[Draft] Manual

BayesTraits

Mark Pagel and Andrew Meade
School of Biological Sciences
University of Reading
Reading RG6 6AJ

www.evolution.rdg.ac.uk

BayesTraits is a computer package for performing analyses of trait evolution among groups of species for which a phylogeny or sample of phylogenies is available. It can be applied to the analysis of traits that adopt a finite number of discrete states, or to the analysis of continuously varying traits. Hypotheses can be tested about models of evolution, about ancestral states and about correlations among pairs of traits.

History
BayesTraits combines a number of computer programs that we have previously made available, including MultiState, MultiAns, Discrete and Continuous. We will increasingly support only this package and so users are encouraged to switch to BayesTraits if currently using one of these older versions. The methods in *BayesTraits* are described in a series of papers that we will refer to throughout this manual (see also references at end). The computer code is all written in C and was produced by Dr Andrew Meade.

We would be grateful if you use *BayesTraits* in your published research if you cite one or more of the relevant articles and indicate that the software is available from www.evolution.rdg.ac.uk.

Methods and Approach
BayesTraits uses Markov chain Monte Carlo (MCMC) methods to derive posterior distributions and maximum likelihood (ML) methods to derive point estimates of, log-likelihoods, the parameters of statistical models, and the values of traits at ancestral nodes of phylogenies. The user can select either standard or conventional MCMC or reversible-jump MCMC. In the latter case the Markov chain searches the posterior distribution of different models of evolution as well as the posterior distributions of the parameters of these models (see below).

BayesTraits can be used with a single phylogenetic tree in which case only uncertainty about model parameters is explored, or, it can be applied to suitable samples of trees such that models are estimated and hypotheses are tested taking phylogenetic uncertainty into account.

Our *BayesPhylogenies* package (www.evolution.rdg.ac.uk) can be used to generate posterior distributions of phylogenetic trees when a gene-sequence alignment or other data set is available.

BayesTraits makes available several methods:

- *BayesMultiState* is used to reconstruct how traits that adopt a finite number of discrete states evolve on phylogenetic trees. It is useful for reconstructing ancestral states and for testing models of trait evolution. It can be applied to traits that adopt two or more discrete states (see Pagel, M., Meade, A. and Barker, D. 2004. *Systematic Biology*, 53, 673-684;

- *BayesDiscrete* is used to analyse correlated evolution between pairs of discrete binary traits. Most commonly the two binary states refer to the presence or absence of some feature, but could also include “low” and “high”, or any two distinct features. Its uses might include tests of correlation among behavioural, morphological, genetic or cultural characters (see Pagel, M. and Meade, A. 2006. *American Naturalist*, 167, 808-825.) Once recent use of *BayesDiscrete* is to test for functional linkage among pairs of genes (Barker, D. and Pagel, M. 2005. *PLoS Computational Biology*, 1, 24-31. DOI: 10.1371/journal.pcbi.0010003);

- *BayesContinuous* is for the analysis of the evolution of continuously varying traits. It can be used to model the evolution of a single trait, to study correlations among pairs of traits, or to study the regression of one trait on two or more other traits (see Pagel, M. 1999. *Nature*, 401, 877-884).

This manual is designed to show how to use the programs that implement these models. Detailed information about the methods can be found in the papers listed at the end (some are available as pdfs on our website). Syntax and a description of all of the commands in *BayesTraits* can be found in the list of commands in the Appendix to the manual.

THE CONTINUOUS-TIME MARKOV MODELS OF TRAIT EVOLUTION FOR DISCRETE TRAITS

Multistate and *Discrete* fit continuous-time Markov models to the discrete character data. This model allows the trait to change from the state it is in at any given moment to any other state over infinitesimally small intervals of time. The rate parameters of the model estimate these transition rates (see Pagel, 1994 for further discussion). The model traverses the tree estimating transition rates and the likelihood associated with different states at each node.

The table shows an example of the model of evolution for a trait that can adopt three states, 0,1, and 2. The q_{ij} are the transition rates among the three states, and these are what the method estimates on the tree, based on the distribution of states among the species. If these rates differ from zero, this indicates that they are a significant component of the model. The main diagonal elements are not estimated but are a function of the other values in their row.

Example of the model of evolution for a trait that adopts three states

State	*0*	*1*	*2*
0	--	q_{01}	q_{02}
1	q_{01}	--	q_{12}
2	q_{20}	q_{21}	--

For a trait that adopts four states, the matrix would have twelve entries, for a binary trait the matrix would have just two entries.

Discrete tests for correlated evolution in two binary traits by comparing the fit (log-likelihood) of two of these continuous-time Markov models. One of these is a model in which the two traits evolve independently on the tree. Each trait is described by a 2 × 2 matrix in the same format as the one above, but in which the trait adopts just two states, "0" and "1". This creates two rate coefficients per trait.

The other model, allows the traits to evolve in a correlated fashion such that the rate of change in one trait depends upon the background state of the other. The dependent model can adopt four states, one for each combination of the two binary traits (0,0; 0,1; 1,0; 1,1). It is represented in the program as shown below and the transition rates describe all possible changes in one state holding the other constant. The main diagonal elements are estimated from the other values in their row as before. The other diagonal elements are set to zero as the model does not allow 'dual' transitions to occur, the logic being that these are instantaneous transition rates and the probability of two traits changing at exactly the same instant of time is negligible. Dual transitions are allowed over longer periods of time, however. See Pagel (1994) and Barker and Pagel (2005) for further discussion of this model.

State	*0,0*	*0,1*	*1,0*	*1,1*
0,0	--	q_{12}	q_{13}	--
0,1	q_{21}	--	--	q_{24}
1,0	q_{31}	--	--	q_{34}
1,1	--	q_{42}	q_{43}	

The values of the transiton rate parameters will depend upon the units of measurement in the phylogeny. In general if the branch lengths are increased by a factor 'c' the transition rates will be decreased by the same factor 'c'. This has implications for modelling the rate parameters in Markov chains as discussed below.

Covarion model. *BayesTraits* implements the covarion model for trait evolution (Tuffley and Steele, *Math. Biosci*. 147:63–91, 1998). This is a variant of the continuous-time Markov model that allows for traits to vary their rate of evolution within and between branches. It is an elegant model that deserves more attention, although users may find it of limited value with comparative data – the model may require many sites to be estimated well.

The Generalised Least Squares model for continuously varying traits

Continuous analyses phylogenetically structured continuously varying data using a generalised least squares (GLS) approach that assumes a Brownian motion model of evolution (see Pagel, 1997, 1999). In the GLS model, non-independence among the species is accounted for by reference to a matrix of the expected covariances among species. This matrix is derived from the phylogenetic tree. The model estimates the variance of evolutionary change (the Brownian motion parameter), sometimes called the 'rate' of change, and the ancestral state of traits at the root of the tree. It can also estimate the covariance of changes between pairs of traits, and this is how it tests for correlation.

The GLS approach means that data can be plotted across species and interpreted using the correlations and regressions obtained from *Continuous*. The GLS approach as implemented in *Continuous* also makes it possible to transform and scale the phylogeny to test the adequacy of the underlying model of evolution, to assess whether phylogenetic correction of the data is required, and to test hypotheses about trait evolution itself – for example, is trait evolution punctuational or gradual, is there evidence for adaptive radiation, is the rate of evolution constant. These ideas will be discussed more under the ***Continuous*** heading below.

Hypothesis Testing: Likelihood ratios and Bayes Factors
BayesTraits does not test hypotheses for you but prints out the information needed to make hypothesis tests. These will be discussed in more detail in conjunction with the examples below, but here we outline the two kinds of tests most often used.

The likelihoods ratio (LR) test is often used to compare two likelihoods derived from nested model (models that can be expressed such that one is a special or general case of the other). The likelihood ratio statistic is calculated as

LR= 2[log-likelihood(better fitting model) – log-likelihood(worse fitting model)]

The likelihood ratio statistic is nominally distributed as a χ^2 with degrees of freedom equal to the difference in the number of parameters between the two models. However, in some circumstances (see Pagel, 1994, 1997 and Barker and Pagel, 2005) the test may follow a χ^2 with fewer degrees of freedom.

