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

Version 2.0, August 3, 2006

Copyright 2002-2006 by the University of Manchester.

PHASE is distributed under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Vivek Gowri-Shankar and Howsun Jow

*bug reports, suggestions and help: vivek.gowri-shankar@cs.man.ac.uk

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 alignement 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,
- some ML search algorithms for inferring the optimal tree with optional topological constraints,
- 6, 7 and 16 state RNA models, standard 4 state DNA models, RY-model, AGY model, empirical amino-acid models and one codon 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.

What's new in *PHASE-2.0*?

- it is faster, in some cases **much** faster,
- more substitution models (*e.g.*, amino-acid, codon, RY-model, AGY-model), and some RNA models used in other software,
- improved pairing mask format to deal properly with pseudo-knots,
- improved checking of user inputs,
- in the ML framework specifically:
  - computation of pairwise ML distances,
  - the star-decomposition heuristic method for topology search,
  - search algorithms were adapted to account for user-specified topological constraints and reduce the search space,
- in the Bayesian framework specifically:
  - MCMC proposals were implemented for ultrametric trees (*i.e.*, molecular clock assumption),
  - different choices of priors on model parameters and branch lengths are available,
  - some evolutionary models to account for heterogeneity in space and/or in time (experimental...),
  - a better consensus program and a crude graphical user interface to view the results,
  - better control of the MCMC proposals,
  - MCMC runs recovery in case of computer crash.

Contents

Why is *PHASE* different from other phylogenetic programs? . . . 1
Acknowledgments . . . 2
What's new in *PHASE-2.0*? . . . 2

Introduction **5**
Aquiring and installing the software . . . 5
Windows installation . . . 5
*nix and Mac OS X installation . . . 5
Description of programs in the *PHASE* package . . . 6
distphase, *optimizer* and *mlphase* . . . 6
mcmcphase and *mcmcsummarize* . . . 6
likelihood . . . 6
simulate . . . 7
Running the programs . . . 7

1 Using programs in the *PHASE* package **8**
1.1 Inputs/outputs in *PHASE* . . . 8
1.1.1 Data file format . . . 8
1.1.2 Control file format . . . 11
1.1.3 Tree file format . . . 12
1.1.4 Substitution model parameters file format . . . 12
1.1.5 Parameters displayed on the screen and output of each program . . . 12
1.1.6 Clusters file format . . . 13
1.2 Producing a consensus secondary structure for *PHASE* . . . 13
1.2.1 *extractfromDCSE* and *secondarystructconsensus* . . . 13
1.2.2 Useful links . . . 14
1.3 Control files . . . 15
1.3.1 Structure of the control files . . . 15
1.3.2 Datafile block . . . 16
1.3.3 Model block . . . 16
1.3.4 Tree block . . . 19
1.4 Using the programs in the *PHASE* package . . . 19
1.4.1 *mcmcphase* . . . 19
1.4.2 *mlphase* . . . 27
1.4.3 *optimizer* . . . 28

1.4.4 *distphase* . . . 29
1.4.5 *likelihood* . . . 30
1.4.6 *simulate* . . . 31
1.4.7 *analyzer* . . . 34
1.4.8 Pitfalls of Markov chain Monte-Carlo techniques . . . 34

2 Substitution models available **36**
2.1 Nucleotide substitution models . . . 36
2.1.1 A Markov model of substitution . . . 36
2.1.2 Transition matrices . . . 37
2.1.3 Nucleotide substitution models implemented in *PHASE* . . . 38
2.1.4 The TWOSTATE (RY) and the THREESTATE (AGY) models . . . 39
2.2 Paired-site substitution models . . . 40
2.2.1 RNA secondary structure . . . 40
2.2.2 Theory of compensatory substitutions . . . 41
2.2.3 Base-paired substitution models implemented in *PHASE* . . . 42
2.2.4 6-state substitution models . . . 42
2.2.5 7-state substitution models . . . 44
2.2.6 16-state substitution models . . . 46
2.3 Codon and amino-acid substitution models . . . 47
2.3.1 Codon model . . . 47
2.3.2 Empirical amino-acid models . . . 47
2.4 Refinements to substitution models . . . 47
2.4.1 The **MIXED** model . . . 47
2.4.2 The **HETEROGENEOUS** model . . . 48
2.4.3 Variation of frequencies across sites . . . 48

Bibliography **58**

Introduction

Aquiring and installing the software

PHASE can be downloaded from http://www.bioinf.man.ac.uk/resources/phase; it is currently available for Windows, Unix/Linux and Mac OS X platforms.

Windows installation

Download the archive *phase-2.0-MSWin-exec.zip* and decompress it into the directory of your choice, for instance **c:\PHASE**. *PHASE* does not require any other installation procedure and you can therefore test the software straight away with the provided example files.

*nix and Mac OS X installation

The software is provided as source code and you have to compile it. Should the process fail, a static built version (linux only) can be found in the archive *phase-2.0-linux-i586-exec.tgz*. Non-linux users, please read some system specific instructions in the *README* file. We will try to provide help in case of trouble.

We assume here that you have a recent GNU C++ compiler *g++* on your platform. *PHASE* will not compile with *gcc v2.96* and older. You can check the gcc version installed on your system by typing "**g++ -v**". *PHASE* uses the *BLAS* and *LAPACK* library routines. By default, generic cut-down versions will be built during the compilation process. The *g77* compiler and the *libg2c* library are required for successful completion of this step but they might be already be present on your system. However, some recent systems are equipped with optimised versions of BLAS and LAPACK (and are missing *g77* and *libg2c*). In such a case, you have to modify the *makefile*, see the *README* file for further instructions.

The *README* file also contains further informations for MAC OS X users.

To compile the program yourself:

- decompress and extract the archive into the directory of your choice
  tar -xvvzf phase-2.0-src.tgz
- enter the newly created *phase-2.0-src* directory
  cd phase-2.0-src
- modify the *makefile* file following the instructions in the *README* file.
- compile
  make (NB: or **gmake** if GNU make is not the default on your system)

There is no "make install", sorry for the inconvenience. System administrators can copy the programs of the package to the directory of their choice (see the *README* file for a list). Users should copy the example files into their own directory.

Description of programs in the *PHASE* package

The *PHASE* (PHylogenetics And Sequence Evolution) package consists of two main programs, *mlphase* and *mcmcphase*.

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

There are six other smaller programs in the package:

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

Below we summarize the behaviour of these programs. Please refer to the first chapter in order to learn how to use them.

distphase, *optimizer* and *mlphase*

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. 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).

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). The *phase_viewer* program is not a essential component but it can help to vizualize some outputs of the *mcmcsummarize* program.

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 & branch lengths) and substitution model (type & parameters). It has recently been extended to generate "random" topologies (birth-death process, sampling effect, etc.).

Running the programs

Programs in the *PHASE* package are run through the command line under both Unix-like systems and MS-windows systems. For Windows operating systems, you have to open a 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 *MS-Windows* version. Once the command window is opened, you have to move to the directory where you extracted the software. At the shell prompt, you can type, for example, **cd c:\Phase**. You can then run any program of the *PHASE* package.

Run the programs by typing their name followed by the arguments they require. In most cases, *PHASE*'s programs take one argument which is the name of a control file (see section 1.3). You can type, for instance:

mcmcphase control\examples-mcmc\hiv6-HKY85I.mcmc [1] or,
./optimizer control/examples-ml/hiv6-TN93I.opt

After installation, if the examples are all present, these commands should work.

[1] Please note that the use of the '\' or '/' characters is dependant on your operating system. On Unix systems you might have to type "**./**" before the program name.

1 - Using programs in the *PHASE* package

1.1 Inputs/outputs in *PHASE*

1.1.1 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*[1] data file format but has a few minor modifications.

A data file is divided into four sections whee two of them are optional in some cases. Comments can be included by beginning the commented lines with a hash (#) symbol. The entire commented line is ignored by the program. Having a look at the example data files in the package (**.dna* and **.rna* in the *sequence-data* directory) will make the following explanations easier to understand.

File content

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 independant,
   - **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 mentionned 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 independant 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. This is in practice completely equivalent to:
5 100 PROT
because the parser prepares the sequences for the programs without analyzing their content.
With,
10 105 CODON
PHASE will expect 10 sequences of $105/3 = 35$ codons.

[1] http://evolution.genetics.washington.edu/phylip.html

Structure mask

The second section of the data file is the structure mask. Your study requires a structure mask whenever sites cannot be all considered independantly (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 directly the entire structure line without label between the first line described previously and your first sequence. If you are using the interleaved format, you also have the possibility to interleave the structure with the sequences (see the interleaved format section below). Choose at your convenience.

