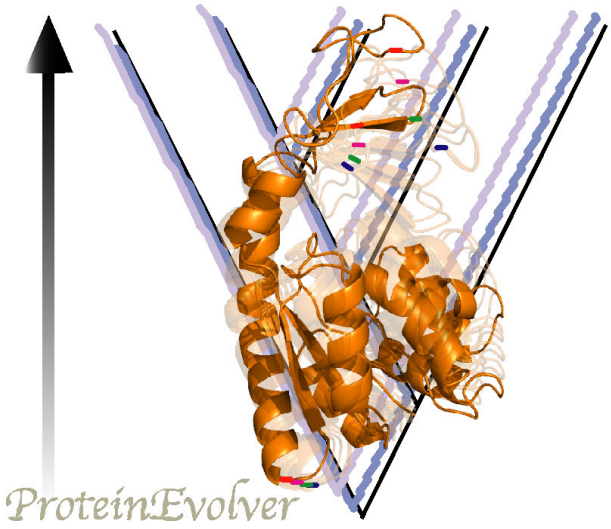
Documentation for **ProteinEvolver**

Phylogenetic simulation of protein evolution accounting for structural constraints

Current version is 2.2.0



Contents

Disclaimer

Credits

The current version of this program was developed at the University of Vigo, Vigo, Spain.

1. Purpose

ProteinEvolver simulates samples of protein sequences evolved along phylogenies, given or also simulated, under diverse substitution models of protein evolution that include structurally constrained substitution (SCS) models.

The present version of *ProteinEvolver*, 2.2.0, provides significant capabilities and improvements respect to the previous version of the program (1.2.0) [1] and, therefore, it is presented as a remarkable version.

The most important new capabilities of this version are the following. First, the interconnected simulation of evolution of protein data and evolutionary history. In population genetics, all the currently available simulators of molecular evolution evolve molecular sequences upon a given evolutionary history, somehow assuming neutral evolution. However, this simulation is highly unrealistic because the evolutionary history is commonly affected by the molecular data. For example, a molecular variant with high fitness is likely to have more descendants that a molecular variant with low fitness. This version of the program considers the fitness of molecular variants to simulate not only their molecular evolution but also their evolutionary history forward in time. This capability was implemented combining selection from the protein folding stability with a birth-death population genetics model [2, 3]. Thus, protein sequences can be evolved under a SCS model and their fitness (stability) is used in the birth-death process. Thus, a molecular variant with high fitness produces a high birth rate (and low death rate) that can lead to many descendants, while a molecular variant with low fitness produces a low birth rate (and high death rate) that can lead to few or none (extinction) descendants. This simulation includes a number of options such as:

- Fitness based on the amount of stability of the studied variant (at any node) or, based on the similarity of stability of the studied variant (at any node) with the stability of an observed protein.
- The birth-death simulation process can finish specifying one of the following options: (a) a given number of sample nodes is reached. (b) a given number of sample nodes that belong to the present time (extinctions are ignored) is reached. (c) a given time from the root to the present (extinctions are ignored) is reached.
- Extinct nodes derived from death events can be kept or pruned from the resulting phylogenetic tree.

- The simulated birth-death tree can be evaluated in terms of tree balance with the Colless index [4, 5].

Additional new capabilities are the following:

- Possible variation of the general site-specific substitution rate under any model (including the implemented SCS models) according to a user-specified input file.
- User-specified site-specific matrices of relative rates of change among amino acids and amino acid frequencies at the equilibrium.
- The program already implements a variety of empirical substitution models of protein evolution (details later) and, in this version, it also included the FLU empirical substitution model [6]. This incorporation was evaluated with *ProtTest3* [7], where the model was correctly identified in data simulated under the cited model.

This new version of the program maintains all the options of the previous version (i.e., user-specified phylogenetic tree and simulation of the phylogenetic tree through the coalescent without or with recombination (including recombination hotspots and coldspots), migration, demographics and longitudinal sampling (among other options). However, all the input files are designed in a new format to more easily allow the specification of a simulation process among the large variety of capabilities.

What is exactly implemented in ProteinEvolver2.2.0?

ProteinEvolver simulates protein sequences along an evolutionary history (i.e., phylogenetic tree) that can be given or also simulated by *ProteinEvolver*. The phylogenetic history can be user-specified, simulated with the coalescent or simulated with the birth-death population process. Next, the protein evolution can be simulated with any traditional empirical substitution models of protein evolution or with a structurally constrained substitution (SCS) model.

As a relevant new capability of *ProteinEvolver2.2.0*, the evolutionary history can be simulated with the birth-death model considering the fitness (in terms of folding stability) of protein sequences simulated at every node. In particular, as indicated previously, a protein sequence with high fitness could produce a high birth rate (and low death rate) that can lead to many descendants, while a protein sequence with low fitness could produce a low birth rate (and high death rate) that can lead to few or none (extinction) descendants.

The implemented SCS models consider contact matrices, configurational entropies per residue in unfolded and misfolded proteins, configurational entropy offset (misfolded) and energy functions [8, 9], as implemented in the previous version of *ProteinEvolver* [1]. The user has to specify a PDB file (which can be downloaded from the Protein Data Bank, http://www.rcsb.org/PDB/home/home.do) and a sequence, both are assigned to the root of the phylogeny.

Indeed, the program implements a variety of empirical substitution models of protein evolution such as *Blosum62* [10], *CpRev* [11], *Dayhoff* [12], *DayhoffDCMUT* [13], *FLU* [6], *HIVb* [14], *HIVw* [14], *JTT* [15], *JonesDCMUT* [13], *LG* [16], *Mtart* [17], *Mtmam* [18], *Mtrev24* [19], *RtRev* [20], *VT* [21], *WAG* [22] or any user-specified matrix for all the sites or for every site (can differ among sites). Indeed, it includes heterogeneous substitution rates among sites by the gamma distribution (+G) and the proportion of invariable sites (+I), but also the user can modify the substitution rate at every site, for example allowing conservation at particular sites (for example, catalytic sites), at any

SCS or empirical model. The molecular evolution is simulated forward in time along a phylogeny.

Regarding the phylogeny, the user can either specify a particular phylogenetic tree or, simulate birth-death history [2, 3] or simulate a coalescent history [23, 24]. In the case of the birth-death history, which is a new capability, the user can either specify the birth and death rates or indicate that the birth and death rates are based on protein fitness, many other parameters (i.e., criterion to finish the birth-death process, presentation of the produced evolutionary history, etc) can also be specified. As in the previous version, it includes the coalescent with recombination [23] (which can be homogeneous or heterogeneous along the sequence [following, 25]), variable population size (by a growth rate and demographic periods), a variety of migration models (such as island [26], stepping-stone [27] and continent-island [28]) with temporal variation of migration rates and convergence of demes or subpopulations, the simulation of haploid or diploid data, and longitudinal sampling [see, 29].

As in previous versions, the output allows multiple options. Alignments can be printed in *phylip*, *fasta* and *nexus* formats. The ancestral sequence (MRCA or GMRCA, most recent common ancestor and grand most recent common ancestor, respectively [see, 30, 31]) can also be printed. Energies for the native structural protein can be printed as a function of the temperature and energies for the simulated proteins can be also printed to study how the incorporation of substitution events influences the structural protein stability. When birth-death or coalescent histories are simulated, the simulated tree or ancestral recombination graph (ARG) [32] can be also printed. We recommend exploring the attached folder "example_input_files", which contains a variety of examples of evolutionary scenarios that can be simulated with *ProteinEvolver*.

