2025 Historian Evaluation Report

Krishna Bhatt (krishbhatt2019@gmail.com, Grade 10)
Department of Bioengineering, UC Berkeley, Holmes Lab

Table of Contents

Overview

Reconstructing ancestral sequence histories is a cornerstone of evolutionary biology, providing insight into the mechanisms of molecular evolution, protein function, and phylogenetic relationships. Accurate ancestral inference requires not only modeling substitutions but also realistically accounting for insertions and deletions (indels), which complicate both alignment and tree estimation. While tools such as Historian have been developed to jointly reconstruct alignments and ancestral states under probabilistic models, systematic benchmarking across diverse evolutionary scenarios remains limited.

The purpose of this work was to design and implement a comprehensive evaluation framework for Historian, enabling rigorous testing across a wide range of phylogenetic conditions. Using INDELible to simulate evolutionary histories, I generated biologically-derived datasets spanning the four major SCOP protein types containing a diversity of balanced and unbalanced trees, star-like and pectinate structures, variable branch lengths, substitution rates, and indel regimes. These simulations provided a

controlled ground truth against which to assess reconstruction accuracy, runtime performance, and robustness.

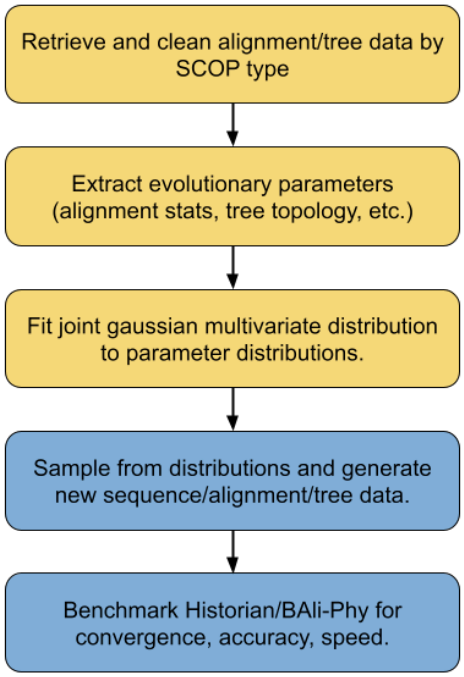
Beyond building the evaluation pipeline, this project involved an iterative process of refining methodology, overcoming computational and parameterization challenges, and comparing Historian with alternative tools such as BAli-Phy. Documenting these iterations not only highlights the effort invested but also demonstrates the systematic reasoning required to achieve reproducible results.

This report presents both the scientific outcomes—quantitative benchmarks of Historian's strengths and weaknesses—and the research process underpinning them. Together, they form a foundation for future comparative studies, potential publications, and further development of benchmarking frameworks in the lab.

- **Evaluation:** https://github.com/blizzard-labs/historianEval
- **Historian:** https://github.com/evoldoers/historian/
- **BAli-phy:** https://github.com/bredelings/BAli-Phy

Flowchart of Evaluation Pipeline

Compared to previous meta-analysis of phylogenetic MCMC samplers, my methodology simulates entirely new sets of ground truth biologically-realistic sequences rather than utilizing previously generated alignments, thus circumventing all process biases.



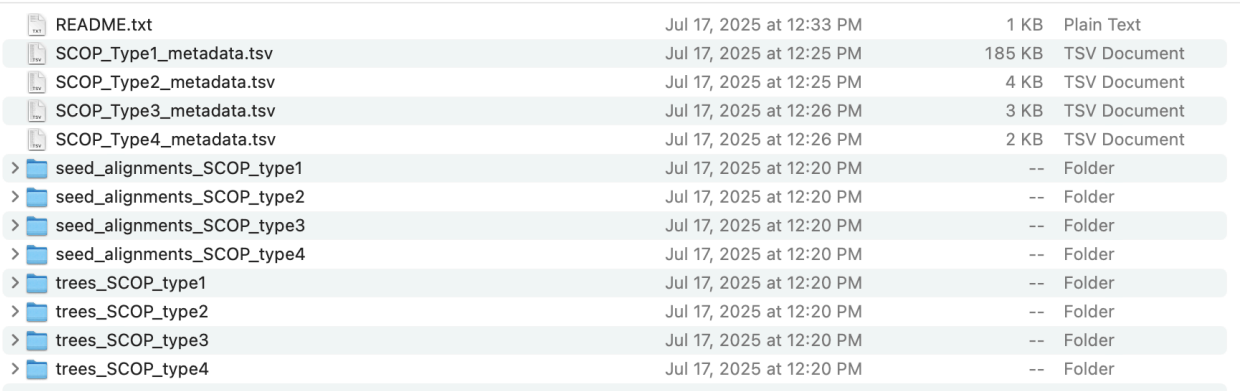
Methods: Simulation Framework

Data Retrieval / Cleaning

Annabel constructed a dataset of protein families using Pfam v36 seed alignments and corresponding phylogenetic trees. Phylogenetic trees were generated with FastTree (default parameters). In total, the dataset comprised 6,712 Pfam families, distributed as follows: 6,363 Type 1 (globular proteins), 151 Type 2 (membrane proteins), 119 Type 3 (fibrous proteins), and 79 Type 4 (non-globular/intrinsically unstructured proteins).

Seed alignments and associated trees were subjected to quality control. Sequences were removed if they (i) were shorter than 10 amino acids, (ii) occurred in alignments containing only a single sequence (in which case the entire Pfam was discarded), or (iii) contained invalid amino acid codes (B, O, U, X, Z). Duplicate sequences were also removed: only the first occurrence was retained within each Pfam, and sequences duplicated across multiple Pfams were excluded from the dataset entirely. All sequences were standardized to uppercase format.

To assign structural classifications, she first extracted all PDB identifiers linked to each Pfam family. These identifiers were then mapped to SCOP structural classes. Pfams were retained only if all associated PDB entries mapped to a single unique SCOP class, ensuring unambiguous structural annotation.



README.txt	Jul 17, 2025 at 12:33 PM	1 KB	Plain Text
SCOP_Type1_metadata.tsv	Jul 17, 2025 at 12:25 PM	185 KB	TSV Document
SCOP_Type2_metadata.tsv	Jul 17, 2025 at 12:25 PM	4 KB	TSV Document
SCOP_Type3_metadata.tsv	Jul 17, 2025 at 12:26 PM	3 KB	TSV Document
SCOP_Type4_metadata.tsv	Jul 17, 2025 at 12:26 PM	2 KB	TSV Document
seed_alignments_SCOP_type1	Jul 17, 2025 at 12:20 PM	--	Folder
seed_alignments_SCOP_type2	Jul 17, 2025 at 12:20 PM	--	Folder
seed_alignments_SCOP_type3	Jul 17, 2025 at 12:20 PM	--	Folder
seed_alignments_SCOP_type4	Jul 17, 2025 at 12:20 PM	--	Folder
trees_SCOP_type1	Jul 17, 2025 at 12:20 PM	--	Folder
trees_SCOP_type2	Jul 17, 2025 at 12:20 PM	--	Folder
trees_SCOP_type3	Jul 17, 2025 at 12:20 PM	--	Folder
trees_SCOP_type4	Jul 17, 2025 at 12:20 PM	--	Folder

GTR Parameter Fitting

For each SCOP class subset, Annabel fit a GGI+GTR model to estimate substitution and indel dynamics. From these fits, both the indel rate and extension probability parameters were recorded. To reduce parameter dimensionality and improve stability, a symmetry was enforced in the indel process by constraining the insertion rate to equal the deletion

rate, and the insertion extension probability to equal the deletion extension probability. Preliminary tests with relaxed constraints yielded similar estimates, supporting the appropriateness of the symmetric assumption.

training samples (double the amount of alignments)	Descr		GGI indel rate (lambda = mu)	GGI extension rate (x=y)	GGI mean indel length
236052	Type 1 (Globular proteins)	Type1	0.2256	0.6672	3.0045
5826	Type 2 (Membrane proteins)	Type2	0.2424	0.7043	3.3814
1260	Type 3 (Fibrous proteins)	Type3	0.2112	0.7289	3.6892
941	Type 4 (Non-globular/Intrinsically unstructured proteins)	Type4	0.2751	0.6578	2.9226

Extracting phylogenetic parameters

Protein evolutionary parameters were estimated using a custom Python workflow (`extract_params.py`) designed to integrate substitution model inference, indel characterization, and phylogenetic tree shape analysis. The workflow operates on multiple sequence alignments (MSAs) of protein families and produces both alignment-specific and dataset-level summaries.

Substitution Model Estimation: For each alignment, substitution model parameters were estimated using ModelTest-NG. Alignments were analyzed under the amino acid (aa) data type setting, with the candidate model space including standard empirical matrices such as LG, WAG, JTT, Dayhoff, DCMut, CpREV, VT, Blosum62, mtREV, and HIV models. If a user-specified tree was available (collected from Pfam- generated by FastTree), it was supplied; otherwise, ModelTest-NG optimized a maximum likelihood tree internally.

From the ModelTest-NG output, the following parameters were extracted:

- **Best-fit model** (based on likelihood scores, AIC, and BIC)
- **Log-likelihood (lnL)** of the model fit
- **Amino acid frequencies** (empirical or estimated, 20 values)
- **Rate heterogeneity parameters**:
    - Gamma distribution shape parameter (α) for across-site rate variation
    - Proportion of invariant sites (p_inv)
- **Model fit scores**: Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and associated model weights

If ModelTest-NG failed to converge or was unavailable, a **fallback estimation procedure** was applied*:

- Amino acid frequencies were directly computed from observed residue counts in the alignment.

- Site variability was estimated using Shannon entropy across alignment columns, from which a rough gamma shape parameter was inferred by method-of-moments.
- The proportion of invariant sites was approximated as the fraction of strictly conserved alignment positions.

*Note that the parameters estimated through the fallback procedure were only used for data visualization and characterization of the dataset rather than fitting the distributions.