Simply put, the structure mask consists of dots (unpaired nucleotides), parenthesis (pairs) and "**123**" (codons). Unpaired sites (nucleotides or amino-acids) are indicated with a dot "**.**" or a hyphen "**-**". Corresponding brackets "**()**"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 16 STRUCT
       (((.(.))))
       123123
Mouse  ACCAGAUGGU
       ATGGTA
Rat    CCCAGUUGGG
       ATGGTU
```

The "Mouse" sequence **ACCAGAUGGU ATG GTA** with a structure mask **(((.(.)))) 123 123** indicates that the sequence is made of the base-pairs **AU-CG-CG-GU**, unpaired sites **A-A**, and two codons **ATG-GTA**. The structure mask is shared among sequences.

The previous version of *PHASE* could not handle pseudoknots properly since they cannot be represented with a simple dot-parenthesis notation, we consequently added the brackets '[', ']','**{**', '**}**', '<', '>' and letters '**A**', '**a**', '**B**', '**b**', '**C**',... so that you can intertwine stems properly, *e.g.*, **GGGACUCCGU** with structure **((.[[.))]]** or equivalently **((.AA.))aa** will correctly produce the pairs **GC-GC-AU-CG**. 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.

Building a pairing mask for RNA genes is not a straightforward task. A section of the manual deals with the issue of producing a consensus secondary structure (section 1.2).

Please notice 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 allowed to use different substitution models for different pairs (or codons), *e.g.*, if they are not from the same genes. This is the purpose of the class section (see 1.1.1 below).

Molecular sequences

The third section of the data file contains the molecular sequences. The IUBMB single-letter code[2] 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 will be described later on. The *non-interleaved* format consists of an identifying label for each sequence followed by the whole sequence. An example is:

[2] http://www.chem.qmul.ac.uk/iubmb/misc/naseq.html#tab1

```
 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. Unlike the *PHYLIP*[3] format there is no real limit on the number of letters in species labels (50 characters or so) 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, three codon positions, ...). You can safely skip this section if you do not plan to concatenate genes or if you use RNA stems without the loops. This section is relevant when the **MIXED** substitution model (see section 2.4.1) is used and you do not need it 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 model used in the **MIXED** substitution model will be standard 4-state models and the third model will be the RY-model (**TWOSTATE**).

When concatenating the LSU and the 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 compulsary 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 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)

[3] http://evolution.genetics.washington.edu/phylip.html

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 1.3.2) 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 respectively be a DNA and a RNA substitution model if your structure mask contains unpaired nucleotides and pairs).

Interleaved format

An alternate 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 some spaces but a new line is interpreted as "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 are permitted not to repeat them in each subsequent block. Whether labels are repeated or not, species must be provided in the same order in each block.

As mentionned previously, you are allowed to interleave the structure mask and the class line with your sequences (refer to the **DATAFILE** block (section 1.3.2) so that the software can read your sequences correctly). This is not compulsory and, by default, *PHASE* will still read the structure line before the sequences and the class line after the sequences as usual. 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 (*i.e.*, you cannot use '1' to '9' and '(',')','[', ... in these labels).

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

1.1.2 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 model(s) and others specific parameters). Control files are the key to using the software and two sections are devoted to them. Section 1.3 describes the structure of this file and describes common features for many programs in the package. Section 1.4 presents the specific parameters for each program.

1.1.3 Tree file format

PHASE can output trees into a file and sometimes the user has to provide a file which contains one or more trees. A tree file is simply a file with one ore more phylogenies written in a computer readable format.

This format is basically the Newick format[4] 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 really not adequate (*e.g.*, with *mcmcphase*) and will be refused. For most algorithms, the root position is irrelevant and programs that can only handle unrooted tree will usually "uproot" a rooted tree using an user-specified "Outgroup".

1.1.4 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 dependant 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. You might need to have a look at the transition matrices section (2.1.2) to understand the concept of "rate ratios" (or exchangeabilities) in substitution models. *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 section 2.1.3 for DNA models and in section 2.2.3 for RNA models.

Producing a model parameters file

Model parameters files and control files share the same structural elements. Some examples can be found in the *input-data* directory (**.mod*). 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. See section 1.4.6 for details.

1.1.5 Parameters displayed on the screen and output of each program

Each program in the package will output information on the screen, and one or more files to store the results permanently. The outputs will be reviewed individually for each program in section 1.4.

The content displayed on the screen is usually quite easy to understand, but you might be a bit confused by the parameters of substitution models. *PHASE* outputs on the screen two kind of matrices:

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

These matrices are described in the transition matrices section (2.1.2). 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.

[4] http://evolution.genetics.washington.edu/phylip/newicktree.html

1.1.6 Clusters file format

When it is relevant, the user is often allowed to specify some monophyletic clusters to reduce the number of possible topologies. Using a cluster file to specify obvious monophyletic groups will greatly reduce the size of the discrete topology 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 (cf. appropriate sections).

1.2 Producing a consensus secondary structure for *PHASE*

PHASE is designed to use the secondary structure of RNA genes but cannot infer it. To use the software and the specific doublet substitution models implemented, the hardest task for the user is probably to provide a consensus secondary structure on top of the standard alignment. Two small programs have been implemented to help you in that task and are distributed with this release. We also thought it might be useful to provide links to programs implemented by the users of the previous *PHASE* version along with links to related databases and other RNA software. You are strongly advised to use *analyzer* to check whether your secondary structure looks correct and to correct obvious errors before using other programs from the package.

1.2.1 *extractfromDCSE* and *secondarystructconsensus*

Two crude programs were implemented to help in the production of a "structure line" (section 1.1.1). The first one, *extractfromDCSE*, is designed specifically to work with sequences downloaded from the European Ribosomal RNA database[5] in the DCSE format. The second one, *secondarystructconsensus*, was written to produce a consensus structure from the aligned secondary structures of a set of species. Outputs from *extractfromDCSE* can be fed to *secondarystructconsensus* with little work in order to produce a consensus secondary structure for the given set of sequences.

extractfromDCSE

extractfromDCSE reads a **DCSE** sequence alignment and produce an alignment file where the structures and the nucleotides are dissociated. In effect, it will transform the **DCSE** structure format into the dot-parenthesis notation used by *PHASE* . Pseudoknots are handled using different symbols (see section 1.1.1 on the use of a structure mask for more information). For each species, the program is producing a structure sequence and a nucleotide sequence. The length of the alignment remains the same. Unless you prepared your final alignment directly in the DCSE format, further editing will be necessary to remove ambiguous regions, gaps, etc.. Maintain your consensus secondary structure properly during this refinement step.

It might be possible for you to find a better tool than this one to parse **DCSE** sequences and produce the intended result (see the links (section 1.2.2) for alternatives). Indeed, *extractfromDCSE* will often choke while reading DCSE inputs since the format uses a single "helix numbering line" for multiple sequences (see the DCSE sequence format[6]). Quite often, a helix found in a RNA sequence cannot be attributed easily to a name written in the helix line since positions are not matching perfectly. You can

[5] http://www.psb.ugent.be/rRNA/
[6] http://rrna.uia.ac.be/dcse/help/aliformat.html

easily resolve conflicts by hand by duplicating the helix numbering line in your DCSE file above and below the problematic sequence(s) and then edit the helix line below to move around the names of the problematic stems under their real position for the considered set of sequences.

To use *extractFromDCSE*, type at the command-line:
extractfromDCSE dcse-alignment output-file
where *dcse-alignment* is the file with the sequences in the **DCSE** format.

secondarystructconsensus

secondarystructconsensus expects an input file which contains a list of structure sequence (one per line) in the *PHASE* dot-parenthesis notation. Pseudoknots are accepted as long as parenthesis are replaced with appropriate symbol, *e.g.*, '['-']'. See section 1.1.1 on the use of structure mask for more information. The output of *extractfromDCSE* can be used in which case you must remove all but the structures from the file and input the length of the alignement at the top.

To use *secondarystructconsensus*, type at the command-line:
secondarystructconsensus structure-file cut-off-1 cut-off-2
where *structure-file* is a valid structural alignement file as described above. Pairs cannot be added to the consensus secondary structure if too many sequences do not have a pair at a considered doublet position. The *cut-off-2*, between 0.0 and 0.5 is the maximum allowed proportion of sequences which do not participate in the pairing. In some cases, some sites in a sequence can be involved in a different pairing interaction than the one suggested by the consensus secondary structure. The *cut-off-1*, between 0.0 and *cut-off-2*, is the maximum allowed proportion of "contradiction" at a doublet position. If you do not know what to use there, try *cut-off-1*=0.1 and *cut-off-2*=0.4.

1.2.2 Useful links

Numerous links to RNA-related material can be found on the BRAliBase page:
http://www.binf.ku.dk/users/pgardner/bralibase/ and on the RNA World Website:
http://www.imb-jena.de/RNA.html

Here is a set of links that can help you adding a consensus secondary structure in your alignment (or obtaining one ready to use). This list has no pretense to be exhaustive and is biased towards works related to *PHASE* . If you think your web page(s), work or program(s) should be listed here or if you feel something is missing then please tell us.

Some RNA databases providing the pairing information

- OGRe: Organellar Genome Retrieval: http://drake.physics.mcmaster.ca/ogre/
  Aligned mitochondrial rRNAs and tRNAs are available for various metazoan organisms. The consensus secondary structure is already in *PHASE* format.
- The European ribosomal RNA database: http://rrna.uia.ac.be/dcse/help/aliformat.html
  A ribosomal RNA database (LSU and SSU). Sequence can be downloaded in the DCSE alignment format.
- Comparative RNA web site: http://www.rna.icmb.utexas.edu
  Sequence/structure data and comparative structure models
- Rfam: an RNA family database: http://www.sanger.ac.uk/Software/Rfam/
- The tmRNA website: http://www.indiana.edu/*sim*tmrna/
- tmRDB: http://psyche.uthct.edu/dbs/tmRDB/tmRDB.html

Automatic methods

- CARNAC: www.lifl.fr/*sim*perrique/rna/
  Produces individual secondary structures for a set of unaligned homologous sequences.