2. Executables and compilation

Executable files are provided for Linux Debian and MacOS X (Intel and G4 processors), and a Makefile is provided for compilation in any OS with a C compiler. This makefile can be optimized for different users, for example using the optimization option -fast instead of –O3, for Mac processors. To compile the program, move to the "*src*" folder and type (it may take a few seconds): *make all*
It should print something like:
Building ProteinEvolver version 2.2.0
gcc -c -O3 -Wall ProteinEvolver2.2.0.c
gcc -lm -O3 -Wall -o ProteinEvolver2.2.0 ProteinEvolver2.2.0.o
Finished compiling.
A second makefile "Makefile_MPI" is provided to compile a MPI version (which could be convenient for experiments that involve an extensive number of simulations [e.g., 33, 34]). This Makefile might need some modifications for particular OS. To compile the program type: *make -f Makefile_MPI.* MPI libraries have to be installed in the computer for running *ProteinEvolver* in parallel. The minimum number of processors is two. An example of execution for 3 processors is: *mpirun -np 3 ProteinEvolver2.2.0*

3. ProteinEvolver Usage

The input of the program consists of a series of parameters (Table 1) that can be specified in input text files, the main one is named "parameters" and all these input text files should be located at the directory of the executable file. Optionally, some of these input parameter values (only those shown in the "parameters" file) can be specified in the command line but still I recommend using the input files because there one can specify all the input parameters. These input parameters are required to define the evolutionary scenario under which the simulations are performed and also some parameters include number of simulations under a given evolutionary scenario (hereafter replicates), format of output files, seed and printing options that control the amount of information that is sent to the screen.

In addition to the "parameters" file, other input files can be required for running (further details in section 3):

- Parameters to simulate a birth-death evolutionary history (i.e., BirthDeath.in).
- Parameters to simulate a coalescent evolutionary history (i.e., Coalescent.in). It can require additional input files such as settings to simulate recombination with recombination hotspots (i.e., UserHetRec.in).
- User-specified phylogenetic tree (i.e., treefile.in).
- Variation of the global substitution rate among sites according to a user-specified distribution (i.e., HetRatesVector.in).
- User-specified empirical substitution model of protein evolution for all the protein sites (i.e., UserEAAM).
- User-specified empirical substitution model of protein evolution for every protein site (i.e., UserEAAMsites).
- Parameters for structurally constrained substitution (SCS) models of protein evolution (i.e., Pop_evol.in). This requires additional input files such as a Protein Data Bank (PDB) file (i.e., 1TRE.pdb) and amino acid contacts matrix (i.e., structures.in).
- A user-specified sequence that is assigned to the root of the given or simulated phylogeny (i.e., seqRoot.in).

Examples for all these input files are provided in the "examples/example_input_files" directory.

3.1. Command line

In Mac systems, the command line is provided by the Terminal, while in Windows, it is provided by the windows console or command prompt. If the user specifies any argument in the command line, *ProteinEvolver* will use the values specified for those parameters, and default values for the parameters not included in the command line. It is highly recommended check the information shown in the screen to be sure that the simulations were parameterized as intended. **Importantly, as indicated previously, for simplicity, only the parameters included in the "parameters" file can be specified by arguments (all the parameters included in other input files cannot be specified as arguments in the command line) and so I recommend using the input files (section 3.2).** Some examples of simulation specified in the command line are the following:

Protein sequences simulated under a SCS substitution model combined with a birth-death evolutionary history (importantly, notice that several input files must be placed in the working directory, for this execution use files provided in the example 1),
./ProteinEvolver2.2.0 -n2 -lBirthDeath.in -@JTT -zPop_evol.in -xseqRoot.in -bsequences -c2 1 0 -$ -jtrees -#4 -y2

DNA data simulated under the traditional JC substitution model +G +I and along a user-specified tree in the input file *treefile*,
./ProteinEvolver2.2.0 -n2 -ptreefile.in -v1 0.5 -a0.7 -i0.5

Protein sequences simulated under an empirical substitution model (FLU) upon a birth-death evolutionary history with user-specified birth-death parameters (importantly, notice that several input files must be placed in the working directory, for this execution use files provided in the example 7),
./ProteinEvolver2.2.0 -n2 -lBirthDeath.in -@FLU -bsequences -c2 1 0 -$ -jtrees -#4 -y2

I strongly recommend the use of the parameters file (instead of the command line, see next section) in order to consider all the parameters implemented in the program, avoid text errors and have a better checking of the input settings.

3.2. The main input file. The "parameters" file

If no argument is specified, the program will only read the text file "*parameters"* which should be placed at the same directory of the executable. If no arguments are specified in the command line, and there is no a "*parameters*" file, the program will stop and throw an error.
In the input files, the simulator ignores anything within brackets.
Then, only type:
./ProteinEvolver2.2.0

Most of the implemented evolutionary scenarios require no only the "*parameters*" file but also other input files. A total of 10 examples of input files (including the "*parameters*" file) to simulate protein sequences under a variety of evolutionary scenarios are provided (and can be modified to build new similar evolutionary scenarios). In particular, in the directory "examples/example_input_files":

- **All_inputFiles** -> Examples of all the input files that the program can use (not all of them are required for a particular simulation)
- **Ex1.BirthDeathHistoryEndBySampleTips_SelectionFromObsStability_combined_SCSmodel_outgroup** -> Example that simulates protein sequences under a SCS model combined with a birth-death model where the fitness is based on the similarity of the folding stability of the simulated protein sequence and the real protein sequence (PDB file). Thus, the birth-death evolutionary history (which is an output in Newick format) is driven by the fitness of the protein sequence at every internal node. The birth-death model finishes when reaching a given number of nodes at the present (named as tip nodes), that in this example is 15. An outgroup is also simulated with a given branch length.
- **Ex2.BirthDeathHistoryEndByTime_SelectionFromObsStability_PhyloExtinctionsPruned_combined_SCSmodel** -> Similar to the previous example but: the birth-death evolutionary process finishes when a user-specified evolutionary time is reached (in the sample, 14300.6; notice that the time unit is 2N, where N

is the effective population size). The extinction nodes (derived from death events) are pruned from the simulated evolutionary history. No outgroup is simulated.

- **Ex3.BirthDeathHistoryEndBySampleTips_SelectionFromHighStability_combined_SCSmodel ->** Similar to example 1 but: the fitness is based on high stability, the higher is the stability the higher is the fitness of the protein variant. The outgroup is not simulated.
- **Ex4.BirthDeathHistoryEndByTime_SelectionFromHighStability_combined_SCSmodel** -> Similar to example 2 but: the fitness is based on high stability, the higher is the stability the higher is the fitness of the protein variant. Extinct nodes are not removed.
- **Ex5.BirthDeathHistoryEndBySampleAll_SelectionFromHighStability_PhyloExtinctionsPruned_combined_SCSmodel_outgroup ->** Similar to example 3 but: the birth-death model finishes when reaching a given number of sample nodes at the present or at any time, that in this example is 15. The extinction nodes (derived from death events) are pruned from the simulated evolutionary history. An outgroup is simulated.
- **Ex6.BirthDeathHistory_EmpiricalModelGlobalGiven_SiteSubsRateVariationGiven** -> First, a birth-death evolutionary history is simulated according to user-specified birth and death rates. Next, protein evolution is simulated upon that evolutionary history under a user-specified matrix of relative rates of change among amino acids that is the same for all the protein sites, but also allows the global substitution rate to vary among protein sites according to a user-specified distribution.
- **Ex7.BirthDeathHistory_EmpiricalModel+G+I_outgroup** -> First, a birth-death evolutionary history is simulated according to user-specified birth and death rates. Next, protein evolution is simulated upon that evolutionary history under the LG empirical substitution model of protein evolution and also allows the global substitution rate to vary among protein sites according to the proportion of invariable sites (+I) and the gamma distribution (+G). An outgroup is simulated.
- **Ex8.TreeGiven_EmpiricalModelGlobalGiven** -> First, a phylogenetic tree is user-specified. Next, protein evolution is simulated upon that phylogenetic tree under a user-specified matrix of relative rates of change among amino acids that is the same for all the protein sites.
- **Ex9.CoalescentWithRecombinationHistory_EmpiricalModelPerSiteGiven_outgroup** -> First, an evolutionary history is simulated with the coalescent with recombination. Next, protein evolution is simulated upon that evolutionary history under a user-specified site-specific matrix of relative rates of change among amino acids (the relative rates of change among amino acids vary among protein sites). An outgroup is simulated.
- **Ex10.TreeGiven_EmpiricalModelPerSiteGiven_SiteSubsRateVariationGiven** -> First, a phylogenetic tree is user-specified. Next, protein evolution is simulated upon that evolutionary history under a user-specified site-specific matrix of relative rates of change among amino acids (the relative rates of change among amino acids vary among protein sites) and, the global substitution rate vary among protein sites according to a user-specified distribution.
- **Example1_forDiverseProteinFamilies** -> Example 1 applied to simulate protein evolution according to settings from 10 different protein families (i.e.,

particular settings such as sequence length, sequence of the root node, PDB file, among others).