Indel Parameter Estimation: Insertions and deletions were quantified directly from the alignments. The procedure consisted of:

1. **Gap Column Analysis:** Each alignment column was scanned to count gaps and residues.
2. **Event Identification:** Consecutive gap runs in individual sequences were grouped into single indel events to avoid inflation of event counts.
3. **Parsimony-Based Classification:** Indel events were classified as insertions or deletions by applying a minimum-event parsimony criterion: if gaps occurred in fewer sequences than residues, the event was classified as a deletion; otherwise, as an insertion. Consecutive columns with gaps of the same type were further grouped into continuous indel regions.
4. **Rate Normalization:** Indel rates were calculated per site per sequence, scaled by the alignment length and number of sequences.
5. **Length Distributions:** For each alignment, mean insertion and deletion lengths were estimated from observed gap runs. Where insertion lengths could not be directly inferred, they were approximated as a fraction (0.8×) of the deletion length, consistent with empirical patterns reported in protein evolution (again only for visualization).

From these analyses, the following parameters were extracted:

- Insertion rate and deletion rate
- Number of insertion and deletion events
- Mean insertion length and mean deletion length
- Total number of gaps across the alignment
- Indel-to-substitution ratio (estimated as the ratio of parsimony-based indel events to observed amino acid substitution counts)

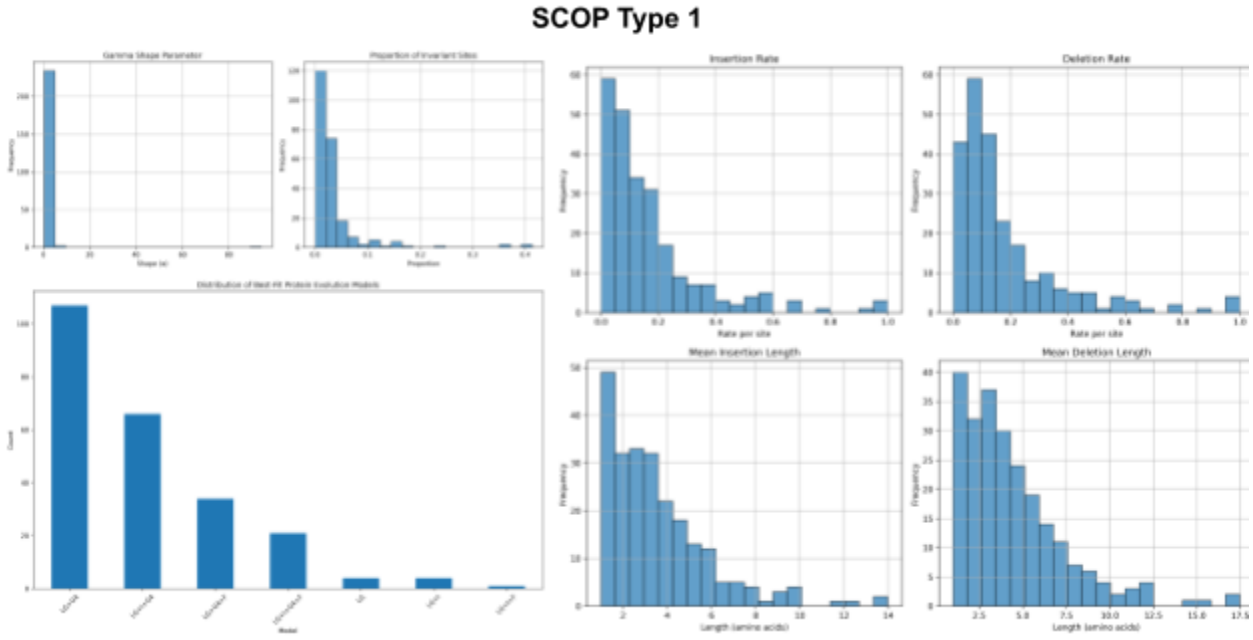
Tree Shape Metrics: For alignments with associated phylogenetic trees (in Newick format), tree topology statistics were computed using a custom python script. The following measures were recorded:

- Number of internal nodes
- Maximum tree depth
- Colless index (a measure of tree imbalance)
- Normalized Colless index (scaled for comparability across trees of different sizes)

These metrics provided information on lineage diversification balance and phylogenetic signal within each Pfam family.

Output and Summaries: All alignment-specific parameters were aggregated into a structured dataset. Each entry included substitution model information, rate heterogeneity, indel parameters, and tree shape metrics. Results were exported as a CSV file, accompanied by a summary statistics table (mean, variance, quartiles) across all alignments.

Additionally, the workflow generated distribution plots for amino acid frequencies, best-fit model counts, gamma shape parameters, invariant site proportions, indel rates, and indel length distributions. These visualizations facilitated comparison of parameter variation across the dataset.

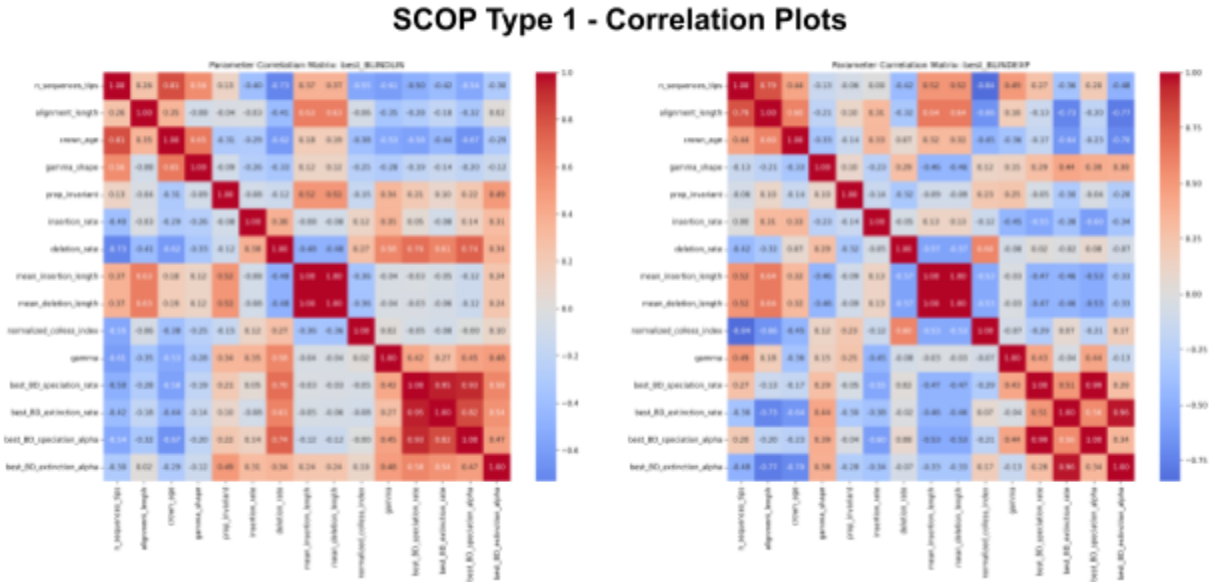


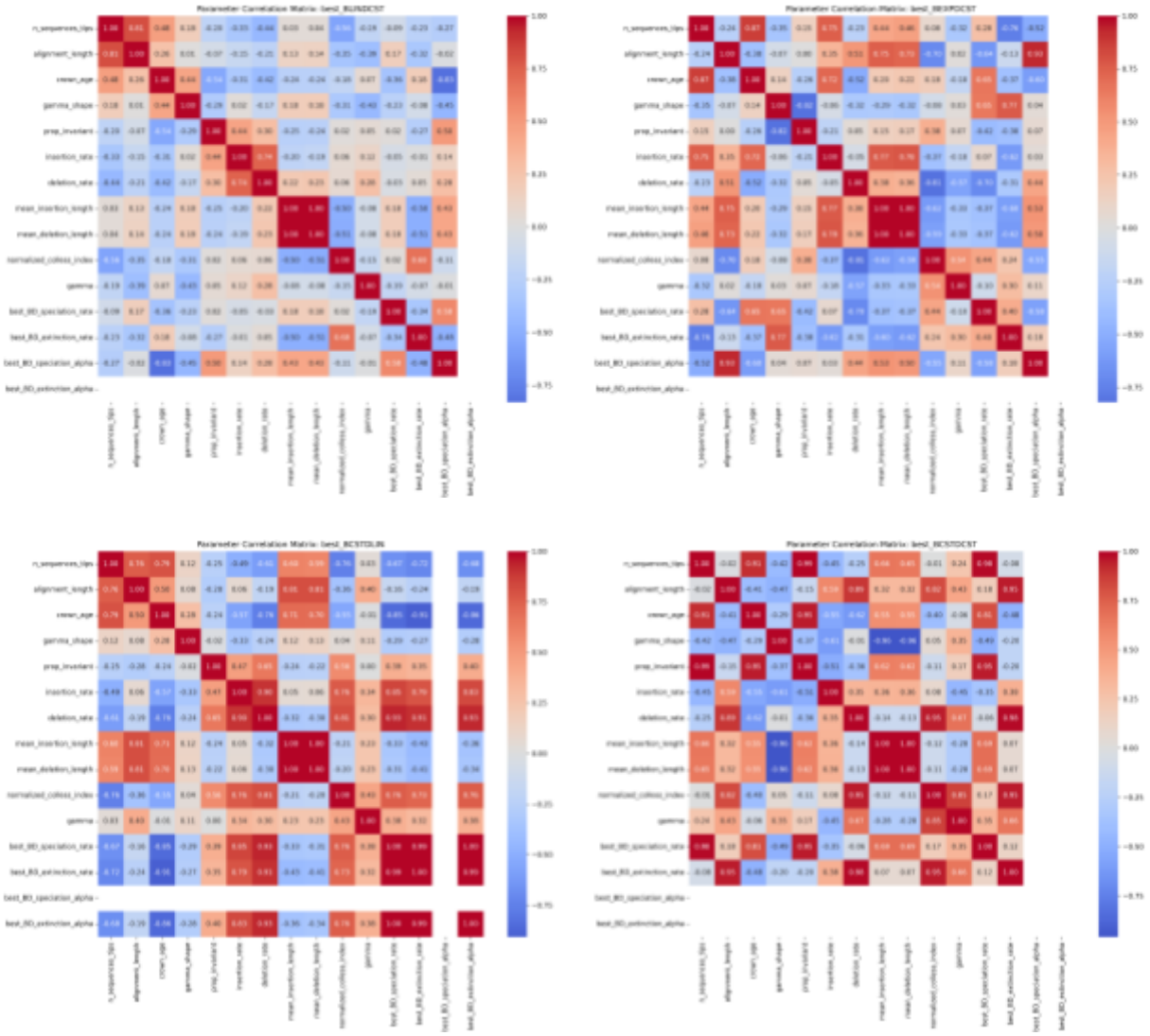
Fitting Multivariate Distributions

To capture realistic evolutionary parameter distributions, I developed a Python workflow (`modelfit.py`) that fits joint parameter distributions conditioned on specific birth–death (BD) diversification models. This pipeline integrates preprocessing, univariate and multivariate distribution fitting, validation, and export of sampled parameters for downstream sequence simulations with INDELible.