Variants of the LR test include the Akaike Information Criterion and the Bayesian Information Criterion. We do not describe these tests here. They are discussed in many works on phylogenetic inference (see for example, Felsenstein. Inferring Phylogenies, 2004).

The LR, Akaike and Bayesian Information Criterion tests presume that the likelihood is at or near its maximum likelihood value. In a MCMC framework tests of likelihood often rely on Bayes factors. The logic is similar to that for the likelihood ratio test, except here we compare the marginal likelihoods of two models rather than their maximum likelihoods.

The marginal likelhood of a model is the integral of the model likelihoods over all values of the models parameters and over possible trees. In practice this marginal likelihood is difficult to estimate but research shows it can be well approximated by the harmonic mean of the likelihoods allowing the Markov chain to run for a very large number of iterations (millions).

BayesTraits calculates the logarithm of the harmonic mean of the likelihoods as the program runs, having ignored the likelihoods during the burn-in period when the model is moving to convergence. The running tally of harmonic means is read from the final iteration of the chain and the values for the independent and dependent models are then compared. The test statistic is just

2(log[harmonic mean(better model)] – log[harmonic mean(worse model)]

Any positive value favours the dependent model, but conventionally a ratio greater than 2 is taken as 'positive' evidence, greater than 5 is 'strong' and greater than 10 is 'very strong' evidence.

General Features of BayesTraits

The package is run as a command line program in a Unix window or as 'batch' runs using an input file. To run the program as a command line start the program in Unix by typing

```
./BayesTraits treefile.trees inputdata.txt
```

It is easiest to have all of the files in the same folder. If not, you will need to type in the paths to the input and data files. Every program within *BayesTraits* minimally requires a treefile and an input data file.

i) treefile: a rooted tree file in nexus format with one or more trees in it and the trees must have branch lengths. The trees in the nexus file must be rooted such that the root forms a binary node at the base of the tree, with a positive branch length leading to the outgroup(s). Failure to root trees properly or including trees without branch lengths is one of the most common errors in using *BayesTraits.* There are two example treefiles bundled with the package: primate.trees and PPI.trees. The example below shows two treefiles from work on a eukaryote phylogeny and shows the format *BayesTraits* uses.

```
#NEXUS
begin trees;
   translate
         0 F_graminea,
         1 N_crassa,
         2 A_nidulans,
         3 S_cerevisiae,
         4 S_mikatae,
         5 S_paradoxus,
         6 S_bayanus,
         7 S_castellii,
         8 S_kluyveri,
         9 C_albicans,
         10 S_pombe,
         11 C_neoformans,
         12 M_grisea,
         13 C_elegans,
         14 D_melanogaster;
         tree No_001 =
(((((((((3:0.0130691766,5:0.0151973184):0.0022974291,4:0.0267966019):0.0
114844024,6:0.0296853757):0.0204440681,7:0.0465196055):0.0120969941,8:0.0
442292708):0.0209112159,9:0.1115276110):0.2651100835,10:0.1513436882):0.1
205270384,(((1:0.0662049379,0:0.0606875577):0.0155381574,12:0.1123581170)
:0.0360606937,2:0.0968434947):0.0236107824):0.0350977591,11:0.1712569743)
:0.1442959071,(14:0.1088280199,13:0.2266176629):0.1442959071);
         tree No_002 =
(((((((((3:0.0155205592,5:0.0127990548):0.0080254949,4:0.0233694460):0.0
097639217,6:0.0308958262):0.0277049990,7:0.0648142009):0.0179206608,8:0.0
428995908):0.0264746423,9:0.1098601223):0.2408391724,10:0.1199050581):0.1
222602939,(((1:0.0598463471,0:0.0709403415):0.0129094643,12:0.1216388258)
:0.0436879034,2:0.0893288737):0.0277157583):0.0379354529,11:0.1931459172)
:0.1114094552,(14:0.1720042840,13:0.2580361822):0.1114094552);
end;
```

ii) input data file: the input data file is a plain text file. It has one line for each species or taxon in the tree. The names must be spelled exactly as in the tree and must not have any spaces within them. They do not have to be in the same order. Following a species name, leave white space or use a tab and enter the first column of data. Repeat this for additional columns of data. *BayesMultistate* and *BayesContinuous* can take any number of traits as input. For *BayesMultistate* these will normally be values such as "0", "1", "2", or "A", "B", "C", etc. If more than one column of data is input in *BayesMultistate*, the program fits a common evolutionary model to them all. For *BayesContinuous* the input data can be integers or floating point numbers. In addition *BayesMultistate* allows some taxa to be assigned a restricted number of states (see example below). *BayesDiscrete* must have exactly two columns of data and they must take the values "0" and "1". In all of the programs, if a data point is missing a '-' symbol can be used.

<u>Example of MultiState data</u>

Taxon 1	A	A	C
Taxon 2	B	B	C
Taxon 3	A	B	-
Taxon 4	C	C	B
…….			
Taxon n	BC	A	B

Taxon 3 has missing data for the third site. Missing data are treated as if the trait could take any of the other states. The first trait for *Taxon n* is uncertain. The code BC signifies that it can take states B or in trait C (with equal probability) but is not allowed to be in state A.

Example of Discrete (binary) data

Taxon 1	0	0
Taxon 2	0	0
Taxon 3	1	0
Taxon 4	0	1
……..		
Taxon n	1	1

Example of Continuous data

Taxon 1	10	9.0
Taxon 2	1.06	0.25
Taxon 3	5	2
Taxon 4	3	4
……..		
Taxon n	1	1.1

Having launched the program by giving it a treefile and and input data file this screen will appear

```
Please Select the model of evolution to use.
1)    MultiState.
2)    Discrete: Independent model
3)    Discrete: Dependent model
4)    Continuous: Random Walk (Model A)
5)    Continuous: Directional (Model B)
6)    Continuous: Regression
```

Type in the number of the model required. These models will be described below in conjunction with each method. Models 2 and 3 are for testing correlated evolution, model 4 fits the non-directional random walk model (Pagel, 1997, 1999) in *Continuous*, model 5 fits the directional model, and model 6 allows regression models to be fitted within *Continuous*.

Once you’ve done this you will be asked to select either maximum likelihood or MCMC as the method of analysis

```
Please Select the analysis method to use.
1)    Maximum Likelihood.
2)    MCMC
```

If maximum likelihood is chosen then the program will calculate the ML values of all parameters separately on each tree in the treefile. The MCMC method estimates the Bayesian posterior distributions of the likelihoods of the data given the model and tree, and the parameters of the model of evolution. As with all Bayesian methods the results of the analyses are qualified in terms of the data, model and the priors.

When performing MCMC analyses it may be a good idea to analyse the data first using ML to get an idea of the ‘optimal’ rate parameters and likelihood for each tree.

Once a method of analysis is chosen the program will print out some information on the size of your tree and the analysis you have chosen. Typing “run” then starts the analysis. During the analysis the program writes out an output file. These features will be discussed separately for each method.

Two useful commands that can be typed at any time are the **info** and **help** commands. **Info** lists your input files and tells you the current status of your choices and settings. **Help** provides a list of all of the commands in the package.

Using BayesTraits in Batch mode

To use BayesTraits in batch mode you simply give the program an additional input file containing the instructions for the run. For example

```
./BayesTraits treefile.trees inputdata.txt <inputcommands.txt
```

tells the program to retrieve its commands from the file inputcommands.txt. This must be a pure text file and the first two lines of it must choose the model of evolution and the method. After those two choice, any of the commands we describe below can be input and in any order. Here is an example:

Example input command file for running in batch mode. A hash before a command makes it act as a comment. All commands here can be found in the appendix.

```
# Select MultiState Analysis
1
# Select MCMC mode of analysis
2
# Reconstruct and internal node, using Most Recent Common Ancestor method

AddMRCA Node-H 51 52 50 49 47 48

# Use a hyperprior that seeds an exponential distribution from a uniform on the
# the interval 0.0 to 30

rjhp exp 0.0 30

# Chooses a rate deviation so the
# acceptance rate is 0.2

ratedev 10

# Allow the chain to burn-in for 10,000 iterations

burnin 10000

# Run the analysis.

run
```

Setting priors in MCMC mode. When using the MCMC analysis method, the prior distributions of the parameters of the model of evolution must be chosen. The values of the rate parameters are dependent upon the branch lengths of the tree. Other things equal, longer branches will require smaller rate parameters and *vice versa*. This is why the user must set the kind of prior (e.g., uniform, exponential) and the prior-interval or range of values the prior covers.