Possible arguments for the parameters file are the following (# means number, *NAME* means a word):

General settings
-n# : Number of replicates
The number of samples to be generated. Each sample is an independent realization of the evolutionary process. This specification is mandatory. Example: *-n10*

Evolutionary history settings
The user has to choose the type of evolutionary history. The evolutionary history can be either simulated with a birth-death process, simulated with the coalescent or specified by the user as a phylogenetic tree. Only one can be selected. In any of case, *NAME* refers to the name of another input file that must be included in the working directory.
-lNAME : Birth-death process. Example: -lBirthDeath.in
-sNAME : Coalescent process. Example: -sCoalescent.in
-pNAME : Specified as a phylogenetic tree. Example: -ptreefile.in
Example in the parameters file where the birth-death process is specified,
*[** birth-death simulation **] [input file with general settings] lBirthDeath.in*
*[** coalescent simulation **] [input file with general settings] [sCoalescent.in]*
*[** input tree/s file **] [input tree/s file] [ptreefile.in]*

Substitution model settings
In addition to the simulation of protein sequences, the program can also simulate DNA sequences under traditional DNA substitution models.

Substitution models of DNA evolution
-f4 # # # #: Nucleotide frequencies
The nucleotide frequencies A C G T are specified in this order. The first number must be 4. By default all frequencies are equal. Example: *-f4 0.4 0.2 0.1 0.3.*

-v1 #, -v6 # # # # # #, -v12 # # # # # # # # # # # #: Relative substitution rates
The first number specifies the total number of relative substitution rates that can be 1 (for transition/transversion rate ratio), 6 (for relative symmetric substitution rates) or 12 (relative asymmetric substitution rates).

If the first number = 1, the second number is the transition/transversion rate ratio. When set to 0.5, the probability of transitions and transversions are the same, like in models "*JC*" [35] or "*F81*" [36] models. Example: *-v1 2.1*

If the first number = 6, the following 6 numbers are the relative symmetric substitution rates A↔C, A↔G, A↔T, C↔G, C↔T and G↔T. Example: *-v6 1.0 5.0 1.0 1.0 5.0 1.0.*

If the first number = 12, the following 12 numbers are the relative asymmetric substitution rates AC, CA, AG, GA, AT, TA, CG, GC, CT, TC, GT and TG. Note that some calculations under this option can be problematic (complex eigenvalues/vectors) if rates are too asymmetric, especially with codon models. Example: *-v12 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 1.0 1.0*

-mNAME : Coding DNA simulation accounting for structural constraints

This argument should be followed by the name of the file with settings for the SCS models (see section 3.3). When this argument is specified the program will simulate coding DNA data (which of course can be translated to amino acid data). In particular, under this model the mutations and substitutions (accepted mutations) occur at DNA level but their energies are tested at amino acid level taking into account the stability of the protein structure (see section 4.2). Importantly, do not activate this argument and the argument "*-z*" (which directly simulates amino acid data considering the stability of the mutations in the protein structure) at the same time. Example: *-mPop_evol.in*

Substitution models of protein evolution

-@NAME : empirical amino acid substitution model

The implemented empirical amino acid models of evolution are the following: *Blosum62* [10], *CpRev* [11], *Dayhoff* [12], *DayhoffDCMUT* [13], *FLU* [6], *HIVb* [14], *HIVw* [14], *JTT* [15], *JonesDCMUT* [13], *LG* [16], *Mtart* [17], *Mtmam* [18], *Mtrev24* [19], *RtRev* [20], *VT* [21], *WAG* [22]; in this case just specify the name of the empirical amino acid model. Example: -@JTT.

In addition, it is possible to specify a user-defined empirical amino acid model by an input file (see details later). There are two options:

- The provided empirical substitution model is the same to all the protein sites "UserEAAM (all sites)". Example: -@UserEAAM.
- The provided empirical substitution model can vary among protein sites "UserEAAMsites (per site)". Example: -@ UserEAAMsites.

In any case the user has to specify the name of the input file with the empirical amino acid model. Example: -@UserEAAM. Example: -@UserEAAMsites.

In these input files, the order of amino acids (to specify the relative rates of change and the frequencies) must follow the specified in the order: A R N D C Q E G H I L K M F P S T W Y V. See the examples distributed with the package to visualize the required format of these input files.

-zNAME : Protein simulation accounting for structural constraints

This argument should be followed by the name of the file with settings for the SCS models (see later). Importantly, when this argument is specified the program will simulate protein data. In particular, under this model the mutations and substitutions (accepted mutations) occur at the amino acid level and their energies take into account the stability of the protein structure. Importantly, do not activate this argument and the argument "*-m*" at the same time. Example: *-zPop_evol.in*

Optional settings for any substitution model of evolution

-a# : alpha shape of the gamma distribution

A gamma distribution (*+G*) can simulate substitution rate variation among sites [37]. Alpha is the shape of this distribution. Smaller alphas imply stronger rate variation. Example: -a0.7.

-i# : proportion of invariable sites

A proportion of sites can be set to be invariable (*+I*). Example: *-i0.3*

-_NAME : factor for variable substitution rate among sites

Every site has a substitution rate indicated by the user in the birth-death, coalescent or input tree (here as branch length) files. With this option, the user can multiply that

substitution rate by a factor between 0 and 1 to allow variation of the substitution rate among sites, especially to introduce conservation of particular sites (i.e., conserved due to selection for maintaining a molecular function). Thus, for a given site: If this factor is 1, its substitution rate is the same than the original substitution rate. If this factor is 0, the substitution rate is 0. If this factor is between 0 and 1, its substitution rate is the original substitution rate multiplied by the factor, so the substitution rate will be smaller than the original substitution rate. After this argument the user should specify the name of an input file, which contains a vector with the values of the factor for every site (see details later). Example: -_HetRatesVector

-xNAME : Sequence assigned to the root node
The user can specify sequence in a text file (see later) that is assigned to the root node of the evolutionary history. This file must contain only a DNA or amino acid sequence (depending on the substitution model applied), and the length has to be equal to the number of sites specified in arguments of other input files (in particular, specification of the evolutionary history through the birth-death, coalescent or input tree files). By default (if not specified), the root sequence is simulated from the nucleotide or amino acid frequencies. Provide a sequence for the root node is particularly recommended for the SCS models because a random sequence could be highly unstable and present a low fitness. Example: *-xseqRoot.in*

Output settings
-bNAME : print sequences
When this argument is activated, aligned sequences are printed to the specified text file in the *Results* folder. Example: *-bsequences*

-c# # #: format for printing sequences
This option specifies the format of the output alignments. The first argument indicates *Phylip* sequential, *Fasta* and *Nexus* formats (1-3, respectively). The second argument indicates that alignments for each replicate are printed into a single file (0) or different files (1). The third argument prints the sequence of all internal nodes (1). Example: - c1 1 0 (which means: phylip format, a file for each replicate, sequences corresponding to internal nodes are not printed).