Data Input and BD Model Partitioning The script operates on parameter tables previously extracted from alignments and trees (e.g., SCOP datasets). Each entry includes alignment characteristics, substitution parameters, indel statistics, tree balance metrics, and BD model inference results.

To separate evolutionary regimes, datasets were partitioned according to the best-supported BD model (as indicated by one-hot encoded BD model indicator columns). Each BD model configuration was treated independently, and the relative frequency of models across the dataset was used to compute empirical model probabilities. These probabilities govern the likelihood of sampling each BD regime in later simulations.





Preprocessing and Transformations For each BD-specific dataset:

- **Numeric filtering:** Only numeric columns were retained; missing values were dropped.
- **Rate parameters:** Parameters such as insertion rate, deletion rate, mean indel lengths, gamma shape, and speciation/extinction rates were log-transformed to stabilize variance and approximate log-normal distributions.
- **Proportions:** Proportion parameters (e.g., invariant site proportion, amino acid frequencies) were logit-transformed to map values from (0,1) to the real line.
- **Transformation metadata:** Log and logit transformations were tracked to allow bias correction and inverse transformation during sampling and export.

Marginal Distribution Fitting: For each parameter, a suite of candidate probability distributions was evaluated, including normal, log-normal, gamma, beta, exponential, Weibull, chi-square, inverse gamma, Pareto, generalized extreme value, gumbel, logistic, Laplace, t, and generalized Pareto families. Parameters were fit via maximum likelihood, and models were compared using Akaike Information Criterion (AIC). The best-fitting distribution for each parameter was recorded for each BD model subset.

Joint Distribution Modeling: To preserve correlations among parameters, joint distributions were fitted within each BD model subset:

- **Primary approach:** A **Gaussian copula** (via the copulas library) was used to capture dependency structures while allowing marginals to follow their best-fitting univariate distributions.
- **Secondary approach:** If copulas were unavailable or insufficient data were present, a multivariate normal distribution was fitted to standardized parameter vectors (mean and covariance estimated from scaled data). (unused)

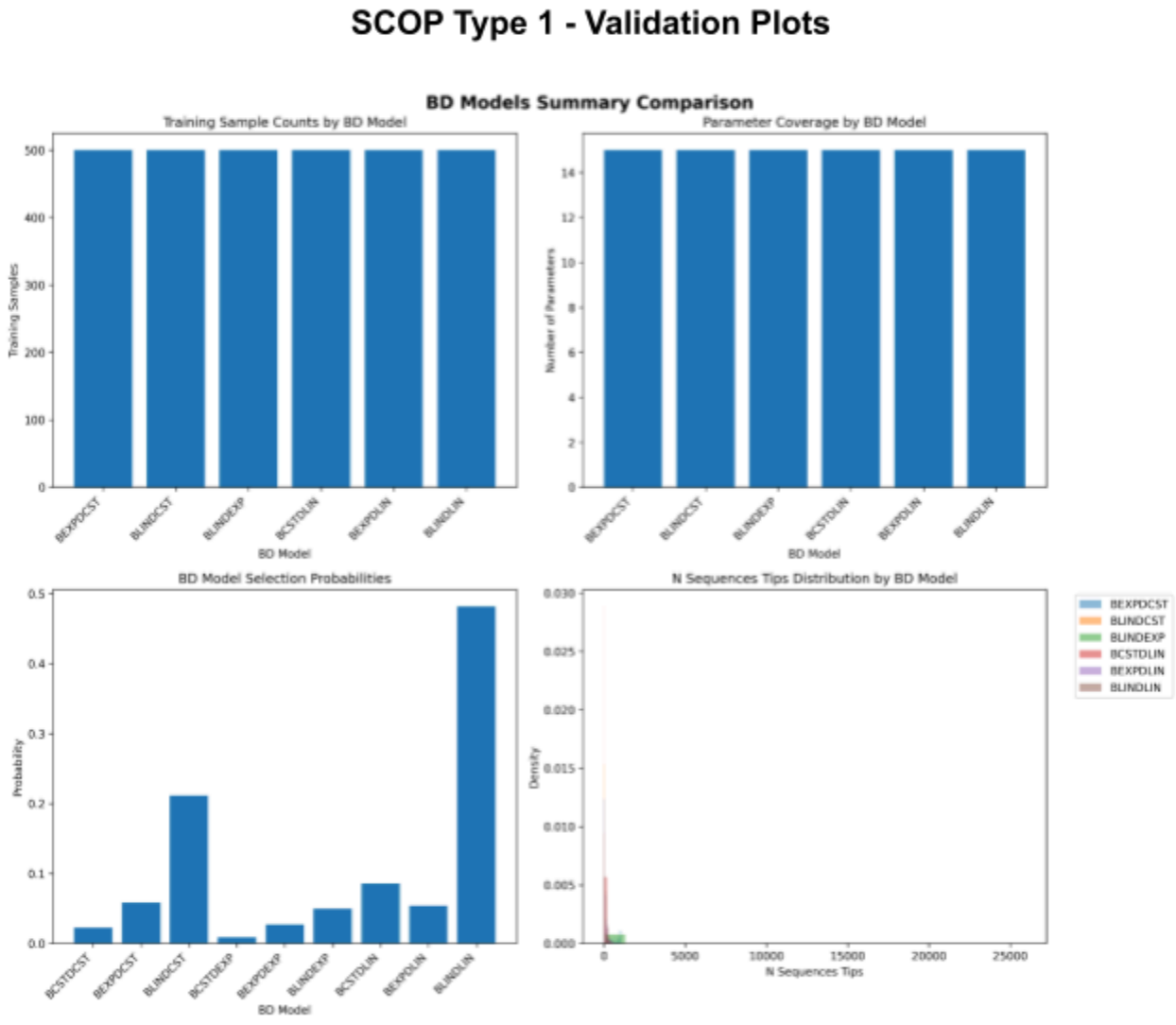
Each fitted joint distribution included the parameter set, sample size, and transformation information.

Parameter Sampling: New parameter sets were generated by sampling from the fitted joint distributions under BD model-specific probabilities. The sampling process included:

1. **BD model selection:** A model was chosen probabilistically, weighted by empirical BD model frequencies.
2. **Random sampling:** A single parameter vector was drawn from the model's copula or multivariate normal distribution.
3. **Rejection filtering:** Samples outside empirical interquartile ranges (±1.5 IQR) were rejected, as were unrealistic values (e.g., negative rates, too few or too many tips).
4. **Tree survival constraint:** To ensure sampled parameters yielded viable trees, a forward **birth–death simulation**(using DendroPy) was run. Parameter sets producing only extinct lineages were discarded.
5. **Bias correction:** For log-transformed parameters, corrections were applied to adjust back-transformed means toward the empirical alignment-level averages.

Validation of Fitted Distributions: The quality of distributional fits was assessed through resampling validation:

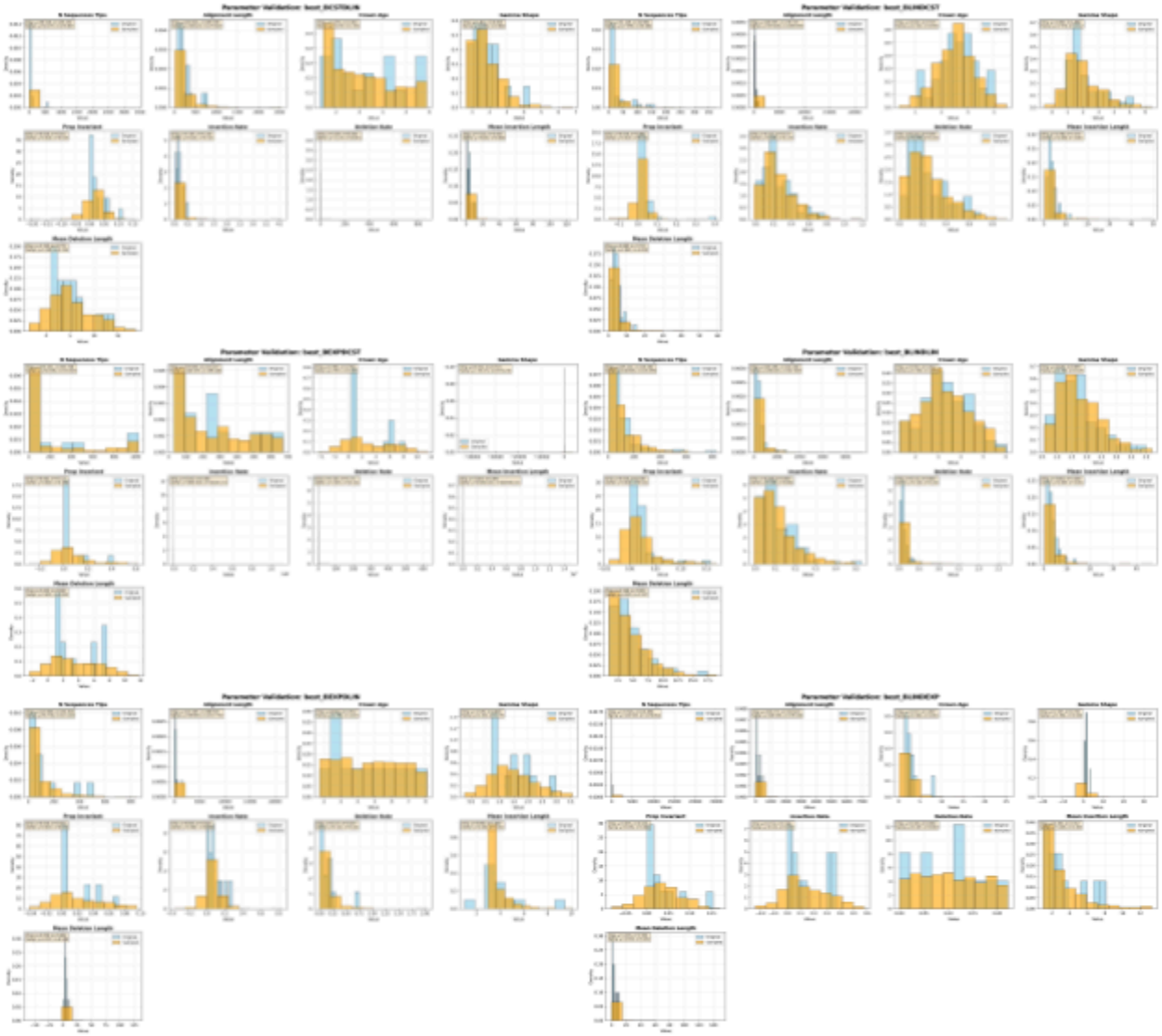
- For each BD model, 500 samples were drawn from the fitted distribution.
- Histograms of simulated vs. empirical parameter distributions were compared for key variables (e.g., crown age, substitution rate heterogeneity, indel rates).
- Model-level summary plots included BD model sample counts, parameter coverage, model probabilities, and cross-model comparisons of parameters such as number of sequences per alignment.



