TraitLab: a Matlab package for fitting and simulating binary tree-like data

Luke J. Kelly[*1], **Geoff K. Nicholls**[2], **Robin J. Ryder**[3] **and David Welch**[4]

[1] *School of Mathematical Sciences, University College Cork, e-mail:* lkelly@ucc.ie

[2] *Department of Statistics, University of Oxford, e-mail:* nicholls@stats.ox.ac.uk

[3] *CEREMADE, CNRS, UMR 7534, Université Paris-Dauphine, Université PSL, e-mail:* ryder@ceremade.dauphine.fr

[4] *School of Computer Science, University of Auckland, e-mail:* david.welch@auckland.ac.nz

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 rate heterogeneity through catastrophes, evolutionary events where many traits are simultaneously lost while new ones arise, and borrowing, whereby traits transfer laterally between species as well as through ancestral relationships. 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 distribution for tree topologies, clade and root ages, and the trait loss, catastrophe and borrowing 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 heterogeneity in the trait loss rate, biases in the data collection process and borrowing of traits between lineages. Coupled pairs of Markov chains can be used to diagnose MCMC mixing and convergence and to debias MCMC estimators. The raw data, MCMC run output, and model fit can be inspected using a number of useful graphical and analytical tools provided within the package or imported into other popular analysis programs. TraitLab is freely available and runs within the Matlab computing environment with its Statistics and Machine Learning toolbox, no other additional toolboxes are required.

Keywords and phrases: Bayesian inference, dating methods, Markov chain Monte Carlo, phylogenetics, binary data, trait data, historical linguistics, stochastic Dollo model.

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. In the past two decades, phylogenetic models have also become popular for other types of data, in particular to describe the changes through time of languages and cultures. These models have been applied to many language families, including Indo-European (Gray and Atkinson, 2003; Chang et al., 2015; Heggarty et al., 2023), Sino-Tibetan (Sagart et al., 2019), Bantu (Currie et al., 2013; Koile et al., 2022), Pama-Nyungan (Bouckaert, Bowern and Atkinson, 2018) and Austronesian (Gray, Drummond and Greenhill, 2009).

TraitLab[1] was developed specifically to fit binary trait data under a stochastic

*Corresponding author.

[1] Available to download from https://github.com/traitlab-mcmc/TraitLab.

Dollo model of evolution, as described by Nicholls and Gray (2008). This is a stochastic analogue of the (Dollo, 1893) parsimony principle, 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 for the rate at which catastrophes occur and the probability of trait death at a catastrophe. TraitLab also allows for borrowing in the Dollo model, whereby species acquire traits from contemporary species outside of ancestral relationships, which introduces a parameter for the rate at which each trait instance attempts to transfer laterally.

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 death and transfer 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 or via a configuration file specified from the command line.

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 3.2.2.

TraitLab can be used to infer phylogenies from any binary data for which the stochastic Dollo model is relevant, but the majority of uses of TraitLab so far have been for inferring or simulating language phylogenies of cognate lexical data (Atkinson et al., 2005; Greenhill, Currie and Gray, 2009; Koch, 2016; Atkinson, 2006; Greenhill, Heggarty and Gray, 2020; Sagart et al., 2019), and this is the use case we consider in Appendix A. Applications of the stochastic Dollo model in other fields include miRNA evolution (Thomson et al., 2014) and cancer biology (McPherson et al., 2016).

Several other packages exist for phylogenetic analysis of binary data, with significantly different assumptions from TraitLab. For instance, MrBayes (Ronquist et al., 2012) implements the finite-sites trait evolution model of Lewis (2001), which assumes that the number of traits is fixed in advance, and that

a trait can be born independently at several points on the tree, whereas the stochastic Dollo model we fit allows for the number of traits to be random, but each trait can only be born once (homoplasy is not allowed). MrBayes has nonetheless also been applied to lexical traits (Gray and Atkinson, 2003). Similarly, BayesPhylogenies (Pagel, Meade and Barker, 2004) also implements MCMC for a phylogenetic model in which a trait can be born at several different points. BEAST (Suchard et al., 2018) proposes MCMC implementations of several models, including the related multi-state stochastic Dollo process of Alekseyenko, Lee and Suchard (2008), but without our modelling extensions, and without the possibilities included in TraitLab includes for simulations of synthetic data from generalized models. Barbançon et al. (2013) compare several packages for the analysis of lexical trait data. Beast 2 (Bouckaert et al., 2019) also has an implementation of the stochastic Dollo model, and of the related pseudo-Dollo model (Bouckaert and Robbeets, 2017). To our knowledge, TraitLab is one of only two implementations of a model with horizontal transfer of lexical items, the other being `contacTrees` (Neureiter et al., 2022).

Many ad hoc approaches exist which attempt to diagnose MCMC convergence in phylogenetics problems but may fail silently. A coupling of the MCMC algorithm, whereby a pair of coupled Markov chains explore the target posterior distribution and meet exactly after a random number iterations, can be used to jointly diagnose convergence across all components of the model and debias estimators constructed from MCMC samples. At the time of writing, TraitLab is the only phylogenetics software to implement this coupling approach.

Further details on the basic model, its implementation and its application to linguistic data can be found in Nicholls and Gray (2008), Atkinson et al. (2005) and Nicholls and Ryder (2011). The extension for rate heterogeneity through catastrophes is described in Ryder and Nicholls (2011) with additional detail in Ryder (2010). Incorporating lateral trait transfer in the stochastic Dollo model is described Kelly and Nicholls (2017) with further details contained in Kelly (2016). The algorithm to couple MCMC for phylogenetic inference is described by Kelly, Ryder and Clarté (2023).

The remainder of the paper is organized as follows. Sections 2.2–2.4 describe the trait evolution and data collection models. Section 2.5 outlines the prior distributions and Section 2.6 describes the corresponding posterior. Section 2.7 discusses the construction of the MCMC algorithm for sampling from the posterior. Directions on how to install and start TraitLab are given in Section 3.1. Section 3.2 specifies the data file format. Sections 3.3-3.5 give step-by-step instructions for running an MCMC analysis while Section 3.6 describes the tools provided for analysing and visualising the output of the MCMC run. Finally, in Section 3.7, we describe tools to simulate data under the Dollo model and various extensions, and how this synthetic data can be used to check for model fit and model misspecification. Appendix A describes a step-by-step analysis of lexical data from Semitic languages.

2. Methods

2.1. Notation

Let g be a binary rooted tree with $2L-1$ nodes: L leaves and $L-1$ internal nodes, of which the *root* is the eldest and the most recent common ancestor of

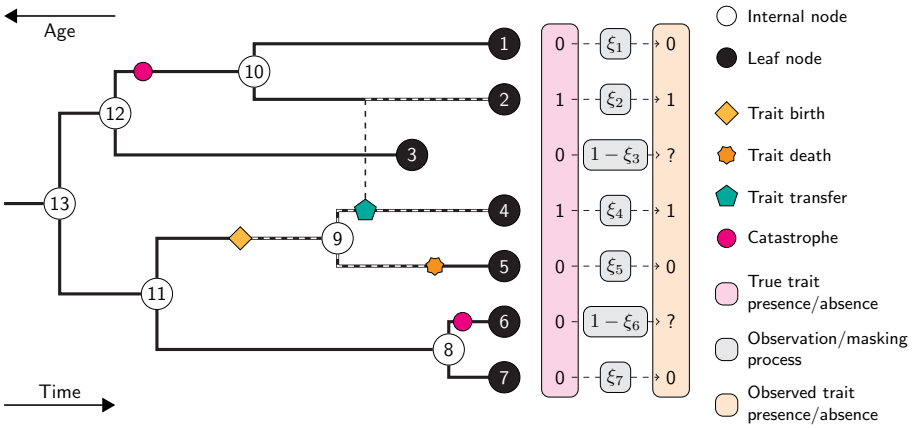

FIGURE 1. *A phylogeny with $L = 7$ leaves and a trait history sampled from the stochastic Dollo model. The leaves correspond to the observed taxa, the indexing of the internal nodes is completely arbitrary. The root is node 13. The trait born on edge 9 was copied into the offspring lineages 4 and 5, was borrowed into edge 2 from 4 and died out on 5. The trait was present at leaves 2 and 4 and absent at the rest but, due to a missing-at-random masking process, its status was not recorded in leaves 3 and 6. The trait in this example did not pass through either catastrophe node on the tree.*

all the leaves. Let $V = \{1, \ldots, 2L-1\}$ be the set of node labels; in particular, the leaves bear the labels $1, \ldots, L$. Let t_i denote the age of node $i \in V$ and $t = (t_1, \ldots, t_{2L-1})$. Node ages increase along paths from the leaves towards the root of the tree. The directed edges (also called branches) of g run forwards in time from a parent node to its child. We index edges by their offspring node so refer to the edge leading into node i from its parent as edge i. Additionally, an edge of infinite length leads into the root from its parent, but we ignore the parent node in our notation. Let E denote the set of edges in g. Thus, $|E| = 2L-1$ as it consists of the $2L-2$ internal edges of g and the edge of infinite length leading into the root node from its parent. The tree is thus defined by $g = (V, E, t)$. Figure 1 displays a rooted, binary tree.