-$: print sumMRCA/GMRCA
This option prints the ancestral sumMRCA/GMRCA sequences in output files. These output files will be incorporated to the *Results* folder. Example: *-$*

-jNAME : print genealogies
When this option is specified, the simulated evolutionary histories are printed, in *Newick* format, to the specified text file, in the *Results* folder. Example: *-jtrees*

-kNAME : print times
When this argument is specified the times of the internal nodes of the genealogy are printed to the specified text file, in the *Results* folder. This option will slow down the simulations. Example: *-ktimes*

*-*NAME : print ARGs*

In case the evolutionary history is based on the coalescent with recombination, when this argument is specified the ancestral recombination graph (ARG) will be printed to the specified text file in the *Results* folder. Example: -**NetworkFile*

-dNAME : print breakpoints
In case the evolutionary history is based on the coalescent with recombination, when this argument is specified breakpoint positions are printed into the specified text file, in the *Results* folder. This option only works for coalescent simulations. Example: *-dbreakpoints*

Other settings
-## : Seed
Seed for the random number generator. If no seed is specified, the computer clock will be used as seed. When a seed is fixed the process can be reproduced if same settings are specified. Example: *-#386*

-y# : noisy level
This option controls the level of information that in printed on the screen when running the program.

0 : Does not print run information, only the simulation progress.
1 : Run settings and run information summarizing the simulations.
2 : Calculation status, run settings for each replicate.
3 : Information about the evolutionary history and molecular evolution.
4 : All the details.

Note that higher levels of noisy will slow down the simulations. The default level is 1 Example: *-y1*

3.3. Birth-Death input file

With this input file a birth-death evolutionary history is simulated. Importantly, this input file must be indicated in the parameters file with the option *-l* followed by the name of the input file with the settings to simulate the birth-death process (i.e., *-lBirthDeath.in*). Several examples of the input file "*BirthDeath.in*" are provided in the examples directory.

l# #: Birth and death rates
The program includes several specifications of the birth and death rates:

- Specify directly the birth and death rates, first the birth rate and next the death rate. Example: *l0.3 0.2*
- Indicate that the birth and death rates should be obtained from the fitness of the protein at every node, where the fitness is based on similarity of the folding stability of the modeled protein and the observed (i.e., real) protein (i.e., file of the Protein Data Bank, PDB). Thus, an evaluated protein with stability similar to the stability of the observed protein will have a high fitness and thus a high birth rate and low death rate. In this case specify -1. Example: *l-1*
- Indicate that the birth and death rates should be obtained from the fitness of the protein at every node, where the fitness is based on the folding stability of the modeled protein. Thus, an evaluated protein with high stability will

have a high fitness and thus a high birth rate and low death rate. In this case specify -2. Example: *l-2*

+# #: Criterion for ending the birth-death process
The program includes several specifications for ending the simulation of the birth-death evolutionary history.
One is based on the sample size (specify as +1 # #):

- The birth-death process finishes when a given sample size (including in the sample nodes that were extinct but it can be smaller if extinction of all lineages) is reached. In this case specify 1. Example (sample size is 15): *+1 1 15*
- The previous option can be more strict, the birth-death process finishes when a given sample size (including in the sample nodes that were extinct) is reached. In this case specify 2. Example (sample size is 15): *+1 2 15*
- The birth-death process finishes when a given sample size (excluding in the sample nodes that were extinct, so only the tip "present" nodes) is reached. In this case specify 3. Example (sample size is 10): *+1 3 10*

Another one is based on the time of the birth-death simulation process. The birth-death process finishes when a specified time is reached. Importantly, the time of the birth-death simulation process is based on 2N generations if the effective population size (N) is specified. Specify as +2 #. Example, *+2 7000.24*

>#: Prune extinct nodes and lineages of the birth-death evolutionary history
If yes (>2) the program removes extinct nodes and lineages of the birth-death evolutionary history. If no (>1), extinct nodes and lineages are kept. Example, *>1*

o# : outgroup and its branch length
If this option is specified the program simulates an outgroup sequence that evolves independently from the sample, along a branch of the specified length. By default the outgroup is not simulated. Example: *o0.001*

u# : Substitution rate
Substitution rate per site and per generation. Example: *u9.1e-2*

e# #: Effective population size; Haploid / Diploid
The effective size (*N*) of the population from which the sample is obtained. The second number means that the data set can be simulated as haploid (1) or diploid (2). Example: *e1000 2*

s#: Number of sites (length of the simulated sequences)
The total sequence length (in nucleotides or amino acids). Example: *s255*

3.4. Coalescent input file

With this input file a coalescent evolutionary history is simulated. Importantly, this input file must be indicated in the parameters file with the option *-s* followed by the name of the input file with the settings to simulate the coalescent process (i.e., *-sCoalescent.in*). Examples of the input file "*Coalescent.in*" are provided in the examples directory. The coalescent simulation implemented in this version of the

program is that one presented in the previous version [1], thus this section was not changed.

s# #: Sample size; Number of sites
The number of sequences to be generated for each sample and the total sequence length (in nucleotides or amino acids). Example: *s6 255*

e# #: Effective population size; Haploid / Diploid
The effective size (*N*) of the population from which the sample was theoretically drawn. If there are several demes, this argument is the effective population size for each deme.
The second number means that the data set can be simulated as haploid (1) or diploid (2). Example: *e1000 2*

=# : Tip dates
The time of the tips can be different with this option. For example, 4 samples: 1995:sequence 1; 2003: sequences 4 and 6; 1997: sequences 2 and 3; 2001: sequences 7 and 8. This option does not work if there is any convergence of demes at younger times. Example (of above): *=4 1995 1 1 2003 4 6 1997 2 3 2001 7 8*

/# : *Generation time*
The time per generation. Example: */300*

g0 # or g1 # (# # #) : Demographics settings. Exponential growth rate or Demographic periods
The first number specifies the model, exponential growth rate (0) or demographic periods (1). These parameters are looking back in time, so it is not a good idea to specify a negative growth rate for the last period, as the coalescent time could become infinite in the past.
Rate of exponential growth per individual per generation, after "0" the growth rate must be specified. Example: *g0 1e-5* (= *g0 0.00001*)
Demographic periods, after "1" the user has to specify the number of periods (from the present to the past) and *N* during those periods. The first number here specifies the number of periods. For each period should be three consecutive numbers indicating the size *N* at the beginning and at the end of the period, and the duration of the period in generations. Example: *g1 3 1000 1250 1000 1300 1550 2000 1560 1000 3000*
The exponential growth rate during the period (positive or negative) will be deduced from the specified *N* at the beginning and at the end of the period. The growth rate derived for the last period will continue into the indefinite past. This implementation is borrowed from [38]. This option is incompatible with the exponential growth rate option (*g*). Again, these parameters are looking back in time, so it is not a good idea to specify a negative growth rate for the last period, as the coalescent time could become infinite in the past.

q# # (#): Migration model and population structure
The first number specifies the migration model (island model=1, stepping-stone model=2, continent-island model=3). The second number specifies the total number of demes or subpopulations sampled. The next *n* numbers specify the number of individuals (or sequences) per deme (note that the specified sample size (*s*) must be equal to the sum of these). For the island-continent model, deme #1 will be the

continent while the other demes will be islands (see details in section 4.3.2). Example: *q2 2 3 3* (a stepping-stone model, two demes with three samples each).

t# (#)(#): Migration rate
This parameter introduces the migration rate, which can be constant or variable with time according to temporal periods. The first number specifies the number of temporal periods, then:

For only 1 period, the second number is the migration rate (constant). Example: *t1 0.001* (only 1 period with migration rate = 0.001).

