SLiM Tree: Simulating evolution along a phylogeny with pure population genetics models and realistic fitness functions

Erin Brintnell, Grace Heemeryck, Roman Frolov,
Dave W Anderson, A.P. Jason de Koning

Updated June 2021

1 SLiM-Tree

SLiM-Tree is a software tool to automates pure population genetics simulations over phylogenetic timescales under realistic models of sequence-fitness relationships. The software is highly versatile and is able to handle both Wright-Fisher and non-Wright-Fisher models of evolution, changes to population size and mutation rate for specific branches of a phylogeny and general mutation, selection and recombination rates without assumptions of their strengths.

Contents

2 Executables

2.1 SLiM-Tree

This version of SLiM-Tree can be run from the command line on any computer that can run both Python and SLiM.

To run the most basic version of SLiM-Tree, type the following into the command line: python3 SLiMTree -i ¡your input tree in newick format¿ -T SLiM-Tree

Additional parameters are available in the Parameters section.

2.2 SLiM-Tree-HPC

This version of SLiM-Tree can be run using high-performance computing clusters. It runs separate branches in parallel in order to increase the speed of the program. Aside from that, it is functionally the same as the primary SLiM-Tree.

To run the most basic version of SLiM-Tree-HPC, type the following into the command line: python3 SLiMTree -i ¡your input tree in newick format¿ -T SLiM-Tree-HPC -p ¡partition¿ -t ¡maximum time¿

More information on the parameters, plus additional parameters, are available in the parameters section.

2.3 Requirements

SLiM-Tree requires the installation of Python3 and SLiM (https://messerlab.org/slim/).

Required python packages: sys, argparse, BioPython, matplotlib, random, pandas, numpy, os, json, string, and math.

3 Parameters

All possible parameters for the software are as follows:

-i: tree - the input file in newick format, which specifies which clades to simulate. Required.

-d: optional file - can change parameters for different branches. [elaborate on format?]

-T: tool - specify whether you are using SLiM-Tree or SLiM-Tree-HPC (for parallelization). Required.

-p: partition - the partition which scripts should be written to for SLiM-Tree-HPC. Required for SLiM-Tree-HPC.

-t: time - the maximum amount of time that scripts should be run for before timing out on SLiM-Tree-HPC. Required for SLiM-Tree-HPC.

-n: population size - the size of each population. Default = 100

-v: the mutation rate. Default = 2.5e-6

-g: the length of the genome in codons. Default = 500

-r: recombination rate. Default = 2.5e-8

-b: burn in multiplier - a value which will be multiplied by the population size to determine the length of the burn-in period to establish mutations. Default = 10

-k: sample size - the size of the sample of genomes to be output at the end of the simulation. Default = 10.

-G: number of genes/coding-regions. Default = 1

-C: the ratio of coding to non-coding regions in the genome. Default = 1.0

-c: boolean specifying whether substitutions should be counted. Default = True

-o: boolean specifying whether the generation of the simulation should be output (helps to keep track of longer simulations). Default = True

-B: boolean specifying whether simulations should be backed up. Default = True

-P: boolean specifying whether polymorphisms and fixed states should be printed as output. Default = True

-w: boolean specifying whether a Wright-Fisher model should be used. Default = True

-sr: split ratio. Only in non-Wright-Fisher models, this gives the ratio of the current population that goes into the left branch upon splitting. (Remainder goes into right branch). Default = 0.5

-s: boolean specifying whether the user wants to add in their own sequence. Default = False. This does not apply to PDB files.

-f: fasta file containing the ancestral sequence. Please only input one sequence.

-gb: A genbank file containing information about the ancestral genome. Please only input one sequence.

-hp: A boolean indicating whether the simulation should model haploidy. Default = false

-R: A boolean specifying whether to randomize fitness profiles provided in the fitness data files folder. Default = True. If false, there must be equal fitness profiles to protein sequence length.

4 Additional Features

4.1 Non-Coding Regions

The user can choose to specify regions of the genome that are considered non-coding when running simulations with profile-based fitness. The non-coding regions of the gene are automatically generated, and all mutations that occur in these regions are considered effectively neutral, and won't be considered when calculating fitness.

The number of coding regions can be specified with the parameter -G or –gene_count. There must be at least one coding region. The default number is 1.

The percentage of the gene that is coding is specified using the parameter -C or –coding_ratio. This is the ratio of coding to non-coding regions over the interval (0, 1.0]. For instance, a ratio of 0.7 means roughly 70% of the genome will be coding, and roughly 30% of the genome will be non-coding. Depending on the length of the gene, it may not be exactly divisible, so the ratios may be slightly different from what is entered. The default ratio is 1.0.

Therefore, in order to model multiple genes, both the number and percentage of coding regions must be modified by the user through the parameters.

To view the coding and non-coding regions, the user can open the script in SLiMgui. Dark blue regions are coding, and teal regions are non-coding. They are evenly spaced apart, unless the user inputs their own sequence using a fasta and genbank file.

4.2 Haploidy

SLiMTree can model haploidy within SLiM by toggling the command line parameter -hp or –haploidy to True. When viewing this in SLiMgui, mutations will fix at 50%. This is because in SLiM, each individual has two genomes, genome1 and genome2. When modeling haploidy, only one of these genomes is considered for fitness calculations, and no mutations occur in genome2.

4.3 Non-Wright-Fisher Models

The user can also choose to run SLiM's non-Wright-Fisher model instead of the Wright-Fisher model by setting parameter -w or –wright_fisher_model to False. By default, SLiM-Tree runs under the Wright-Fisher model. For the non-Wright-Fisher models, selection is absolute, and therefore there are some differences in calculating fitness in order to regulate population size. A more thorough description of the differences between Wright-Fisher and non-Wright-Fisher models can be found in the SLiM manual.

4.4 Protein-Based Fitness

UPDATE THIS The default mode of calculating fitness involves using fitness profiles as described previously. An alternative way of calculating fitness involves predicting the probability of a protein folding into a specific structure. The user can provide a contact map (in csv format) and amino acid sequence, or a Protein Data Bank (PDB) file, which will provide information on both the amino acid sequence and contacts. If the user provides both a contact map and PDB file, the contacts will be taken from the contact map rather than the model, though the amino acid sequence will come from the PDB file. If only a contact map is provided, the sequence will be randomly generated.

For the user-provided contact map, it should be a csv file in the form of a matrix that has the same length and height as the provided amino acid sequence. It should be composed of two values, 0 and 1, with 0 indicating there is no contact, and 1 indicating that there is contact, between the two residues at those indices in the file.

When a PDB file is provided, the contacts are calculated using software developed by Ben Rafferty and Zachary Flohr (Network for Computational Nanotechnology, Purdue University, West Lafayette, Indiana) [cite]. Licensing and the README for the original program are available in the folder ContactMapLicensing for SLiM-Tree.

In order to use protein-based fitness, at least two arguments must be modified. The first is -fc, or –fitness_profile_calc, which indicates whether or not fitness profiles should be used to calculate fitness. When this is False, protein-based fitness will be considered instead.

Next is the addition of either a PDB file or a contact map. The path to a PDB file can be added with the argument -pdb or –pdb_file. The path to the contact map can be added with the argument -cm or –contact_map. At least one of these is necessary in order to calculate protein-based fitness, and the program will terminate otherwise.

In order to calculate fitness, the free energy of folding was calculated using an equation from Pollock, Thiltgen, and Goldstein [https://www.pnas.org/content/109/21/E1352 - cite later]. Slight modifications were made to adapt it for use in SLiM. [elaborate once we have those more]

4.5 User-Provided Sequences

Rather than using pseudo-randomly generated sequences by SLiM-Tree, the user can choose to input a nucleotide sequence of their choosing. This can be done using both a fasta file and a genbank file.

In order to input a nucleotide sequence, the user must first set the parameter -s / –user_provided_sequence to True, in order to signal to the program that a sequence is being provided. Then the user can use both -f and -gb (–fasta_file or –genbank_file) in order to input their sequence. It's important to note that only one sequence can be inputted at a time, and both a fasta and genbank fule must be used.

Users can technically input their own amino acid sequences through the protein-based fitness approach using PDB files as seen in section 4.3.

4.6 Different Branch Parameters

The user can also add an additional file which can support different parameters for different branches, such as population size.

Specifications for the file: Branch names must match those in the Newick-formatted tree file. Each branch is indicated using @[branch name]. For instance, for branch A, the starting line would be "@A". Immediately below, any changed parameters should be specified. If they are not specified, default parameters will be used. For instance, to change the population size to 200 and mutation rate to 2.5e-5 for branch A, the file would look like this:

@A -n 200 -v 2.5e-5

Repeat as necessary for all branches. Please note that not all parameters can be changed between branches. For example, the length of the genome, coding regions, and model cannot be changed between populations.

4.7 Feature Compatibility

Some features are compatible with each other, whereas others are not.

Non-Wright-Fisher models are compatible with all other features currently implemented. Be aware that, if you decide to implement your own features, some differences between Wright-Fisher and non-Wright-Fisher models could lead to errors when switching between model types.

Protein-based fitness calculation is not compatible with non-coding regions, as the protein-based fitness approach calculates based on on the amino acid structure of the protein, and not the nucleotides.

Currently, user-provided sequences are not compatible with protein-based fitness. This is because user-provided sequences involve considering non-coding regions. If you want to add in your own sequence for protein-based fitness calculation, use a PDB file in order to get that amino acid sequence.

As well, there is no need to consider inputting the number and ratio of coding to non-coding regions if you're using the user-provided sequences approach, as it automatically considers the coding and non-coding regions from the provided genbank file.

5 Files

Here is a list of all the files in the package for your convenience, if you wish to modify any of them.

In the main folder:

- __main__.py -

- README.md - The README.md file for this program.

- ContactMap.py, GenericMap.py, utils.py - these are files from Rafferty and Flohr [cite] and are used to generate contact maps from PDB files.

- getUserDefinedSequence.py - This is code that takes in the fasta and genbank file inputted by the user and calculates coding regions and the ancestral sequence.

- SLiMTree.py - This is the main code for the program and what is run by the user. It parses all of the parameters and prepares, then runs, the simulation.

- writeSLiM.py - This is the code that writes the SLiM scripts for the program in the traditional SLiM-Tree.

- writeSLiMHPC.py - This is the code that writes the bash and SLiM scripts for this program, to be run on high-performance clusters.

- writeSLiMProtein.py - This extends writeSLiM and overrides a few methods to calculate fitness based on proteins.

In the ContactMapLicensing folder: - LICENSE.txt and README - These are files from the program created by Rafferty and Flohr [cite] which is used by

SLiM-Tree to generate contact maps from PDB files. These are not used by the program, and are provided for licensing purposes.

In the DataPostProcessing folder: - find_branch_lengths.py -

- FindPercentPolymorphic.R -

- perform_analysis_fixation_counts.R -

- perform_statistical_analysis_branch_lengths.R -

- process_data.py

In the ExampleBashScripts folder: - percent_polymorphic.sh -

- run_post_processing.sh -

- run_simulation.sh -

- run_simulations.sh

In the fitnessDataFiles folder:

- 5687A600 - ???

- mj_matrix_eij.csv - A matrix of interresidue contact energies in RT units, used to calculate protein-based fitness. Data from Miyazawa and Jernigan [cite]. Any modification to this file must follow the same indices of amino acids as in that paper and in the SLiM-Tree code.

- slim_codon_nums.csv - A csv file matching SLiM codon numbers with their respective codons and amino acids.

- table_fitness_profiles.csv - A csv file with fitness profiles [cite Mobina?]. This could be replaced with the user's choice of fitness profiles if desired.

- table_stationary_distributions.csv - A csv file containing stationary distributions for fitness profiles. This can also be replaced with the user's choice of fitness profiles if desired.

References