- MARNA: A Server for Multiple Alignment of RNAs: http://www.bio.inf.uni-jena.de/Software/MARNA/
  Multiple alignment of RNAs considering the provided primary and secondary structures simultaneously.

- RNAalifold in the Vienna RNA Package and Pfold :
  http://www.tbi.univie.ac.at/*sim*ivo/RNA/, http://www.daimi.au.dk/*sim*compbio/rnafold/
  Adds a consensus secondary structure to an existing RNA alignment.

By-eye alignment with the secondary structure as a guide

- Xstem and Ystem: [add link]
  Multiple alignment editor, reads the DCSE format, lets you add your own sequences and provide outputs for the doublet models in *MrBayes*[7] and *PHASE* .

- Bioedit: http://www.mbio.ncsu.edu/BioEdit/bioedit.html
  Another multiple alignment editor that can use secondary structure information.

- jRNA: http://hymenoptera.tamu.edu/rna/
  How to do a "clean" structural "by-eye" alignment with some Perl scripts to produce *POY*, *MrBayes*[8] and *PHASE* input files.

1.3 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 example with the package (in the *control* directory). The easiest and safest way to use *PHASE* is most probably to copy one of these examples and to adapt it to your need.

1.3.1 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 should be issued if the user mistypes a parameter. In any cases, please check your control files against the provided examples to ensure the software will be doing what you want.

A block is a container. It contains control lines but can also contain 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 colored 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*.

[7] http://morphbank.ebc.uu.se/mrbayes/
[8] http://morphbank.ebc.uu.se/mrbayes/

1.3.2 Datafile block

Almost all programs in the *PHASE* package require a **DATAFILE** block to parse analyzed sequences. As stated previously, 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 necessary information which is not included in the datafile itself (see section 1.1.1 for the format of this file); it contains the following control lines:

- *Data file*: the location of the molecular sequences 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 (see section 1.1.1).
  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, *e.g.*, as was shown during the description of the datafile, you can have four independant models when concatenating LSU and SSU genes. 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

1.3.3 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 (see section 2.1 for background information on substitution models of nucleotide evolution). We describe here the main components of this model block. Be aware that most programs in the package can use (and sometimes require) specific fields in 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. See sections 2.1.3 and 2.2.3 for a review of these models. You can now use a custom 3-state model (section 2.1.4) for AGY-analysis, a 2-state model (section 2.1.4) for, *e.g.*, 0/1 or R-Y analysis (please **do not recode** your data in the latter cases, the software can handle standard DNA sequences as is, *i.e.*, with 4 nucleotides).

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.
  Model = REV
  Nucleotide substitution models implemented include **JC69**, **K80**, **HKY85**, **TN93**, **REV**, **T92**. Base-paired substitution models implemented include **RNA6A**, **RNA6B**, **RNA7A**, **RNA7D**, **RNA7F**, **RNA16A**, **RNA16B**, **RNA16C**, **RNA16D**, **RNA16E**, **RNA16F**, **RNA16I**, **RNA16J**, **RNA16K**. You can also use **TWOSTATE**, the 2-state model, or **THREESTATE** the custom 3-state model.

- *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

Codon and amino-acid substitution models

Codon and amino-acid substitution models (section 2.3) for protein-coding genes were recently added. You can now use the simple codon model **YNH98**, and you can also use empirical amino-acid substitution matrices with the generic **AAEMPIRICAL**. These models will use the same fields as the substitution models described previously plus some specific fields.

Please note that **YNH98** has approximately 60 states and is really slow. Though it is allowed to use a gamma ASRV model with codon models, you probably should not do it and consider using models that let the dN/dS ratio vary instead (work in progress, not implemented in this version).

For the codon substitution model:

- *Genetic code file*: the genetic code to use. This will also be used to define the number of states of your model.
  Genetic code file = input-data/GeneticCode/Universal.txt

For the empirical amino-acid substitution model:

- *Genetic code file*: the empirical amino-acid model can read a nucleotide datafile if you provide it with a genetic code. This field is useless with amino-acid sequences.
  Genetic code file = input-data/GeneticCode/MammalianMitochondrial.txt

- *Empirical Values*: a file that contains empirical exchangeabilities and frequencies. We (almost) kept the *PAML*[9] format. Some files are provided as examples and you can easily import/create others.
  Empirical Values = input-data/EmpiricalModels/wag.dat

- *+F*: a yes/no flag to determine whether frequencies are free parameters of the model. Please note that it does not mean that frequencies will be equal to the empirical values found in your dataset, frequencies parameters will be evaluated along with other parameters during the inference process. To emulate the traditional meaning of +F, you also need to turn off the frequency proposals during MCMC runs or prevent the optimization of substitution parameters during ML optimizations (frequencies should be properly initialized from the sequences in both cases). By default, empirical models are not +F and frequencies found in the files containing the empirical parameters are used.
  +F = yes

Mixed model for combined analyzes of heterogeneous data

Several models are sometimes required when studying heterogeneous sequences. The **MIXED** model (see section 2.4.1) allows these models to work concurrently.

- *Model*: this field contains the name of the model which is **MIXED**.
  Model = MIXED

[9] http://abacus.gene.ucl.ac.uk/software/paml.html

- *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 behavior of the automatic assignment described in the **DATAFILE** block section 1.3.2.
  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 = 3
{MODEL1}
Model = REV
Invariant sites = yes
{\MODEL1}
{MODEL2}
Model = RNA7A
Discrete gamma distribution of rates = yes
Number of gamma categories = 5
{\MODEL2}
{MODEL3}
Model = RNA7D
{\MODEL3}
{\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 (see section 2.4.2) allows these models to work concurrently. Please note that we are still in the process of investigating and testing these methods.

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

- *Model*: this field contains the name of the model which is **HETEROGENEOUS**.
  Model = HETEROGENEOUS

- *Number of models*: the number of models used concurrently. See Foster (2004) for statistical testing techniques. We have not implemented the reversible jump MCMC yet 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
Invariant sites = no
{\BASEMODEL}
{\MODEL}

You **must not** use an invariant model (+I) 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).

1.3.4 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.

1.4 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* dependant on the presence and/or value of others are in *orange*.

1.4.1 *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 a MCMC run is aborted before its end then you can restore it using the same command. If *PHASE* detect a .rsb and a .rsc file, 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.

Please, check the possible issues and pitfalls with MCMC techniques (section 1.4.8). Known issues with Bayesian methods and specifically *PHASE* are listed there.

The *mcmcphase* program saves the results of an inference 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-dependant, 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 burnin 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*

In the control file of *mcmcphase* one can/must have:

- a ***DATAFILE** block*: see the data file block section (1.3.2).
- a ***MODEL** block*: see the model block section (1.3.3). Some fields specific to *mcmcphase* are added to this **MODEL** block:
  - *Starting model parameters file*: to reduce the necessary "burnin" 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 section 1.1.4.
    Starting model parameters file = input-data/s70-TN93dG20-7DdG20.mod
- a ***TREE** block*: see the tree block section for *mcmcphase* below (1.4.1).
- a ***PERTURBATION** block*: control 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.
  Random seed = 3
- *Burnin iterations*: the number of "burnin" cycles (*i.e.*, cycles before the beginning of the sampling). During the "burnin", only likelihood values are stored. It is good practice to check that the likelihood reached a plateau before the end of the "burnin" (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 proposals parameters during the "burnin" and aims at reasonable acceptance rates to improve mixing properties during sampling. It is consequently slightly better to use a longer "burnin" rather than discarding samples once the run is finished.
  Burnin iterations = 150000
- *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 iterations = 600000

- *Sampling period*: the number of cycles between extraction of two consecutive samples.
  Sampling period = 100

- *Output file*: the basename for all the output files (*basename-best.tre*, *basename-best.mod*, *basename.mp*, *basename.smp*, ...).
  Output file = results/hiv6-dna-mcmc

- *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).
  Output format = phylip

TREE block for *mcmcphase*

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

- *Tree*: this is the main field, four possible trees can be used:
  - "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 of the "Rooted MCMC tree with molecular clock" but for heterogeneous model. Use with the **HETEROGENEOUS** model.