For more than 1 period, the second number/s are the time/s for the beginning of a new migration rate and the third/s numbers are the corresponding migration rate/s for each period. Example: *t2 100 0.001 0.005* (2 periods, the first period occurs from t = 0 to t = 100 with a migration rate = 0.001, the second period occurs from t = 100 to the end of the simulation with a migration rate = 0.005). Example: *t3 100 800 0.002 0.001 0.003* (3 periods: from t = 0 to t = 100 with migration rate = 0.002, from t = 100 to t = 800 with migration rate = 0.001, from t = 800 to the end of the simulation with a migration rate = 0.003).

%# (# # #) : Events of convergence of demes
The first number specifies the total number of convergent events. For each convergence event should be three consecutive numbers. The first number and the second number are the numbers of the demes to converge. The third number is the time to that convergence. With this option the user can build the demes evolutionary tree but it is only available when the migration model is activated (despite the migration rate could be zero). Examples: %1 1 2 2000 (for 2 initial demes (q2), convergence of deme 1 with deme 2 at time 2000 to create a new deme 3). *%3 1 2 400 3 4 1900 5 6 2000* (for 4 initial demes (q4) convergence of deme 1 with deme 2 at time 400 to create a new deme 5, convergence of deme 3 with deme 4 to create a new deme 6 at time 1900, convergence of deme 5 with deme 6 at time 2000 to create a final new deme 7).

r# : Homogeneous recombination rate
The homogeneous recombination rate per site and per generation. All sites will share this same rate. This is also the background recombination when simulating recombination hotspots. Example: *r2.1e-6* (= *r0.0000021*)

w# : Fixed number of events of recombination
This option fixes the number of events of recombination per replicate, so every sample will have the same number of recombination events. What it does is to filter out replicates with a different number of recombination events, so it will take more time. We recommend to use this option with careful, with sense according to the recombination rate (e.g. do not fix the number of events of recombination = 0 when the recombination rate is very high, or do not fix the number to a high value when the recombination rate is very low or nil), otherwise the execution may never ends until the program crashes. Example: *w2*

hNAME : Recombination hotspots
This option activates the simulation of recombination hotspots. Parameters for recombination hotspots must be introduced in the file here indicated (see later). Example: *hUserHetRec.in*

u# : Substitution rate
Substitution rate per site and per generation. Example: *u0.0015*

o# : outgroup and its branch length
If this option is specified the program simulates an outgroup sequence that evolves independently from the sample, along a branch of the specified length. By default the outgroup is not simulated. Example: *o0.001*

3.5. User-specified input tree file

The user can specify an input phylogenetic tree upon which molecular evolution can be simulated. Importantly, this input file must be indicated in the parameters file with the option *-p* followed by the name of the input file with the phylogenetic tree (i.e., *-ptreefile.in*). Examples of the input file "*treefile.in*" are provided in the examples directory.
Note that when this option is activated the birth-death and coalescent simulations are not performed (in that case, the user-specified tree is applied).
This option is convenient when the user already has a phylogenetic tree (e.g., inferred from real data) or to avoid the specification of parameters required to simulated birth-death or coalescent evolutionary processes. This procedure is similar to other software such as *SeqGen* [39] or *Evolver* [40] (but note that models implemented in these programs do not consider the evolution of protein sequences under structural constraints).
The input file consists of a text line with a range of sites and a tree in Newick format. The range specifies the first and last site (nucleotide or amino acid position) where the following tree should work. Importantly, all the sites should be covered by a phylogenetic tree, so it always should start at 1 and finish at the total number of sites. The file does not allow empty lines. The specified phylogenetic tree should be rooted to provide a temporal/directional perspective required to simulate the molecular evolution.
An example is shown below,

```
1 255 ((taxonA:0.1,taxonB:0.1):0.4,(taxonC:0.1,taxonD:0.1):0.5,taxonE:1.0);
```

Note that by this procedure the total number of sites, number of taxa and name of taxa are specified. This example considers a total of 255 sites and 5 taxa (taxonA, taxonB, taxonC, taxonD, taxonE).

3.6. Input files for the structurally constrained substitution models

The main input file to specify structurally constrained substitution (SCS) models must be specified in the parameters file by the argument "*-m*" (for the simulation of coding DNA data) or "*-z*" (for the simulation of protein data). Examples of the main input file for these models can be found in the "examples" folder, with name "*Pop_evol.in*". Theory is described in posterior sections.
Importantly, this simulation requires that the root sequence must codify the amino acid sequence of the PDB file (for coding data simulation) or be equal to the amino acid sequence of the PDB file (for protein simulation), see details in the theory section. **As a consequence, for these models it is highly recommended to fix the root sequence by**

an input file (argument "*-x*" in the parameters file) because a root sequence computed using the nucleotide or amino acid frequencies could not satisfy such a condition. Possible arguments for the SCS models are the following (# means number, *NAME* means a word):

Structural settings

PDB= NAME: File from the Protein Data Bank

A file from the Protein Data Bank (http://www.rcsb.org/PDB/home/home.do) (PDB) must be specified. Example: *PDB= 1TRE.PDB*

CHAIN= NAME: Chain of the PDB file

The particular chain of the PDB file to be considered must be specified. Example: *CHAIN= A*

FILE_STR= NAME: List of contact matrices

A file with list of contact matrices must be specified. The folder with example input files already contains a file "*structures.in*" with a huge list of contact matrices downloaded from the PDB and which can be applied to compute energies from any PDB protein structure. However, the authors could provide other files with contact matrices (considering particular user-specified details) upon request. Example: *FILE_STR= structures.in*

TEMP= #: Temperature

The temperature used to compute protein energies must be specified (see section 4.2). We recommend a range between 1.25 and 2.0. Example: *TEMP= 1.5*

S0= #: Configurational entropy per residue (unfolded)

The configurational entropy per residue for the unfolded protein must be specified. We recommend a range between 0.025 and 0.075. Example: *S0= 0.05*

SC1= #: Configurational entropy per residue (misfolded)

The configurational entropy per residue for the misfolded protein must be specified (see section 4.2). We recommend a range between 0.025 and 0.075. Example: *SC1= 0.05*

SC0= #: Configurational entropy offset (misfolded)

The configurational entropy offset for the misfolded protein must be specified (see section 4.2). By default this it is 0. Example: *SC1= 0.0*

REM3= #: Third cumulant in REM calculation

The third cumulant in REM calculation (to compute the structural protein energy) must be specified (see section 4.2). By default this it is 0. Example: *REM3= 0*

*NEUTRAL= #: If 1, Neutral landscape (**neutral SCS model**), otherwise population size dependent selection (**fitness SCS model**)*

If this parameter is set to 1, the neutral substitution models of structural protein stability will be applied (see later). Note that this model does not use the population size (next parameter), by default we recommend this neutral model. Example: *NEUTRAL= 1*

NPOP= #: Population size for protein structures

The population size to simulate molecular evolution under fitness substitution models of structural protein stability. Importantly, note that this option requires that the setting "*NEUTRAL=*" must be set to 0, so a fitness substitution model is specified. Example: *NPOP= 10*

Other settings

TYPE_BL= #: Type of branch lengths (by mutations or substitutions)

This argument specifies if branch lengths are either considered by events of mutation (1) or substitution (2). Note that the expected number of events = branch length × number of sites. Example: *TYPE_BL= 2*

OUTPUT_LEVEL= #: Amount of output files

This argument specifies the amount of output files printed using protein stability substitution models (details shown later). "2", all output files are printed. "1", only the final output files are printed. "0", the output files are not printed. Importantly, note that a total of 4 output files related with these substitution models are printed per branch and therefore, it can slow down the simulations and may need a lot of space in the hard disk. Example: *OUTPUT_LEVEL= 1*

3.7. Additional input files

The following input files are optional and if required, must be also present in the same directory of the executable of *ProteinEvolver*.