Export for Simulation: The final step exported parameter samples in a format suitable for INDELible simulations. This included:

- Back-transformation of log/logit parameters to original scales.
- Conversion of proportions to valid (0,1) ranges and rounding of integer-valued parameters (e.g., number of sequences, alignment length).

- Computation of total indel rate (sum of insertion and deletion rates).
- One-hot encoding of BD models to mark the diversification regime of each parameter set.
- Export to CSV along with summary statistics (means, variances, quartiles).



This pipeline ensures that simulated datasets reflect both the empirical diversity of evolutionary parameters and the correlations between them, producing biologically realistic parameter sets conditioned on BD model structure.

Generating Guide Trees

To generate phylogenetic trees consistent with sampled tree-shape statistics, I developed a custom Python workflow(`tree_gen_bd.py`). This pipeline integrates birth–death

process simulation with simulated annealing optimization, refining trees until they approximate a user-specified target of topological balance and diversification tempo.

Birth–Death Simulation Framework: Trees were generated using a birth–death (BD) diversification model implemented in DendroPy (vX.X). The birth rate (λ) and death rate (μ) were parameterized as either constant (CST), exponential (EXP), or linear (LIN) functions of time. For time-varying regimes, the present-day rates were back-calculated to obtain ancestral rates, and time was discretized into a fixed number of intervals. Within each interval, lineages were propagated forward according to probabilistic birth (speciation) and death (extinction) events. Lineages going extinct were removed, and only trees with at least one surviving lineage at the present were retained.

Sampling supplied the crown age (time of root), number of taxa, present-day rates, and time-scaling coefficients (α_birth, α_death). The script supported all nine BD model classes (e.g., BCSTDCST, BEXPDCST, BLINDLIN, etc.), allowing independent specification of functional forms for λ(t) and μ(t).

Tree Statistics and Target Metrics: To evaluate tree shape, two statistics were calculated for each simulated tree:

- *Normalized Colless imbalance* – a measure of tree asymmetry (balance of splits).
- *Pybus–Harvey γ-statistic* – a measure of lineage accumulation over time relative to a pure-birth expectation.

These were compared against the sampled target vector ([target_colless, target_gamma]), with similarity quantified using the Euclidean distance between the observed and target statistics.

Simulated Annealing Optimization: Because raw BD simulations deviated substantially from the target statistics (as per observation), I implemented a simulated annealing search procedure to iteratively refine trees:

1. *Initialization:* A tree was generated under the chosen BD model. Its statistics and distance to the target were computed.
2. *Proposal moves:* At each iteration, a local tree rearrangement was performed to generate a candidate neighbor tree. Two types of moves were available:
    - **Nearest Neighbor Interchange (NNI):** Randomly swapped subtrees around an internal edge.

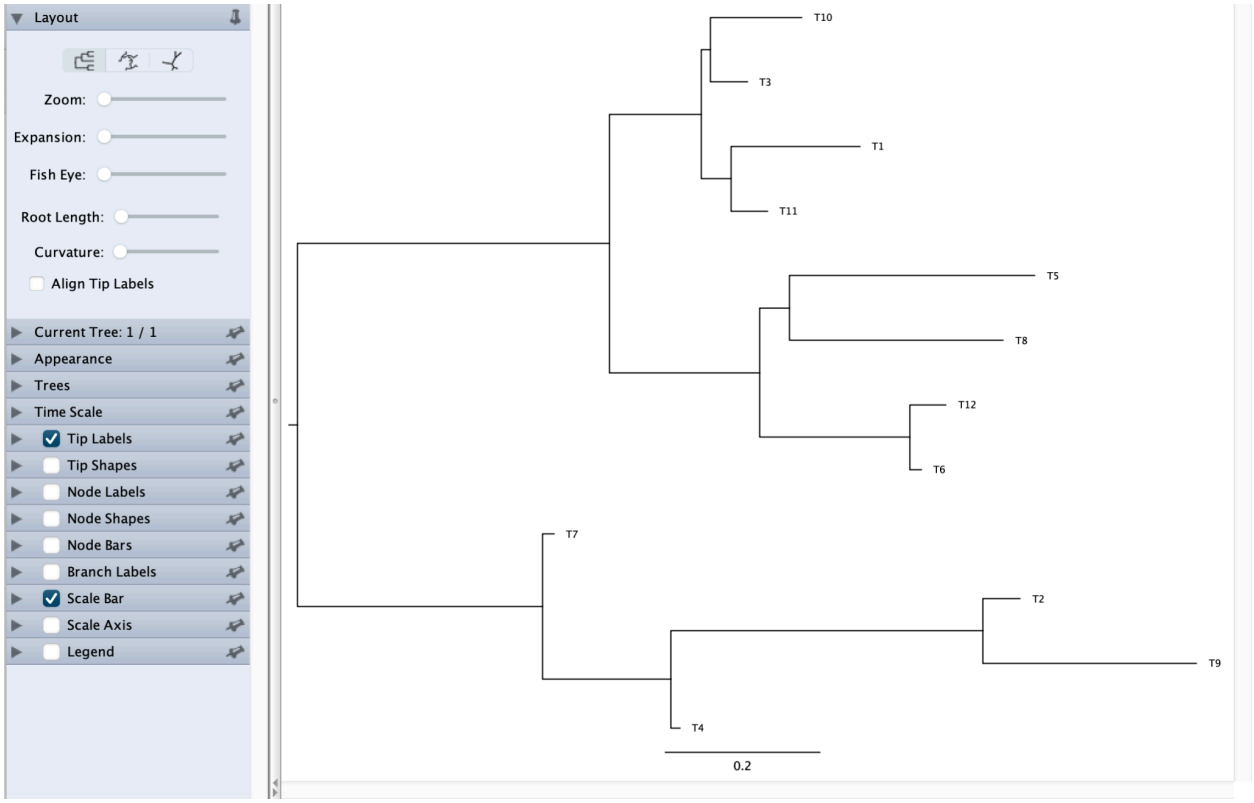
- **Subtree Pruning and Regrafting (SPR):** Randomly detached a subtree and regrafted it to another edge, with node ages reassigned to maintain temporal consistency.

3. *Acceptance criterion:* The candidate tree was accepted if it decreased the distance to the target. If not, it was accepted with probability exp(–Δ/temperature), where Δ is the increase in distance and temperature is the current annealing parameter.
4. *Cooling schedule:* The temperature decreased geometrically (T ← T * cooling_rate) until it fell below a minimum threshold or the maximum number of iterations was reached.

Throughout the optimization, the best tree (lowest distance to target) was tracked and updated whenever a superior candidate was found.

Output: The final optimized tree was written to file in Newick format. Along with the tree, the script reported the final best-fit Colless and γ-statistics, the distance to the target, and the simulated annealing trajectory (distance improvements across iterations).

Sample Tree (SCOP Type 1, Experiment 1, Sequence 1):



INDELible Simulation

To generate protein sequence data with realistic insertion and deletion dynamics, I used INDELible (vX.X), a widely used phylogenetic simulator that models both substitutions and indels under maximum likelihood–parameterized models. For each replicate dataset, the pipeline automatically generated a control file specifying substitution, indel, and tree parameters.

Substitution Model Specification

- *Matrix:* I used the LG model of amino acid substitution, consistent with empirical fits from Pfam alignments.
- *Rate heterogeneity:* Across-site variation was modeled with a discrete gamma distribution (5 categories), parameterized by the gamma shape (α) sampled from empirical alignments.
- *Invariant sites:* A proportion of invariant sites (p_inv) was included, also carried over from empirical estimates.
- *Frequencies:* Amino acid frequencies were either fixed to LG defaults or updated to reflect observed Pfam-specific frequencies, depending on the replicate.

Indel Model Specification: Indels were simulated using negative binomial (geometric) length distributions with empirically scaled rates. For each replicate:

- *Insertion and deletion rates:* The per-site per-branch insertion and deletion rates were set equal (symmetric constraint), matching empirical estimates from alignment analysis. These values directly controlled the frequency of gap events along the tree.
- *Indel extension probabilities:* Insertions and deletions followed a geometric distribution, with the extension probability parameter (p) controlling the mean indel length. Specifically:
    - The probability of extending an indel by one additional residue was p_ext.
    - The mean indel length was calculated as 1 / (1 – p_ext).
    - I enforced symmetry by setting insertion and deletion extension probabilities equal.
- Constraints**:** To prevent excessive sequence length divergence, maximum indel lengths were capped (e.g., 100 residues), consistent with previous simulation studies.

