (at most 50 characters and there must be a space between the label and the sequence). Note that only the first block requires species labels and you do not have to repeat them in each subsequent block. Whether labels are repeated or not, species must be provided in the same order in each block.

The structure mask and the class line can be interleaved with the sequences (refer to the **DATAFILE** block section for details). This is not compulsory and by default **PHASE** will still read the structure line before the sequences and the class line after the sequences. When these two special sequences are interleaved with the others, you do not have to give them a label, though it will be tolerated. However, note that this label must be between 4 and 20 characters and not contain any digits or structure-specific characters.

```
2 16 STRUCT
struc  ....
Mouse  ACCG
Rat    ACUG
class  1 1 1 1
struc  123123123123
Mouse  UGGUUCCAUAAA
Rat    UGGCUCGAUAUA
class  2 2 2 2 2 2 2 2 2 2 2 2
```

Control file format

Most programs in the package use a control file. The purpose of this file is to initialize properly each program (i.e. sequences, substitution models, and other specific parameters). Control files are the key to using the software and two subsequent sections are devoted to them.

Tree file format

PHASE can output trees, and some programs require a file with one or more trees. A tree file is simply a file with one or more phylogenies written in a computer readable format. This format is basically the Newick format with some restrictions. **PHASE** does not accept labels for internal nodes and will not appreciate having a branch length associated with the root. Some programs in the **PHASE** package can use multifurcating trees (e.g. `optimizer`) but there are cases where they are inappropriate (e.g. with `mcmcphase`) and will be refused. For most algorithms, the root position is irrelevant and programs that can only handle unrooted trees will usually unroot a rooted tree using an user-specified outgroup.

Substitution model parameters file format

With a model parameters file, one can provide initial values for the parameters of the substitution models used. **PHASE** also creates a model parameters file to store the results concerning a substitution model after a run; these could be Maximum Likelihood Estimate (MLE) parameters or Mean Posterior Estimate (MPE) parameters.

Model parameters file content

The content of this file is highly dependent on the substitution model used and we can only describe it in general terms. The fields used to assign a value to each parameter are hopefully quite self-explanatory as long as you know the underlying substitution model. **PHASE** uses a rate ratio parameterization: one of the exchangeability parameters is fixed to 1.0 and each *"Rate ratio i"* parameter in this file stands for the corresponding parameter α_i in the transition matrix of the corresponding model. Transition matrices for all implemented substitution models are given in subsequent sections.

Producing a model parameters file

Model parameters files and control files share the same structural elements. Although it is quite easy to understand the content of a model parameters file without explanations when reading it, you might find it harder to produce your own file from scratch if you want to initialise a substitution model with specific values. It is possible to use the `simulate` program to generate a stub of this file for each model implemented in the **PHASE** package. This skeleton can be modified easily to suit your needs.

Parameters displayed on the screen and output of each program

Each program in the package will output information on the screen, and in one or more files to store the results permanently. The outputs will be reviewed individually in the corresponding program sections. **PHASE** outputs on the screen two kind of matrix:

- a "rate ratios" matrix R
- a transition matrix Q

These matrices are described in the **Transition matrices** section. Other parameters should have a straightforward meaning. **PHASE** uses a rate ratio parameterization: one of the exchangeabilities is fixed to 1.0 as a reference.

Clusters file format

Sometimes it can be useful to specify some monophyletic clusters to reduce the number of possible topologies in the search space. These "weak" constraints on the topology can save a lot of time in both the ML and Bayesian framework. With cluster constraints, `mlphase` will not waste time optimizing unlikely trees, and the topology proposals of `mcmcphase` will be restricted to a limited set of plausible moves, increasing the acceptance rate and improving the mixing. Another possible use of these clusters is obviously to force some "ground-truth" clades not favored by the likelihood criterion. Clusters can also be used to fix the position of the root in some MCMC analyses with rooted trees.

A clusters file contains a list of monophyletic groups and/or topologies (in Newick format). Consider the following definitions for 7 species:

```
cluster1 Species5 Species6 Species7;
topo2 (Species4,(Species3,Species2));
```

The first line specifies one clade with three species Species5+Species6+Species7, the second line specifies 2 clades: Species3+Species2 and Species4+Species3+Species2. The names used are not very important but you might refer to them when specifying an outgroup.

Control files

Most programs in the **PHASE** package have their options set using a simple text file. We call this file the `control-file`. Although the content of this file may differ for each program in the package, its structure remains the same. Some control files are provided as examples with the package (in the `control` directory). The easiest and safest way to use **PHASE** is probably to copy one of these examples and to adapt it to your needs.

Structure of the control files

A control file contains logical blocks (e.g. **DATAFILE** block, **MODEL** block,...) and control lines. Lines preceded by a hash (#) symbol are considered comments and ignored. Comments can be placed anywhere.

A control line is used to define a parameter and gives it a value. It has the format:

```
label = value
```

The order in which control lines are provided in the control file is not important but they must appear in the correct block. Note that **PHASE** is case sensitive: "Tree **f**ile" and "Tree **F**ile" are two different labels. Warnings or errors will be displayed if the user mistypes a parameter.

A block is a container for control lines and other blocks. The block *BLOCKNAME* begins with the tag:

```
{BLOCKNAME}
```

and ends with the tag:

```
{\BLOCKNAME}
```

Tags must be put alone on their line. By convention the name of blocks are all uppercase.

In the remainder of this document, parameters of the control files are coloured depending on their status. *Compulsory parameters* are in *red* and you must provide a value for them. *Optional parameters* are in *green* and they do not need to appear in the control file. Often, a default value will be assumed for optional parameters. Some fields are dependent on the presence and/or values of other parameters, and their presence (or absence) is compulsory under certain conditions. These *conditional parameters* are in *orange*.

Datafile block

Almost all programs in the **PHASE** package require a **DATAFILE** block to parse analyzed sequences. The **DATAFILE** block begins with the tag **{DATAFILE}** alone on a line and ends with the tag **{\DATAFILE}** alone on a line. The **DATAFILE** block contains some basic information:

- *Data file*: the location of the molecular sequence file to be used.

  ```
  Data file = sequence-data/sequences.dna
  ```

- *Interleaved data file*: a yes/no option that specifies whether the molecular data is interleaved.

  ```
  Interleaved data file = yes
  ```

- *Interleaved structure*: a yes/no option that specifies whether the mask and the class line are interleaved. This field does not make sense if you used the *non-interleaved* format for your molecular sequences. The default value is **no**.

  ```
  Interleaved structure = no
  ```

- *Heterogeneous data models*: a yes/auto/no parameter which specifies whether the data file contains a class section. The default value is **no** and consequently the class section of your data file will be ignored if you forget this field. If you answer **yes**, **PHASE** will look for the class line in your dataset. If you answer **auto**, **PHASE** will generate internally the class line from the structure line: unpaired sites are attributed to class 1, pairs are attributed to class 2, codons are attributed to class 2 or class 3 (class 2 is there is no pair, class 3 otherwise). This automatic assignment was only designed to handle the most common cases. When concatenating genes you might want to use substitution models that are a bit more complex and use more than one model. For these cases, the automatic method cannot handle your needs and you have to use a class line in your datafile.

  ```
  Heterogeneous data models = yes
  ```

Model block

Most programs in the **PHASE** package require the specification of a substitution model for sequence evolution. This is the purpose of the **MODEL** block. The **MODEL** block is delimited by the **{MODEL}** and **{\MODEL}** tags. It contains the name of the substitution model followed by parameters (and sometimes blocks) specific to the model. We describe here the main components of this model block.

"Simple" substitution models

Depending on the data to be analyzed, the **PHASE** package can be used with a wide variety of DNA substitution models or RNA-specific base-paired models. Also available are a 3-state model for AGY analysis, and a 2-state model for 0/1 or RY analysis (please do not recode your data in the latter cases, the software can convert from standard DNA sequences).

The content of the **MODEL** block is quite similar for all these models and the parameters are:

- *Model*: the name of the model, by convention it is always upper case. Nucleotide substitution models implemented include JC69, K80, F81, HKY85, T92, TN93, REV. Base-paired substitution models implemented include RNA6A, RNA6B, RNA7A, RNA7B, RNA7C, RNA7D, RNA7E, RNA7F, RNA7G, RNA16A, RNA16B, RNA16C, RNA16D, RNA16E, RNA16F, RNA16I, RNA16J, RNA16K. Other options are TWOSTATE and THREESTATE.

  ```
  Model = REV
  ```

- *Discrete gamma distribution of rates*: the discrete gamma model can be used to account for among-site rate variation. Use yes/no values to turn this option on/off. When a discrete gamma model is used, **PHASE** expects the number of gamma categories to be specified. By default the discrete gamma model is not used.

  ```
  Discrete gamma distribution of rates = yes
  ```

- *Number of gamma categories*: when the discrete gamma model is used, you have to provide an integer to specify the desired number of discrete gamma categories.

  ```
  Number of gamma categories = 5
  ```

- *Invariant sites*: alternatively, or in conjunction with the discrete gamma model, the user can allow a proportion of sites to be invariant, i.e. with zero rate of evolution. The default value is **no**.

  ```
  Invariant sites = yes
  ```

Mixed model for combined analyses of heterogeneous data

Several models are sometimes required when studying heterogeneous sequences. The **MIXED** model allows these models to work concurrently.

- *Model*: this field contains the name of the model, which must be **MIXED**.

  ```
  Model = MIXED
  ```

- *Number of models*: the number of models used concurrently. If a class section was provided with the data file then the number of models should be the same as the number of classes. If you used the automatic class selection derived from the structure mask then this parameter has to correspond with the behaviour of the automatic assignment described in the **DATAFILE** block section.

```
Number of models = 3
```

- *{MODELi} block*: each model used in the mixed model must be defined in its own block. If the number of models is n then the **MODEL** block must contains n blocks whose name are **MODEL1**, **MODEL2**,... **MODELn**. The content of these blocks is the content described previously for standard substitution models. Each model is used with the corresponding class section in the datafile.

Here is an example:

```
{MODEL}
  Model = MIXED
  Number of models = 2
  {MODEL1}
    Model = REV
    Discrete gamma distribution of rates = yes
    Number of gamma categories = 5
  {\MODEL1}
  {MODEL2}
    Model = RNA7A
  {\MODEL2}
{\MODEL}
```

Time-heterogeneous model

Time-heterogeneous methods can only be used with the program `mcmcphase` during a Bayesian analysis. Heterogeneity is modelled using a limited number of substitution models (user-specified at the moment). Each branch is assigned one of these models and some MCMC proposals are designed to change this assignment. The **HETEROGENEOUS** model allows these models to work concurrently. Please note that this model remains somewhat experimental.

All substitution models (**MIXED** model included) can be "transformed" into a **HETEROGENEOUS** model by inclusion in a **BASEMODEL** block:

- *Model*: this field contains the name of the model, which must be **HETEROGENEOUS**.

  ```
  Model = HETEROGENEOUS
  ```

- *Number of models*: the number of models used concurrently. See Foster [2004] for statistical testing techniques. We have not implemented reversible jump MCMC, so this has to be a fixed number.

  ```
  Number of models = 3
  ```

- *{BASEMODEL} block*: The content of this block was described previously. This is the **MODEL** block of a standard or **MIXED** substitution model.

Here is an example:

```
{MODEL}
  Model = HETEROGENEOUS
  Number of models = 3
  {BASEMODEL}
    Model = REV
    Discrete gamma distribution of rates = yes
    Number of gamma categories = 5
  {\BASEMODEL}
{\MODEL}
```

You must not use an invariant model inside a heterogeneous model unless you initialize and tune the MCMC proposals so that the proportion of invariant sites is a shared constant fixed *a priori* (there can be only one single parameter for the proportion of invariant sites).

Tree block

Most programs in the **PHASE** package will expect a **TREE** block. Its content is quite program-specific and it will therefore be described later when appropriate.

Using the programs in the PHASE package

Each program in the **PHASE** package requires a specific control file, the content of which is described here. As in the previous section, *compulsory parameters* appear in *red*, *optional parameters* in *green* and *conditional parameters* dependent on the presence and/or value of others are in *orange*.

mcmcphase

Using *mcmcphase*

The `mcmcphase` program performs Bayesian estimation of phylogenies and uses Markov chain Monte-Carlo techniques (MCMC) to produce large samples from the posterior probability density. To use `mcmcphase`, simply type at the command-line:

```
mcmcphase mcmcphase-control-file
```

where `mcmcphase-control-file` is a valid control file for the `mcmcphase` program. If an MCMC run is aborted before it completes, then you can restore it using the same command. If **PHASE** detects `.rsb` and `.rsc` files, it will prompt you to restore the run rather than starting a new one. We strongly advise you to backup these two files before restarting a run: if the run is aborted again just after restarting, there is a chance that they will get damaged preventing further recovery.

The section on possible issues and pitfalls with MCMC techniques, below, describes known issues with Bayesian methods and **PHASE** specifically.

The `mcmcphase` program saves results in many files. Be aware that it might require a large amount of disk space for large studies, depending on the evolutionary model that you are using.

- `*-best.tre` and `*-best.mod` files: the phylogeny and the parameters of the substitution model when the best state (i.e. the state with the highest likelihood) was visited. The best configuration is not very important in a MCMC analysis but a "strange" best state will be a visible sign that something went wrong. The best tree and the best model can also be used as starting points for subsequent phylogenetic analyzes.
- `*.mp` file: the file with the sampled parameters of the substitution model(s). Each sample occupies one line. The parameters are model-dependent, you should be able to figure out which is which without too much trouble. For a standard substitution model, parameters appear in this order:
  - the proportion of invariant sites if an invariant category is used (+I models)
  - the gamma shape parameter (α) if the discrete gamma model is used (+dG models)
  - the frequencies of the states as they appear in the substitution matrix
  - the exchangeability parameters

When a more complex model is used (e.g. **MIXED** or **HETEROGENEOUS**), substitution parameters for each model are printed sequentially. Except for the first model, each set of parameters should be preceded by the average substitution rate of the model. The average substitution rate for the first model is always 1.0 and therefore this value is not reported. Ancestral frequencies for the **HETEROGENEOUS** model are reported at the beginning, before the first model.

- `*.smp` file: the sampled topologies. To avoid wasting disk space, `mcmcphase` will output the sampled topologies using indices rather than species name. Species are numbered according to their appearance order in the datafile.

- `*.bl` file: the branch lengths for the previous topologies (for use with other **PHASE** programs).
- `*.bm` file: the substitution model for each branch (**HETEROGENEOUS** model only).
- `*.out` file: a file with similar content to the screen output.
- `*.plt` file: the evolution of the likelihood, prior on tree and prior on substitution model parameters during the run. Sampling of these values starts at the beginning of the run (i.e. they are stored during the burn-in too).
- `*.hpm` file: the hyperparameters used for the prior on the substitution model (if any).
- `*.hpt` file: the hyperparameters used for the prior on branch lengths and topology (if any).

Control file for *mcmcphase*

Example control files are provided in the appendix. Please remember that **PHASE** is case-sensitive and will not be tolerant of spelling mistakes. To prevent mistakes, the program aborts with an error message (hopefully explicit) if requirements are not met. Unused control-lines must be removed or commented with a # otherwise the program will refuse to start (in most cases superfluous lines are caused by typing mistakes on optional parameters and you have to correct them to prevent the software from using the default option). In the control file of `mcmcphase` the compulsory and optional sections are:

- a ***DATAFILE** block*: see the data file block section.
- a ***MODEL** block*: see the model block section. Some fields specific to `mcmcphase` are added to this **MODEL** block:
  - *Starting model parameters file*: to reduce the necessary "burn-in" time, the chain can be initialized with some user-specified model parameters. Otherwise the sequences are used to initialize the substitution model. For more information on this file, see the substitution model file format section.
- a ***TREE** block*: see the tree block section for `mcmcphase`.
- a ***PERTURBATION** block*: for the mixing properties of `mcmcphase` (see below).
- *Random seed*: the seed for the random number generator. System time is used if not specified.
- *Burnin iterations*: the number of "burn-in" cycles (i.e. cycles before starting the sampling). During the "burn-in", only likelihood values are stored. It is good practice to check that the likelihood reached a plateau before the end of the "burn-in" (in the `.plt` file) but you should also check that substitution parameters reached equilibrium at the beginning of the sampling (in the `.mp` file). Note that **PHASE** adjusts proposal parameters during the "burn-in" and aims at reasonable acceptance rates to improve mixing properties during sampling. It is consequently slightly better to use a longer "burn-in" rather than discarding samples once the run is finished.
- *Sampling iterations*: the number of cycles for sampling. Ten times the number of iterations needed to reach convergence is a rough rule of thumb. We strongly encourage you to repeat runs (with different random seeds) and compare the results. Clade support values should be almost the same (+/- 4% should be reasonnable in most cases).
- *Sampling period*: the number of cycles between extraction of two consecutive samples.
- *Output file*: the basename for all the output files (e.g. `basename-best.tre`, `basename-best.mod`, `basename.mp`).
- *Output format*: the format used for the topologies in the `.smp` file, it can be **phylip** (with a semi-colon at the end) or **bambe** (without semi-colon).

TREE block for *mcmcphase*

The ***TREE*** *block* contains information on the tree you wish to use:

- *Tree*: the main field has four possible options:
  - "Unrooted MCMC tree": the standard tree you should probably use.
  - "Rooted MCMC tree with molecular clock": tree are constrained to be ultrametric (molecular clock assumption). The position of the root matters and consequently a rooted tree is used.
  - "Heterogeneous MCMC tree": when the substitution process is not stationary, the position of the root matters. The tree used with heterogeneous models is consequently rooted. Use with the **HETEROGENEOUS** model.
  - "Heterogeneous MCMC tree with molecular clock": equivalent to the "Rooted MCMC tree with molecular clock" but for heterogeneous model. Use with the **HETEROGENEOUS** model.
- *Clusters file*: this parameter constrains the topology space, and if present, the value expected for this field is the name of a clusters file. Specifying the obvious clades (i.e. those with Bayesian Posterior Probability 100%) can improve mixing when a large number of species is used.
- *Outgroup*: the outgroup can be either a species in your datafile or a cluster from your clusters file, if provided. This parameter is compulsory with the "Unrooted MCMC tree" though it is only used to root your tree for the sake of readability. With the three other rooted trees, the outgroup is optional but has a much stronger meaning when used since it constrains the root position to be in the branch leading to the outgroup.
- *Starting tree file*: one can choose to initialize the chain randomly or with a user-defined tree. We do not encourage the use of an initial topology in a standard analysis. Starting several MCMC chains (e.g. 4 chains) from different random initial states to assess convergence is standard. However, this option is useful if you want to perform a Bayesian analysis with a fixed topology or to "continue" a previous run that failed to converge because the required number of iterations was underestimated.

PERTURBATION block

The ***PERTURBATION*** *block* contains the mixing parameters used for the proposals and, because it was convenient, it also contains your definition for the priors. We describe here all the possible options as exhaustively as possible for reference purposes. You should not have a real need for all these parameters for standard use and default values for optional parameters are fine in most cases. Try to adapt an existing control file to suit your needs.

At the top-level of the hierarchy, the **PERTURBATION** block contains:

- a ***PERTURBATION_TREE*** *block*: see below.
- a ***PERTURBATION_MODEL*** *block*: see below.
- *Tree, proposal priority* and *Model, proposal priority*: these two priorities are relative to each other and give the proportions of proposals attempted in the **PERTURBATION_TREE** block and **PERTURBATION_MODEL** block. They have a high impact on the mixing properties. Note that proposals in the model space are usually more expensive from a computational point of view (the average time per iteration will increase) so it is better to keep the ratio of these two values as high as possible as long as the exploration of the parameters space does not suffer from it. As long as the chain is run for long enough, your results do not depend on this choice unless extreme values are chosen. Use the values in the examples as a guide and decrease/increase the tree proposal priority if your model is more/less complex.

PERTURBATION_TREE block

The **PERTURBATION_TREE** block is included in the **PERTURBATION** block and contains mixing parameters relative to branch length proposals and topology proposals:

- *Topology changes, proposal priority*: priority for the topology changes proposal. This priority is given relative to the two following priorities. During a topology change, one of the available proposal (e.g. SPR, NNI for the standard MCMC tree) is attempted. Use **0** to perform an analysis with a fixed topology.
- *Branch lengths, proposal priority*: priority for the branch lengths proposal, it is given relative to the previous priority and the following priority. The moves used to modify branch lengths depend on the type of tree used (unrooted/rooted/ultrametric).
- *Branch lengths, prior*: the prior to be used for branch lengths, compulsory for the trees that are not ultrametric, no optional value is assumed because this prior can have a non negligible impact on your results. Options are **exponential(10)** or **uniform(0,5)**. You can add a level of hierarchy and introduce a hyperparameter. In such a case, the field *Hyperpriors, proposal priority* will become compulsory.
- *Tree height, prior*: the "equivalent" of the branch lengths prior for ultrametric tree, the prior is given on the height of the tree and all sets of branch lengths that match this requirement have uniform probability.
- *Hyperpriors, proposal priority*: priority shared among all the prior parameters of the tree, this parameter is compulsory if you introduced a prior that requires some hyperparameters in the previous fields.

At the moment, the only prior available for tree topology is the uniform prior.

PERTURBATION_MODEL block

The **PERTURBATION_MODEL** block is included in the **PERTURBATION** block and contains mixing parameters related to the substitution model. These parameters depend on the substitution model.