3.7.1. Heterogeneous recombination along sequences

The user can optionally simulate recombination hotspots following the algorithm implemented in SNPsim [25] when using coalescent simulations. This file must be specified in the coalescent settings (*hNAME*; see previous sections) and should be placed in the directory of the executable. An example file "*UserHetRec.in*" is included in the folder with examples. As for the coalescent simulation, the model of recombination hotspots implemented in this version of the program is that one presented in the previous version [1], thus this section was not changed.

Possible arguments for *recombination hotspots file* are (# means number):

k# : Hotspot recombination rate

Expected recombination rate at the hotspots sites. If the hotspots are homogeneous (option *-t#* is not invoked) all the hotspots have the same rate. Example: *k1e-4* (= *k0.0001*)

h# : Expected number of hotspots

This is the expected number of hotspots for a given sample in the absence of interference. This parameter corresponds to the intensity parameter for a Poisson distribution from which the actual number of hotspots is drawn. For a given sample, the actual number of hotspots will change around this value. It does not have to be an integer. NOTE: When interference is specified we need to divide this number by the interference interval (*z#*) to obtain the expected number of hotspots. It does not have to be an integer. Example: *h1.1*

q# : Fixed number of hotspots

This option fixes the number of hotspots inside the region of interest, so every sample will have the same number. In this case the hotspot locations are chosen from a uniform distribution. If the hotspots overlap, they will be displaces to the closest available location. Note that in this case no recombination events will originate from a hotspot located outside the region of interest. Example: *q3*

v# : Hotspot imprecision
The hotspot imprecision corresponds to the variance of a Normal distribution for the specific site to recombine around the hotspot center (chosen by a Poisson process). The bigger the imprecision, the wider is the hotspot. If the imprecision is 0, all the recombination events happen exactly at the hotspot center. See figures 2 and 3. Example: *v0*

m# : Hotspot width
This option specifies the width of the hotspots. In this case any site in the hotspot has the same probability of recombination. If the width is 1 all the recombination events happen exactly at the hotspot center. This parameter has to be bigger than 0. See figures 2 and 3. Example: *m1*

t# : Hotspot heterogeneity
This parameter indicates that there is hotspot heterogeneity, that is, hotspots may have different recombination rates. This heterogeneity is accomplished through the use if the continuous gamma distribution. The shape parameter of this distribution (%) will control the strength of this heterogeneity. The smaller the shape the strongest the heterogeneity. This is similar to the application of Yang [37]. Example: *t0.5*

z# : Hotspot interference
This parameter indicates whether the location of the hotspots is not independent of each other. If this parameter is 1 there is no interference, if it is between 0 and 1 hotspots tend to cluster, and if it is bigger than 1 hotspots will tend to be pushed away from each other. Example: *z1*

3.7.2. User-specified sequence assigned to the root node

Sequence evolution is simulated forward in time, from the past to the present, along the branches of the evolutionary history. The number of substitution events is given by the branch lengths while the type of substitution events is given by the substitution model of evolution. Therefore, the simulation of sequence evolution starts from the sequence of the root (i.e., GMRCA or MRCA) node. By default, the sequence of the root node is simulated according to the nucleotide or amino acid frequencies. However, the user can optionally specify its own root sequence by a text file (the name of this file must be specified in the main settings (*-xNAME*), which should be located in the directory of the executable. The file just contains a single sequence. An example file "*seqRoot.in*" is included in the folder with examples.
This option is important for the SCS models where the sequence assigned to the root node must be equal (for the simulation of proteins) or codify (for the simulation of coding DNA) to the amino acid sequence of the PDB file. **Consequently, under SCS models, it is highly recommended introduce directly the amino acid sequence of the PDB file or a coding sequence that codifies for such a PDB sequence**.

3.7.3. User-specified global and site-specific empirical substitution models

The user can optionally specify a particular empirical amino acid matrix from a text file, which has to be located in the working directory. It must include the matrix of relative rates of change among amino acids and the 20 amino acid frequencies, following the format of the popular program PAML [41]. Thus, the order of amino acids must be: A R N D C Q E G H I L K M F P S T W Y V.

The user can either specify a matrix and frequencies that would be applied to all the sites (the same model for all the protein sites; i.e., see the example input file "*userEAAM*") or specify a matrix and frequencies for every site (a different model for every protein site; i.e., see the example input file "*userEAAMsites*"). For the latter, the format is the same but applied to every site (see the example of input file).

3.7.4. Variation of the global substitution rate among sites

The user can optionally modify the substitution rate applied to every site. This file must be specified in the main settings by "-_*NAME*" and should be located in the working directory. An example file is included with the package (file named "*HetRatesVector.in*") and it is shown below.

The file consists two sets of numbers specified after an "*r*". First, the sequence length (in the example, 255 nucleotides or amino acids). Second, a factor for each site separated by a space. Thus, the number of factors must be equal to the number of sites.

```
[rate heterogeneity by a vector]
[options and comments within brackets are ignored]

[rates] r255
1 1 1 1 1 1 1 1 1 0.6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0.7 0 1 0.6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0.5 0.5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 0.7 0 0 0 0 0 0.6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0.8 0 0 0 0.7 1 1 1 1 1
1 1 1 1 1
```

The factor specified at each site, which can takes values only between 0 and 1, multiplies the original substitution rate for that site. Thus, a factor of 1 means that the original substitution rate remains invariant while a factor of 0 means that the substitution rate is 0 and therefore that site will remain totally conserved. For example, this option can be useful when one already knows the importance of certain sites in the protein, for example catalytic sites could have a factor of 0 in order to keep their required activity.

3.8. Default settings

The user must specify at least the number of replicates, either in the command line,

./ProteinEvolver2.2.0 -n1

Or using a *parameters* file,

./ProteinEvolver2.2.0 [the parameters file is present in the working directory]

If the number of replicates is specified, by default it simulates a few DNA sequences under the coalescent and a basic DNA substitution model of evolution (JC).

Table 1. Key arguments for *ProteinEvolver2.2.0*. The user can specify several parameters to define different evolutionary scenarios. Some of these arguments (Argms) can be entered in the command line, but we recommend use only the input text files for simplicity.

Parameter	**Argms**	Example of specification	Process	Input file
Number of replicates	**n**	1	All	parameters
Birth-death process	**l**	BirthDeath.in	Evolutionary history	parameters
Coalescent process	**s**	Coalescent.in	Evolutionary history	parameters
Input phylogenetic tree	**p**	treefile.in	Evolutionary history	parameters
Nucleotide frequencies	**f**	0.4 0.3 0.1 0.2	Substitution models of DNA evolution	parameters
Transition / transversion ratio	**v1**	2.1	Substitution models of DNA evolution	parameters
Relative symmetrical substitution rates	**v6**	1.0 2.3 2.1 3.0 4.2 1.0	Substitution models of DNA evolution	parameters
Relative asymmetrical substitution rates	**v12**	0.1 0.2 0.3 0.4 0.9 0.6 0.9 0.8 0.9 0.1 0.2 0.3	Substitution models of DNA evolution	parameters
SCS models[1] for DNA	**m**	Pop_evol.in	Substitution models of DNA evolution	parameters
Empirical amino acid substitution model	**@**	JTT	Substitution models of protein evolution	parameters
SCS models[1] for proteins	**z**	Pop_evol.in	Substitution models of protein evolution	parameters
Rate variation among sites[2]	**a**	0.4	Substitution models of evolution	parameters
Proportion of invariable sites	**i**	0.2	Substitution models of evolution	parameters
Variable substitution rates per site	**_**	HetRatesVector.in	Substitution rate	parameters
User-specified sequence for the root node	**x**	seqRoot.in	Molecular evolution	parameters
Print sequences to a file	**b**	sequences	Output information	parameters
Format of simulated multiple sequence alignments		2 1 0	Output information	parameters
Print root (GMRCA or MRCAs) sequence	**$**		Output information	parameters
Print simulated trees	**j**	trees	Output information	parameters
Print times of nodes of genealogies	**k**	times	Output information	parameters