Uniform or uninformative priors should be used if possible as these assume all values of the parameters are equally likely *a priori* and are therefore easily justified. Uniform priors can be used when the signal in the data is strong. But in a comparative study there will typically only be one or a few data points (unlike the many hundreds or thousands in a typical gene-sequence alignment) and so a stronger prior than a uniform may be required.

Priors are the soft underbelly of Bayesian analyses. The guiding principle is that if the choice of prior is critical for a result, you must have a good reason for choosing that prior. It is often useful to run maximum likelihood analyses on your trees to get a sense of the average values of the parameters. One option if a uniform with a wide interval does not constrain the parameters is to use a uniform prior with a narrower range of values, and this might be justified either on biological grounds or perhaps on the ML results. The ML results will not define the range of the prior but can give an indication of its midpoint.

NOTE: A rule of thumb when choosing a constrained or informed (non-uniform) prior is that if the posterior distribution of parameter values seems truncated at either the upper or lower end of the constrained range, then the limits on the prior must be changed.

The program allows exponential, gamma and beta distributed priors (beta priors not fully implemented yet). The exponential distribution always has its mode at zero and then slopes down, whereas the gamma can take a variety of uni-modal shapes or even mimic the exponential. The exponential prior is useful when the general feeling is that smaller values of parameters are more likely than larger ones. If the parameters are thought to take an intermediate value, a gamma prior with an intermediate mean can be used.

Priors are set differently in normal and hyperprior MCMC modes. In the normal MCMC mode you specify the mean of the prior distribution for exponential priors or the mean and variance if using a gamma prior. Priors can be set individually for each parameter (**prior** command) or more commonly all parameters can follow the same prior (**priorall** command).

Because it can be difficult to arrive at suitable values for the parameters of the prior distributions when using exponential or gamma priors, *BayesTraits* allows the use of a *hyperprior*. A hyperprior is simply a distribution – usually a uniform -- from which are drawn values to seed the values of the exponential or gamma priors. We recommend using hyperpriors as they provide an elegant way to reduce some of the uncertainty and arbitrariness of choosing priors in MCMC studies. For an example of selecting priors and using a hyperprior see Pagel, M., Meade, A. and Barker, D. 2004 Bayesian estimation of ancestral character states on phylogenies. *Systematic Biology*, 53, 673-684.

When using the hyperprior approach (**hyperprior** or **reversible-jump hyperprior** commands) you specify the range of values for the uniform distribution that is used to seed the prior distribution. Thus, for example **Hyperpriorall** exponential 0 10 seeds the mean of the exponential prior from a uniform on the interval 0 to 10. **Hyperpriorall** gamma 0 10 0 10 seeds the mean and variance of the gamma prior from uniform hyperpriors both on the interval 0 to 10.

The rate-parameter proposal mechanism in MCMC mode
Multistate and Discrete estimate the transition rate parameters of continuous-time Markov models of trait evolution (see above). The numerical values of these depend upon the branch lengths of the tree. In the MCMC mode, at each iteration of the Markov chain a new set of rate parameters of the model of evolution is proposed. *BayesTraits* does this by changing the current values by an amount given by the **ratedev** parameter.

To choose an appropriate value of **ratedev**, we run the Markov chain and monitor the acceptance rate of newly proposed values of the rate parameters (final column of the output). Try to choose a value of **ratedev** that produces an acceptance rate of between 20 and 40%. If the acceptance rate is too high, the model will accept nearly all changes and there will be much autocorrelation among successive states of the chain. If the rate is too low, the model will not explore the parameter space effectively.

Burn-in
The burn-in period of a MCMC run is the early part of the run while the chain is reaching convergence. It is impossible to give hard and fast rules for how many iterations to give to burn-in. We often find that a minimum of 10,000 and seldom more than 50,000 is sufficient. The length of burn-in is set with the **burnin** command. During burn-in nothing is printed out. More complex models or larger trees may require longer burn-in periods.

Sampling or Thinning
Because successive iterations of most Markov chains are autocorrelated, there is frequently nothing to be gained from printing out each line of output. Instead the chain is sampled or thinned to ensure that successive output values are roughly independent. This is the job of the **sample** command. It instructs the program only to print out every n^{th} sample of the chain. Choose this value such that the autocorrelation among successive points is low (this can be checked in most statistics programs or even Excel). For many comparative datasets, choosing every 300th or so iteration is more than adequate to achieve a low autocorrelation.

Output from the program
BayesTraits routinely writes output files named after the input data file (see below) and called inputfile.log.txt. When using maximum likelihood, the output file normally has a line of output given the maximum likelihood solutions associated with each tree in the input tree file (see below). In the

MCMC mode the program runs a large number of iterations in the Markov chain. The output file contains a line of output for each iteration you have chosen to be printed out.

The output itself normally contains columns for the iteration number, the log-likelihood, and the parameters of the model of evolution. In most cases it is too wide to be displayed on a written page so we will not print them out here, or we will show truncated versions. However, they can be viewed in Excel and the first line of the output file contains the column information.

Iterations and Stopping
The total number of iterations is selected in advance (**iterations** command). It is best to choose a large number unless you have some idea of how many iterations of the chain are required to produce adequate posterior samples of results. Whatever you choose the chain will run until it reaches the value you set or until the program is stopped manually by typing "ctrl C".

Capabilities
In the examples we give to illustrate the methods we make use of only a small number of the capabilities of the program. A full list of commands and what they do is available at the end of this manual, and a list is available from within the program by typing **Help**.

BayesMultistate

This is the information header that is printed if you choose an ML analysis in *Multistate*.

Information Header for ML analysis
```
Model:                              Multistate
Tree File Name:             treefile.trees
Data File Name:             inputdata.txt
Log File Name:              inputdata.txt.log.txt
Summary:                    False
Analysis Type:              Maximum Likelihood
ML attempts per tree:       10
No of Rates:                2
Base frequency (PI's)       None
Character Symbols           0,1
Using a covarion model:     False
Restrictions:
q01                         None
q10                         None
Tree Information
Trees:                      1
Taxa:                       15
Sites:                      2
States:                     2
```

This is the information header that is printed if you choose a MCMC analysis in *Multistate*.

Information Header for MCMC analysis
```
Model:                  Multistates
Tree File Name:             treefile.trees
Data File Name:             inputdata.txt
Log File Name:              inputdata.txt.log.txt
Summary:                    False
Analysis Type:              MCMC
Sample Period:              100
Iterations:                 5010000
Burn in:                    1000
Rate Dev:                   2.000000
No of Rates:                2
Base frequency (PI's)       None
Character Symbols           0,1
Using a covarion model:     False
Restrictions:
   q01                      None
   q10                      None
```

```
Prior Information:
   Prior Categories:         100
   q01                   uniform 0.00 100.00
   q10                   uniform 0.00 100.00
Tree Information
    Trees:                   1
    Taxa:                    15
    Sites:                   2
    States:                  2
```

Some of the items are self-explanatory. Syntax and a description of all of the commands can be found in the list of commands.

Using MultiState to estimate the model of evolution and ancestral states for a binary trait

This is an example showing how to calculate a likelihood and estimate the model of evolution using the data on primate mating systems that we reported in an earlier study (Pagel and Meade, American Naturalist, 2006). For purposes of this example, consider that primate mating systems can be classified as multimale (females mate with more than one male) or unimale/monogamous. We assign a “1” to primates with a multimale mating system and a “0” otherwise.