With simple substitution models, i.e. any nucleotide or base-paired model used alone, the block has only one level of hierarchy and proposal priorities are relative to each other. The following parameters are used in the **PERTURBATION_MODEL** block to perturb the frequencies:

- *Frequencies, proposal priority*: an integer value to specify how often we try to perturb the frequencies with respect to other parameters. This parameter is compulsory except for models with fixed frequencies (e.g. **JC69**, **K80** and **EMPIRICAL** amino-acid model without the *+F* option). Use the value **0** to prevent the perturbation (i.e. if you want the frequencies to remain equal to the empirical frequencies or to the values provided in an initial substitution model).
- *Frequencies, prior*: prior on frequencies. This prior can be "dirichlet(x_1,...,x_n)", where n is the number of frequencies of your model, or "dirichlet(x)" (which is equivalent to dirichlet(x,...,x)) or "flat" (the default, equivalent to dirichlet(1)).
- *Frequencies, proposal minimum acceptance rate* and *Frequencies, proposal maximum acceptance rate*: during the "burn-in", `mcmcphase` will try to adapt the proposal step so as to reach an acceptance rate within the specified range. By default this range is [0.20, 0.25]. If you want to change the default values, provide the two parameters. Use the range [0.0, 1.0] to turn off the dynamic adaptation of the proposal step.
- *Frequencies, initial Dirichlet tuning parameter*: the initial proposal parameter (the higher the Dirichlet parameter, the lower the step). Remember this is just an initial value which will change during the "burn-in" unless you turned off the dynamic adaptation of the step (in which case this parameter is compulsory). Allowed values are in the range $[100.0, 1e10]$ and the default value is 1000.0.

Similar parameters are used for the rate ratios:

- *Rate ratios, proposal priority*: an integer value to specify how often we try to perturb each rate ratio with respect to other parameters. Rate ratios are treated individually but they share the same priority. This parameter is usually compulsory except for models with fixed rates (e.g. **JC69** or amino-acid models). You can use the value **0** to have constant rate ratios but you should probably not do that unless you provide initial parameters with a "`.mod`" file.

- *Rate ratios, prior*: prior for the rate ratios. The default prior is **uniform(**.0005**,** 2000.0**)**. **PHASE** does not propose a Dirichlet prior for the rates. You can use all the priors **PHASE** proposes for single parameter (uniform(a,b), exponential(λ), normal(μ,σ), lognormal(γ,β,θ), gamma(γ,β,θ))

- *Rate ratios, proposal minimum acceptance rate* and *Rate ratios, proposal maximum acceptance rate*: rate ratios are treated individually but they share the same range for the acceptance rate. The default range is [0.20, 0.25] and you can turn off the dynamic modification of the step with the range [0.0, 1.0].

- *Rate ratios, initial step*: allowed initial values are in the range [0.005, 10.0]. The default value (when the dynamic step option is on) is **0.2**. You cannot provide a different initial step for each rate ratio.

The following parameters are appropriate when using a +dG (discrete gamma categories) or/and a +I (invariant category) model:

- *Gamma parameter, proposal priority*: compulsory if a gamma model is used

- *Gamma parameter, prior*: uniform, exponential, ...; default value is **uniform(0,1000.0)**

- *Gamma parameter, proposal minimum acceptance rate*: default value = **0.20**

- *Gamma parameter, proposal maximum acceptance rate*: default value = **0.25**

- *Gamma parameter, initial step*: default value = **0.2**

- *Invariant parameter, proposal priority*: compulsory if an invariant model is used

- *Invariant parameter, prior*: uniform, exponential, ...; default value is **uniform(0,1.0)**

- *Invariant parameter, proposal minimum acceptance rate*: default value = **0.20**

- *Invariant parameter, proposal maximum acceptance rate*: default value = **0.25**

- *Invariant parameter, initial step*: default value = **0.05**.

PERTURBATION_MODEL block for MIXED models

When the **MIXED** model is used, the **PERTURBATION_TREE** block remains unchanged but the **PERTURBATION_MODEL** block is slightly different. For each substitution model in the **MIXED** model, you must have distinct *PERTURBATION_MODELi blocks* in the **PERTURBATION_MODEL** block. Parameters which are specific to the **MIXED** model will appear at the top-level of the **PERTURBATION_MODEL** block.

- *Model* i*, proposal priority*: specify a priority for each model (relative to the priority of other models and the priority used for the proposal of new average rates). We recommend giving a priority approximately proportional to the number of free parameters in the substitution model.

- *Average rates, proposal priority*: specify a proposal priority to modify the average substitution rate of each model (except the first one, since its average substitution rate is fixed to 1.0). This priority is relative to the previous ones.

- *Average rates, prior*: uniform, exponential, ...; default value is **uniform(0.005,200.0)**. **PHASE** does not propose a Dirichlet prior on the average substitution rates.

- *Average rates, proposal minimum acceptance rate* and *Average rates, proposal maximum acceptance rate*: the acceptance range `mcmcphase` aims for, [0.20, 0.25] by default.

- *Average rates, initial step*: allowed initial values are in the range [0, 1.0]. The default value (when the dynamic step option is on) is **0.2**.

Here is an example with the **MIXED** model and three substitution models: JC69+dG, RNA7A+dG, TN93+I.

```
{PERTURBATION}
  Tree, proposal priority = 7
  Model, proposal priority = 1
  {PERTURBATION_TREE}
    ...
  {\PERTURBATION_TREE}
  {PERTURBATION_MODEL}
    Model 1, proposal priority = 2
    Model 2, proposal priority = 24
    Model 3, proposal priority = 7
    Average rates, proposal priority = 1
    {PERTURBATION_MODEL1}
      Gamma parameter, proposal priority = 1
    {\PERTURBATION_MODEL1}
    {PERTURBATION_MODEL2}
      Frequencies, proposal priority = 3
      Rate ratios, proposal priority = 2
      Gamma parameter, proposal priority = 1
    {\PERTURBATION_MODEL2}
    {PERTURBATION_MODEL3}
      Frequencies, proposal priority = 3
      Rate ratios, proposal priority = 2
      Invariant parameter, proposal priority = 1
    {\PERTURBATION_MODEL3}
  {\PERTURBATION_MODEL}
{\PERTURBATION}
```

Each cycle, `mcmcphase` will attempt a proposal for the substitution model with probability *"Model, proposal priority" / "Tree, proposal priority"*. If the substitution model proposal is selected, the program will modify the average substitution rate of the second model with probability given in equation 1; the probability of modifying the average substitution rate of the third model is the same. The average substitution rate of the first model is the reference, it is never modified and remains equal to 1.0. With probability given in equation 2, `mcmcphase` will modify the parameters of the substitution model i (i.e. any parameters other than the average substitution rate). The priorities inside the corresponding **PERTURBATION_MODEL*i*** block are used, the figures inside each **PERTURBATION_MODEL*i*** block do not have any effect outside the block.

$$P(average\,rate) = \frac{Average\,rates,\,proposal\,priority}{total\,priority} \quad (1)$$

$$P(model\,i) = \frac{Model\,i,\,proposal\,priority}{total\,priority} \quad (2)$$

$$total\,priority = (N-1) * Average\,rates,\,proposal\,priority + \sum_{i=1}^{N} Model\,i,\,priority\,priority$$

In our example, if the third model is selected for the next modification, we define:

$priority_{total} = priority_{frequencies} + priority_{gamma} + priority_{invariant} + number_{rates\ ratios} * priority_{rate\ ratios}$

and we modify either all the frequencies (probability 3), or one of the rate ratios (probability 4 each), or the invariant parameter (probability 5), or the gamma shape parameter (probability 6).

$$P = \frac{priority_{frequencies}}{priority_{total}} \quad (3)$$

$$P = \frac{priority_{rate\ ratios}}{priority_{total}} \quad (4)$$

$$P = \frac{priority_{invariant}}{priority_{total}} \quad (5)$$

$$P = \frac{priority_{gamma}}{priority_{total}} \quad (6)$$

PERTURBATION Block for HETEROGENEOUS models

The perturbation block for the **HETEROGENEOUS** model is similar to the perturbation block for the **MIXED** model. A unique **PERTURBATION_BASE_MODEL** block contained in the **PERTURBATION_MODEL** block is shared among substitution models. The content of the **PERTURBATION_BASE_MODEL** block is as described above, since its content is strictly the same as a **PERTURBATION_MODEL** block for standard or **MIXED** substitution models. Extra parameters are found at the top level of the **PERTURBATION_MODEL** block to set up the **HETEROGENEOUS** model:

- *Model, proposal priority*: the priority for a proposal on a specific internal substitution model (as a guideline, you can probably use the number of free parameters in the model).
- an ***ANCESTRAL_FREQUENCIES*** *block*: **HETEROGENEOUS** models use a rooted tree and have an ancestral frequency distribution at the root. This block contains the parameters required for the mixing (and priors) of ancestral frequency parameters. When the **MIXED** model is used as a "base_model", there should be an **ANCESTRAL_FREQUENCIES*i*** block for each substitution model. The content of this block uses the same frequency parameters as described for the **PERTURBATION_MODEL** block.

Here is a complex example for a heterogeneous DNA + 7-state RNA model.

```
{PERTURBATION}
  Tree, proposal priority = 2
  Model, proposal priority = 1
  {PERTURBATION_TREE}
    ...
  {\PERTURBATION_TREE}
  {PERTURBATION_MODEL}
    Model, proposal priority = 30
    Average rates, proposal priority = 0
    {ANCESTRAL_FREQUENCIES1}
      Frequencies, proposal priority = 1
      Frequencies, initial Dirichlet tuning parameter = 2000
    {\ANCESTRAL_FREQUENCIES1}
    {ANCESTRAL_FREQUENCIES2}
      Frequencies, proposal priority = 1
      Frequencies, prior = dirichlet(2,1,2,2,1,2,1)
    {\ANCESTRAL_FREQUENCIES2}
    {PERTURBATION_BASEMODEL}
      Model 1, proposal priority = 6
      Model 2, proposal priority = 24
      Average rates, proposal priority = 1
      {PERTURBATION_MODEL1}
        ...
      {\PERTURBATION_MODEL1}
      {PERTURBATION_MODEL2}
        ...
      {\PERTURBATION_MODEL2}
    {\PERTURBATION_BASEMODEL}
  {\PERTURBATION_MODEL}
{\PERTURBATION}
```

Priors

Priors have been mentioned during the description of the **PERTURBATION** block. We will summarize the priors available in **PHASE** here.

The standard (and only) prior available for n frequency parameters $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ is a Dirichlet distribution:

$$f(\pi|\alpha) = \frac{\Gamma(\sum_{i=1}^{n} \alpha_i)}{\prod_{i=1}^{n} \Gamma(\alpha_i)} \prod_{i=1}^{n} \pi_i^{\alpha_i - 1} \tag{7}$$

where α=$\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ with $\alpha_i > 0$ parameters specifying the distribution. $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is the mean of the distribution (after normalization). The mean does not change if α is multiplied by a constant but the variance gets smaller as α_i values grow. $\alpha_i = c, \quad \forall i$ is a special case of the distribution you can use with **PHASE**. $\alpha_i = 1, \quad \forall i$ is the commonly used "uninformative" flat prior. For a model with 5 frequencies, you can use:

```
*, prior = dirichlet(2,2,2,2,2)
```

or

```
*, prior = dirichlet(2)
```

For independent parameters, you can use:

- the uniform distribution, **uniform**(**a**, **b**):

$$f(x|a,b) = \frac{1}{b-a} \quad a \leq x \leq b \tag{8}$$

- the exponential distribution, **exponential**(λ) ($\mu = 1/\lambda$, $\sigma = 1/\lambda$):

$$f(x|\lambda) = \lambda e^{-\lambda x} \quad x > 0 \tag{9}$$

- the normal distribution, **normal**(μ, σ):

$$f(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{10}$$

- the lognormal distribution, **lognormal**(γ, β, θ), where $\beta = 1$ and $\theta = 0$ by default:

$$f(x|\gamma,\beta,\theta) = \frac{\exp\left[-\frac{(\ln((x-\theta)/\beta))^2}{2\gamma^2}\right]}{(x-\theta)\gamma\sqrt{2\pi}} \quad x > \theta; \quad \gamma > 0; \quad \beta > 0; \tag{11}$$

- the gamma distribution, **gamma**(γ, β, θ), where $\beta = 1$ and $\theta = 0$ by default:

$$f(x|\gamma,\beta,\theta) = \frac{(\frac{x-\theta}{\beta})^{\gamma-1} \exp\left[-\frac{x-\theta}{\beta}\right]}{\beta\Gamma(\gamma)} \quad x > \theta; \quad \gamma > 0; \quad \beta > 0; \tag{12}$$

Parameters of these prior distributions can become hyperparameters, for example:

```
*, prior = exponential(uniform(0,50))
```

When hyperparameters are introduced, you have to specify a proposal priority for them. Launch `mcmcphase` without specifying them and the missing parameters that were expected will be printed on the screen).

Using *mcmcsummarize*

The `mcmcsummarize` program is used to exploit the large sample produced by `mcmcphase`. The program builds a consensus tree from the output of `mcmcphase`, and computes mean posterior estimates for substitution model parameters. To use `mcmcsummarize`, simply type at the command-line:

```
mcmcsummarize mcmcphase-control-file
```

where `mcmcphase-control-file` is the control file that was used by the `mcmcphase` program to produce the results.

`mcmcsummarize` constructs a majority-rule consensus topology. All clusters with Bayesian posterior probability (BPP) higher than 50% are represented in the consensus tree. We are actually using the "extended majority rule" (as does `consense` in **PHYLIP**) and clusters whose BPP is lower than 50% are added sequentially as long as they do not contradict the clades established previously. Clade values are the Bayesian posterior probabilities of species bipartitions when the standard unrooted tree is used (i.e. no molecular clock and no **HETEROGENEOUS** model). When a tree with explicit root positioning is used, a clade is not considered monophyletic if it is "torn apart" by the root.

`mcmcsummarize` will also attempt to produce branch lengths for the consensus topology found at the previous step. Beware when interpreting these distances. For a standard MCMC analysis with an *"Unrooted MCMC tree"*, branch lengths returned are the mean posterior estimates of the evolutionary distances between the two complementary clades they define. This mean posterior estimate was computed with the samples where the bipartition was present. For MCMC analyses with ultrametric trees, branch lengths are the mean posterior estimates of the evolutionary distance from the clade ancestor node to the tips (once again computed with the samples where the clade was present). This can sometimes lead to the awkward situation where a clade can be younger than the clades it contains (and result in a negative branch length). For the *"Heterogeneous MCMC tree"*, branches are computed to have MPEs of the distance from ancestral nodes to the root and should probably not even be considered unless the root was fixed during the analysis.

If you need to produce branch lengths for your consensus tree, you should either optimize the consensus tree using ML methods or start an MCMC run with the consensus topology as a fixed tree (start with the consensus topology and turn off topology proposals).

The `mcmcsummarize` program generates a `.cns` file and a `.cnt` file. The `.cnt` contains a list of tree all having the consensus topology but with different branch lengths, and the `.cns` file records the meaning of these lengths (clade BPPs, evolutionary distance, and more for the **HETEROGENEOUS** model). `mcmcsummarize` will also generate two consensus substitution models. They are produced using respectively the mean and median values of the parameters distribution in the `.mp` file.

mlphase

Using *mlphase*

The `mlphase` program can be used to find the phylogeny and, optionally, evolutionary model parameters that yield the maximum likelihood. `mlphase` cannot work with ultrametric trees (molecular clock) and time heterogeneous methods. Topologies are always considered to be unrooted. Four algorithms are provided for topology search:

- Simple exhaustive search
- Branch-and-bound exhaustive search
- Heuristic stepwise addition
- Star-tree decomposition

The branch-and-bound search and the exhaustive search are theoretically guaranteed to return the best tree unless the search gets trapped in a local minimum during the optimization process. However, they cannot handle large datasets. The heuristic stepwise addition and the star-tree decomposition methods are less likely to find the optimal tree but they are computationally tractable for a larger number of taxa. To reduce the search space and the computation time, constraints can be placed on the topologies considered during ML inference. With a clusters file one can specify monophyletic clade topologies which must be preserved during the inference process. The program will look for an optimal topology consistent with these clade arrangements.

To use `mlphase`, type at the command-line:

```
mlphase mlphase-control-file
```

where `mlphase-control-file` is a valid control file for `mlphase`. The `mlphase` program saves the results of an inference in a single file. Results are also displayed on screen during the run.

Control file for *mlphase*

An example of a valid control file for `distphase` can be found in the appendix. The control file of the `mlphase` program contains:

- a ***DATAFILE** block*: see the data file block section.
- a ***MODEL** block*: see the model block section. Some fields specific to `mlphase` are added to this **MODEL** block:
  - *Optimize model parameters*: set this field to **no** if the model parameters are to be fixed. By default, `mlphase` will optimize substitution parameters simultaneously to branch lengths during the inference.
  - *Starting model parameters file*: by default, sequence data is used to initialize the substitution model, but the initial state can also be loaded from a file. This should probably be done when not optimizing the substitution parameters. For more information on this file, see the substitution model file format section.
- a ***TREE** block*: mainly used to specify the search algorithm to be used. It contains the following fields:
  - *Search algorithm*: you can use *"Exhaustive"*, *"Branch-and-bound"*, *"Stepwise addition"* or *"Star decomposition"*
  - *Clusters file*: the name of a clusters file, used to constrain the topology space.
  - *Outgroup*: the outgroup can be either a species in your data file, or a cluster from your clusters file if provided. This parameter is compulsory, though it is only used to root your tree for the sake of readability.
- *Random seed*: the seed for the random number generator. System time is used if not specified.
- *Output file*: the location of a file to store the results.

optimizer

Using *optimizer*

The `optimizer` program is used to compute maximum-likelihood estimates (MLEs) for the branch lengths and, optionally, substitution model parameters of a given model of evolution. Like `mlphase`, `optimizer` works only with time-reversible models (and unrooted non-ultrametric trees). The user provides a fixed tree topology (or a list of topologies) and specify a substitution model with free parameters. One can give initial values for branch lengths and substitution model parameters to speed-up the convergence or to check robustness of the results by repeating a result from different initial states. To use `optimizer`, type at the command-line:

```
optimizer optimizer-control-file
```

where `optimizer-control-file` is a valid control file for the `optimizer` program.

When launched, `optimizer` displays the initial tree and the initial likelihood on the screen and begins the optimization. Once it is finished, the ML substitution model parameters are printed on the screen and saved in the "`.out`" file with the ML tree and the value of the maximum likelihood. The ML tree is also saved in the "`.tre`" file and a "`.mod`" file is created to store the MLEs for the substitution model parameters.

Control file for *optimizer*

`optimizer` is quite similar to `mlphase` and this is reflected in the content of their control files. An example of a valid control file for `optimizer` can be found in the appendix. The control file of the `optimizer` program contains:

- a ***DATAFILE** block*: see the data file block section.
- a ***MODEL** block*: see the model block section. Some fields specific to `optimizer` are added to this **MODEL** block:
  - *Optimize model parameters*: set this field to **no** if the model parameters are to be fixed. By default, `optimizer` will optimize substitution parameters simultaneously to branch lengths during the inference.
  - *Starting model parameters file*: by default, sequence data is used to initialize the substitution model, but the initial state can also be loaded from a file. This should probably be done when not optimizing the substitution parameters. For more information on this file, see the substitution model file format section.
- a ***TREE** block*: mainly used to give the topologies that are to be optimized. It contains the following fields:
  - *Tree file*: the name of the file containing the phylogeny (or phylogenies), in Newick format.
  - *Clusters file*: the name of a clusters file; not used in practice unless you wish to specify an outgroup clade.
  - *Outgroup*: the outgroup can be either a species in your data file, or a cluster from your clusters file if provided. This parameter is compulsory, though it is only used to root your tree for the sake of readability.
- *Random seed*: the seed for the random number generator. System time is used if not specified.
- *Output file*: the location of a file to store the results.

distphase

Using *distphase*

The `distphase` program is used to compute pairwise ML distances between sequences in your dataset. Given any reversible substitution model implemented in **PHASE** (all but the **HETEROGENEOUS** model) and a set of fixed substitution parameters (e.g. known frequencies, exchangeabilities), `distphase` numerically optimizes the distances between pairs of taxa and outputs a distance matrix. This matrix can subsequently be used as an input for a tree reconstruction algorithm, e.g. UGPMA, Neighbour-Joining. See the distance matrix programs in **PHYLIP**.

To use `distphase`, one has to provide a complete substitution model for nucleotide evolution with user-defined parameters. For instance, you can use the `.mod` output of `optimizer` when optimizing the star-tree or an MCMC consensus tree. You can also consider using one of the two consensus models produced by `mcmcphase`. Alternatively, you can use `simulate` to produce the "stub" of a `.mod` file and fill it with your own values.

To use `distphase`, type at the command-line:

```
distphase distphase-control-file
```

where `distphase-control-file` is a valid control file for the `distphase` program.

Control file for *distphase*

An example of a valid control file for `distphase` can be found in the appendix. In its control file, the `distphase` program requires the specification of:

- a ***DATAFILE*** *block*: see the data file block section.
- a ***MODEL*** *block*: see the model block section.
  - *Model parameters file*: The name of the file containing parameter values for the model. The inference programs of the **PHASE** package produce such a file, or one can be generated with the `simulate` program.
- *Output format*: controls how the matrices are written on the screen and in the output file: "lower-triangular", "upper-triangular" and "square" are valid options. When not specified, the default is "square".
- *Output file*: the location of a file to store the results.

likelihood

Using *likelihood*

The `likelihood` program is used to compute the likelihood of a model of evolution (i.e. tree + parameterized substitution model) given a set of studied sequences. `likelihood` is also used for analyzing results of other programs, and can perform ancestral sequences marginal reconstruction and compute posterior probabilities of a site being in a specific rate category. Please note that 7-state RNA models will reconstruct mismatch pairs (MM) and the **TWOSTATE** model can only infer R/Y sequences (in practice, the sequences are written with 0/1).