Print simulated ARG	*	NetworkFile	Output information	parameters
Print recombination breakpoints	**d**	breakpoints	Output information	parameters
Seed	#	4	All	parameters
Level of output information printed on the screen	**y**	2	Output information	parameters
Birth and death rates (given or calculated)	**l**	0.4 0.2	Birth-death evolutionary process	BirthDeath.in
Ending the simulation of the birth-death process	+	1 3 15	Birth-death evolutionary process	BirthDeath.in
Prune extinct nodes	>	1	Birth-death evolutionary process	BirthDeath.in
Outgroup and its branch length	**o**	0.001	Birth-death evolutionary process	BirthDeath.in
Substitution rate	**u**	9.1e-2	Birth-death evolutionary process	BirthDeath.in
Effective population size; Haploid/Diploid	**e**	1000 2	Birth-death evolutionary process	BirthDeath.in
Alignment length (nt or aa)	**s**	255	Birth-death evolutionary process	BirthDeath.in
Sample size and alignment length (nt or aa)	**s**	8 255	Coalescent evolutionary process	Coalescent.in
Effective population size; Haploid/Diploid	**e**	1000 2	Coalescent evolutionary process	Coalescent.in
Tip dates[3]	=	2 1995 1 3 2003 4 8	Coalescent evolutionary process	Coalescent.in
Generation Time	/	400	Coalescent evolutionary process	Coalescent.in
Exponential growth rate	**g0**	$2.1x10^{-5}$	Coalescent evolutionary process	Coalescent.in
Demographic periods[4]	**g1**	1 1000 5000 200	Coalescent evolutionary process	Coalescent.in
Migration model (island, stepping-stone, island-continent) and population structure	**q**	1 4 2 2 3 1	Coalescent evolutionary process	Coalescent.in
Migration rate (constant or variable with time)	**t**	1 0.0005	Coalescent evolutionary process	Coalescent.in
Convergence of demes[5]	**%**	1 2 5000	Coalescent evolutionary process	Coalescent.in
Homogeneous recombination rate	**r**	5×10^{-6}	Coalescent evolutionary process	Coalescent.in
Fixed number of recombination events	**w**	3	Coalescent evolutionary process	Coalescent.in

Recombination hotspots	**h**	UserHetRec.in	Coalescent evolutionary process	Coalescent.in
Substitution rate	**u**	5.1×10^{-4}	Coalescent evolutionary process	Coalescent.in
Outgroup and its branch length	**o**	0.1	Coalescent evolutionary process	Coalescent.in
Number of phylogenetic trees, alignment length (nt or aa) and rooted phylogenetic tree		1 255 ((taxonA:0.1,taxonB:0.1):0.4,(taxonC:0.1,taxonD:0.1):0.5,taxonE:1.0);	User-specified phylogenetic tree	treefile.in
Sequence length	**r**	255	Substitution rate	HetRatesVector.in
Factor that multiplies the original substitution rate		0 0 0 … 1 1 1	Substitution rate	HetRatesVector.in
Sequenced assigned to the root node		MRHPLVMG…	Molecular evolution	seqRoot.in
PDB file (must be in the working directory)	**PDB=**	1TRE.pdb	Substitution models of protein evolution	Pop_evol.in
Chain of the PDB file	**CHAIN=**	A	Substitution models of protein evolution	Pop_evol.in
Input file of amino acid contacts (must be in the working directory)	**FILE_STR=**	structures.in	Substitution models of protein evolution	Pop_evol.in
Thermodynamic temperature	**TEMP=**	1.8	Substitution models of protein evolution	Pop_evol.in
Configurational entropy per residue (unfolded)	**S0=**	0.05	Substitution models of protein evolution	Pop_evol.in
Configurational entropy per residue (misfolded)	**SC1=**	0.05	Substitution models of protein evolution	Pop_evol.in
Configurational entropy offset (misfolded)	**SC0=**	0	Substitution models of protein evolution	Pop_evol.in
Third cumulant in REM calculation	**REM3=**	0	Substitution models of protein evolution	Pop_evol.in
Type of SCS model (Neutral or Fitness)	**NEUTRAL=**	1	Substitution models of protein evolution	Pop_evol.in
Effective population size for the fitness SCS model	**NPOP=**	10	Substitution models of protein evolution	Pop_evol.in
Consideration of branch lenghts	**TYPE_BL=**	2	Substitution models of protein evolution	Pop_evol.in

Amount of information about SCS models printed as output files	**OUTPUT_LEVEL=**	2	Substitution models of protein evolution	Pop_evol.in

[1]SCS models can be neutral or fitness-based landscape.
[2]Shape of the gamma distribution.
[3]In presence of convergence of demes, the time of the tip dates must be younger than the time of the convergence of demes.
[4]From 1000 to 5000 effective population size during 200 generations.
[5]Deme 1 is converging with deme 2 at time 5000 to make a new ancestral deme 3.

3.9. Output Files

All output files produced by the program are written into a folder "*Results*" created during the execution of a run.
- *Sequences file*: contains the multiple alignment of simulated DNA or protein sequences in *Phylip* sequential, *Fasta* or *Nexus* formats.
- *Breakpoints file*: contains the breakpoints ordered by event time. Disabled when the birth-death process or input tree/s are specified.
- *Trees file*: contains the output trees in newick format simulated with the birth-death process or by the coalescent process.
- *Network file*: contains the simulated ARG by a branches format (each line contains two connected nodes) [42]. Disabled when the birth-death process or input tree/s are specified.
- *Times file*: contains information about time of internal nodes and branch lengths.
- *CatMRCA/GMRCA*: contains the sequence of the root node: catMRCA (concatenated MRCA fragments) and/or GMRCA (sequence of the GMRCA node) in the ARG.
- *Settings file*: contains a summary of applied settings showed in the screen at the end of the simulation. This option must be activated in the source code.

3.9.1. Output files from the SCS models

The substitution models that account for the structural protein stability can produce additional output files that are printed within an output folder "*ProteinStability*", located in the output folder "*Results*".
Note that these output files are printed for each branch, so be careful when printing all output files in extensive simulations because it could require a lot of space in the disk and might slow down the simulations. See the argument *OUTPUT_LEVEL* in the section to control the amount of printed output files.
The name for each output file consist of the prefix "*ProteinStability_*" followed by the number of replicate "*Replicate#_"*, the branch number "*Branch#_"*, and details about the particular settings specified: PDB name "*NAME*", temperature "*T#*", entropy "*S#*", protein population size "*N#*", and optionally GC bias for coding DNA simulation "*GC#*" (note that "#" means a number).
Example from the simulation of protein evolution,
ProteinStability_Replicate5_Branch12_1TREA_DeltaG_2.dat
ProteinStability_Replicate5_Branch12_1TREA_T1.80_S00.05_N10_ave.dat
ProteinStability_Replicate5_Branch12_1TREA_T1.80_S00.05_N10_stab.dat
ProteinStability_Replicate5_Branch12_1TREA_T1.80_S00.05_N10_final.dat

Files "*DELTAG_2.dat*" show the protein energy values at amino acid level (ΔG/L) as a function of the temperature (T) computed by the REM2 (Random Energy Model) energy function (see later).
Files "*ave.dat*" show information about the user-specified settings, computed energy, fitness and, transition and mutation loads.
Files "*dna.dat*" show information about the user-specified settings and GC content.
Files "*stab.dat*" show, for each accepted mutation (substitution), the structural energy of the mutated and native proteins, derived fitness and number of mutation attempts to reach such a substitution. It is recommended to explore the influence of substitution events in the protein structure.