The example data file **MatingSystem.txt** can be used for this example, and a file of primate phylogenetic trees is in **Primates.trees**.

Using Maximum Likelihood

Start by changing into the directory that the program is in and type.

```
./BayesTraits Primates.trees MatingSystem.txt
```

when prompted for the model of evolution, select *MultiState* by typing.

```
1
```

when prompted for the analysis method, select Maximum Likelihood by typing .

```
1
```

The current state of the options should be printed to the screen.

The trait can adopt two forms. The program recognises this and forms a model of evolution with two rate parameters (q_{01} and q_{10}; see example above) allowing transitions between the two states in both directions.

Type run and the program returns the output shown below for each of the 500 trees in the sample (here the output is shown for only the first 15 trees)

```
Model:                      Multistates
Tree File Name:             Primates.trees
Data File Name:             MatingSystem.txt
Log File Name:              MatingSystem.txt.log.txt
Summary:                    False
Analysis Type:              Maximum Likelihood
ML attempt per tree:        10
No of Rates:                 2
Base frequency (PI's)       None
Character Symbols           0,1
Using a covarion model:     False
Restrictions:
   q01      01              None
   q10                      None
```

```
Tree Information
    Trees:                  500
    Taxa:                   60
    Sites:                  1
    States:                 2

Tree No     Lh          q01         q10         Root P(0)   Root P(1)
1       -26.128542  3.025912    1.855425    0.851787    0.148213
2       -25.680979  3.297484    2.040188    0.824616    0.175384
3       -25.423403  2.723676    0.370298    0.997926    0.002074
4       -23.943186  2.540748    1.358797    0.914664    0.085336
5       -25.266580  2.693711    1.121502    0.970575    0.029425
6       -25.132260  2.366744    2.641578    0.704291    0.295709
7       -24.694278  2.885220    2.132934    0.780474    0.219526
8       -25.725982  3.114022    2.247482    0.848629    0.151371
9       -26.071698  2.667690    2.854395    0.607014    0.392986
10      -27.581348  2.566995    2.770696    0.658500    0.341500
11      -28.519134  2.682646    2.624153    0.648157    0.351843
12      -25.169791  2.775758    2.516614    0.769022    0.230978
13      -24.263356  2.673320    0.867233    0.975213    0.024787
14      -26.872402  3.071314    1.538853    0.913724    0.086276
15      -27.706916  3.216948    2.371269    0.767664    0.232336
```

The output shows the tree number and its likelihood given the model, the values of the two rate coefficients and the reconstructed probabilities of the two states at the root of the tree.

In maximum likelihood there is no natural way to combine these results across trees. One might find the average likelihood or the distribution of likelihoods (or rate coefficients or root probabilities) to illustrate variation among trees, but these are not posterior distributions in the Bayesian sense. If the trees represented truly independent outcomes of evolution (say, having re-run evolution many times) then the results could be combined. By comparison, this problem vanishes in a MCMC analysis.

Testing the model. To see if transitions to multimale mating systems occur at a higher rate than transitions to unimale or monogamy, fit a model in which the two rates are constrained to be the same.

Create this restriction by typing

restrict q01 q10

These forces these two rates to be the same.

Check that the restrictions have been made by typing

info

Run the model and check the log-likelihood. A rule of thumb (see Pagel, Systs Biol., 1999) is that if this model is two or more log-likelihood units worse than the unconstrained model, then the two rate coefficients differ.

Here is the output for the first ten trees

```
Tree No Lh          q01         q10         Root P(0)   Root P(1)
1     -26.300189  2.642451    2.642451    0.739499    0.260501
2     -25.847932  2.912631    2.912631    0.713887    0.286113
3     -26.297972  2.315916    2.315916    0.858032    0.141968
4     -24.138187  2.291604    2.291604    0.782778    0.217222
5     -25.564786  2.358357    2.358357    0.836450    0.163550
6     -25.142777  2.532060    2.532060    0.728201    0.271799
7     -24.771434  2.624345    2.624345    0.696932    0.303068
8     -25.824838  2.783177    2.783177    0.780507    0.219493
9     -26.076215  2.770236    2.770236    0.626792    0.373208
```

```
10    -27.587910      2.689143        2.689143        0.674509        0.325491
```

Constraining the model has made very little difference to the likelihood, although it does influence the reconstructed root probability. This suggests that the model could be reduced to one parameter.

Reconstruct an ancestral state. BayesTraits has several approaches to reconstructing ancestral states. One is to use the '**Addnode**' command. The **Addnode** command gives the program a list of species whose common ancestor is the node you wish to reconstruct. The program then finds the proportion of the likelihood associated with each of the possible states at the node. The node must exist in the tree for the command to work. That is, the list of species must define a monophyletic group that is represented in the tree, or if a set of trees is used, the node must be present in each of them. If it is not present in a tree, the program prints out dashes for the columns in the output corresponding to the reconstructed states and then moves to the next tree in the sample.

Alternatively, you can use the **AddMRCA** or most recent common ancestor. As with **Addnode**, you give a list of species whose ancestral state you wish to reconstruct. An MRCA reconstruction finds the node in each tree in the sample that minimally contains all of the species (tips) whose common ancestral state is of interest. In any given tree the MRCA might also include other species. The attraction of MRCA's is that one exists in each tree. When you use this command, the program prints out at the top of the output how many species on average had to be included in the monophyletic group that encompasses the species of interest. If the species of interest form a monophyletic node in each tree in the sample this number will be equal to the number of species.

The command to set-up an **Addnode** reconstruction for a group of species proceeds by listing the species whose ancestral node is sought. In this example it is the list beginning with

```
AddNode Node01 51 52 50 49 47 48
```

This creates a dummy node Node01 that here corresponds to the common ancestor to the Great Apes. You can number these in order to distinguish them from one another and call the nodes anything ytouy wish.

The command to set-up a reconstruction for a MRCA of a group of species proceeds the same way

```
AddMRCA Node02 51 52 50 49 47 48
```

To get more information on the nodes type

```
info
```

The information about the two nodes to reconstruct shows that Node01 is present in 99.4% of the trees or 497 out of the 500 trees in the sample. Node02 can be defined as the MRCA with an average of 6.066 nodes below it in the trees. This means that the node containing only these six species exists in nearly every tree (497 in this case) and in the other three trees it contains an additional species.

To run the program type

```
run
```

The output should be displayed to the screen and is written to the log file. Partial output is shown below. On average the node is reconstructed to be in state 0 (a unimale/monogamous mating system) with over 90% certainty, although the degree of certainty varies from tree to tree.

```
MRCA:           Node02          6.066000
        51      Pongo_pygmaeus
        52      Pongo_pygmaeus_abelii
        50      Gorilla_gorilla
        49      Homo_sapiens
        47      Pan_paniscus
        48          Pan_troglodytes
Tree Information
    Trees:              500
    Taxa:               60
    Sites:              1
    States:             2

Tree No    Lh           Node-1 P(0)    Node-1 P(1)
1          -26.128542   0.926693       0.073307
2          -25.680979   0.926260       0.073740
3          -25.423403   0.997766       0.002234
4          -23.943186   0.942308       0.057692
5          -25.266580   0.962753       0.037247
6          -25.132260   0.696813       0.303187
7          -24.694278   0.906045       0.093955
8          -25.725982   0.826409       0.173591
9          -26.071698   0.782306       0.217694
10         -27.581348   0.778269       0.221731
11         -28.519134   0.779776       0.220224
12         -25.169791   0.814635       0.185365
13         -24.263356   0.992560       0.007440
14         -26.872402   0.936499       0.063501
15         -27.706916   0.775460       0.224540
```

Testing an ancestral state: Fossilizing a node The **Addnode** and **AddMRCA** commands allow you to see the proportion of the likelihood associated with each of the alternative states. To test whether a particular state is 'significantly' more likely at a node, you can use the fossil command. This command takes the same list of species, and fixes the node at the value you specify. For example

```
fossil Node1 0 51 52 50 49 47 48
```