  Tree = Unrooted MCMC tree

- *Clusters file*: this parameter is optional and is used to constrain the topology space, if present, the value expected for this field is the name of a clusters file (see section 1.1.6). Specifying the obvious clades (*i.e.*, those with Bayesian Posterior Probability 100%) can seriously improve your mixing when a large number of species is used. Consider the possibility.
  Clusters file = input-data/hiv6-dna/hiv6.clusters

- *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 readability purpose. 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.
  Outgroup = Thermotogata_maritima

- *Starting tree file*: one can choose to initialize the chain randomly or with a user-defined tree/topology. We do not want to encourage the use of an initial topology in a standard analysis. Starting several MCMC chains (*e.g.*, 4 chains?) from different random initial state to assess convergence is compulsory. 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.
  Starting tree file = input-data/s10-simul/s10-rooted.tre

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 purpose. 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 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 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:
  Tree, proposal priority = 10
  Model, proposal priority = 1

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.
  Topology changes, proposal priority = 10

- *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, proposal priority = 50

- *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. **exponential(10)** or **uniform(0,5)** can be acceptable. You can add a level of hierarchy and introduce a hyperparameter. In such a case, the field *Hyperpriors, proposal priority* will become compulsory. More on priors follows.
  Branch lengths, prior = exponential(10)

- *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 set of branch lengths that matches this requirement have uniform probability.
  Tree height, prior = exponential(uniform(0,10))

- *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.
  Hyperpriors, proposal priority = 1

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

PERTURBATION_MODEL block

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

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 usually 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, proposal priority = 3

- *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 prior by default, equivalent to dirichlet(1)). More on priors follows.
  Frequencies, prior = dirichlet(2)

- *Frequencies, proposal minimum acceptance rate* and *Frequencies, proposal maximum acceptance rate*: during the "burnin", *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, proposal minimum acceptance rate = 0.0
  Frequencies, proposal maximum acceptance rate = 1.0

- *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 "burnin" 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.
  Frequencies, initial Dirichlet tuning parameter = 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 be doing that unless you provide initial parameters with a *".mod"* file.
  Rate ratios, proposal priority = 1

- *Rate ratios, prior*: prior for the rate ratios, see section on priors. The default prior is **uniform(**.0005**,** 2000.0**)**. *PHASE* does not yet 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, prior = uniform(.0005, 2000.0)**

- *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, proposal minimum acceptance rate = 0.3
  Rate ratios, proposal maximum acceptance rate = 0.6

- *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.
  Rate ratios, initial step = 0.5

Similarly, you can/must use the following parameters 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**.

and,

- *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 when MIXED models are used

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 complete distinct *PERTURBATION_MODELi blocks* as previously described and put them in the **PERTURBATION_MODEL** block. Parameters which are specific to the **MIXED** model will appear at the top-level of the **PERTURBATION_MODEL** block. These parameters are:

- *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. **Model 1, proposal priority = 8**
  Model 2, proposal priority = 24
- *Average rates, proposal priority*: specify a proposal priority to modify the average substitution rate of each model (but the first one since its average substitution rate is fixed to 1.0). This priority is relative to the previous ones.
  Average rates, proposal priority = 1
- *Average rates, prior*: uniform, exponential, ... Default value is **uniform(0.005,200.0)**. *PHASE* does not yet 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, proposal minimum acceptance rate = 0.15
  Average rates, proposal maximum acceptance rate = 0.25
- *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**.
  Average rates, initial step = 0.14

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}

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 as been described above since its content is strictly the same as a **PERTURBATION_MODEL** block for standard or **MIXED** substituttion 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).
  Model, proposal priority = 30
- *the **ANCESTRAL_FREQUENCIES** block*: **HETEROGENEOUS** models are using 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 (these blocks) was described above (fill it with control-lines related to frequencies).

  Here is a complex example for an 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}

Priors

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

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

$$f(\pi|\alpha) = \frac{\Gamma(\sum_{i=1}^{n} \alpha_i)}{\prod_{i=1}^{n} \Gamma(\alpha_i)} \prod_{i=1}^{n} \pi_i^{\alpha_i - 1} \tag{1.1}$$

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)
??, prior = dirichlet(2)

For independent parameters, you can use:

- the uniform distribution, **uniform**$(\mathbf{a}, \mathbf{b})$:

$$f(x|a, b) = \frac{1}{b - a} \quad a \leq x \leq b \tag{1.2}$$

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

$$f(x|\lambda) = \lambda e^{-\lambda x} \quad x > 0 \tag{1.3}$$

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

$$f(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1.4}$$

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

$$f(x|\gamma, \beta, \theta) = \frac{\exp\left[-\frac{(\ln((x-\theta)/\beta))^2}{2\gamma^2}\right]}{(x - \theta)\gamma\sqrt{2\pi}} \quad x > \theta; \quad \gamma > 0; \quad \beta > 0; \tag{1.5}$$

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

$$f(x|\gamma, \beta, \theta) = \frac{(\frac{x-\theta}{\beta})^{\gamma-1} \exp\left[-\frac{x-\theta}{\beta}\right]}{\beta\Gamma(\gamma)} \quad x > \theta; \quad \gamma > 0; \quad \beta > 0; \tag{1.6}$$

Parameters of these prior distributions can become hyperparameters, *e.g.*,

??, 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 *mcmcphase* outputs and compute 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*[10]) and clusters whose BPP is lower than 50% are added sequentially as long as they do not contradict the clades established previously. It might be worth pointing out there that 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). On the contrary, 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 analysis with a 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 a 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, which can be used with *phase_viewer* or just edited manually. 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.

1.4.2 *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. Considered topologies are always 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

[10] http://evolution.genetics.washington.edu/phylip.html

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 clade file (see section 1.1.6) 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. Be warned that the optimizer might crash unexpectedly sometimes and you can change the initial values and/or random seed to overcome that (hopefully rare) problem.

Control file for *mlphase*

The control file of the *mlphase* program contains:

- a *DATAFILE block*: see the data file block section (1.3.2).
- a *MODEL block*: see the model block section (1.3.3). 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 considered fixed. By default, *mlphase* will optimize substitution parameters simultaneously to branch lengths during the inference.
    Optimize model parameters = no
  - *Starting model parameters file*: by default, sequences are 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 section 1.1.4.
    Starting model parameters file = input-data/s70-TN93dG20-7DdG20.mod
- 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"*
    Search algorithm = Exhaustive
  - *Clusters file*: this parameter is optional and is used to constrain the topology space, if present, the value expected for this field is the name of a clusters file (see section 1.1.6).
    Clusters file = input-data/hiv6-dna/hiv6.clusters
  - *Outgroup*: the outgroup can be either a species in your datafile or a cluster from your clusters file if provided. This parameter is compulsory though it is only used to root your tree for readability purpose.
    Outgroup = outgroup_clade
- *Random seed*: the seed for the random number generator. System time is used if not specified.
  Random seed = 3
- *Output file*: the name of the file to output the results.

1.4.3 *optimizer*

Using *optimizer*

The program *optimizer* 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 (see input section 1.1.4) is created to store the MLE 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-file. The control-file of the *optimizer* program must/may provide:

- a *DATAFILE block*: see the data file block section (1.3.2).
- a *MODEL block*: see the model block section (1.3.3). 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 considered fixed. By default, *optimizer* will optimize substitution parameters simultaneously to branch lengths during the inference.
    Optimize model parameters = yes
  - *Starting model parameters file*: by default, sequences are used to initialize the substitution model but the initial state can also be loaded from a file. This is probably not a very useful thing to do in practice unless you decided not to optimize the substitution parameters. For more information on this file, see section 1.1.4.
    Starting model parameters file = input-data/s70-simul/s70-TN93dG20-7DdG20.mod
- a *TREE block*: mainly used to give the topologies that are to be optimized:
  - *Tree file*: the name of the file containing the phylogeny (phylogenies), *i.e.*, topologies in the Newick format with or without initial branch lengths values.
    Tree file = input-data/hiv6-dna/hiv6.tre
  - *Clusters file*: this parameter is optional and clusters are not used in practice unless you wish to specify an outgroup clade. If present, the value expected for this field is the name of a clusters file (see section 1.1.6).
    Clusters file = input-data/hiv6-dna/hiv6.clusters
  - *Outgroup*: the outgroup can be either a species in your datafile or a cluster from your clusters file if provided. This parameter is compulsory though it is only used to root your tree for readability purpose.
    Outgroup = outgroup_clade
- *Random seed*: the integer value provided with this field is used to initialize the random number generator (used to draw random initial branch lengths if they are not provided).
  Random seed = 1
- *Output file*: the basename for the three files *basename.tre*, *basename.mod* and *basename.out*. They contain the results generated by *optimizer*.
  Output file = results/hiv6-dna/hiv6-TN93I.opt

1.4.4 *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* — at the moment, all are but the

HETEROGENEOUS model — and a set of fixed substitution parameters (*e.g.*, known frequencies, exchangeabilities, ...), *distphase* optimizes numerically 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 matrices programs[11] in *PHYLIP*[12] and Joe Felsenstein's phylogeny programs page[13]).

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 a MCMC consensus tree. You can also consider using one of the two consensus models produced by *mcmcphase*. Alternatively, you can decide to use *simulate* to produce the "stub" of *.mod* file (see section 1.1.4) 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*

In its control file, the *distphase* program requires the specification of:

- a *DATAFILE block*: see the data file block section (1.3.2).
- a *MODEL block*: see the model block section (1.3.3). The user also has to provide values for the model parameters:
  - *Model parameters file*: This field is compulsory in the **MODEL** block. It is the name of the file containing parameter values for the model. Another program of the *PHASE* package probably produced that file for you (*e.g.*, *mcmcphase*, *mlphase* or *optimizer*). You can alternatively use *simulate*) to produce it and fill it with appropriate values. For more information on this file, see section 1.1.4.
    Model parameters file = input-data/s10-simul/s10-TN93dF20.mod
- *Output format*: controls how the matrices are written on the screen and the output file "lower-triangular", "upper-triangular" and "square" are valid format. When not specified, *distphase* default to "square".
  Output format = lower-triangular
- *Output file*: the file used to store the outputs.
  Output file = results/s10-simul/s10-TN93dF20.dst

1.4.5 *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 (section 2.2) will reconstruct mismatch pairs (MM) and **TWOSTATE** model (section 2.1.4) 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.