To use `likelihood`, one has to provide a phylogeny for the taxa under investigation (i.e. topology and branch lengths) and a substitution model for nucleotide evolution with user-defined parameters. You cannot use the **HETEROGENEOUS** model with `likelihood`.

To use `likelihood`, type at the command-line:

```
likelihood likelihood-control-file
```

where `likelihood-control-file` is a valid control file for the `likelihood` program.

Control file for `likelihood`

An example of a valid control file for *likelihood* can be found in the appendix. The *likelihood* control file requires the specification of:

- a ***DATAFILE*** *block*: see the data file block section.
- a ***MODEL*** *block*: see the model block section.
  - *Model parameters file*: The name of the file containing parameter values for the model. The inference programs of the **PHASE** package produce such a file, or one can be generated with the `simulate` program.
- a ***TREE*** *block*: this block is compulsory and has only one parameter:
  - *Tree file*: the name of the file containing the phylogeny, a tree in the Newick format with branch length values.

- *Site-specific substitution rates*: the location of a file for storing the results from computation of Bayesian posterior estimates of site-specific substitution rate categories [Mayrose et al., 2004, Yang and Wang, 1995, see, e.g.]:

$$P(r_i|X,\hat{\theta},\hat{\nu}) = \frac{P(X|r_i,\hat{\theta},\hat{\nu}) \times P(r_i)}{\sum_{i=1}^{k} P(X|r_i,\hat{\theta},\hat{\nu}) \times P(r_i)}.$$

- *Ancestral sequences*: the location of a file for storing the results of marginal ancestral sequence reconstruction [Yang et al., 1995, see].

- *Rate vs. composition*: a yes/no parameter (no by default). Using Bayesian posterior probabilities for each rate category and the observed composition at each site for a set of selected species, `likelihood` proposes empirical frequencies $\mathbf{\Pi} = \{\pi_1, \ldots, \pi_n\}$ for each rate category:

$$P(\mathbf{\Pi}|r) = \frac{\sum_{i=1}^{nb_{sites}} P(\mathbf{\Pi}|X_i)P(r|X_i)}{\sum_{i=1}^{nb_{sites}} P(r|X_i)}.$$

  Results are displayed on the screen.

- *Selected species*: This parameter is necessary when *Rate vs. composition* is used. It specifies the set of species to be used in the procedure described above. Options are:

  - the name of one taxon in your datafile
  - "all" to perform the procedure with all the taxa
  - the name of a file which contains a set of species (one species per line, do not use ';' to end the line)

simulate

Using `simulate`

The `simulate` program is used:

1. To generate examples of "`.mod`" files for all the substitution models implemented in **PHASE**. A `.mod` file is used to provide initial or fixed values for the model parameters to some programs in the package.
2. To generate molecular sequences which evolved from a random initial sequence according to a specified model of evolution, i.e. phylogeny and substitution model. Various options are available to generate a random topology and corresponding branch lengths.

To use `simulate`, type at the command-line:

```
simulate simulate-control-file
```

where `simulate-control-file` is a valid control file for the `simulate` program.

In its first mode of operation, `simulate` creates a single `.mod` file and you can modify this file with your own initial values. In its second mode of operation, `simulate` generates the sequences and saves them in a file specified by the user. The format of this file is described in the data file format section. If the **MIXED** model is used, heterogeneous sequences are generated in sequential order. The **HETEROGENEOUS** model cannot be used at the moment. The tree used to generate the sequences (provided by the user or "randomly" created) is displayed on the screen. Eventually, the likelihood of the generated molecular sequences given the model is returned if possible (i.e. if you provided all the optional parameters necessary to build a complete data file in the **PHASE** format).

Control file for `simulate`

In the appendix, two control files are provided as examples. The control file of the `simulate` program must provide:

- a ***MODEL** block*: see the model block section.
- *Retrieve the name of the model's parameters*: a boolean field to specify the user's aim. Use **yes** for the first mode of usage mentioned above (generate a `.mod` file with default values) and **no** for the second mode (evolve sequences along a phylogeny given a substitution model).
- *Model parameters file*: if `simulate` is used to generate an example of a substitution model parameters file, the parameters are saved in this file. When `simulate` is used to generate sequences, the user must provide parameters for the substitution model and they are read from this file.

The following fields may be required when `simulate` is used to generate sequences.

- *Random seed*: the integer value provided with this field is used to initialise the random number generator. CPU time is used if the seed is not provided.
- a ***TREE** block*: see the **TREE** block for `simulate` section, below. The block contains either the location of tree file or the parameters required necessary to generate a random tree.
- *Number of symbols from class* i: you have to specify the number of symbols (e.g. number of nucleotides, number of paired sites, or number of codons) you want to generate for each class in your final sequence. Unless the **MIXED** model is used, this will be "Number of symbols from class 1".
- *Structure for the elements of class* i: `simulate` can add a stucture mask in the generated data file. For each class in your model (1 in standard cases, more with the **MIXED** model), you must specify the appropriate structure. For example, with a **MIXED** model with DNA and RNA:

```
Structure for the elements of class 1 = .
Structure for the elements of class 2 = ()
```

- *Data file type* and *Number of nucleotides from class* i: `simulate` can produce an input file following the format defined in the data file format section. To produce this file, you have to specify yourself the type written in the first line. You also have to specify the number of nucleotides generated by each model so that `simulate` can produce a "class line" and compute the total length of your alignment.For example, with a **MIXED** model with DNA and RNA:

```
Data file type = STRUCT
Number of nucleotides from class 1 = 500
Number of nucleotides from class 2 = 400   # 200*2
```

- *Output file*: the name of the file where generated sequences are saved.

TREE block for `simulate`

The **TREE** block for the `simulate` program may contain the following parameters:

- *Generate tree*: `simulate` can either generate a random tree (**yes**) or use a supplied phylogeny. If *Generate tree* is equal to **no**, `simulate` expects a phylogeny from the user (with branch lengths).
- *Tree file*: If *Generate tree* is equal to **no** then `simulate` reads the user tree from the specified file. Otherwise, the tree generated by `simulate` is stored in the given file. In both cases, the standard Newick format used by **PHASE** is required/used.

When `simulate` has to generate a random tree, the following parameters are relevant:

- *Number of species*: the number of terminal nodes in your tree.
- *Number of species in the outgroup*: to force the creation of an outgroup clade with the specified number of species.
- *Generating model*: you have to specify a generating model for the topology. You can choose among *"Yule process"*/*"Birth-death process"* [Kendall, 1948, Nee et al., 1994, Yang and Rannala, 1997], *"Beta-splitting process"* [Aldous, 1996], *"Uniform process"* (random-addition). The *"Yule process"* and the *"Birth-death process"* encompass a prior for the branch lengths.
- *Beta parameter*: required when using the beta-splitting process to generate a random topology. It can be a value between -2.0 (comb) and +inf (close to a medium split). You are allowed to write *"+inf"*. For information, beta=-1.5 corresponds to the *"Uniform process"* and beta=0 to the (reconstructed) birth-death process.
- *Branches prior*: The *"Beta-splitting process"* and the *"Uniform process"* only define the mechanism to generate a topology and you also need to specify a process to generate branch lengths. This field is compulsory when they are used. You can use *"Uniform"* for a uniform prior between a lower and upper-bound, *"Exponential"*, *"Pure-birth process"* (automatic/compulsory with Yule process on topology) and *"Birth-death process"* (automatic/compulsory with Birth-death process on topology). The two last processes will generate an ultrametric tree.
  - *Branch prior, upper bound* and *Branch prior, lower bound*: when *"Uniform"* is used, you have to specify an upper bound for the branch lengths. The lower bound is optional (0 by default).
  - *Branch prior, exponential parameter*: compulsory when *"Exponential"* is used. Branches are drawn from the distibution $p(x) = \lambda \exp -lambdax$ where λ, the specified value, is the inverse of the mean expectation.
  - *Birth rate*: this parameter is compulsory when when using a *"Pure-birth"*/*"Birth-death"* process on the branch lengths or if you chose the *"Yule process"* and the *"Birth-death process"*.
  - *Death rate*: this parameter is compulsory when when using a *"Birth-death"* process on the branch lengths or if you chose the *"Birth-death process"*.
- *Tree height*: When using the Yule process or the Birth-death process, or when branch lengths are using a Yule or Birth-death prior, the tree generated will be ultrametric and you have to specify its height.

analyzer

The `analyzer` program does not require any control file. To use `analyzer`, type at the command-line:

```
analyzer
```

or

```
analyzer datafile
```

where `datafile` is a file following the data file format described above. The `analyzer` program will ask for the fields traditionally used in the **DATAFILE** block in order to parse your sequences. Once this is done you will be prompted for the class to check if the data file contains more than one class (if you answered "yes" or "auto" to the question "Heterogeneous data models?"). `analyzer` will then require a `.lmp` file to locate the "bad" sites, e.g. the unpaired sites with too many gaps or the paired sites too many mismatche.

Three ".lmp" files are provided with the **PHASE** package, in the `input-data` directory: `dna.lmp`, `RY.lmp` and `rna.lump`. The `dna.lmp` and `RY.lmp` files are used with partitions of unpaired nucleotides. Sites with too many ambiguities, e.g. `-`, **X** and **N**/**?** (plus **R** and **Y** for `dna.lmp`), will be reported. The threshold is specified by the user. The third file (`rna.lmp`) is used with a partition of paired sites; mismatches (e.g. **AC**, **UU**) are grouped into the single state **MM** and ambiguities (e.g. **C-**, **UR**) are grouped into the state **XX**. Sites with too many **MM** and **XX** states are reported.

Pitfalls of Markov chain Monte-Carlo techniques

One can doubt that maximum-likelihood algorithms always find the true global maximum of the likelihood function. Similarly, MCMC techniques can fail to converge to the stationary distribution of the posterior probabilities. Failure to visit all highly probable regions of the parameter space because of local maxima in the likelihood curve can be a possible reason for this. However poor proposal mechanisms and/or failure to run the chain long enough are usually the main cause of sample defect [Huelsenbeck et al., 2002, see]. When using **PHASE**, you should also be wary of a possible overparameterization: RNA models and heterogenous methods are complex parameter-rich techniques which can give misleading results if not used with care. Unfortunately, it is not always easy to identify these traps but here is a set of advice and rules which might help you to check that **PHASE** is behaving properly:

- Scrutinize outputs carefully, and check that parameters are biologically plausible.
- Do long runs.
- Monitor the convergence of **several** model parameters; monitoring only the likelihood is not enough.
- Repeat the experiment using different random starting trees to check that all the chains give similar results (i.e. substitution model parameters, consensus tree, likelihood, clade supports).
- A correspondingly large amount of data is required for complex models (e.g. RNA models and heterogeneous methods) if substitution parameters are to be estimated during the inference process. Be cautious when performing phylogenetic inference with small RNA molecules and/or few species and/or sequences with low divergence. Always look for possible signs of overparameterization (e.g. high variance). Concatenating genes can help.
- If exchangeability parameters (i.e. rate ratios) reach their upper-bound value, your substitution model is probably slightly overparameterized. This is related to a badly-chosen "flat" prior on rate ratios which is not as uninformative as we previously thought [Zwickl and Holder, 2004]. Consider using a set of fixed parameters or switching to a less complex model if possible (e.g. from 16-state RNA models to 7-state models, from **RNA7A** to **RNA7D**, reduce the number of models with **HETEROGENEOUS**). You can also swith to an exponential prior on the rate ratios but be aware that it means the prior has an impact on the results.
- When using a uniform prior on branch lengths and a +I model, branches might strike their upper-bound value. Stick to the exponential prior in this case.
- When using the **MIXED** model, check carefully the values of the average substitution rate for each model and of the gamma shape parameter(s) if relevant. The rare occurrence of entrapments in local maximum we spotted were related to these parameters.

Substitution Models

Nucleotide substitution models

Substitution models are a description of the way sequences evolve in time by nucleotide replacements. The **PHASE** package provides a wide range of substitution models. These consist primarily of standard nucleotide substitution models and RNA (base-paired) substitution models; a simple codon model and empirical amino-acid models are also available, but are not well-supported.

A Markov model of substitution

Replacements within DNA sequences can be described and modelled by a Markov process with four states. Each state represents one base: **A**denine, **C**ytosine, **G**uanine or **T**hymine (see figure 1).

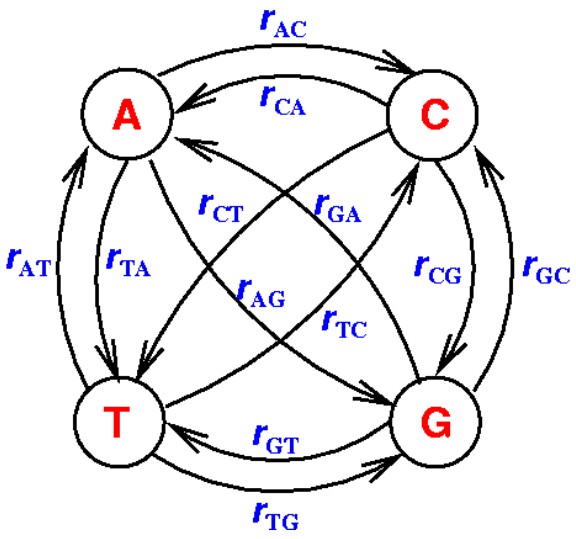

Figure 1: Markov model for nucleotide evolution in DNA sequences

Many assumptions are made in order to make phylogenetic reconstructions more computationally tractable. First, each nucleotide is supposed to evolve independently of other sites and of its past history. Further, the Markov process of substitution is assumed to be the same across all sites (*spatial homogeneity*). Finally, the process is assumed to remain constant over time (*stationary*) and *time homogeneous*, i.e. nucleotide frequencies and substitution rates can be assumed constant through time and across all sites in an alignment.

One might concede that assumptions made for the nucleotide evolutionary process are not strictly valid. Actual data shows some discrepancies, e.g. heterogeneous selection pressure, or unequal base frequencies among species. We can relax these assumptions and allow for substitution rate variation across sites with the discrete gamma model [Yang, 1994]. It is also possible to use multiple substitution processes simultaneously when heterogeneous data are analysed.

In spite of their name, DNA models can naturally be used for the treatment of the loops within RNA sequences. In RNA loops, nucleotides are not subject to any structural constraints and they are assumed to evolve independently from other sites. Therefore, the use of similar Markov models for nucleotide evolution in RNA loops is appropriate.

Standard DNA models are not well adapted for RY recoding. RY and AGY substitution models are implemented in **PHASE**. These models are designed for nucleotide sequences, but they are not 4-state models. Both models group **C** and **T** into a single state **Y** (pyrimidine) and the RY model goes further and groups **A** and **G** into a purine **R** state. The **TWOSTATE** and the **THREESTATE** models are described after the standard 4-state DNA models.

Transition matrices

The mathematical expression of a DNA Markov model uses a matrix Q of substitution rates in which each element r_{ij} represents the rate of substitution from nucleotide i to nucleotide j. The diagonal elements of the instantaneous rate matrix must satisfy the equation

$$r_{ii} = -\sum_{j \neq i} r_{ij} \tag{13}$$

so that each row of Q sums to zero. The process must be *homogeneous* and *stationary*; if π_A, π_C, π_G and π_T are the four equilibrium bases frequencies then the rates must obey the following constraint:

$$\pi_i r_{ij} = \pi_j r_{ji} \quad \forall i, j \tag{14}$$

also known as the *time-reversibility* constraint. To enforce this constraint we define α_{ij} so that

$$Q(i,j) = r_{ij} = m_r \times \pi_j \alpha_{ij} \quad \forall i, \forall j \neq i \tag{15}$$

where m_r is a constant factor described later. The *time-reversibility* condition is satisfied with a symmetric choice of α_{ij}. In practice, **PHASE** uses one of these α_{ij} parameters as a reference and sets its value to 1.0. Depending on the model, other parameters (we call them *rate ratios*) are fixed or inferred during an analysis.

With Q we can compute the transition probability matrix over time t.

$$\begin{aligned} \frac{dP(t)}{dt} &= P(t) \times Q \\ P(t) &= \exp(Qt) \\ &= \exp(\{\pi_j \alpha_{ij}\} \times m_r \times t) \end{aligned}$$

The transition probability matrix $P(t) = \{p_{ij}(t)\}$ is used to compute the probability that nucleotide i will be nucleotide j after time t (i can be equal to j). The "rate ratios" matrix in **PHASE** refers to the matrix $\{\alpha_{ij}\}$ and the "transition rates" matrix refers to Q.

Inference methods used do not permit the separation of mr, a factor proportional to the average substitution rate of the model, and t, branch lengths of the evolutionary tree which reflect an amount of change. The longer the branch, the bigger the evolutionary distance between its two incident nodes. We have to impose a scaling on the branch length. In practice, we fix the average rate of substitutions of our model to be one per "unit of time". This is done by adding a constraint for the factor m_r.

$$m_r \times \sum_{i=1}^{nb_{states}} \sum_{j \neq i} \pi_i r_{ij} = 1.0 \tag{16}$$

This last constraint does not hold when multiple substitution models are used simultaneously in the **MIXED** model. The average substitution rate of the first model is still fixed equal to 1.0 but the average substitution rate of other models is now a free parameter.

Nucleotide substitution models implemented in PHASE

One can refer to Whelan et al. [2001] for a comprehensive review of the following substitution models and their hierarchical relationships. The transition rate matrices of these models can highlight their differences. They are presented by increasing complexity, i.e. ordered according to their number of free parameters (equilibrium frequencies and/or rates). In nucleotide substitution models, the A↔G transitition is used as a reference by **PHASE**: $\alpha_{AG} = \alpha_{GA} = 1$.

JC69 model [Jukes and Cantor, 1969]

The Jukes-Cantor model assumes equal base frequencies and equal mutation rates, therefore it does not have any free parameters. $\pi_i = \frac{1}{4} \quad \forall i, \quad \alpha_{ij} = 1.0 \quad \forall i, \forall j \neq i$

$$Q = m_r \times \begin{pmatrix} & A & C & G & T \\ A & * & 0.25 & 0.25 & 0.25 \\ C & 0.25 & * & 0.25 & 0.25 \\ G & 0.25 & 0.25 & * & 0.25 \\ T & 0.25 & 0.25 & 0.25 & * \end{pmatrix}$$

Table 1: **JC69** transition matrix

K80 model [Kimura, 1980]

The Kimura model assumes equal base frequencies and accounts for the difference between transitions and transversions with one parameter. $\pi_i = \frac{1}{4} \quad \forall i, \quad \alpha_{transition} = 1.0, \quad \alpha_{transversion} = \alpha_1$

$$Q = m_r \times \begin{pmatrix} & A & C & G & T \\ A & * & 0.25\alpha_1 & 0.25 & 0.25\alpha_1 \\ C & 0.25\alpha_1 & * & 0.25\alpha_1 & 0.25 \\ G & 0.25 & 0.25\alpha_1 & * & 0.25\alpha_1 \\ T & 0.25\alpha_1 & 0.25 & 0.25\alpha_1 & * \end{pmatrix}$$

Table 2: **K80** transition matrix

F81 model [Felsenstein, 1981]

The F81 model permits unequal base frequencies and assumes the same rate for transitions and transversions.

$$Q = m_r \times \begin{pmatrix} & A & C & G & T \\ A & * & \pi_C & \pi_G & \pi_T \\ C & \pi_A & * & \pi_G & \pi_T \\ G & \pi_A & \pi_C & * & \pi_T \\ T & \pi_A & \pi_C & \pi_G & * \end{pmatrix}$$

Table 3: **F81** transition matrix

HKY85 model [Hasegawa et al., 1985]

The HKY85 model does not assume equal base frequencies and accounts for the difference between transitions and transversions with one parameter. $\alpha_{transition} = 1.0, \quad \alpha_{transversion} = \alpha_1$

$$Q = m_r \times \begin{pmatrix} & A & C & G & T \\ A & * & \pi_C\alpha_1 & \pi_G & \pi_T\alpha_1 \\ C & \pi_A\alpha_1 & * & \pi_G\alpha_1 & \pi_T \\ G & \pi_A & \pi_C\alpha_1 & * & \pi_T\alpha_1 \\ T & \pi_A\alpha_1 & \pi_C & \pi_G\alpha_1 & * \end{pmatrix}$$

Table 4: **HKY85** transition tatrix

T92 model [Tamura, 1992]

Please note that we refer here to the time-homogeneous version of the model Galtier and Gouy [1998] used in their paper. This substitution model is a simplified version of the HKY model with only 2 frequency parameters: G+C and A+T.

$$Q = m_r \times \begin{pmatrix} & A & C & G & T \\ A & * & \frac{\pi_{C+G}}{2}\alpha_1 & \frac{\pi_{C+G}}{2} & \frac{\pi_{A+T}}{2}\alpha_1 \\ C & \frac{\pi_{A+T}}{2}\alpha_1 & * & \frac{\pi_{C+G}}{2}\alpha_1 & \frac{\pi_{A+T}}{2} \\ G & \frac{\pi_{A+T}}{2} & \frac{\pi_{C+G}}{2}\alpha_1 & * & \frac{\pi_{A+T}}{2}\alpha_1 \\ T & \frac{\pi_{A+T}}{2}\alpha_1 & \frac{\pi_{C+G}}{2} & \frac{\pi_{C+G}}{2}\alpha_1 & * \end{pmatrix}$$

Table 5: **T92** transition matrix

TN93 model [Tamura and Nei, 1993]

The TN93 model has four frequency parameters. It accounts for the difference between transitions and transversions, and differentiates the two types of transition (purine↔purine & pyrimidine↔pyrimidine). $\alpha_{AG} = \alpha_{GA} = 1.0, \quad \alpha_{transversion} = \alpha_1, \quad \alpha_{CT} = \alpha_{TC} = \alpha_2$

$$Q = m_r \times \begin{pmatrix} & A & C & G & T \\ A & * & \pi_C\alpha_1 & \pi_G & \pi_T\alpha_1 \\ C & \pi_A\alpha_1 & * & \pi_G\alpha_1 & \pi_T\alpha_2 \\ G & \pi_A & \pi_C\alpha_1 & * & \pi_T\alpha_1 \\ T & \pi_A\alpha_1 & \pi_C\alpha_2 & \pi_G\alpha_1 & * \end{pmatrix}$$

Table 6: **TN93** transition matrix

REV model [Tavare, 1986]

The REV model is the most general model for nucleotide substitution that is subject to the time-reversibility constraint. It has four frequencies and five rate parameters.

$$Q = m_r \times \begin{pmatrix} & A & C & G & T \\ A & * & \pi_C\alpha_1 & \pi_G & \pi_T\alpha_2 \\ C & \pi_A\alpha_1 & * & \pi_G\alpha_3 & \pi_T\alpha_4 \\ G & \pi_A & \pi_C\alpha_3 & * & \pi_T\alpha_5 \\ T & \pi_A\alpha_2 & \pi_C\alpha_4 & \pi_G\alpha_5 & * \end{pmatrix}$$

Table 7: **REV** transition matrix

The TWOSTATE (RY) and the THREESTATE (AGY) models

The mathematics of these two models are the same as for standard 4-state models. They only differ by their number of states.