This tells the program to set Node1 (arbitrary designation) to state 0. Repeating this but fixing the state to "1" gives the likelihoods associated with that state. Partial output is shown below first fixing the node to the state 0 and then to 1. Comparing likelihoods from the same trees, there is on average about two log-units' improvement when the node is fossilized to state 0, although the improvement varies from tree to tree. A difference of two log units is conventionally taken as evidence for a 'significant' difference (see Pagel, 1999 Syst Biol).

```
Log-likelihoods when node fossilised top state 0
Tree No    Lh           Node-1 P(0)    Node-1 P(1)
1          -26.158901   1.000000       0.000000
2          -25.711727   1.000000       0.000000
3          -25.423680   1.000000       0.000000
4          -23.956182   1.000000       0.000000
5          -25.271851   1.000000       0.000000
6          -25.282182   1.000000       0.000000
7          -24.732213   1.000000       0.000000
8          -25.783000   1.000000       0.000000
9          -26.228078   1.000000       0.000000
10         -27.701514   1.000000       0.000000

Log-likelihoods when node fossilised top state 1
Tree No    Lh           Node-1 P(0)    Node-1 P(1)
1          -28.574470   0.000000       1.000000
2          -28.138353   0.000000       1.000000
3          -28.429778   0.000000       1.000000
4          -26.498034   0.000000       1.000000
5          -27.879264   0.000000       1.000000
6          -26.768547   0.000000       1.000000
7          -27.158833   0.000000       1.000000
8          -27.794742   0.000000       1.000000
```

```
9       -27.709327      0.000000        1.000000
10      -29.519256      0.000000        1.000000
```

Using Markov-chain Monte Carlo

We can perform the same analyses as above far more easily and consistently using MCMC methods. In addition to running a conventional Markov chain, *BayesMultistate* implements a reversible-jump MCMC method (see Pagel and Meade, Am. Nat., 2006) that automatically finds the posterior distribution of models of evolution for the data. For a binary trait the models are easy to enumerate: there can be two distinct rates, the two rates can be the same, or one or the other rate can be zero while the remaining one is positive. In the above example, we discovered that a model with one rate parameter was virtually indistinguishable from a model with two rate parameters. The RJ-MCMC model will discover this automatically, finding the models that best explain the data. The frequency of a given model in the posterior distribution of models is the posterior belief in that 'hypothesis'.

Estimating the model. Having opened the program and chosen *MultiState*, run the RJ-MCMC by selecting MCMC from the choice of methods of analysis (method 2). A summary list will be printed out showing the default settings of the model:

```
Model:                      Multistates
Tree File Name:             Primates.trees
Data File Name:             MatingSystem.txt
Log File Name:              MatingSystem.txt.log.txt
Summary:                    False
Analysis Type:            MCMC
Sample Period:              100
Iterations:                 5010000
Burn in:                    1000
Rate Dev:                   2.000000
No of Rates:                2
Base frequency (PI's)       None
Character Symbols           0,1
Using a covarion model:     False
Restrictions:
   q01                      None
   q10                      None
Prior Information:
   Prior Categories:        100
   q01                uniform 0.00 100.00
   q10                uniform 0.00 100.00
Tree Information
    Trees:                  500
    Taxa:                   60
    Sites:                  1
    States:                 2
```

At this stage you could run the model using conventional MCMC. If you do you will see that the estimated rate coefficients are very different from those under maximum likelihood and that the acceptance rate using the uniform priors (last column of output) is far too high – acceptance rates of around 20-40% are preferred. This situation arises because the uniform priors are not restrictive enough. We are asking a lot of the data to constrain the Markov chain given that there is only one 'site' It may also have something to do with the **ratedev** parameter. The **ratedev** command specifies how big a change is proposed to the rate coefficients at each iteration of the chain. Here it is set a 2.00 (see above). Larger **ratedev** values will have lower acceptance rates, other things equal. If you re-run the analysis setting a narrow width (say 0-5, see **priorall** command) on the uniform, the output will resemble the maximum likelihood results. However, unless you have a good prior reason for making the uniform width narrower, we suggest you use a hyperprior approach.

When using the hyperprior you choose a distribution for the prior (exponential, gamma) and then the program estimates this prior from the data and using a uniform hyperprior to seed the prior (see Pagel, Meade and Barker, 2004). When rate coefficients appear to be too large under MCMC with uniform priors (say as compared to the ML results), we suggest you choose an exponential or gamma prior. The program expects you to specify the distribution for the prior and the interval of the uniform hyperprior distribution that seeds it. Typing in the RJ hyperprior command tells the program to use the reversible jump model with priors obtained from a hyperprior approach. For example, typing

```
rjhp exp 0.0 30
```

specifies an exponential prior seeded from a uniform on the interval 0 to 30. Running this gives acceptance rates in the 15-40% region and far better estimates of the rate coefficients. Alternatively a narrower interval could be chosen such as “0 10”, perhaps on the basis of having viewed the ML results. You can play with combinations of hyperprior values and **ratedev** values. Alternatively

```
rjhp gamma 0 10 0 10
```

specifies a gamma prior with its mean and variance seeded from uniform distributions on the interval 0 to 10.

Here is some partial output from running this analysis using an exponential, seeded by a uniform hyperprior on the interval of 0 – 30 using the default **ratedev**:

```
Iteration    Lh       Harmonic Mean  Model   q01          q10        Acceptance
1000   -26.770390   -26.770390    0Z   3.187412    0.000000    0.040000
1100   -23.999161   -26.395738    00   2.585303    2.585303    0.170000
1200   -25.426837   -26.030607    00   1.985240    1.985240    0.470000
1300   -26.408923   -26.031545    00   1.635713    1.635713    0.320000
1400   -25.413955   -26.031289    00   2.624669    2.624669    0.240000
1500   -26.366382   -26.025587    00   3.545666    3.545666    0.440000
1600   -23.897902   -25.988378    00   2.808191    2.808191    0.490000
1700   -22.596043   -25.857302    00   1.960398    1.960398    0.280000
1800   -23.770145   -25.742906    00   1.914063    1.914063    0.230000
1900   -24.456131   -25.655812    00   4.547834    4.547834    0.420000
2000   -26.518204   -25.687223    00   4.296446    4.296446    0.360000
2100   -25.739580   -25.744233    00   3.438055    3.438055    0.360000
```

The likelihood column shows the log-likelihoods (Lh) of successive iterations of the chain. These are comparable to those found under ML (although they will be smaller because ML is the ‘maximum likelihood’). The column labelled ‘Harmonic Mean’ is the logarithm of the running harmonic mean of the likelihoods. It will be used for hypothesis testing. The column labelled “Model” shows the number of distinct rate categories in the model of evolution. Here “00” means that both rate parameters are in the same rate category -- that is, the RJ model has discovered that a one parameter model is adequate for these data. Occasionally a 0Z appears in this column. This indicates that the second rate parameter has been set to zero. The rate coefficients are printed out and their distribution is the posterior distribution of rates for this model of evolution. The final column shows the acceptance rate of the parameter proposal mechanism. These are relatively high and suggest perhaps a larger **ratedev** value.

The posterior distribution of models directly measures the Bayesian posterior belief in which model best explains the data. The conclusion from running the MCMC analysis is that a one-parameter model is sufficient for these data. Had the results come out 50% 2 parameters and 50% 1 parameter models, then there would be no strong reason to choose one over the other.

Reconstructing an ancestral state. Start the program in the same way and select MultiState and MCMC from the options.

Add a node to be reconstructed using the **AddMRCA** command, here specifying an arbitrary node H that reconstructs the ancestral state of the Great Apes

```
AddMRCA Node-H 51 52 50 49 47 48
```

Then select a **ratedev** value and specify the hyperprior

ratedev 8

rjhp exp 0 30

Type **info** to see that the node has been accepted.