[11] http://evolution.genetics.washington.edu/phylip/progs.data.dist.html
[12] http://evolution.genetics.washington.edu/phylip.html
[13] http://evolution.genetics.washington.edu/phylip/software.html

Control file for *likelihood*

In its control file, the *likelihood* program requires the specification of:

- a *DATAFILE block*: see the data file block section (1.3.2).
- a *MODEL block*: see the model block section (1.3.3). The user also has to provide values for the model parameters and the following control-line must be added to the **MODEL** block
  - *Model parameters file*: This field is compulsory in the **MODEL** block. It is the name of the file containing parameter values for the model. Another program of the *PHASE* package probably produced that file for you but, if it is not the case, *simulate* can help you to produce it. For more information on this file, see section 1.1.4.
    Model parameters file = input-data/s10-simul/s10-TN93dF20.mod
- a *TREE block*: this block is compulsory and has only one parameter:
  - *Tree file*: the name of the file containing the unique phylogeny, a tree in the Newick format **with** branch length values.
    Tree file = input-data/s10-simul/s10-rooted.tre
- *Site-specific substitution rates*: compute Bayesian posterior estimates of site-specific substitution rate category (see, *e.g.*, Mayrose et al., 2004; Yang and Wang, 1995):

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

  Results are stored in the given file.
  Site-specific substitution rates = results/s10-simul/s10-TN93dF20-lik.rat
- *Ancestral sequences*: marginal ancestral sequence reconstruction (see Yang et al., 1995). Results are stored in the given file.
  Ancestral sequences = results/s10-simul/s10-TN93dF20-lik.anc
- *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.
  Rate vs. composition = yes
- *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. To fill this field, you can use:
  - 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)

  Selected species = all

1.4.6 *simulate*

Using *simulate*

simulate is used:

1. to generate examples of "*.mod*" files for all the substitution models implemented in *PHASE* . A *.mod* file (see section 1.1.4) 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 one 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 (1.1.1). If the **MIXED** model described in section 2.4.1 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 optionnal parameters necessary to build a complete data file in the *PHASE* format).

Control file for *simulate*

The control file of the *simulate* program must provide

- a *MODEL block*: see the model block section (1.3.3).
- *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 mentionned above (generate a *.mod* file with default values) and **no** for the second mode (evolve sequences along a phylogeny given a substitution model).
  Retrieve the name of the model's parameters = no
- *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.
  Model parameters file = simulate.mod

  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
  Random seed = 1
- a *TREE block*: this block is described in the **TREE** block for *simulate* section (1.4.6) below. The block contains either a user-specified tree (basically the location of a *.tre* file) or the necessary parameters to generate a random one.
- *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, you just have to fill this field for the unique "class" 1.
  Number of symbols from class 1 = 500
  Number of symbols from class 2 = 500
  Number of symbols from class 3 = 100
  Number of symbols from class 4 = 200

- *Structure for the elements of class* i: with your help, *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 have specify the appropriate structure. For example, if a **MIXED** model DNA+RY+CODON+RNA is used, you would write:

Structure for the elements of class 1 = .
Structure for the elements of class 2 = .
Structure for the elements of class 3 = 123
Structure for the elements of class 4 = ()

- *Data file type* and *Number of nucleotides from class* i: with your help, *simulate* can produce an input file following the format defined in the data file format section (1.1.1). To produce this file, you have to specify yourself the type written in the first line (see section 1.1.1). 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 the model with 4 classes used in the previous exemples:

  Data file type = STRUCT #a structure line is provided
  Number of nucleotides from class 1 = 500
  Number of nucleotides from class 2 = 500
  Number of nucleotides from class 3 = 300 **#100*3**
  Number of nucleotides from class 4 = 400 **#200*2**

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

  Output file = simulated-data/sequences.mix

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 or use a supplied phylogeny. If *Generate tree* is equal to **no**, *simulate* expects a phylogeny from the user (with branch lengths).

  Generate tree = yes

- *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 (section 1.1.3) used by *PHASE* is required/used.

  Tree file = 10-species.tre

When *simulate* has to generate a random tree, the following parameters must/may be used:

- *Number of species*: the number of tips in your tree.

  Number of species = 70

- *Number of species in the outgroup*: optional parameter, you can force the creation of an outgroup clade with the specified number of species.

  Number of species in the outgroup = 8

  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.

  Generating model = Beta-splitting process

  Beta parameter: when using the beta-splitting process to generate a random topology, one has to specify the *Beta parameter*. 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.

  Beta parameter = -1

  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 length. 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.

  Branches prior = 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.
Tree height = 0.8

Here are a list of parameters associated with *Branches prior*:

- *Branch prior, upper bound* and *Branch prior, lower bound*: when *"Uniform"* is used, you have to specify a upper bound for the branch lengths. The lower bound is optionnal (0 by default).
Branches prior, upper bound = 0.8
Branches prior, lower bound = 0.001
- *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. **Branch prior, exponential parameter = 10**
- *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"*
Birth rate = 10
- *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"*
Death rate = 5

1.4.7 *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 in section 1.1.1. The *analyzer* program will ask for the fields traditionally used in the DATAFILE block (see section 1.3.2) 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 (this is the case if you answered "yes" or "auto" to the question "Heterogeneous data models?"). *analyzer* will then require a *.lmp* file that is used 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, *input-data/dna.lmp*, *input-data/RY.lmp* and *input-data/rna.lump*. The two first *.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.lump*) is used with a partition of paired sites. mismatches — *e.g.*, **AC**, **UU**, ... — are lumped into the single state **MM** and ambiguity — *e.g.*, **C-**, **UR**, ... — are lumped into the state **XX**. Sites with too many **MM** and **XX** are reported.

1.4.8 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 (see Huelsenbeck et al., 2002). 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, 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 consequent 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 tought (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**, etc.). You can also swith to an exponential prior on the rate ratios but be conscious 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.

2 - Substitution models available

2.1 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 of standard nucleotide substitution models, specific base-pair substitution models, a simple codon model and empirical amino-acid models (plus a few "home-made" models).

2.1.1 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 2.1). Lots

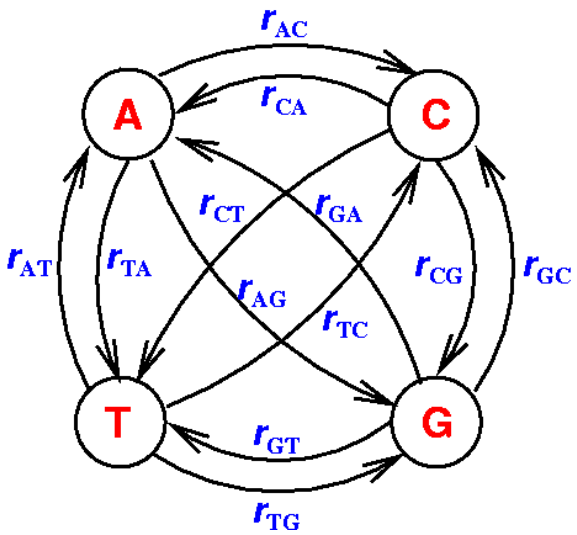


Figure 2.1: Markov model for nucleotide evolution in DNA sequences

of assumptions are made in order to make phylogenetic reconstructions more computationally feasible. First, each nucleotide is supposed to evolve independently of other sites evolution and of its past history. We suppose there is no interaction between sites and we treat them independently. 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, 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 (see section 2.4.1).

In spite of their name, DNA models can naturally be used for the treatment of the loops within RNA sequences (see figure 2.2). 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 R-Y recoding. We recently implemented a RY and a AGY substitution models in *PHASE* . These models are designed for nucleotides sequences, but they are not 4-state models. Both models lump **C** and **T** into a single state **Y** (pyrimidine) and the RY model goes further and lumps **A** and **G** into a purine **R** state. **Do not recode your sequence when using these models, this is useless**. The **TWOSTATE** and the **THREESTATE** models are described after the standard 4-state DNA models.

2.1.2 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

$$r_{ii} = -\sum_{j \neq i} r_{ij} \tag{2.1}$$

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:

$$\pi_i r_{ij} = \pi_j r_{ji} \quad \forall i, j \tag{2.2}$$

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

$$Q(i,j) = r_{ij} = m_r \times \pi_j \alpha_{ij} \quad \forall i, \forall j \neq i \tag{2.3}$$

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[1] 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.

$$m_r \times \sum_{i=1}^{nb_{states}} \sum_{j \neq i} \pi_i r_{ij} = 1.0 \tag{2.4}$$

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.

[1] the term branch length would be more correct

2.1.3 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-Cantor, 69)

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

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

Table 2.1: **JC69** transition matrix

K80 model (Kimura, 80)

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 2.2: **K80** transition matrix

HKY85 model (Hasegawa-Kishino-Yano, 85)

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 2.3: **HKY85** transition tatrix

TN93 model (Tamura-Nei, 93)

The TN93 model has four frequencies parameters. It accounts for the difference between transitions and transversions and differentiates the two kinds of transitions (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 2.4: **TN93** transition matrix

REV model (Tavare, 86)

The REV model is the most general model for nucleotide substitution 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 2.5: **REV** transition matrix

T92 model (Tamura, 92)

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. Nevertheless, the model is still a 4-state model.