TWOSTATE model (general time-reversible RY)

The **TWOSTATE** model has two states (**0** and **1**) and two associated frequency parameters. It can handle standard nucleotide sequences (**A**, **G** and **R** are mapped to the state **0**, **C**, **T**/**U** and **Y** to the state **1**). It is not necessary to recode the data since **PHASE** will do that internally for you. The **TWOSTATE** model will also work fine with **0**/**1** sequences. Though this might not

prove very useful, it can also be used the with doublets of nucleotides: **A:U**, **G:U** and **G:C** are mapped to state **0**, whereas **U:A**, **U:G** and **U:C** are mapped to the state **1**)

$$Q = m_r \times \begin{pmatrix} & 0/R & 1/Y \\ 0/R & * & \pi_1 \\ 1/Y & \pi_0 & * \end{pmatrix}$$

Table 8: **TWOSTATE** transition matrix

THREESTATE model (general time-reversible AGY)

The **THREESTATE** model has three states (**A**, **G** and **Y**) and three associated frequency parameters. It also has two unrestricted exchangeability parameters. Similarly to the **TWOSTATE** model, it can handle standard nucleotide sequences without need for recoding (**C** and **T** are mapped to the state **Y** internally).

$$Q = m_r \times \begin{pmatrix} & A & Y & G \\ A & * & \pi_Y\alpha_1 & \pi_G \\ Y & \pi_A\alpha_1 & * & \pi_G\alpha_2 \\ G & \pi_A & \pi_Y\alpha_2 & * \end{pmatrix}$$

Table 9: **THREESTATE** transition matrix

Paired-site substitution models

RNA substitution models are an attempt to add biological realism to the evolutionary modelling. The assumption that each nucleotide site evolves independently must be modified for RNA molecules. Paired-site substitution models can account for the *secondary structure* of these molecules.

RNA secondary structure

In the double helical structure of the DNA molecule, two *complementary* nucleotide strands are held together with hydrogen bonds between the *Waston-Crick* pairs **A-T** and **C-G**. RNA molecules can fold themselves in their *tertiary structure* because of the same hydrogen bonding mechanism. Helices, also known as *stems*, are formed intra-molecularly.

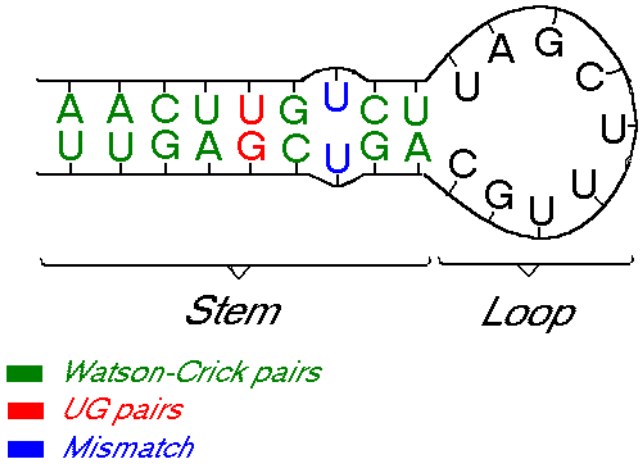

Figure 2: A RNA molecule secondary structure

There are 16 possible base-pairings, however, only six of these (**AU**, **GU**, **GC**, **UA**, **UG**, **CG**) are stable enough to form actual base-pairs. The rest are called mismatches and occur at very low frequencies in helices. RNA molecules, such as ribosomal RNAs and transfer RNAs, have an important role. Their structure cannot easily be disrupted without impact on their function, and selection acts to maintain the secondary structure. Yet, the *primary structure* of the stems (i.e. their nucleotide sequence) can still vary and in fact we observe that RNA helical regions are quite variable in sequence. The nature of the bases is not important and substitutions are possible as long as they preserve the secondary structure. One could model the evolution of stems using the DNA models described above but there may be a substantial bias in results because paired substitutions would seem far less probable than they are in reality [Jow et al., 2002, see]. Statistics become invalid and it can have an effect on inferred phylogenies.

The secondary structure is left unchanged when complementary substitutions occur in the DNA gene coding for the RNA molecule. The process can be a single step process (double substitution) or a two step process (two single substitutions). These two processes are described in the following section.

Theory of compensatory substitutions

From the individual sequence viewpoint, complementary mutations are a two-step process typically involving a **U-G** or a **G-U** pair as a transition state. These pairs are thermodynamically less stable than Waston-Crick pairs but they are still more likely to arise than any other mismatches. Nonetheless, in phylogenetic studies we are not considering individual copies of a gene but we are rather modelling consensus sequences for a large number of individuals. From the population genetics viewpoint, evolution in stems can either occur by two single substitutions or by simultaneous compensatory substitutions, [Higgs, 1998, Savill et al., 2001, see, e.g.]. The first mechanism is by fixation of the slightly deleterious UG or GU pair in the population before the second mutation occurs. The second mechanism happens when natural selection against intermediate mutants is too strong. In such a case, deleterious pairs are kept low in frequency until a second mutation takes place in one of the sporadic mutant sequences by chance. Afterwards, the new neutral variant may replace the original one due to drift in gene frequencies (see figure).

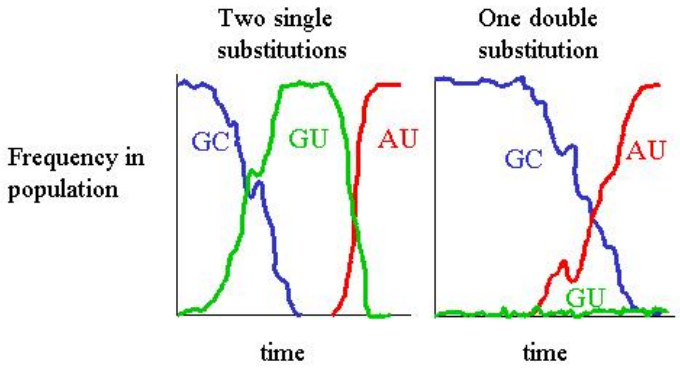

Figure 3: Substitution mechanisms for paired-sites

Therefore, even if simultaneous mutations are very unlikely to occur in a single organism, it is reasonable, although not compulsory, to allow double substitutions in models from the population point of view. The experimental results from **PHASE** bear this out, especially with nuclear data [Tillier and Collins, 1998]. Since natural selection against intermediate mutants with any other mismatch pairs than **U-G** or **G-U** is usually much stronger, one can notice two groups of states within which rapid interchange occurs, while interchange between the two groups, although possible, is really slow (see figure).

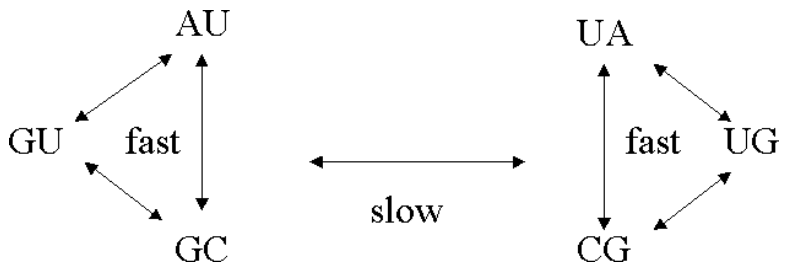

Figure 4: Mutation rate between paired-sites

Base-paired substitution models implemented in PHASE

Like DNA models, RNA substitution models are Markov models but they consider pairs of nucleotides as their elementary states rather than single sites. The **PHASE** software has 16-state models to account for the 16 possible pairs that can be formed with 4 bases. These 16-state models sometimes have a lot of parameters (and in any cases are computationaly more expensive since they have more states). Consequently, you might prefer them the 6-state and 7-state models where mismatch pairs are respectively discarded or grouped into a single state **MM**. The time-reversibility contraint and the average mutation rate are set as they were for DNA models. One can refer to Savill et al. [2001] for a review of the following substitution models and their hierarchical relationships. With base-paired models, **PHASE** usually uses the mutability rate of the double transition **AU**↔**GC** as a reference for the rate ratios. When double substitutions are not allowed the **AU**↔**GU** exchangeability is used as a replacment. These RNA models are known by different names. For clarity, we used Savill et al. [2001] nomenclature in the software and we attempt to acknowledge the the initial authors in this manual.

6-state substitution models

RNA6A model

Six state models completely ignore mismatches and consider substitutions between the six stable base-pairs only. Mismatch pairs are assigned to one or more of the 6 states in some deterministic fashion (the treatment of a mismatch is quite similar to the treatment of a gap with the DNA models and are treated as ambiguities, e.g. **GG** is treated as **GC**,**GU**,**UG** or **CG**). The **RNA6A** model is the most general six state model with 15 rate parameters and 6 frequencies (and 2 constraints) as shown in table 10.

$$Q = m_r \times \begin{pmatrix} & AU & GU & GC & UA & UG & CG \\ AU & * & \pi_{GU}\alpha_1 & \pi_{GC} & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_3 & \pi_{CG}\alpha_4 \\ GU & \pi_{AU}\alpha_1 & * & \pi_{GC}\alpha_5 & \pi_{UA}\alpha_6 & \pi_{UG}\alpha_7 & \pi_{CG}\alpha_8 \\ GC & \pi_{AU} & \pi_{GU}\alpha_5 & * & \pi_{UA}\alpha_9 & \pi_{UG}\alpha_{10} & \pi_{CG}\alpha_{11} \\ UA & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_6 & \pi_{GC}\alpha_9 & * & \pi_{UG}\alpha_{12} & \pi_{CG}\alpha_{13} \\ UG & \pi_{AU}\alpha_3 & \pi_{GU}\alpha_7 & \pi_{GC}\alpha_{10} & \pi_{UA}\alpha_{12} & * & \pi_{CG}\alpha_{14} \\ CG & \pi_{AU}\alpha_4 & \pi_{GU}\alpha_8 & \pi_{GC}\alpha_{11} & \pi_{UA}\alpha_{13} & \pi_{UG}\alpha_{14} & * \end{pmatrix}$$

Table 10: **RNA6A** transition matrix

RNA6B model

The **RNA6B** model is formed by restriction of the **RNA6A** model. The **RNA6B** model has only 3 rate parameters and 6 frequencies, it uses a rate of single transitions α_1 and a rate of double transversions α_2. The reference for the rate ratios are the rates of double transition.

$$Q = mr * \begin{pmatrix} & AU & GU & GC & UA & UG & CG \\ AU & * & \pi_{GU}\alpha_1 & \pi_{GC} & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 \\ GU & \pi_{AU}\alpha_1 & * & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 \\ GC & \pi_{AU} & \pi_{GU}\alpha_1 & * & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 \\ UA & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & * & \pi_{UG}\alpha_1 & \pi_{CG} \\ UG & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & \pi_{UA}\alpha_1 & * & \pi_{CG}\alpha_1 \\ CG & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & \pi_{UA} & \pi_{UG}\alpha_1 & * \end{pmatrix}$$

Table 11: **RNA6B** transition matrix

RNA6C model [Tillier, 1994]

The **RNA6C** model is formed by restriction of the **RNA6B** model by imposing base-pair reversal symmetry. **RNA6C** has 3 frequency parameters instead of 6: $\pi_{AU} = \pi_{UA}$, $\pi_{GU} = \pi_{UG}$ and $\pi_{GC} = \pi_{CG}$.

$$Q = mr * \begin{pmatrix} & AU & GU & GC & UA & UG & CG \\ AU & * & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2} & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 \\ GU & \frac{\pi_{AU+UA}}{2}\alpha_1 & * & \frac{\pi_{GC+CG}}{2}\alpha_1 & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 \\ GC & \frac{\pi_{AU+UA}}{2} & \frac{\pi_{GU+UG}}{2}\alpha_1 & * & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 \\ UA & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & * & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2} \\ UG & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & \frac{\pi_{AU+UA}}{2}\alpha_1 & * & \frac{\pi_{GC+CG}}{2}\alpha_1 \\ CG & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & \frac{\pi_{AU+UA}}{2} & \frac{\pi_{GU+UG}}{2}\alpha_1 & * \end{pmatrix}$$

Table 12: **RNA6C** transition matrix

RNA6D model [Tillier, 1994]

The **RNA6D** model is formed by restriction of the **RNA6C** model. Double transitions are not allowed. Note that we cannot allow all double substitutions to be equal to 0 otherwise pairs would be subdivided into two non interchangeable groups of three states. **AU↔GU** becomes the reference.

RNA6E model

The **RNA6E** is nested in the **RNA6B** model and is more complex than the **RNA6D**. Double transitions are not allowed but symmetry of equilibrium frequencies is not assumed. The exchangeability **AU↔GU** is the reference.

7-state substitution models

RNA7A model [Higgs, 2000]

The **RNA7A** model is the most general of the seven state models. It has 21 rate parameters (including the reference rate **AU↔GC**) and 7 frequencies. All mismatches are treated in a single

$$Q = mr * \begin{pmatrix} & AU & GU & GC & UA & UG & CG \\ AU & * & \frac{\pi_{GU+UG}}{2} & 0 & \frac{\pi_{AU+UA}}{2}\alpha_1 & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2}\alpha_1 \\ GU & \frac{\pi_{AU+UA}}{2} & * & \frac{\pi_{GC+CG}}{2} & \frac{\pi_{AU+UA}}{2}\alpha_1 & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2}\alpha_1 \\ GC & 0 & \frac{\pi_{GU+UG}}{2} & * & \frac{\pi_{AU+UA}}{2}\alpha_1 & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2}\alpha_1 \\ UA & \frac{\pi_{AU+UA}}{2}\alpha_1 & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2}\alpha_1 & * & \frac{\pi_{GU+UG}}{2} & 0 \\ UG & \frac{\pi_{AU+UA}}{2}\alpha_1 & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2}\alpha_1 & \frac{\pi_{AU+UA}}{2} & * & \frac{\pi_{GC+CG}}{2} \\ CG & \frac{\pi_{AU+UA}}{2}\alpha_1 & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2}\alpha_1 & 0 & \frac{\pi_{GU+UG}}{2} & * \end{pmatrix}$$

Table 13: **RNA6D** transition matrix

$$Q = mr * \begin{pmatrix} & AU & GU & GC & UA & UG & CG \\ AU & * & \pi_{GU} & 0 & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_1 & \pi_{CG}\alpha_1 \\ GU & \pi_{AU} & * & \pi_{GC} & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_1 & \pi_{CG}\alpha_1 \\ GC & 0 & \pi_{GU} & * & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_1 & \pi_{CG}\alpha_1 \\ UA & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_1 & \pi_{GC}\alpha_1 & * & \pi_{UG} & 0 \\ UG & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_1 & \pi_{GC}\alpha_1 & \pi_{UA} & * & \pi_{CG} \\ CG & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_1 & \pi_{GC}\alpha_1 & 0 & \pi_{UG} & * \end{pmatrix}$$

Table 14: **RNA6E** transition matrix

state MM. The **RNA7A** model is described by the following rate matrix.

$$Q = m_r \times \begin{pmatrix} & AU & GU & GC & UA & UG & CG & MM \\ AU & * & \pi_{GU}\alpha_1 & \pi_{GC} & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_3 & \pi_{CG}\alpha_4 & \pi_{MM}\alpha_5 \\ GU & \pi_{AU}\alpha_1 & * & \pi_{GC}\alpha_6 & \pi_{UA}\alpha_7 & \pi_{UG}\alpha_8 & \pi_{CG}\alpha_9 & \pi_{MM}\alpha_{10} \\ GC & \pi_{AU} & \pi_{GU}\alpha_6 & * & \pi_{UA}\alpha_{11} & \pi_{UG}\alpha_{12} & \pi_{CG}\alpha_{13} & \pi_{MM}\alpha_{14} \\ UA & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_7 & \pi_{GC}\alpha_{11} & * & \pi_{UG}\alpha_{15} & \pi_{CG}\alpha_{16} & \pi_{MM}\alpha_{17} \\ UG & \pi_{AU}\alpha_3 & \pi_{GU}\alpha_8 & \pi_{GC}\alpha_{12} & \pi_{UA}\alpha_{15} & * & \pi_{CG}\alpha_{18} & \pi_{MM}\alpha_{19} \\ CG & \pi_{AU}\alpha_4 & \pi_{GU}\alpha_9 & \pi_{GC}\alpha_{13} & \pi_{UA}\alpha_{16} & \pi_{UG}\alpha_{18} & * & \pi_{MM}\alpha_{20} \\ MM & \pi_{AU}\alpha_5 & \pi_{GU}\alpha_{10} & \pi_{GC}\alpha_{14} & \pi_{UA}\alpha_{17} & \pi_{UG}\alpha_{19} & \pi_{CG}\alpha_{20} & * \end{pmatrix}$$

Table 15: **RNA7A** transition matrix

RNA7B model

The **RNA7B** model is naturally derived from the **RNA7A** model by imposing base-pair reversal symmetry: $\pi_{XY} = \pi_{YX}$. This removes three frequency parameters.

$$Q = m_r \times \begin{pmatrix} & AU & GU & GC & UA & UG & CG & MM \\ AU & * & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2} & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_3 & \frac{\pi_{GC+CG}}{2}\alpha_4 & \pi_{MM}\alpha_5 \\ GU & \frac{\pi_{AU+UA}}{2}\alpha_1 & * & \frac{\pi_{GC+CG}}{2}\alpha_6 & \frac{\pi_{AU+UA}}{2}\alpha_7 & \frac{\pi_{GU+UG}}{2}\alpha_8 & \frac{\pi_{GC+CG}}{2}\alpha_9 & \pi_{MM}\alpha_{10} \\ GC & \frac{\pi_{AU+UA}}{2} & \frac{\pi_{GU+UG}}{2}\alpha_6 & * & \frac{\pi_{AU+UA}}{2}\alpha_{11} & \frac{\pi_{GU+UG}}{2}\alpha_{12} & \frac{\pi_{GC+CG}}{2}\alpha_{13} & \pi_{MM}\alpha_{14} \\ UA & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_7 & \frac{\pi_{GC+CG}}{2}\alpha_{11} & * & \frac{\pi_{GU+UG}}{2}\alpha_{15} & \frac{\pi_{GC+CG}}{2}\alpha_{16} & \pi_{MM}\alpha_{17} \\ UG & \frac{\pi_{AU+UA}}{2}\alpha_3 & \frac{\pi_{GU+UG}}{2}\alpha_8 & \frac{\pi_{GC+CG}}{2}\alpha_{12} & \frac{\pi_{AU+UA}}{2}\alpha_{15} & * & \frac{\pi_{GC+CG}}{2}\alpha_{18} & \pi_{MM}\alpha_{19} \\ CG & \frac{\pi_{AU+UA}}{2}\alpha_4 & \frac{\pi_{GU+UG}}{2}\alpha_9 & \frac{\pi_{GC+CG}}{2}\alpha_{13} & \frac{\pi_{AU+UA}}{2}\alpha_{16} & \frac{\pi_{GU+UG}}{2}\alpha_{18} & * & \pi_{MM}\alpha_{20} \\ MM & \frac{\pi_{AU+UA}}{2}\alpha_5 & \frac{\pi_{GU+UG}}{2}\alpha_{10} & \frac{\pi_{GC+CG}}{2}\alpha_{14} & \frac{\pi_{AU+UA}}{2}\alpha_{17} & \frac{\pi_{GU+UG}}{2}\alpha_{19} & \frac{\pi_{GC+CG}}{2}\alpha_{20} & * \end{pmatrix}$$

