Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

TraitLab: A MATLAB Package for Fitting and Simulating Binary Tree-like Data

Geoff K. Nicholls
University of Oxford

Robin J. Ryder
Université Paris-Dauphine

David Welch
University of Auckland

Abstract

TraitLab is a software package for simulating, fitting and analysing tree-like binary data under a stochastic Dollo model of evolution. The model also allows for "catastrophes", evolutionary events where many traits are simultaneously lost while new ones arise. The core of the package is a Markov chain Monte Carlo (MCMC) sampling algorithm that enables the user to sample from the Bayesian joint posterior distributions for tree topologies, clade and root ages, and the trait loss and catastrophe rates for a given data set. Data can be simulated according to the fitted Dollo model or according to a number of generalized models that allow for borrowing (horizontal transfer) of traits, heterogeneity in the trait loss rate and biases in the data collection process. Both the raw data and the output of MCMC runs can be inspected using a number of useful graphical and analytical tools provided within the package. TraitLab is freely available and runs within the MATLAB computing environment.

Keywords: Bayesian inference, dating methods, Markov chain Monte Carlo, phylogenetics, binary data, trait data, historical linguistics.

1. Introduction

The ability to fit phylogenetic models to genetic sequence data has become central to many areas of the biological sciences thanks in part to a number of complex software tools that aid this process. By contrast, binary trait data, which represent the absence or presence of distinguishing characterisitcs in individuals, have received relatively little attention and there are few publicly available tools for fitting such data.

TraitLab, the software we describe here, was developed specifically to fit binary trait data under a Dollo model of evolution (Dollo 1893) in which any trait shared among individuals is assumed to have descended from the same evolutionary innovation. Examples of data to which this model can be or has been applied are morphological traits ('has wings', 'has opposable thumbs'), cultural traits ('uses curvilinear designs in woodcarving', 'makes pottery') or lexical traits ('uses word with Old Saxon root *al* to mean "all" ', 'uses word with Latin root *totus* to mean "all" '). The Dollo assumption insists that such complex traits arise only once in the evolutionary history of the set of taxa being studied so that, for example, for a set of taxa including birds and insects, 'has wings' would not be a valid trait as insect and bird wings evolved independently, but could be replaced by the valid traits 'has bird wings' and 'has insect wings'. The basic Dollo model assumes that the data can be fully described by a dated tree representing the evolutionary relationships between taxa and parameters for the birth and death rates of traits. An extended Dollo model, also implemented in TraitLab, allows so-called catastrophes to occur in which multiple traits are born and die simultaneously, representing rapid evolutionary bursts. This allows for heterogeneity in the rate of trait evolution along different lineages. The extended model introduces two further parameters, being the rate at which catastrophes occur and probability of trait death at a catastrophe.

TraitLab fits the model within a Bayesian framework using Markov chain Monte Carlo (MCMC) techniques to draw samples from the posterior distribution of the parameters for given data. Any or all of the parameters – the dated binary tree, the trait birth and death rates, the catastrophe occurrence rate and the trait death rate – can be estimated or fixed. The program is controlled via a simple graphical user interface in the MATLAB computing environment.

The data we consider consist of N traits that have been recorded as present, absent or missing (presence or absence undetermined) in L taxa. The data D are therefore an $L \times N$ binary matrix where the (i, j)th entry $D_{ij} = 1$ if the jth trait is present in the ith taxon, $D_{ij} = 0$ if it is absent and $D_{ij} = ?$ if its status is undetermined. In addition, clade constraints that place constraints on the topology and ages of certain parts of the tree can also be handled by TraitLab and are discussed in Section 6.2.

The remainder of the paper is organised as follows. In Section 2 we describe the trait evolution and data collection models. Section 3 gives an outline of the likelihood and posterior calculations, including the form of the prior distributions, and Section 4 discusses the construction of the MCMC algorithm for sampling from the posterior. Directions on how to install and start TraitLab are given in Section 5. Section 6 specifies the data file format. Section 7 gives step-by-step instructions for running an MCMC analysis while Section 8 describes the tools provided for analysing and visualising the output of the MCMC run. Finally, in Section 9, we describe how data can be simulated under the Dollo model and various extensions within TraitLab and how this can be used to check for model fit and model mis-specification.

Further details on the model, its implementation and its application to linguistic data can be found in Nicholls and Gray (2008), Ryder and Nicholls (2011) and Ryder (2010).

2. A stochastic Dollo model of evolution

In the basic Dollo model of trait evolution, three events may occur: traits are born, traits die and traits are duplicated when the lineages containing them split. We assume that all events are independent and waiting times between events are exponentially distributed with the stated rates. In each lineage, traits are born at a constant rate λ so if l lineages are being followed at time t, trait births occur at total rate $l\lambda$ at t. Each trait in each lineage dies at constant rate μ so if there are k_i traits in lineage $i \in \{1, \ldots, l\}$ at time t, the total trait death rate in all lineages is $\mu \sum_{i=1}^{l} k_i$ at t. Finally, lineages split at constant rate θ. The total lineage splitting rate when there are l lineages is $l\theta$. When lineage i splits at time t, it is replaced by two exact copies of itself. That is, at time t^+, immediately after time t, two new lineages, j and k, say, are created which possess exactly the same traits as lineage i possessed at time t^-, immediately before t. After time t, lineage i no longer exists and lineages j and k continue to evolve independently as all other lineages. We say that lineage i is the parent of lineages j and k (the children of i).

Catastrophes occur independently along each lineage at rate ρ. At a catastrophe, each trait dies with probability κ and Poisson(ν) new traits are born. We impose the condition that $\frac{\lambda}{\mu} = \frac{\nu}{\kappa}$ to ensure that the expected number of traits in any lineage remains constant over time and that the process is reversible.

2.1. Observation model

The observed traits are not chosen uniformly at random from all possible traits, thus the observation model needs to be carefully modelled so the likelihood of the data can be accurately defined. Typically, traits which are not observed at any taxon (either because the trait died out, or because of missing data) do not make it into the data set. Furthermore, the traits may be chosen so that traits observed at only one taxon are also omitted. This has the effect of thinning the trait evolution process described above, so that those traits that evolve, survive and are observed at zero or just one taxon will not be registered in the observation process. Further variations on this registration process are discussed in Ryder and Nicholls (2011) but are not currently implemented in TraitLab.

Secondly, some data may be missing from the matrix where it has not been possible to record the presence or absence of a particular trait in a taxon. We model each trait as missing uniformly at random within each taxon, so that each trait is missing independently from taxon i with probability ξ_i (so is recorded with probability $1 - \xi_i$).

3. Likelihood, prior and posterior calculations

Here we give a brief outline of the form of the posterior distribution for this problem. A full description of the posterior, including details of an efficient recursion for the likelihood calculation, can be found in Ryder and Nicholls (2011).

First we need to introduce some notation. Let g be a binary rooted tree with L leaves, $L - 1$ internal nodes and a single "Adam" node, labelled $2L$, that is adjacent to the root

node of g and is infinitely old. Let $V = \{1, \ldots, 2L\}$ be the set of node labels. The ith node of g has age t_i and $t = (t_1, \ldots, t_{2L})$ where we set $t_{2L} = \infty$ and ages increase towards the root of the tree. The directed edges of g are pairs of nodes (i, j) such that $t_i < t_j$. Let $E = \{(i, j)|(i, j)$ is an edge in $g\}$ so $|E| = 2L - 1$, as it consists of the $2L - 2$ internal edges of g and the edge of infinite length connecting the Adam and root nodes. The tree is thus defined by $g = (V, E, t)$. A catastrophe has the same effect as lengthening the edge it occurs on by some fixed block of time (the size of which depends on κ and ν). It is therefore convenient to extend the definition of a tree to include the number of catastrophes that occur on each edge. Let k_i be the number of catastrophes that occur on edge (i, j) and $k = (k_1, \ldots, k_{2L-2})$ be the vector of catastrophe counts for all finite edges. The extended tree is defined by $g = (V, E, t, k)$. Denote by $[g]$ the set of all points (τ, i), i.e., time τ on edge i, on the tree g.

Let $z_D = \{z_1, \ldots, z_N\}$, $z_a \in [g]$, be the locations of the birth events of the N observed traits. Recall that only traits which are observed at at least d taxa (with $d = 1$ or $d = 2$) are registered in the data. Augment the data matrix D to include all traits whether registered or not, so including those traits that survive in no taxa or in just one taxon. Denote this augmented matrix D^* with dimensions $L \times N^*$. Let Z, a random point in $[g]$, be the birth time of some (possibly unobserved) trait. Let $\mathcal{E}_Z$ be the event that this trait makes it into the observed data. Then the Poisson point process of birth locations of observed traits has intensity

$$\tilde{\lambda}(z) = \lambda \Pr(\mathcal{E}_Z | g, \mu, \kappa, \xi, Z = z)$$

at $z \in [g]$ and probability density

$$f_{Z_D}(z_D) = \frac{1}{N!} e^{-\Lambda([g])} \prod_{a=1}^{N} \tilde{\lambda}(z_a)$$

with respect to the element of volume $dz_D = dz_1 dz_2 ... dz_N$ in $[g]^N$, where

$$\begin{aligned} \Lambda([g]) &= \int_{[g]} \tilde{\lambda}(z) dz \\ &= \sum_{(i,j) \in E} \int_{t_i}^{t_j} \tilde{\lambda}((\tau, i)) d\tau. \end{aligned}$$

The distribution of the number N of registered traits is $N \sim \text{Poisson}(\Lambda([g]))$.

Let I be a matrix of indicator functions corresponding to the missing elements of that extended data, so $I_{a,i} = 1$ if and only if $D^*_{a,i} = ?$. For a matrix Y, let Y_a denote the ath column of Y. Then, following the notation of Ryder and Nicholls (2011), the likelihood is

$$\mathsf{P}[\mathbf{D} = D | g, \mu, \lambda, \kappa, \xi, \mathbf{D} = R(\tilde{\mathbf{D}})] = \frac{e^{-\Lambda([g])}}{N!} \prod_{a=1}^{N} \Big(\prod_{i=1}^{L} \xi_i^{I_{i,a}} (1 - \xi_i)^{1 - I_{i,a}} \Big) \lambda \int_{[g]} \sum_{d^* \in \mathcal{D}_a} \mathsf{P}[\mathbf{D}^*_a = d^* | g, \mu, \kappa, \xi, Z_a = z_a] d \tag{1}$$

The difficulty of calculating this function is in the calculation of the two integrals, $\Lambda([g])$ and the integral involving $\mathsf{P}[\mathbf{D}_a^* = d^*|g, \mu, \xi, Z_a = z_a]$. Both integrals can be calculated efficiently using a recursion over the tree that is similar to Felsenstein's pruning algorithm (Felsenstein 1981). The interested reader is referred to Ryder and Nicholls (2011) for a detailed discussion.

Our prior on the birth rate λ, death rate μ and catastrophe rate ρ is $p(\lambda, \mu, \rho) \propto \frac{1}{\lambda\mu\rho}$, where μ and ρ also have very conservative bounds placed on them to ensure that the posterior is proper. We take a uniform prior over $[0, 1]$ for the death probability at a catastrophe κ and each missing data parameter ξ_i. The prior on the tree g is chosen in such a way that the induced prior on the root age of the tree is approximately uniform over the interval $[t, T]$, where t is the maximum of the lower bounds imposed by the clade constraints (or zero of there are no clade constraints) and T is maximum root age imposed by the user. Nicholls and Gray (2008) show that the prior distribution with density

$$f_G(g|T) \propto \prod_{i \in S} (t_r - t_i^-)^{-1}$$

satisfies this criterion, where S is the set of all nodes with clade constaint bounds on their maximum age falling at or beyond T. When clade constraints are placed on the tree, as described in Section 6.2, the prior is defined to be zero for any tree not satisfying those constraints.

We multiply the expression for the likelihood in Equation (1) by the priors to obtain the posterior density

$$p(g, \mu, \lambda, \kappa, \rho, \xi|\mathbf{D} = D) \propto \frac{1}{\lambda\mu\rho} f_G(g|T) \mathsf{P}[\mathbf{D} = D|g, \mu, \lambda, \kappa, \xi, \mathbf{D} = R(\tilde{\mathbf{D}})] \tag{2}$$

which is valid for parameters $\mu, \lambda, \kappa, \rho > 0$, $0 \leq \xi_i \leq 1$ and trees g satisfying the clade constraints and with root age less than T.

4. MCMC algorithm

We sample from the posterior distribution using MCMC. Given our choice of prior for λ, we are able to integrate the posterior analytically with respect to λ to obtain the target distribution $p(x|D)$ where $x = ((E, V, t, k), \mu, \kappa, \rho, \xi)$. Thus a state of the Markov chain is some value for each component of x. Brief descriptions of the different mechanisms used to propose new states in the chain are listed in Table 1. There are four proposals that alter the tree topology (moves 2–5 in Table 1), which are described in Drummond, Nicholls, Rodrigo, and Solomon (2002), five proposals that alter the heights of some or all nodes in the tree (moves 1, 6, 7, 11 and 12), proposals that add, delete or shift catastrophes (moves 13, 14 and 18, respectively, described in Ryder and Nicholls (2011)) while the remaining five moves (8, 15, 16, 19, 20) multiply one or more of the scalar parameters μ, κ, ρ and $\xi = (\xi_1, \ldots, \xi_L)$ by a randomly chosen factor.

Move	Description
1	vary internal vertex time
2	interchange neigbouring edges
3	interchange randomly chosen edges
4	move edge to new neigbouring location
5	move edge to new random location
6	rescale whole tree
7	rescale random subtree
8	rescale μ
9	(unused)
10	(unused)
11	random walk on leaf time
12	vary root time
13	add a catastrophe
14	delete a catastrophe
15	scale ρ
16	rescale κ
17	(unused)
18	move catastrophe to adjacent edge
19	rescale a single ξ_i
20	rescale all elements of ξ

Table 1: The moves used in the MCMC sampler to explore the state space. Moves 9, 10 and 17 were used in earlier versions of TraitLab but are no longer used.

5. Installing and running TraitLab

5.1. System requirements

TraitLab is written in the MATLAB programming language. Thus, to run TraitLab, MATLAB (version 6, release 12.1 or later) must already be installed on your system. MATLAB is proprietary software which runs on Windows, Mac OS X or Unix/Linux machines. TraitLab only requires the basic MATLAB installation without any additional packages. For more information on MATLAB, see `www.mathworks.com` or ask your local mathematics, statistics or engineering department who may be able to help.

5.2. Download and installation

Download the TraitLab zip archive from `https://sites.google.com/site/traitlab/`

and extract all files, preserving the subdirectories, to `C:\TraitLab` (or any other convenient location - if using another directory, use that name instead of `C:\TraitLab` in the following.)

5.3. Running TraitLab

Start MATLAB in the usual manner and make `C:\TraitLab` the current directory.

To start TraitLab and bring up the main GUI, type `TraitLab` at the MATLAB command line.

6. Data file format

TraitLab accepts data in the Nexus file format which is standard in phylogenetic analysis (see Maddison, Swofford, and Maddison (1997) for a full description of the format). The binary data matrix is recorded in the `Data` block, while any clade constraints are recorded in the `Clades` block, which is specific to TraitLab. In the following, we describe the structure of these blocks and provide an example of a data file in Section 6.4.

6.1. Data block

The first command in the `Data` block must be the `Dimensions` command with values for `ntax` (the number of taxa) and `nchar` (the number of characters or traits) defined.

The `Format` command where the missing character is defined is optional. If it is not present, the missing character is assumed to be '?'. The gap character, '-' by default, may also be defined here but is not used in TraitLab. All gaps will be reclassified as missing.

The `Matrix` command is compulsory. Rows of the matrix are labelled with the taxa names, which may not contain any whitespace. The content of the matrix must be zeros and ones to indicate absence or presence of the trait. The matrix may be in standard format (where

the rows are `nchar` characters long) or in interleaved format (where the matrix is split in sections of a manageable size). If the matrix is interleaved, each section of the matrix must have the rows labelled by the taxa names and sections must be separated by blank lines. Comments may only occur between interleaved sections of the matrix — comments at the start or end of rows will cause errors.

The `Charlabels` command, followed by a list of exactly `nchar` trait names may also appear in the `Data` block. All other commands in the `Data` block are ignored.

6.2. Clades block

Prior knowledge about the structure of the tree and divergence times can be encoded in the `clades` block. The `clades` block consists of a series of clade commands, one for each known clade. It has the following form:

```
BEGIN Clades;

Clade Name = Clade_1
Taxa = taxon_a,...,taxon_k
Rootmin = t1
Rootmax = t2
Originatemin = t3
Originatemax = t4;

Clade Name = Clade_2 ...

End;
```

Each `clade` command must include a name and the list of taxa that define the clade. The time of the most recent common ancestor (the root) of the clade can be bounded by defining `rootmin` and `rootmax`. The time that clade diverged from all other taxa (the node above the root node) can be bounded by defining `originatemin` and `originatemax`. See Figure 1 for an example of the difference between root and originate bounds. All time bounds on a clade are optional.

Offset leaves

If the taxa are sampled at significantly different times, the clades block is used to encode this information. If a clade has only one taxon, then the rootmax and rootmin definitions for that clade are upper and lower bounds for the sampling time of that taxon. Time zero is defined as the time of the most recently sampled taxon. The upper and lower time bounds must not be the same, so if a taxon was known to have been sampled 500 years before the most recently sampled taxon, set rootmax to be 501 and rootmin to be 499.

