Historian Evaluation Proposal

Original Document (2024): Historian Evaluation Procedure

Evaluation Goals

This evaluation will focus on benchmarking and evaluating ancestral sequence sequence reconstructors that jointly estimate trees and alignments, specifically Historian and BAliPhy v3.

Typically, such reconstructors rely on a guide tree to construct MSAs, or conversely, utilize a singular alignment to develop a tree. As such, evaluation generally utilizes a prior "true" tree or alignment from biological datasets (ex. BAliBASE, Mattbench, Homstrad, and Sisyphus) that have been derived through some previous software (such as RAxML). This leads to propagating bias when testing joint-reconstructors

Generating model trees → Simulating evolution → Running Historian/BAliPhy → Comparison to "True" trees, alignments, parameters

Historian: https://github.com/evoldoers/historian
BAli-Phy: https://github.com/bredelings/BAli-Phy
DendroPy: https://jeetsukumaran.github.io/DendroPy/
Indel-Seq-Gen: http://bioinfolab.unl.edu/~cstrope/iSG/#Software

Some more reference evaluations

- ProtPal– https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0034572
- BAliPhy (Original) – https://academic.oup.com/sysbio/article-abstract/54/3/401/1727336?redirectedFrom=fulltext&login=false
- Another coestimation– https://pubmed.ncbi.nlm.nih.gov/19541996/
- Evaluating BAliPhy (MS Aligner)– https://academic.oup.com/sysbio/article/68/3/396/5133545
- Evaluating BAliPhy 2 (phylogeny aware... MS aligner)– https://academic.oup.com/bioinformatics/article/37/24/4677/6329826
- MCMC Exploration – https://pmc.ncbi.nlm.nih.gov/articles/PMC4395846/

- Historian (Original) – https://academic.oup.com/bioinformatics/article/33/8/1227/2926463#394512885
- Tree comparison – https://academic.oup.com/sysbio/article-abstract/64/2/205/1630737?redirectedFrom=fulltext

Step 1: Generating model tree topologies

Using DendroPy to generate trees from biologically meaningful models:

```
import dendropy

# Birth-death tree (λ = 1.0, μ = 0.5, 20 taxa)
bd_tree = dendropy.simulate.treesim.birth_death_tree(
    birth_rate=1.0, death_rate=0.5, num_extant_tips=20
)

bd_tree.write(path="bd_tree.nwk", schema="newick")
```

Options:

- Birth_death_tree(...) – simulates speciation/extinction
- uniform_pure_birth_tree(...) – yule model
- dendropy.model.coalescent – simulate coalescent trees for population-level data

Dendropy can be used to generate a diverse corpse of tree types including balanced, pectinate, star-like, and bushy in order to ensure that the evaluation covers all evolutionary patterns. Additionally, we can force ultrametricity using **TreeTime**:

```
treetime clock --tree bd_tree.nwk --aln root.fasta --outdir
treetime_out/
```

Step 2: Simulation of evolution

The Dendro-py generated trees would be used as the guide for indel-seq-gen. We would use the GTR model for site evolution

```
./indel-seq-gen -m GTR -l 1000 -r 0.05 -i 0.01 -e 0.01 -d 3 -f
indel_dist.txt -q bd_tree.nwk -o true_simulated.fasta
```

We will simulate various models with 1000, 500, 250, and 100 sequences respectively. Here, parameters are varied across models with a uniform distribution around its mean (specified above)

- Note: Other implementations (like Liu, et. al) extract parameters from the NemATOL alignment of 6862 species of nematode using software such as PAUP

Gap length distribution— we will use three geometric single-event gap-length distributions: short, medium and long with medians of 2, 4, and 7 respectively. The probabilities are borrowed from (Liu, et. al 2009).

Long Gap Length Distribution:

```
[0.1028, 0.0899, 0.0792, 0.0702, 0.0627, 0.0565, 0.0514, 0.0470,
0.0433, 0.0400, 0.0369, 0.0341, 0.0314, 0.0289, 0.0266, 0.0245,
0.0225, 0.0206, 0.0188, 0.0171, 0.0155, 0.0141, 0.0127, 0.0114,
0.0100, 0.0087, 0.0075, 0.0063, 0.0052, 0.0042]
```

Medium Gap Length Distribution:

```
[0.2012, 0.1600, 0.1280, 0.1024, 0.0819, 0.0655, 0.0524, 0.0419,
0.0336, 0.0268, 0.0215, 0.0172, 0.0137, 0.0110, 0.0088, 0.0070,
0.0056, 0.0045, 0.0036, 0.0029, 0.0023, 0.0018, 0.0015, 0.0012,
0.0009, 0.0008, 0.0006, 0.0005, 0.0004, 0.0003, 0.0002]
```

Short Gap Length Distribution:

```
[0.4613, 0.2527, 0.1545, 0.0896, 0.0419]
```

- Insertion and deletion probabilities here are set identically.