Table 16: **RNA7B** transition matrix

RNA7C model

The **RNA7C** model is obtained from the **RNA7A** model by setting all double-substitution rates to be 0. Since changes to and from the mismatch state are considered as single substitution, there is no need to keep double-transitions between the two main groups of pairs as was done for **RNA6D**. **AU↔GU** becomes the reference.

$$Q = m_r \times \begin{pmatrix} & AU & GU & GC & UA & UG & CG & MM \\ AU & * & \pi_{GU} & 0 & 0 & 0 & 0 & \pi_{MM}\alpha_4 \\ GU & \pi_{AU} & * & \pi_{GC}\alpha_1 & 0 & 0 & 0 & \pi_{MM}\alpha_5 \\ GC & 0 & \pi_{GU}\alpha_1 & * & 0 & 0 & 0 & \pi_{MM}\alpha_6 \\ UA & 0 & 0 & 0 & * & \pi_{UG}\alpha_2 & 0 & \pi_{MM}\alpha_7 \\ UG & 0 & 0 & 0 & \pi_{UA}\alpha_2 & * & \pi_{CG}\alpha_3 & \pi_{MM}\alpha_8 \\ CG & 0 & 0 & 0 & 0 & \pi_{UG}\alpha_3 & * & \pi_{MM}\alpha_9 \\ MM & \pi_{AU}\alpha_4 & \pi_{GU}\alpha_5 & \pi_{GC}\alpha_6 & \pi_{UA}\alpha_7 & \pi_{UG}\alpha_8 & \pi_{CG}\alpha_9 & * \end{pmatrix}$$

Table 17: **RNA7C** transition matrix

RNA7D model [Tillier and Collins, 1998]

The **RNA7D** model is a biologically plausible restriction of the **RNA7A** model. The restrictions in the **7D** model are analogous to the restrictions made in the **6B**. There is one more frequency parameter for the mismatch state and one more rate ratio parameter for the substitution rates involving this state. The reference for the rate ratios are the rate of double transitions. This model is described by the following rate matrix (table 18).

$$Q = m_r \times \begin{pmatrix} & AU & GU & GC & UA & UG & CG & MM \\ AU & * & \pi_{GU}\alpha_1 & \pi_{GC} & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 & \pi_{MM}\alpha_3 \\ GU & \pi_{AU}\alpha_1 & * & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 & \pi_{MM}\alpha_3 \\ GC & \pi_{AU} & \pi_{GU}\alpha_1 & * & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 & \pi_{MM}\alpha_3 \\ UA & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & * & \pi_{UG}\alpha_1 & \pi_{CG} & \pi_{MM}\alpha_3 \\ UG & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & \pi_{UA}\alpha_1 & * & \pi_{CG}\alpha_1 & \pi_{MM}\alpha_3 \\ CG & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & \pi_{UA} & \pi_{UG}\alpha_1 & * & \pi_{MM}\alpha_3 \\ MM & \pi_{AU}\alpha_3 & \pi_{GU}\alpha_3 & \pi_{GC}\alpha_3 & \pi_{UA}\alpha_3 & \pi_{UG}\alpha_3 & \pi_{CG}\alpha_3 & * \end{pmatrix}$$

Table 18: **RNA7D** transition matrix

RNA7E model

The **RNA7E** model is a restriction of the **RNA7C** and **RNA7D**.

$$Q = m_r \times \begin{pmatrix} & AU & GU & GC & UA & UG & CG & MM \\ AU & * & \pi_{GU} & 0 & 0 & 0 & 0 & \pi_{MM}\alpha_1 \\ GU & \pi_{AU} & * & \pi_{GC} & 0 & 0 & 0 & \pi_{MM}\alpha_1 \\ GC & 0 & \pi_{GU} & * & 0 & 0 & 0 & \pi_{MM}\alpha_1 \\ UA & 0 & 0 & 0 & * & \pi_{UG} & 0 & \pi_{MM}\alpha_1 \\ UG & 0 & 0 & 0 & \pi_{UA} & * & \pi_{CG} & \pi_{MM}\alpha_1 \\ CG & 0 & 0 & 0 & 0 & \pi_{UG} & * & \pi_{MM}\alpha_1 \\ MM & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_1 & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_1 & \pi_{CG}\alpha_1 & * \end{pmatrix}$$

Table 19: **RNA7E** transition matrix

RNA7F model

The **RNA7F** model is a restriction of the **RNA7B** and **RNA7D**. Symmetry is imposed on frequencies and exchangeabilities.

$$Q = m_r \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & MM \\
AU & * & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2} & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & \pi_{MM}\alpha_3 \\
GU & \frac{\pi_{AU+UA}}{2}\alpha_1 & * & \frac{\pi_{GC+CG}}{2}\alpha_1 & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & \pi_{MM}\alpha_3 \\
GC & \frac{\pi_{AU+UA}}{2} & \frac{\pi_{GU+UG}}{2}\alpha_1 & * & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & \pi_{MM}\alpha_3 \\
UA & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & * & \frac{\pi_{GU+UG}}{2}\alpha_1 & \frac{\pi_{GC+CG}}{2} & \pi_{MM}\alpha_3 \\
UG & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & \frac{\pi_{AU+UA}}{2}\alpha_1 & * & \frac{\pi_{GC+CG}}{2}\alpha_1 & \pi_{MM}\alpha_3 \\
CG & \frac{\pi_{AU+UA}}{2}\alpha_2 & \frac{\pi_{GU+UG}}{2}\alpha_2 & \frac{\pi_{GC+CG}}{2}\alpha_2 & \frac{\pi_{AU+UA}}{2} & \frac{\pi_{GU+UG}}{2}\alpha_1 & * & \pi_{MM}\alpha_3 \\
MM & \frac{\pi_{AU+UA}}{2}\alpha_3 & \frac{\pi_{GU+UG}}{2}\alpha_3 & \frac{\pi_{GC+CG}}{2}\alpha_3 & \frac{\pi_{AU+UA}}{2}\alpha_3 & \frac{\pi_{GU+UG}}{2}\alpha_3 & \frac{\pi_{GC+CG}}{2}\alpha_3 & *
\end{pmatrix}$$

Table 20: **RNA7F** transition matrix

RNA7G model

The **RNA7G** model combines the restrictions of the **RNA7E** and **RNA7F** models. Symmetry is imposed on frequencies and exchangeabilities.

$$Q = m_r \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & MM \\
AU & * & \frac{\pi_{GU+UG}}{2} & 0 & 0 & 0 & 0 & \pi_{MM}\alpha \\
GU & \frac{\pi_{AU+UA}}{2} & * & \frac{\pi_{GC+CG}}{2} & 0 & 0 & 0 & \pi_{MM}\alpha \\
GC & 0 & \frac{\pi_{GU+UG}}{2} & * & 0 & 0 & 0 & \pi_{MM}\alpha \\
UA & 0 & 0 & 0 & * & \frac{\pi_{GU+UG}}{2} & 0 & \pi_{MM}\alpha \\
UG & 0 & 0 & 0 & \frac{\pi_{AU+UA}}{2} & * & \frac{\pi_{GC+CG}}{2} & \pi_{MM}\alpha \\
CG & 0 & 0 & 0 & 0 & \frac{\pi_{GU+UG}}{2} & * & \pi_{MM}\alpha \\
MM & \frac{\pi_{AU+UA}}{2}\alpha & \frac{\pi_{GU+UG}}{2}\alpha & \frac{\pi_{GC+CG}}{2}\alpha & \frac{\pi_{AU+UA}}{2}\alpha & \frac{\pi_{GU+UG}}{2}\alpha & \frac{\pi_{GC+CG}}{2}\alpha & *
\end{pmatrix}$$

Table 21: **RNA7G** transition matrix

16-state substitution models

PHASE contains a general 16-state model (**RNA16**), however this model has $119 + 15$ free parameters and is not well suited for phylogenetic inference, especially maximum-likelihood inference. This model was primarily implemented because it was convenient in the C++ class hierarchy but you might be able to use it with an empirical matrix.

RNA16A model

RNA16A is a simplified 16-state model, it reduces some of the complexity of the **RNA16** model by cutting down on the number of rate parameters from 120 to 5. It uses a rate of single transitions α_1, a rate of double transversions α_2, a mismatch↔non-mismatch transition rate α_3 for transitions requiring only one substitution and a mismatch↔mismatch transition rate α_4 for transitions requiring one substitution too. The reference rate is the rate of double transitions. Some base-pair substitutions are not allowed (null substitution rate). The transition matrix for the **RNA16A** model is given in table 22.

$$Q = m_r \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\
AU & * & \pi_{GU}\alpha_1 & \pi_{GC} & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 & \pi_{AA}\alpha_3 & \pi_{AG}\alpha_3 & \pi_{AC}\alpha_3 & 0 & 0 & 0 & 0 & \pi_{CU}\alpha_3 & 0 & \pi_{UU}\alpha_3 \\
GU & \pi_{AU}\alpha_1 & * & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 & 0 & 0 & 0 & \pi_{GA}\alpha_3 & \pi_{GG}\alpha_3 & 0 & 0 & \pi_{CU}\alpha_3 & 0 & \pi_{UU}\alpha_3 \\
GC & \pi_{AU} & \pi_{GU}\alpha_1 & * & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_2 & 0 & 0 & \pi_{AC}\alpha_3 & \pi_{GA}\alpha_3 & \pi_{GG}\alpha_3 & 0 & \pi_{CC}\alpha_3 & 0 & \pi_{UC}\alpha_3 & 0 \\
UA & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & * & \pi_{UG}\alpha_1 & \pi_{CG} & \pi_{AA}\alpha_3 & 0 & 0 & \pi_{GA}\alpha_3 & 0 & \pi_{CA}\alpha_3 & 0 & 0 & \pi_{UC}\alpha_3 & \pi_{UU}\alpha_3 \\
UG & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & \pi_{UA}\alpha_1 & * & \pi_{CG}\alpha_1 & 0 & \pi_{AG}\alpha_3 & 0 & 0 & \pi_{GG}\alpha_3 & 0 & 0 & 0 & \pi_{UC}\alpha_3 & \pi_{UU}\alpha_3 \\
CG & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_2 & \pi_{UA} & \pi_{UG}\alpha_1 & * & 0 & \pi_{AG}\alpha_3 & 0 & 0 & \pi_{GG}\alpha_3 & \pi_{CA}\alpha_3 & \pi_{CC}\alpha_3 & \pi_{CU}\alpha_3 & 0 & 0 \\
AA & \pi_{AU}\alpha_3 & 0 & 0 & \pi_{UA}\alpha_3 & 0 & 0 & * & \pi_{AG}\alpha_4 & \pi_{AC}\alpha_4 & \pi_{GA}\alpha_4 & 0 & \pi_{CA}\alpha_4 & 0 & 0 & 0 & 0 \\
AG & \pi_{AU}\alpha_3 & 0 & 0 & 0 & \pi_{UG}\alpha_3 & \pi_{CG}\alpha_3 & \pi_{AA}\alpha_4 & * & \pi_{AC}\alpha_4 & 0 & \pi_{GG}\alpha_4 & 0 & 0 & 0 & 0 & 0 \\
AC & \pi_{AU}\alpha_3 & 0 & \pi_{GC}\alpha_3 & 0 & 0 & 0 & \pi_{AA}\alpha_4 & \pi_{AG}\alpha_4 & * & 0 & 0 & 0 & \pi_{CC}\alpha_4 & 0 & \pi_{UC}\alpha_4 & 0 \\
GA & 0 & \pi_{GU}\alpha_3 & \pi_{GC}\alpha_3 & \pi_{UA}\alpha_3 & 0 & 0 & \pi_{AA}\alpha_4 & 0 & 0 & * & \pi_{GG}\alpha_4 & \pi_{CA}\alpha_4 & 0 & 0 & 0 & 0 \\
GG & 0 & \pi_{GU}\alpha_3 & \pi_{GC}\alpha_3 & 0 & \pi_{UG}\alpha_3 & \pi_{CG}\alpha_3 & 0 & \pi_{AG}\alpha_4 & 0 & \pi_{GA}\alpha_4 & * & 0 & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \pi_{UA}\alpha_3 & 0 & \pi_{CG}\alpha_3 & \pi_{AA}\alpha_4 & 0 & 0 & \pi_{GA}\alpha_4 & 0 & * & \pi_{CC}\alpha_4 & \pi_{CU}\alpha_4 & 0 & 0 \\
CC & 0 & 0 & \pi_{GC}\alpha_3 & 0 & 0 & \pi_{CG}\alpha_3 & 0 & 0 & \pi_{AC}\alpha_4 & 0 & 0 & \pi_{CA}\alpha_4 & * & \pi_{CU}\alpha_4 & \pi_{UC}\alpha_4 & 0 \\
CU & \pi_{AU}\alpha_3 & \pi_{GU}\alpha_3 & 0 & 0 & 0 & \pi_{CG}\alpha_3 & 0 & 0 & 0 & 0 & 0 & \pi_{CA}\alpha_4 & \pi_{CC}\alpha_4 & * & 0 & \pi_{UU}\alpha_4 \\
UC & 0 & 0 & \pi_{GC}\alpha_3 & \pi_{UA}\alpha_3 & \pi_{UG}\alpha_3 & 0 & 0 & 0 & \pi_{AC}\alpha_4 & 0 & 0 & 0 & \pi_{CC}\alpha_4 & 0 & * & \pi_{UU}\alpha_4 \\
UU & \pi_{AU}\alpha_3 & \pi_{GU}\alpha_3 & 0 & \pi_{UA}\alpha_3 & \pi_{UG}\alpha_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{CU}\alpha_4 & \pi_{UC}\alpha_4 & *
\end{pmatrix}$$

Table 22: **RNA16A** transition matrix

RNA16B model [Schöniger and von Haeseler, 1994]

RNA16B is a simplification from the **RNA16A** model, it reduces drastically the complexity by cutting down on the number of exchangeability parameters to 1. Only simple transitions/transversions are allowed.

RNA16C model

RNA16C is a slight simplification from the **RNA16A** model, only one frequency parameter is used for the mismatches. Conceptually, it represents an extension of the **RNA7D** model into 16-state space.

RNA16D model [Savill et al., 2001]

By conception, the **RNA16D** model (and its simplifications) are different to the standard models that have been reviewed so far. They are an attempt to transform the standard 4-state models by taking into account pairing constraints. The **RNA16D** was proposed by Savill et al. [2001] as a generalization of **RNA16E** and **RNA16F** [Muse, 1995] to acknowledge the fact that **G:U**/**U:G** pairs are of intermediate fitness compared to standard Watson-Crick pairs and other mismatches. **RNA16D** has 4 frequency parameters ($\pi_A, \pi_C, \pi_G, \pi_T$, a transition rate (set to 1.0 as a reference) and a transversion rate α_1. Two parameters, λ_1 and λ_2, control the fitness of Watson-Crick and **G:U**/**U:G** pairs. The matrix is given in table 25 using the standard equilibrium frequency $\times$ exchangeability form but see Savill et al. [2001] since the effects of λ_1 and λ_2 on a standard DNA model are easier to understand from their transition matrix. If we define $1/\kappa = 2(\lambda_1^2 - 1)(\pi_A\pi_U + \pi_C\pi_G) + 2(\lambda_2^2 - 1)\pi_G\pi_U + 1$, the equilibrium frequencies for the 4 watson-crick pairs **X:Y** are $\pi_{XY} = \kappa\pi_X\pi_Y\lambda_1^2$, $\pi_{GU} = \pi_{UG} = \kappa\pi_G\pi_U\lambda_2^2$ and for each mismatch pair **X:Y** we have $\pi_{XY} = \pi_{YX} = \kappa\pi_X\pi_Y$.

RNA16E and RNA16F model [Muse, 1995]

RNA16D reduces to **RNA16E** if $\lambda_2 = 1$ (see matrix 26). In **RNA16E**, **G:U** pairs are treated as mismatches. **RNA16D** reduces to **RNA16F** if $\lambda_2 = \lambda_1$ (see matrix 27). In **RNA16F**, **G:U** pairs are treated as standard Watson-Crick pairs.

RNA16I model, RNA16J model and RNA16K model

If **RNA16B** can be considered as a F81-like model for doublet of nucleotides, then **RNA16I**, **RNA16J** and **RNA16K** model are respectively the REV-like, TN93-like and HKY85-like doublet versions of these standard 4-state models. We believe **RNA16I** and **RNA16K** are the doublet models implemented in MrBayes (with **RNA16B**) and they are provided in **PHASE** for completeness. If you plan on using just these models and only in a Bayesian framework, consider using MrBayes instead since it might be faster. Nevertheless, we are convinced that substitution models that allow for the double-transition perform better and we want to promote their use (see the Compensatory substitutions section).

In any cases, these models have 16 frequency parameters and 6/3/2 rate parameters respectively (5/2/1 if the reference rate is excluded). Only simple transitions/transversions are allowed. We give the transition matrix for the **RNA16I** model in table 28. **RNA16I** reduces to **RNA16J** with $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_5$ and **RNA16J** further reduces to **RNA16K** with $\alpha_4 = ref = 1$.

$$Q = m_r \times \begin{array}{c} \\ AU \\ GU \\ GC \\ UA \\ UG \\ CG \\ AA \\ AG \\ AC \\ GA \\ GG \\ CA \\ CC \\ CU \\ UC \\ UU \end{array} \begin{pmatrix} AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\ * & \pi_{GU} & 0 & 0 & 0 & 0 & \pi_{AA} & \pi_{AG} & \pi_{AC} & 0 & 0 & 0 & 0 & \pi_{CU} & 0 & \pi_{UU} \\ \pi_{AU} & * & \pi_{GC} & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{GA} & \pi_{GG} & 0 & 0 & \pi_{CU} & 0 & \pi_{UU} \\ 0 & \pi_{GU} & * & 0 & 0 & 0 & 0 & 0 & \pi_{AC} & \pi_{GA} & \pi_{GG} & 0 & \pi_{CC} & 0 & \pi_{UC} & 0 \\ 0 & 0 & 0 & * & \pi_{UG} & 0 & \pi_{AA} & 0 & 0 & \pi_{GA} & 0 & \pi_{CA} & 0 & 0 & \pi_{UC} & \pi_{UU} \\ 0 & 0 & 0 & \pi_{UA} & * & \pi_{CG} & 0 & \pi_{AG} & 0 & 0 & \pi_{GG} & 0 & 0 & 0 & \pi_{UC} & \pi_{UU} \\ 0 & 0 & 0 & 0 & \pi_{UG} & * & 0 & \pi_{AG} & 0 & 0 & \pi_{GG} & \pi_{CA} & \pi_{CC} & \pi_{CU} & 0 & 0 \\ \pi_{AU} & 0 & 0 & \pi_{UA} & 0 & 0 & * & \pi_{AG} & \pi_{AC} & \pi_{GA} & 0 & \pi_{CA} & 0 & 0 & 0 & 0 \\ \pi_{AU} & 0 & 0 & 0 & \pi_{UG} & \pi_{CG} & \pi_{AA} & * & \pi_{AC} & 0 & \pi_{GG} & 0 & 0 & 0 & 0 & 0 \\ \pi_{AU} & 0 & \pi_{GC} & 0 & 0 & 0 & \pi_{AA} & \pi_{AG} & * & 0 & 0 & 0 & \pi_{CC} & 0 & \pi_{UC} & 0 \\ 0 & \pi_{GU} & \pi_{GC} & \pi_{UA} & 0 & 0 & \pi_{AA} & 0 & 0 & * & \pi_{GG} & \pi_{CA} & 0 & 0 & 0 & 0 \\ 0 & \pi_{GU} & \pi_{GC} & 0 & \pi_{UG} & \pi_{CG} & 0 & \pi_{AG} & 0 & \pi_{GA} & * & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \pi_{UA} & 0 & \pi_{CG} & \pi_{AA} & 0 & 0 & \pi_{GA} & 0 & * & \pi_{CC} & \pi_{CU} & 0 & 0 \\ 0 & 0 & \pi_{GC} & 0 & 0 & \pi_{CG} & 0 & 0 & \pi_{AC} & 0 & 0 & \pi_{CA} & * & \pi_{CU} & \pi_{UC} & 0 \\ \pi_{AU} & \pi_{GU} & 0 & 0 & 0 & \pi_{CG} & 0 & 0 & 0 & 0 & 0 & \pi_{CA} & \pi_{CC} & * & 0 & \pi_{UU} \\ 0 & 0 & \pi_{GC} & \pi_{UA} & \pi_{UG} & 0 & 0 & 0 & \pi_{AC} & 0 & 0 & 0 & \pi_{CC} & 0 & * & \pi_{UU} \\ \pi_{AU} & \pi_{GU} & 0 & \pi_{UA} & \pi_{UG} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{CU} & \pi_{UC} & * \end{pmatrix}$$