Files "*final.dat*" show the final results derived from the substitutions introduced on a branch. It indicates the mean of fitness, energy, entropy, transition and mutation loads and, rejected mutations.

3.10. Solving message errors

- Parameter values for the birth-death process. Take into account that the time of evolution depends on the population size (units of 2N generations) if the population size is specified. This is particularly important if the birth-death process ends according to a user-specified time of evolution.

- During the birth-death process many extinctions could occur disabling the possibility to produced a desired number of present nodes. Moreover, if the first node becomes extinct, thus without descendants, the phylogenetic history could have only one node, which complicates the printing of the phylogenetic tree, among others.

- Parameter values for the SCS models. Entropy and temperature must be chosen carefully, specially for fitness SCS models. See above and in the examples the recommended settings.

- There is a common message of error due to not enough memory in coalescent simulations with very high recombination rate. Note that recombination increases the number of nodes of the ARG and the number of recombinant fragments within nodes (see section 4.3.1). Then, when ρ (= 4Nrl) is very high, the ARG must save in memory a high amount of information, and with time such amount exponentially increases. Thus, at a given time the machine could not have enough available memory and the program stops with an error message like the following:
*malloc: *** mmap(size=1584291840) failed (error code=12)*
**** error: can't allocate region*
Could not reallocate segments (1584144000 bytes)
In this situation we recommend to reduce the value of the following parameters: recombination rate, population size, length of the sequences and sample size. Of course, the other option is just to use another machine with more memory.

- Using a fixed number of recombination events keep in mind that the recombination rate introduced should generate an accordingly number of recombination events. Otherwise the simulation could never finish (because the assumption of the fixed number of recombination events is never successful) leading to crash the execution.

- Using growth rates or demographic periods you could find an error like:
ERROR: Coalescent time (nan) is infinite
This might suggest that the growth rate is too negative
and the coalescent time is therefore infinite.
Try a smaller value
This is because the population increases too much going back in time and thus, the coalescent time gets infinite. The best option to solve this error is to reduce the growth rate. In the case of demographic periods this could be done for example by using longer times for such period.

- Asymmetric substitution rates (for example, from user-specified amino acid matrices) could be problematic. In these cases the program writes an output message about "complex roots".

- Input tree from user. The file with the input tree must follow: first site should be 1 and the last site should be the total number of sites, no empty lines, the tree should be rooted.

I recommend using the provided example input files and modify them carefully, considering the information described in this documentation.
If you experience any unexpected error or there is any doubt, please do not hesitate to contact me marenas@uvigo.es
Thanks for your contribution!

4. ProteinEvolver model

ProteinEvolver can simulate multiple alignments of DNA and protein sequences under diverse evolutionary scenarios. The new version, *ProteinEvolver2.2.0*, combines for the first time the simulation of the evolutionary history with the simulation of molecular evolution where the fitness observed at the molecular level is considered to build the evolutionary history.

4.1. The standard birth-death evolutionary process

Given the birth and death rates, the birth-death process is applied forward in time to decide whether the next event is a birth or a death. The birth-death process (and the whole simulation process) ends when a user-specified criterion is reached, including a particular sample size (with or without including extinction nodes) or time of evolution is reached. The implemented standard birth-death process consists of a series of steps that follow traditional birth-death models [2, 3], see also the nice and illustrative Chapter 10 of the book Phylogenetic Comparative Methods by Luke J. Harmon [https://lukejharmon.github.io/pcm/].

1. Starting from an "active" node (i.e., the root node) at current time t_c, one can calculate the time to the next event (birth to obtain 2 descendants or death to obtain the extinction of the node). This time t_n usually follows an exponential distribution with rate based on the number of active demes (s), birth rate (b) and death (d) rate,
$$t_n = e^{(s(b+d))}$$
2. Evaluate if the simulation ends before reaching the next event $(t_c + t_n)$.
3. If yes, the simulation of the evolutionary history finishes.
4. If no, evaluate the type of the next event (a birth or a death). The probability of a birth event is $P_b = b/(b + d)$ while the probability of a death event is $P_d = d/(b + d)$. Take a random number (x) between 0 and 1 and assign a birth event if $x < P_b$, or a death event if it does not.
5. If a birth event is selected. Then, randomly select an active node. Add two descendant active nodes to such a selected node with branch lengths t_n. Therefore, two active nodes are obtained from the original (ancestral) node.

6. If a death event is selected. Then, randomly select an active node and this node is considered as death.
7. Return to step 1 while the user-specified criterion for ending the process is not reached and there is at least one active node in the tree. Otherwise, the simulation finishes.

This process provides a phylogenetic tree that includes both nodes that reached the present time and nodes that were extinct. One can prune the extinct nodes if desired (option implemented in the program), to return a birth-death tree of survivors, which could represent a typical sample of present individuals.

Once the birth-death evolutionary history is simulated, the program simulates molecular evolution upon such evolutionary history. Thus, in this simulation (as well as in simulations of molecular evolution upon coalescent evolutionary histories or user-specified phylogenetic trees) the molecular evolution is simulated upon a given evolutionary history in a somehow two separate steps process (i.e., the fitness of the molecular data is not considered in the phylogenetic tree, thus the phylogenetic tree assume a neutral evolution while the molecular evolution could consider selection, which is an unrealistic discrepancy). Also notice that this birth-death process considers the same birth and death rates for all the nodes and over time, which can also be unrealistic.

4.2. Combining the birth-death evolutionary history from population genetics with molecular evolution that considers selection of the protein folding stability

In order to build a more realistic simulation combining the birth-death process with molecular evolution, we can consider the fitness of the molecular data at every node to build the birth-death evolutionary history of the corresponding data.

First, a protein sequence assigned to every node, starting from the root node. The fitness of this sequence is calculated in terms of folding stability as free energy *(ΔG*) following the Boltzmann distribution [43],

$$f(A) = \frac{1}{1 + e^{\Delta G/kT}}$$

where the fitness takes values from 0 to 1. Next, the program allows the user to consider whether a high fitness based on the folding stability is preferred or whether a fitness similar to the fitness of a real sequence (i.e., sequence from a PDB file) is preferred (one of these options must be selected by the user in the birth-death input file). In case of considering the first option, a high fitness based on the folding stability is preferred, the birth rate is assumed equal to the fitness. In case of considering the second option, a fitness similar to the fitness of a real sequence is preferred, the birth rate is assumed as 1 minus the root square of the difference (in terms of RMSD) of fitness of the studied and real sequences. Thus, the smaller is this difference, the higher is the birth rate. In any case, the death rate is assumed as 1 minus the birth rate, allowing a constant global rate (birth and death rates). We recommend considering a fitness similar to the fitness of a real sequence because in nature a high stability not necessarily indicates a high fitness but a stability close to a real stability could suggest a high fitness because the real stability is the result of a previous selection process.

In contrast to the standard birth-death process presented before, one could expect that every node can have a particular birth and death rates according to the fitness of its

corresponding molecular sequence. The implemented birth-death process that considers the birth and death rates is described below.

1. Following the expressions presented above, the birth and death rates are calculated for every active node, starting from the root node.
2. Calculation of the time to the next event (birth to obtain 2 descendants or death to obtain the extinction of the node). Again, this time t_n usually follows an exponential distribution with rate based on the number of active demes (s), birth rate (b) and death (d) rate (see the expression below). