Delphy Whitepaper, version 1

The Delphy Authors

1 Jul 2024

1 Introduction

Over the past decade, next-generation sequencing has enabled pervasive sequencing of pathogen genomes. During the recent pandemic, for example, GISAID accumulated over 16 million SARS-CoV-2 genomes [1, 2]. The potential for high-resolution surveillance of future outbreaks is thus clear, but so is the challenge of managing the resulting deluge of genomic data.

Phylogenetics is the fundamental tool we use to organize and interpret pathogen genome sequences. By inferring plausible histories that link point observations across time and space, phylogenetic trees uncover understanding that is hard to glean from the sequences themselves. Examples include: (a) classifying samples into distinct lineages with common properties [3]; (b) dating the origin of epidemics or lineages within them [4, 5, 6]; (c) providing early warning signs of new pathogen variants with consequential effects [7]; (d) discerning whether an outbreak is growing because of recurrent zoonotic spillovers of because of human-to-human transmission [4, 8]; (e) revealing the underlying community and geographic transmission routes of pathogens [8, 9, 10, 11]; and (f) untangling whether unexplained growth is due to the appearance of a new variant with increased fitness or is instead driven by exogenous conditions changing and allowing all existing variants to grow in tandem [12]. These insights inform the public health interventions that may or may not contain and suppress outbreaks. At a more granular level, phylogenetic trees are also at the heart of modern outbreak reconstruction methods [13, 14, 15].

Among the approaches to phylogenetics, Bayesian phylogenetics, as embodied by tools like BEAST [16], BEAST2 [17] and Mr. Bayes [18], is the gold standard against which other approximate approaches, like maximum-likelihood (ML) [20, 21] and parsimony [22, 23], are judged. Bayesian phylogenetics fully embraces and quantifies the uncertainty in tree reconstructions, and allows for simultaneous inference of important covariates like geographic spread [8] or changes in viral population through time [19]. However, existing Bayesian phylogenetics tools are so computationally expensive that current standard practice for large datasets is to infer an approximate tree topology first, then post-process it in ad hoc ways to infer uncertainties and covariates [13, 24, 25, 26]. Both real Bayesian phylogenetics analyses and its approximations are also sufficiently complicated to carry out that they are essentially inaccessible to most epidemiologists and public health bodies, remaining instead the purview of a handful of specialized research groups worldwide.

In this work, we aim to make Bayesian phylogenetics fast, accurate, scalable and accessible. Our key goals are: (a) allowing any epidemiologist anywhere in the world to analyze their own data using state-of-the-art methods, without any special training; (b) being able to take raw aligned sequences and generate publication-quality figures within an hour, thus making Bayesian phylogenetics a standard part of the real-time response toolkit of public health authorities worldwide; and (c) enabling efficient scaling to match the increasing rate at which genomic sequences are likely to be produced in the future, without sacrificing accuracy to do so.

With these goals at the forefront, we have reimagined and rebuilt every portion of the Bayesian phylogenetics pipeline from scratch. The result is Delphy, a new tool for Bayesian phylogenetics that meets all of our stated goals—fast, scalable, accurate and accessible. As UShER [22] and matOptimize [23] did for parsimony methods, Delphy derives its speed and scalability from a reframing of the model behind Bayesian phylogenetics in terms of trees with explicit mutation events; as a result, a Delphy run can produce results in minutes, instead of days or weeks. Unlike approximate methods, Delphy implements Bayesian phylogenetics

exactly, thus preserving its accuracy. Finally, we have leveraged recent technological progress in modern web browsers to distribute Delphy as if it were a simple web page from the user's point of view (currently accessible at `https://delphy.fathom.info`); by removing any need for compilation and installation, and presenting a streamlined and interactive user interfacing for inputting data, monitoring runs, and allowing for in-depth exploration and exporting of the results, we believe that Delphy will be accessible to most epidemiologists and public health bodies worldwide, without any special training.

We are working on a preprint to describe our methods in detail [28]. This whitepaper outlines the main ideas and presents key benchmarks.

Delphy's source code is available at `https://github.com/broadinstitute/delphy` (core) and `https://github.com/fathominfo/delphy-web` (web interface). Delphy is developed in a collaboration between the Sabeti Lab at the Broad Institute and Fathom Information Design.

2 Organization

The remainder of this whitepaper is split into two parts. A general overview begins in Section 3, where we discuss the specific model that Delphy implements. Section 4 walks through the Delphy web interface from the perspective of a user, and serves as a brief tutorial for it. The overview concludes with Section 5, which presents a detailed comparison of results of Delphy runs vs. equivalent BEAST2 runs for three real-world datasets. Finally, following this overview, Section 6 discusses in substantial detail the key ideas underpinning Delphy's accuracy and speed.

3 Model setup

As described below, while any model suitable for Bayesian phylogenetics could in principle be implemented in Delphy, our initial release implements the following hard-coded model, suitable for tracking incipient, exponentially growing outbreaks:

- A single-partition HKY substitution model with a strict molecular clock [29].
- Optional site-rate heterogeneity using a continuous Gamma distribution [30].
- A coalescent prior for ancestry using an exponentially growing population [31, 32].
- Priors for all parameters are unchanged from the defaults in BEAUTi2 [17] given the above choices, except that we optionally allow for the mutation rate to be fixed.

As a proof-of-concept of the underlying flexibility hidden by the above hard-coded choices, we have also implemented a two-partition model with a time-irreversible trasition rate matrix, suitable for analyzing mpox sequences [8].

4 Web interface

The Delphy web interface, currently hosted at `https://delphy.fathom.info`, is designed with a strong focus on ease of use by anyone with sequence data, regardless of their additional computational skills. It also emphasizes privacy: all analysis happens on the user's machine, and after the transparent downloading of the core WebAssembly modules, no data is ever transmitted over the network. In terms of analyses, we have particularly emphasized answering many of the typical questions addressed in Bayesian phylogenetic studies with minimal effort. Finally, to allow more sophisticated users to perform their own downstream analyses, all the raw results can be exported in the same format as would be output by an equivalent BEAST2 run.

Figure 1(a) shows the landing page for Delphy. Here, a user can drag-and-drop a FASTA file with a multiple-sequence alignment of dated sequences, as follows:

```
>Sample001|...|2024-06-20
ACGTACGTACGT----NNNNACGT
>Sample002|...|2024-05-15
ACGTAAA---GTACGTACGTACNN
...
```

Each sequence in the FASTA file should have an ID line consisting of fields separated by vertical pipes ('|'). The first field should be a unique sequence ID, and the last field should be a date at day-level precision in the format `YYYY-MM-DD`. We have explicitly scoped out alignment of raw sequence assemblies, assuming our users can produce such alignments on their own (e.g., using `mafft`[1] [33]). We have also purposely kept the input format very rigid for simplicity: we assume our users can format their input data as above, manually if necessary. A present limitation is that we do not implement tip-date sampling, so uncertain dates (e.g., `2024` or `2024-06`) are not supported. A reasonable, temporary workaround is to impute dates to the middle of their uncertainty period (e.g., `2024-07-01` or `2025-06-15`, respectively). We will eventually relax this restriction.

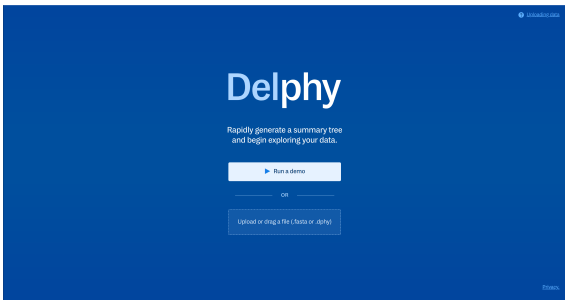
After selecting an input file, Delphy builds a rough initial tree through sequential attachment of each input sequence in turn at its most parsimonious position, and timed as if the underlying mutation rate matched that of SARS-CoV-2 (1×10^{-3} substitutions per site per year). The user is presented with this initial tree in a "Trees" panel (Figure 1(b)) and a "Play" button is prominently displayed to begin the MCMC run. We have carefully chosen all the default parameters to be suitable for analyzing the early stages of an exponentially growing outbreak (see Section 3). The default tree sampling frequency is chosen heuristically to sample mostly uncorrelated trees. A few model parameters can be tuned under the "Advanced Options" screen, but at present, most model details are hard-coded.

Once the users presses "Play", posterior samples are generated and stored at the selected frequency. The screen in the middle shows an updated view of the maximum-clade-credibility tree (MCC) of these samples: we strongly emphasize nodes with high posterior support ($> 90\,\%$ by default) and deemphasize all others to visually guide the user's eye towards the stable parts of the MCC. Simultaneously, the graphs on the right show traces and histograms of key observables. As usual in MCMC, these observables initially change rapidly during a burn-in phase, then stabilize in the production phase. By clicking on a trace, a user can select for themselves when the burn-in phase is over, and all analyses will only use the subsequent production samples. Delineating the burn-in period is *essential* to obtaining reliable statistics from the MCMC run. At any time, the user can pause and resume the run. By scrubbing the samples at the bottom left corner, a user can view the individual posterior samples, as well as see which one is providing the topology of the MCC. We envision extending this screen in the future to provide more guidance on effective sample sizes and summary statistics of many observables.

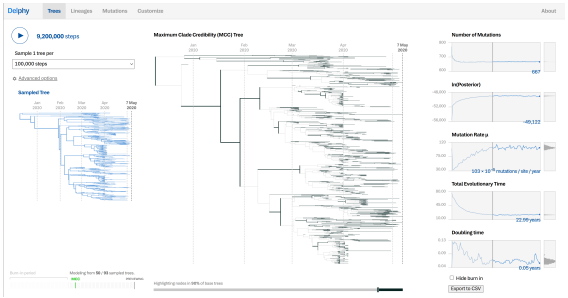
After enough samples have been taken, as evidenced by traces with fluctuation times much smaller than the total number of steps, a user can stop the MCMC run and proceed to the "Lineages" panel (Figure 1(c)) to begin exploring and summarizing the posterior distribution. They can hover over any tip to see its ID, or alternately, search for a tip given its ID with the search box at the top. They can also hover over any inner node in the MCC[2] to quickly see its number of descendant tips, its posterior support and its time distribution. Each time distribution shows the median time, and on hovering, can also reveal the location of any percentile of the distribution, with the lower and upper bounds of the 95% highest-posterior density (HPD) range highlighted automatically. Clicking on a node selects it durably, allowing a user to analyze the relationships between different nodes: in this way, a user can define and analyze the natural lineages present in their genomic data. The tree root is always selected, and when two other nodes are selected where one is not ancestral to the other, their MRCA on the MCC is also automatically selected. Delphy also shows the

[1] For example, the alignment for the benchmarks presented in Figures 2 and 3 was prepared with the following command line: `mafft --thread -1 --auto --addfragments <unaligned_genomes.fasta> --keeplength <ref.fasta>`.

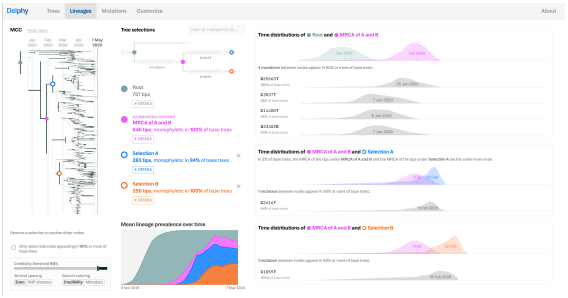
[2] An inner mode of the MCC represents the MRCA of of its descendants tips in each posterior tree. In the MCC tree, its time is the mean time over those trees for which the descendant tips form a monophyletic clade, which is the most common convention for MCC times. In all other places, including the tMRCA distribution, it is the time of the MRCA of those descendant tips, regardless of whether they form a monophyletic clade [34].



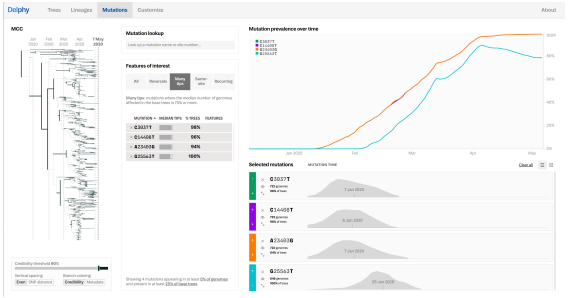
(a) Landing page



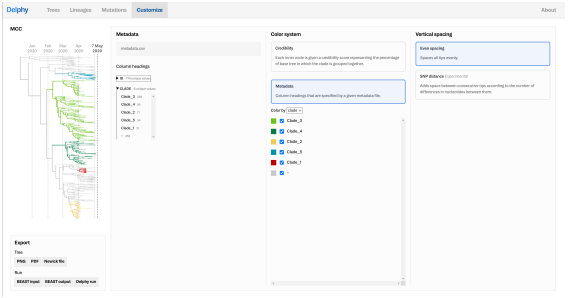
(b) Trees panel



(c) Lineages panel



(d) Mutations panel



(e) Customize panel

Figure 1: Delphy Web interface. See text for details

mutations and their associated time distributions between all selected nodes[3], both as a schematic at the top, as well as in a detailed panel on the right-hand side. At the bottom, Delphy also calculates the probability distribution for the nearest selected ancestor of a random member of the population at a given time ("mean lineage distribution"). In the limit of very densely sampled datasets, this reduces to the counts of active branches downstream of the selected nodes at particular times, but in areas where sampling is lower, the underlying coalescent model and population curve also informs the lineage distribution. The "Details" inset in each selection shows the 95% confidence range (2.5% to 97.5% percentiles) for these distributions for each lineage.

A complementary view of the data is provided in the "Mutations" panel (Figure 1(d)). There, any mutation in the dataset can be queried for its time distribution and its prevalence. Hovering over a single mutation highlights the nodes in the MCC above which the mutation appears, and highlights the 95% confidence range for the prevalence curves. By default, Delphy automatically selects high-confidence mutations that rise to high prevalence, and provides users a convenient way to easily pick out other interesting mutations (e.g., recurring mutations, reversals).

The "Lineages" and "Mutations" panels are linked: clicking on a mutation in the Lineages panel brings up that mutation in the Mutations panel, whereas expanding a mutation in the Mutations panel and clicking on the panel corresponding to its location on the tree highlights the corresponding lineage in the "Lineages" panel.

Finally, we have implemented a crude but effective mechanism for decorating tips with discrete metadata labels and propagating those labels up the MCC using parsimony. These metadata labels are used to color the branches of the MCC, and their inferred values can also be seen in the "Details" insets of the "Lineages" panel. On opening the "Customize" panel (Figure 1(e)), a user can drag-and-drop a metadata comma-separated-values (`.csv`) or tab-separated-values (`.tsv`) file with such metadata. The file can be assembled by hand or exported from a spreadsheet, and looks like this:

```
id,Place,SymptomsCough,SymptomsFever
Sample001,Brussels,Y,N
Sample002,Boston,N,Y
...
```

There must be one column labeled `id` that matches the unique IDs of each sequence in the input data. Thereafter, every other column is just treated as a category of free-form, discrete metadata labels. In Figure 1(e), the labels correspond to particular clades studied in [35], but other typical uses include geographic labels (to visualize the spread of a pathogen in space) and hosts (to pinpoint host jumps). A more detailed example of using metadata labels to this effect can be accessed at `https://delphy.fathom.info/?mpox2024-post-spillover-nigeria-only-final-with-metadata.dphy`, part of our efforts to understand the origins of the recent mpox epidemic [8].

Because metadata propagation is done by a naïve parsimony algorithm on the MCC, it is only suitable for categories where transitions between categories happen at comparable rates, all of which are much lower than once per typical branch length. Moreover, the phylogenetic uncertainty inherent in the posterior distribution is not currently taken into account. Finally, when parsimonious assignments are ambiguous, Delphy disambiguates them at random. This happens every time a user switches panels or enables/disables the inclusion of metadata labels, and results in some uncertain details of the metadata assignments changing randomly as a user navigates the interface. We intend to improve our treatment of metadata in the medium-term, but already find the current naïve treatment useful enough to include it in Delphy's initial release.

Finally, at the bottom left of the "Customize" panel, a user can export the final results in various forms. The MCC, as colored by the current metadata selection, can be exported as a PNG image or a PDF file, suitable either for direct inclusion in reports and papers, or for post-processing in a vector graphics program like Inkscape or Adobe Illustrator. The user can also export the MCC in Newick tree format for more fine-tuned figure preparation, e.g., with FigTree [36]. For power users, we provide the option to export a

[3]These summarize the mutations present over all posterior trees between corresponding nodes, defined as the MRCAs of the descendant tips in each posterior tree.

BEAST2 XML input file suitable for BEAST2 v2.6.2 that will execute a statistically identical run as the one performed by Delphy (detailing every single choice and parametrization of the model), as well as export the results of the Delphy run in the same format that would be produced by that BEAST2 run. Hence, downstream tools like Tracer [37], LogAnalyser [16], LogCombiner [16] and Baltic [39], among others, can be used to further examine the results of the run. To conclude, Delphy can also save every detail of the current run in a compact binary form (`.dphy`) that can be reloaded quickly on a subsequent use of the web interface, which is useful for sharing run results with others, as well as for restoring the results of a long run if Delphy unexpectedly crashes. This format is currently undocumented and subject to change.

5 Benchmarks

An important feature of Delphy is its ability to produce a corresponding BEAST2 input XML file for the same input data set, prior choices and form of posterior. Analogously, Delphy can export its results in two log files, one for parameters and one for trees, that have the same format as would be produced by a BEAST2 run using this XML file. Hence, it is possible to easily compare Delphy vs. BEAST2 in terms of both results and running times with minimal effort. We emphasize that because the model setup uses few special features of BEAST2, the benchmarks are really testing the performance of the underlying BEAGLE library [40], which is shared by BEAST, BEAST2 and Mr. Bayes.

We use three real-world datasets for these benchmarks:

- The SARS-CoV-2 dataset analyzed by Lemieux et al in 2021 [35] (the subset that is publicly available in GenBank). It has 757 sequences aligned to the 29,903-base reference genome `NC_045512.2` (included), spanning the period 2020–03–03 to 2020–05–09. On average, each sequence is missing 1.8% of its bases, with a range of 0 to 5285 bases (0.0–18.0%).

- The Zika dataset analyzed by Metsky et al in 2017 [5]. It has 167 sequences aligned to the 10,807-base reference genome `KX197192.1` (not included here nor in [5]), spanning the period 2014-12-12 to 2016-10-12. On average, each sequence is missing 17.9% of its bases, with a range of 535 to 8233 bases (5.0–76.2%).

- The Ebola dataset analyzed by Gire et al 2014 [4]. It has 81 sequences aligned to the 18,959-base reference genome `KJ660346.2` (included), spanning the period 2014-03-17 to 2014-06-18. On average, each sequence is missing 0.3% of its bases, with a range of 1 to 519 bases (0.0–2.7%).

While larger datasets exist, especially for SARS-CoV-2, the above three datasets are representative of the early stages of an exponentially growing outbreak, and hence should be well-described by the model we currently implement in Delphy (Section 3).

We have published at `https://www.github.com/broadinstitute/delphy-2024-paper-data` an end-to-end set of scripts for building and running these benchmarks from scratch using publicly available data. This repository also includes the raw outputs of all Delphy and BEAST2 runs, and the scripts to produce all of the analyses and plots described below.

Figures 2 and 3 show the results for the Lemieux et al 2021 SARS-CoV-2 sequences, respectively with and without site-rate heterogeneity. The MCCs produced by BEAST2[4] and Delphy clearly match closely (compare also to Figure 3A in [35]). Note that large vertical rearrangements in otherwise equivalent MCCs around inner nodes with low posterior support are common and not significant. The histograms for many key observables (tree height, total branch length, HKY model parameters, population curve parameters) also match closely. Since BEAST2 uses discrete Gamma categories for modeling site-rate heterogeneity [41] while Delphy uses a continuous Gamma distribution (see Section 6.6), there is a major difference in the distributions for the site-rate heterogeneity parameter α. Conversely, when site-rate heterogeneity is disabled, the trees

[4] All MCCs in this section were derived using BEAST2's TreeAnnotator with the option `-heights ca`: inner node times are the mean time of the MRCA of the descendant tips, whether or not they form a monophyletic clade. This was done to match how Figure 3A in [35] was produced.

and parameters sampled by both BEAST2 and Delphy should be statistically indistinguishable, and this is borne out in the second Figure.

Figures 4, 5, 6 and 7 show the corresponding results for the Zika and Ebola datasets, with and without site-rate heterogeneity (compare to Figure 2b in [5] and Figure 3B in [4], respectively). Note also that for the Ebola dataset with site-rate heterogeneity, the parameter α takes much higher values than in the Zika and SARS-CoV-2 datasets, indicating much less spread in site-specific rates. Hence, the discrepancy between a continuous Gamma and a discrete Gamma approximation is much less pronounced.

All benchmarks compare the running times of Delphy and BEAST2 in terms of "Effective Samples per Minute", i.e., the rate at which the respective MCMC chains produce statistically independent samples, estimated from the traces of all the different observables using LogAnalyser [17]. The insets show the implied speed-up resulting from Delphy's use of an explicit-mutation representation. As explained in Section 6.6, the use of K-category discrete Gamma distribution for site-rate heterogeneity traditionally makes the core of the MCMC calculation about K times more expensive, whereas Delphy's approach of using a continuous Gamma distribution makes the running times relatively insensitive to whether site-rate heterogeneity is or is not enabled; hence the higher speed-ups when site rate heterogeneity is enabled.

Also noteworthy is that speedups for Delphy are significantly larger for larger datasets, owing to several effects. First and foremost, we expect the cost of BEAST2 runs to scale roughly quadratically in the number of samples. In order to keep the number of steps per sample constant (and thus maintain a roughly constant effective sample size in a run), the total number of steps needs to increase proportionally. On top of that, there is a parallel effect where the number of site patterns increases whenever an additional sample introduces an extra mutation into the tree in a previously unmutated site [23]. In contrast, we expect the cost of Delphy runs to scale roughly linearly in the number of samples. A second effect is that as trees grow, the depth of any given node grows; whereas the cost of Felsenstein pruning is proportional to the absolute depth of the grafting point in an SPR move, the cost of calculating posterior ratios in Delphy is independent of this depth. Finally, larger datasets parallelize better because proportionally more time is spent in local moves, which can be parallelized, thus amortizing the cost of global moves, which must be executed serially (see Section 6.7 below). For example, the Delphy runs for the 81-sequence Ebola dataset can only sustain a load of just over 1 vCPU in the test machines (resulting in ~ 0.5 million steps per second), whereas the SARS-CoV-2 runs can keep 3–4 vCPUs in use (resulting in ~ 2.5 million steps per second). The implementation of the serial part of the MCMC in Delphy is currently particularly inefficient, and this impacts the Ebola runs most seriously. We expect to improve this aspect in future Delphy releases.

We stress that performance comparisons are quite sensitive to the exact options used to compile and run all the software. For example, Delphy running as a WebAssembly module inside a web browser is typically 2-3 times slower than Delphy running as a native command-line program; both Delphy and BEAST2's performance depends on the number of threads/cores used; and the underlying BEAGLE library could be compiled with or without GPU support. However, we believe that the speed-ups of several orders of magnitude demonstrated here for larger datasets are sufficient in magnitude to be robust with respect to such fine-tuning. For completeness: all runs were performed on an AWS `c5a.2xlarge` instance with 8 vCPUs and 16 GB of RAM; we used the native command-line version of Delphy; we ran BEAST2 with the maximum number of threads (`-threads -1`); and we did not compile BEAGLE with GPU support, since we did not observe appreciable speedups from GPUs in these datasets, in line with the experience of [35] (private communication).

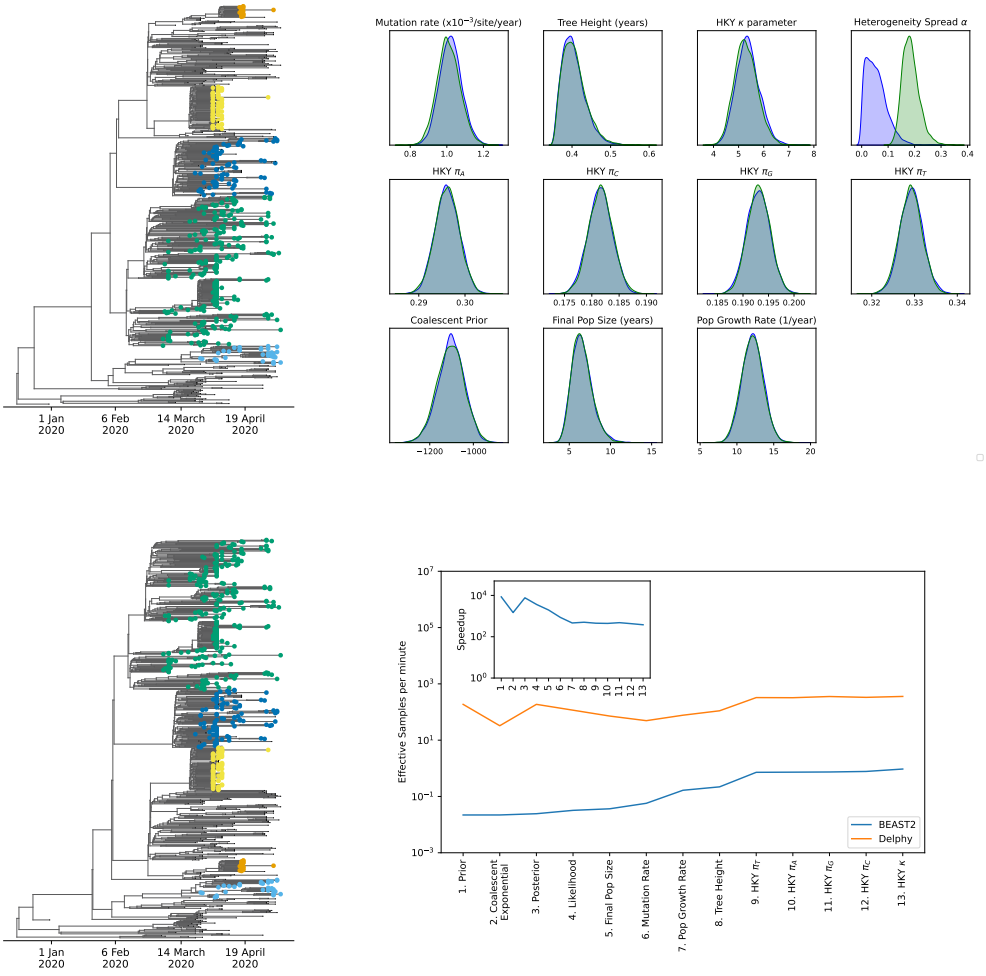


Figure 2: Delphy vs BEAST2, SARS-CoV-2 data [35], with site-rate heterogeneity. Top left: MCC calculated from Delphy run. Bottom left: MCC calculated from BEAST2 run. Compare both to Figure 3A in [35]. Top right: Histograms of keys observables from Delphy run (green) and BEAST2 run (blue). Discrepancies in the site-rate heterogeneity parameter α arise from a continuous-Gamma model (Delphy) vs. a discrete 4-category approximation (BEAST2). Bottom right: Runtime efficiency of Delphy vs BEAST2.

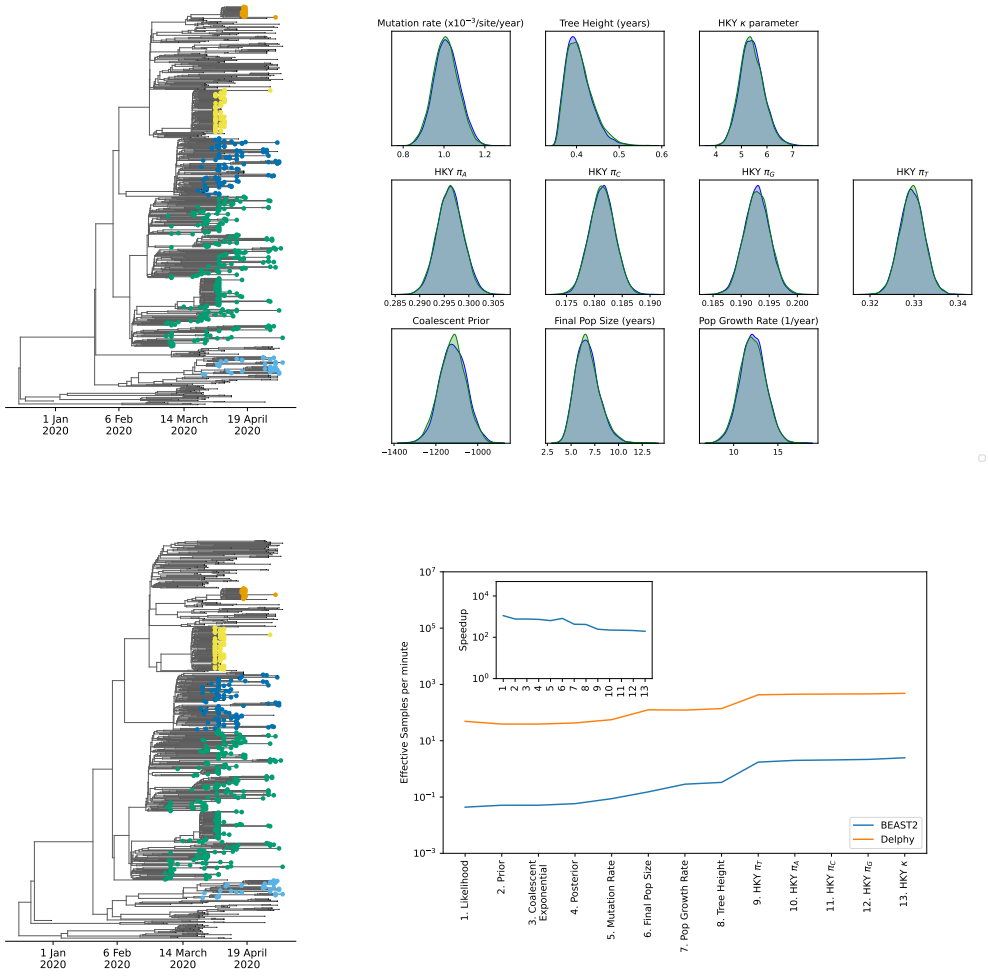


Figure 3: Delphy vs BEAST2, SARS-CoV-2 data [35], without site rate heterogeneity. See Fig 2 for panel descriptions.

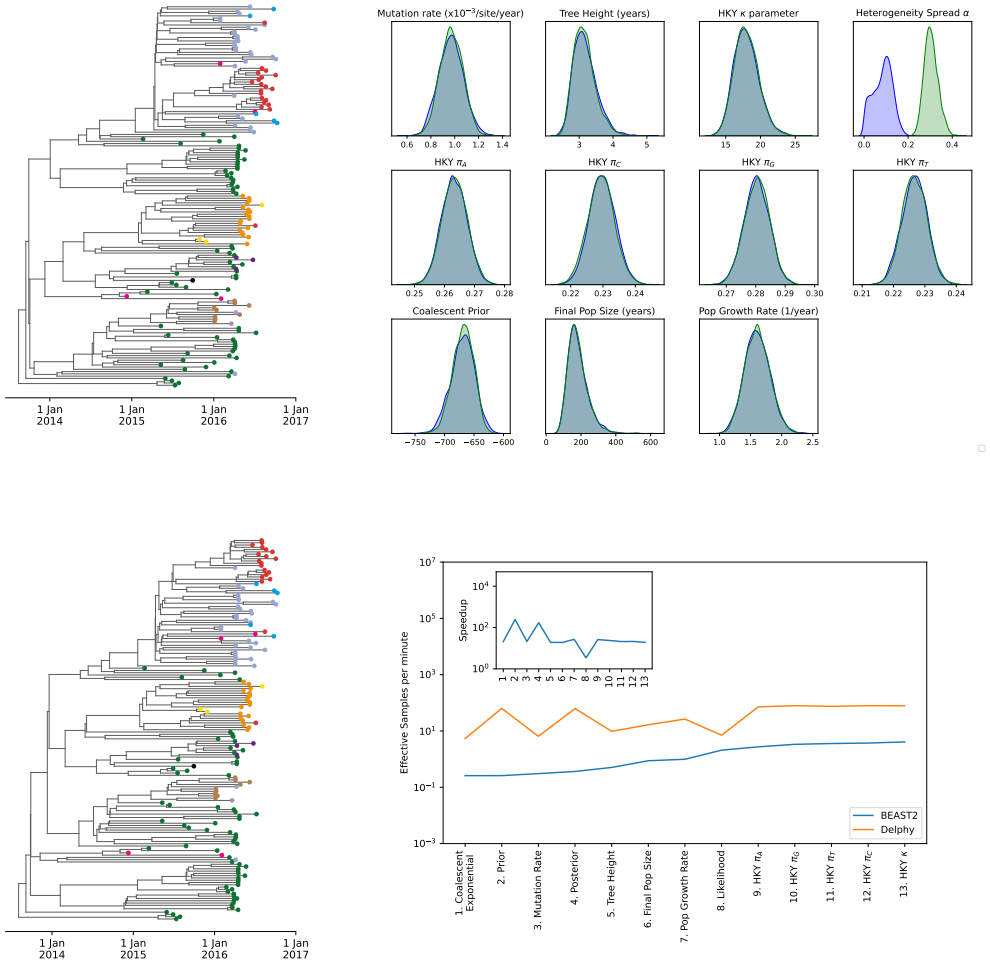


Figure 4: Delphy vs BEAST2, Zika data [5], with site-rate heterogeneity. See Fig 2 for panel descriptions. Colors as in Figure 2b in [5].

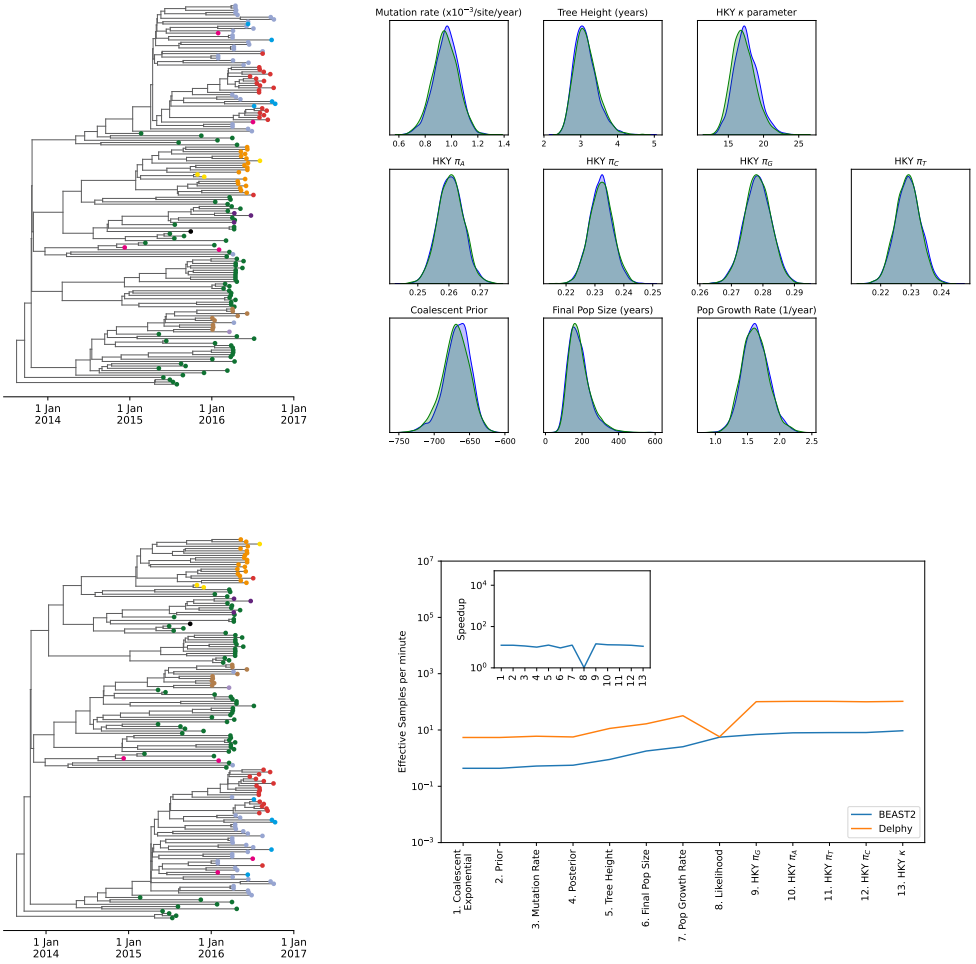


Figure 5: Delphy vs BEAST2, Zika data [35], without site rate heterogeneity. See Fig 2 for panel descriptions. Colors as in Figure 2b in [5].

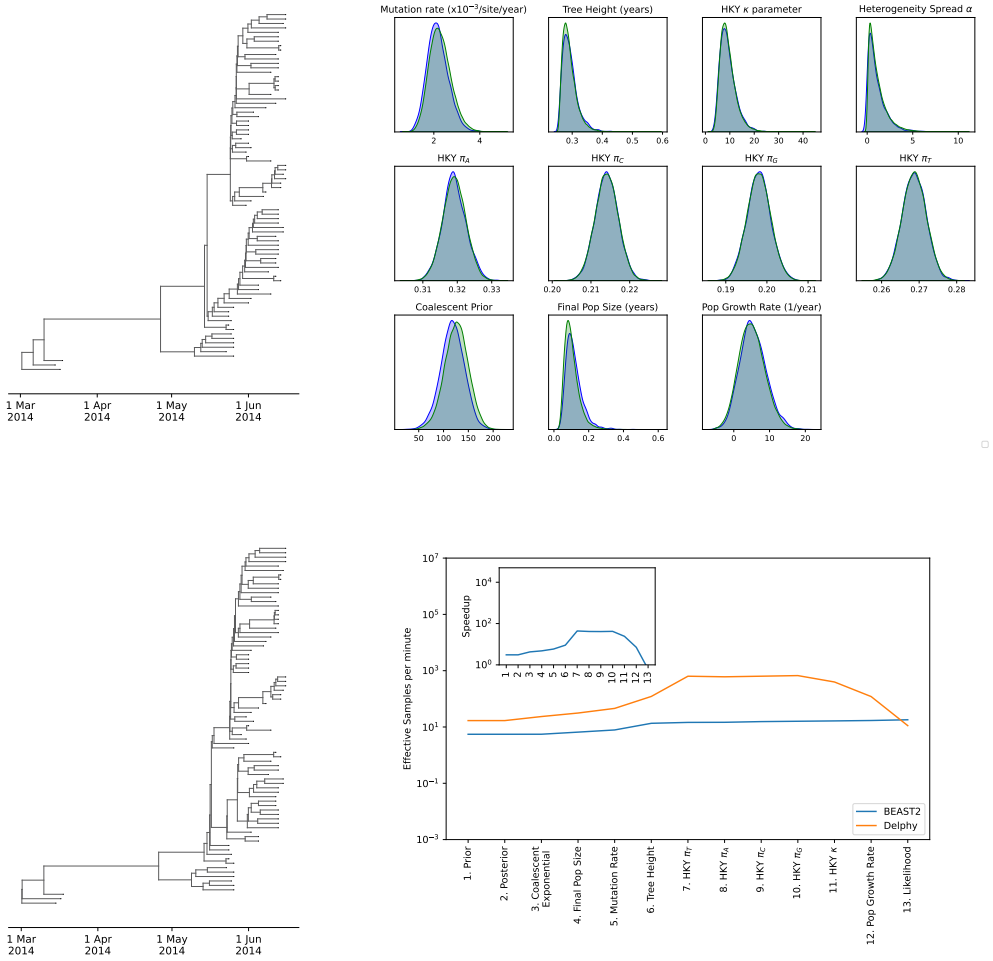


Figure 6: Delphy vs BEAST2, Ebola data [4], with site-rate heterogeneity. See Fig 2 for panel descriptions. Compare to Figure 3B in [4].

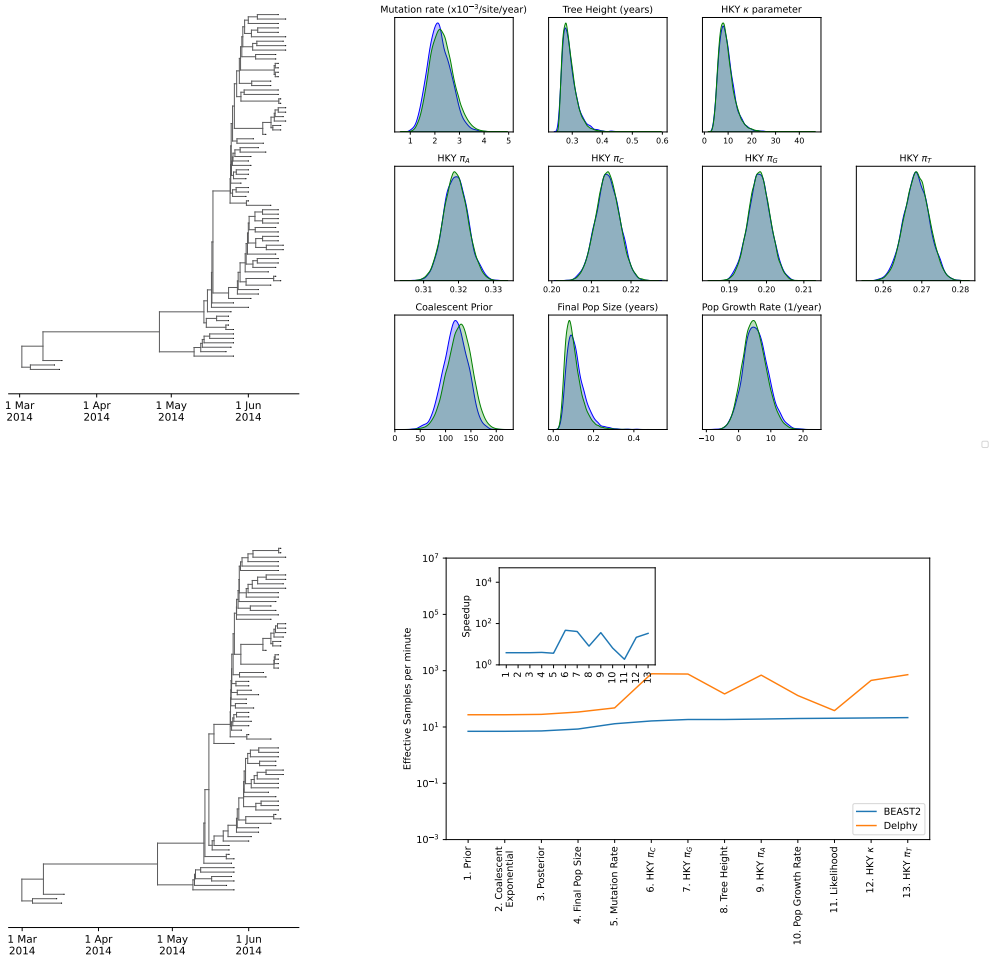


Figure 7: Delphy vs BEAST2, Ebola data [4], without site rate heterogeneity. See Fig 2 for panel descriptions. Compare to Figure 3B in [4].

6 Internals

Having discussed what Delphy calculates, how users might typically interact with Delphy, and how the results compare to equivalent BEAST runs, we now proceed to discuss the internals of Delphy. We begin in Section 6.1 with a description of the key idea of representing trees as Explicit Mutation-Annotated Trees (EMATs), followed in Section 6.2 with an explicit rendition of the posterior distribution sampled in a Delphy run. Section 6.3 discusses how typical SPR moves used in Bayesian phylogenetics are adapted to the explicit-mutation representation, whereas Section 6.4 shows how this representation enables a novel and effective variation on SPR moves, which we call mutation-directed SPR moves (mdSPR). Section 6.5 proceeds to global MCMC moves and how they can be very efficiently implemented on EMATs. An unusual aspect of Delphy, covered in Section 6.6, is its use of a continuous Gamma distribution to model site-rate heterogeneity. Section 6.7 presents the basic framework Delphy uses to parallelize the MCMC run, while Section 6.8 explains how we overcome the biggest obstacle to do this, namely, parallelizing the coalescent prior. Finally, Section 6.9 explains how we have streamlined the calculation of MCCs to allow online updates throughout an MCMC run and scale them to much larger trees in the future.

6.1 Explicit Mutation-Annotated Trees

Inspired by the Mutation-Annotated Trees (MATs) introduced by UShER [22] and matOptimize [23] to build parsimonious phylogenetic trees, we introduce *Explicit Mutation-Annotated Trees (EMATs)* (see Figure 8). An EMAT is a timed binary tree where each leaf (tip) represents a dated sample, and each inner node represents the most recent common ancestor of its children. Every other point in the tree represents some species along the lineage that connects a node to its parent. Compared to UShER's MATs, an EMAT has timed nodes and explicitly timed mutations. Less importantly, it allows the same site to be mutated multiple times along a single branch, though this is rare.

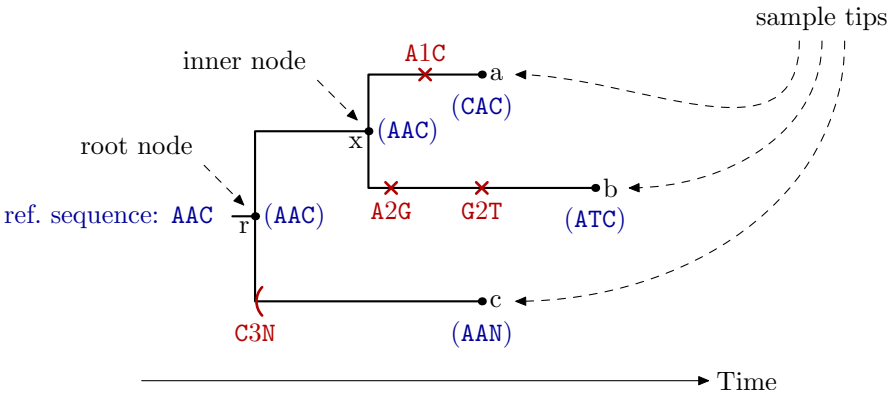


Figure 8: An Explicit Mutation-Annotated Tree (EMAT). The nodes a, b and c are dated tips, whose sequences were determined experimentally. The node x is an inner node, the most recent common ancestor of a and b. The root node r is the most recent common ancestor of all the tips. An EMAT explicitly encodes the reference sequence above the root, the mutation events (red crosses) and the missations (red parentheses). From this representation, the sequence of the species represented at any point in the tree is determined implicitly (shown in parentheses in blue below every node).

All tips represent a sample with a known sequence, although that sequence may have missing data and/or gaps. As usual, all sequences are assumed to be aligned onto an L-site reference, and the sequences specify for each site ℓ ranging from 1 to L one of: (a) the state (`A`, `C`, `G`, `T`) at that site; or (b) the fact that that site is in an alignment gap ('`-`') or was not (reliably) sequenced ('`N`'). As usual, we do not differentiate between gaps and missing data. We also do not distinguish ambiguous base calls (e.g., `Y` = `C` or `T`) and missing sites

(N); in practice, this is not a significant limitation. The entire sequence is encoded implicitly in terms of "mutations" and "missations" as follows.

Each branch (i.e., the segments that connect a node to its parent) is decorated with mutations, which are tuples (t, i, a, ℓ, b) that represents a change in the sequence at site ℓ: the sequence of the species immediately ancestral to that at time t on the branch ending at node i has a state a at site ℓ, while that of the species immediately descended from it has a state b there. Every species on a path on the tree that does not pass through a mutation has the same sequence. Viewed as a whole, an EMAT thus encodes a plausible genealogy of the sample sequences and the exact history of mutations (when and where) that produced the observed set of distinct sampled sequences. Every point on the tree has a specific sequence, but this implicit delta encoding in terms of mutations is particularly efficient. Only a single complete sequence needs to be represented explicitly: we use a reference sequence close to the sequence of the root node.

Each branch is also decorated at its start with "missations", in analogy with mutations. A missation is a tuple (i, a, ℓ) that signals that every species below the starting point of branch i, including all downstream tips, are missing data at site ℓ; in the species immediately ancestral to the missation, the state of site ℓ is a. Given an arbitrary point x on the tree, we define $\xi(x)$ as the set of sites for which at least one tip downstream of x has sequence data. We use $\xi(x)$ to implement our strategy for dealing with missing data, which we call "N-pruning" (see below). By demanding that missations be placed at the highest possible point on a tree, we ensure a unique set of missations for a given set of sequences and tree topology. This is needed for correct sampling: each possible way of imputing all the missing data is thus associated with exactly one and only one EMAT. Partly inspired by MAPLE [21], missations are actually encoded as contiguous ranges of missing sites (since such groupings are very common in real sequencing data) and the ancestral state is encoded only in terms of its difference from the reference sequence. This representation remains compact even in the face of long sequences with many large gaps (e.g., as is typical in mpox sequences).

In light of the success of UShER, it should be clear that the above representation allows for a very efficient encoding of the sequences in a genomic epidemiology dataset, including the portions of the sequences that are missing. As sketched below, it also allows for very efficient calculations and Monte Carlo moves.

6.2 Posterior distribution

Formally, Delphy samples trees $\mathcal{T}$ and model parameters $\boldsymbol{\theta}$ from a posterior distribution $P(\mathcal{T}, \boldsymbol{\theta})$ that depends on the input genomes' dates and sequences and some prior assumptions. Specifically,

$$P(\mathcal{T}, \boldsymbol{\theta}) \propto \left\{ \prod_{\ell \in \xi(r)} \pi^{(\ell)}_{s_r^{(\ell)}} \right\} \cdot \exp\left[- \int_{x \in \mathcal{T}} \lambda(x)\, dx \right] \cdot \prod_{(a,\ell,b) \in \mathcal{M}} Q^{(\ell)}_{ab} \cdot \pi_\varphi(\mathcal{T}|\boldsymbol{\theta}) \cdot \pi(\boldsymbol{\theta}), \tag{1}$$

with

$$\lambda(x) = \sum_{\ell \in \xi(x)} Q^{(\ell)}_{s_x^{(\ell)}}.$$

The notation is as follows:

- $\boldsymbol{\theta}$ is a set of model parameters, such as mutation rates, that determine $\pi^{(\ell)}_a$ and $Q^{(\ell)}_{ab}$ and enter in $\pi_\varphi(\mathcal{T}|\boldsymbol{\theta})$;
- ℓ ranges over all L sites of the genome (note the two restrictions to $\xi(x)$);
- $s_x^{(\ell)}$ is the state of site ℓ in the sequence of the species at point x on the tree;
- r is the root of the tree;
- $\int_{x \in \mathcal{T}} h(x)\, dx$ is an integral of an arbitrary function $h(x)$ over every point x in the tree $\mathcal{T}$, defined as the sum of branch integrals, each of which integrates over time along the branch;
- $\lambda(x)$ is the sequence-dependent genome mutation rate for the species at point x on the tree;

- $\mathcal{M}$ is the set of all mutations on the tree, each with the form (t, i, a, ℓ, b) described above (the branch index i and mutation time t are omitted above for clarity);
- $Q_{ab}^{(\ell)}$ is the element of the transition rate matrix for site ℓ that encodes the rate at which a site with state a mutates to a state b [42];
- $Q_a^{(\ell)} := -Q_{aa}^{(\ell)}$ is the magnitude of one of the diagonal elements of this matrix;
- $\pi_a^{(\ell)}$ is the stationary state distribution of this matrix, defined as its zero left-eigenvector ($\sum_a \pi_a^{(\ell)} Q_{ab}^{(\ell)} = 0$);
- $\pi_\varphi(\mathcal{T}|\boldsymbol{\theta})$ is an ancestry prior on the tree dependent only on the tree topology and model parameters, not on the tip sequences; at the moment, we implement only a coalescent prior (Section 6.8);
- $\pi(\boldsymbol{\theta})$ is the prior distribution on the model parameters (Section 3).

It can be shown [28] that if the tip sequences consist of either pure known states ('`A`', '`C`', '`G`', '`T`') or pure unknown states ('`-`', '`N`'), then the marginal posterior distribution obtained by integrating over all possible mutational configurations for fixed tree topology and model parameters is exactly

$$P'(\mathcal{T}, \boldsymbol{\theta}) \propto \mathcal{L}_{\text{tree}}(\mathcal{T}|\boldsymbol{\theta}) \cdot \pi_\varphi(\mathcal{T}|\boldsymbol{\theta}) \cdot \pi(\boldsymbol{\theta}),$$

where $\mathcal{L}_{\text{tree}}(\mathcal{T}|\boldsymbol{\theta})$ is the traditional tree likelihood, as computed by Felsenstein pruning [38]. This is precisely the form of the posterior used in BEAST. Thus, in a very concrete sense, the results from a Delphy run are statistically indistinguishable from an equivalent BEAST run, as illustrated in the benchmarks of Section 5.

The treatment of missing data above is relatively novel. If all sites of all tip sequences have data, then $\xi(x) = \{1, \ldots, L\}$ for all points x, and the restricted sums above simply range over all L sites of the genome. If a particular site ℓ of a particular tip i is missing, we can integrate that posterior over all possible mutational configurations on the branch that leads to it. If the state at the beginning of the branch is a, this operation is equivalent to the sum $\sum_b P_{ab}^{(\ell)}(t)$ of the elements of row a of the evolution matrix for that site along just that branch, which is trivially 1. Thus, the contribution to the posterior arising from that branch at that site simply drops out, while the form of the posterior is preserved. If we repeat this process for all sites with missing data at any tip, and recurse over any inner nodes left childless as a result, we obtain the above posterior with site sums restricted to $\xi(x)$. We call this process "`N`-pruning", because it resembles maximally pruning all subtrees each site-specific tree that end entirely in `N` at their tips.

The key improvement resulting from using an explicit mutation representation is that the posterior $P(\mathcal{T}|\boldsymbol{\theta})$ is a straight product of simple terms, instead of the product of sums of products that appears in Felsenstein pruning. Hence, ratios of posteriors for trees that differ only locally result in near-complete cancellation, and the ratios can be computed in a handful of operations, proportional to the number of mutations in nearby branches, instead of many thousands of operations, proportional to the number of variable sites in the entire dataset. This representation trades off some statistical efficiency for substantial mathematical simplifications. However, particularly in genomic epidemiology datasets, where N sample sequences can generally be placed into a parsimony tree involving only $\sim N$ mutations [23, 27], this is a very favorable trade-off.

6.3 Generic SPR moves

Delphy implements two kinds of local MCMC moves: simple moves and SPR moves. Delphy uses two kinds of simple moves: rearranging mutations along a branch, and displacing inner node times without changing tree topologies or mutational histories. SPR moves are discussed in the remainder of this section. Together with a battery of global moves, discussed below in Section 6.5, we have found these sufficient to effectively sample the posterior distributions trees for genomic epidemiology datasets, as evidenced by the benchmarks in Section 5. In particular, we emphasize the lack of evidence thus far that the MCMC easily gets trapped in localized metastable states.

All local MCMC moves that alter the tree topology in Delphy are built on the subtree pruning and regrafting (SPR) operation. With it, we can implement the typical subtree slide [16] and Wilson-Balding moves [43]. We also implement a novel mutation-directed SPR move described below, effectively a well-targeted Wilson-Balding move, which has proven key to making otherwise improbable large-scale tree rearrangements. The key complication is that along with a topological rearrangement of the tree, an SPR move in Delphy also needs to propose a new (local) mutational history that is compatible with the context around the regrafting point.

To keep moves local and simple to implement, we fix the mutational history everywhere except in the branch that joins the grafting point P to the root of the pruned subtree X (see Figure 9). Hence, the ratio of the posterior ratio before and after subtree pruning and regrafting contains only terms due to the length and mutational history of this branch. We propose a new mutational history by adapting the method of Nielsen [44] to efficiently almost always propose no mutations on all the sites where the sequences at X and P agree. Hence, the cost of proposing and evaluating such an MCMC move scales roughly as the number of mutations on the P–X branch.

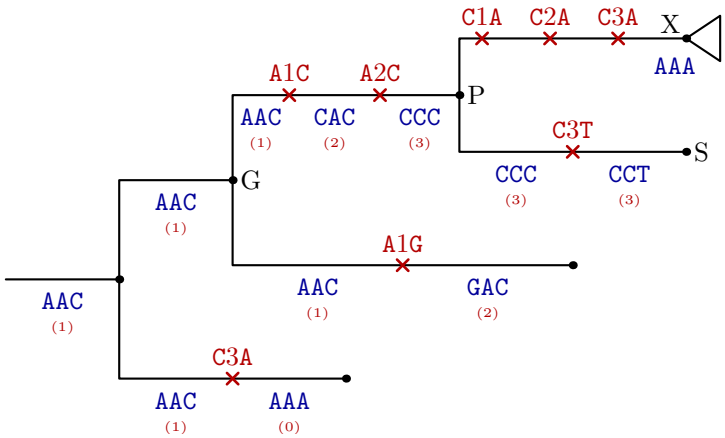


Figure 9: Setup for an SPR move. A subtree rooted at X will be detached from the remaining tree, where it is currently grafted at P. The sequence at X is `AAA`, while the sequence at P is `CCC`. Each branch is divided by its mutations into distinct regions, here annotated with their sequence (blue) and the number of differences between that sequence and the sequence at X (red, in parentheses).

Special care must be taken when pruning and/or regrafting changes the root of the tree. Similarly, pruning and/or regrafting may change the missations around the regrafting point, possibly extending beyond the single branch P–X. Full details will be described in our upcoming preprint [28].

6.4 Mutation-directed SPR moves

In Figure 9, the possible grafting points for the SPR move that prunes the subtree rooted at X are classified into regions along each branch, demarcated from each other by an intervening mutations; all points in a single region are in the same branch and have the same sequence. It is easy to efficiently calculate the number of sequence differences between the sequence at X and every point in each region, by a simple local walk of the tree: these numbers are written in red inside parentheses. These annotations clearly suggest an effective way to propose a new grafting point for X: overwhelmingly propose points where those differences are minimized. Indeed, a parsimony improvement algorithm, such as matOptimize, would *always* pick one of the points with minimal sequence difference, with ties broken arbitrarily.

The explicit-mutation representation makes it clear how to adapt the heuristic that drives parsimony improvement algorithms into an exact MCMC move that correctly samples tree from the posterior distribution. We call such moves "mutation-directed SPR moves" (mdSPR). They generalize in a disciplined way

the intuition that small changes to the tree are likely to result in a plausible new tree, but that sometimes large-scale changes are essential to correctly sample all plausible trees, not just a subset of closely related ones. While a long enough sequence of local SPR moves are in principle sufficient to turn any tree topology into any other, mdSPR moves ensure that large-scale rearrangement are possible without going through extremely improbable intermediate states. As we describe below, mdSPR moves provide a clear rationale for deciding what constitutes a local move, what balance to strike between local and global moves, how to choose global rearrangements that are likely to result in posterior increases.

To proceed, we note a substantial simplification to the posterior of Eq. (1) when we specialize to a Jukes-Cantor substitution model with no site rate heterogeneity, that is, one where every site mutates at the same rate, and no particular state transition is preferred over any other. Concretely, $Q_{ab}^{(\ell)} = \mu\{-\delta_{ab} + (1-\delta_{ab})/3\}$ and $\pi_a^{(\ell)} = 1/4$, where μ is the site mutation rate. If we further suppose that no tip sequences has missing data, we obtain

$$P_{JC}(\mathcal{T}, \boldsymbol{\theta}) \propto e^{-\mu L T} \cdot \left(\frac{\mu}{3}\right)^M \cdot f(\mathcal{T}, \boldsymbol{\theta}). \tag{2}$$

Here, T is the total branch length (i.e., the sum of the lengths of each branch in the tree) and M is the total number of mutations. The first factor of Eq. (1) reduces to $(1/4)^L$ for all $\mathcal{T}$ and $\boldsymbol{\theta}$, so has been subsumed into the constant of proportionality.

Denote by $N(x)$ the number of differences between the sequences at X and an arbitrary point x on the tree. The above picture suggests that, to a good approximation, the ratio of the posteriors for the "old" tree $\mathcal{T}_o$, where X is attached at P, and the "new" tree $\mathcal{T}_n$, where X is attached at P', is given by

$$\frac{P(\mathcal{T}_n)}{P(\mathcal{T}_o)} \approx \frac{\exp[-\mu L(t_X - t_{P'})] \cdot [\mu/3]^{N(P')}}{\exp[-\mu L(t_X - t_P)] \cdot [\mu/3]^{N(P)}}.$$

Here, $L = |\xi(X)|$ is the number of sites not missing at X and μ is an effective site mutation rate, given by $\mu = \lambda(X)/L$. The approximation would be exact only if the substitution model were indeed Jukes-Cantor, there were indeed no missing data, and there were exact cancellation in the ancestry priors. Nevertheless, intuitively, the approximation captures the dominant factors in the posterior ratio. We can thus use it to make a proposal, and thus obtain an MCMC move with near-unity acceptance probability; the deviations from unity exactly correct for the approximate nature of the proposal.

Concretely, we adopt the following form for the proposal probability:

$$\alpha(\mathsf{o} \to \mathsf{n}) \propto \alpha_{\text{graft}}(\mathsf{o} \to \mathsf{n}) \cdot \alpha_{\text{mut}}(\mathsf{o} \to \mathsf{n}), \tag{3}$$

where

$$\alpha_{\text{graft}}(\mathsf{o} \to \mathsf{n}) := \exp[-\mu L(t_X - t_{P'})] \cdot [\mu/3]^{N(P')} \cdot (t_X - t_{P'})^{N(P')}, \tag{4}$$

$$\alpha_{\text{mut}}(\mathsf{o} \to \mathsf{n}) \approx 1/(t_X - t_{P'})^{N(P')}. \tag{5}$$

In other words, the proposal probability decomposes into a choice of the grafting point P', determined by $\alpha_{\text{graft}}(\mathsf{o} \to \mathsf{n})$, followed by a proposal of a particular mutational history to join P' to X, determined by $\alpha_{\text{mut}}(\mathsf{o} \to \mathsf{n})$. In the limiting scenario we have in mind, where branches are very short relative to the typical site mutation rate, the approximate value of $\alpha_{\text{mut}}(\mathsf{o} \to \mathsf{n})$ results from all the identities of the mutations being fixed, and their times being chosen uniformly along the length of the P'–X branch. Note that in practice, it is important to calculate $\alpha_{\text{mut}}(\mathsf{o} \to \mathsf{n})$ exactly and to allow for any possible history between P' and X, so that the integral of $\alpha_{\text{mut}}(\mathsf{o} \to \mathsf{n})$ over all possible such histories is exactly 1.

Eq. (4) is the key to implementing mutation-directed SPR moves. It gives us an unnormalized probability density for every possible regrafting point P' in terms of easily calculable quantities (see Figure 10). In particular, the quantity $N(P')$ can be computed trivially in an EMAT through a depth-first traversal of the tree starting at P (see annotations in Figure 9). It is this property of EMATs that greatly facilitates the calculation of $N(x)$ everywhere in the tree that motivates our naming this technique *mutation-directed*

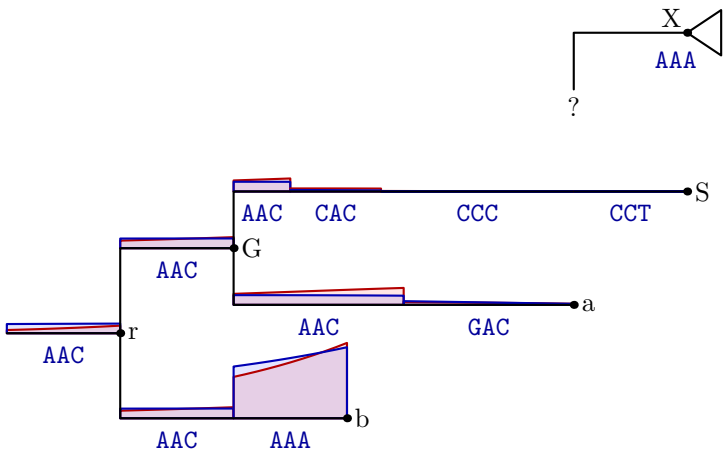


Figure 10: Cartoon of grafting site probabilities calculated by mutation-directed SPR move (red) vs. distribution implied by the posterior (blue). In this example, the real posterior has an HKY substitution model with $\kappa = 5.25$, $\pi = [0.29, 0.18, 0.19, 0.33]$, which is representative of the SARS-CoV-2 data in [35]. Note that probabilities are broadly similar. The parsimonious region with sequence AAA, where attaching X necessitates no mutations on the P'–X branch, is clearly the most likely choice, but all other regions are possible, and their combined weight need not be negligible.

SPR moves: in effect, the explicit mutations on the tree serve as an excellent and efficient guide to choosing regrafting points well, even over large distances.

To be more concrete, we can divide the tree left over after pruning X into K regions as follows. For each branch i, let m be the number of mutations, and let $t_1, \ldots, t_m$ be their times; let t_0 be the time of the start of the branch and let t_{m+1} be the time of the end of the branch. Then record the region in branch i spanning the time (t_j, t_{j+1}) for each j between 0 and m (inclusive). Finally, remove all regions that are entirely to the future of t_X; those regions that span t_X are also truncated at t_X. All points x in each region have the same value of $N(x)$, which we also record while constructing the regions. In the end, a region k thus consists of a tuple (i_k, t_k, t'_k, N_k), respectively the branch index, the region's start and end times, and the number of sequence differences between any point in the region and X. Then define a weight w_k for each region as follows:

$$w_k = \int_{t_k}^{t'_k} dt \, \exp[-\mu L(t_X - t)][\mu(t_X - t)/3]^{N_k}. \tag{6}$$

To pick a new regrafting point, pick a region k in proportion to the weights $\{w_k\}$. Then pick a point between t_k and $t_{k'}$ in proportion to

$$p(t|k) \propto \exp[-\mu L(t_X - t)][\mu(t_x - t)/3]^{N_k}. \tag{7}$$

In practice, we simplify these weights even more, and take into account the complications of root changes and missing data. Moreover, although we illustrate mdSPR moves with a complete exploration of the tree for possible grafting points, in practice we most often perform only localized explorations. In particular, a walk that never crosses a mutation explores variations of polytomy resolutions, while one where at most one mutation is crossed on the path between the old grafting point and any new candidate grafting point enables cheap localized refinements. Full details on these complications and variations will be presented in [28].

6.5 Global moves

A useful property of EMATs is that conditional on a fixed tree topology and genetic history, the MCMC moves for the associated model parameters are nearly trivial. We give a few illustrative examples below.

6.5.1 Mutation rate

For a single-partition substitution model, the transition rate matrices have the form

$$Q_{ab}^{(\ell)} = \mu\nu^{(\ell)}q_{ab},$$

where q_{ab} is a site-independent reduced transition matrix satisfying $\sum_a \pi_a q_{ab} = 0$ and $\sum_a \pi_a q_{aa} = -1$, and the site relative rates $\nu^{(\ell)}$ average to 1 à priori (see below). This form cleanly separates out the mutation rate μ from the relative rates of transitions between particular bases and from the effects of site-rate heterogeneity.

The conditional posterior distribution for μ follows from Eq. (1):

$$P_\mu(\mu) \propto e^{-\mu L\tilde{T}} \cdot \mu^M \cdot \pi_\mu(\mu). \tag{8}$$

Here, $\pi_\mu(\mu)$ is the prior on μ, M is the total number of mutations and $\tilde{T}$ is an effective total branch length of the tree, defined as

$$\tilde{T} = \frac{\int_{x\in\mathcal{T}} \lambda(x)\,\mathrm{d}x}{\mu L}.$$

The right-hand side is independent of μ, as follows from expanding the definitions of $\lambda(x)$ and $Q_{ab}^{(\ell)}$. For a Jukes-Cantor model with no site-rate heterogeneity, $\tilde{T} = T$, where T is the total branch length of the tree; for general models, we expect $\tilde{T} \approx T$.

It is straightforward to implement generic MCMC moves on μ that sample the above conditional posterior. Moreover, for typical forms of $\pi_\mu(\mu)$, namely an improper uniform prior and a $1/\mu$ prior, the conditional posterior in Eq. (8) is a Gamma distribution, so we can simply directly Gibbs sample μ:

$$\mu \sim \begin{cases} \mathrm{Gamma}(M-1, L\tilde{T}), & \text{for } \pi_\mu(\mu) = 1; \\ \mathrm{Gamma}(M, L\tilde{T}), & \text{for } \pi_\mu(\mu) = 1/\mu. \end{cases}$$

The above procedure also makes it clear that the conditional average of μ is close to $M/(L\tilde{T})$ (exactly so for a $1/\mu$ prior), a physically pleasing result.

It is straightforward to generalize the above result to substitution models with b partitions and up to b independent mutation rates.

6.5.2 Other substitution model parameters

As for the mutation rate μ, we can construct a joint conditional posterior distribution for the variables that specify the evolution model parameters, $\pi_a^{(\ell)}$ and $Q_{ab}^{(\ell)}$. For example, in a single-partition HKY model, the stationary frequencies π_a are specified directly (and are the same for all sites), while the values of $Q_{ab}^{(\ell)}$ are derived from those frequencies and the transition-transversion ratio κ. Hence, the joint conditional posterior distribution for κ and $\{\pi_a\}$ that follows from Equation (1) is:

$$P_Q(\kappa, \{\pi_a\}) \propto \prod_a \pi_a^{R_a} \cdot e^{-\mu L \sum_a q_a \tilde{T}_a} \cdot \prod_{a,b} (q_{ab})^{M_{ab}} \cdot \pi_Q(\kappa, \{\pi_a\}). \tag{9}$$

Here, $\pi_Q(\kappa, \{\pi_a\})$ is the prior on the substitution model parameters, R_a is the number of sites with state a in the root sequence, M_{ab} is the number of a-to-b mutations in the tree, and $\tilde{T}_a$ is an effective total branch length in state a, defined as

$$\tilde{T}_a = \frac{1}{q_a \mu L} \int_{x\in\mathcal{T}} dx \sum_{\ell\in\xi(x)} [s^{(\ell)}(x) = a] Q_a^{(\ell)}.$$

As above, $\tilde{T}_a$ is actually independent of the values of μ and q_a. Its value can be calculated efficiently on an EMAT, and its value can be updated continuously after local MCMC moves.

The above conditional posterior suggests very cheap moves for making small trial changes to κ and $\{\pi_a\}$, for example, by scaling and delta-exchange, respectively. A long sequence of such moves approximates a Gibbs sampler for κ and $\{\pi_a\}$, and can be implemented efficiently if there are no intervening topological or mutational changes to the tree. Unlike with the mutation rate, the above conditional posterior does not immediately suggest a direct Gibbs sampler for any common choice of prior $\pi_Q(\kappa, \{\pi_a\})$.

It is straightforward to generalize the above result to all common substitution models (e.g., GTR) and to multiple partitions whose parameters may be linked or unlinked in any arbitrary way.

6.6 Continuous-gamma site-rate heterogeneity

In the standard approach to modeling site-rate heterogeneity, the Gamma distribution for the site relative rates $\{\nu^{(\ell)}\}$ is approximated by a discrete distribution with K categories because it is not possible to analytically integrate the tree likelihood for all choices of $\{\nu^{(\ell)}\}$. Instead, the integration is approximated by K evaluations of the tree likelihood at evenly distributed quantiles of $\nu^{(\ell)}$, with a concomitant K-fold increase in cost to every MCMC move that changes the tree likelihood. In contrast, in our explicit approach, we do not attempt to integrate these degrees of freedom analytically, but instead let the integration occur implicitly via sampling. In other words, both α and $\{\nu^{(\ell)}\}$ are explicit, continuous parameters of the inference, and individual MCMC moves are not appreciably more expensive when including vs excluding site-rate heterogeneity effects.

It is tempting to evaluate conditional posterior distributions for α and $\{\nu^{(\ell)}\}$ separately, and then implement separate moves on each. The relevant joint conditional posterior distribution again follows from Equation (1):

$$P_\nu(\alpha, \{\nu^{(\ell)}\}) \propto \pi_\alpha(\alpha) \cdot \prod_\ell e^{-\mu\nu^{(\ell)}\tilde{T}^{(\ell)}} \cdot \left[\nu^{(\ell)}\right]^{M^{(\ell)}} \cdot \frac{\alpha^\alpha}{\Gamma(\alpha)} \left[\nu^{(\ell)}\right]^{\alpha-1} e^{-\alpha\nu^{(\ell)}}. \tag{10}$$

Hence, the individual conditional posterior distributions are:

$$P_\alpha(\alpha) \propto \pi_\alpha(\alpha) \cdot \prod_\ell \frac{\alpha^\alpha}{\Gamma(\alpha)} \left[\nu^{(\ell)}\right]^{\alpha-1} e^{-\alpha\nu^{(\ell)}},$$

$$P_{\nu^{(\ell)}}(\nu^{(\ell)}) \propto e^{-\mu\nu^{(\ell)}\tilde{T}^{(\ell)}} \cdot \left[\nu^{(\ell)}\right]^{M^{(\ell)}} \cdot \left[\nu^{(\ell)}\right]^{\alpha-1} e^{-\alpha\nu^{(\ell)}}.$$

Here, $\pi_\alpha(\alpha)$ is the prior on α, $M^{(\ell)}$ is the number of mutations in site ℓ and $\tilde{T}^{(\ell)}$ is an effective total branch length for site ℓ, given by

$$\tilde{T}^{(\ell)} := \frac{1}{\mu} \int_{x \in \mathcal{T}} dx \, [\ell \in \xi(x)] Q_a^{(\ell)}.$$

We note that $\tilde{T}^{(\ell)}$ can be efficiently calculated in time proportional to the sum of the tree size and the number of mutations and missations.

Observe that the conditional posterior for $\nu^{(\ell)}$ given above is the distribution Gamma$(\alpha+M^{(\ell)}, \alpha+\mu\tilde{T}^{(\ell)})$. The tree topology and mutational history of site ℓ simply change the prior's Gamma parameters in a way that compares the number of mutations on this site expected in the absence of site-rate heterogeneity ($\mu\tilde{T}^{(\ell)}$) with the actual number ($M^{(\ell)}$), and α modulates the extent to which discrepancies are ascribed to real differences in rates vs stochastic fluctuations. Moreover, because the posterior distribution for $\nu^{(\ell)}$ is a Gamma distribution, we can draw new conditional samples of $\{\nu^{(\ell)}\}$ efficiently. For α, though, we are again reduced to making trial changes, e.g., by scaling, and accepting or rejecting them in the usual way.

In practice, the above scheme of separately sampling α and $\{\nu^{(\ell)}\}$ is not effective. The values of α and $\{\nu^{(\ell)}\}$ are very correlated, and independent moves along each of them do not effectively explore the joint parameter space. However, one can marginalize the $\{\nu^{(\ell)}\}$ in Equation (10) to get an effective conditional posterior distribution for α:

$$\tilde{P}_\alpha(\alpha) \propto \pi_\alpha(\alpha) \cdot \prod_\ell \frac{\Gamma(\alpha + M^{(\ell)})}{\Gamma(\alpha)} \frac{\alpha^\alpha}{(\alpha + \mu\tilde{T}^{(\ell)})^{\alpha+M^{(\ell)}}}$$

Although the function $\tilde{P}_\alpha(\alpha)$ is expensive to calculate, it is relatively smooth, so repeated trial changes to α will have high acceptance probabilities and quickly approximate a Gibbs sampler of $\tilde{P}_\alpha(\alpha)$. After thus obtaining a new value of α, we directly sample all L new values of $\nu^{(\ell)}$ as described above.

6.7 Scaling strategy

In matOptimize, the search for a parsimonious tree is parallelized by observing that the effect of an SPR moves is quite localized, so two SPR moves in different portions of the tree can first be evaluated separately, and their combined effect deduced from those separate evaluations. In other words, the net effect of two non-conflicting SPR moves is independent of the order in which those moves are performed. The MCMC SPR moves on EMATs described above have an analogous property: if we partition a tree into non-overlapping subtrees and independently perform SPR moves in each subtree in parallel, the net result is equivalent to performing those moves serially by interleaving the subtree moves in any way. This idea is depicted schematically in Figure 11

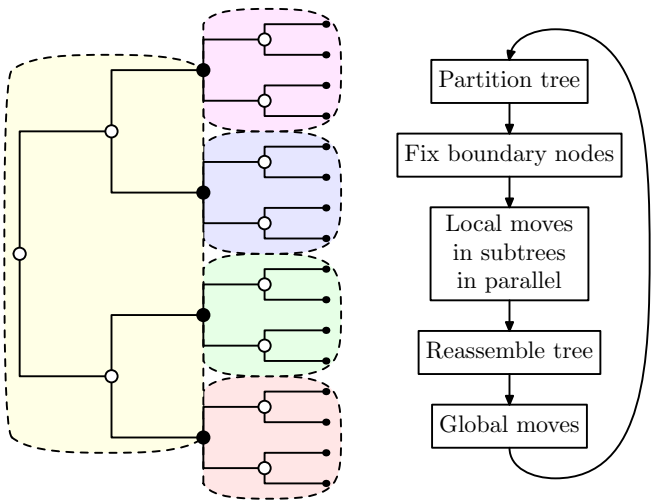


Figure 11: Schematic of Delphy's scaling strategy. In each cycle, the sequence and times of the boundary nodes (filled) is fixed, while the sequence, times and topology of the remaining nodes (unfilled) can change. Within each shaded subtree, local MCMC moves are confined to that subtree. To ensure ergodicity, the fixed boundary nodes on successive cycles do not overlap.

6.8 Parallelizable coalescent prior

As is common in Bayesian phylogenetics of viruses, our ancestry prior $\pi_\varphi(\mathcal{T}|\boldsymbol{\theta})$ is the Kingman coalescent [31, 32], given by

$$\pi_{\text{anc}}(\mathcal{T}|\boldsymbol{\theta}) := \exp\left[-\int \binom{k(t)}{2} \frac{\mathrm{d}t}{N(t)}\right] \cdot \prod_{\text{inner nodes } j} \frac{1}{N(t_j)}, \tag{11}$$

where $k(t)$ is the number of active branches at time t and $N(t)$ is the product of the effective population size and the generation time.

A significant complication in parallelizing the coalescent prior is that it directly couples all the nodes at the end of branches that cross a given time t. Hence, at first glance, the use of a coalescent prior seems in conflict with the scaling strategy described in the previous section. In this section, we sketch out how we resolve that conflict through the introduction of an augmented coalescent prior. Full details are provided in [28]

First, instead of evaluating Eq (11) piecewise directly, we start by discretizing it into finite time segments of width Δ over the range of times spanned by the tree:

$$\pi_\varphi(\mathcal{T}|\boldsymbol{\theta}) \approx \exp\left[-\sum_i \binom{\bar{k}_i}{2}\frac{\Delta}{\bar{N}_i}\right] \cdot \prod_{\text{inner nodes } j} \frac{1}{N(t_j)}. \tag{12}$$

Here and below, a barred variable, say $\bar{k}_i$, stands for the approximate value of a corresponding function of time, say $k(t)$, over the time interval i. In our current implementation, we partition the range over which $k(t) > 0$ into around 400 intervals, periodically changing Δ if the range of $k(t)$ changes drastically. In practice, Δ is readjusted a handful of times during the burn-in period as the tree height converges, but remains stable throughout the production period of the MCMC run. Because the coalescent prior introduces only modest gradients into the posterior, the crudeness of the above approximation does not seem to impact results appreciably.

Second, we split the tree $\mathcal{T}$ into P partitions. It follows that we can decompose the number of active lineages $\bar{k}_i$ into a sum over the numbers of active lineages $\bar{k}_{p,i}$ in partition p, $\bar{k}_i = \sum_p \bar{k}_{p,i}$. Rewriting Eq. (12) in terms of partitions yields

$$\pi_\varphi(\mathcal{T}|\boldsymbol{\theta}) \approx \prod_p \left\{ \prod_i \exp\left[-\Delta\left(\frac{\bar{k}_{p,i}\cdot[\sum_q \bar{k}_{q,i}]}{2\bar{N}_i} - \frac{\bar{k}_{p,i}}{2\bar{N}_i}\right)\right] \cdot \prod_{j\in p}\frac{1}{N(t_j)}\right\}. \tag{13}$$

This form emphasizes the coupling between partitions p and q owing to the quadratic term in the exponential.

An analogy with electrostatics can be drawn at this point: the potential energy of a set of particles can be calculated either as the sum of pairwise direct Coulomb interactions between all pairs of particles, or by a sum of local interactions between charges and an induced local potential. In other words, we can replace direct interaction at a distance between charges with a local interaction mediated by a potential that transmits information through space. This picture suggests introducing a set of P augmenting Gaussian variables $\tilde{k}_{p,i}$ that couple linearly to the $\bar{k}_{p,i}$ according to the following *augmented coalescent prior*:

$$\tilde{\pi}_\varphi(\mathcal{T}|\boldsymbol{\theta},\{\tilde{k}_{p,i}\}) := \prod_p \left\{ \prod_i \mathcal{C}_{p,i}\exp\left[-\Delta\left(\frac{(\tilde{k}_{p,i}-\mu_{p,i})^2}{2\sigma_{p,i}^2} + \frac{\tilde{k}_i\cdot\bar{k}_{p,i}}{\bar{N}_i} - \frac{\bar{k}_{p,i}}{2\bar{N}_i}\right)\right] \cdot \prod_{j\in p}\frac{1}{2N(t_j)}\right\}. \tag{14}$$

Above, we've introduced the sum $\tilde{k}_i$ analogous to $\bar{k}_i$, defined as $\tilde{k}_i := \sum_p \tilde{k}_{p,i}$. The mean $\mu_{p,i}$ and variance $\sigma^2_{p,i}$ of the free variable $\tilde{k}_{p,i}$, as well as the normalization constants $\mathcal{C}_{p,i}$ are not yet specified. We make the following choices:

$$\mu_{p,i} = \bar{k}_{p,i}, \qquad \sum_p \bar{\sigma}^2_{p,i} = \bar{N}_i, \qquad \mathcal{C}_{p,i} = \left(2\pi\sigma^2_{p,i}/\Delta\right)^{-1/2}.$$

We claim that these choices lead to the augmented coalescent prior reducing to Eq. (13) once the augmenting variables $\tilde{k}_{p,i}$ are integrated out, that is,

$$\int \left\{\prod_{p,i} d\tilde{k}_{p,i}\right\} \tilde{\pi}_\varphi(\mathcal{T}|\boldsymbol{\theta},\{\tilde{k}_{p,i}\}) = \pi_\varphi(\mathcal{T}|\boldsymbol{\theta}).$$

The calculation is tedious but straightforward.

The above transformation paves the way for parallelizing the coalescent prior. First, notice that for fixed $\{\bar{k}_{p,i}\}$, the variables $\{\tilde{k}_{p,i}\}$ can be Gibbs-sampled, because they are straightforward Gaussians. Second, for fixed $\{\tilde{k}_{p,i}\}$, Eq. (14) is a product of independent terms, one for each partition p. Hence, we can make independent local moves in each partition p with only knowledge of $\tilde{k}_{p,i}$, $\tilde{k}_i$ and $\bar{N}_i$.

The above considerations lead to the following parallelization scheme:

1. Cut up the tree $\mathcal{T}$ into P partitions.

2. Calculate $\bar{k}_{p,i}$ for every partition p, then calculate their sum $\bar{k}_i$.

3. Sample random Gaussian fields $\tilde{k}_{p,i}$ as per Equation (14), then calculate their sum $\tilde{k}_i$.

4. Distribute the P partitions over P parallel workers. Worker p receives the subtree in partition p, and the full set of values for $\tilde{k}_{p,i}$ and $\tilde{k}_i$.

5. Apply local MCMC moves within each worker using the augmented coalescent prior of Equation (14). By the above arguments, each worker has enough information to do this correctly and independently.

6. Reassemble the modified subtrees from each of the workers into a global tree.

7. Apply global MCMC moves, such as changes in mutation rates or evolution model parameters. This is done with the staircase coalescent prior given by Equation (12).

Intuitively, at the beginning of each cycle, a central process has a global view of the tree. It takes a fuzzy "snapshot" of this global view, in the form of $\{\tilde{k}_{p,i}\}$ and $\tilde{k}_i$. Then, each worker adjusts the various $\{\bar{k}_{p,i}\}$ in accordance with that earlier snapshot, not in accordance to the instantaneous $\bar{k}_i$, which they cannot access. Periodically, the tree is reassembled to be repartitioned and resnapshotted. Statistically, this process ends up transmitting the global knowledge needed to impose the coalescent prior to every worker.

6.9 Fast MCC trees

Delphy's web interface rederives an MCC every time a new posterior tree is sampled, so it needs to do this quickly. In contrast, BEAST2's TreeAnnotator takes tens of seconds to deduce the MCC of the SARS-CoV-2 results shown in Figures 2 and 3. Here, we describe how we derive such MCCs in milliseconds instead.

First, recall how MCCs are usually derived. For each posterior tree, we enumerate all its *clades*, i.e., the sets of tips descended from each node. Any particular clade is present in a fraction of the posterior trees, called the clade's *posterior support*. The MCC is defined as a tree with: (a) the topology of the posterior tree with highest product of posterior support over all its nodes; and (b) node times that summarize those of the corresponding nodes in the posterior trees. Most commonly, these latter times are the average (or median) of the times of the inner node for the same clade in each posterior tree where it is present.

What most slows down TreeAnnotator's MCC calculation is its representation of clades as explicit sets of tip indices. This leads to high memory consumption and high runtimes as explicit sets are built, inserted and retrieved from hash tables, and compared to each other. Instead, we introduce a probabilistic approach below that has proven much more effective in practice.

In Delphy's MCC calculation, we represent the clade of a node i as a fixed-size C-bit fingerprint F_i. We assign completely random fingerprints to tips. The fingerprint of a clade that is the union of two smaller and disjoint clades is calculated as the exclusive-or (XOR) of those two clade's fingerprints. Hence, these fingerprints can be calculated very efficiently and in time linear in the tree size for a single posterior tree, via a post-order traversal of the tree. When this procedure completes, the fingerprint for an arbitrary inner node is the XOR of the fingerprints of all its descendant tips:

$$F_i = \bigotimes_{\text{tips } j \text{ below } i} F_j.$$

If the tip fingerprints are uniformly random, then so are the fingerprints of the inner nodes. Thus, an MCC of M posterior trees, each with N inner nodes, has at most MN distinct clades, whose fingerprints are all essentially independent random C-bit fingerprints. We can reduce the probability of a fingerprint collision to an arbitrary level p by using a sufficiently large value of C, as follows:

$$p \approx 1 - \exp\left[-\frac{(MN)^2}{2 \cdot 2^C}\right] \approx \frac{(MN)^2}{2 \cdot 2^C}, \qquad \text{(large } C\text{)}.$$

Equivalently,

$$C \approx \log_2 \left[\frac{(MN)^2}{2p} \right].$$

Concretely, using $C = 64$ bits for 1000 posterior tree samples, each with 1000 inner nodes, yields $p \approx 3 \times 10^{-8}$. Scaling up to pandemic scales, e.g., 1000 trees with 10 M inner nodes would require $C \gtrsim 104$ to achieve $p \lesssim 10^{-6}$. The natural choice of $C = 128$ bits would lead to $p \approx 1 \times 10^{-19}$. Hence, we see no practical danger in using 64-bit fingerprints for near-real-time analysis of smaller datasets and 128-bit fingerprints for pandemic-scale datasets.

In practice, we build an MCC in four passes. First, we build a map of clade fingerprints to lists of tuples "(posterior tree sample index, node index)". Second, we calculate the log-credibility of each clade i as $\log(M_i/M)$, where M_i counts how many posterior trees contain clade i. Third, we compute for each posterior tree the sum of the log-clade-credibilities of each inner node, and record the tree with the highest sum. This tree defines the topology of the MCC. Finally, we iterate over the nodes of the MCC, and for each one, using the map from the first pass, we iterate over the corresponding nodes in the posterior trees to calculate the mean (or median) node time of each MCC node. Neglecting the weak $\log N$ dependence of C, which we have argued above is immaterial for all practical datasets, this four-pass algorithm runs in time $\mathcal{O}(MN)$, i.e., linear in the size of the results of the MCMC.

Funding Statement

This work is made possible by support from Flu Lab and a cohort of generous donors through TED's Audacious Project, including the ELMA Foundation, MacKenzie Scott, the Skoll Foundation, and Open Philanthropy. Funding was also provided by the National Institutes of Health (West African Research Network for Infectious Diseases), Grant # U01AI151812; the US Centers for Disease Control (Office of Advanced Molecular Genomics), Grant # 75D30122C14365; and the NIH/NIAID Genomics Centers for Infectious Diseases, Grant # 3U19AI110818.

Conflict of Interest Statement

Patrick Varilly, Ben Fry, Mark Schifferli, Katherine Yang, Ivan Specht, Michael Mitzenmacher and Pardis C. Sabeti are inventors on a pending patent application related to this work. P.C.S. is a co-founder of and shareholder in Sherlock Biosciences, Inc.; a co-founder of, shareholder in and consultant to Delve Bio; and a Board member of and shareholder in Danaher Corporation. Ben Fry is the Founder of Fathom Information Design, a design and software development firm in Boston. Mark Schifferli and Katherine Yang are the Fathom staff who led the Delphy team for Fathom.

References

[1] Khare, S. et al "GISAID's role in pandemic response," *China CDC Weekly* **3**, pp. 1049–1051 (2021).

[2] GISAID, `https://gisaid.org` (accessed 27 June 2024).

[3] Rambaut, A. et al. "A dynamic nomenclature proposal for SARS-CoV-2 lineages to assist genomic epidemiology," *Nature Microbiology* **5**, pp. 1403–1407 (2020).

[4] Gire, S.K., Goba, A., Andersen, K.G., Sealfon, R.S.G., Park, D.J. et al. "Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak," *Science* **345**, pp. 1369–1372 (2014).

[5] Metsky, H.C., Matranga, C.B., Wohl, S., Schaffner, S.F. et al. "Zika virus evolution and spread in the Americas," *Nature* **546**, pp. 411–415 (2017).

[6] O'Toole, Á. et al. "APOBEC3 deaminase editing in mpox virus as evidence for sustained human transmission since at least 2016," *Science* **382**, pp. 595–600 (2023).

[7] Cov-lineages Issue #38, "Proposed new B.1 sublineage circulating in India," `https://github.com/cov-lineages/pango-designation/issues/38` (accessed 24 June 2024).

[8] Parker, E., Omah, I.F., Varilly, P., et al. "Genomic epidemiology uncovers the timing and origin of the emergence of mpox in humans," (submitted, preprint DOI: `10.1101/2024.06.18.24309104`).

[9] Kraemer, M.U.G. et al, "Spatiotemporal invasion dynamics of SARS-CoV-2 lineage B.1.1.7 emergence," *Science* **373**, pp. 889–895 (2021).

[10] McCrone, J.T. et al. "Context-specific emergence and growth of the SARS-CoV-2 Delta variant," *Nature* **610**, pp. 154–160 (2022).

[11] Viana, R. et al. "Rapid epidemic expansion of the SARS-CoV-2 Omicron variant in southern Africa," *Nature* **603**, pp. 679–686 (2022).

[12] Adams, G. et al, "Preliminary genomic analysis of the RSV surge 2022 in Massachusetts reveals a polyphyletic epidemic driven by the expansion of multiple lineages," `https://virological.org/t/preliminary-genomic-analysis-of-the-rsv-surge-2022-in-massachusetts-reveals-a-polyphyletic-epidemic-driven-by-the-expansion-of-multiple-lineages/914` (accessed 27 June 2024).

[13] Didelot, X., Fraser, C., Gardy, J. and Colijn, C. "Genomic infectious disease epidemiology in partially sampled and ongoing outbreaks," *Mol. Biol. Evol.* **34**, pp. 997–1007 (2017).

[14] Lau, M.S.Y., Marion, G., Streftaris, G. and Gibson, G. "A systematic Bayesian integration of epidemiological and genetic data," *PLoS Comput. Biol.* **11** e1004633 (2015).

[15] Klinkenberg, D., Backer, J.A., Didelot, X., Colijn, C. and Wallinga, J. "Simultaneous inference of phylogenetic and transmission trees in infectious disease outbreaks," *Genetics* **198** e1005495 (2017).

[16] Suchard, M.A., Lemey, P., Baele, G., Ayres, D.L., Drummond, A.J. and Rambaut, A. "Bayesian phylogenetic and phylodynamic data integration using BEAST 1.10", *Virus Evolution* **4**, vey016 (2018).

[17] Bouckaert, R., Vaughan, T. G., Barido-Sottani, J., Duchêne, S., Fourment, M., Gavryushkina, A., Heled, J. et al. "BEAST 2.5: An advanced software platform for Bayesian evolutionary analysis," *PLoS computational biology* **15** (4), e1006650 (2019).

[18] 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. "MRBAYES 3.2: Efficient Bayesian phylogenetic inference and model selection across a large model space," *Syst. Biol.* **61**, pp. 539–542 (2012).

[19] Rambaut, A., Pybus, O.G., Nelson, M.I., Viboud, C., Tautenberger, J.K. and Holmes, E.C. "The genomic and epidemiological dynamics of human influenza A virus," *Nature* **453**, pp. 615–619 (2008).

[20] Minh, B.Q., Schmidt, H.A., Chernomor, O., Schrempf, D., Woodhams, M.D., von Haeseler, A. and Lanfear, R. "IQ-TREE 2: new models and efficient methods for phylogenetic inference in the genomic era," *Mol. Biol. Evol.* **37**, pp. 1530–1534 (2020).

[21] De Maio, N. et al, "Maximum likelihood pandemic-scale phylogenetics," *Nature Genetics* **55**, pp. 746–752 (2023).

[22] Turakhia, Y., Thornlow, B., Hinrichs, A.S. et al. "Ultrafast Sample placement on Existing tRees (UShER) enables real-time phylogenetics for the SARS-CoV-2 pandemic," *Nat Genet* **53**, pp. 809–816 (2021).

[23] Ye, C., Thornlow, B., Hinrichs, A., Kramer, A., Mirchandani, C., Torvi, D., Lanfear, R., Corbett-Detig, R. and Turakhia, Y. "matOptimize: a parallel tree optimization method enables online phylogenetics for SARS-CoV-2," *Bioinformatics* **38**, pp. 3734–3740 (2022).

[24] Sagulenko, P., Puller, V. and Neher, R.A. "TreeTime: Maximum-likelihood phylodynamic analysis," *Virus Evolution* **4**, vex042 (2018).

[25] Hadfield, J. et al. "Nextstrain: real-time tracking of pathogen evolution," *Bioinformatics* **34**, pp. 4121–4123 (2018).

[26] Du Plessis, L. et al. "Establishment and lineage dynamics of the SARS-CoV-2 epidemic in the UK," *Science* **371**, pp.708–712 (2021).

[27] Kramer, A.M., Thornlow, B. et al. "Online phylogenetics with matOptimize produces equivalent trees and is dramatically more efficient for large SARS-CoV-2 phylogenies than de novo and maximum-likelihood implementations," *Syst. Biol.* **72**, pp. 1039–1051 (2023).

[28] Varilly, P., Schifferli, M., Yang, K. et al. *Delphy: scalable and near-real-time Bayesian phylogenetics* (in preparation).

[29] Hasegawa, M., Kishino, H. and Yano, T. "Dating the human-ape splitting by a molecular clock of mitochondrial DNA," *Journal of Molecular Evolution* **22**, pp. 160–174 (1985).

[30] Yang, Z. "Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites," *Mol. Biol. Evol.* **10**, pp. 1396–1401 (1993).

[31] Kingman, J.F.C. "The Coalescent," *Stochastic Processes and the Applications* **13**, pp. 235–248 (1982).

[32] Kingman, J.F.C. "On the Genealogy of Large Populations," *Journal of Applied Probability* **19**, pp. 27–43 (1982)

[33] Katoh, K. and Standley, D.M. "MAFFT multiple sequence alignment software version 7: improvements in performance and usability," *Molecular Biology and Evolution* **30**, pp. 772–780 (2013).

[34] Heled, J. and Bouckaert, R. "Looking for trees in the forest: summary tree from posterior samples," *BMC Evolutionary Biology* **13**, 221 (2013).

[35] Lemieux, J.E., Siddle, K.J., et al. "Phylogenetic analysis of SARS-CoV-2 in Boston highlights the impact of superspreading events," *Science* **371**, eabe3261 (2021).

[36] FigTree, `https://github.com/rambaut/figtree/` (accessed 24 Jun 2024).

[37] Rambaut, A., Drummond, A.J., Xie, D., Baele, G. and Suchard M.A. "Posterior summarisation in Bayesian phylogenetics using Tracer 1.7," *Systematic Biology* **67**, pp. 901–904 (2018).

[38] Felsenstein, J. "Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach," *J. Mol. Evol.* **17**, pp. 368–376 (1981).

[39] Baltic, `https://github.com/evogytis/baltic` (accessed 24 Jun 2024).

[40] Ayres, D.L. et al. "BEAGLE: an application programming interface for statistical phylogenetics," *Syst. Biol.* **61**, pp. 170–173 (2012).

[41] Yang, Z. "Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods," *J. Mol. Evol.* **39**, pp. 306–314 (1994).

[42] Liò, P. and Goldman, N. "Models of Molecular Evolution and Phylogeny," *Genome Res.* **8**, pp. 1233–1244 (1998).

[43] Wilson, I.J. and Balding, D.J. "Genealogical Inference from Microsatellite Data," *Genetics* **150**, pp. 499–510 (1998).

[44] Nielsen, R. "Mapping Mutations on Phylogenies," *Syst. Biol.* **51**, pp. 729–739 (2002).