Table 23: **RNA16B** transition matrix

$$Q = m_r \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\
AU & * & \pi_{GU} & 0 & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & 0 & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} \\
GU & \pi_{AU} & * & \pi_{GC} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} \\
GC & 0 & \pi_{GU} & * & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} & 0 \\
UA & 0 & 0 & 0 & * & \pi_{UG} & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} \\
UG & 0 & 0 & 0 & \pi_{UA} & * & \pi_{CG} & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} \\
CG & 0 & 0 & 0 & 0 & \pi_{UG} & * & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & 0 & 0 \\
AA & \pi_{AU} & 0 & 0 & \pi_{UA} & 0 & 0 & * & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & 0 & 0 \\
AG & \pi_{AU} & 0 & 0 & 0 & \pi_{UG} & \pi_{CG} & \frac{\pi_{MM}}{10} & * & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & 0 & 0 & 0 \\
AC & \pi_{AU} & 0 & \pi_{GC} & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & * & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} & 0 \\
GA & 0 & \pi_{GU} & \pi_{GC} & \pi_{UA} & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & * & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & 0 & 0 & 0 & 0 \\
GG & 0 & \pi_{GU} & \pi_{GC} & 0 & \pi_{UG} & \pi_{CG} & 0 & \frac{\pi_{MM}}{10} & 0 & \frac{\pi_{MM}}{10} & * & 0 & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \pi_{UA} & 0 & \pi_{CG} & \frac{\pi_{MM}}{10} & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & * & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & 0 & 0 \\
CC & 0 & 0 & \pi_{GC} & 0 & 0 & \pi_{CG} & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & \frac{\pi_{MM}}{10} & * & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & 0 \\
CU & \pi_{AU} & \pi_{GU} & 0 & 0 & 0 & \pi_{CG} & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & * & 0 & \frac{\pi_{MM}}{10} \\
UC & 0 & 0 & \pi_{GC} & \pi_{UA} & \pi_{UG} & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & 0 & * & \frac{\pi_{MM}}{10} \\
UU & \pi_{AU} & \pi_{GU} & 0 & \pi_{UA} & \pi_{UG} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{MM}}{10} & \frac{\pi_{MM}}{10} & *
\end{pmatrix}$$

Table 24: **RNA16C** transition matrix

$$Q = m_r \times \frac{1}{\kappa} \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\
AU & * & \frac{\pi_{GU}}{\lambda_1\lambda_2\pi_U} & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AA}}{\lambda_1\pi_A} & \frac{\alpha\pi_{AG}}{\lambda_1\pi_A} & \frac{\pi_{AC}}{\lambda_1\pi_A} & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_1\pi_U} & 0 & \frac{\alpha\pi_{UU}}{\lambda_1\pi_U} \\
GU & \frac{\pi_{AU}}{\lambda_1\lambda_2\pi_U} & * & \frac{\pi_{GC}}{\lambda_1\lambda_2\pi_G} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AG}}{\lambda_2\pi_G} & \frac{\alpha\pi_{GG}}{\lambda_2\pi_G} & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_2\pi_U} & 0 & \frac{\alpha\pi_{UU}}{\lambda_2\pi_U} \\
GC & 0 & \frac{\pi_{GU}}{\lambda_1\lambda_2\pi_G} & * & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{AC}}{\lambda_1\pi_C} & \frac{\alpha\pi_{AG}}{\lambda_1\pi_G} & \frac{\alpha\pi_{GG}}{\lambda_1\pi_G} & 0 & \frac{\alpha\pi_{CC}}{\lambda_1\pi_C} & 0 & \frac{\alpha\pi_{CU}}{\lambda_1\pi_C} & 0 \\
UA & 0 & 0 & 0 & * & \frac{\pi_{GU}}{\lambda_1\lambda_2\pi_U} & 0 & \frac{\alpha\pi_{AA}}{\lambda_1\pi_A} & 0 & 0 & \frac{\alpha\pi_{AG}}{\lambda_1\pi_A} & 0 & \frac{\alpha\pi_{AC}}{\lambda_1\pi_A} & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_1\pi_U} & \frac{\alpha\pi_{UU}}{\lambda_1\pi_U} \\
UG & 0 & 0 & 0 & \frac{\pi_{AU}}{\lambda_1\lambda_2\pi_U} & * & \frac{\pi_{GC}}{\lambda_1\lambda_2\pi_G} & 0 & \frac{\alpha\pi_{AG}}{\lambda_2\pi_G} & 0 & 0 & \frac{\alpha\pi_{GG}}{\lambda_2\pi_G} & 0 & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_2\pi_U} & \frac{\alpha\pi_{UU}}{\lambda_2\pi_U} \\
CG & 0 & 0 & 0 & 0 & \frac{\pi_{GU}}{\lambda_1\lambda_2\pi_G} & * & 0 & \frac{\alpha\pi_{AG}}{\lambda_1\pi_G} & 0 & 0 & \frac{\alpha\pi_{GG}}{\lambda_1\pi_G} & \frac{\pi_{AC}}{\lambda_1\pi_C} & \frac{\alpha\pi_{CC}}{\lambda_1\pi_C} & \frac{\alpha\pi_{CU}}{\lambda_1\pi_C} & 0 & 0 \\
AA & \frac{\alpha\pi_{AU}}{\lambda_1\pi_A} & 0 & 0 & \frac{\alpha\pi_{AU}}{\lambda_1\pi_A} & 0 & 0 & * & \frac{\pi_{AG}}{\pi_A} & \frac{\alpha\pi_{AC}}{\pi_A} & \frac{\pi_{AG}}{\pi_A} & 0 & \frac{\pi_{AC}}{\pi_A} & 0 & 0 & 0 & 0 \\
AG & \frac{\alpha\pi_{AU}}{\lambda_1\pi_A} & 0 & 0 & 0 & \frac{\alpha\pi_{GU}}{\lambda_2\pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1\pi_G} & \frac{\pi_{AA}}{\pi_A} & * & \frac{\alpha\pi_{AC}}{\pi_A} & 0 & \frac{\pi_{GG}}{\pi_G} & 0 & 0 & 0 & 0 & 0 \\
AC & \frac{\pi_{AU}}{\lambda_1\pi_A} & 0 & \frac{\pi_{GC}}{\lambda_1\pi_C} & 0 & 0 & 0 & \frac{\alpha\pi_{AA}}{\pi_A} & \frac{\alpha\pi_{AG}}{\pi_A} & * & 0 & 0 & 0 & \frac{\alpha\pi_{CC}}{\pi_C} & 0 & \frac{\alpha\pi_{CU}}{\pi_C} & 0 \\
GA & 0 & \frac{\alpha\pi_{GU}}{\lambda_2\pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1\pi_G} & \frac{\alpha\pi_{AU}}{\lambda_1\pi_A} & 0 & 0 & \frac{\alpha\pi_{AA}}{\pi_A} & 0 & 0 & * & \frac{\pi_{GG}}{\pi_G} & \frac{\alpha\pi_{AC}}{\pi_A} & 0 & 0 & 0 & 0 \\
GG & 0 & \frac{\alpha\pi_{GU}}{\lambda_2\pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1\pi_G} & 0 & \frac{\alpha\pi_{GU}}{\lambda_2\pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1\pi_G} & 0 & \frac{\pi_{AG}}{\pi_G} & 0 & \frac{\pi_{AG}}{\pi_G} & * & 0 & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \frac{\alpha\pi_{AU}}{\lambda_1\pi_A} & 0 & \frac{\pi_{GC}}{\lambda_1\pi_C} & \frac{\pi_{AA}}{\pi_A} & 0 & 0 & \frac{\alpha\pi_{AG}}{\pi_A} & 0 & * & \frac{\alpha\pi_{CC}}{\pi_C} & \frac{\alpha\pi_{CU}}{\pi_C} & 0 & 0 \\
CC & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1\pi_C} & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1\pi_C} & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & * & \frac{\pi_{CU}}{\pi_C} & \frac{\pi_{CU}}{\pi_C} & 0 \\
CU & \frac{\alpha\pi_{AU}}{\lambda_1\pi_U} & \frac{\alpha\pi_{GU}}{\lambda_2\pi_U} & 0 & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1\pi_C} & 0 & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & \frac{\pi_{CC}}{\pi_C} & * & 0 & \frac{\pi_{UU}}{\pi_U} \\
UC & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1\pi_C} & \frac{\alpha\pi_{AU}}{\lambda_1\pi_U} & \frac{\alpha\pi_{GU}}{\lambda_2\pi_U} & 0 & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & 0 & 0 & 0 & \frac{\pi_{CC}}{\pi_C} & 0 & * & \frac{\pi_{UU}}{\pi_U} \\
UU & \frac{\alpha\pi_{AU}}{\lambda_1\pi_U} & \frac{\alpha\pi_{GU}}{\lambda_2\pi_U} & 0 & \frac{\alpha\pi_{AU}}{\lambda_1\pi_U} & \frac{\alpha\pi_{GU}}{\lambda_2\pi_U} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{CU}}{\pi_U} & \frac{\pi_{CU}}{\pi_U} & *
\end{pmatrix}$$

Table 25: **RNA16D** transition matrix

$$Q = m_r \times \frac{1}{\kappa} \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\
AU & * & \frac{\pi_{GU}}{\lambda_1 \pi_U} & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AA}}{\lambda_1 \pi_A} & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_A} & \frac{\pi_{AC}}{\lambda_1 \pi_A} & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_U} & 0 & \frac{\alpha\pi_{UU}}{\lambda_1 \pi_U} \\
GU & \frac{\pi_{AU}}{\lambda_1 \pi_U} & * & \frac{\pi_{GC}}{\lambda_1 \pi_G} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AG}}{\pi_G} & \frac{\alpha\pi_{GG}}{\pi_G} & 0 & 0 & \frac{\alpha\pi_{CU}}{\pi_U} & 0 & \frac{\alpha\pi_{UU}}{\pi_U} \\
GC & 0 & \frac{\pi_{GU}}{\lambda_1 \pi_G} & * & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{AC}}{\lambda_1 \pi_C} & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{GG}}{\lambda_1 \pi_G} & 0 & \frac{\alpha\pi_{CC}}{\lambda_1 \pi_C} & 0 & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_C} & 0 \\
UA & 0 & 0 & 0 & * & \frac{\pi_{GU}}{\lambda_1 \pi_U} & 0 & \frac{\alpha\pi_{AA}}{\lambda_1 \pi_A} & 0 & 0 & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_A} & 0 & \frac{\alpha\pi_{AC}}{\lambda_1 \pi_A} & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{UU}}{\lambda_1 \pi_U} \\
UG & 0 & 0 & 0 & \frac{\pi_{AU}}{\lambda_1 \pi_U} & * & \frac{\pi_{GC}}{\lambda_1 \pi_G} & 0 & \frac{\alpha\pi_{AG}}{\pi_G} & 0 & 0 & \frac{\alpha\pi_{GG}}{\pi_G} & 0 & 0 & 0 & \frac{\alpha\pi_{CU}}{\pi_U} & \frac{\alpha\pi_{UU}}{\pi_U} \\
CG & 0 & 0 & 0 & 0 & \frac{\pi_{GU}}{\lambda_1 \pi_G} & * & 0 & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_G} & 0 & 0 & \frac{\alpha\pi_{GG}}{\lambda_1 \pi_G} & \frac{\pi_{AC}}{\lambda_1 \pi_C} & \frac{\alpha\pi_{CC}}{\lambda_1 \pi_C} & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_C} & 0 & 0 \\
AA & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & 0 & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & 0 & * & \frac{\pi_{AG}}{\pi_A} & \frac{\alpha\pi_{AC}}{\pi_A} & \frac{\pi_{AG}}{\pi_A} & 0 & \frac{\pi_{AC}}{\pi_A} & 0 & 0 & 0 & 0 \\
AG & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & 0 & 0 & \frac{\alpha\pi_{GU}}{\pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_G} & \frac{\pi_{AA}}{\pi_A} & * & \frac{\alpha\pi_{AC}}{\pi_A} & 0 & \frac{\pi_{GG}}{\pi_G} & 0 & 0 & 0 & 0 & 0 \\
AC & \frac{\pi_{AU}}{\lambda_1 \pi_A} & 0 & \frac{\pi_{GC}}{\lambda_1 \pi_C} & 0 & 0 & 0 & \frac{\alpha\pi_{AA}}{\pi_A} & \frac{\alpha\pi_{AG}}{\pi_A} & * & 0 & 0 & 0 & \frac{\alpha\pi_{CC}}{\pi_C} & 0 & \frac{\alpha\pi_{CU}}{\pi_C} & 0 \\
GA & 0 & \frac{\alpha_1\pi_{GU}}{\pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & 0 & \frac{\alpha\pi_{AA}}{\pi_A} & 0 & 0 & * & \frac{\pi_{GG}}{\pi_G} & \frac{\alpha\pi_{AC}}{\pi_A} & 0 & 0 & 0 & 0 \\
GG & 0 & \frac{\alpha_1\pi_{GU}}{\pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_G} & \alpha_1 0 & \frac{\alpha\pi_{GU}}{\pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_G} & 0 & \frac{\pi_{AG}}{\pi_G} & 0 & \frac{\pi_{AG}}{\pi_G} & * & 0 & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & \frac{\pi_{GC}}{\lambda_1 \pi_C} & \frac{\pi_{AA}}{\pi_A} & 0 & 0 & \frac{\alpha\pi_{AG}}{\pi_A} & 0 & * & \frac{\alpha\pi_{CC}}{\pi_C} & \frac{\alpha\pi_{CU}}{\pi_C} & 0 & 0 \\
CC & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_C} & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_C} & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & * & \frac{\pi_{CU}}{\pi_C} & \frac{\pi_{CU}}{\pi_C} & 0 \\
CU & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_U} & \frac{\alpha_1\pi_{GU}}{\pi_U} & 0 & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_C} & 0 & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & \frac{\pi_{CC}}{\pi_C} & * & 0 & \frac{\pi_{UU}}{\pi_U} \\
UC & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_C} & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{GU}}{\pi_U} & 0 & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & 0 & 0 & 0 & \frac{\pi_{CC}}{\pi_C} & 0 & * & \frac{\pi_{UU}}{\pi_U} \\
UU & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_U} & \frac{\alpha_1\pi_{GU}}{\pi_U} & 0 & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{GU}}{\pi_U} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{CU}}{\pi_U} & \frac{\pi_{CU}}{\pi_U} & *
\end{pmatrix}$$

Table 26: **RNA16E** transition matrix

$$Q = m_r \times \frac{1}{\kappa} \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\
AU & * & \frac{\pi_{GU}}{\lambda_1 \lambda_1 \pi_U} & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AA}}{\lambda_1 \pi_A} & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_A} & \frac{\pi_{AC}}{\lambda_1 \pi_A} & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_U} & 0 & \frac{\alpha\pi_{UU}}{\lambda_1 \pi_U} \\
GU & \frac{\pi_{AU}}{\lambda_1 \lambda_1 \pi_U} & * & \frac{\pi_{GC}}{\lambda_1 \lambda_1 \pi_G} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{GG}}{\lambda_1 \pi_G} & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_U} & 0 & \frac{\alpha\pi_{UU}}{\lambda_1 \pi_U} \\
GC & 0 & \frac{\pi_{GU}}{\lambda_1 \lambda_1 \pi_G} & * & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{AC}}{\lambda_1 \pi_C} & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{GG}}{\lambda_1 \pi_G} & 0 & \frac{\alpha\pi_{CC}}{\lambda_1 \pi_C} & 0 & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_C} & 0 \\
UA & 0 & 0 & 0 & * & \frac{\pi_{GU}}{\lambda_1 \lambda_1 \pi_U} & 0 & \frac{\alpha\pi_{AA}}{\lambda_1 \pi_A} & 0 & 0 & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_A} & 0 & \frac{\alpha\pi_{AC}}{\lambda_1 \pi_A} & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{UU}}{\lambda_1 \pi_U} \\
UG & 0 & 0 & 0 & \frac{\pi_{AU}}{\lambda_1 \lambda_1 \pi_U} & * & \frac{\pi_{GC}}{\lambda_1 \lambda_1 \pi_G} & 0 & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_G} & 0 & 0 & \frac{\alpha\pi_{GG}}{\lambda_1 \pi_G} & 0 & 0 & 0 & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{UU}}{\lambda_1 \pi_U} \\
CG & 0 & 0 & 0 & 0 & \frac{\pi_{GU}}{\lambda_1 \lambda_1 \pi_G} & * & 0 & \frac{\alpha\pi_{AG}}{\lambda_1 \pi_G} & 0 & 0 & \frac{\alpha\pi_{GG}}{\lambda_1 \pi_G} & \frac{\pi_{AC}}{\lambda_1 \pi_C} & \frac{\alpha\pi_{CC}}{\lambda_1 \pi_C} & \frac{\alpha\pi_{CU}}{\lambda_1 \pi_C} & 0 & 0 \\
AA & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & 0 & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & 0 & * & \frac{\pi_{AG}}{\pi_A} & \frac{\alpha\pi_{AC}}{\pi_A} & \frac{\pi_{AG}}{\pi_A} & 0 & \frac{\pi_{AC}}{\pi_A} & 0 & 0 & 0 & 0 \\
AG & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & 0 & 0 & \frac{\alpha\pi_{GU}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_G} & \frac{\pi_{AA}}{\pi_A} & * & \frac{\alpha\pi_{AC}}{\pi_A} & 0 & \frac{\pi_{GG}}{\pi_G} & 0 & 0 & 0 & 0 & 0 \\
AC & \frac{\pi_{AU}}{\lambda_1 \pi_A} & 0 & \frac{\pi_{GC}}{\lambda_1 \pi_C} & 0 & 0 & 0 & \frac{\alpha\pi_{AA}}{\pi_A} & \frac{\alpha\pi_{AG}}{\pi_A} & * & 0 & 0 & 0 & \frac{\alpha\pi_{CC}}{\pi_C} & 0 & \frac{\alpha\pi_{CU}}{\pi_C} & 0 \\
GA & 0 & \frac{\alpha\pi_{GU}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & 0 & \frac{\alpha\pi_{AA}}{\pi_A} & 0 & 0 & * & \frac{\pi_{GG}}{\pi_G} & \frac{\alpha\pi_{AC}}{\pi_A} & 0 & 0 & 0 & 0 \\
GG & 0 & \frac{\alpha\pi_{GU}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_G} & 0 & \frac{\alpha\pi_{GU}}{\lambda_1 \pi_G} & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_G} & 0 & \frac{\pi_{AG}}{\pi_G} & 0 & \frac{\pi_{AG}}{\pi_G} & * & 0 & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_A} & 0 & \frac{\pi_{GC}}{\lambda_1 \pi_C} & \frac{\pi_{AA}}{\pi_A} & 0 & 0 & \frac{\alpha\pi_{AG}}{\pi_A} & 0 & * & \frac{\alpha\pi_{CC}}{\pi_C} & \frac{\alpha\pi_{CU}}{\pi_C} & 0 & 0 \\
CC & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_C} & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_C} & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & * & \frac{\pi_{CU}}{\pi_C} & \frac{\pi_{CU}}{\pi_C} & 0 \\
CU & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{GU}}{\lambda_1 \pi_U} & 0 & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_C} & 0 & 0 & 0 & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & \frac{\pi_{CC}}{\pi_C} & * & 0 & \frac{\pi_{UU}}{\pi_U} \\
UC & 0 & 0 & \frac{\alpha\pi_{GC}}{\lambda_1 \pi_C} & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{GU}}{\lambda_1 \pi_U} & 0 & 0 & 0 & \frac{\alpha\pi_{AC}}{\pi_C} & 0 & 0 & 0 & \frac{\pi_{CC}}{\pi_C} & 0 & * & \frac{\pi_{UU}}{\pi_U} \\
UU & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{GU}}{\lambda_1 \pi_U} & 0 & \frac{\alpha\pi_{AU}}{\lambda_1 \pi_U} & \frac{\alpha\pi_{GU}}{\lambda_1 \pi_U} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\pi_{CU}}{\pi_U} & \frac{\pi_{CU}}{\pi_U} & *
\end{pmatrix}$$