In total, there will be **50 different models** varying in parameters including tree topology, number of taxa, gap lengths, rate heterogeneity, tree heights, and general substitution/indel parameters to represent a wide variety of biological realities. This models will be simulated various times to create a comprehensive testing bench.

Step 3: Tree and Alignment Generation

Running both methods (Historian / BAliPhy) without providing true alignments/trees.

To ensure that all of the tests are fair, similar parameters and settings have to be used across Historian and BAli-Phy. For instance:

- All of the runs will be managed by a wrapper script that accounts for batching as well as the wall-clock time of each run.
- If run on a computer cluster, each batch can run in parallel on separate processors. However, no further speed-ups will be applied to either program to ensure authenticity of tests, and both will be run on CPU. The time can be managed through a simple time module.

```
historian -o historian_output.stk true_simulated.fasta

baliphy true_simulated.fasta --iterations 10000 --burnin 2000
```

Experiment #1: Wall-clock time

- Wall clock time can be measured directly through a wrapper script

Experiment #2: Tree Topology Accuracy

Standard tools such as Robinson-Foulds (RF) distance:

```
compareTrees.missingBranch -r true_tree.nwk -e
inferred_tree.nwk
```

in addition to specialized metrics such as “Align”--
https://doi.org/10.1093/bioinformatics/bti720

- A brief overview of these metrics and their characteristics are described by Kuhner and Yamato (2015) https://doi.org/10.1093/sysbio/syu085
- Likelihood-based tree error (log-likelihood under “true” alignment collected)

Experiment #3: Evolutionary Parameters Inference

A primary tool of Historian and BAliPhy are extracting evolutionary parameters from a set of sequences while coestimating the alignment and trees. The accuracy of these evolutionary parameters will be compared to the initial parameter set defined through the evolutionary model and measured through raw accuracy (%).

Experiment #4: Measuring MCMC Exploration & Convergence

Mean Compute Time statistics calculations have been covered here – https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4395846/ (Time to explore nearest vicinity and return back to the local maxima)

> **From the previous document (not fully committed to implementation) ...** In addition, we will be using a modified version of rSPR (rooted subtree-prune-and-regraft) distances available here. This is due to SPR operators being closely related to common rearrangements in phylogenetic programs, making it ideal for analyzing MCMC behavior including mixing. More information is described by Whidden and Matsen (2015). Additionally, due to the accuracy of SPR, its implementation will provide a dynamic testing environment for evaluating Historian under various conditions.

- Let each state = a pair (alignment, tree topology)
- Define an edit distance (in this case RF distance) between trees
- Calculate: MCT($i \leftrightarrow j$) = $E[T_{i \to j}] + E[T_{j \to i}]$
- Average over sampled states

Here, high MCT would signify poor exploration of the posterior while low MCT would signify good mixing.

We can also measure Effective Sample Size (ESS): estimates how many independent samples an MCMC chain has produced. **Formula:**

- ESS = N / (1 + 2 sum(rho_k) over all k)
- Where N is the number of samples and rho_k is the "lag-k autocorrelation" of the trace

This can be tracked through tools like Arviz (python). We can compute ESS for the log posterior likelihood, branch lengths, alignment lengths, etc. Here, low ESS (around below 200) is considered slow mixing/non-convergence.

These two statistics would likely have been implemented within BAliPhy/Historian as diagnostic outputs.

Experiment #5: Alignment Accuracy

Alignment accuracy can simply be measured through software such as FastSP or SPFP/SPFN

```
fastsp -r true_alignment.stk -e historian_output.stk
```

This provides traditional metrics of false positives/negatives from which we can derive metrics such as precision and recall.

Software Structure

The goal of the evaluation software is to be reusable under multiple scenarios (such as the possible web based gamification software for the future...)

```
ancestral_aligner_evaluator/
├── main.py
├── config/
│   └── default.yaml
├── simulation/
│   └── simulate.py
├── runners/
│   ├── historian.py
│   └── bali_phy.py
├── evaluation/
│   ├── compare_alignments.py
│   ├── metrics.py
│   └── tree_distance.py
├── utils/
│   ├── fasta.py
│   ├── trees.py
│   └── io_utils.py
├── results/
│   └── (output files go here)
└── logs/
    └── (log files)
```

```
data/
├── simulated/
│   ├── run_01/
│   │   ├── tree.nwk
│   │   ├── true_alignment.fasta
```

```
│   │   ├── simulated_sequences.fasta
│   │   └── ancestral_sequences.txt
│   ├── run_02/
│   └── ...
├── results/
│   ├── historian/
│   └── bali_phy/
```