SELVa: overview

Simulator of Evolution with Landscape Variation (SELVa) is a simulator of sequence evolution that allows the fitness landscape to vary according to user-specified rules.

Overview

SELVa is an event-driven simulator where at any point in time along any branch of the rooted phylogenetic tree, one of the following events can occur: a point mutation in the sequence (governed by the current fitness landscape), or an instantaneous change in the fitness landscape. The mutational process follows the description in Chapter 12.5.4 of *Molecular Evolution: A Statistical Approach (*Yang 2014): briefly, the mutation events are modeled as a Markov chain with inter-event times described by an exponential distribution with the mean parameter derived from the current fitness landscape, and once a mutation event occurs, the allele that is transitioned to is chosen probabilistically according to the stationary distribution of the current fitness vector. Meanwhile, the landscape-change events can occur either deterministically at equal time intervals, or stochastically (in which case the landscape change becomes just another event of the simulation, along with point mutations), and the value of the new landscape is chosen according to user-specified rules. The various options for changing landscapes are described below.

Currently, only point mutations are supported (no indels).

The fitness landscape is specified by a vector giving the fitness of each allele. The probability of a sequence position changing to a character depends only on its "fitness" according to the current landscape; it does not depend on the character that is mutated away from. This limitation will be addressed in future versions of the simulator.

The user provides the phylogenetic tree in the Newick format and a config file that defines the sequence alphabet in and other parameters of the simulation. These parameters specify the overall rules of the simulation (the length of the sequence, the number of processors used, whether to print the intermediate fitness values, etc.), as well as the rules that determine when, where, and how the fitness landscape changes.

The branch lengths of the Newick tree are interpreted as evolution time for the purposes of the simulation, and

The config file

The config file is a whitespace-delimited text file containing parameter-value pairs. The config file options are described in detail in the following sections.

Rules governing the structure of the simulation

The user must specify the phylogenetic tree in the Newick file format and give its name (and path) as the value of the `TREE_FILE` (e.g., `/path/to/tree/file.tre`) parameter in the config file. The user must also provide the allele alphabet as the `ALPHABET` string in the config file.

The natural unit of simulation for SELVa is an "instance" corresponding to one landscape history on the provided phylogenetic tree. For each instance, a sequence of length `LENGTH` is generated at random

from the stationary distribution associated with the initial fitness landscape. Then, over the course of the simulation, the sequence evolves along the phylogenetic tree, subject to the current fitness landscape, while the fitness landscape itself is undergoing changes. Multiple independent landscape histories (instances) can be simulated at the same time – their number is given by the `NUM_INSTANCES` parameter. If more than one processor is available, the runs may be divided among multiple threads, setting the `NUM_THREADS` parameter to a desired number. Note: a single run cannot be multithreaded, so `NUM_THREADS` should be `<= NUM_INSTANCES`. All instances that are run in a single execution of SELVa are governed by the same parameters.

Rules for changes in the fitness landscape

- How is the initial landscape specified?
    - *as a vector in a file*

      In this case, the config should contain the lines:

      ```
      INITIAL_FITNESS file
      FITNESS_FILE /path/to/fitness/file.txt
      ```

      Where `file.txt`, whose path on your system is given by `path/to/fitness/file.txt` should contain a list of space separated numbers that correspond to the fitnesses of each allele given in the `ALPHABET` parameters in the order in which they are listed in the `ALPHABET` string (consequently, the length of the "fitness vector" and the `ALPHABET` string should be the same).
    - *as a probability distribution from which the "fitnesses" of every allele are drawn* (thus, the fitnesses of the allele are iid). Currently supported distributions are the lognormal with μ = `0` and σ that is specified by the user as `DIST_PARAM`, and the gamma distribution, with $\alpha = \beta$ = `DIST_PARAM`. In this cases, the config file should contain the lines:

      ```
      INITIAL_FITNESS lognorm OR INITIAL_FITNESS gamma
      DIST_PARAM param
      ```

      where *`param`* is a real number giving the sole parameter of the distribution (μ or $\alpha = \beta$, as described above)
- When does the landscape change?
    - stochastically. The landscape changes are a Poisson process whose mean rate parameter is specified as `LANDSCAPE_CHANGE_PARAMETER`, which in this case is interpreted as the rate of the Poisson process. In this case, the config file contains the lines:

      ```
      LANDSCAPE_CHANGE_TIMING stochastic
      LANDSCAPE_CHANGE_PARAMETER lambda
      ```

      where *`lambda`* is a real number giving the mean rate λ of the landscape change process.
    - deterministically at equally-spaced times. Either the length of the interval (in tree length units) or the number of intervals should be given (both are specified by the `LANDSCAPE_CHANGE_PARAMETER`, whose interpretation is context-dependent). In the latter case, the interval length will be calculated based on the longest path from the root to the leaf in the tree.

      If the interval length is specified, the config file should contain the lines:

      ```
      LANDSCAPE_CHANGE_TIMING fixed_interval_length
      LANDSCAPE_CHANGE_PARAMETER interval_length
      ```

      where interval_length is a nonnegative real number.

      If the number of desired landscape changes is given, the config file should contain the lines:

LANDSCAPE_CHANGE_TIMING fixed_num_changes
LANDSCAPE_CHANGE_PARAMETER *num_changes*
where num_changes is a natural number.

If landscape change should *never* occur, these parameters can be set appropriately (e.g., set the change timing to fixed interval length and the interval length to 0, etc.).

A note regarding numerical issues. The small loss of precision inherent in using floating-point numbers may lead to artefacts in the landscape timing calculation. One can occur when branch lengths are multiples of the length of the (fixed) inter-change interval. In this case, it is possible for the landscape change to take place sometimes *just before* the branching event (in which case both daughter branches start with the same landscape), and sometimes *just after* the branching event (in which case each daughter branch starts with its own landscape).

- How is the new fitness vector calculated?
  - as a random permutation of the "old" fitness vector (the new vector is guaranteed to be different from the old one)
    In this case the config file should contain the line:
    NEW_FITNESS_RULE shuffle
  - as a random vector drawn from the same distribution as the initial vector. The config file should then contain the line:
    NEW_FITNESS_RULE iid
  - the new landscape is generated from the old one by increasing or decreasing the fitness of the current allele, with the fitnesses of the other alleles remaining unchanged. If the current allele has index *i*, then the new fitness $<f_1^{new}, \ldots f_n^{new}>$ vector has values:
    $f_j^{new} = f_j^{old}$ for $i \neq j$
    $f_i^{new} = f_i^{old} + k \cdot \Delta t$, where k = AGE_DEPENDENCE_COEFFICIENT and Δt is the length of the fixed interval between landscape changes (as determined by LANDSCAPE_CHANGE_TIMING and LANDSCAPE_CHANGE_PARAMETER and described above). If k is negative, the current allele's fitness decreases with time.

    This option does not work with `stochastic` `LANDSCAPE_CHANGE_TIMING`. Also, since this option depends on having a unique current allele, it does not work with sequence `LENGTH > 1` or `SHARED_LANDSCAPE true` (see below for description of this parameter).

    The config file should contain the lines:
    NEW_FITNESS_RULE current_allele_dependent
    AGE_DEPENDENCE_COEFFICIENT k
    where k is the real-valued change in fitness per time unit
- Is the landscape branch-specific, or does it depend only on the length from the root and is shared by all tree branches?
  - If the config file contains the line
    SHARED_LANDSCAPE true
    then the all points in the tree that are at an equal distance from the root share the same landscape (this option, naturally, does not work with allele-specific landscape change).
  - If the config file contains the line

`SHARED_LANDSCAPE false`
then non-overlapping paths in the tree have independent fitness landscapes. If the landscape is updated stochastically, the times of landscape changes will be different as well. **This is the default behavior.**

Advanced parameters concerning the rate of substitution

Understanding these parameters and their explanation is easier if one is familiar with the concept of a substitution-rate matrix and the corresponding stationary distribution vector. The reader is referred to, for example, Computational Molecular Evolution *(Yang 2006).*

Different fitness landscapes (fitness vectors) can lead to different rates at which substitutions take place. It may however be desirable to fix the expected substitution rate to always be 1, so that tree branch lengths would indeed reflect the number of substitutions. This is the approach taken in the `evolver` program that is part of the PAML package. It is the default behavior of SELVa and can also be explicitly selected by setting the `Q_NORMALIZATION` parameter to `constant_rate`.

An alternative approach is to stipulate that the mean substitution rate is 1 only *if the landscape is flat*, i.e., if all elements of the fitness vector are equal, and may deviate from unity otherwise. This is achieved by setting `Q_NORMALIZATION` to `constant_for_flat`.

If the substitution rate is allowed to vary, the user has the option of scaling the stochastic landscape change rate to it by setting `SCALE_LANDSCAPE_CHANGE_TO_SUBSTITUTION_RATE` to `true`. In this case, the actual parameter λ is obtained by multiplying `LANDSCAPE_CHANGE_PARAMETER` by the average substitution rate. This calculation is computationally expensive and the simulation running time.

The technical details of the meaning of this parameter are given in the Apendix.

Output

Output of the simulated sequences

SELVa prints out the sequence(s) generated for each tree node over the course of the simulation in the FASTA format in the file `allnodes.merged.fasta`. If multiple instances were simulated at the same time, the sequence at each node is a concatenation of the sequences generated by each instance (the order of instances is the same for all nodes).

Landscape change information

The user has the additional option of printing out the information about all landscape changes, namely, the time of change and the value of the new vector. This option is turned on by setting `PRINT_LANDSCAPE_INFO` to `true`. **Since keeping track of this information significantly increases the memory usage and may therefore impact the running time as well (by causing the simulation to use virtual memory) this user is advised to turn on this option only if he or she will use this information.**

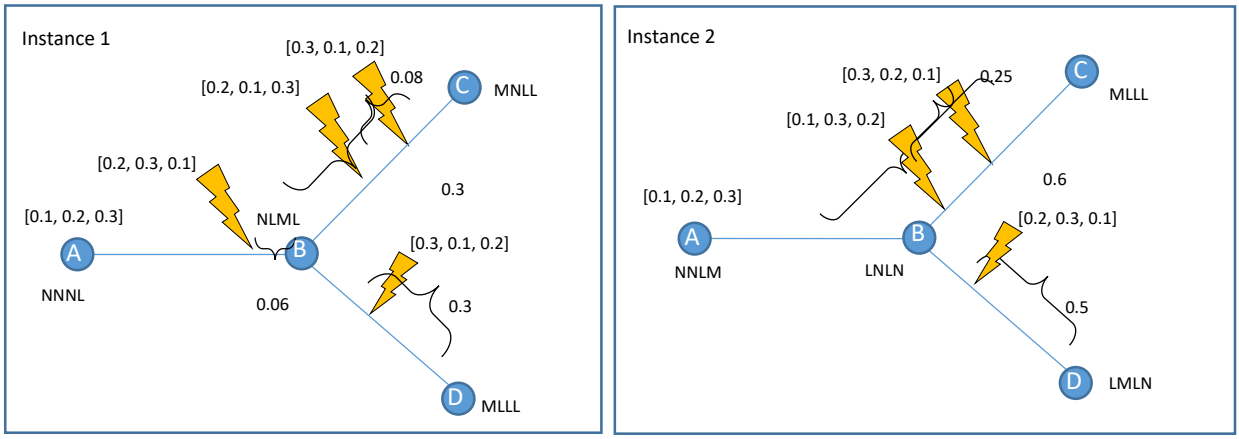
The landscape change times are printed in a FASTA-like format in the file `changetimes.merged.fasta`. Since a branch on a rooted tree can be uniquely identified by the node that it leads up to (its endpoint), we identify each point in the tree by the endpoint of the branch it occurs on and the time remaining until that node. This information is printed in the `changetimes.merged.fasta` file immediately following the "FASTA header" of the endpoint node. The (potentially multiple) landscape change times that occur on the same branch are printed in brackets as a comma-and-space separated vector. Change times for different instances of the simulation (different independent landscape histories) are separated by a semicolon followed by a space.

Similarly to the change time information, the actual fitness vectors are printed in the `fitnesses.merged.fasta` file following the "header" for the endpoint of the branch on which they occur. The fitness vectors are printed as bracketed comma-separated vectors. When multiple landscape changes occur on the same branch, the different landscape vectors are separated by colons (no space); they are listed in the same order as the corresponding change times in the `changetimes.merged.fasta` file, i.e., chronologically (within the given branch). All fitness vectors that occur on the given branch in one simulation instance are placed inside curly braces, with the curly brace-enclosed lists of fitness vectors for different instances separated by a semicolon followed by a space. **The order of simulation instances is the same in all output files.**

The initial landscape is postulated to occur at time 0.0 before the root node.

Example

Consider the course of a hypothetical simulation summarized in the figure below (not drawn to scale). Two simultaneous instances of the simulation (NUM_RUNS 2) are run on a three-letter alphabet "LMN"; in both cases, the initial landscape is given by the fitness vector (0.1, 02, 0.3) (read from a file). The landscape change times are determined stochastically and independently for each branch, and the new landscape is obtained by permuting the previous fitness vector. Landscape changes are marked by a "lightning bolt" with the new fitness vector given next to the "bolt". The distance from the landscape change time to the branch endpoint is given in the diagrams. The four-letter sequence (LENGTH 4) of each of the four (ancestral and extant) species is given next to the corresponding tree node.



The contents of the output files corresponding with this example would be:

allnodes.merged.fasta:

```
>A
NNNLNNLM
>B
NLMLLNLN
>C
MNLLMLLL
>D
MLLLLMLN
```

changetimes.merged.fasta:

```
>A
[0.0]; [0.0];
>B
[0.06]; [];
>C
[0.3, 0.08]; [0.6, 0.25];
>D
[0.3]; [0.5];
```

fitnesses.merged.fasta:

```
>A
{[0.1, 0.2, 0.3]}; {[0.1, 0.2, 0.3]};
>B
{[0.2, 0.3, 0.1]}; {};
>C
{[0.2, 0.1, 0.3]:[0.3, 0.1, 0.2]}; {[0.1, 0.3, 0.2]:[0.3, 0.2, 0.1]};
>D
{[0.3, 0.1, 0.2]}; {[0.2, 0.3, 0.1]};
```

Appendix. The Q matrix and its normalization: technical details

This section follows Yang and Nielsen 2008 "Mutation-Selection Models of Codon Substitution and Their Use to Estimate or Section 2.4.2 of Molecular Evolution: a Statistical Approach *(Yang 2014).*

We use the mutation-selection model but focusing only on selection part of the model by setting all the mutation parameters to 1.

Internally, the fitness vector is first converted to the unnormalized substitution rate matrix Q^{raw}, with $q_{ij,i\neq j}^{raw} = \frac{F_j - F_i}{1-e^{F_i - F_j}}$ proportional to the instantaneous rate of substation of allele i to allele j, and $q_{ii}^{raw} = q_i^{raw} = -\sum_{j:j\neq i} q_{ij,i\neq j}^{raw}$ the negative of the total rate of substitution *from* allele i. Then, if the alleles are distributed according to the stationary distribution π associated with Q^{raw}, the expected substitution rate for the position is $-\sum_i q_i^{raw}\pi_i$. Dividing Q^{raw} by $-\sum_i q_i\,\pi_i$ leads to $-\sum_i q_i^{normalized}\,\pi_i = 1$, i.e., the expected substitution rate equal to 1. This is the behavior chosen by setting `Q_NORMALIZATION` to `constant_rate` (or leaving it to the default option).

Alternatively, we can choose to scale Q^{raw} to get the expected substitution rate to be 1 only for the flat fitness vector such as <1,….,1>, which produces Q^{raw} that consists of 1's off the diagonal and –(|A|-1) on the diagonal, where |A| is the alphabet size. Q^{scaled} is then obtained by dividing Q^{raw} by $\sum_{|A|} \frac{1}{|A|}(|A|-1) = (|A|-1)$, $\frac{1}{|A|}$ being is the stationary probability of any allele (which have equal fitness). This is the behavior selected by setting `Q_NORMALIZATION` to `constant_for_flat`.

If the substitution rate is allowed to vary, the user has the option of scaling the stochastic landscape change rate to it by setting `SCALE_LANDSCAPE_CHANGE_TO_SUBSTITUTION_RATE` to `true`. In this case, the actual parameter λ is obtained by multiplying `LANDSCAPE_CHANGE_PARAMETER` by the substitution rate $-\sum_i q_i \pi_i$. Since this calculation requires the computationally expensive calculation of the stationary vector π following every landscape change, this will increase the computation time.