Table 27: **RNA16F** transition matrix

$$Q = m_r \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\
AU & * & \pi_{GU} & 0 & 0 & 0 & 0 & \pi_{AA}\alpha_2 & \pi_{AG}\alpha_5 & \pi_{AC}\alpha_4 & 0 & 0 & 0 & 0 & \pi_{CU}\alpha_1 & 0 & \pi_{UU}\alpha_2 \\
GU & \pi_{AU} & * & \pi_{GC}\alpha_4 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{GA}\alpha_2 & \pi_{GG}\alpha_5 & 0 & 0 & \pi_{CU}\alpha_3 & 0 & \pi_{UU}\alpha_5 \\
GC & 0 & \pi_{GU}\alpha_4 & * & 0 & 0 & 0 & 0 & 0 & \pi_{AC} & \pi_{GA}\alpha_1 & \pi_{GG}\alpha_3 & 0 & \pi_{CC}\alpha_3 & 0 & \pi_{UC}\alpha_5 & 0 \\
UA & 0 & 0 & 0 & * & \pi_{UG} & 0 & \pi_{AA}\alpha_2 & 0 & 0 & \pi_{GA}\alpha_5 & 0 & \pi_{CA}\alpha_4 & 0 & 0 & \pi_{UC}\alpha_1 & \pi_{UU}\alpha_2 \\
UG & 0 & 0 & 0 & \pi_{UA} & * & \pi_{CG}\alpha_4 & 0 & \pi_{AG}\alpha_2 & 0 & 0 & \pi_{GG}\alpha_4 & 0 & 0 & 0 & \pi_{UC}\alpha_3 & \pi_{UU}\alpha_5 \\
CG & 0 & 0 & 0 & 0 & \pi_{UG}\alpha_4 & * & 0 & \pi_{AG}\alpha_1 & 0 & 0 & \pi_{GG}\alpha_3 & \pi_{CA} & \pi_{CC}\alpha_3 & \pi_{CU}\alpha_5 & 0 & 0 \\
AA & \pi_{AU}\alpha_2 & 0 & 0 & \pi_{UA}\alpha_2 & 0 & 0 & * & \pi_{AG} & \pi_{AC}\alpha_1 & \pi_{GA} & 0 & \pi_{CA}\alpha_1 & 0 & 0 & 0 & 0 \\
AG & \pi_{AU}\alpha_5 & 0 & 0 & 0 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_1 & \pi_{AA} & * & \pi_{AC}\alpha_3 & 0 & \pi_{GG} & 0 & 0 & 0 & 0 & 0 \\
AC & \pi_{AU}\alpha_4 & 0 & \pi_{GC} & 0 & 0 & 0 & \pi_{AA}\alpha_1 & \pi_{AG}\alpha_3 & * & 0 & 0 & 0 & \pi_{CC}\alpha_1 & 0 & \pi_{UC}\alpha_2 & 0 \\
GA & 0 & \pi_{GU}\alpha_2 & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_5 & 0 & 0 & \pi_{AA} & 0 & 0 & * & \pi_{GG} & \pi_{CA}\alpha_3 & 0 & 0 & 0 & 0 \\
GG & 0 & \pi_{GU}\alpha_5 & \pi_{GC}\alpha_3 & 0 & \pi_{UG}\alpha_4 & \pi_{CG}\alpha_3 & 0 & \pi_{AG} & 0 & \pi_{GA} & * & 0 & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \pi_{UA}\alpha_4 & 0 & \pi_{CG} & \pi_{AA}\alpha_1 & 0 & 0 & \pi_{GA}\alpha_3 & 0 & * & \pi_{CC}\alpha_1 & \pi_{CU}\alpha_2 & 0 & 0 \\
CC & 0 & 0 & \pi_{GC}\alpha_3 & 0 & 0 & \pi_{CG}\alpha_3 & 0 & 0 & \pi_{AC}\alpha_1 & 0 & 0 & \pi_{CA}\alpha_1 & * & \pi_{CU}\alpha_4 & \pi_{UC}\alpha_4 & 0 \\
CU & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_3 & 0 & 0 & 0 & \pi_{CG}\alpha_5 & 0 & 0 & 0 & 0 & 0 & \pi_{CA}\alpha_2 & \pi_{CC}\alpha_4 & * & 0 & \pi_{UU}\alpha_4 \\
UC & 0 & 0 & \pi_{GC}\alpha_5 & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_3 & 0 & 0 & 0 & \pi_{AC}\alpha_2 & 0 & 0 & 0 & \pi_{CC}\alpha_4 & 0 & * & \pi_{UU}\alpha_4 \\
UU & \pi_{AU}\alpha_2 & \pi_{GU}\alpha_5 & 0 & \pi_{UA}\alpha_2 & \pi_{UG}\alpha_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{CU}\alpha_4 & \pi_{UC}\alpha_4 & *
\end{pmatrix}$$

Table 28: **RNA16I** transition matrix

$$Q = m_r \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\
AU & * & \pi_{GU} & 0 & 0 & 0 & 0 & \pi_{AA}\alpha_1 & \pi_{AG}\alpha_1 & \pi_{AC}\alpha_2 & 0 & 0 & 0 & 0 & \pi_{CU}\alpha_1 & 0 & \pi_{UU}\alpha_1 \\
GU & \pi_{AU} & * & \pi_{GC}\alpha_2 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{GA}\alpha_1 & \pi_{GG}\alpha_1 & 0 & 0 & \pi_{CU}\alpha_1 & 0 & \pi_{UU}\alpha_1 \\
GC & 0 & \pi_{GU}\alpha_2 & * & 0 & 0 & 0 & 0 & 0 & \pi_{AC} & \pi_{GA}\alpha_1 & \pi_{GG}\alpha_1 & 0 & \pi_{CC}\alpha_1 & 0 & \pi_{UC}\alpha_1 & 0 \\
UA & 0 & 0 & 0 & * & \pi_{UG} & 0 & \pi_{AA}\alpha_1 & 0 & 0 & \pi_{GA}\alpha_1 & 0 & \pi_{CA}\alpha_2 & 0 & 0 & \pi_{UC}\alpha_1 & \pi_{UU}\alpha_1 \\
UG & 0 & 0 & 0 & \pi_{UA} & * & \pi_{CG}\alpha_2 & 0 & \pi_{AG}\alpha_1 & 0 & 0 & \pi_{GG}\alpha_2 & 0 & 0 & 0 & \pi_{UC}\alpha_1 & \pi_{UU}\alpha_1 \\
CG & 0 & 0 & 0 & 0 & \pi_{UG}\alpha_2 & * & 0 & \pi_{AG}\alpha_1 & 0 & 0 & \pi_{GG}\alpha_1 & \pi_{CA} & \pi_{CC}\alpha_1 & \pi_{CU}\alpha_1 & 0 & 0 \\
AA & \pi_{AU}\alpha_1 & 0 & 0 & \pi_{UA}\alpha_1 & 0 & 0 & * & \pi_{AG} & \pi_{AC}\alpha_1 & \pi_{GA} & 0 & \pi_{CA}\alpha_1 & 0 & 0 & 0 & 0 \\
AG & \pi_{AU}\alpha_1 & 0 & 0 & 0 & \pi_{UG}\alpha_1 & \pi_{CG}\alpha_1 & \pi_{AA} & * & \pi_{AC}\alpha_1 & 0 & \pi_{GG} & 0 & 0 & 0 & 0 & 0 \\
AC & \pi_{AU}\alpha_2 & 0 & \pi_{GC} & 0 & 0 & 0 & \pi_{AA}\alpha_1 & \pi_{AG}\alpha_1 & * & 0 & 0 & 0 & \pi_{CC}\alpha_1 & 0 & \pi_{UC}\alpha_1 & 0 \\
GA & 0 & \pi_{GU}\alpha_1 & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_1 & 0 & 0 & \pi_{AA} & 0 & 0 & * & \pi_{GG} & \pi_{CA}\alpha_1 & 0 & 0 & 0 & 0 \\
GG & 0 & \pi_{GU}\alpha_1 & \pi_{GC}\alpha_1 & 0 & \pi_{UG}\alpha_2 & \pi_{CG}\alpha_1 & 0 & \pi_{AG} & 0 & \pi_{GA} & * & 0 & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \pi_{UA}\alpha_2 & 0 & \pi_{CG} & \pi_{AA}\alpha_1 & 0 & 0 & \pi_{GA}\alpha_1 & 0 & * & \pi_{CC}\alpha_1 & \pi_{CU}\alpha_1 & 0 & 0 \\
CC & 0 & 0 & \pi_{GC}\alpha_1 & 0 & 0 & \pi_{CG}\alpha_1 & 0 & 0 & \pi_{AC}\alpha_1 & 0 & 0 & \pi_{CA}\alpha_1 & * & \pi_{CU}\alpha_2 & \pi_{UC}\alpha_2 & 0 \\
CU & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_1 & 0 & 0 & 0 & \pi_{CG}\alpha_1 & 0 & 0 & 0 & 0 & 0 & \pi_{CA}\alpha_1 & \pi_{CC}\alpha_2 & * & 0 & \pi_{UU}\alpha_2 \\
UC & 0 & 0 & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_1 & 0 & 0 & 0 & \pi_{AC}\alpha_1 & 0 & 0 & 0 & \pi_{CC}\alpha_2 & 0 & * & \pi_{UU}\alpha_2 \\
UU & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_1 & 0 & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{CU}\alpha_2 & \pi_{UC}\alpha_2 & *
\end{pmatrix}$$

Table 29: **RNA16J** transition matrix

$$Q = m_r \times \begin{pmatrix}
 & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \\
AU & * & \pi_{GU} & 0 & 0 & 0 & 0 & \pi_{AA}\alpha_1 & \pi_{AG}\alpha_1 & \pi_{AC} & 0 & 0 & 0 & 0 & \pi_{CU}\alpha_1 & 0 & \pi_{UU}\alpha_1 \\
GU & \pi_{AU} & * & \pi_{GC} & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{GA}\alpha_1 & \pi_{GG}\alpha_1 & 0 & 0 & \pi_{CU}\alpha_1 & 0 & \pi_{UU}\alpha_1 \\
GC & 0 & \pi_{GU} & * & 0 & 0 & 0 & 0 & 0 & \pi_{AC} & \pi_{GA}\alpha_1 & \pi_{GG}\alpha_1 & 0 & \pi_{CC}\alpha_1 & 0 & \pi_{UC}\alpha_1 & 0 \\
UA & 0 & 0 & 0 & * & \pi_{UG} & 0 & \pi_{AA}\alpha_1 & 0 & 0 & \pi_{GA}\alpha_1 & 0 & \pi_{CA} & 0 & 0 & \pi_{UC}\alpha_1 & \pi_{UU}\alpha_1 \\
UG & 0 & 0 & 0 & \pi_{UA} & * & \pi_{CG} & 0 & \pi_{AG}\alpha_1 & 0 & 0 & \pi_{GG} & 0 & 0 & 0 & \pi_{UC}\alpha_1 & \pi_{UU}\alpha_1 \\
CG & 0 & 0 & 0 & 0 & \pi_{UG} & * & 0 & \pi_{AG}\alpha_1 & 0 & 0 & \pi_{GG}\alpha_1 & \pi_{CA} & \pi_{CC}\alpha_1 & \pi_{CU}\alpha_1 & 0 & 0 \\
AA & \pi_{AU}\alpha_1 & 0 & 0 & \pi_{UA}\alpha_1 & 0 & 0 & * & \pi_{AG} & \pi_{AC}\alpha_1 & \pi_{GA} & 0 & \pi_{CA}\alpha_1 & 0 & 0 & 0 & 0 \\
AG & \pi_{AU}\alpha_1 & 0 & 0 & 0 & \pi_{UG}\alpha_1 & \pi_{CG}\alpha_1 & \pi_{AA} & * & \pi_{AC}\alpha_1 & 0 & \pi_{GG} & 0 & 0 & 0 & 0 & 0 \\
AC & \pi_{AU} & 0 & \pi_{GC} & 0 & 0 & 0 & \pi_{AA}\alpha_1 & \pi_{AG}\alpha_1 & * & 0 & 0 & 0 & \pi_{CC}\alpha_1 & 0 & \pi_{UC}\alpha_1 & 0 \\
GA & 0 & \pi_{GU}\alpha_1 & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_1 & 0 & 0 & \pi_{AA} & 0 & 0 & * & \pi_{GG} & \pi_{CA}\alpha_1 & 0 & 0 & 0 & 0 \\
GG & 0 & \pi_{GU}\alpha_1 & \pi_{GC}\alpha_1 & 0 & \pi_{UG} & \pi_{CG}\alpha_1 & 0 & \pi_{AG} & 0 & \pi_{GA} & * & 0 & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \pi_{UA} & 0 & \pi_{CG} & \pi_{AA}\alpha_1 & 0 & 0 & \pi_{GA}\alpha_1 & 0 & * & \pi_{CC}\alpha_1 & \pi_{CU}\alpha_1 & 0 & 0 \\
CC & 0 & 0 & \pi_{GC}\alpha_1 & 0 & 0 & \pi_{CG}\alpha_1 & 0 & 0 & \pi_{AC}\alpha_1 & 0 & 0 & \pi_{CA}\alpha_1 & * & \pi_{CU} & \pi_{UC} & 0 \\
CU & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_1 & 0 & 0 & 0 & \pi_{CG}\alpha_1 & 0 & 0 & 0 & 0 & 0 & \pi_{CA}\alpha_1 & \pi_{CC} & * & 0 & \pi_{UU} \\
UC & 0 & 0 & \pi_{GC}\alpha_1 & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_1 & 0 & 0 & 0 & \pi_{AC}\alpha_1 & 0 & 0 & 0 & \pi_{CC} & 0 & * & \pi_{UU} \\
UU & \pi_{AU}\alpha_1 & \pi_{GU}\alpha_1 & 0 & \pi_{UA}\alpha_1 & \pi_{UG}\alpha_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{CU} & \pi_{UC} & *
\end{pmatrix}$$

Table 30: **RNA16K** transition matrix

Refinements to substitution models

In this section we introduce some refinements made to the substitution models described above.

Invariant and discrete gamma models

Substitution rates are definitely variable over sites of a sequence for many real dataset if not all. Including the heterogeneity of rates in substitution models is widely recognized as an important factor in the fitting to data. One attempt to take this acknowledged biological fact into account is to suppose that a proportion of sites are invariant while others evolve at the same single rate. **PHASE** provides this invariant model. One extra parameter controls the proportion of sites with zero rate of evolution.

Models that allows continuous variability of mutation rates over sites are more realistic and the gamma model of Yang [1994] outperforms the invariant model. The discrete gamma model is implemented in **PHASE**. The continuous rate distribution is approximated with a discrete distribution which is computational tractable and sites are divided into k equally probable rate categories. A single parameter α governs the shape of this distribution and the substitution rates for all categories. The mean $E(r)$ of the gamma distribution is the average mutation rate of our substitution model as stated earlier and its variance is $V(r) = E(r)^2/\alpha$. A small alpha suggests that rates differ significantly between sites with few sites having high rates and others being practically invariant; on the contrary, large α models weak rate heterogeneity (see figure). When $\alpha \to +\infty$, the gamma model reduces to the single rate model. Computational requirement of the discrete gamma model is roughly linear, *i.e.*, the application of a discrete gamma model with k categories is about k times slower than the use of a model where rate heterogeneity is not considered.

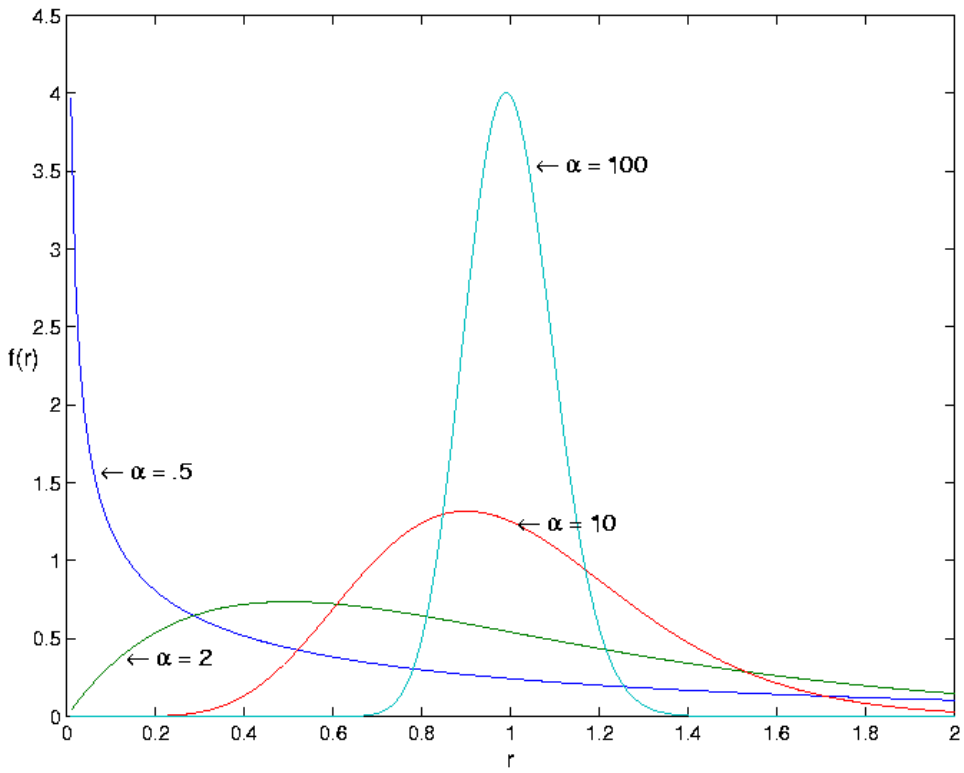

Figure 5: Probability density function of several gamma distributions of rate heterogeneity with mean $E(r) = 1$

The MIXED model

Since the current trend in phylogenetic analysis is to use several genes and/or several sorts of sequences at once, models were designed for combined analyses of heterogeneous sequence data from the same set of species [Yang, 1996]. **PHASE** allows the use of multiple substitution models

simultaneously to handle concatenated sequences, each substitution model having its own independent set of parameters. The average mutation rate of the first model is still set to 1.0 but the average mutation rate of the others are now free parameters of the model. The **MIXED** model for combined analysis of heterogeneous data is equivalent to the model with proportional branch lengths described in Yang [1996].

The HETEROGENEOUS model

The homogeneity hypothesis implies that the substitution process ultimately reaches an equilibrium and it is also assumed that the process was already stationary at the very beginning, i.e. at the root of the phylogeny. If the homogeneity and stationarity assumptions were true, equal nucleotide frequencies would be expected in past and present-day sequences. Actually, we can observe discrepancy's in nucleotide frequencies in many real data sets of present species: model assumptions are clearly violated when using real sequences.

It has been noticed that sequences of similar composition tend to be grouped together irrespective of their real phylogenetic relationships [Lockhart et al., 1994, Tarrio et al., 2001, see e.g.]. In an attempt to avoid this bias, we developed a **HETEROGENEOUS** model in a Bayesian framework which models coarsely the heterogeneity using a small pool of homogeneous processes. Each branch of the tree "chooses" a substitution model among them. The likelihood computation now depends on the position of the root which is why a heterogeneous rooted tree was implemented in **PHASE** (ultrametricity is optional). The composition observed at this root becomes a free parameter of the model [Yang and Roberts, 1995, Galtier and Gouy, 1998, see, e.g.].

Algorithms developed in **PHASE** are very similar to those implemented in **PFOUR** by Peter Foster. The **PHASE** framework might be a bit more general since the full substitution model is allowed to vary over the tree. However, you are strongly advised to limit yourself to variation of the composition vector as Foster [2004] did. Unfortunately, we did not have time to implement a way to use a single exchangeability matrix over the whole tree yet. There is a workaround (a bit unsatisfactory): you can start the MCMC chain from a model properly initialized and turn off the perturbation of rate ratios so that they have constant values. Use the same trick to fix the gamma shape parameter and the proportion of invariant sites to a single constant value. (Do not use a +I model with **HETEROGENEOUS** without constraining the proportion of invariant sites to a constant value).

PHASE is missing an efficient MCMC proposal to modify the position of the root. You are advised to use an outgroup in the **TREE** block to constrain the position of the common ancestor. Remember that this outgroup can also be a monophyletic cluster.

Bibliography

D. Aldous. Probability distributions on cladograms. In *Random Discrete Structures*, pages 1–18. Aldous, D. and Pemantle, R., springer (ima volumes math. appl. 76) edition, 1996.

J. Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *J. Mol. Evol.*, 17:368–376, 1981.

P. G. Foster. Modeling compositional heterogenity. *Syst. Biol.*, 53(3):485–495, 2004.

N. Galtier and M. Gouy. Inferring pattern and process: maximum-likelihood implementation of a nonhomogeneous model of DNA sequence evolution for phylogenetic analysis. *Mol. Biol. Evol.*, 15(7):871–879, 1998.

M. Hasegawa, H. Kishino, and T. Yano. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.*, 22(2):160–174, 1985.

P. G. Higgs. Compensatory neutral mutation and the evolution of RNA. *Genetica*, 102:91–101, 1998.

P. G. Higgs. Rna secondary structure: physical and computational aspects. *Quart. Rev. of Bioph.*, 22:199–253, 2000.

J.P. Huelsenbeck, B. Larget, R.E. Miller, and F. Ronquist. Potential applications and pitfalls of bayesian inference of phylogeny. *Syst. Biol.*, 51(5):673–688, 2002.

H. Jow, C. Hudelot, M. Rattray, and P. G. Higgs. Bayesian phylogenetics using an RNA substitution model applied to early mammalian evolution. *Mol. Biol. Evol.*, 19(9):1591–1601, 2002.

T. H. Jukes and C. R. Cantor. Evolution of protein molecules. In *Mammalian Protein Metabolism*, volume 3, pages 21–132. Munro, H. N., ed., academic press, new york edition, 1969.

D. G. Kendall. On the generalized "birth-and-death" process. *Ann. Math. Stat.*, 19:1–15, 1948.

M. Kimura. A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences. *J. Mol. Evol.*, 16(2):111–120, 1980.

P. J. Lockhart, M. A. Steel, M. D. Hendy, and D. Penny. Recovering evolutionary trees under a more realistic model of sequence evolution. *Mol. Biol. Evol.*, 11:605–612, 1994.

I. Mayrose, D. Graur, N. Ben-Tal, and T. Pupko. Comparison of site-specific rate-inference methods for protein sequences: Empirical bayesian methods are superior. *Mol. Biol. Evol.*, 21(9):1781–1791, 2004.

S. V. Muse. Evolutionary analyses of DNA sequences subject to constraints on secondary structure. *Gen.*, 139:1429–1439, 1995.

S. Nee, R. M. May, and P. H. Harvey. The reconstructed evolutionary process. *Philos. Trans. R. Soc. Lond. B*, 344:305–311, 1994.

N. J. Savill, D. C. Hoyle, and P. G. Higgs. RNA sequence evolution with secondary structure constraints: Comparison of substitution rate models using maximum likelihood methods. *Gen.*, 157:399–411, 2001.

M. Schöniger and A. von Haeseler. A stochastic model for the evolution of autocorrelated DNA sequences. *Mol. Phyl. Evol.*, 3:240–247, 1994.