Then type

run

The output looks like this and tends to favour a "0" at the node as with ML.

```
Iteration    Lh     Harmonic Mean   Model        Node-H P(0)  Node-H P(1)         Acceptance
50000  -25.394096    -25.394096      0Z          1.000000     0.000000            0.190000
50100  -23.986422    -25.104061      0Z          1.000000     0.000000            0.180000
50200  -26.178406    -25.328024      0Z          1.000000     0.000000            0.030000
50300  -24.289847    -25.422638      00          0.918426     0.081574            0.280000
50400  -23.255129    -25.231827      00          0.799601     0.200399            0.230000
50500  -24.151832    -25.082918      00          0.945233     0.054767            0.240000
50600  -27.752323    -25.769331      00          0.888063     0.111937            0.290000
50700  -26.087429    -26.135776      00          0.633256     0.366744            0.260000
50800  -28.954741    -26.790314      0Z          1.000000     0.000000            0.290000
50900  -24.805656    -27.098071      0Z          1.000000     0.000000            0.270000
51000  -25.238036    -27.011407      00          0.842720     0.157280            0.300000
51100  -25.066133    -26.935216      00          0.889033     0.110967            0.230000
51200  -25.054813    -26.865085      '00         0.841599     0.158401            0.290000
```

To test whether there is support for one state over the other at that node use the **fossil** command in place of the AddMRCA

fossil node1 0 51 52 50 49 47 48

This sets the node to a value of "0". Let this run at least several millions of iterations and note the harmonic mean. After one such run of over 2,000,000 iterations (a few minutes of time on a fast desktop) the harmonic mean was ~ -27.0.

Repeat this run using **fossil** node1 1 51 52 50 49 47 48

After a run of over 2,000,000 iterations the harmonic mean was ~-30.5

The Bayes Factor test is just twice the difference between these two numbers or here a value of about 7. This is strong support for a "0" or unimale mating system at this node, in agreement with the ML analysis.

Note: Harmonic means can be unstable so this analysis should be repeated maybe 5 times and with very long runs (10s or 100s of million of iterations) to be sure of the result.

BayesDiscrete

Having seen how to use *BayesMultistate* to estimate models of trait evolution and to reconstruct ancestral states, this section turns to the analysis of correlated evolution among pairs of traits. The examples in this section use the tree files in **Primates.trees and PPI.trees and the data in Primates.txt and PPI.txt.** We will not present the results in as much detail as those for the Multistate examples, assuming that users can consult those example for instructions on opening and running the program, restricting parameters, creating and studying internal nodes, and choosing priors in MCMC analyses.

We will show how to use *Discrete* to perform tests of correlated evolution. The first is the analysis of correlated evolution between mating system in primates (as described above in the *Multistate* example) and whether or not female primates prominently advertise their oestrous – theory predicts that females will advertise in multimale mating systems (see Pagel and Meade, 2006).

The second example shows how *Discrete* can be used to test for functional gene links. Barker and Pagel (2005) show how a pattern of co-evolution in the presence/absence of two genes measured across species, can be used to identify pairs of genes likely to be functionally related. We use Barker and Pagel's (2005) phylogeny of the eukaryotes along with data on the presence/absence of two genes: YMR143W and YIL069C.

The gene YMR143W is a protein component of the small (40S) ribosomal subunit in eukaryotes. It is identical to Rps16Bp and has similarity to E. coli S9 and rat S16 ribosomal proteins. Gene YIL069C is also a protein component of the small (40S) ribosomal subunit, having identity to Rps24Ap and similarity to rat S24 ribosomal protein. Both being part of the ribosome they may co-evolve to maintain ribosomal shape and function.

The test of correlated evolution compares the fit of two models of evolution, one in which the two traits evolve independently on the tree, and one in which they evolve in a correlated fashion. Using maximum likelihood the models can be compared using a likelihood ratio statistic. In the MCMC mode, we show how to test for correlated evolution using Bayes Factors. Discussions of hypothesis testing can be found in Pagel (1994, 1997, 1999), Barker and Pagel (2005) and Pagel and Meade (2006).

Mating System and Oestrous Advertisement

Maximum likelihood

Open the program using

```
./BayesTraits Primates.trees Primates.txt
```

select the Dependent model and maximum likelihood analysis.

Then type **mltries** 25 and then **run**. The **mltries** simply tells the program to use more than the default number (10) of optimization attempts in finding the likelihood.

The output shows the likelihood associated with each tree, the values of the rate coefficients of the correlated evolution model, and probabilities associated with pairs of values at the root. Recall that there are now two binary traits being analyzed, and these specify four possible pairs at the root. Repeat this analysis but choosing the Independent model.

The likelihoods for the first ten or so trees look like this:

```
BayesDiscrete

Dependent Model          Independent Model
Tree No.        log-likelihood    log-likelihood
1               -33.592781        -41.179723
2                -33.108115       -41.396814
3               -34.340770        -41.697821
4               -32.790228        -41.561578
5               -34.901145        -42.776106
6               -34.065972        -41.710494
7                -32.215732       -39.984244
8               -33.668641        -41.890039
9               -33.735904        -41.806004
10              -35.023375        -43.524174
```

The differences are about 7-8 log-units per tree. Again, there is no natural way to combine these results, but they do show that there is tree to tree variability.

The likelihood ratio statistic is calculated as

LR= 2(log-likelihood(Dependent model) – log-likelihood(Independent model)). The likelihood ratio statistic is nominally distributed as a χ^2 with degrees of freedom equal to the difference in the number of parameters between the two models. Here that is four: the independent model requires two parameters per trait and the dependent model has eight parameters. A likelihood ratio test would then be 2[log difference]~14-16, and this is highly significant when assessed against a chi-squared distribution with 4 degrees of freedom.

MCMC

The analysis of correlated evolution using MCMC is straightforward: simply run a Markov chain that simultaneously samples the posterior distribution of trees contained in the tree file and the parameters of the model of evolution. The overall results are summarised by the harmonic mean.

Using a **ratedev** of 10 and a hyperprior seeding an exponential from a uniform 0-30 distribution, the MCMC run produces a number of interesting results. First, the reversible jump Discrete model reduces the full eight parameters of the Discrete dependent model to 1 or 2 rate categories on average (see Pagel and Meade, 2006 for a discussion). This is not to say that only 1 or 2 of the rates are needed, but that the eight rates fall into one or two different rate categories. The column labelled 'model string' shows which parameters are in the same rate category (same integer value) and which have been assigned to the zero (Z) category. The order of the rate coefficients in the string is the same as that in the printout.

The output also shows that nearly every model that the Markov chain visits is a Dependent model, that is, it implies correlated evolution (column with D and I in it, although there are very few Is). It is possible for the RJ model to create an Independent model even though this is a Dependent run. The scarcity of the I models shows that the data strongly support correlated evolution.

After a run of about 3 million iterations the following harmonic mean was obtained (may vary from run to run) ~ -38.86

Repeat this analysis but confining the RJ chain to Independent models. Use a **ratedev** of 8 and set **rjhp** exp 0 30. The output shows that most of the models have 1 or 2 parameters (there are four parameters now, two for each trait in the Independent model). A run of several million iterations yielded a harmonic mean of ~ -44.9.

The log-Bayes Factor test is just twice the difference of these two harmonic means, here about 12 showing very strong support for the correlation.

Functional Gene Links

Opening the program with **./BayesTraits PPI.trees PPI.txt** run a correlated evolution analysis of two genes -- YMR143W and YIL069C – both of which are involved in ribosomal functioning. We scored each gene for being presenct or absent in each of 15 eukaryotic species, based on reciprocal BLAST procedures (see Barker and Pagel, 2005). The data can be viewed in PPI.txt.

First choose the dependent or independent models of evolution under Discrete and then choose maximum likelihood as the mode of analyis.

Maximum likelihood

Set **mltries** to 25.

The likelihoods for the first ten or so trees look like this:

BayesDiscrete

```
          Dependent Model      Independent Model
Tree No.  log-likelihood       log-likelihood
```

1	-10.353423	-16.621997
2	-10.588725	-17.128112
3	-10.712233	-17.031504
4	-10.921718	-17.615133
5	-10.650607	-16.912456
6	-11.026678	-17.880038
7	-10.695006	-17.291076
8	-10.808824	-17.495943
9	-10.771289	-17.142453
10	-10.629907	-17.252674

The differences are about 7 log-units per tree. Again, there is no natural way to combine these results, but they do show that there is tree to tree variability.

The likelihood ratio statistic is calculated as

LR= 2(log-likelihood(Dependent model) – log-likelihood(Independent model)). The likelihood ratio statistic is nominally distributed as a χ^2 with degrees of freedom equal to the difference in the number of parameters between the two models. Here that is four: the independent model requires two parameters per trait and the dependent model has eight parameters. A likelihood ratio test would then be 2[log difference]~14, and this is highly significant when assessed against a chi-squared distribution with 4 degrees of freedom.

This result suggests that these two genes have co-evolved throughout the history of the eukaryotes. Interestingly, YMR143W and YIL069C are both involved in the small (40S) ribosomal subunit, and so these two genes' coevolution may have something to do with linked changes to ribosomal functioning.

Note on detecting functional gene links: We have recently shown (Barker, Meade and Pagel, *Bioinformatics*, in press) that the detection of functional gene links using our correlated evolution approach can be improved if rates of gene gain are set to a low value *a priori*, conforming to our expectation that a gene may be gained once, but that it is highly unlikely for the same gene to be gained twice.

Re-doing the maximum likelihood analyses of these two genes but **restrict**ing the two initial rate of gain parameters (q_{12} and q_{13} in the Dependent model and $alpha_1$ and $alpha_2$ in the Independent model) to 0.05, increases the likelihood ratio considerably.

It is easy to confirm this using the **restrict** command. Having chosen the Dependent analysis type

```
restrict q12 0.05

restrict q13 0.05
```

or in the Independent analysis type

```
restrict alpha1 0.05

restrict alpha2 0.05
```

Running these analyses will yield Dependent log-likelihoods in the ~ -16.5 range and Independent log-likelihoods of ~ -30. These are worse likelihoods that the unconstrained model (as they must be) but the relative difference in goodness of fit as increased sharply. The new likelihood ratio is on the order of 2(13.5) = 27 compared to the LR of about 14 for the unconstrained model.

Evidence that this restricted model provides a more accurate description of the co-evolution of these two genes comes from examining the inferred ancestral state of the two genes. In the unrestricted Independent analysis, the root probabilities for both genes are reconstructed at about 50% for each state (see last columns of output) – that is the model is uncertain as to whether the genes were present or absent ancestrally. However, looking at the raw data (in PPI.txt) it is apparent that the genes are both present in multicellular and unicellular eukaryotes, good evidence that they were both found in the common ancestor to eukaryotes. The restricted Independent model gets these two ancestral states correct, giving close to 100% support for both genes being present ancestrally. We discuss this technique further in the *Bioinformatics* paper and see below how a similar improvement can be had from within a MCMC setting.

MCMC

Opening the program with **./BayesTraits PPI.trees PPI.txt** run a correlated evolution analysis using MCMC methods of two genes whose presence and absence has been evaluated across 15 eukaryotic species (see Barker and Pagel, 2005).

First choose the dependent or independent models of evolution under Discrete and then choose MCMC as the mode of analyis.

Use a **ratedev** of 100 and a hyperprior seeding a gamma 0 10 0 10 distribution (**rjhp** gamma 0 10 0 10). This tells the program to seed the mean and the variance of the gamma prior distribution from a uniform hyperprior on the interval 0 10 for both parameters. The MCMC run produces a number of interesting results. First, the reversible jump Discrete model reduces the full eight parameters of the Discrete dependent model to 1 or 2 rate categories on average (see Pagel and Meade, 2006 for a discussion). This is not to say that only 1 or 2 of the rate coefficients are needed in the model but that the eight rates fall into only one or two different rate categories. The column labelled 'model string' shows which parameters are in the same rate category (same integer value) and which have been assigned to the zero (Z) category. The order of the rate coefficients in the string is the same as that in the printout.

The output also shows that nearly every model that the Markov chain visits is a Dependent model, that is, it implies correlated evolution (column with D and I in it, although there are very few Is). It is possible for the RJ model to create an Independent model even though this is a Dependent run. The scarcity of the I models shows that the data strongly support correlated evolution.

After a run of about 3 million iterations the following harmonic mean was obtained (may vary from run to run) ~ -17.1

Repeat this analysis but confining the RJ chain to Independent models. Use a **ratedev** of 100 and set **rjhp** gamma 0 10 0 10 again. The output shows that most of the models have 1 or 2 parameters (there are four parameters now, two for each trait in the Independent model). A run of several million iterations yielded a harmonic mean of ~ -20.1

The log-Bayes Factor test is just twice the difference of these two harmonic means. The test statistic is

2log[harmonic mean(dependent model)] – log[harmonic mean(independent model)]

This yields a log-Bayes Factor of about 6 or strong support for correlated evolution. As with the ML result, the BayesFactor test suggests that YMR143W and YIL069C have co-evolved throughout the history of the eukaryotes. The slightly weaker support for this result using MCMC may derive from the fact that the BayesFactor test integrates over uncertainty in the model and with a small phylogenetic tree such as used here, that uncertainty can be high.

We do not yet have the ability to run the restricted model in our reversible jump method. However, it is possible to use the program to run an ordinary Markov chain on these data, and using the output from the reversible jump model, set some parameters equal to each other or to zero. At the same time, the rate of gain parameters can be restricted to small values. Doing this for the PPI.txt data increases the BayesFactor to about 18, a stronger result than for the unrestricted model.

BayesContinuous

BayesContinuous can be used to test trait evolution, correlated evolution and to perform regressions for traits that vary on a continuous scale. It will perform analyses in ML and MCMC modes.

We do not currently have a manual available for the *BayesTraits* implementation of Continuous. Those wishing to use *BayesContinuous* can download the *Continuous* manual from our website (www.evolution.rdg.ac.uk). This manual was written for the Macintosh version of Continuous and gives a sense of how to use the models to test hypotheses. Users can also contact Andrew Meade (a.meade@rdg.ac.uk) for further advice on using *BayesContinuous*.

References

Pagel, M. and Meade, A. 2006. Bayesian analysis of correlated evolution of discrete characters by reversible-jump Markov chain Monte Carlo. *American Naturalist*, 167, 808-825.

Venditti, C., Meade, A. and Pagel, M. 2006. Detecting the Node-Density Artefact in Phylogeny Reconstruction. *Systematic Biology,* 55, 637-643.

Barker, D. and Pagel, M. 2005. Predicting functional gene links using phylogenetic-statistical analysis of whole genomes. *PLoS Computational Biology*, 1, 24-31. DOI: 10.1371/journal.pcbi.0010003

Pagel, M., Meade, A. and Barker, D. 2004 Bayesian estimation of ancestral character states on phylogenies. *Systematic Biology*, 53, 673-684.

Pagel, M. and Meade, A. 2004. A Phylogenetic Mixture Model for Detecting Pattern-Heterogeneity in Gene Sequence or Character-State Data. *Systematic Biology*, 53, 571-581.

Pagel, M. 1999 Inferring the historical patterns of biological evolution. *Nature* (Research Article), 401, 877-884.

Pagel, M. 1999. The maximum likelihood approach to reconstructing ancestral character states of discrete characters on phylogenies. *Systematic Biology*, 48, 612-622.

Pagel, M. 1997. Inferring evolutionary processes from phylogenies. *Zoologica Scripta* (Journal of the Royal Swedish Academy) 25th Anniversary Special Issue on Phylogenetics and Systematics, 26(4), 331-348.

Pagel, M. 1994. Detecting correlated evolution on phylogenies: a general method for the comparative analysis of discrete characters. *Proceedings of the Royal Society* (B) 255, 37-45.

Disclaimer

We have tested all of the routines in this package but cannot ensure their accuracy in all situations. Always examine results to see if they make sense in the light of previous biological knowledge of your system. We would be grateful if you would report bugs or unusual behavious in the program to us.

Appendix: List of commands in BayesTraits

Command:	AddMRCA
Purpose:	To reconstuct an internal node using the most recent common ancestor approach.
Shortcut:	mrca
Parameters:	A node name and a list of taxa names or number that define a node to reconstruct.
Example:	AddMRCA Node1 Taxa1 Taxa2 Taxa3
	mrca Node1 1 2 3 4

Command:	AddNode
Purpose:	To reconstruct an internal node.
Shortcut:	Addn
Parameters:	A node name and a list of taxa names or number that define a node to reconstruct.
Example:	AddNode Node1 Taxa1 Taxa2 Taxa3
	Addn Node1 1 2 3 4

Command:	AddTaxa
Purpose:	To add a taxa to a node to reconstruct or fossilise.
Shortcut:	AddTaxa
Parameters:	A node name and list of taxa to add.
Example:	AddTaxa Node0 Taxa1 Taxa2 Taxa3

Command:	BurnIn
Purpose:	To set the number of iterations to burn the MCMC chain in.
Shortcut:	bi
Parameters:	An integer
Example:	BurnIn 55000

Command:	Covarion
Purpose:	To turn on the covarion model.
	Can only be used with Multistate or Discrete data
Shortcut:	cv
Parameters:	None
Example:	Covarion
	cv

Command:	DelNode
Purpose:	Remove an internal node previously used to reconstruct or to fossilise.
Shortcut:	deln
Parameters:	A node name
Example:	DelNode Node1
	deln Node1

Command:	Delta
Purpose:	To estimate or set the delta parameter.
	Can only be used with continuous data.
Shortcut:	dl
Parameters:	None to estimate, or a floating point value to set
Example:	delta
	dl 0.25

Command:	DelTaxa
Purpose:	To remove taxa from a reconstructed or fossilised node.
Shortcut:	DelTaxa
Parameters:	A node name and a list of taxa names or numbers.
Example:	DelTaxa Node1 Taxa1 Taxa2

Command:	EvenRoot
Purpose:	To place the root midway between the in group and out group, in every tree.
Shortcut:	er
Parameters:	None
Example:	EvenRoot

Command:	ExcludeTaxa
Purpose:	To remove taxa form the analyses.
Shortcut:	Et
Parameters:	A list of taxa names or taxa numbers to remove.
Example:	ExcludeTaxa Taxa1 Taxa2 Taxa3 et 1 2 3

Command:	Exit
Purpose:	To quite the program
Shortcut:	Quit
Parameters:	None
Example:	Exit Quit

Command:	Fossil
Purpose:	To fix an internal node to a given state. Can only be used with Multistate or Discrete data.
Shortcut:	fo
Parameters:	A node name, a state to fix and an internal node defined by a list of taxa names or numbers.
Example:	Fossil Node-A 0 Taxa1 Taxa2 Taxa3 Fo Node-B 1 1 2 3

Command:	Help
Purpose:	To display a list of commands.
Shortcut:	He
Parameters:	None
Example:	Help

Command:	Hyperprior
Purpose:	To set a prior in which the parameters are drawn form a uniform range. Can only be used with Multistate or Discrete data.
Shortcut:	hp
Parameters:	A rate name, prior distribution and minimum and maximum for each parameter needed to describe the distribution.
Example:	Hyperprior q01 exp 0 100 Hp q10 gamma 0 10 0 30

Command:	HyperPriorAll
Purpose:	To set the same hyper prior for all rates. Can only be used with Multistate or Discrete data.
Shortcut:	hpall
Parameters:	A prior distribution and minimum and maximum for each parameter needed to describe the distribution.
Example:	HyperPriorAll exp 0 10 Hpall gamma 0 10 0 30

Command:	Info
Purpose:	To show the current setting for the analysis.
Shortcut:	In
Parameters:	None
Example:	Info

Command:	Iterations

Purpose:	To set the number of iterations to run the MCMC chain.
Shortcut:	it
Parameters:	The number of iterations to run the chain for.
Example:	Iterations 8000000
	it 1000000

Command:	Kappa
Purpose:	To estimate or set the kappa parameter.
Shortcut:	ka
Parameters:	None to estimate, or a floating point value to set
Example:	kappa
	ka 0.5

Command:	Lambda
Purpose:	To estimate or set the lambda parameter. Can only be used with continuous data.
Shortcut:	la
Parameters:	None to estimate, or a floating point value to set.
Example:	lambda
	la 0.75

Command:	LogFile
Purpose:	To set the name for the log file. By default it is the name of the data file with ".log.txt" appended to it.
Shortcut:	lf
Parameters:	A file name for the log.
Example:	LogFile logFile1.txt

Command:	Mltries
Purpose:	To set the number of times to run the optimisation algorithm. If results vary from run to run. This parameter should be increased form default of 10.
Shortcut:	mlt
Parameters:	An integer
Example:	Mltries 15
	mlt 30

Command:	Pis
Purpose:	To set the frequencies of the states. States frequencies can be estimated, uniform, empirical or none.
Shortcut:	Pi
Parameters:	uni, emp, est or none
Example:	Pis Uni
	Pis Est

Command:	Prior
Purpose:	To set the prior for a parameter.
Shortcut:	pr
Parameters:	A rate name, distribution (gamma, exp, uniform beta) and a list of parameters that define the distribution.
Example:	Prior q01 exp 10
	Pr q10 gamma 10 10

Command:	PriorAll
Purpose:	To set a single prior for all parameters.
Shortcut:	Pa
Parameters:	A distribution (gamma, exp, uniform beta) and a list of parameters that define the distribution.
Example:	PriorAll gamma 10 20
	pa uniform 0 30

Command:	RateDev
Purpose:	To set the deviation of the normal distribution, that changes to the rates are drawn form. This should be set so a the acceptance of the rate parameters is roughly 20%
Shortcut:	rd
Parameters:	A floating point number
Example:	RateDev 0.1
	rd 14.5

Command:	Restrict
Purpose:	To reduce the number of parameters required in the model. By setting two or more rates equal to each other.
Shortcut:	res
Parameters:	The rates to be set equal to each other or the rates to be set equal to a constant.
Example:	Restrict q01 q12 q32
	Res q01 q12 q32 1.75

Command:	RestrictAll
Purpose:	To restrict all rates to a single estimated parameter or a constant
Shortcut:	Resall
Parameters:	A single rate or a constant
Example:	RestrictAll q01
	Resall 0.8

Command:	RevJump
Purpose:	To toggle the use of reverse jump for model selection. Can only be used with Multistate or Discrete data
Shortcut:	rj
Parameters:	A prior and its parameter to use for all rates.
Example:	RevJump exp 10
	rj gamma 10 20

Command:	RevJumpHP
Purpose:	To toggle the use of reverse jump for model selection, using a hyper prior.
Shortcut:	rjhp
Parameters:	A prior distribution and minimum and maximum for each parameter needed to describe the distribution.
Example:	RevJumpHP gamma 0 10 0 10
	rjhp exp 0 10

Command:	Run
Purpose:	To start the analysis running.
Shortcut:	Ru
Parameters:	None
Example:	Run

Command:	Sample
Purpose:	To specify the sample period in MCMC
Shortcut:	sa
Parameters:	An integer
Example:	Sample 500

Command:	SaveTrees
Purpose:	To save the tree to a tree file. Any excluded taxa will be removed.
Shortcut:	st
Parameters:	A tree file name.
Example:	SaveTrees NewTrees.tree
	st NewTrees.trees

Command:	TaxaInfo

Purpose:	To list taxa numbers and names
Shortcut:	ti
Parameters:	None
Example:	TaxaInfo

Command:	TestCorrel
Purpose:	To toggle the test for correlation. Can only be used with continuous data.
Shortcut:	Correlation
Parameters:	None
Example:	TestCorrel TC

Command:	UnRestrict
Purpose:	To remove a restriction form a rate or set of rates.
Shortcut:	Unres
Parameters:	A rate or set of rates to un restrict
Example:	UnRestrict q01 q10 q32 Unres q32