Tree Integration: Each replicate's guide tree (generated via BD simulation with simulated annealing optimization; see above) was supplied as the evolutionary scaffold. Branch lengths were in substitutions/site, ensuring correct scaling of both substitution and indel processes. Indels accumulated along branches according to the assigned rates and lengths, producing both observed extant sequences and the true ancestral sequences at internal nodes.

Output Organization: Indelible produced the following files per replicate–

- *sequences.fasta* – extant sequences at the tips of the tree, with gaps introduced by simulated indels.
- *alignment.fasta* – the "true" multiple sequence alignment containing both substitutions and indels.
- *root.fasta* – the simulated ancestral root sequence.
- *ancestral_sequences/* – internal node sequences for the entire tree.
- *guide.tree* – the phylogenetic tree in Newick format.
- *control.txt* – the Indelible control file, archived for reproducibility.

Sample Control File (SCOP Type 1, Experiment 1, Sequence 1)

```
[TYPE] AMINOACID 1

[MODEL] modelname
   [submodel] LG
   [rates] 0.0027955525823731 1.8572344303576052 0
   [insertmodel] NB 0.2540312363048775 1
   [deletemodel] NB 0.19035489647947484 1
   [insertrate] 0.1807644134358665
   [deleterate] 0.2512426098671751

[TREE] treename
((((T10:0.11865619276654615,T3:0.04793393598395823):0.011315554829437
025,(T1:0.16600975060946332,T11:0.04645831453823457):0.03873929862426
5395):0.11865619276654615,((T5:0.3165914290332732,T8:0.27581082173867
144):0.038739298624265395,(T12:0.04645831453823457,T6:0.0146967993495
92338):0.19404867945699156):0.19404867945699156):0.4030592667566244,(
T7:0.014696799349592338,((T2:0.04793393598395823,T9:0.275810821738671
44):0.4030592667566244,T4:0.011315554829437025):0.16600975060946332):
0.3165914290332732);

[PARTITIONS] partitionname
  [treename modelname 533]
```

```
[EVOLVE] partitionname 1 simulated
```

Methods: Evaluation Pipeline

Historian Source Modifications

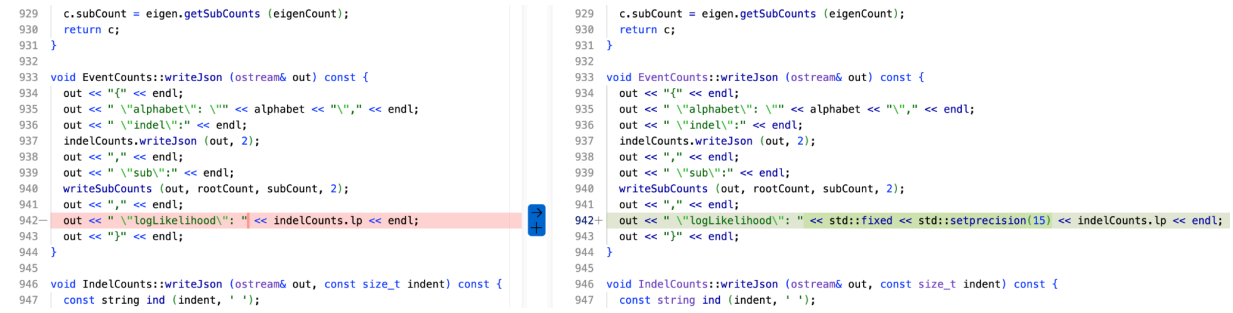
Current implementations of Historian only record likelihood at each MCMC iteration with the -trace flag. In order to monitor MCMC moves to later evaluate mixing/convergence, I had to make some modifications to the Historian source code. These changes saved the following new parameters to the trace file:

- *Log Likelihood:* The log-likelihood of the current MCMC state (tree & alignment)
- *Indels*: The number of indels in the current MCMC state alignment
- *Substitutions:* The number of substitutions in the current MCMC state alignment
- *Mean Indel Length*: Mean length of indels in alignment
- *Indel Rate*: Rate of indels in alignment

These parameters were selected to mimic the current output format for a BAli-Phy trace file for ease of comparison.

Git Code Diff:

model.cpp (ln. 942)

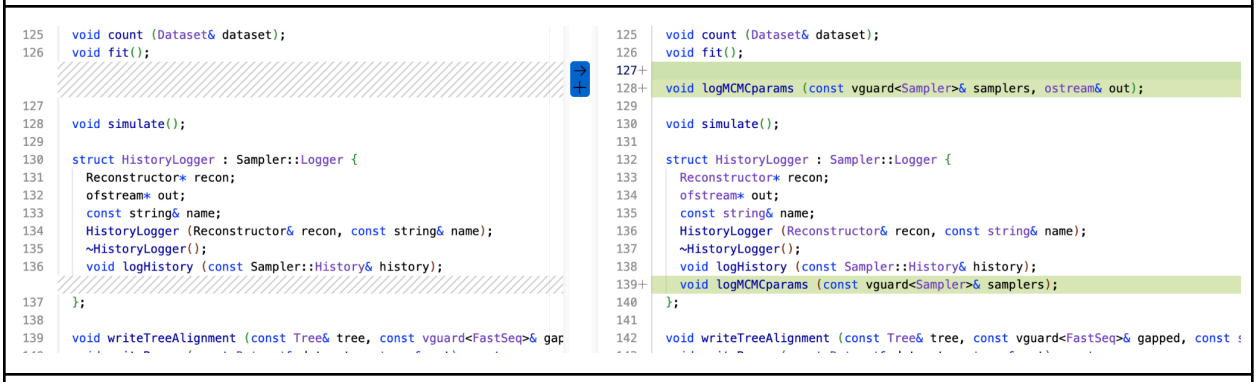


```
 c.subCount = eigen.getSubCounts (eigenCount);
 return c;
}

void EventCounts::writeJson (ostream& out) const {
 out << "{" << endl;
 out << " \"alphabet\": \"" << alphabet << "\"," << endl;
 out << " \"indel\":" << endl;
 indelCounts.writeJson (out, 2);
 out << "," << endl;
 out << " \"sub\":" << endl;
 writeSubCounts (out, rootCount, subCount, 2);
 out << "," << endl;
-out << " \"logLikelihood\": " << indelCounts.lp << endl;
 out << "}" << endl;
}

void IndelCounts::writeJson (ostream& out, const size_t indent) const {
 const string ind (indent, ' ');
```

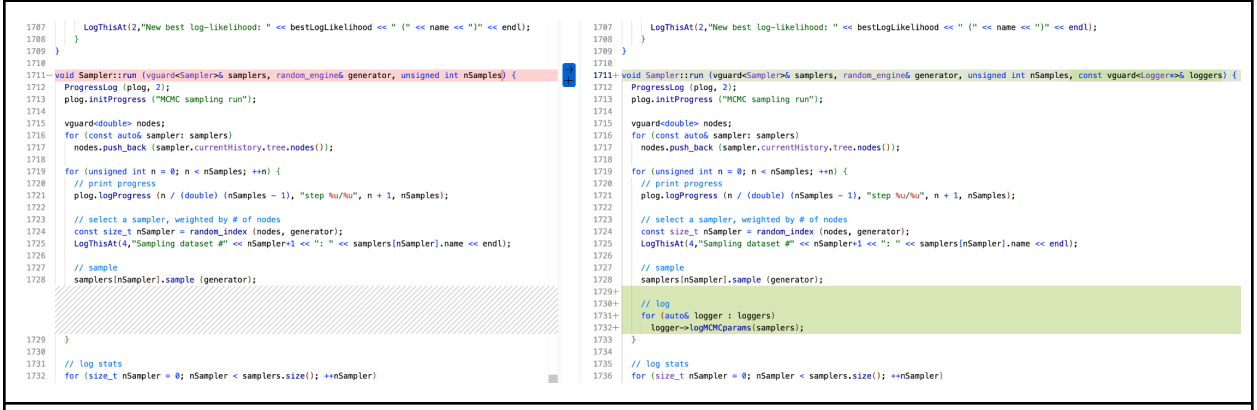
```
 c.subCount = eigen.getSubCounts (eigenCount);
 return c;
}

void EventCounts::writeJson (ostream& out) const {
 out << "{" << endl;
 out << " \"alphabet\": \"" << alphabet << "\"," << endl;
 out << " \"indel\":" << endl;
 indelCounts.writeJson (out, 2);
 out << "," << endl;
 out << " \"sub\":" << endl;
 writeSubCounts (out, rootCount, subCount, 2);
 out << "," << endl;
+out << " \"logLikelihood\": " << std::fixed << std::setprecision(15) << indelCounts.lp << endl;
 out << "}" << endl;
}

void IndelCounts::writeJson (ostream& out, const size_t indent) const {
 const string ind (indent, ' ');
```

recon.cpp (ln. 1292-1327)

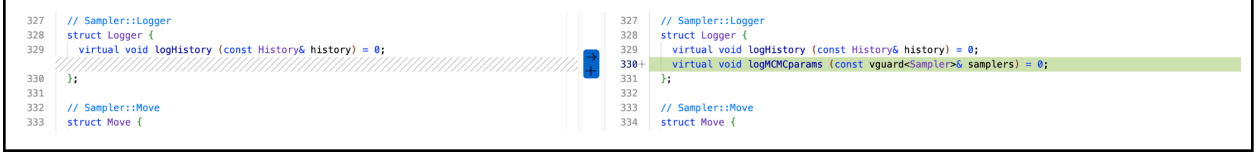
recon.h (ln. 127-139)



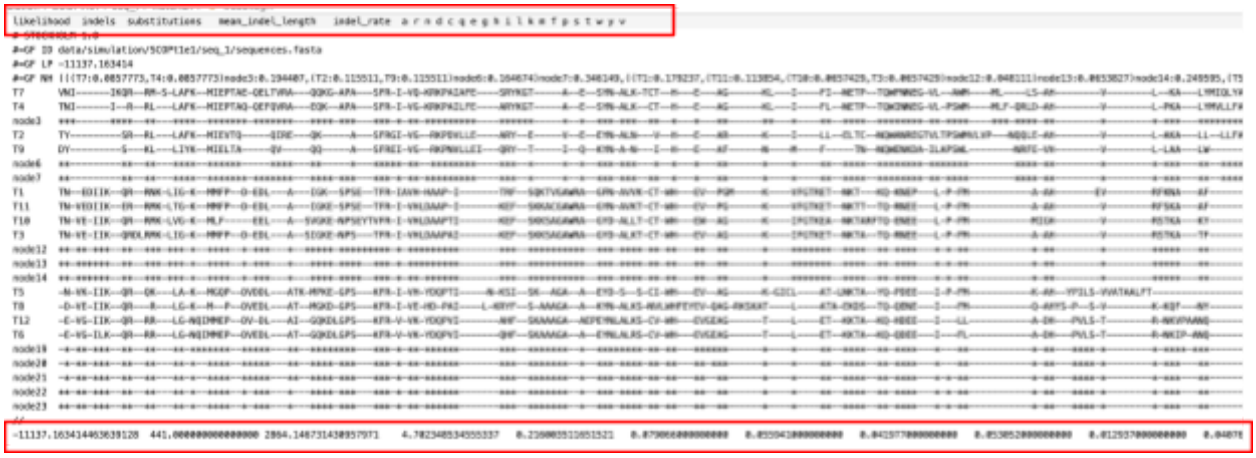
sampler.cpp (ln. 1711-1732)



sampler.h (ln. 330)



Sample Trace File (SCOP Type 1, Experiment 1, Sequence 1, Iteration 0):



Historian Control Setup

I benchmarked Historian (vX.X) using a standardized process to ensure standardization of procedure with BAli-Phy for an equal test.

Model Configuration: Historian was provided with the previously defined LG08 substitution matrix used in INDELible simulations. The insertion and deletion rates were taken from sampling and used to create one symmetric "indel rate" by averaging the two (for standardization with BAli-Phy). The indel extension rate was computed by 1 / (1 – μ), where μ is the average indel length. These were supplied via custom JSON model files, including the LG08 parameters.

- Across-site variation was modeled with 5 discrete gamma categories, parameterized by the sampled gamma shape (α).
- Prop. Invariant Sites (p_inv) was not provided as Historian does not support it.

MCMC Execution: Historian was run with the -mcmc flag on each iteration, with a total of 100k iterations per sequence. Posterior samples were written to a trace file with the -trace flag in addition to a running log of the stdout and stderr with -v5. The chain was halted upon reaching a time constraint specified by the experiment.

Sample Command (SCOP Type 1, Experiment 1, Sequence 1):

```
./tools/historian reconstruct -seqs
data/simulation/SCOPt1e1/seq_1/sequences.fasta -mcmc -model
data/simulation/SCOPt1e1/seq_1/historian/lg.json -gamma 5 -shape
1.8572344303576052 -samples 2000 -trace
data/simulation/SCOPt1e1/seq_1/historian/trace.log -v5
```

BAli-Phy Control Setup

I also benchmarked with BAli-Phy (vX.X), which jointly estimates alignments, trees, and evolutionary parameters in a fully Bayesian framework.

Model Configuration: BAli-Phy was also provided with the lg08 substitution matrix, and indel rate and extension rates were defined under the default rs07 indel model, which models indel lengths via a geometric-like distribution.

- Across-site variation was modeled with 5 discrete gamma categories, parameterized by the sampled gamma shape (α).
- Prop. Invariant Sites (p_inv) was not provided for standardization with Historian.
- LG08 parameters were fixed for standardization with Historian.

MCMC Execution: BAli-Phy was run with one independent chain per dataset, also with 100k iterations per sequence. The chain was halted upon reaching a time constraint specified by the experiment.

Sample Command (SCOP Type 1, Experiment 1, Sequence 1):

```
bali-phy data/simulation/SCOPt1e1/seq_1/sequences.fasta -A
Amino-Acids -i 48000 -n data/simulation/SCOPt1e1/seq_1/baliphy -S
lg08 +> f(lg08_freq) +> Rates.gamma(5, alpha=1.8572344303576052) -I
rs07(rate=0.43200702330304164, mean_length=4.702348534555336)
```

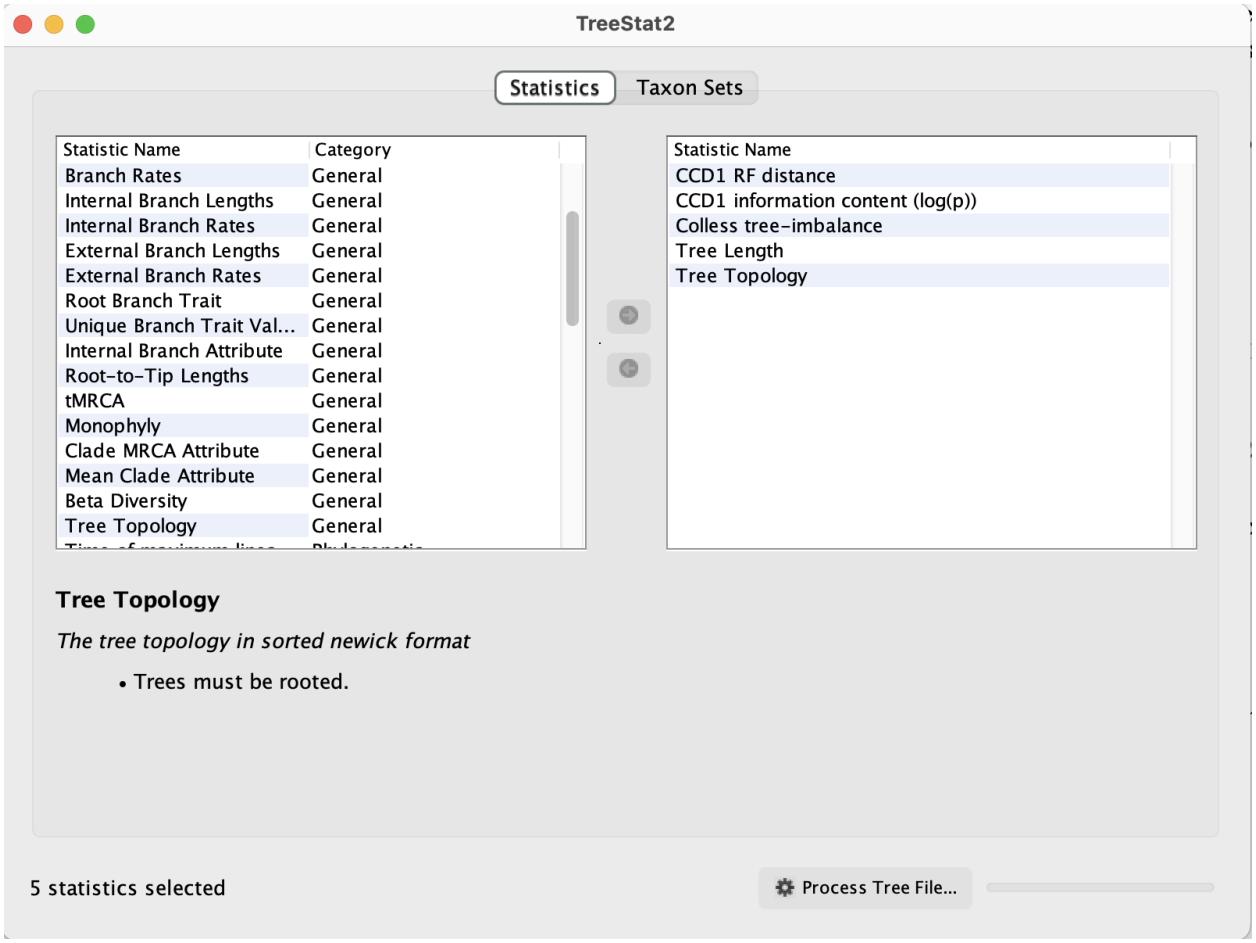
Parsing Trace Files & TreeStat2

After processing, Historian trace files were parsed into a tsv format compatible with Tracer (BEAST software) as well as BAli-Phy's default output with a custom python script (`trace_parser.py`). This parses through the BAli-Phy and Historian posterior files to construct a Nexus-formatted file of all trees compatible with TreeStat2, in addition to posterior file of alignments and a tab-separated value trace file compatible with Tracer.

The posterior tree file was provided to TreeStat2, a BEAST software provided by their in-house "App-Launcher" that provides a trace on tree parameters to analyze convergence of topology. TreeStat2 has support for a variety of metrics, however for these purposes, I only extract the following:

- *CCD1 RF distance*: RF distance to the (conditional-clade) consensus under two cutoffs. Big, frequent swings = the chain is roaming tree space; flat = stuck.

- *CCD1 information content (log p)*: Acts like a tree-only "posterior score." Watch for stationarity and autocorrelation.
- *Tree Topology*: Exports the topology at the current iteration.
- *% Unique Topologies*: The proportion of unique topologies generated
- *Colless Tree Imbalance*: Measure of tree imbalance (defined above)
- *Tree Length:* Sum of all branch lengths in the tree, representing total evolutionary change.



After processing my TreeStat, a lag-1 and lag-k trace of the CCD1 RF distance was computed through the ete3 library and appended to the final trace file (`clean_treestat.py`).

Metrics: SP, TC, RFL Distance, ESS

To assess the accuracy and convergence of reconstruction tools, we employed multiple complementary metrics, each targeting a different aspect of the inference process. For alignment and tree comparisons, we used the posterior decoding alignment (computed

using BAli-Phy's decoding function) and the ccd1-map tree as the representative posterior summary for both Historian and BAli-Phy, ensuring consistent comparisons to the simulated ground truth.

```
cut-range C1.P1.fastas --skip=200 | alignment-chop-internal --tree
treetraceCCD1-MAP.tree | alignment-max > C1-max.fasta
```

Alignment Accuracy: Alignment accuracy was measured using the FastSP algorithm, which provides two widely used statistics:

- *Sum-of-Pairs (SP) score*: the proportion of residue pairs that are correctly aligned in the inferred alignment relative to the true alignment.
- *Total Column (TC) score:* the proportion of entire columns in the inferred alignment that match the true alignment exactly.