K. Tamura. Estimation of the number of nucleotide substitutions when there are strong transition-transversion and g+c content biases. *Mol. Biol. Evol.*, 9:678–687, 1992.

K. Tamura and M Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Mol. Biol. Evol.*, 10(3):512–526, 1993.

R. Tarrio, F. Rodriguez-Trelles, and F. J. Ayala. Shared nucleotide composition biases among species and their impact on phylogenetic reconstructions of the Drosophilidae. *Mol. Biol. Evol.*, 18(8):1464–1473, 2001.

S. Tavare. Some probabilistic and statistical problems on the analysis of DNA sequences. *Lect. Math. Life Sc.*, 17:262–272, 1986.

E. R. M. Tillier. Maximum likelihood with multiparameter models of substitution. *J. Mol. Evol.*, 39:409–417, 1994.

E.R.M. Tillier and R.A. Collins. High apparent rate of simultaneous compensatory basepair substitutions in ribosomal RNA. *Gen.*, 148:1993–2002, 1998.

S. Whelan, P. Liò, and N. Goldman. Molecular phylogenetics: state-of-the art methods for looking into the past. *Tr. Gen.*, 17(5):262–272, 2001.

Z. Yang. Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: Approximate methods. *J. Mol. Evol.*, 39:306–314, 1994.

Z. Yang. Among-site rate variation and its impact on phylogenetic analyses. *Tr. Ecol. Evol.*, 11: 367–372, 1996.

Z. Yang and B. Rannala. Bayesian phylogenetic inference using dna sequences: a markov chain monte carlo method. *Mol. Biol. Evol.*, 14(7):717–724, 1997.

Z. Yang and D. Roberts. On the use of nucleic acid sequences to infer early branchings in the tree of life. *Mol. Biol. Evol.*, 12:451–458, 1995.

Z. Yang and T. Wang. Mixed model analysis of dna sequence evolution. *Biometrics*, 51:552–561, 1995.

Z. Yang, S. Kumar, and M. Nei. A new method of inference of ancestral nucleotide and amino acid sequences. *Gen.*, 141:1641–1650, 1995.

D. J. Zwickl and M. T. Holder. Model parameterization, prior distributions, and the general time-reversible model in bayesian phylogenetics. *Syst. Biol.*, 53:877–888, 2004.

Appendix A

Control File Examples

Most programs in the **PHASE** package have options set using a control file. The following examples, plus some additional ones, are provided with the software distribution, in the `example/control` directory.

Control file for *mcmcphase* (1)

```
# A simple example with mcmcphase.

# A standard DATAFILE block for analysing DNA sequences
{DATAFILE}
Data file = sequence-data/whales31-cytb.dna
Interleaved data file = no
# Ignore the class line at the end of this sequence file and
# group the nucleotides in a single class.
Heterogeneous data models = no
{\DATAFILE}


# A standard 4-state DNA model, across-site rate heterogeneity is accounted
# for with the discrete gamma model (6 rate categories)
{MODEL}
    Model = TN93
    Discrete gamma distribution of rates = yes
    Number of gamma categories  = 6
    Invariant sites             = no
{\MODEL}

# The standard TREE block for mcmcphase
{TREE}
Tree = Unrooted MCMC tree
Outgroup = Chevrotain
{\TREE}

# Tuning parameters
{PERTURBATION}
Tree, proposal priority = 20
Model, proposal priority = 1
{PERTURBATION_TREE}
Topology changes, proposal priority = 1
Branch lengths, proposal priority = 10
# One has to specify the prior on branch lenghts, a uniform(0,10) could have
# been used but see the scientific literature for possible issues.
Branch lengths, prior = exponential(10)
{\PERTURBATION_TREE}
{PERTURBATION_MODEL}
    Frequencies, proposal priority = 1
```

```
    Rate ratios, proposal priority = 1
    Gamma parameter, proposal priority = 1
{\PERTURBATION_MODEL}

{\PERTURBATION}

Random seed = 29072011

Burnin iterations = 100000
Sampling iterations = 100000
Sampling period = 100

Output file   = results/mcmcphase.whales31-TN93dG6.mcmc
Output format = phylip
```

Control file for *mcmcphase* (2)

```
# An example similar to Hudelot et al. (2003)

# A standard DATAFILE block for RNA sequences having a secondary structure.
{DATAFILE}
Data file = sequence-data/mammals69.rna
Interleaved data file = no
# Use the "automatic method" to analyse this dataset:
# unpaired nucleotides ('.' in the secondary structure) are
# handled by the MODEL1 of the MIXED model (see below).
# pairs (corresponding parenthesis in the secondary structure)
# are handled by the MODEL2 of the MIXED model (see below)
Heterogeneous data models = auto
{\DATAFILE}


#Set up a MIXED model with REV for loops and 7D for stems
{MODEL}
Model = MIXED
Number of models = 2
  {MODEL1}
  Model = REV
  Discrete gamma distribution of rates = yes
  Number of gamma categories = 6
  Invariant sites = no
  {\MODEL1}
  {MODEL2}
  Model = RNA7D
  Discrete gamma distribution of rates = yes
  Number of gamma categories = 6
  Invariant sites = no
{\MODEL2}
{\MODEL}

# Use a standard unrooted tree.
# The outgroup is compulsory but does not affect the results.
{TREE}
Tree = Unrooted MCMC tree
Outgroup = ORNANAMIT
{\TREE}


# Tuning parameters for the MCMC runs.
{PERTURBATION}

# Relative proposals probabilities between the tree and the substitution model
Tree, proposal priority = 8
```

```
Model, proposal priority = 1

{PERTURBATION_TREE}
# We use 10/40 for topology change vs branch length changes.
# It is not exactly equivalent to 1/4 because this is also given relative to the
# proposal priority for hyperparameters that are introduced with the
# the prior on branch lengths (Hyperpriors, proposal priority)
Topology changes, proposal priority = 10
Branch lengths, proposal priority = 40
Hyperpriors, proposal priority = 1

# We use a vague prior exp(lambda) on branch lengths rather than the default exp(10)
Branch lengths, prior = exponential(uniform(0,100))
# A lambda hyperparameter has been introduced. It needs a "proposal priority"
# but this is not used because it is the only hyperparameter
Branch lengths exponential hyperparameter, proposal priority = 1
{\PERTURBATION_TREE}

{PERTURBATION_MODEL}
# Relative probabilities for the proposals on the two models and the average
# substitution rate of MODEL2
Model 1, proposal priority = 10
Model 2, proposal priority = 10
Average rates, proposal priority = 1
{PERTURBATION_MODEL1}
    Frequencies, proposal priority = 2
    Rate ratios, proposal priority = 1
    Gamma parameter, proposal priority = 1
{\PERTURBATION_MODEL1}
{PERTURBATION_MODEL2}
    Frequencies, proposal priority = 2
    Rate ratios, proposal priority = 1
    Gamma parameter, proposal priority = 1
{\PERTURBATION_MODEL2}
{\PERTURBATION_MODEL}

{\PERTURBATION}

Random seed = 29072011

Burnin iterations = 750000
Sampling iterations = 1500000
Sampling period = 150


Output file   = results/mcmcphase.mammals69-REVdG6-7DdG6.mcmc
Output format = phylip
```

Control file for *mlphase*

```
# Phylogenetic tree reconstruction in the ML framework with mlphase.
# The dataset in this example is small and mlphase can be used.

# A standard DATAFILE block (see manual).
{DATAFILE}
Data file = sequence-data/hiv6.dna
Interleaved data file = yes
{\DATAFILE}

# A standard MODEL block (see manual), with an option to optimize or
# use user-defined substitution parameters.
{MODEL}
Model = HKY85
```

```
Discrete gamma distribution of rates = no
Invariant sites                      = yes

# If this value is 'no', you must specify a parameters file as the next option.
Optimize model parameters = yes

# This optional field must be used if you do not optimize the substitution
# parameters. It can also be used if you wish to start the ML optimizations
# from a specific set of parameter values
# Starting model parameters file = notusedhere.mod
{\MODEL}

# A TREE block
{TREE}
# You must specify an outgroup although it is used for representational
# purposes only, and it does not affect the results.
# This outgroup must be the name of a species in your datafile or the name
# of a monophyletic clade in your clusters file (see below).
Outgroup = outgroup

# See manual for the available heuristic/exhaustive search methods.
Search algorithm = Star decomposition

# Optional: specify a file that contains monophyletic clades.
# Tree topologies that do not match these constraints are not evaluated.
Clusters file = sequence-data/hiv6.cls
{\TREE}

Random seed = 29072011

Output file = results/mlphase.hiv6-HKY85I.out
```

Control file for *optimizer*

```
# Searching for the optimal model and branch lengths in the
# ML framework with optimizer.
# The likelihoods of a set of candidate tree topologies are compared.

# A standard DATAFILE block (see manual).
{DATAFILE}
Data file = sequence-data/hiv6.dna
Interleaved data file = yes
{\DATAFILE}

# A standard MODEL block (see manual), with an option to optimize or
# use user-defined substitution parameters.
{MODEL}
Model = TN93
Discrete gamma distribution of rates = no
Invariant sites = yes

# If this value is 'no', you must specify a parameters file as the next option.
Optimize model parameters = yes

# This optional field must be used if you do not optimize the substitution
# parameters. It can also be used if you wish to start the ML optimizations
# from a specific set of parameter values
# Starting model parameters file = notusedhere.mod
{\MODEL}

# A TREE block, specifying the set of candidate tree topologies.
{TREE}
# A file containing one or more candidate topologies.
```

```
Tree file = sequence-data/hiv6.tre

# You must specify an outgroup although it is used for representational
# purposes only, and it does not affect the results.
# This outgroup must be the name of a species in your datafile or the name
# of a monophyletic clade in your clusters file (see below).
Outgroup = outgroup

# Optional: specify a file that contains monophyletic clades.
# It only affects the presentation of the phylogeny, not the max-likelihood.
Clusters file = sequence-data/hiv6.cls
{\TREE}

Random seed = 29072011

Output file = results/optimizer.hiv6-TN93I
```

Control file for *likelihood*

```
# An example with the program likelihood.
# likelihood computes the likelihood of a phylogeny when the
# substitution parameters are known

# This program is useful because:
# 1)it can perform ancestral sequence reconstruction
# 2)it can compute the Bayesian Posterior Probabilities
# for the different categories of a mixture model (it can be used
# to compute site-specific substitution rate, Yang, 95)
# 3)it can also output the site-specific loglikelihoods used in
# some statistical tests.
# 4)it can also be used for the method proposed in Gowri-Shankar
# et al.(2006) to approximate the equilibrium frequencies in
# different rate categories

# A standard DATAFILE block for RNA sequences
{DATAFILE}
Data file = sequence-data/s70-TN93dG6-7DdG6.rna
Interleaved data file = no
Heterogeneous data models = auto
{\DATAFILE}

# A standard MODEL block for RNA sequences
{MODEL}
Model = MIXED
Number of models = 2
{MODEL1}
    Model = TN93
    Discrete gamma distribution of rates = yes
    Number of gamma categories = 6
    Invariant sites = no
{\MODEL1}
{MODEL2}
    Model = RNA7D
    Discrete gamma distribution of rates = yes
    Number of gamma categories = 6
    Invariant sites = no
{\MODEL2}

# User-specified model parameters are compulsory
# (see the manual/examples for more info on "model files").
Model parameters file = results-check/optimizer.s70-TN93dG6-7DdG6.mod
{\MODEL}
```

```
{TREE}
Tree file = results-check/optimizer.s70-TN93dG6-7DdG6.tre
{\TREE}

# The five following fields are optional

# Filename for results from the ancestral sequences marginal reconstruction.
# BEWARE1: 7-state models reconstruct MM pairs, R/Y models reconstruct 0/1
#          sequences.
# BEWARE2: When used with base-pair models (for instance), the program REPEATS
#          the BPP for the pair at each nucleotide position.
Ancestral sequences = results/likelihood.s70-TN93dG6-7DdG6.anc

# Filename for the results of the site-specific "BPP" rate category estimation.
# BEWARE: When used with base pair models (for instance), the program REPEATS
#         the MAP category estimate and the BPP at each nucleotide position
Site-specific substitution rates = results/likelihood.s70-TN93dG6-7DdG6.rat

# Filename to output the site-specific loglikelihood.
# BEWARE1: These site-specific likelihoods are not given in the order you might
#          expect. Invariant sites are at the end for technical reasons.
#          Moreover, sites are classed according to their data type
#         (loops/stems,..) with MIXED models.
# BEWARE2: With base-pair models (for instance), the site-specific loglikelihood
#          appears ONLY ONCE (it would probably not serve any purpose to repeat
#          it, and it could be error-prone).
Site-specific loglikelihood = results/likelihood.s70-TN93dG6-7DdG6.ssl

# Compute rate-specific composition.
# Composition for a given rate category is computed from the frequencies
# observed at each site weighted by the BPP of the site-specific rate category.
Rate vs. composition = yes

# You can limit the rate-specific composition estimation to a set of species:
# For a single species simply use its name;
# For all the species use "all" or simply omit the field (default behaviour);
# For a specific subset specify the name of a file (and use the name of one
# species per line in this file).
Selected species = all
```

Control file for *distphase*

```
# The program distphase generates a matrix of pairwise ML distances for a
# given set of sequences (base-pair models can be used with RNA stems).
# This matrix can be used with a tree reconstruction algorithm
# (UGPMA/NJ), not available in the PHASE package.

# The drawback of this program is that the parameters of the substitution
# model must be specified, and thus already need to have been found by
# another method.

# A standard DATAFILE block (see manual).
{DATAFILE}
Data file = sequence-data/s10-TN93dG10.dna
Interleaved data file = no
Heterogeneous data models = no
{\DATAFILE}

# A standard MODEL block (see manual) which also contains the name of the "model file"
{MODEL}
Model = TN93
Discrete gamma distribution of rates = yes
Number of gamma categories = 10
```

```
Invariant sites = no

# Model parameters are required with distphase, in the form of a "model file".
# A skeleton for this file, which can then be filled out with user-defined
# parameters, can be produced with the 'simulate' program; or the file can be
# generated with optimized parameters by using mlphase or optimizer.
Model parameters file = sequence-data/s10-TN93dG10.mod
{\MODEL}

Random seed = 29072011

Output file = results/distphase.s10-TN93dG10.out
#Output format = lower-triangular
```

Control file for *simulate* (1)

```
# Use this control file with simulate to evolve DNA sequences along
# the branches of a tree.

# The user has to specify the substitution model used.
# Here we use the TN93 model and rate heterogeneity across sites
# is modelled with the discrete gamma model (10 categories).
{MODEL}
    Model = TN93
    Discrete gamma distribution of rates = yes
    Number of gamma categories = 10
    Invariant sites = no
{\MODEL}

# The user also has to specify the parameters for this substitution model.
# Parameters are contained in a "model file"
Model parameters file = sequence-data/s10-TN93dG10.mod

# The format of this "model file" is not straightforward and depends on the
# substitution model. It is recommended to let PHASE generate a skeleton file
# for you first. Change the following field to "yes" to do so, simulate will
# not generate any sequences and will simply create/overwrite the model file.
# Replace the default parameters with your own values and do not forget to
# change the following field back to "no"
Retrieve the name of the model's parameters = no

# Change the random seed to generate different set of sequences
Random seed = 29072011

# Simulate on a fixed tree; see the manual (or the other example control
# file) for details on random tree generation.
{TREE}
Tree file = sequence-data/s10.tre
Generate tree = no
{\TREE}

# In this simple case we have to specify the lenght of the sequences, ie, the
# number of nucleotides, but things can become more complicated with complex
# substitution models.
# When using a base-pair or a codon model, you do not specify a number of
# nucleotides but a number of pairs or a triplets.
# When a MIXED model is used (ie more that one type of data), you have to
# specify a number of symbols for each substitution model.
Number of symbols from class 1 = 4000

# The file where the sequences are written
Output file = results/simulate.s10-TN93dG10.dna
```

```
# Optional parameters follow. They are used to produce a sequence file fully
# compatible with PHASE (ie, that can be used directly with other programs
# of the package) but they are not compulsory. Manual editing of these files
# might be easier so you should not bother with these unless it is necessary.

# The third token to use in the first line of your sequence file (can also be
# CODON or STRUCT; see manual). For simple nucleotide sequences generated
# with a DNA substitution model use 'DNA'.
Data file type = DNA

# If you use STRUCT in the previous case, you have to tell simulate how the
# structure line should be produced. For instance:
# Structure for the elements of class 1 = .
# There should be more than one field when a MIXED model is used.
# If class x is a nucleotide or amino-acid model use:
#  Structure for the elements of class x = "."
# If class x is a doublet model use:
#  Structure for the elements of class x =  "()"
# If class x is with a codon model use:
#  Structure for the elements of class x = "123"

# The last optional parameter is the total number of nucleotides in the
# alignment (the second field in the first line).
Number of nucleotides from class 1 = 4000
```

Control file for *simulate* (2)

```
# A control file to be used with simulate, with some advanced settings:
#  Generate a RNA sequence with secondary structure information.
#  Generate a random tree.
# Please see the other example control file for the basic settings.

# Set up a MIXED model
{MODEL}
Model = MIXED
Number of models = 2
{MODEL1}
    Model = TN93
    Discrete gamma distribution of rates = yes
    Number of gamma categories = 6
    Invariant sites = no
{\MODEL1}
{MODEL2}
    Model = RNA7D
    Discrete gamma distribution of rates = yes
    Number of gamma categories = 6
    Invariant sites = no
{\MODEL2}
{\MODEL}

# Change the following field to yes to generate the skeleton file
# for this MIXED model.
Retrieve the name of the model's parameters = no

Model parameters file = sequence-data/s70-TN93dG6-7DdG6.mod

Random seed = 29072011

# We generate a random tree (but the number of species is specified by the user)
# see Aldous, 1996 and Yang, 1997 for information on these topics.
{TREE}
# The generated phylogeny (tree topology + branch lengths)
# is stored in the following file
```

```
Generate tree = yes
Tree file = results/simulate.s70.tre

# Total number of species in the generated tree
# (including the outgroup if there is one).
Number of species = 70
# Optional parameter to create an outgroup cluster.
Number of species in the outgroup = 7

# The process used to generate the tree; options are:
# Yule process, Birth-death process, Uniform process, Beta-splitting process.
Generating model = Birth-death process

# The Beta-splitting process requires the specification of beta in [-2;+inf]
# Beta parameter=-2 equivalent to comb
# Beta parameter=-1.5 equivalent to the uniform process
# Beta parameter=0 equivalent to a Yule or birth-death process
# Beta parameter=+inf or beta=inf or beta=+infinity is a symmetric binary trie

# Yule process and Birth-death process incorporate automatically
# a probability distribution for branch lengths which cannot be changed.
# When using the uniform and/or beta-splitting process one has to specify
# a prior distribution for the branch lengths.
# Choose among Uniform, Exponential, Pure-birth process, Birth-death process
# If using Uniform then you also have to specify two extra parameters:
# "Branch prior, lower bound" and "Branch prior, upper bound"
# If using Exponential then you have to specify
# "Branch prior, exponential parameter" (=1/mean)

# When using the Yule or the Birth-death process as a generating model or when
# using the Pure-birth process or the Birth-death process for the prior on
# branch lengths, you have to specify a "Birth rate", and a "Death rate" if
# appropriate NB: distance from root to tips is supposed to be 1.0 for this
# procedure. See Yang, Rannala 1997 for more information.
Birth rate = 10
Death rate = 5

# When using a Yule or birth-death process, the height of the tree is
# rescaled from 1.0 to the user's choice.
Tree height = .8

# When using the Yule or the Birth-Death process, you can specify a species
# sampling (see Yang,Rannala (1997)) (it is 1.0 by default).
# Accounting for the fact that your dataset does not contain all existing
# species affects the prior on branch lenghts.
Species sampling = .05
# You can choose to use a different value for the outgroup (if you previously
# used the field "Number of species in the outgroup").
Outgroup sampling = .001
{\TREE}


# The number of symbols to generate (a symbol can be a pair or a complete codon
# depending on your model). Since a MIXED model is used here, you have to
# specify a length for each model/class/type of data.
Number of symbols from class 1 = 1500
# For the stems, there are 2000 nucleotides but only 1000 symbols.
Number of symbols from class 2 = 1000

# The output of the program
# IMPORTANT 1: PHASE is generating the nucleotides for the first class, then the
#              nucleotides for the second class before concatenating the
#              results. Heterogeneous data types are not intertwined.
#              In this case it means that the first 1500 nucleotides are not
#              paired, the 2000 remaining nucleotides are generated with the
#              RNA model.
```

```
# IMPORTANT 2: Note that the two nucleotides of a pair are at position i/i+1
#              in the generated sequences
# IMPORTANT 3: 7-state RNA models are using AA for the mismatch state,
#              the THREESTATE model is using C for the Y state,
#              the TWOSTATE model is using A and C for its two states.
Output file = results/simulate.s70-TN93dG6-7DdG6.rna


# Optional parameters follow. They are used to produce a sequence file fully
# compatible with other programs in the PHASE package and they can be left out.

# The data file type to use at the first line of the sequence file
Data file type = STRUCT

# To produce a correct structure line in this case, PHASE needs the following:
Structure for the elements of class 1 = .
Structure for the elements of class 2 = ()
# with a codon model we would have used "123" (without quotes)

# This is the number of nucleotides for each data type which is used to
# compute the total length of the alignment properly.
Number of nucleotides from class 1 = 1500
Number of nucleotides from class 2 = 2000
```