2.2. A stochastic Dollo model of evolution

In the basic Dollo model of species diversification, three types of events may occur: traits are born, traits die, and traits are duplicated when the lineages containing them split. All evolutionary events are independent, so traits do not interact within or across species and speciation events occur independently of trait events. Traits are born in each species according to a Poisson process with constant rate λ in each species. Each trait instance in each lineage dies at constant per capita rate μ. When a lineage i splits, it is replaced by two identical copies of itself, j and k, say, which continue to evolve independently of all other lineages. The trait process is in equilibrium at the root, since the edge leading into has infinite length, then evolves along the tree to the leaves where the presence or absence of each trait is recorded. Figure 1 displays the history of a trait drawn from the stochastic Dollo model including the extensions described in Section 2.4.

2.3. Likelihood

To calculate the likelihood of data recorded at the leaves given the tree and model parameters, we must integrate over the unobserved evolutionary events on the phylogeny under the model. Fix an ordering of the leaf nodes. Each fully observed trait displays a binary pattern of presence-absence across the leaves. For example, the trait in Figure 1 would display the pattern $p = (0, 1, 0, 1, 0, 0, 0)$ if it were fully observed. We cannot observe a trait absent in all of the leaves so do not consider the pattern $(0, \ldots, 0)$ so the space of observable patterns is $\mathcal{P} = \{0, 1\}^L \setminus \{(0, \ldots, 0)\}$.

Let N_p denote the number of traits in D displaying pattern $p \in \mathcal{P}$ at the leaves and denote $\mathbb{E}[N_p \mid g, \lambda, \mu]$ its expected frequency under the model. We have a Poisson process of trait births with independent thinning so $N_p \mid g, \lambda, \mu \sim \text{Poisson}(\lambda z_p)$, where $z_p = \mathbb{E}[N_p \mid g, \lambda = 1, \mu]$. Thus, in the basic stochastic Dollo model the likelihood of the pattern counts given the parameters is

$$\pi\big((N_p : p \in \mathcal{P}) \mid g, \lambda, \mu\big) = \prod_{p \in \mathcal{P}} \frac{(\lambda z_p)^{N_p} e^{-\lambda z_p}}{N_p!} \propto \lambda^{\sum_{p \in \mathcal{P}} N_p} e^{-\lambda \sum_{p \in \mathcal{P}} z_p} \prod_{p \in \mathcal{P}} z_p^{N_p}.$$

We place an improper prior $\pi(\lambda) \propto 1/\lambda$ on the birth rate, then we integrate it out of the prior-weighted likelihood analytically to obtain

$$\begin{aligned} \pi\big((N_p : p \in \mathcal{P}) \mid g, \mu\big) &= \int_0^\infty \pi\big((N_p : p \in \mathcal{P}) \mid g, \lambda, \mu\big)\pi(\lambda)\mathrm{d}\lambda \\ &\propto \prod_{p \in \mathcal{P}} \left(\frac{z_p}{\sum_{q \in \mathcal{P}} z_q}\right)^{N_p}, \end{aligned} \tag{1}$$

a multinomial distribution across the patterns conditional on the total number of traits observed. See Nicholls and Gray (2008) for further details on the model and likelihood computation.

2.4. Model extensions

2.4.1. Catastrophic rate heterogeneity

Catastrophes are added to this model to account for rate heterogeneity along edges. Catastrophes occur independently along each lineage at rate ρ. At a catastrophe, each trait dies with probability κ and a Poisson(ν) number of new traits are born. We impose the condition that $\lambda/\mu = \nu/\kappa$ so that the expected number of traits in any lineage remains constant over time and the process is reversible. This is equivalent to instantaneously advancing the trait process at each catastrophe on the edge by $-\mu^{-1}\log(1-\kappa)$ units of time: it is therefore convenient to extend the definition of a tree to include the number of catastrophes that occur on each edge. Denote k_i the number of catastrophes that occur on branch i and $k = (k_1, \ldots, k_{2L-2})$ the vector of catastrophe counts for all finite edges. The extended tree is defined by $g = (V, E, t, k)$ and the parameter space is also expanded to include the catastrophe strength κ. See Ryder and Nicholls (2011) for further details of the catastrophe process and how to incorporate it in the likelihood calculation.

2.4.2. Observation model for traits

Some kinds of traits are deliberately removed from the data. For example, in the absence of lateral trait transfer, it is common to discard traits displayed at just one taxon. On the other hand, when lateral trait transfer is suspected, singleton traits are evidence of a lack of borrowing. If only traits in observed in at least d taxa are registered in the data (with $d = 1$ or $d = 2$), then we restrict $\mathcal{P}$ accordingly and make the corresponding marginalisation of the likelihood exactly. 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 $1 - \xi_i$ or else recorded with probability ξ_i. When missing data is allowed in the model, the parameter space is extended to include $\xi = (\xi_1, \ldots, \xi_L)$. See Ryder (2010) and Ryder and Nicholls (2011) for further details of these extensions for missingness and data registration operations.

2.4.3. Borrowing

To account for borrowing of traits between evolving species, we allow for a global lateral transfer process whereby each trait-instance in each lineage attempts to transfer a copy of itself to other branches at rate β. An attempted borrowing succeeds if the trait is not already present on the destination branch. When lateral transfer occurs, the trait process is no longer time-reversible since the rate at which traits transfer depends on the number of instances of each trait and the number of edges. In this setting, a catastrophe advances the trait process along the affected edge by $-\mu^{-1}\log(1-\kappa)$ units of time relative to the other edges, and during a catastrophe traits may be born, die or transfer into the affected edge but cannot transfer out.

When the model allows for lateral transfer, the parameter space is extended to include β. Furthermore, if catastrophes are allowed in the model, then we expand the tree g to include their locations along edges. See Kelly and Nicholls (2017) for a complete description of lateral transfer in the stochastic Dollo model accounting for catastrophes, missing data and data registration, and Kelly (2016) for a detailed discussion of the computation.

2.5. Priors

TraitLab operates in the subjective Bayes framework. This puts an onus on the user to specify priors encoding the prior knowledge which is actually available. The following priors may need to be adjusted for any given application. Those given below may be appropriate for analysis of lexical trait data, in which age is measured in units of years.

Our prior on the death rate $\mu \sim \Gamma(0.001, 0.001)$, with mean 1 and variance 1000, and likewise for the borrowing rate β. These broad priors were chosen to represent a lack of prior knowledge. The user may change these priors by editing `LogPriorParm.m` The prior on the catastrophe rate ρ is $\Gamma(1.5, 5000)$. The parameters of this Gamma prior were chosen to place 90% prior mass on an interval of 1300 and 28000 years between catastrophes. The user may change

this prior by editing `LogRhoPrior.m`. So that the effects of catastrophes are identifiable with respect to the underlying process, we place a $\mathrm{U}(0.25, 1)$ prior on κ, the death probability at a catastrophe, thus ruling out weak catastrophes. We take an independent $\mathrm{U}(0, 1)$ prior for each missing data parameter ξ_i. Two classes of priors on trees are available. One, which we call an exponential prior, is an exponential branching process with rate θ stopped at the instant of the Lth branching event (counting the branching at the root as the first event). This determines a distribution

$$f_E(g|\theta) \propto \theta^{L-1} \exp(-\theta|g|) \tag{2}$$

where $|g|$ is the sum of all branch lengths beneath the root. The scale parameter θ is given a conjugate Jeffreys prior $1/\theta$ then integrated out, yielding the marginal tree prior $f_E(g) \propto |g|^{-L+1}$.

The other prior on the tree g is chosen in such a way that the marginal prior on the root age is approximately uniform over the interval $[s, T]$, where s is the maximum of the lower bounds on node ages imposed by the clade constraints (or zero of there are no clade constraints) and T is a maximum root age imposed by the user. Nicholls and Gray (2008) show that the prior distribution with density

$$f_G(g \mid T) \propto \mathbf{1}_{\{t_r < T\}} \prod_{i \in F(g) \setminus \{r\}} \frac{T - s_i^-(g)}{t_r - s_i^-(g)} \tag{3}$$

where $F(g)$ is the set of *free* nodes in g with ages upper bounded only by T and $s_i^-(g)$ is the least age node i can take on the tree g, induces an approximately uniform marginal prior distribution over the root age, provided that the greatest upper bound imposed by the clade constraints is not too close to T. See Nicholls and Ryder (2011) for further details of this tree prior. When clade constraints are placed on the tree, as described in Section 3.2.2, the prior is defined to be zero for any tree not satisfying those constraints.

2.6. Posterior distribution

We multiply the expression for the likelihood (1) by the corresponding priors to obtain the posterior density for the model, with parameters on their valid ranges and trees satisfying all of the clade constraints and with root age less than T. TraitLab determines the likelihood and requisite priors from the model specification. Similar to the birth rate λ in (1), we integrate out the catastrophe rate ρ to obtain a Negative Multinomial distribution for catastrophe counts along branches. If desired, ρ can be fixed at a constant value and included in the model. See Kelly, Ryder and Clarté (2023, Supplement A) for further details of these properties.

2.7. MCMC algorithm

Let x denote the tree and parameters to infer for the chosen model after marginalisation of λ and ρ, which ranges from $x = ((E, V, t), \mu)$ for the basic model to $x = ((E, V, t, k), \mu, \beta, \kappa, \xi)$ for the full model including lateral trait transfer, catastrophes and missing data. The target of our inference is the posterior distribution $\pi(x \mid D)$, We sample from the posterior distribution using MCMC. TraitLab

TABLE 1
The moves currently used in the MCMC sampler to explore the state space. Move indices correspond to their indices within TraitLab. Some moves that were used in earlier versions of TraitLab have been removed.

Primary target	Move	Description
Topology	2	Interchange parents of neighbouring node pair
	3	Interchange parents of randomly chosen node pair
	4	Move edge to neighbouring location
	5	Move edge to randomly chosen location
Node times	1	Vary internal node time
	11	Vary leaf time
	6	Rescale tree
	7	Rescale subtree
	12	Rescale tree above clade bounds
Catastrophes	13	Add catastrophe
	14	Delete a catastrophe
	17	Move catastrophe on edge
	18	Move catastrophe to neighbouring edge
	15	Resample all catastrophes on edge
Model parameters	8	Rescale death rate μ
	21	Rescale transfer rate β
	16	Rescale catastrophe strength κ
	19	Rescale one missing data parameter ξ_i
	20	Rescale all missing data parameters ξ

uses the Metropolis–Rosenbluth–Teller–Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) to construct an ergodic Markov chain $X_0, X_1, \ldots$ whose limiting distribution is $\pi(x \mid D)$, where a state of the Markov chain is some value for each component of x.

General updates to the state x are intractable, so our MCMC uses a mixture of local proposal kernels, where each kernel targets a different component of the state. 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 and which are described by Drummond et al. (2002), five proposals that alter the heights of some or all nodes in the tree, five proposals that add, delete or shift catastrophes and which are described in Ryder and Nicholls (2011) and Kelly, Ryder and Clarté (2023), while the remaining five moves multiply one or more of the scalar parameters μ, β, κ and $\xi = (\xi_1, \ldots, \xi_L)$ by a randomly chosen factor. See Kelly, Ryder and Clarté (2023, Supplement A) for a recent description of these moves.

2.8. Coupling

Following the framework developed by Jacob, O'Leary and Atchadé (2020) and Biswas, Jacob and Vanetti (2019), TraitLab allows the user to sample a pair of lag-l coupled Markov chains, $(X_s)_{s\geq 0}$ and $(Y_s)_{s\geq 0}$, which can be used to estimate convergence bounds and debias MCMC estimators. In short, having first sampled $X_0, \ldots, X_l$ from the marginal MCMC kernel targeting $\pi(x \mid D)$, the pair (X_s, Y_{s-l}) are drawn from a coupling of their marginal transition kernels at each iteration $s > l$. The chains meet exactly at a random finite time $\tau^{(l)}$ and remain together thereafter. Provided that the tail of $\tau^{(l)}$ tail decays, at

most, geometrically then we can use the meeting times to estimate a bound on the total variation distance between the marginal distribution of X_s and the target $\pi(x \mid D)$. Kelly, Ryder and Clarté (2023) describe the algorithm in detail and illustrate its utility in diagnosing convergence across all components of the phylogenetic model from the meeting times of pairs of chains run in parallel. Coupling may also be used to check for overparameterization; for example, if catastrophes are only weakly identified in the model then coupled chains will struggle to meet. Section 3.5 describes how to apply the coupling scheme in TraitLab.

2.9. Computational cost

The computational cost at each iteration of the MCMC is dominated by evaluating the likelihood. Depending on the model configuration, TraitLab uses one of two computational schemes:

- In the absence of lateral trait transfer ($\beta = 0$), Nicholls and Gray (2008) describe how to efficiently compute the non-zero terms in (1) directly using a variant of Felsenstein's pruning algorithm (Felsenstein, 1981). The computational cost of this likelihood calculation is linear in the number of taxa and the number of site patterns. Under the model with no lateral transfer, TraitLab's implementation thus allows for a large number of taxa (at least several hundreds).
- When lateral trait transfer is included in the model ($\beta > 0$), the likelihood has the same form as in (1). However we cannot evaluate the expected pattern frequencies $(z_p : p \in \mathcal{P})$ through recursions so must instead solve an expanding sequence of initial value problems (Kelly and Nicholls, 2017). In the final linear system, there is one differential equation for each pattern in $\mathcal{P}$. As $|\mathcal{P}| = 2^L - 1$, the computational cost of evaluating the likelihood to machine precision grows exponentially with L. Consequently, computation with the lateral transfer model is only feasible for $L \leq 20$ taxa.

The family of MCMC proposals in Table 1 are designed to be computationally cheap per MCMC iteration at the possible expense of sampling efficiency. The lag-coupled MCMC implementation in TraitLab can be used to accurately assess how well the MCMC algorithm explores the posterior for a given problem. Sampling from the coupled Markov proposal kernel for a pair of chains takes a little over twice as long as sampling from the marginal kernel for a single chain, and again the likelihood calculation dominates the computational cost since currently TraitLab computes it in serial for each chain.

3. Package

3.1. Installing and running TraitLab

3.1.1. System requirements

TraitLab is written in the Matlab programming language. Matlab is proprietary software which runs on Windows, Mac OS X or Unix/Linux machines. TraitLab only requires the basic Matlab installation and its Statistics and Machine Learning toolbox.

3.1.2. Download and installation

Download the TraitLab code directory as a ZIP file from https://github.com/traitlab-mcmc/TraitLab and extract its contents preserving the subdirectory structure. At the top level of the TraitLab directory are functions to launch TraitLab and scripts to set up the environment. Lower level functions are contained in sub-directories, the `README` files within describe their contents.

The borrowing likelihood calculation requires either the installation of the Matlab Communications toolbox or the `mex` compilation of substitute functions written in C in the `borrowing` directory. This requires that Matlab's `mex` compiler has been setup. The files may be compiled from the top level of the TraitLab directory by executing `mexFiles` in the Matlab command window. The compiled functions have the same names and syntax as their Communication toolbox counterparts.

3.1.3. Launching TraitLab

Start Matlab at the top level of the TraitLab directory. The `startup.m` script is automatically executed to set the correct path. Otherwise, start Matlab in the usual manner, make the TraitLab folder the current directory and execute `startup` in the Matlab command window.

To start TraitLab and bring up the main GUI, type `TraitLab` at the Matlab command line. Section 3.4 describes how to start MCMC runs using an input file instead of the GUI, either through a command line or the Matlab command window, which is useful for running long chains or launching multiple experiments.

3.2. Data format and loading data

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. Block names, formatting and other properties in the Nexus file are case insensitive in TraitLab; taxa labels are case sensitive. In the following, we describe the structure of these blocks and provide an example of a data file in Section 3.2.4.

3.2.1. Data block

The first command in the `Data` block must be the `Dimensions` command with values for `ntax` (the number of taxa L above) and `nchar` (the number of characters or traits N above) 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 and may not contain any whitespace characters. The content of the

matrix must be zeros, ones and question marks, to indicate absence, presence or missingness 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.

3.2.2. 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, and 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
Taxa = ...

End;
```

Each `clade` command must include a name and the list of taxa that define the clade. TraitLab will then force those taxa to be monophyletic: there must be a subtree consisting of exactly those taxa. If desired, time constraints can be added to 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 clade root node) can be bounded by defining `originatemin` and `originatemax`. See Figure 2 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, then 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 = 501` and `rootmin = 499`.

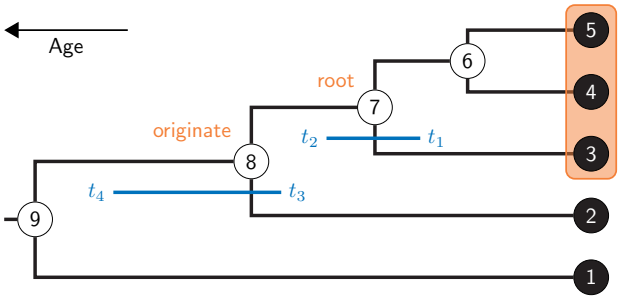

FIGURE 2. *Suppose a clade comprises taxa 3, 4 and 5. If* `rootmin` $= t_1$ *and* `rootmax` $= t_2$*, then the age of the node labelled root, node 7 here, is constrained to lie in the interval* $[t_1, t_2]$*. Similarly, if* `originatemin` $= t_3$ *and* `originatemax` $= t_4$*, then the age of the node labelled originate, node 8 here, 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 leaf node 2.*

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

3.2.4. 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 then 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?11001111001011011000
taxon_3  01101101??1110111?001010011000
```

```
taxon_4  0101111001110111001100000100100
taxon_5  1101111011110l1??0110100100100
taxon_6  1111110101117011110010110110l1
taxon_7  1111???10110101111107?011011011
taxon_8  1111110001111011100010110110l1
taxon_9  1111110l?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;
```

3.3. Initializing an MCMC run

MCMC analyses are set up and run in TraitLab from the main GUI shown in Figure 3. There are five 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), which parts of the data to ignore (if any), and the run parameters including run length and output file location. These five types of information correspond to the five colours of the panels in the GUI and are each explained in detail in this subsection.

3.3.1. Data source

Specify the data source by clicking the `Select data file` button and choose a Nexus file conforming to the TraitLab requirements described in Section 3.2. 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. If the data were synthesized using TraitLab, this will also be indicated.

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.

FIGURE 3. *The main TraitLab GUI.*

3.3.2. 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. Taxa numbers 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 3.3.5 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 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.

3.3.3. Initial tree

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

If the `Start from a random tree` option is selected, an Exponential tree (that is, a tree with exponentially distributed times between branching events) with specified branching rate θ (so that the mean branch length is proportional to $1/\theta$) is generated, then a short MCMC chain targeting the prior is run to

create the initial state for the main analysis. If clades are imposed, then 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 a failed run.

If a run is started from a state in an output file which did not have lateral transfer in the model but the new run does, then TraitLab will return an error as there is no valid initial value for β in the initialising output file. Only the number of catastrophes per branch is written to file and not their locations along branches. Thus, when starting a run with lateral transfer from an output state, the location of each initial catastrophe is sampled uniformly along its corresponding branch.

3.3.4. 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 may not be observed. In addition, the user must specify here which of the model parameters, g, μ, κ and ξ are to be estimated. The catastrophe rate ρ can be integrated out analytically or fixed at a specified value.

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 (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. One may additionally weight this prior so that its distribution over topologies is uniform. The other tree-prior option is the so called "exponential prior" in which trees are given prior weight according to branch lengths which have an exponential penalty, as in (2). If the `Vary tree topology` box is unchecked, the tree will remain in the same shape throughout the run with only the node ages 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 2.4.2. 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 not unusual for traits present in only one taxon to be discarded; if in doubt, we recommend that the checkbox remain checked.

If the data matrix includes traits coded as missing (that is, it includes `?` characters), the `Model missing data` box should be checked. If it is left unchecked, then 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 2.2 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 has the option of fixing ψ, and thus μ, at a specified value or letting μ vary over the run in order to estimate it.

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 during the run in the case of κ and integrated out in the case of ρ.

The model allows for trait borrowing between branches at rate β when the `Allow for lateral transfer` checkbox is ticked. Radio buttons and a text box allow the user to specify an initial value for β or let TraitLab initialise it with a uniform draw centred on the initial value for μ. If the `Vary lateral transfer rate` checkbox is checked (the default), then β will vary over the course of the MCMC run, unchecking it means that β will be fixed at its initial value.

3.3.5. Imposing clades

If clade constraints are present in the data file, they may be applied during a run by checking the `Impose clades ignoring:` checkbox.

To omit clades from the analysis, list the clades to be omitted in the space provided using the respective clade numbers (see Section 3.3.2 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 = inf`) by checking the box `Ignore ages for clades` and listing the clade numbers in the space provided.

Difficulties can arise when omitting taxa that appear in imposed clades. TraitLab warns the user about such cases in the Matlab command window and gives the option to either keep the clade with the taxon/taxa removed, or to ignore the clade completely. Note that the first option can lead to errors, such as inconsistencies with other clades.

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 3.4) as it is the only way to avoid user interaction.

3.3.6. 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. Choose r and j so that the total number of saved sampled states is somewhere in the range 1000–10000. Sub-sampling reduces the correlation between samples and keeps the output files of a manageable size.

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 data set in question. Generally, the greater the number of taxa, the longer the run needs to be. In some cases, it may take many days or weeks to obtain a satisfactory number of samples for larger problems. 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. Simple checks for a lack of convergence are discussed in Section 3.6. If the computational budget allows, then the coupling approach in Section 3.5 is a state-of-the-art approach for diagnosing convergence across all components of the tree and model.

The `Seed random numbers` checkbox is mainly used for debugging purposes and is generally left unchecked. If checked, then a seed for the random number generator needs to be specified (it can be any real number) and will be recorded in the output `.par` file. Separate runs with the same options and random seed will be identical. If left unchecked, then the random number generator will be shuffled and no seed will be recorded.

3.3.7. Output files

Output files are saved in the location specified using the `Select output file` button. If the default `tloutput` is the chosen file name, the output files created: `tloutput.txt`, `tloutput.par`, `tloutput.nex` and `tloutputcat.nex`. If missing data is included in the model, then `tloutputXI.txt` is also created. The coupling algorithm creates additional files which are described in Section 3.5.

A record of all parameter values and options used to create the run are written to `tloutput.par`. It can be used to perform similar runs in batch mode; see Section 3.4 for details.

Each sampled tree is recorded in `tloutput.nex` and the corresponding parameter values `tloutput.txt` files. The number of catastrophes on each branch of the trees in `tloutput.nex` are recorded in `tloutputcat.nex`. Parameters recorded in `tloutput.txt` include the root time for the sampled tree, the trait death rate μ, the death probability at a catastrophe κ and the borrowing rate β. Note that since λ and ρ are integrated analytically, we return direct samples from their respective marginal posterior distributions at each iteration of the MCMC. Also in the `.txt` file are unnormalized values of the log-prior density for the state, the integrated log-likelihood (1) (where λ has been integrated out) and the Poisson log-likelihood (using the sampled value of λ). Finally, the ξ parameters related to missing data are stored in the `tloutputXI.txt` file.

3.3.8. 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. See Figure 8 for an example.

To plot each tree as it is sampled, check the `Draw trees` checkbox. If the box is unchecked, then the most recently sampled tree can be plotted by clicking the `View latest tree` button.

To see the trace plots of the sampled root time, trait loss rate, log-prior density and integrated log-likelihood, 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 an unnormalized log-likelihood (1) of approximately −3550 and, of the updates between sample 4 and sample 5, 33% of the states proposed by move type 1 were accepted, 24% of those proposed by move type 2 were accepted and so on. The indices of the moves are given in Table 1. Note that, depending on the options chosen, some moves may not be proposed so the proportion accepted will be `NaN`.

3.4. Running TraitLab in batch mode

TraitLab can be run without the GUI interface from the Matlab command window or directly from a Unix, Linux or Mac command line. Instead of using the GUI to specify all parameters for a run, they are read from a `.par` file. Rather than write an input file from scratch, it may be more straightforward to set up a short model run using the GUI and then edit the resulting `.par` for the other runs in batch mode.

To run TraitLab without using the GUI, we use the function `batchTraitLab` whose first argument is the path to the `.par` file specifying the run parameters. An optional second argument takes a numeric variable which is appended to the names of created files, so is useful when replicating experiments.

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

```
matlab -batch 'batchTraitLab("<path to .par file>")' > out.log
```

at the command line, where the path to the `.par` file is a enclosed in double quotation marks and the entire command in single quotation marks. Any on-screen output will be dumped to `out.log` while the normal `.nex`, `.txt`, `.par`

and `XI.txt` output files will be created in the location specified in the input `.par` file.

TraitLab's batch mode is more restrictive than the GUI in initialising the model with catastrophes. If `Random_initial_cat_death_prob = 1`, then the chain is initialized with $\kappa \sim Unif(0.25, 1)$ and allowed to vary in the MCMC; otherwise κ is fixed at the value of `Initial_cat_death_prob`. Similarly, if `Random_initial_cat_rate = 1`, then ρ is integrated out analytically; otherwise it is fixed at the value of `Initial_cat_rate`.

As with the GUI, when initialising a chain with a sample from a previous run, the MCMC will ignore any specified initial value for a parameter in the input `.par` file, but will inherit whether the parameter is fixed or allowed to vary.

3.5. Coupled MCMC algorithm

In order to run a pair of lag-coupled Markov chains, edit the input `.par` file to set `Coupled_markov_chains = 1` and `Coupling_lag = <lag>`, where `<lag>` is an appropriate choice of lag l. It is a requirement of TraitLab that l be an integer multiple of `Sample_interval`. The chains will run for `Run_length` iterations or until the chains meet at random time $\tau^{(l)}$, whichever is greater. In the event that $\tau^{(l)} <$ `Run_length`, TraitLab will only run the X chain for the iterations after $\tau^{(l)}$ since the Y chain samples are identical to X after meeting. As TraitLab only checks whether the chains have met when storing samples according to the sub-sample rate, the recorded value of $\tau^{(l)}$ is slightly conservative.

We then run a pair of coupled chains via `batchTraitLab`. As we typically run many pairs of coupled chains, we use the second argument of `batchTraitLab` to add a suffix to the corresponding output files. Once the algorithm starts drawing coupled samples, TraitLab alternates between printing the sample index and log-likelihood for the X chain and the Y chain. The coupled analyses produce a similar set of output files to the marginal experiments, except `_x` or `_y` is appended to the corresponding filenames. A `.tau` file records the meeting time divided by the sub-sample rate, so we multiply the value in the `.tau` file by the sub-sample rate to obtain $\tau^{(l)}$.

Increasing the lag l reduces the variance of $\tau^{(l)}$ and thus provides better estimates of the total variation bound. As the bound is not tight, there is a diminishing return of increasing the lag; Biswas, Jacob and Vanetti (2019) advocate increasing the lag until estimated total variation bounds stabilise. We recommend running a sufficient number of coupled pairs of chains for each lag to obtain the desired level of confidence in estimates. The `example` directory contains a synthetic data set `L8-data.nex` and parameter files for coupled and marginal experiments, `L8-coupled.par` and `L8-marginal.par` respectively. Further guidance on running coupled experiments is provided by Kelly, Ryder and Clarté (2023). Scripts to set up and run coupled experiments, analyse their output and estimate convergence bounds are available at `https://github.com/lukejkelly/CoupledPhylogenetics`.

Moves which rescale multiple node times by a single random factor (moves 6, 7 and 12 in Table 1) cannot be coupled efficiently so are disabled when running the coupled MCMC algorithm. Similarly, the coupling performs better when constraints distinguish rate parameters from time parameters; for example, Kelly,

FIGURE 4. *The analysis GUI.*

Ryder and Clarté (2023) fix the death rate μ and only infer node ages in their synthetic data analyses.

3.6. 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 4. To access the analysis GUI, choose `Analyse output` from the `Mode` menu of the main TraitLab GUI.

The output inspection tools described in Section 3.6.2 allow the user to explore the output of MCMC runs, including 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 3.6.2. Note that many of these tools are fairly generic and can equally well be performed, along with other functions not available in TraitLab, by software such as Tracer (Rambaut et al., 2018).

The data inspection tools described in Section 3.6.3 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.

3.6.1. 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.

3.6.2. Exploring MCMC output

Autocorrelations, trace plots and histograms The `Output statistics` box provides functions for calculating and plotting histograms, autocorrelation functions and trace plots of key parameters and statistics. Visual inspection of the trace plot of a parameter of interest is a simple way to determine if a chain is still converging towards is target. Similar plots can be drawn for the age of internal nodes; see below for a description. For further information about how to interpret autocorrelation plots and associated statistics, see Geyer (1992). See Kelly, Ryder and Clarté (2023, Section 2.2) for an extensive overview of methods to diagnose MCMC convergence in phylogenetic problems.

To plot the autocorrelation functions, it is necessary to specify the maximum lag for which autocorrelations are calculated in the `Lag` text box. To plot marginal 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.

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, which 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.

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. 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 MRCA histograms.

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, of which a and 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 parent of a and 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 that 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.

Histogram of MRCA for given taxa The marginal posterior distribution of the time to the most recent common ancestor (MRCA) for any subset 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 with the `Show all names/numbers` option selected.

3.6.3. 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 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 are generated. The taxa per trait histogram is a histogram made from the column sums of the data matrix. The traits per taxon histogram is 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, then the two pairs of histograms should be qualitatively very similar. On the contrary, 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 Nicholls and Gray (2008, Equation 10) for further details. The distance matrix plot in Figure 5 and the distance depth relation plot in Figure 6 are based on this calculation.

The distance matrix graph is simply a heat plot of a matrix where this distance has been calculated for each pair. It is best understood by referring to the example in Figure 5.

For the depth distance relation plot, the maximum a posteriori estimate for the MRCA 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 6 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, then the points should lie along the line $y = x$.

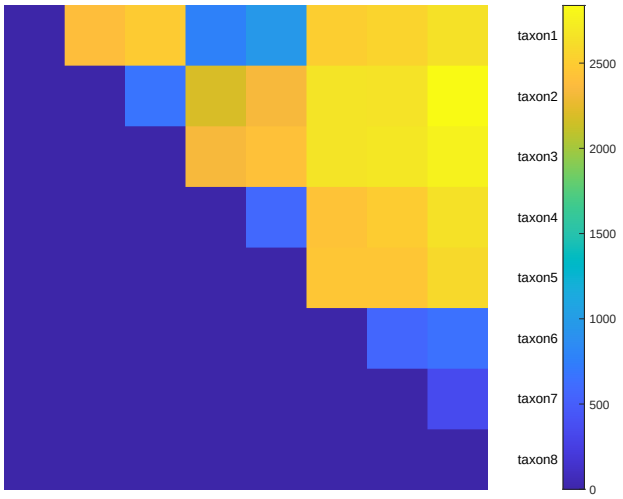

FIGURE 5. *An example of the distance matrix graph. Row labels are the taxa names, the column labels are the same as the row labels. The pixel at row i, column j represents the maximum a posteriori estimate of the MRCA between taxon i and taxon j. The colour bar displays the corresponding distance for each colour. Note that the ordering of taxa is as in the data file, so the obvious clustering that is shown here will not occur if rows in the data file are randomly ordered.*

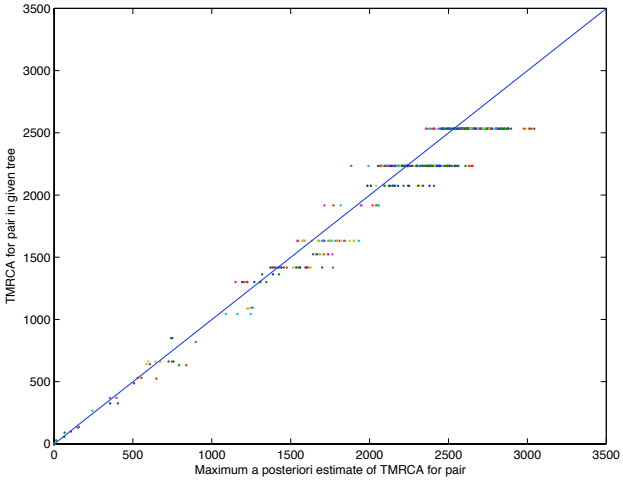

FIGURE 6. *An example of the depth distance relation graph. Each point represents a pair of taxa. The position of the point along the x-axis is the maximum a posteriori estimate of time to the most common ancestor for the given taxa pair, while the position along the y-axis is given by the time of the 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 shows a very good fit of data and model.*

3.6.4. 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 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, then the tree is multifurcating. The threshold can be changed to another value, but values lower than 50% may lead to errors. The parameter `Subsample` can be set to an integer value greater than 1 to subsample the output; 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.

3.6.5. 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 check-box 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 brief explanatory text. Alternatively, figures may be saved to other formats using the `print` functionality within Matlab.

3.7. Synthesizing data and model checking

TraitLab allows the user to simulate trait and clade data under any of the fitted models, such as 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 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 synthesise GUI is shown in Figure 7.

Selecting a tree The tree along which traits evolve can either be a randomly generated exponential tree (with exponential branch lengths) or any rooted tree (with branch lengths specified) stored in the Newick format in a Nexus file.

3.7.1. 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,

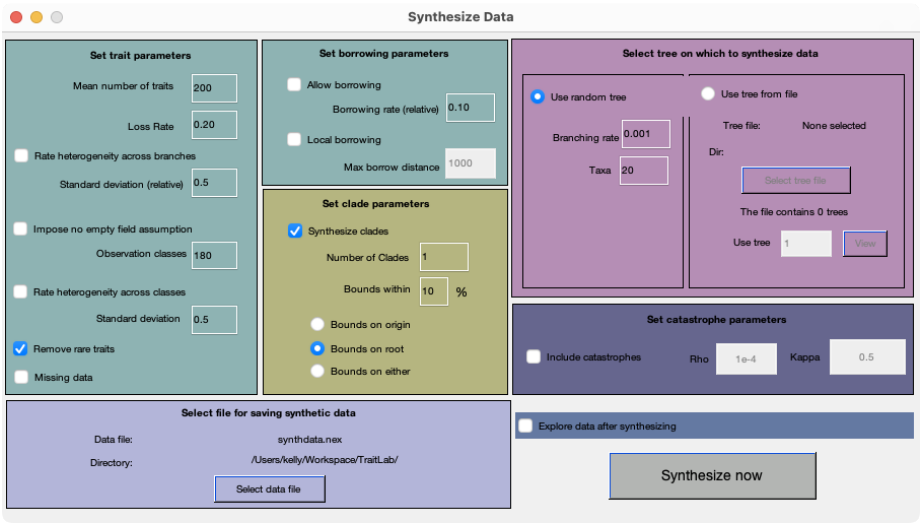

FIGURE 7. *The data-synthesizing GUI.*

we parameterize the model using the mean number of traits per taxon, K, and the trait loss rate, $\psi = 1 - \exp(-1000\mu)$. 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, $0 < \kappa \leq 1$, must be specified in their respective text boxes. The trait birth rate at a catastrophe is $\nu = \kappa\lambda/\mu$.

Rate heterogeneity across branches Another form of rate heterogeneity, different from catastrophes, is to let the trait death rate μ vary across each branch of the tree so that for branch i, there is a specific death rate $\mu_i \sim \Gamma\left(\text{mean} = \mu, \text{variance} = (\sigma\mu)^2\right)$. It is necessary to specify the relative standard deviation parameter, σ.

Observation classes: the no-empty-field assumption If the `Impose no empty field assumption` box is checked, then TraitLab 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, and is discussed by Nicholls and Gray (2008, Section 9.5). The meanings here are the observation classes, so the assumption is that every language will have a word for each basic meaning. If this option is checked, it is necessary to specify the number of observation classes n_{obs}. This option will have the greatest effect on the

simulated data when n_{obs} is close to the mean number of traits per taxon, K. Note that it is natural to set $n_{\text{obs}} \leq K$.

If the no-empty-field assumption is imposed, then heterogeneity in trait death rates across observation classes may be incorporated by checking the `Rate heterogeneity across classes` box and specifying a relative standard deviation, ς. 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 $\mu_{ij} \sim \Gamma(\text{mean} = \mu_i, \text{variance} = (\varsigma\mu_i)^2)$, where $\mu_i \sim \Gamma(\text{mean} = \mu, \text{variance} = (\sigma\mu)^2)$ as above.

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 is borrowed, it first chooses a target branch i uniformly at random from the extant lineages. If the trait is not currently present in lineage i, then it 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 chosen uniformly at random from all lineages which share a common ancestral lineage with the source lineage in the last d years.

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

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 1s (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.

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. The accuracy of the clade bounds is specified in the `Bounds within` text box. An accuracy of c means that if the chosen node has a time t in the tree, then a lower bound of $(1 - c/100)t$ and an upper bound of $(1 + c/100)t$ will be created.

3.7.2. Output of synthesize GUI

The synthetic data are saved to the `.nex` file specified by the user, the default is `synthdata.nex`. It contains the synthetic trait matrix, the tree on which the data were synthesized and associated model parameters. A `.par` file, `synthdata.par`, is generated for record keeping purposes: all options and parameter values used to produce the data are recorded here. It is not the same as the `.par` file for an MCMC run.

4. Summary

TraitLab implements a stochastic Dollo model for binary data and various extensions for rate heterogeneity via catastrophes and lateral trait transfer. It is well suited to data for which homoplasy (parallel evolution) is implausible. It offers tools to verify the convergence of the MCMC and the model fit, as well as to simulate from a variety of generalized models.

Appendix A: Analysis of Semitic lexical data

As an example, we provide in this section the steps used to analyse a real data set. The data concern core vocabulary of various Semitic languages and were collected and made public by Kitchen et al. (2009). The data set is included in TraitLab in the Nexus file `semitic09.nex` in the `example` directory. An analysis of these data with TraitLab first appeared in Nicholls and Ryder (2011).

Data file 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 data file also includes a `CLADES` block with seven clades constraints. In this case, all but one clade comprises a single language and thus gives information on one of the leaves of the tree. A subset of the clades block is:

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

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

Loading the data Launch Matlab with TraitLab the current directory according to the instructions in Section 3.1.3, then type `TraitLab` in the command window to launch the TraitLab GUI. In the `Specify data source` panel, click `Select data` and navigate to the `semitic09.nex` file in the `example` directory.

In the Matlab command window, we are given a list of clades and a list of languages, as well as a list of languages with more than 5% missing data. The TraitLab GUI reports that the data file contains 25 taxa and 674 traits.

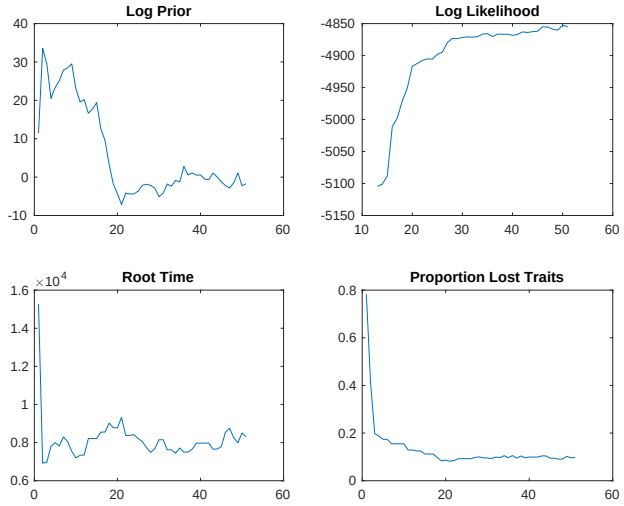

FIGURE 8. *Trace of key statistics after the first MCMC run for the Semitic data. These plots (the log-likelihood, for example) show that the MCMC has not yet reached stationarity.*

Initial run 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 subsampled to every 100 iterations, thus giving a sample of size 50.

Extra information is given in the Matlab command window. First, the data are prepared. By default, all traits which are absent at all taxa or present at only 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 8: the plot includes the traces of the log-prior, log-likelihood, root age, and proportion of lost traits (the proportion of traits which existed at some point on the tree but died before reaching any leaves). It is immediately obvious that the MCMC has not converged for this short run, so we launch a longer run: five million iterations, subsampling to every 5000 iterations for a final sample size of 1000.

At first glance, this second run (not shown) seems satisfactory from the trace plots of key statistics. We move to the `Analyse output` mode to further assess whether the chain has reached stationarity and mixed well.

Checking MCMC convergence In the `Analyse output` mode, the data and run output are loaded automatically. The log-likelihood is plotted over the entire run; we zoom in to display only iterations 102 to 1001. The plot suggests that a burn-in of 100 iterations from the initial state is satisfactory. In the green `Output statistics` panel, we check the trace and autocorrelation plots of the prior, root time and log-likelihood, as well as the parameters μ and κ, discarding the initial state and first 100 samples as burn-in. These plots, displayed in Figure 9, 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 leaf node internally. These num-

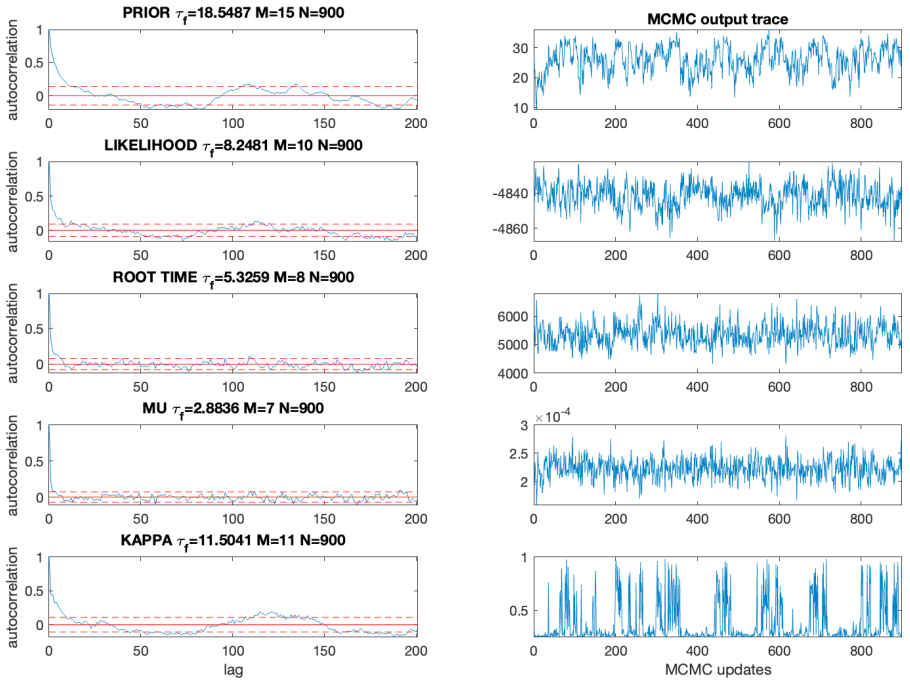

FIGURE 9. *Autocorrelation and traces of parameters and statistics after a long MCMC run for the Semitic data.*

bers can then be passed to `Show age histogram` to get the distribution of their MRCA. 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 are also given MCMC trace and autocorrelation plots for that node age the indicator that the languages given form a subtree (this gives an indication of the MCMC behaviour of the tree topology). Figure 10 displays these plots for the MRCA of Tigre and Tigrinya. Summary statistics are also given in the Matlab command window; in this example, it reports an 85% posterior probability that Tigre and Tigrinya form a clade.

These visual approaches may silently fail to diagnose a lack of convergence across all components of the model. Where the computational budget allows, we recommend using the couplings described in Section 2.8 and Kelly, Ryder and Clarté (2023) to confidently diagnose convergence jointly across all components.

Checking model fit Since the MCMC run appears to have converged, we can now check for model misfit. Using the `Analyse data` panel, we observe in Figure 11 that points on the `Distance depth relation` plot, previously described in Figure 6, are largely above the line $y = x$ but the histograms for the data in the file only partially resemble those for synthetic data Figure 12, suggesting that the fit could be improved.

The consensus tree with catastrophes in Figure 13 shows strong support for catastrophe events on the branches above Ugaritic, with almost no rate heterogeneity elsewhere in the tree; there is a 28% posterior probability for no

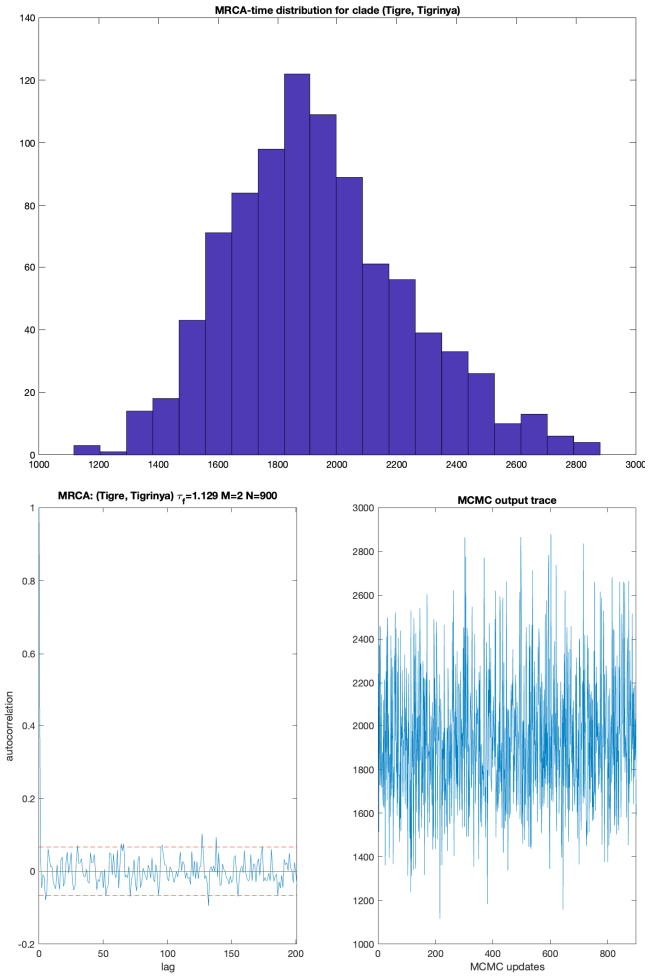

FIGURE 10. *Histogram, autocorrelation and trace plots for the age of the MRCA of Tigre and Tigrinya after a long MCMC run for the Semitic data.*

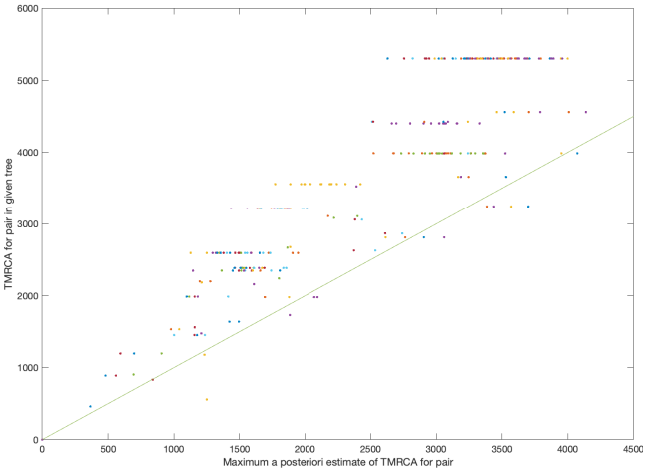

FIGURE 11. *Distance depth relation plot for the Semitic data.*

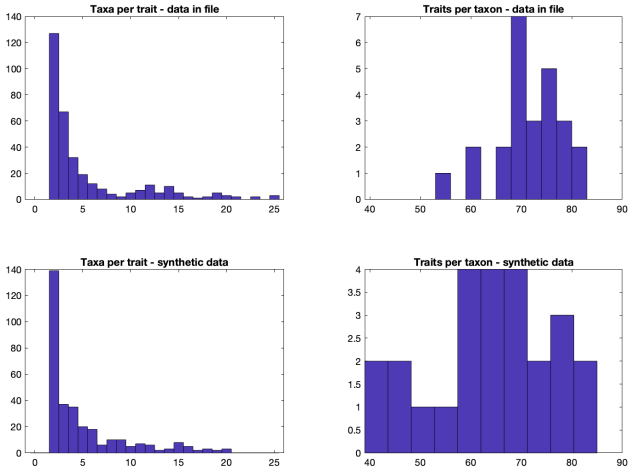

FIGURE 12. *Comparing observed and synthetic trait frequencies for the Semitic analyses.*

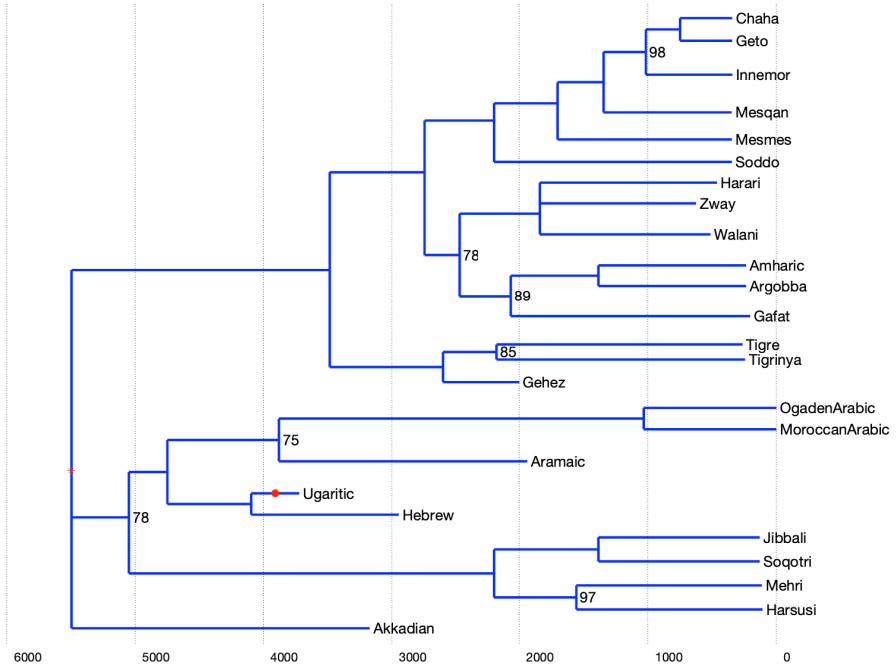

FIGURE 13. *Consensus tree with a single catastrophe after a long MCMC run for the Semitic data.*

catastrophes, compared with a 2% prior probability.[2] This indicates that there may be model misfit for Ugaritic.

We now perform a Bayesian cross-validation analysis of the six leaf age 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 `Ignore ages for clades` box, we list clade number 1, for example, 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 plot the resulting posterior ages, Figure 14 displays the resulting histogram. We see that the posterior distribution on ages overlaps with the clade age constraint of [2500, 2700], indicating support for the historically attested constraint. We repeat this for all time constraints, and find problems with the branching of Biblical Aramaic and the leaf ages of Ugaritic and Gehez. Given the evidence suggesting that they are outliers, we therefore decide to remove Ugaritic and Gehez from our final analysis. This improves the model fit, and all remaining constraints are then supported (including Aramaic branching).

Final run We now perform a final run, from which we will draw our conclusions. We do not include catastrophes since they seem unnecessary, we also omit

[2]To run a Markov chain targeting the prior, we set up a run as usual but tick the box `Omit traits listed below` and specify `1:674` to ignore all the traits in the data.

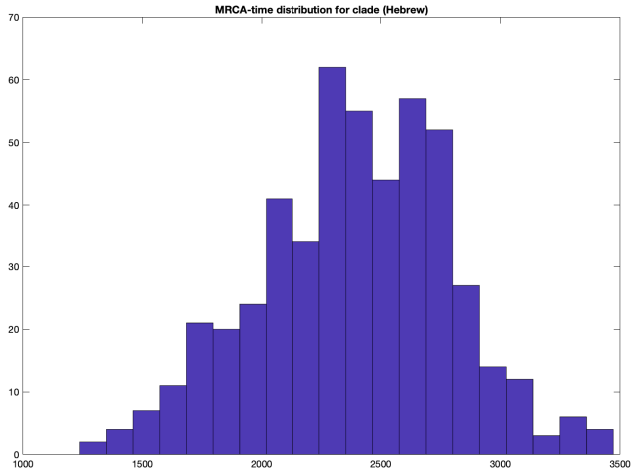

FIGURE 14. *Histogram of samples targeting the posterior distribution on the age of Hebrew after relaxing the clade constraint which restricted it to [2500, 2700].*

taxa 1 and 17 (Gehez and Ugaritic) and ignore clades 2 and 5. After starting the run, a message appears in the Matlab command window asking whether to delete or ignore Ugaritic from the central clade: we choose to delete it in this case. With this run, we build the consensus tree displayed in Figure 15 and estimate a 95% highest posterior density interval of [4354, 5624] years Before Present for the root age of the Semitic family.

References

ALEKSEYENKO, A. V., LEE, C. and SUCHARD, M. (2008). Wagner and Dollo: A Stochastic Duet by Composing Two Parsimonious Solos. *Syst. Biol.* **57** 772–784.

ATKINSON, Q. D. (2006). From species to languages: a phylogenetic approach to human prehistory, PhD thesis, The University of Auckland.

ATKINSON, Q., NICHOLLS, G., WELCH, D. and GRAY, R. (2005). From words to dates: water into wine, mathemagic or phylogenetic inference? *Trans. Philol. Soc.* **103** 193–219.

BARBANÇON, F., EVANS, S. N., NAKHLEH, L., RINGE, D. and WARNOW, T. (2013). An experimental study comparing linguistic phylogenetic reconstruction methods. *Diachronica* **30** 143–170.

BISWAS, N., JACOB, P. E. and VANETTI, P. (2019). Estimating convergence of Markov chains with L-lag couplings. In *NeurIPS* 7389–7399.

BOUCKAERT, R. R., BOWERN, C. and ATKINSON, Q. D. (2018). The origin and expansion of Pama–Nyungan languages across Australia. *Nature ecology & evolution* **2** 741–749.

BOUCKAERT, R. R. and ROBBEETS, M. (2017). Pseudo Dollo models for the evolution of binary characters along a tree. *BioRxiv* 207571.

BOUCKAERT, R., VAUGHAN, T. G., BARIDO-SOTTANI, J., DUCHÊNE, S., FOURMENT, M., GAVRYUSHKINA, A., HELED, J., JONES, G., KÜHNERT, D., DE MAIO, N., MATSCHINER, M., MENDES, F. K., MÜLLER, N. F.,

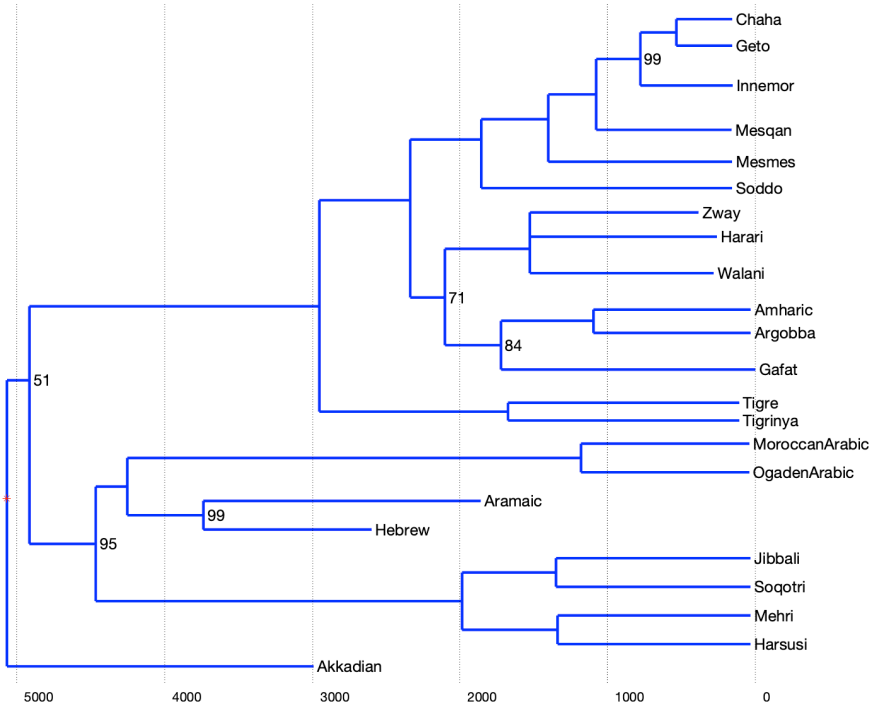

FIGURE 15. *Consensus tree with a single catastrophe in the final MCMC run for the Semitic data with Gehez and Ugaritic removed.*

Ogilvie, H. A., du Plessis, L., Popinga, A., Rambaut, A., Rasmussen, D., Siveroni, I., Suchard, M. A., Wu, C. H., Xie, D., Zhang, C., Stadler, T. and Drummond, A. J. (2019). BEAST 2.5: An advanced software platform for Bayesian evolutionary analysis. *PLOS Comput. Biol.* **15**.

Chang, W., Hall, D., Cathcart, C. and Garrett, A. (2015). Ancestry-constrained phylogenetic analysis supports the Indo-European steppe hypothesis. *Language* 194–244.

Currie, T. E., Meade, A., Guillon, M. and Mace, R. (2013). Cultural phylogeography of the Bantu Languages of sub-Saharan Africa. *Proceedings of the Royal Society B: Biological Sciences* **280** 20130695.

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

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

Geyer, C. J. (1992). Practical Markov Chain Monte Carlo. *Stat. Sci.* **7** 473–483.

Gould, S. J. (1970). Dollo on Dollo's Law: Irreversibility and the Status of Evolutionary Laws. *J. Hist. Biol.* **3** 189–212.

Gray, R. D. and Atkinson, Q. D. (2003). Language-tree divergence times support the Anatolian theory of Indo-European origin. *Nature* **426** 435–439.

Gray, R. D., Drummond, A. J. and Greenhill, S. J. (2009). Language phylogenies reveal expansion pulses and pauses in Pacific settlement. *Science* **323** 479–483.

Greenhill, S. J., Currie, T. E. and Gray, R. D. (2009). Does horizontal transmission invalidate cultural phylogenies? *Proc. R. Soc. B* **276** 2299–2306.

Greenhill, S. J., Heggarty, P. and Gray, R. D. (2020). Bayesian phylolinguistics. *The handbook of historical linguistics* **2** 226–253.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57** 97–109.

Heggarty, P., Anderson, C., Scarborough, M., King, B., Bouckaert, R., Jocz, L., Kümmel, M. J., Jügel, T., Irslinger, B., Pooth, R., Liljegren, H., Strand, R. F., Haig, G., Macák, M., Kim, R. I., Anonby, E., Pronk, T., Belyaev, O., Dewey-Findell, T. K., Boutilier, M., Freiberg, C., Tegethoff, R., Serangeli, M., Liosis, N., Stroński, K., Schulte, K., Gupta, G. K., Haak, W., Krause, J., Atkinson, Q. D., Greenhill, S. J., Kühnert, D. and Gray, R. D. (2023). Language trees with sampled ancestors support a hybrid model for the origin of Indo-European languages. *Science* **381** eabg0818.

Jacob, P. E., O'Leary, J. and Atchadé, Y. F. (2020). Unbiased Markov chain Monte Carlo methods with couplings. *J. Roy. Statist. Soc. B* **82** 543–600.

Kelly, L. J. (2016). A Stochastic Dollo model for lateral transfer, PhD thesis, University of Oxford.

Kelly, L. J. and Nicholls, G. K. (2017). Lateral transfer in Stochastic Dollo models. *Ann. Appl. Stat.* **11** 1146–1168.

Kelly, L. J., Ryder, R. J. and Clarté, G. (2023). Lagged couplings diagnose Markov chain Monte Carlo phylogenetic inference. *Ann. Appl. Stat.* **17** 1419–1443.

Kitchen, A., Ehret, C., Assefa, S. and Mulligan, C. J. (2009). Bayesian Phylogenetic Analysis of Semitic Languages Identifies an Early Bronze Age Origin of Semitic in the Near East. *Proc. R. Soc. B* **276** 2703–2710.

Koch, M. (2016). *Geschichte der gesprochenen Sprache von Bayerisch-Schwaben Phonologische Untersuchungen mittels diatopisch orientierter Rekonstruktion*. BiblioScout.

Koile, E., Greenhill, S. J., Blasi, D. E., Bouckaert, R. and Gray, R. D. (2022). Phylogeographic analysis of the Bantu language expansion supports a rainforest route. *Proc. Natl. Acad. Sci. U.S.A.* **119** e2112853119.

Lewis, P. O. (2001). A Likelihood Approach to Estimating Phylogeny from Discrete Morphological Character Data. *Syst. Biol.* **50** 913–925.

Maddison, D. R., Swofford, D. L. and Maddison, W. P. (1997). Nexus: An Extensible File Format for Systematic Information. *Syst. Biol.* **46** 590–621.

McPherson, A., Roth, A., Laks, E., Masud, T., Bashashati, A., Zhang, A. W., Ha, G., Biele, J., Yap, D., Wan, A., Prentice, L. M., Khattra, J., Smith, M. A., Nielsen, C. B., Mullaly, S. C., Kalloger, S., Karnezis, A., Shumansky, K., Siu, C., Rosner, J., Chan, H. L., Ho, J., Melnyk, N., Senz, J., Yang, W., Moore, R., Mungall, A. J., Marra, M. A., Bouchard-Côté, A., Gilks, C. B., Huntsman, D. G., McAlpine, J. N., Aparicio, S. and Shah, S. P. (2016). Divergent modes of clonal spread and intraperitoneal mixing in high-grade serous ovarian cancer. *Nature Genet.* **48** 758–767.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.* **21** 1087–1092.

Neureiter, N., Ranacher, P., Efrat-Kowalsky, N., Kaiping, G. A., Weibel, R., Widmer, P. and Bouckaert, R. R. (2022). Detecting contact in language trees: a Bayesian phylogenetic model with horizontal transfer. *Humanit. Soc. Sci. Commun.* **9** 1–14.

Nicholls, G. K. and Gray, R. D. (2008). Dated Ancestral Trees from Binary Trait Data and its Application to the Diversification of Languages. *J. Roy. Statist. Soc. B* **70** 545–566.

Nicholls, G. K. and Ryder, R. J. (2011). Phylogenetic Models for Semitic Vocabulary. In *Proceedings of the International Workshop on Statistical Modelling* (D. Conesa, A. Forte, A. López-Quílez and F. Muñoz, eds.) 431–436.

Pagel, M., Meade, A. and Barker, D. (2004). Bayesian estimation of ancestral character states on phylogenies. *Syst. Biol.* **53** 673–684.

Rambaut, A., Drummond, A. J., Xie, D., Baele, G. and Suchard, M. A. (2018). Posterior Summarization in Bayesian Phylogenetics Using Tracer 1.7. *Syst. Biol.* **67** 901–904.

Ronquist, F., Teslenko, M., Van Der Mark, P., Ayres, D. L., Darling, A., Höhna, S., Larget, B., Liu, L., Suchard, M. A. and Huelsenbeck, J. P. (2012). MrBayes 3.2: efficient Bayesian phylogenetic

inference and model choice across a large model space. *Syst. Biol.* **61** 539–542.

Ryder, R. J. (2010). Phylogenetic Models of Language Diversification, PhD thesis, University of Oxford.

Ryder, R. J. and Nicholls, G. K. (2011). Missing Data in a Stochastic Dollo Model for Binary Trait Data, and its Application to the Dating of Proto-Indo-European. *J. Roy. Statist. Soc. C.*

Sagart, L., Jacques, G., Lai, Y., Ryder, R. J., Thouzeau, V., Greenhill, S. J. and List, J.-M. (2019). Dated language phylogenies shed light on the ancestry of Sino-Tibetan. *Proc. Natl. Acad. Sci. U.S.A.* **116** 10317–10322.

Suchard, M. A., Lemey, P., Baele, G., Ayres, D. L., Drummond, A. J. and Rambaut, A. (2018). Bayesian phylogenetic and phylodynamic data integration using BEAST 1.10. *Virus Evol.* **4**.

Thomson, R. C., Plachetzki, D. C., Mahler, D. L. and Moore, B. R. (2014). A critical appraisal of the use of microRNA data in phylogenetics. *Proc. Natl. Acad. Sci. U.S.A.* **111** E3659-E3668.