Together, these scores quantify both local residue-level alignment accuracy (SP) and global column-level fidelity (TC).

Tree Accuracy: To evaluate phylogenetic inference, we compared inferred posterior trees to the true simulated tree using the Robinson–Foulds (RF) distance, which measures the number of bipartitions present in one tree but not the other, and the Robinson–Foulds Length (RFL) distance, which additionally accounts for differences in branch lengths between trees. These metrics jointly quantify both topological and metric discrepancies.

Convergence Diagnostics: To evaluate the quality of MCMC sampling, we computed the Effective Sample Size (ESS) for posterior likelihood traces and parameter trajectories. ESS provides a measure of the number of effectively independent samples obtained after accounting for autocorrelation, and was used to assess convergence after a fixed number of iterations or a fixed wall-clock runtime.

Experimental Design

The primary goal of this experiment was to evaluate the accuracy and efficiency of tree reconstruction and alignment inference across large and diverse synthetic protein datasets. To review, I designed a simulation-based benchmarking framework to provide a ground-truth reference against which competing methods could be compared.

- **Machine Specification:** 2021 iMac 24" (MacOS 15.6), M1, 16 GB memory
- **Tools:** ModelTest-NG, RPANDA, BioPython, Dendropy, Ete3, Copulas, Historian, BAli-Phy, FastSP

- **Parallelization:** Sequences were batched into groups of two that ran simultaneously (total of 4 concurrent process / batch)

Every simulation dataset yielded for each replicate: a true tree topology, extant sequences, and the true multiple sequence alignment.

Experiment 1: Representative Sets (Equal Iterations)

For the first experiment, I used four representative parameter sets, one for each respective SCOP type, with indel rates and extension rates equal to those derived in Annabel's modelfitting of the GGI model. The purpose of this experiment was to analyze Historian and BAli-Phy's performance on realistic and representative datasets, specifically testing long-run convergence, accuracy, and efficiency over a set number of iterations, independent of any extreme parameters.

- *Number of Iterations:* 2000 iterations per sequence to fully evaluate convergence
- *Number of Sequences:* 4 (1 per SCOP dataset) → 8 runs

Experiment 2: Diverse Sets (Equal Time)

In the second experiment, I used 5 diverse parameters sets per SCOP type, allowing indel rates along with other parameters to vary as per sampling restrictions (discussed previously). Here, the purpose of this experiment was to evaluate how Historian and BAli-Phy's performs under more varied conditions (indel length, number of indels, sequence length, etc.) Additionally, this limits each program to an equal runtime.

- *Maximum Run-time:* 2 hours of wall-clock time
- *Number of Sequences*: 20 (5 per SCOP dataset) → 40 runs

Experiment 3: Representative Sets (Equal Time)

Similar to the first experiment, the purpose of this experiment was to analyze Historian and BAli-Phy's performance on realistic and representative datasets, however over a more realistic constraint of time. This experiment used the same set up as Experiment #1, including four representative parameter sets, one for each respective SCOP type. All parameters are kept the same as well.

- *Maximum Run-time:* 7 hours of wall-clock time
- *Number of Sequences:* 4 (1 per SCOP dataset) → 8 runs

Results: Quantitative

Key Decision and Iteration

Throughout the project, I went through innumerable iterations at each step, refining and reworking the code to achieve the evaluation's goals. Some of the most significant decisions/iterations are selected and listed below, along with my reasoning.

Dataset Selection

At the outset, I considered using a single dataset of alignments and trees for each major domain—plants, bacteria, mammals, invertebrates, and others. To extract the necessary data, I first developed a Python script to process trees and alignments from TreeFam using species-specific markers. However, this proved unreliable as TreeFam was deprecated and often contained empty datasets. I then turned to the OrthoMam dataset, initially focusing on mammalian X chromosome sequences shorter than 10k bases, which could be processed into my parameter distributions. Later, after discussions with Ian and Annabel , I recognized the importance of incorporating natural indel diversity across datasets and proceeded with datasets covering each SCOP structural class. To ensure compatibility with RPANDA for topological analysis, all trees were rooted using the midpoint method.

Encoding Birth/Death Model Type

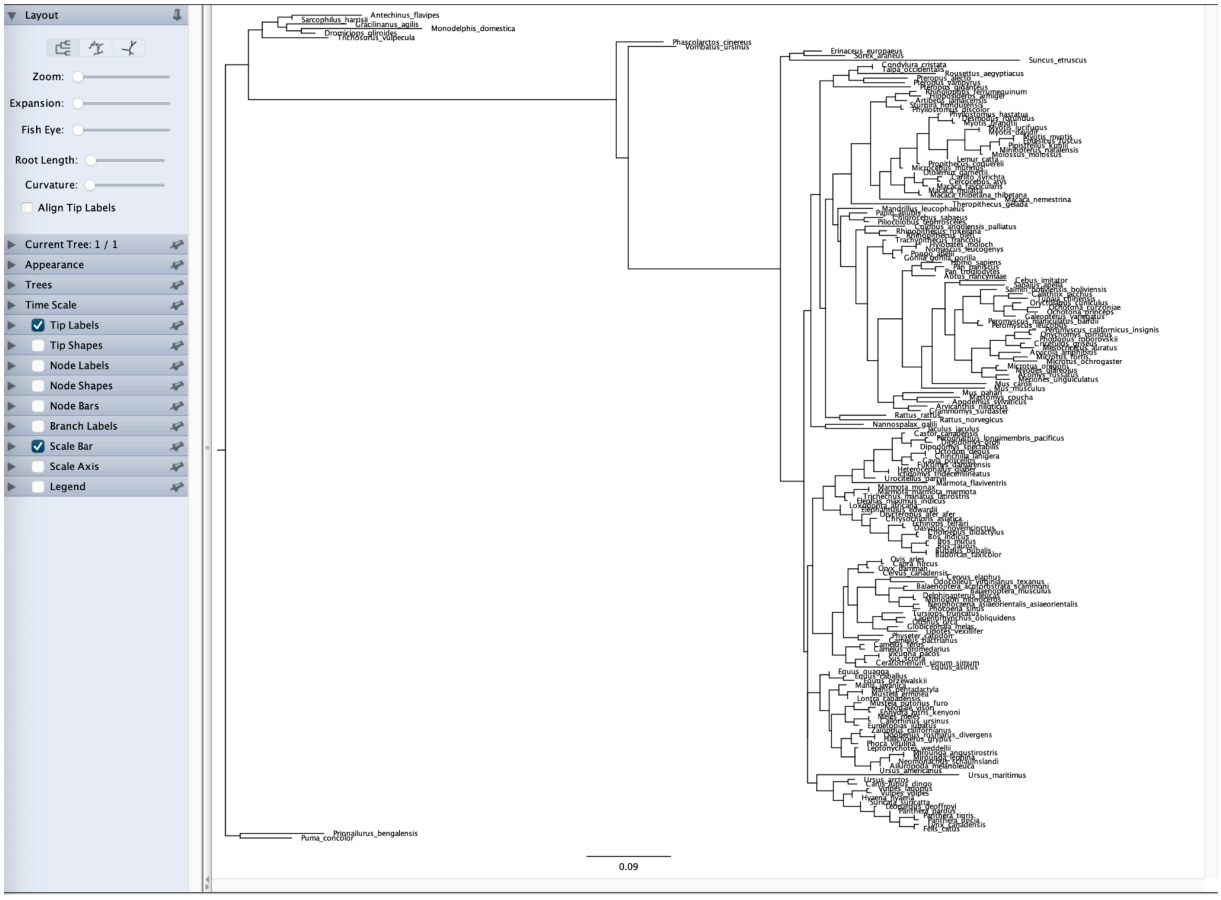
Initially, I encoded the birth–death (BD) model type as a one-hot encoded vector, selecting the best-fitting model during sampling as the one with the highest probability. However, this approach created inconsistencies because other key parameters—such as extinction rates, birth rates, or the alpha value—did not always reflect the model definition. For example, constant birth models sometimes specified explicit rates, while certain non-constant models lacked specified rates. To address this, I pivoted to a mixed scheme where each BD model type was fit with its own distribution, and the model itself was selected according to a fixed frequency.

Generating Birth/Death Trees

My strategy for generating BD trees evolved through multiple stages. The first method relied on using AIC scores from RPANDA to identify the best BD and coalescent models, then splitting the tree optimally so that the first portion was generated by the BD model

and the second by the coalescent. While this approach incorporated statistical model fit, it often produced highly inconsistent trees that did not resemble the dataset from which they were derived. In the second method, I used OrthoMam sequences, which generally adhered to the mammalian “tree of life.” I constructed a topological consensus tree by comparing splits and using this as a reference. For each sequence tree, I then calculated the Robinson–Foulds (RF) distance to the consensus, iteratively applying subtree prune-and-regraft (SPR) moves until a threshold RF distance was achieved. Although effective for OrthoMam, this method was less applicable to the SCOP datasets, which lacked a common reference structure. Furthermore, RF distance alone provided little insight into the underlying tree structure.

Mammalian Topological Consensus Tree:



The third and final method achieved the best balance of rigor and representativeness. For each SCOP dataset tree, I recorded both the Colless index and gamma statistic to characterize the distribution. During simulation, I generated trees under the best-fitting BD model from RPANDA in time increments, and then applied simulated annealing to converge toward a target Colless and gamma value sampled from the empirical

distribution. This procedure consistently produced trees more representative of the original datasets. To avoid sampling errors that resulted in trees going extinct, I introduced additional criteria ensuring that a viable tree was first generated before proceeding with further iterations.

INDELible instead of Indel-Seq-Gen

Early attempts to use Indel-Seq-Gen revealed several technical challenges that ultimately made it impractical for this study. The software relies on an older C++ version with conflicting namespaces, including reserved variable names such as beta and next, which prevented successful compilation without direct modification of the source code. Additional problems arose when attempting to set a custom substitution model (e.g., LG08, which is not available by default). The Indel-Seq-Gen documentation specifies that insertion and deletion length distributions should be listed as <file1>/<file2>, but the program incorrectly parses the "/" as part of the file path rather than as a separator. This error could not be resolved through the use of spaces, quotes, or alternative delimiters, and would have required manual debugging and recompilation of the source code.

Indel-Seq-Gen Documentation:

indel	Format: {$\#_1$, $\#_2$, <*file*$_1$>/<*file*$_2$>} $\#_1$: Max indel size $\#_2$: Indel probability distribution If $\#_2 = 0$: Use Chang & Benner If $\#_2 > 0$: P(ins)=P(del)=$\#_2$ If $\#_2/\#_3$: P(ins)=$\#_2$, P(del)=$\#_3$ <*file*$_1$>/<*file*$_2$>	λ^m λ^p λ^l	These options specify the different indel models and parameters. Only the first two parameters are required. The last parameter, <file$_1$>/<file$_2$ >, can be used to specify the two file names, which provide the user defined insertion length distribution (file$_1$) and deletion length distribution (file$_2$). If no distribution file is provided, the distributions given by Chang and Benner (2004) will be used. If only one distribution file is given, it will be used for both insertions and deletions.

I also explored a potential pivot to the Python-based simulation package Pyvolve, but its input structure is not directly compatible with Indel-Seq-Gen formats and would have required substantial reformatting and preprocessing. Given these limitations, I instead adopted the more widely used simulation framework INDELible, which natively supports the LG08 model and provides built-in functionality for geometric indel length distributions. This choice avoided the need for manual code modifications and ensured compatibility with downstream analyses.

Selecting Experimental Parameter Sets

Experimental parameter sets were chosen to reflect realistic insertion–deletion processes. For the first experiment, I used indel parameters estimated through Annabel's pipeline for

the Crete conference, sampling directly from this distribution under the requirement that sampled values matched the empirical estimates. For the second experiment, I sampled from the same distribution but with broader restrictions (quartile range, number of sequences, successful tree generation, etc.) in order to generate a diverse indel dataset. This two-pronged approach ensured that both empirically grounded and systematically varied parameter sets were available for evaluating Historian's performance.

Limitations and Future Work

My evaluation was limited in scope to two reconstruction tools, Historian and BAli-Phy, both of which were benchmarked under controlled synthetic datasets generated from fitted parameter distributions. While this approach allowed me to establish ground-truth comparisons, future work should extend to additional phylogenetic inference frameworks such as BEAST and MrBayes, which implement alternative MCMC samplers and could provide a broader basis for comparison. All experiments were run on a single machine; scaling to larger HPC clusters or cloud platforms would enable more replicates and more complex datasets. Moreover, the exclusive reliance on synthetic data means that results may not fully capture the complexities of empirical protein families, and future studies should incorporate curated biological datasets. Additionally, my modifications to the Historian source code, while necessary to enforce consistency with simulation parameters, may have introduced computational slowdowns relative to the original implementation.

Future Project Trajectory

In this work, I developed multivariate parameter distributions describing a range of datasets, stratified by SCOP protein classes, and encompassing diverse evolutionary features such as substitutional variation, indel dynamics, and tree shape statistics. These distributions provided the basis for generating realistic synthetic datasets that capture the heterogeneity observed across protein families.

Looking ahead, there are several directions for improvement. Expanding the training corpus to incorporate a broader range of empirical alignments will increase the robustness and generalizability of the fitted distributions. Additional work on parameter transformations and modeling frameworks may also yield better fits, particularly for long-tailed or strongly correlated parameters. Beyond SCOP, incorporating new groups and datasets—such as RNA families or other curated protein databases—would further extend the applicability of this approach. Ultimately, I envision that synthetic datasets

generated through these empirically grounded models could become a **standard practice in the evaluation of ancestral reconstruction methods**, reducing bias introduced by reliance on a single guide alignment and providing a more comprehensive and equitable benchmark for future tools.

A Personal Note

This project was a valuable learning opportunity for me. In the process of developing the framework, I read about statistical phylogenetics (including MCMC methods, Markov Chains, Likelihood methods, finite-state transducers, and many more) and was able to collaborate with the lab on many of these topics. Additionally, I learned a lot about related projects in the lab and working in research as a team. Although my primary goal this summer was preparing the evaluation for publication, I would love to stay in contact in the future and possibly come back to the lab for events like conferences, presentations, etc.

Key Tool Usage and References

Holmes, I. H. (2016). Historian: accurate reconstruction of ancestral sequences and evolutionary rates. *Bioinformatics*, *33*(8), 1227–1229. https://doi.org/10.1093/bioinformatics/btw791

Gupta, M., Zaharias, P., & Warnow, T. (2021). Accurate large-scale phylogeny-aware alignment using BAli-Phy. *Bioinformatics*, *37*(24), 4677–4683. https://doi.org/10.1093/bioinformatics/btab555

Darriba, D., Posada, D., Kozlov, A. M., Stamatakis, A., Morel, B., & Flouri, T. (2019). ModelTest-NG: a new and scalable tool for the selection of DNA and protein evolutionary models. *Molecular Biology and Evolution*, *37*(1), 291–294. https://doi.org/10.1093/molbev/msz189

Morlon, H., Lewitus, E., Condamine, F. L., Manceau, M., Clavel, J., & Drury, J. (2015). RPANDA: an R package for macroevolutionary analyses on phylogenetic trees. *Methods in Ecology and Evolution*, *7*(5), 589–597. https://doi.org/10.1111/2041-210x.12526

Mistry, J., Chuguransky, S., Williams, L., Qureshi, M., Salazar, G. A., Sonnhammer, E. L. L., Tosatto, S. C. E., Paladin, L., Raj, S., Richardson, L. J., Finn, R. D., &

Bateman, A. (2020). Pfam: The protein families database in 2021. *Nucleic Acids Research*, *49*(D1), D412–D419. https://doi.org/10.1093/nar/gkaa913

Tang, X., & Wong, D. F. (2001). FAST-SP. *Association for Computing Machinery*, 521–526. https://doi.org/10.1145/370155.370523

Huerta-Cepas, J., Serra, F., & Bork, P. (2016). ETE 3: Reconstruction, analysis, and visualization of phylogenomic data. *Molecular Biology and Evolution*, *33*(6), 1635–1638. https://doi.org/10.1093/molbev/msw046

Sukumaran, J., & Holder, M. T. (2010). DendroPy: a Python library for phylogenetic computing. *Bioinformatics*, *26*(12), 1569–1571. https://doi.org/10.1093/bioinformatics/btq228

Appendix and Other Resources

Summary of all Collected Parameters

Collected Parameter	Summary
Pfam Identifier	Unique accession code identifying the Pfam protein family analyzed.
# of Sequences	Total number of sequences included in the alignment.
Alignment Length	Number of aligned positions (columns) in the multiple sequence alignment.
Best Substitution Model	Evolutionary model that best fits the observed substitution patterns.
Log-Likelihood of Best Substitution Model	Maximum likelihood score of the alignment under the best-fitting substitution model.
AIC Score of Best Substitution Model	Model fit score based on the Akaike Information Criterion, balancing likelihood and complexity.
BIC Score of Best Substitution Model	Model fit score based on the Bayesian Information Criterion, penalizing model complexity more strongly than AIC.
AA Frequencies	Estimated stationary frequencies of amino acids in the

	alignment.
Gamma Shape Parameter	Parameter describing the degree of rate variation among sites.
Prop. Invariant	Proportion of alignment sites estimated to be evolutionarily invariable.
Estimated Insertion Rate	Average rate of insertion events per site or branch
Estimated Deletion Rate	Average rate of deletion events per site or branch.
Insertion Events	Total number of inferred insertion events across the phylogeny.
Deletion Events	Total number of inferred deletion events across the phylogeny.
Mean Insertion Length	Average length of inserted sequence fragments.
Mean Deletion Length	Average length of deleted sequence fragments.
Total Gaps	Number of gap characters in the alignment introduced by insertions or deletions.
Indel to Sub. Ratio	Ratio of insertion/deletion events to substitution events.
# of Internal Nodes	Number of branching points (non-leaf nodes) in the phylogenetic tree.
Tree Max. Depth	Maximum path length from the root to the most distant leaf in the phylogenetic tree.
Colless Index	A measure of the imbalance or asymmetry in the phylogenetic tree.
Normalized Colless Index	Colless index scaled to account for tree size, allowing comparison across trees.
Pybus & Harvey γ	Statistic testing whether branching events deviate from a constant-rate diversification model.
Pybus & Harvey γ p-value	Significance level of the γ statistic, indicating whether rate variation is supported.

Speciation Rate	Estimated rate at which new lineages arise over evolutionary time.
Extinction Rate	Estimated rate at which lineages go extinct over evolutionary time.
Net Diversification	Speciation rate minus extinction rate, reflecting overall lineage growth.
Relative Extinction	Ratio of extinction to speciation, showing how much turnover occurs in lineages.
Best Tree Log Likelihood	Maximum likelihood score of the reconstructed phylogenetic tree.
Best Tree AIC	Akaike Information Criterion score for the best-fitting tree, balancing fit and complexity.
Best Tree BIC	Bayesian Information Criterion score for the best-fitting tree, penalizing complexity more strongly.
Best Birth-Death Model	Birth-death process model that best explains lineage diversification in the tree.
Birth Coefficient Parameter	Fitted parameter for the speciation (birth) rate in the birth-death model.
Death Coefficient Parameter	Fitted parameter for the extinction (death) rate in the birth-death model.
Best Coalescent Model	Coalescent model that best fits the timing of coalescent events in the tree.
# of Tips	Total number of terminal taxa (leaves) in the phylogenetic tree.
Tree Length	Sum of all branch lengths in the tree, representing total evolutionary change.
Crown Age	Estimated time to the most recent common ancestor of all sampled lineages.
All Trees Models AIC Ranking	Comparative ranking of tree models based on their AIC scores.