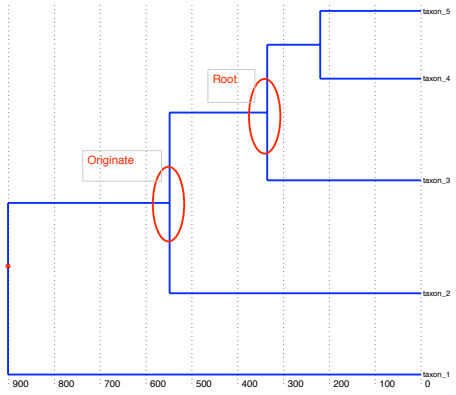

Figure 1: Suppose a clade consists of `taxon_3, taxon_4` and `taxon_5`. If `rootmin`= t_1 and `rootmax`= t_2, then the node labelled root here is constrained to lie in the interval (t_1, t_2). Similarly if `originatemin`= t_3 and `originatemax`= t_3, the node labelled originate is constrained to lie in the interval (t_3, t_4). Note that one "child" node of the originate node is necessarily the root node of the clade in question, but the other child node is arbitrary (here, it is the leaf node of `taxon_2`).

6.3. Other blocks

When data are synthesized using TraitLab, a `trees` block and a `synthesize` block are generated. The `trees` block contains the tree on which the data were synthesized and the `synthesize` block contains the parameter values used for the synthesis. If either of these blocks is found, TraitLab will assume that the data are synthetic.

The `characters` block which may contain taxa names and/or trait labels can be read by TraitLab but we advise against its use. Include any relevant information in the `data` block instead. All other blocks are ignored by TraitLab.

6.4. Example data file

The simple data file shown below shows the basic structure of a TraitLab data file. It has the required `data` block and a `clades` block. The `data` block specifies that there are 9 taxa and 30 traits and gives the data matrix. The `clades` block specifies two clades, one with two taxa and with maximum and minimum age bounds on the root, the other with three taxa and a lower bound on the time it split from the rest of the tree.

```
#NEXUS

BEGIN DATA;

DIMENSIONS NTAX=9 NCHAR=30;
FORMAT MISSING=? GAP=-  INTERLEAVE ;
```

```
MATRIX

taxon_1  00?1111010110101000101001?0000
taxon_2  101111010?1100111100101101l000
taxon_3  01101101??1110111?001010011000
taxon_4  0101111001110111001100000100100
taxon_5  1101111011110l1??0110100100100
taxon_6  1111110101l1?01111001011011011
taxon_7  1111???101101011110??011011011
taxon_8  1111110001111011100010l1011011
taxon_9  11111101?1111011110010l1?11011
;
END;

BEGIN CLADES;

CLADE  NAME = Clade_1
ROOTMIN = 346
ROOTMAX = 422
TAXA = taxon_4, taxon_5;

CLADE  NAME = Clade_2
ORIGINATEMIN = 346
TAXA = taxon_1, taxon_8, taxon_9;

END;
```

7. Running an MCMC analysis

MCMC analyses are set up and run in TraitLab from the main GUI shown in Figure 2. There are four types of information that the user must specify before starting a run: the data file, the initial state of the chain (which may be random), the model (including which parameters to estimate and priors) and the run parameters including run length and output file location. These four types of information correspond to the four colours of the panels in the GUI and are each explained in detail in Sections 7.1–7.4 below. In Section 7.5, we describe how to start MCMC runs without using the GUI, which is useful for performing long runs or multiple runs in the background.

7.1. Data source

Specify the data source by clicking the "select data file" button and choose a nexus file

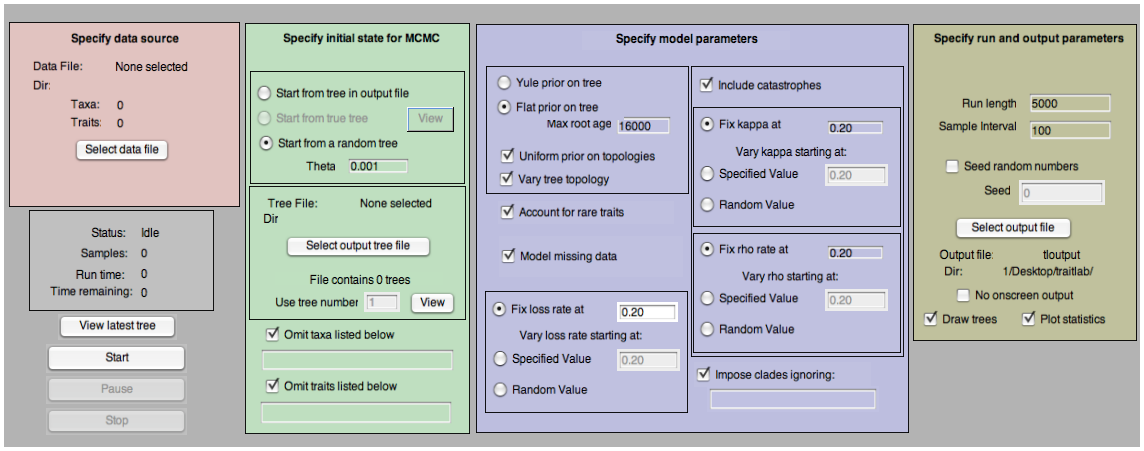

Figure 2: The main TraitLab GUI.

conforming to the TraitLab requirements described in Section 6. Progress in reading the file can be monitored in the MATLAB command window.

If the file is successfully read, the number of taxa and traits will be shown above the "select data file" button. These numbers should agree with the respective numbers of rows and columns in the matrix in the data file. If any clades were found in the file, the number found is shown below the "select data file" button. If the data were synthesized using TraitLab, this will also be indicated below the "select data file" button.

The names of the taxa and any clades found in the data file will appear in the MATLAB command window. The numbers that appear next to the trait and clade names are the order in which they appear in the file and are used to specify taxa and clades to be ignored (see Sections 7.1.1 and 7.3.1).

Omitting taxa and traits

Any of the taxa and traits can be omitted from a run so long as at least two taxa remain to be analysed. Omit taxa by checking the "omit taxa listed below" checkbox and listing taxa numbers in the space provided. A taxon's number corresponds to its row number in the data matrix in the data file. They are also printed out to the MATLAB command window after the data file is read in. Omitting taxa that appear in clade constraints can cause problems; see Section 7.3.1 for more details.

Similarly, traits are omitted by checking the appropriate check box and listing a vector of trait numbers. Trait numbers correspond to columns of the data matrix as they appear in the data file. It is left to the user to find the correct column numbers for traits to be masked.

The lists of numbers must satisfy MATLAB formatting. That is, numbers must be separated with commas or spaces. Sequences of consecutive numbers can be abbreviated

using the the colon operator (for example, abbreviate 2,3,4,5 as 2:5). Thus, to omit taxa 1,10,11,12,13 and 20, type "1 10:13 20" in the space provided.

7.2. Initial tree

The initial tree may be random, be chosen from a previous TraitLab run or, if the data were synthesized within TraitLab, be the true tree on which the data were synthesized.

If random initial tree option is selected, the initial tree is a Yule tree (i.e., a tree with exponentially distributed branch lengths) with specified branching rate, θ (so that the mean branch length is $1/\theta$). If clades are imposed, the random tree is constructed using a heuristic method to ensure that all constraints are satisfied.

To start from a tree that appeared in a previous run, the output file where the tree is to be found needs to be specified using the "select output tree file" button. Once the file has been selected, a tree in the file is specified by the "use tree number" text box. Use the adjacent "view" button to view the specified tree. The trees in the file must have taxa names which are a superset of the taxa names in the current run.

Note that if the initial tree is chosen from an output file, the user should ensure that the chosen tree satisfies all constraints to be imposed in the new run including the maximum root age and any clade constraints. Specifying a tree that does not satisfy all the necessary constraints will result in all proposed states being rejected and a failed run.

7.3. Model specification

The options in this section allow the user to decide how the data are to be modelled. The following can all be included in or excluded from the fitted model: the catastrophe process, the modelling of missing data and the observation process that accounts for the fact that rare traits found in only a single taxon are typically not observed. In addition, the user must specify here which of the model parameters, g, μ, κ, ρ and ξ are to be estimated and what the initial value should be. The form of the prior on the tree needs to be specified and any clade constraints found in the data file can be imposed here.

Radio buttons are used to determine the prior imposed on the tree. We recommend the default prior which induces a uniform marginal prior distribution on the root age as described in Section 3. If the user selects this option, an upper bound on the root age, T, must be specified in the "max root age" box. If clades are imposed, the specified value of T must be greater than the largest of the `rootmin` and `originatemin` bounds imposed. The other option is a Yule prior in which trees are given prior weight according to branch lengths which have an exponential penalty. If the "vary tree topology" box is unchecked, the tree will remain in same shape throughout the run with only the branch lengths varying.

The "account for rare traits" checkbox determines whether or not the likelihood takes into account an observation process in which traits that only occur in a single lineage are recorded. Check the box in the case in which the data collection process discards traits observed at only one taxon, as described in Section 2.1. In this case, if the data matrix

contains any traits that are observed in just a single taxon, those columns of the matrix are automatically removed before the analysis begins. When the box is unchecked, the likelihood is calculated as if all traits observed in at least one taxon have been recorded. Note that, for real data sets, it is unusual for traits present in only one taxon to be recorded; if in doubt, it is thus recommended that the checkbox remain checked.

If the data matrix includes traits coded as missing (i.e., it includes "?" characters), the "model missing data" box should be checked. If it is unchecked, all missing traits will be recoded as absent and ξ parameters in the likelihood will be fixed at zero.

The trait death rate, μ, defined in Section 2 is an estimable parameter when clades with time constraints are imposed. Instead of working directly with μ, we reparametrize it as a loss rate, ψ, defined by

$$\psi = 1 - \exp(-1000\mu).$$

Thus ψ is a number between 0 and 1 and can be interpreted as the mean proportion of traits that are lost in a lineage over a period of 1000 years. The user then has the option of fixing ψ at a specified value or letting this value vary over the run in order to estimate it. In the latter case, the user must either specify a starting value or choose to start from a random value. These three options are determined by selecting the appropriate radio button.

If the data are thought to include rate heterogeneity, the catastrophe process may be included in the model by checking the "Include catastrophes" checkbox. In this case, there are two further parameters to consider, the rate at which catastrophes occur, ρ, and the probability of trait death at a catastrophe, κ. As with ψ, they can either be fixed at a specified value, or estimated with specified or random initial value by selecting the appropriate radio button.

Imposing clades

If clade constraints are present in the data file, they may be applied during a run by checking the "impose clades" checkbox. To omit clades from the analysis, list the clades to be omitted in the space provided using the respective clade numbers (see Section 7.1.1 for details on clade numbers and list formatting). It is also possible to ignore the age ranges of certain clades (as if `rootmin` $= 1$ and `rootmax` $= \infty$) by checking the box "Ignore ages for clades" and listing the clade numbers in the space provided.

Note that difficulties can arise when omitting taxa that appear in imposed clades. For example, consider clade C that consists of taxa x, y and z with a root time between t_1 and t_2. If taxon z is omitted from the analysis, it is often incorrect to simply omit it from clade C to get the new clade C' consisting of just taxa x and y with the same root time constraints as C. This is because x and y could have a common ancestor at time t where $t < t_1$. The user is given the option, in the MATLAB command window, of simply deleting the offending taxon from the clades in question (as was done to get from clade C to clade C' in the example) or leaving it and causing an error which can be remedied by adding the offending clades to the clades to ignore list. To delete the taxon from the clade, type "1" at the command line when given the option "delete taxa/ignore? 1/0". Be

aware that deleting taxa from clades in this way can have unintended consequences: the new clade may be inconsistent with other clades or, when all but one taxon are deleted from a clade, the clade root time constraint is reinterpreted as an offset leaf (see Section 6.2.1). Furthermore, this can lead to errors in the MCMC estimation. If an error occurs after ignoring some taxa, we suggest adding the problematic clades to the list of clades to ignore.

If problems are encountered omitting taxa that occur in clade constraints, the cleanest solution is to create another data file containing the same data block but a clade block that has been altered to achieve the desired constraints with the offending taxa removed. This is the recommended course of action when the run is started in batch mode (see Section 7.5) as the process described above requires user interaction.

7.4. Run and output parameters

The last panel that needs to be completed relates to the length of the run, the location of the output files and the required level of interactive monitoring of the run.

Specify the total length of the MCMC run, r, and sub-sample interval, j. The initial state of the chain will be saved to the output files as will every jth state thereafter. Thus the total number of sampled states that are saved is the integer part of $(r/j) + 1$ samples. We are unable to give a priori guidance as to how long a particular run needs to be as it depends heavily on the particular dataset in question. Generally, the greater the number of taxa, the longer the run needs to be. Exploratory runs should be made to check convergence for the parameters of interest and it is sensible to make at least one very long run to check for any unexpected mixing behaviour in the chain. In some cases, it may take many days or weeks to obtain a satisfactory number of samples for larger problems. The subsample rate is chosen so that the output files are of a managable size. Choose a number so that the total number of saved sampled states is somewhere in the range 1000-10000. Checking a chain for convergence is discussed further in Section 8.

The "seed random numbers" checkbox is mainly used for debugging purposes and is generally left unchecked. If it is checked, a seed for the random number generator needs to be specified (it can be any real number). Separate runs with all options the same including the random seed will be identical. If the checkbox is unchecked, a random seed will be generated based on the date and time.

Output files

Output files are saved in the location specified using the "select output file" button. Supposing `tlout` is the chosen file name, five output files are created: `tlout.nex`, `tloutcat.nex`, `tlout.txt`, `tloutXI.txt` and `tlout.par`.

The `tlout.par` contains a record of all parameter values and options used to create the run. It can be used to perform similar runs in batch mode (see Section 7.5).

Each sampled state is recorded in the `tlout.nex` and `tlout.txt` files. The sampled trees are in the `tlout.nex` file, information on the catastrophes is in the `tloutcat.nex`, and

sampled scalar values (parameters and densities) are in the `tlout.txt` file. The recorded parameter values are the root time for the sampled tree, the trait death rate μ, the cladistic loss probability p (currently not estimated so always equals 1), the catastrophe rate ρ, the death probability at a catastrophe κ and the birth rate λ. Note that λ is integrated analytically rather than using the MCMC sampler so that the value recorded here is an independent draw from the marginal posterior distribution of λ at the given state. Also in the `.txt` file are values of the log prior density for the state, the integrated log likelihood (where λ has been integrated out) and the log likelihood (using the value of λ recorded). Finally, the ξ parameters related to missing data are stored in the `tloutXI.txt` file.

Monitoring a run

The progress of the run can be monitored via values output to the MATLAB command window, a plot of the most recently sampled tree and trace plots of the parameter and log density values. There is also a status box on the left of the GUI showing the number of samples already obtained, the run time to date and an estimate of the time remaining.

To plot each tree as it is sampled, check the "draw trees" checkbox. If the box is unchecked, the most recently sampled tree can be plotted by clicking the "view latest tree" button on the left of the GUI.

To see the trace plots of the sampled root time, trait loss rate, log prior density and integrated log likelihood density, check the "plot statistics" checkbox. Note that the log likelihood trace plot is truncated to omit the early sampled values so that the vertical scale of the plot is reasonable. If the data were synthesized using TraitLab, horizontal lines showing the "true" parameter values for the data are shown on the trace plots.

Unless the "no onscreen output" checkbox is checked, when each sample is taken, a row of numbers will appear in the MATLAB command window. These numbers show the sample number, the log likelihood value and several statistics showing the proportion of proposed MCMC updates accepted since the previous sample. For example, the line

```
(5,-3550.486804) 0.33 0.24 0.09 0.14 0.03 0.21 0.38 . . .
```

means that sample 5 has log likelihood of around -3550 and, of the updates between sample 4 and sample 5, 0.33 of the states proposed by move type 1 were accepted, 0.24 of those proposed by move type 2 were accepted and so on. The ordering of the moves is the same as that given in Table 1. Note that, depending on the options chosen, some moves may not be proposed so the proportion accepted will be `NaN`.

7.5. Running TraitLab in batch mode

TraitLab can be run without the GUI interface directly from a Unix, Linux, Mac or MATLAB command line. Instead of using the GUI to specify all parameters for a run, they are stored in the file called `batchtlinput.txt` in the main TraitLab directory. This has exactly the same form as a `.par` output file which is automatically created at the start of every run and has options corresponding to those in the main TraitLab GUI. Either

modify `batchtlinput.txt` field by field, or take an existing `.par` file, modify as necessary and save as `batchtlinput.txt`. It may be easiest, if many similar runs are being made, to set up a short model run using the GUI and then use the `.par` file as a template for the other runs in batch mode.

To start a run in batch mode from a command shell, make the main TraitLab directory the current working directory and type

```
\proglang{MATLAB} -nodisplay < batchTraitLab.m > outlog &
```

at the command line. Any onscreen output will be dumped to outlog while the normal `.nex`, `.txt` and `.par` output files will be created in the location specified in `batchtlinput.txt`. For long runs, consider adding `nohup` as a prefix to the above command.

To run the batch version from the MATLAB command line, make the main TraitLab directory the current directory and type `batchTraitLab`.

8. Analysing output and inspecting the data

The results of MCMC runs and the raw data can be inspected in the analysis GUI, shown in Figure 3. The tools provided for exploring the output of MCMC runs include viewing sampled trees that are saved in a `.nex` output file, making trace plots and histograms of sampled statistics from a `.txt` output file and making histograms of the root time of user specified clades through the run. These functions are described in Section 8.2. Note that many of these functions are fairly generic and can equally well be performed, along with other functions not available in TraitLab, by programs such as Tracer (Rambaut and Drummond 2009).

The data inspection tools provided, described in Section 8.3, allow the user to make histograms of the number of traits per taxon and of the number of taxa per trait, comparisons with synthetic data and plots based on a maximum *a posteriori* estimator of the time of the most recent common ancestor for a pair of taxa. All of the plots made in TraitLab can be saved or printed as one would a standard MATLAB figure.

Access the analysis GUI by choosing "analyse output" in the mode menu from the main TraitLab GUI.

8.1. Loading data and output to the analysis GUI

When the analysis GUI is called, the output from the most recent completed run (if there is one) along with the currently loaded data is automatically loaded into the analysis GUI. If no run is in memory, nothing is passed to the analysis GUI and data and output can be loaded using the "load output" and "load data" buttons. Note that it is left to the user to ensure that the loaded output and data are compatible.

8.2. Exploring MCMC output

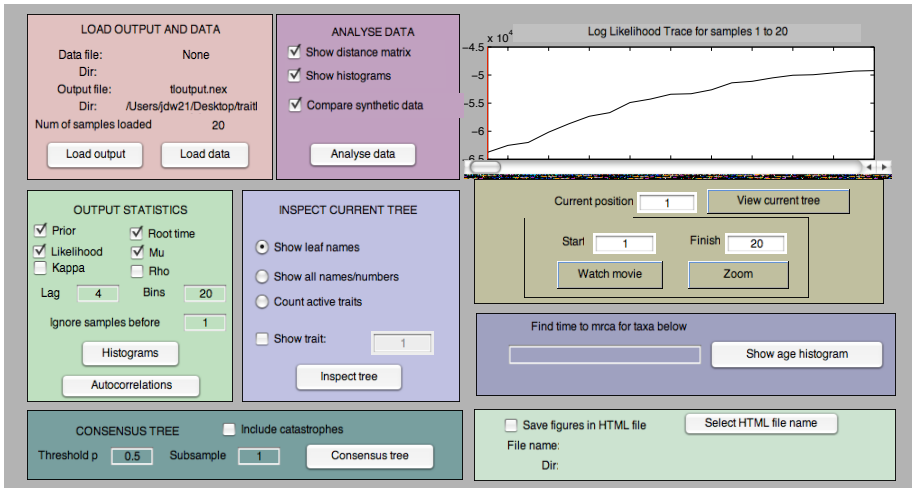

Figure 3: The analysis GUI.

Autocorrelations, trace plots and histograms

The "output statistics" box provides functions for calculating and plotting histograms, lag autocorrelation functions and trace plots of the parameters and the log density functions. Visual inspection of the trace plot of a parameter of interest is one of the best ways of determining whether or not the chain is mixing well and has been run long enough for the samples to be useful for scientific purposes. These plots can be made for any of the log prior density, the log likelihood density, the tree root time, the catastrophe occurrence rate, ρ, the probability of trait death at a catastrophe, κ, or the trait death rate, μ, stored in the `.txt` output file. Choose which statistics to plot using the appropriate checkboxes. Similar plots can be drawn for the age of internal nodes; see Section 8.2.3.

To plot the autocorrelation functions, it is necessary to specify the maximum lag for which autocorrelations are calculated in the "lag" text box. The maximum value of the lag should be significantly smaller than the number of samples being analysed. To ignore the first part of the run as burn-in, specify the first sample to use in the "ignore samples before" text box. Trace plots of the selected statistics are automatically drawn alongside the lag autocorrelation plots. The title of the lag autocorrelation plot has the variable name, the integrated autocorrelation time, τ_f, the number M, an estimate of the number of samples after which the normalized autocovariance function is approximately zero and N, the number of samples analysed. A rough estimate of the effective sample size obtained for statistic f is N/τ_f. An example of an autocorrelation plot is shown in Figure 4. For further information about how to interpret autocorrelation plots and associated statistics, see Nicholls (1998) or Geyer (1992).

To plot histograms of the sampled statistics, specify the number of bins to use in the histogram. A burn-in can be specified using the "ignore samples before" text box. The histograms of model parameters represent their respective estimated marginal posterior

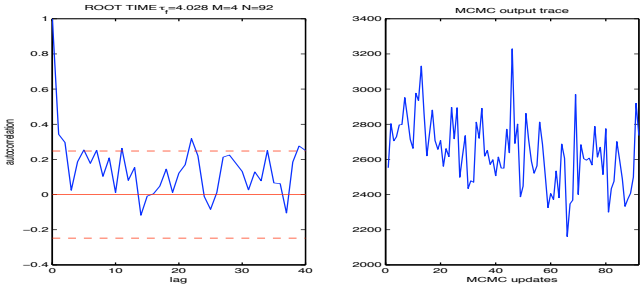

Figure 4: The lag autocorrelation function of the variable (root time, in this case) is plotted on the left. On the right is the trace plot of the variable.

distributions.

Viewing sampled trees

At the top right of the GUI is a trace plot of the log likelihood of sampled states in the loaded run. Trees from the loaded run can be viewed using the "view current tree" button and (when data are also loaded) more closely investigated using the "inspect current tree" functions. The current tree is the tree at the current position. The current position is shown as a red line on the log likelihood plot and can be specified either in the "current position" text box or by using the slider below the trace plot. To zoom in on a section of the trace plot, specify the first and last samples of the section in the "start" and "finish" text boxes and click the "zoom" button.

The sampled trees between "start" and "finish" can be watched as a movie by clicking the "watch movie" button. The movie shows approximately 3 samples per second and can be halted by closing the window in which it shows.

When data are loaded, the "inspect tree" function can be used to closely inspect sampled trees. If the "show leaf names" radio button is selected, the standard plot of the current tree is drawn when the "inspect tree" button is clicked. The "show all names/numbers" option is mainly used for debugging and shows the node numbering used to internally represent the tree. The numbers on the leaf nodes shown in this representation are those used to specify taxa for the TMRCA histograms (see Section 8.2.3).

The "count active traits" option shows the standard tree plot of the current state where, at each internal node of the tree, the number of traits found exclusively in the clade defined by that node is shown. For example, suppose that the current tree has three taxa a, b and c in which (a, b) form a clade and there are 20 traits found in a and b that are not found in c. Then the number 20 will be shown at the internal node (a, b). Note that the number shown at the root of the tree is always the total number of traits in the data set being analysed (after rare traits have been stripped from the raw data).

By checking the "show trait" checkbox and specifying a trait number, the path of the specified trait on the current tree can be viewed. According to the standard Dollo model,

the trait must have been present on the lineages shown in black, so must have been born either above the root of the tree or on the lineages shown in red. It was either not present or died at some point on one of the lineages in blue. An example of a trait history is shown in Figure 5.

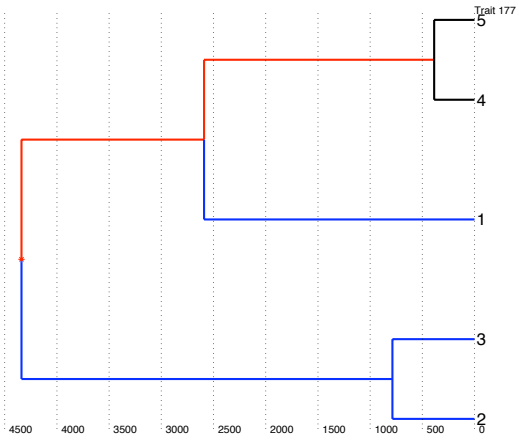

Figure 5: An example of the use of the "show trait" function for inspecting a tree. The specified trait here is found in taxa 4 and 5, so must have been present on the lineages shown in black and could have arisen on any of the lineages shown in red (or above the root).

Histogram of TMRCA for given taxa

The marginal posterior distribution of the time to the most recent common ancestor (TMRCA) for any group of taxa in the analysis can be estimated using the "show age histogram" button. Specify the taxa of interest by using their respective numbers in the text box provided. The taxa numbers can be found using the "inspect tree" function (see Section 8.2.2). The first part of the run can be ignored as burn-in using the "ignore samples before" text box and the number of bins to use in the histogram is specified in the "bins" text box, both found in the output statistics section of the GUI.

When the "show age histogram" button is clicked, the TMRCA for the given taxa in each sampled tree is found and this information is output to the MATLAB command window (copy it to a text file to analyse it elsewhere) as well as being made into a histogram. The posterior mean, the 95% highest probability density (HPD) interval and the posterior probability that the given taxa form a clade are all output to the MATLAB command window. Note that the computation of the HPD interval assumes that the marginal posterior distibution of the TMRCA is unimodal; this assumption can easily be checked by inspecting the histogram and is usually verified in practice.

8.3. Inspecting the data

Clicking the "analyse data" button will produce a number of plots which are described below. Note that both data and output need to be loaded to view these plots since a tree is required for the construction of the the distance-depth plot.

Data histograms

If the "show histograms" checkbox is checked, histograms of the number of taxa per trait and the number of traits per taxon will be generated. The taxa per trait histogram is a histogram made from the column sums of the data matrix. If there are k taxa and L traits in total, each trait occurs in between 0 and k taxa, thus the horizontal axis in the graph runs from 0 to k and the vertical axis (the frequency) runs from 0 to (at most) L.

The traits per taxon histogram is a histogram made from the row sums of the data matrix. It is the empirical distribution of the number of traits found in each taxon.

If the "compare synthetic data" checkbox is checked, synthetic data according to the standard stochastic Dollo model are generated on the current tree and histograms of the synthetic data are displayed below those for the loaded data.

If the model fits the data well and the tree is a representative draw from the posterior distribution, the two pairs of histograms should be qualitatively very similar. Consistent qualitative differences between the two pairs of histograms may indicate some significant model misspecification.

Distance depth and distance matrix plots

When all traits, including those found only in a single taxon, are observed it is possible to analytically calculate the maximum a posteriori estimate of the time to the most recent common ancestor for a pair of taxa under the standard Dollo model (see equation 10 of Nicholls and Gray (2008)). The distance matrix plot (check the "show distance matrix" checkbox) and the distance depth relation plot (which is always shown when the "analyse data" button is pressed) are based on this calculation. Note that we assume that rare traits (traits that occur in only one taxon) are not observed, so, in order to make an accurate estimate of the TMRCA, rare traits are synthesized and blended with the data in the file to make these plots.

The distance matrix graph is simply a heat plot of a matrix where this distance has been calculated for each pair. Bluer colours represent closer taxa, red more distant and green intermediate. It is best understood by referring to the example in Figure 6.

For the depth distance relation plot, the maximum a posteriori estimate for the TRMCA for each pair is calculated and the time of the MRCA of each pair in the specified tree is found. The two times for each pair are then plotted against each other. See Figure 7 for an example. The tree used is the current output tree (or, when the graph appears after synthesizing data, the tree on which the data were synthesized). If the standard Dollo model and the specified tree fit the data well, the points should lie along the line $y = x$. Systematic deviation of the points away from the line $y = x$ may indicate some significant model misspecification.

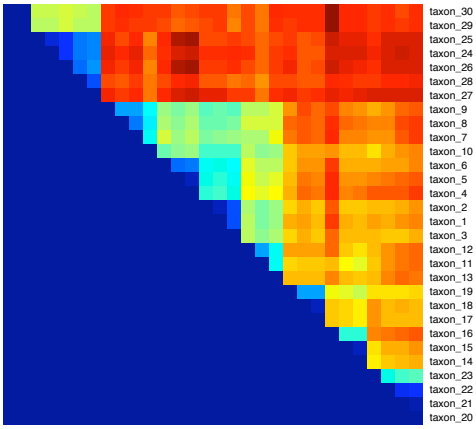

Figure 6: An example of the distance matrix graph. The column labels are the same as the row labels. The pixel at row i, column j represents the maximum a posteriori estimate of the TMRCA between taxon i and taxon j. The darker the blue the closer the taxa are, the darker the red the further apart they are. Lighter and green colours are intermediate. Note that the ordering of taxa is as it occurs in the data file, so the obvious clustering that is shown here will not occur if rows in the data file are randomly ordered.

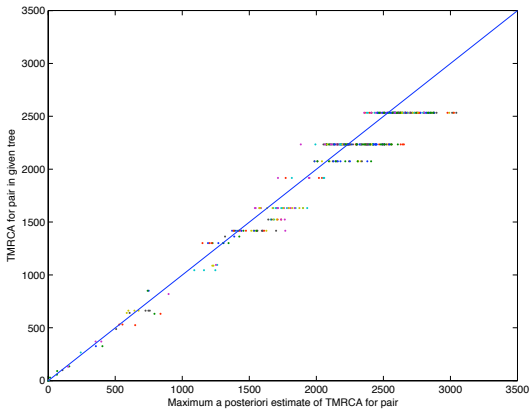

Figure 7: An example of the depth distance relation graph. Each point represents a pair of taxa. The position of the point along the x-axis is the maximum a posteriori estimate of time to the most common ancestor for the given taxa pair, while the position along the y-axis is given by the time of the most recent common ancestor of the pair in the given tree. The line shown in blue is $y = x$. The points are horizontally stratified as a single ancestral node in the tree may be the common ancestor of many pairs of taxa. The example here show a very good fit of data and model.

8.4. Building a consensus tree

A consensus tree is a convenient way of summarizing a sample of trees. A consensus tree shows all splits with at least $p = 50\%$ support in the posterior sample. If a split receives between 50% and 95% support, it is labelled; splits receiving at least 95% support are unlabelled. If no split receives more than 50% support, the tree is multifurcating. The threshold p can be changed to another value, but values lower than 50% will lead to errors. The integer parameter *subsample* can be set to a value greater than 1 to subsample the trees; this will speed up the computation of the consensus tree.

On a consensus tree, the length of a branch displayed is the average length of that branch in the trees of the posterior sample where it is present. Similarly, the number of catastrophes displayed on a branch is the average number of catastrophes on that branch in the trees of the posterior sample where the branch is present, rounded to the nearest integer.

8.5. Saving as HTML

To ease the sharing of results, the GUI offers an option to save your analysis as an HTML file. Tick the box "Save figures in HTML file", then either select the name of your HTML file or let TraitLab choose one for you, based on the date and time. So long as the checkbox is selected, all figures you produce from the Analysis GUI are saved as `.bmp` files, and an HTML file is created with those figures and some explanatory text. Note however that figures created before ticking the box are not included, nor is the "movie of trees" described in Section 8.2.2.

9. Synthesizing data and model checking

TraitLab allows the user to simulate trait and clade data under any of the fitted models, i.e., the Dollo model with or without catastrophes and with or without missing data, and also allows simulation under various model extensions. The model extensions include borrowing (lateral transfer of traits between lineages), heterogeneity in the trait death rate, μ, across edges in the tree and heterogeneity in μ across different groups of traits. In this section, we provide step by step instructions on how to synthesize data in TraitLab, which involves specifying a tree, a model of trait evolution and a clade model. We give full descriptions of each of the model extensions as we go along.

To get started with the synthesize GUI, choose "synthesize data" in the mode menu from the main TraitLab GUI. The GUI is shown in Figure 8.

9.1. Selecting a tree

The tree along which traits evolve can either be a randomly generated Yule tree (with exponential branch lengths) or any rooted tree (with branch lengths specified) stored in the Newick format in a nexus file. Select the appropriate radio button to choose between a random tree and a stored tree. If a random Yule tree is chosen, the branching rate, θ,

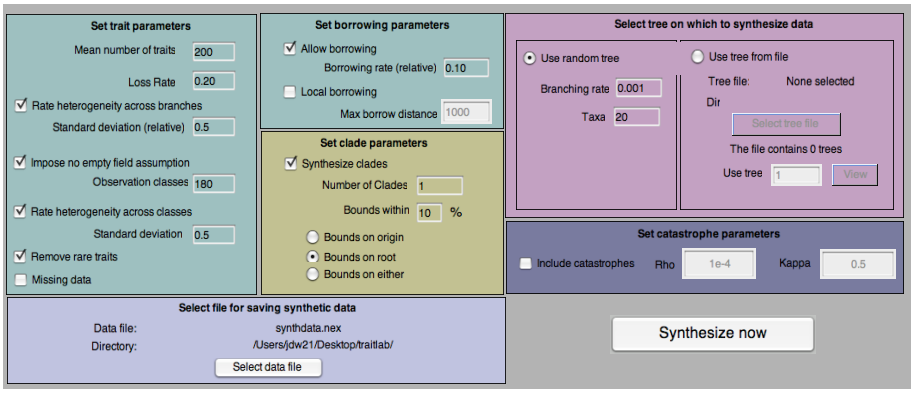

Figure 8: The data synthesizing GUI.

and the number of taxa need to be specified. The mean branch length in a Yule tree is $1/\theta$ years.

To synthesize data on a tree from a file, click the "Select tree file" button. TraitLab will attempt to extract all readable trees from the specified file. If there is more than one tree found in the file, specify which tree to use in the "use tree" text box. Check that you have selected the correct tree with the "view" button. If there are trees in a file but TraitLab is unable to read them, check that they are written in the correct format with a root and branch lengths specified.

9.2. Defining the trait evolution model

All models are based on the basic Dollo model of evolution, which is completely specified on a given tree by defining the trait birth and death rates. Equivalently, we parametrize the model using the mean number of traits per taxon, K, and the trait loss rate, $\psi = 1 - \exp(-1000\mu)$. These must specified in the text boxes provided. The loss rate, ψ, is a number between 0 and 1 and can be interpreted as the mean proportion of traits lost in a lineage over a period of 1000 years. Given the mean number of traits per taxon, K, and a value of μ, the per taxon trait birth rate, λ, is defined by $\lambda = K\mu$.

Catastrophes

Catastrophes are included in the simulation when the "include catastrophes" checkbox is checked. In this case, the catastrophe occurrence rate, $\rho > 0$, and the probability of trait death at a catastrophe, $0 < \kappa < 1$, must be specified in their respective text boxes. The trait birth rate at a catastrophe, ν, is automatically calculated so that that reversibility condition, $\nu/\kappa = \lambda/\mu$, is respected.

Rate heterogeneity and the no-empty-field assumption

Another form of rate heterogeneity, different from catastrophes, is to let the trait death rate, μ, vary across each lineage of the tree so that for each lineage, $1 \leq i \leq L-1$, there is a specific death rate μ_i. μ_i is a gamma distributed random variable with mean μ and variance $(\sigma_b \mu)^2$. It is necessary to specify the relative standard deviation parameter, σ_b. Check the "rate heterogeneity across branches" checkbox to include this in the model.

Checking the "impose no empty field assumption" checkbox will produce data as if traits are collected based on pre-specified observation classes for which it is assumed every observed taxon will have at least one trait in each class. This is called the no-empty-field assumption as we assume that each observation class is occupied by at least one trait in each taxon at all times. This is a natural assumption with lexical data where linguists set out to collect words from different languages associated with a list of meanings. The meanings here are the observation classes, so the assumption is that every language will have a word for these basic meanings.

If this option is checked, it is necessary to specify the number of observation classes. Note that the option will have the greatest effect on the simulated data when the number of observation classes is close to the mean number of traits per taxon, K, and that K is a natural upper bound for the number of observation classes.

If the no-empty-field assumption is imposed, heterogeneity in trait death rates across observation classes may be simulated by checking the "rate heterogeneity across classes" checkbox and specifying a relative standard deviation, σ_c in the associated "standard deviation" text box. Traits in observation class j then have a death rate $\mu_c(j) \sim \Gamma(\text{mean} = \mu, \text{variance} = (\sigma_c \mu)^2)$.

If the no-empty-field assumption is imposed and the "Missing data" box is checked, data will go missing "in blocks": within an observation class and for a given taxon, either all traits are observed, or all are missing. The interested reader is referred to Section 4.2.2 of Ryder (2010).

When rates vary across both branches and observation classes, a trait on branch i in observation class j dies at rate $\mu_{b,c}(i,j) \sim \Gamma(\text{mean} = \mu_b(i), \text{variance} = (\sigma_c \mu_b(i))^2)$ where $\mu_b(i) \sim \Gamma(\text{mean} = \mu, \text{variance} = (\sigma_c \mu_b)^2)$.

Borrowing

Borrowing, or horizontal transfer of traits, is observed in lexical, cultural and genetic trait data. We allow users to simulate data under two models of borrowing — global or local borrowing. Global borrowing is when all lineages are equally likely to borrow from one another, whereas local borrowing means that only lineages that are close, in the sense that they share a recent common ancestor within some specified time period, may borrow traits from each other.

Global borrowing is synthesized by checking the "allow borrowing" checkbox and specifying a relative borrowing rate, b. Each trait is borrowed at constant rate $b\mu$. When a trait, k, is borrowed, it chooses a lineage, i, uniformly at random from those existant. If trait k is

not present in lineage i, k is added to i, otherwise nothing happens.

To make borrowing local instead of global, check the "local borrowing" checkbox and specify a borrowing distance, d. All traits are still borrowed at constant rate $b\mu$ but the target lineage is not chosen from all others. Instead, only lineages that share a common ancestral lineage with the source lineage (the one in which the trait to be borrowed is found) in the last d years may borrow the trait. The target lineage is chosen uniformly at random from all such lineages.

When death rates are heterogeneous and borrowing occurs, a trait on branch i in observation class j is borrowed at rate $b\mu_{b,c}(i,j)$.

9.3. Missing data and rare traits

Once trait data have been simulated down a tree according to the specified model, the observation model is simulated so that some traits may be marked missing and rare traits are ignored. If the "include missing data" checkbox is checked, a parameter $\xi_i \sim \beta(1, 1/3)$ is drawn for each leaf i. At leaf i, each entry in the simulated matrix is then made missing with probability $1 - \xi_i$. Columns of the matrix with no 1's (so traits not observed at any taxa) are removed. If the "remove rare traits" checkbox is checked, traits that are observed at only one taxon are also discarded. In many real data sets, such traits are not observed.

9.4. Synthesizing clades

Check the "synthesize clades" checkbox to synthesize clades. When there are L taxa in the synthetic data, up to $L - 1$ clades can be synthesized corresponding to the $L - 1$ internal nodes of the tree. It is necessary to specify the accuracy of the clade bounds, c, in the "bounds within" text box. An accuracy of c means that, if the chosen node has a time t in the tree, a lower bound of $(1 - c/100)t$ and an upper bound of $(1 + c/100)t$ will be created.

By clicking on the appropriate radio buttons, the bounds can be on the root, on the divergence time (originate bounds) or be chosen uniformly at random between the two. In real data, the bounds are most commonly on the root.

9.5. Output of synthesize GUI

The synthetic data will be saved to the `.nex` file specified by the user by clicking the "select data file" button, say `synfile.nex`. This file will contain the synthetic trait matrix, the tree on which the data were synthesized and associated model parameters. A `.par` file, `synfile.par`, will also be generated for record keeping purposes. All options and parameter values used to produce the data as specified in the GUI are recorded here. Checkbox options are represented by 0 for unchecked and 1 for checked.

When the "synthesize now" button is pressed, the data will be synthesized according to the specified parameters and options. Progress can be monitored in the MATLAB command window. A figure showing the tree on which the data were synthesized will appear.

If you check the box "Explore data after synthesizing" diagnostic tools are run on the

synthetic data. An explanation of how to interpret the histograms and the depth distance graph that are generated is given in Section 8.3 above.

10. Example: Analysis of Semitic lexical data

As an example, we provide in this section the steps used to analyse a real dataset. The data concern core vocabulary of various Semitic languages and were collected and made public by Kitchen, Ehret, Assefa, and Mulligan (2009). An analysis of these data with TraitLab appeared first in Nicholls and Ryder (2011), with more detail. The dataset is included in TraitLab in the Nexus file `semitic09.nex` and can thus be used to discover TraitLab's functionalities.

10.1. Data

The data file is in Nexus format. The DATA block includes 674 traits for 25 taxa. Each taxon is a Semitic language (Gehez, Tigre, Amharic...) and each trait is a meaning category; the meaning categories cover 96 meanings from the core vocabulary. In this file, some data points are listed as "gaps" (- in the data file); these are treated as missing by TraitLab. The beginning of the data block is:

```
#NEXUS

BEGIN DATA;
DIMENSIONS  NTAX=25 NCHAR=674;
FORMAT DATATYPE=RESTRICTION MISSING=? GAP=- INTERLEAVE=yes;

MATRIX

Gehez           1000100000001000000000010000000001000000000
Tigre           1000010000000100000000001000000001000000000
Tigrinya        1000001000001000000000001000000001000000000
Amharic         1000100000001000000000----------01000000000
Argobba         0100100000001000000000000100000000100000000
Harari          1000100000001000000000000010000000010000000
Zway            1000100000000010000000----------00010000000
Walani          1000100000000001000000000001000000010000000
Gafat           0010------------------0000010000000001000000
Soddo           1000100000000100000000000100000000000100000
Mesqan          1000100000000000100000000010000000000010000
Innemor         1000100000000000001000000100000000000001000
Mesmes          0001100000000000010000001000000000000000100
Geto            1000100000000000001000000010000000000000010
Chaha           1000100000000000001000000010000000000000010
```

```
Hebrew          1000000100000000000100000000100000000000001
Ugaritic        1000000010000000000100001000000000000000001
Aramaic         1000000001000100000000000100000000000000000
Akkadian        1000000001000000000100000100000000000000000
MoroccanArabic  1000000000100000000010000100000000000000000
OgadenArabic    1000000000100000000000100000100000000000000
Mehri           1000000000010000000010000000001000000000000
Jibbali         1000000000010000000001000000000100000000000
Harsusi         0001000000100000000010000000001000000000000
Soqotri         0001000000100100000000000000000100000000000
;
END;
```

The data file also includes a CLADES block, with 6 clades. In this case, each clade comprises only one language and thus gives information on one of the leaves of the tree. The clades block is:

```
BEGIN CLADES;

CLADE  NAME = hebrew
ROOTMIN =2500 ROOTMAX =2700
ORIGINATEMIN = 3200 ORIGINATEMAX = 4200
TAXA = Hebrew ;

CLADE  NAME = ugaritic
ROOTMIN =3300 ROOTMAX =3500
ORIGINATEMIN = 3400 ORIGINATEMAX = 4400
TAXA = Ugaritic ;

CLADE  NAME = aramaic
ROOTMIN =1700 ROOTMAX =1900
ORIGINATEMIN = 2850 ORIGINATEMAX = 3850
TAXA = Aramaic ;

CLADE  NAME = amharic
ORIGINATEMIN = 700 ORIGINATEMAX = 1700
TAXA = Amharic ;

CLADE  NAME = gehez
ROOTMIN =1600 ROOTMAX =1800
TAXA = Gehez ;

CLADE  NAME = akkadian
ROOTMIN =2700 ROOTMAX =2900
```

```
TAXA = Akkadian  ;

END;
```

meaning for example that the sample of Hebrew in the data corresponds to a language spoken between 2500 and 2700 Before Present, and that we know that the branch leading to Hebrew split off from its closest cousins between 3200 and 4200 Before Present.

Loading the data

Launch MATLAB, change the current folder to `C://traitlab`, and type TraitLab in the command window to start TraitLab. In the pink "Specify data source" panel, click "Select data" and navigate to the `semitic09.nex` file.

In the main MATLAB window, we are given the list of clades, the list of languages, as well as the list of languages with more than 5% missing data.

For now, we can keep all the parameter values unchanged and click the "Start" button to initiate the MCMC run. By default, TraitLab will use a run length of 5000 iterations, subsampling every 100 iterations, thus giving a sample of size 50. On a 2011 iMac under Kubuntu with an Intel i5 2.70GHz CPU and 8 GB RAM using MATLAB R2012a, this run takes about 80 seconds.

Extra information is given in the main MATLAB window. First, the data are prepared. By default, all traits which are present at only 0 or 1 taxon are removed from the data, leaving 334 traits. During the run, the log-likelihood of each sample is displayed alongside the acceptance ratios for each move, and a figure of the current tree is plotted, as well as traces of several key statistics, as shown in Figure 10.1.1. It is immediately obvious that the MCMC has not converged, so we lauch a longer run: 5 million iterations, subsampling every 5000 iterations, giving a sample size of 1000. On the same computer as previously, this run takes about 14 hours.

At first glance, this second run seems satisfactory, given the key statistic traces. We move to the "Analyse output" mode, to verify that the chain has indeed reached stationarity and mixed well.

In the "Analyse output" mode, the data and run are preloaded. The root time is plotted over the entire run; we zoom in to display only iterations 100 to 1001. The plot indicates that a burn-in of 100 is satisfactory. In the green "Output statistics panel", we check the traces and autocorrelations of the prior, root time, log-likelihood, as well as the parameters μ, κ and ρ, ignoring the samples before iteration 100. These plots are also satisfactory. Finally, we check the traces and autocorrelations of a few internal and leaf ages. To do this, first use the "Inspect current tree" panel with the option "Show all names/numbers". A tree is displayed, with the numbers used by TraitLab to represent each node internally. These numbers can then be used to "show the age histogram" of internal node ages. For example, we check the age of the leaf Hebrew and the age of the internal node above Tigre and Tigrinya by entering the relevant node numbers. Along with the age histogram, we

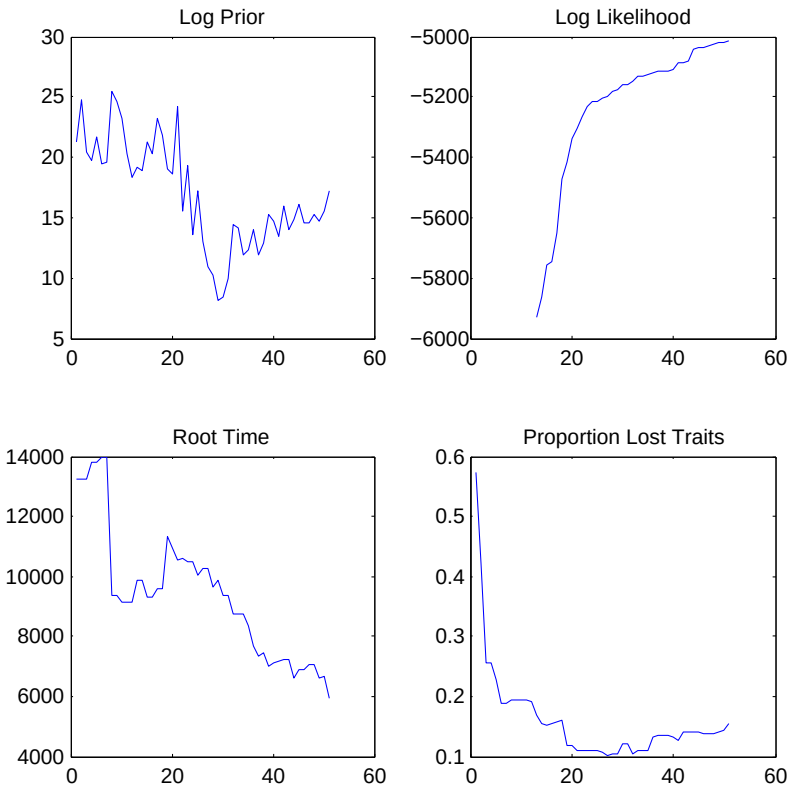

Figure 9: Trace of key statistics after first MCMC run for the Semitic data. These plots (for example the log-likelihood, top right) show that the MCMC has not yet reached stationarity.

are also given the MCMC trace and autocorrelations for that node age, and the trace and autocorrelations for the indicator that the languages given form a subtree (this gives an indication of the MCMC behaviour of the tree topology). Summary statistics are also given in the main MATLAB window.

Since the MCMC run has converged, we can now check for model misfit. Using the "Analyse data" panel, we observe that points on the "Distance Depth Relation" plot are close to the line $y = x$, and the histograms for the data in the file resemble those for synthetic data, indicating reasonable overall fit.

The consensus tree with catastrophes shows strong support for catastrophe events on the branches above Ugaritic, with almost no rate heterogeneity elsewhere in the tree; there is a 33% posterior probability for no catastrophes, compared with a 1% prior probability. This indicates that there may be model misfit for Ugaritic. We now perform a Bayesian cross-validation analysis of all 10 calibration constraints, as discussed by Ryder and Nicholls (2011). To do this, we repeat the MCMC run, with one modification in the settings: in the "Impose clades ignoring" box, we list (for example) clade number 1, then let the MCMC run for a sufficient number of iterations. Clade number 1 gives constraints on the age and branching time of Hebrew. With this setting on, these constraints are no longer imposed, letting the Hebrew leaf age and branching time vary freely. At the end of the run, we use the "show age histogram" feature of the Analysis mode to compare the resulting posterior ages with the constraint. We find that they overlap, indicating support of the historically attested constraint. We repeat this for all constraints, and find problems with some constraints: the branching of Biblical Aramaic and the leaf ages of Ugaritic and Gehez. We therefore decide to remove Ugaritic and Gehez from are final analysis, given the evidence that they are outliers. This improves the model fit, and all remaining constraints are then supported (including Aramaic branching).

We now perform are final run, from which we will draw our conclusions. In the MCMC settings, we omit the taxa 1 and 17 (Gehez and Ugaritic) and impose all clades. We do not include catastrophes, since they seem unnecessary. With this run, we can build a consensus tree and estimate the root age of the Semitic family. Our 95% HPD interval for the age of the root is [3800, 5100] BP.

References

Dollo L (1893). "Les Lois de l'Évolution." *Bulletin de la Société belge de Géologie, de Paléontologie et d'Hydrologie*, **7**, 164–166. Translated in Gould [1970].

Drummond A, Nicholls G, Rodrigo A, Solomon W (2002). "Estimating Mutation Parameters, Population History and Genealogy Simultaneously From Temporally Spaced Sequence Data." *Genetics*, **161**(3), 1307–1320.

Felsenstein J (1981). *Inferring Phylogenies*. Sinauer Associates Sunderland, Mass., USA.

Geyer CJ (1992). "Practical Markov Chain Monte Carlo." *Statistical Science*, **7**(4), 473–483.

Gould S (1970). "Dollo on Dollo's Law: Irreversibility and the Status of Evolutionary Laws." *Journal of the History of Biology*, **3**(2), 189–212.

Kitchen A, Ehret C, Assefa S, Mulligan CJ (2009). "Bayesian Phylogenetic Analysis of Semitic Languages Identifies an Early Bronze Age Origin of Semitic in the Near East." *Proceedings of the Royal Society B: Biological Sciences*, **276**(1668), 2703–2710.

Maddison DR, Swofford DL, Maddison WP (1997). "Nexus: An Extensible File Format for Systematic Information." *Systematic Biology*, **46**(4), 590–621.

Nicholls GK (1998). *Physics 707 Inverse Problems Lecture Notes*, chapter 9. University of Auckland. URL http://www.stats.ox.ac.uk/~nicholls/.

Nicholls GK, Gray RD (2008). "Dated Ancestral Trees from Binary Trait Data and its Application to the Diversification of Languages." *Journal of the Royal Statistical Society, series B*, **70**(3), 545–566.

Nicholls GK, Ryder RJ (2011). "Phylogenetic Models for Semitic Vocabulary." In *Proceedings of the International Workshop on Statistical Modelling*.

Rambaut A, Drummond A (2009). "**Tracer** v1.5." http://tree.bio.ed.ac.uk/software/tracer/.

Ryder RJ (2010). *Phylogenetic Models of Language Diversification*. Ph.D. thesis, University of Oxford.

Ryder RJ, Nicholls GK (2011). "Missing Data in a Stochastic Dollo Model for Binary Trait Data, and its Application to the Dating of Proto-Indo-European." *Journal of the Royal Statistical Society Series C*.

Affiliation:

Robin J. Ryder
Centre de Recherche en Mathématiques de la Décision
Université Paris-Dauphine
75 775 Paris Cedex 16
E-mail: ryder@ceremade.dauphine.fr
URL: http://www.ceremade.dauphine.fr/~ryder/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd