Likelihood computation when both average and relative dispersal rates vary across intervals

Background

The P-matrix computation function implemented in the current version of BEAST are designed to work when a branch spanning multiple relative-rate epochs *or* multiple average-rate epochs, but not *both*. Here I designed a toy example to demonstrate that this is the case and an extension to the current implementation in BEAST (contained in this pull request) correctly computes the P matrix along a branch spanning multiple relative-rate epochs *and* multiple average-rate epochs.

Setup

In this toy example I assume a 2-taxon tree which overlaps with four relative-rate epochs and three average-rate epochs. The exact length of each tree branch, the temporal mapping of the epochs, and the exact values of the rates are illustrated in the figure below.

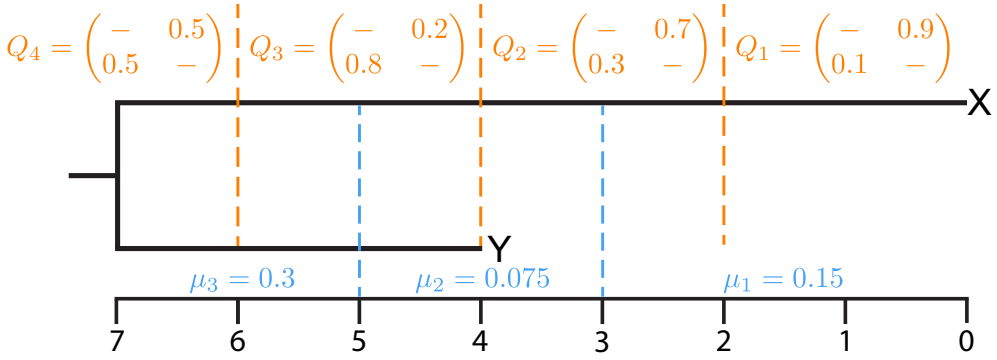

Figure 1: The 2-taxon tree and the Q and μ epochs mapped over the tree.

See `loglik_epochal_toy.R` or `loglik_epochal_toy.xml` for the other parameter values used in this toy example.

I computed the likelihood for each of the four possible `site` patterns (2 tips of the tree times 2 states of the discrete character) using each of the following 5 ways:

1. using R (with some functions I wrote) to correctly compute the P matrix along each branch spanning multiple Q and μ epochs;
2. using R to compute the P matrix along each branch that only correctly deal with case where either Q or μ must be constant over the branch, imitating BEAST's current implementation;
3. using an extended version of BEAST that allows the P matrix to be computed correctly along a branch spanning multiple Q and μ epochs;
4. using the current version of BEAST, and;
5. using Monte-Carlo simulations to generate a large number of datasets (5 million in this experiment) and then counting the frequency of each `site` pattern.

Results

It appears that ways 1 and 3 produce identical log-likelihood values, which are then effectively identical to the values computed using Monte-Carlo simulations; while ways 2 and 4 agree with each other (*i.e.*, produce identical log-likelihood values), the values they produce are distinct from the values computed using Monte-Carlo simulations. To me, these results suggest that: the current implementation in BEAST does not compute the P matrix correctly for a branch multiple Q and μ epochs, and the extension contained in this pull request has fixed the issue.

The exact log-likelihood values computed under each way are listed below (each line contains four values, showing the log-likelihood computed for each of the four possible site patterns: {1,1}, {1,2}, {2,1}, {2,2}):

- Ways 1 and 3 (R/BEAST, parent to child) = −1.4850114, −2.2196994, −0.8398153, −1.4564242.
- Way 5 (Monte-Carlo simulation) = −1.4852762, −2.2186080, −0.8402774, −1.4558202.
- Ways 2 and 4 (R/BEAST, child to parent) = −1.5994150, −2.4388870, −0.7423071, −1.4493919.

Supplementary Info: Readme

- `loglik_epochal_toy.R`: main entry of the R code.
- `loglik_epochal_functions.R`: functions used in `loglik_epochal_toy.R`.
- `loglik_epochal_toy.xml`: BEAST XML template file (which is read in by `loglik_epochal_toy.R` to calculate the likelihoods using BEAST).
- `beast_bothvary.jar` and `beast_onevary.jar`: compiled BEAST java executables that are used to perform ways 3 and 4, respectively; the only difference between the source codes used to compile them is that the former one includes one extra commit (commit `8c1e767f` contained in this pull request).