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

Table 2.6: **T92** transition matrix

2.1.4 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 waste time recoding 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 2.7: **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 2.8: **THREESTATE** transition matrix

2.2 Paired-site substitution models

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

2.2.1 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 usually come as single strands but left in their environment they fold themselves in their *tertiary structure* because of the same hydrogen bonding mechanism. Helices, also known as *stems*, are formed intramolecularly .

There are 16 possible base-pairings, however of these, only six (**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 lethal consequences and selection is acting 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

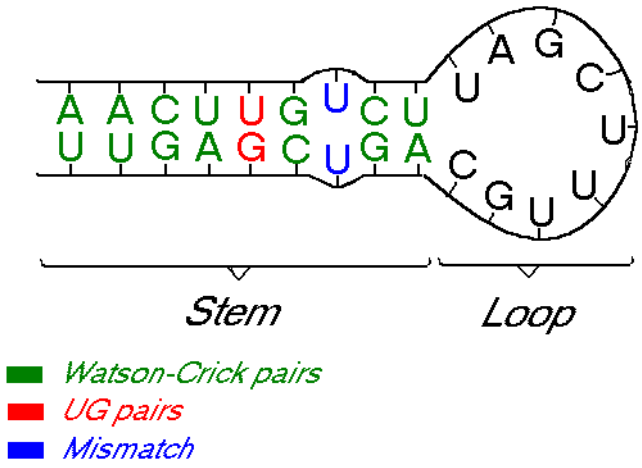


Figure 2.2: A RNA molecule secondary structure

described above but there may be a substantial bias in results because paired substitutions would seem far less probable than they are in reality (see Jow et al., 2002). 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 descibed in the theory of compensatory substitutions section below.

2.2.2 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, (see, *e.g.*, Higgs, 1998; Savill et al., 2001). 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 2.3).

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 you can have with *PHASE* seems to confirm that, 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 in which rapid interchange occurs, while interchange between the two groups, although possible, is really slow (see figure 2.4).

2.2.3 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 contains 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 lumped into a single state **MM**. The time-reversibility contraint and the

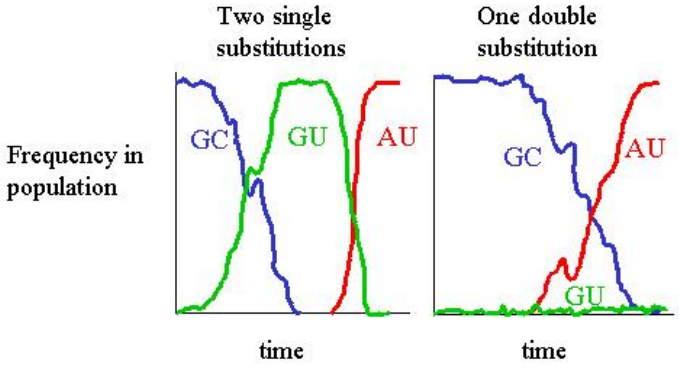


Figure 2.3: Substitution mechanisms for paired-sites

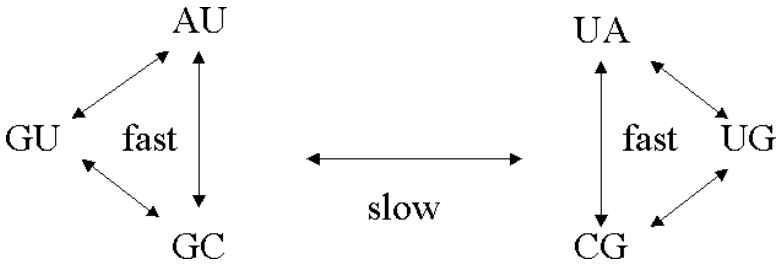


Figure 2.4: Mutation rate between paired-sites

average mutation rate are set as they were for DNA models. One can refer to Savill et al. (2001) for a better 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 properly the initial authors in this manual.

2.2.4 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 2.9.

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

Table 2.9: **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. The transition rate matrix is given in table 2.10.

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

Table 2.10: **RNA6B** transition matrix

RNA6C model (Tillier, 94)

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 2.11: **RNA6C** transition matrix

RNA6D model (Tillier, 94)

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.

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

Table 2.12: **RNA6D** transition matrix

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.

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

Table 2.13: **RNA6E** transition matrix

2.2.5 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 state MM. The **RNA7A** model is described by the following rate matrix (table 2.14).

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

Table 2.14: **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 2.15: **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 2.16: **RNA7C** transition matrix

RNA7D model (aka OTRNA, Tillier et al, 98)

The **RNA7D** model (Tillier and Collins, 1998) 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 2.17).

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

Table 2.17: **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 2.18: **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 2.19: **RNA7F** transition matrix

2.2.6 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 2.20.

RNA16B model (aka SH, 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 parameters is used for the mismatches

RNA16D model (Savill et al, 2001)

By conception, the **RNA16D** model (and its simplifications) are different from standard models we reviewed till now. They are visibly an attempt to transform the standard 4-state models by taking into account the 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 2.23 using the standard equilibrium frequency × 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 mismath 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 2.24). In **RNA16E**, **G:U** pairs are treated as mismatches.

RNA16D reduces to **RNA16F** if $\lambda_2 = \lambda_1$ (see matrix 2.25). 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*[2] 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 (2.2.2)).

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 2.26. **RNA16I** reduces to **RNA16J** with $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_5$ and **RNA16J** further reduces to **RNA16K** with $\alpha_4 = ref = 1$.

2.3 Codon and amino-acid substitution models

2.3.1 Codon model

Only one codon model is implemented in *PHASE* at the moment: **YNH98**. It was proposed by Yang et al. (1998) and is a simplification of an earlier model (Goldman and Yang, 1994). Assuming you provide a genetic code with c codons, the model has c frequency parameters $\{\pi_i\}$, a transition/transversion ratio $\kappa = \alpha_2$ and a nonsynonymous/synonymous rate ratios $d_N/d_S = \omega = \alpha_1$.

Omitting the scaling factor, the transition parameter r_{ij} of the matrix Q for this model are:

$$r_{ij} = \begin{cases} 0 & \text{if i and j differ by more than 1 nucleotide} \\ \pi_j & \text{synonymous transition} \\ \pi_j\kappa & \text{synonymous transversion} \\ \pi_j\omega & \text{nonsynonymous transition} \\ \pi_j\omega\kappa & \text{nonsynonymous transversion} \end{cases}$$

2.3.2 Empirical amino-acid models

2.4 Refinements to substitution models

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

2.4.1 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 to use multiple substitution models simultaneously to handle concatenated sequences, each substitution model having its own in dependant 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).

[2] http://morphbank.ebc.uu.se/mrbayes/

2.4.2 The HETEROGENEOUS model

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 an **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 (see, *e.g.*, Yang and Roberts, 1995; Galtier and Gouy, 1998).

Algorithms developed in *PHASE* are very similar to those implemented in *P4*[3] by Peter Foster. *PHASE* framework might be a bit more general since the full substitution model is allowed 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 (section 1.4.1) to constrain the position of the common ancestor. Remember that this outgroup can also be a monophyletic cluster (section 1.1.6).

2.4.3 Variation of frequencies across sites

Section of the manual not ready yet...

[3] http://www.nhm.ac.uk/zoology/external/p4.htm

$$Q = m_r \times \begin{array}{c|cccccccccccccccc} & 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{array}$$

Table 2.20: **RNA16A** transition matrix

$$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{array}{c} \begin{array}{cccccccccccccccc} AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU & UC & UU \end{array} \\ \left(\begin{array}{cccccccccccccccc} * & \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{array} \right) \end{array}$$

Table 2.21: **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 2.22: **RNA16C** transition matrix

$$Q = m_r \times \frac{1}{\kappa} \times \begin{array}{c|cccccccccccccccc} & AU & GU & GC & UA & UG & CG & AA & AG & AC & GA & GG & CA & CC & CU \\ AU & * & \pi_{GU}\frac{1}{\lambda_1\lambda_2\pi_U} & 0 & 0 & 0 & 0 & \pi_{AA}\frac{\alpha_1}{\lambda_1\pi_A} & \pi_{AG}\frac{\alpha_1}{\lambda_1\pi_A} & \pi_{AC}\frac{1}{\lambda_1\pi_A} & 0 & 0 & 0 & 0 & \pi_{CU}\frac{\alpha_1}{\lambda_1\pi_U} \\ GU & \pi_{AU}\frac{1}{\lambda_1\lambda_2\pi_U} & * & \pi_{GC}\frac{1}{\lambda_1\lambda_2\pi_G} & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{AG}\frac{\alpha_1}{\lambda_2\pi_G} & \pi_{GG}\frac{\alpha_1}{\lambda_2\pi_G} & 0 & 0 & \pi_{CU}\frac{\alpha_1}{\lambda_2\pi_U} \\ GC & 0 & \pi_{GU}\frac{1}{\lambda_1\lambda_2\pi_G} & * & 0 & 0 & 0 & 0 & 0 & \pi_{AC}\frac{1}{\lambda_1\pi_C} & \pi_{AG}\frac{\alpha_1}{\lambda_1\pi_G} & \pi_{GG}\frac{\alpha_1}{\lambda_1\pi_G} & 0 & \pi_{CC}\frac{\alpha_1}{\lambda_1\pi_C} & 0 \\ UA & 0 & 0 & 0 & * & \pi_{GU}\frac{1}{\lambda_1\lambda_2\pi_U} & 0 & \pi_{AA}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & \pi_{AG}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & \pi_{AC}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 \\ UG & 0 & 0 & 0 & \pi_{AU}\frac{1}{\lambda_1\lambda_2\pi_U} & * & \pi_{GC}\frac{1}{\lambda_1\lambda_2\pi_G} & 0 & \pi_{AG}\frac{\alpha_1}{\lambda_2\pi_G} & 0 & 0 & \pi_{GG}\frac{\alpha_1}{\lambda_2\pi_G} & 0 & 0 & 0 \\ CG & 0 & 0 & 0 & 0 & \pi_{GU}\frac{1}{\lambda_1\lambda_2\pi_G} & * & 0 & \pi_{AG}\frac{\alpha_1}{\lambda_1\pi_G} & 0 & 0 & \pi_{GG}\frac{\alpha_1}{\lambda_1\pi_G} & \pi_{AC}\frac{1}{\lambda_1\pi_C} & \pi_{CC}\frac{\alpha_1}{\lambda_1\pi_C} & \pi_{CU}\frac{\alpha_1}{\lambda_1\pi_C} \\ AA & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & * & \pi_{AG}\frac{1}{\pi_A} & \pi_{AC}\frac{\alpha_1}{\pi_A} & \pi_{AG}\frac{1}{\pi_A} & 0 & \pi_{AC}\frac{1}{\pi_A} & 0 & 0 \\ AG & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & 0 & \pi_{GU}\frac{\alpha_1}{\lambda_2\pi_G} & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G} & \pi_{AA}\frac{1}{\pi_A} & * & \pi_{AC}\frac{\alpha_1}{\pi_A} & 0 & \pi_{GG}\frac{1}{\pi_G} & 0 & 0 & 0 \\ AC & \pi_{AU}\frac{1}{\lambda_1\pi_A} & 0 & \pi_{GC}\frac{1}{\lambda_1\pi_C} & 0 & 0 & 0 & \pi_{AA}\frac{alpha_1}{\pi_A} & \pi_{AG}\frac{\alpha_1}{\pi_A} & * & 0 & 0 & 0 & \pi_{CC}\frac{\alpha_1}{\pi_C} & 0 \\ GA & 0 & \pi_{GU}\frac{\alpha_1}{\lambda_2\pi_G} & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G} & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & \pi_{AA}\frac{\alpha_1}{\pi_A} & 0 & 0 & * & \pi_{GG}\frac{1}{\pi_G} & \pi_{AC}\frac{\alpha_1}{\pi_A} & 0 & 0 \\ GG & 0 & \pi_{GU}\frac{\alpha_1}{\lambda_2\pi_G} & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G} & 0 & \pi_{GU}\frac{\alpha_1}{\lambda_2\pi_G} & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G} & 0 & \pi_{AG}\frac{1}{\pi_G} & 0 & \pi_{AG}\frac{1}{\pi_G} & * & 0 & 0 & 0 \\ CA & 0 & 0 & 0 & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & \pi_{GC}\frac{1}{\lambda_1\pi_C} & \pi_{AA}\frac{1}{\pi_A} & 0 & 0 & \pi_{AG}\frac{\alpha_1}{\pi_A} & 0 & * & \pi_{CC}\frac{\alpha_1}{\pi_C} & \pi_{CU}\frac{\alpha_1}{\pi_C} \\ CC & 0 & 0 & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C} & 0 & 0 & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C} & 0 & 0 & \pi_{AC}\frac{\alpha_1}{\pi_C} & 0 & 0 & \pi_{AC}\frac{\alpha_1}{\pi_C} & * & \pi_{CU}\frac{1}{\pi_C} \\ CU & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U} & \pi_{GU}\frac{\alpha_1}{\lambda_2\pi_U} & 0 & 0 & 0 & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C} & 0 & 0 & 0 & 0 & 0 & \pi_{AC}\frac{\alpha_1}{\pi_C} & \pi_{CC}\frac{1}{\pi_C} & * \\ UC & 0 & 0 & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C} & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U} & \pi_{GU}\frac{\alpha_1}{\lambda_2\pi_U} & 0 & 0 & 0 & \pi_{AC}\frac{\alpha_1}{\pi_C} & 0 & 0 & 0 & \pi_{CC}\frac{1}{\pi_C} & 0 \\ UU & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U} & \pi_{GU}\frac{\alpha_1}{\lambda_2\pi_U} & 0 & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U} & \pi_{GU}\frac{\alpha_1}{\lambda_2\pi_U} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{CU}\frac{1}{\pi_U} \end{array}$$

Table 2.23: **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 \\
AU & * & \pi_{GU}\frac{1}{\lambda_1\pi_U} & 0 & 0 & 0 & 0 & \pi_{AA}\frac{\alpha_1}{\lambda_1\pi_A} & \pi_{AG}\frac{\alpha_1}{\lambda_1\pi_A} & \pi_{AC}\frac{1}{\lambda_1\pi_A} & 0 & 0 & 0 & 0 & \pi_{CU}\frac{\alpha_1}{\lambda_1\pi_U} & 0 \\
GU & \pi_{AU}\frac{1}{\lambda_1\pi_U} & * & \pi_{GC}\frac{1}{\lambda_1\pi_G} & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{AG}\frac{\alpha_1}{\pi_G} & \pi_{GG}\frac{\alpha_1}{\pi_G} & 0 & 0 & \pi_{CU}\frac{\alpha_1}{\pi_U} & 0 \\
GC & 0 & \pi_{GU}\frac{1}{\lambda_1\pi_G} & * & 0 & 0 & 0 & 0 & 0 & \pi_{AC}\frac{1}{\lambda_1\pi_C} & \pi_{AG}\frac{\alpha_1}{\lambda_1\pi_G} & \pi_{GG}\frac{\alpha_1}{\lambda_1\pi_G} & 0 & \pi_{CC}\frac{\alpha_1}{\lambda_1\pi_C} & 0 & \pi_{CU}\frac{\alpha_1}{\lambda_1\pi_C} \\
UA & 0 & 0 & 0 & * & \pi_{GU}\frac{1}{\lambda_1\pi_U} & 0 & \pi_{AA}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & \pi_{AG}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & \pi_{AC}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & \pi_{CU}\frac{\alpha_1}{\lambda_1\pi_U} \\
UG & 0 & 0 & 0 & \pi_{AU}\frac{1}{\lambda_1\pi_U} & * & \pi_{GC}\frac{1}{\lambda_1\pi_G} & 0 & \pi_{AG}\frac{\alpha_1}{\pi_G} & 0 & 0 & \pi_{GG}\frac{\alpha_1}{\pi_G} & 0 & 0 & 0 & \pi_{CU}\frac{\alpha_1}{\pi_U} \\
CG & 0 & 0 & 0 & 0 & \pi_{GU}\frac{1}{\lambda_1\pi_G} & * & 0 & \pi_{AG}\frac{\alpha_1}{\lambda_1\pi_G} & 0 & 0 & \pi_{GG}\frac{\alpha_1}{\lambda_1\pi_G} & \pi_{AC}\frac{1}{\lambda_1\pi_C} & \pi_{CC}\frac{\alpha_1}{\lambda_1\pi_C} & \pi_{CU}\frac{\alpha_1}{\lambda_1\pi_C} & 0 \\
AA & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & * & \pi_{AG}\frac{1}{\pi_A} & \pi_{AC}\frac{\alpha_1}{\pi_A} & \pi_{AG}\frac{1}{\pi_A} & 0 & \pi_{AC}\frac{1}{\pi_A} & 0 & 0 & 0 \\
AG & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & 0 & \pi_{GU}\frac{\alpha_1}{\pi_G} & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G} & \pi_{AA}\frac{1}{\pi_A} & * & \pi_{AC}\frac{\alpha_1}{\pi_A} & 0 & \pi_{GG}\frac{1}{\pi_G} & 0 & 0 & 0 & 0 \\
AC & \pi_{AU}\frac{1}{\lambda_1\pi_A} & 0 & \pi_{GC}\frac{1}{\lambda_1\pi_C} & 0 & 0 & 0 & \pi_{AA}\frac{alpha_1}{\pi_A} & \pi_{AG}\frac{\alpha_1}{\pi_A} & * & 0 & 0 & 0 & \pi_{CC}\frac{\alpha_1}{\pi_C} & 0 & \pi_{CU}\frac{\alpha_1}{\pi_C} \\
GA & 0 & \pi_{GU}\frac{\alpha_1}{\pi_G} & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G} & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & 0 & \pi_{AA}\frac{\alpha_1}{\pi_A} & 0 & 0 & * & \pi_{GG}\frac{1}{\pi_G} & \pi_{AC}\frac{\alpha_1}{\pi_A} & 0 & 0 & 0 \\
GG & 0 & \pi_{GU}\frac{\alpha_1}{\pi_G} & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G} & 0 & \pi_{GU}\frac{\alpha_1}{\pi_G} & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G} & 0 & \pi_{AG}\frac{1}{\pi_G} & 0 & \pi_{AG}\frac{1}{\pi_G} & * & 0 & 0 & 0 & 0 \\
CA & 0 & 0 & 0 & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A} & 0 & \pi_{GC}\frac{1}{\lambda_1\pi_C} & \pi_{AA}\frac{1}{\pi_A} & 0 & 0 & \pi_{AG}\frac{\alpha_1}{\pi_A} & 0 & * & \pi_{CC}\frac{\alpha_1}{\pi_C} & \pi_{CU}\frac{\alpha_1}{\pi_C} & 0 \\
CC & 0 & 0 & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C} & 0 & 0 & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C} & 0 & 0 & \pi_{AC}\frac{\alpha_1}{\pi_C} & 0 & 0 & \pi_{AC}\frac{\alpha_1}{\pi_C} & * & \pi_{CU}\frac{1}{\pi_C} & \pi_{CU}\frac{1}{\pi_C} \\
CU & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U} & \pi_{GU}\frac{\alpha_1}{\pi_U} & 0 & 0 & 0 & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C} & 0 & 0 & 0 & 0 & 0 & \pi_{AC}\frac{\alpha_1}{\pi_C} & \pi_{CC}\frac{1}{\pi_C} & * & 0 \\
UC & 0 & 0 & \pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C} & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U} & \pi_{GU}\frac{\alpha_1}{\pi_U} & 0 & 0 & 0 & \pi_{AC}\frac{\alpha_1}{\pi_C} & 0 & 0 & 0 & \pi_{CC}\frac{1}{\pi_C} & 0 & * \\
UU & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U} & \pi_{GU}\frac{\alpha_1}{\pi_U} & 0 & \pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U} & \pi_{GU}\frac{\alpha_1}{\pi_U} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \pi_{CU}\frac{1}{\pi_U} & \pi_{CU}\frac{1}{\pi_U}
\end{pmatrix}$$

Table 2.24: **RNA16E** transition matrix

$$Q = m_r \times \frac{1}{\kappa} \times$$

	AU	GU	GC	UA	UG	CG	AA	AG	AC	GA	GG	CA	CC	CU
AU	$*$	$\pi_{GU}\frac{1}{\lambda_1\lambda_1\pi_U}$	0	0	0	0	$\pi_{AA}\frac{\alpha_1}{\lambda_1\pi_A}$	$\pi_{AG}\frac{\alpha_1}{\lambda_1\pi_A}$	$\pi_{AC}\frac{1}{\lambda_1\pi_A}$	0	0	0	0	$\pi_{CU}\frac{\alpha_1}{\lambda_1\pi_U}$
GU	$\pi_{AU}\frac{1}{\lambda_1\lambda_1\pi_U}$	$*$	$\pi_{GC}\frac{1}{\lambda_1\lambda_1\pi_G}$	0	0	0	0	0	0	$\pi_{AG}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{GG}\frac{\alpha_1}{\lambda_1\pi_G}$	0	0	$\pi_{CU}\frac{\alpha_1}{\lambda_1\pi_U}$
GC	0	$\pi_{GU}\frac{1}{\lambda_1\lambda_1\pi_G}$	$*$	0	0	0	0	0	$\pi_{AC}\frac{1}{\lambda_1\pi_C}$	$\pi_{AG}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{GG}\frac{\alpha_1}{\lambda_1\pi_G}$	0	$\pi_{CC}\frac{\alpha_1}{\lambda_1\pi_C}$	0
UA	0	0	0	$*$	$\pi_{GU}\frac{1}{\lambda_1\lambda_1\pi_U}$	0	$\pi_{AA}\frac{\alpha_1}{\lambda_1\pi_A}$	0	0	$\pi_{AG}\frac{\alpha_1}{\lambda_1\pi_A}$	0	$\pi_{AC}\frac{\alpha_1}{\lambda_1\pi_A}$	0	0
UG	0	0	0	$\pi_{AU}\frac{1}{\lambda_1\lambda_1\pi_U}$	$*$	$\pi_{GC}\frac{1}{\lambda_1\lambda_1\pi_G}$	0	$\pi_{AG}\frac{\alpha_1}{\lambda_1\pi_G}$	0	0	$\pi_{GG}\frac{\alpha_1}{\lambda_1\pi_G}$	0	0	0
CG	0	0	0	0	$\pi_{GU}\frac{1}{\lambda_1\lambda_1\pi_G}$	$*$	0	$\pi_{AG}\frac{\alpha_1}{\lambda_1\pi_G}$	0	0	$\pi_{GG}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{AC}\frac{1}{\lambda_1\pi_C}$	$\pi_{CC}\frac{\alpha_1}{\lambda_1\pi_C}$	$\pi_{CU}\frac{\alpha_1}{\lambda_1\pi_C}$
AA	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A}$	0	0	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A}$	0	0	$*$	$\pi_{AG}\frac{1}{\pi_A}$	$\pi_{AC}\frac{\alpha_1}{\pi_A}$	$\pi_{AG}\frac{1}{\pi_A}$	0	$\pi_{AC}\frac{1}{\pi_A}$	0	0
AG	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A}$	0	0	0	$\pi_{GU}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{AA}\frac{1}{\pi_A}$	$*$	$\pi_{AC}\frac{\alpha_1}{\pi_A}$	0	$\pi_{GG}\frac{1}{\pi_G}$	0	0	0
AC	$\pi_{AU}\frac{1}{\lambda_1\pi_A}$	0	$\pi_{GC}\frac{1}{\lambda_1\pi_C}$	0	0	0	$\pi_{AA}\frac{alpha_1}{\pi_A}$	$\pi_{AG}\frac{\alpha_1}{\pi_A}$	$*$	0	0	0	$\pi_{CC}\frac{\alpha_1}{\pi_C}$	0
GA	0	$\pi_{GU}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A}$	0	0	$\pi_{AA}\frac{\alpha_1}{\pi_A}$	0	0	$*$	$\pi_{GG}\frac{1}{\pi_G}$	$\pi_{AC}\frac{\alpha_1}{\pi_A}$	0	0
GG	0	$\pi_{GU}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G}$	0	$\pi_{GU}\frac{\alpha_1}{\lambda_1\pi_G}$	$\pi_{GC}\frac{\alpha_1}{\lambda_1\pi_G}$	0	$\pi_{AG}\frac{1}{\pi_G}$	0	$\pi_{AG}\frac{1}{\pi_G}$	$*$	0	0	0
CA	0	0	0	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_A}$	0	$\pi_{GC}\frac{1}{\lambda_1\pi_C}$	$\pi_{AA}\frac{1}{\pi_A}$	0	0	$\pi_{AG}\frac{\alpha_1}{\pi_A}$	0	$*$	$\pi_{CC}\frac{\alpha_1}{\pi_C}$	$\pi_{CU}\frac{\alpha_1}{\pi_C}$
CC	0	0	$\pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C}$	0	0	$\pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C}$	0	0	$\pi_{AC}\frac{\alpha_1}{\pi_C}$	0	0	$\pi_{AC}\frac{\alpha_1}{\pi_C}$	$*$	$\pi_{CU}\frac{1}{\pi_C}$
CU	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U}$	$\pi_{GU}\frac{\alpha_1}{\lambda_1\pi_U}$	0	0	0	$\pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C}$	0	0	0	0	0	$\pi_{AC}\frac{\alpha_1}{\pi_C}$	$\pi_{CC}\frac{1}{\pi_C}$	$*$
UC	0	0	$\pi_{GC}\frac{\alpha_1}{\lambda_1\pi_C}$	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U}$	$\pi_{GU}\frac{\alpha_1}{\lambda_1\pi_U}$	0	0	0	$\pi_{AC}\frac{\alpha_1}{\pi_C}$	0	0	0	$\pi_{CC}\frac{1}{\pi_C}$	0
UU	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U}$	$\pi_{GU}\frac{\alpha_1}{\lambda_1\pi_U}$	0	$\pi_{AU}\frac{\alpha_1}{\lambda_1\pi_U}$	$\pi_{GU}\frac{\alpha_1}{\lambda_1\pi_U}$	0	0	0	0	0	0	0	0	$\pi_{CU}\frac{1}{\pi_U}$

Table 2.25: **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 2.26: **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 2.27: **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 2.28: **RNA16K** transition matrix

Bibliography

Aldous, D.
1996. Probability distributions on cladograms. In *Random Discrete Structures*, Pp. 1–18. Aldous, D. and Pemantle, R.

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

Galtier, N. and M. Gouy
1998. 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.

Goldman, N. and Z. Yang
1994. A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol. Biol. Evol.*, 11:725–736.

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

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

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

Hudelot, C., V. Gowri-Shankar, H. Jow, M. Rattray, and P. Higgs
2003. RNA-based phylogenetics methods: Application to mammalian mitochondrial RNA sequences. *Mol. Phyl. Evol.*, 28(2):241–252.

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

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

Jukes, T. H. and C. R. Cantor
1969. Evolution of protein molecules. In *Mammalian Protein Metabolism*, volume 3, Pp. 21–132. Munro, H. N., ed.

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

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

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

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

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

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

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

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

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

Tamura, K. and M. Nei
1993. 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.

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

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

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

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

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

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

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

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

Yang, Z., R. Nielsen, and M. Hasegawa
1998. Models of amino acid substitution and applications to mitochondrial protein evolution. *Mol. Biol. Evol.*, 15:1600–1611.

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

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

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

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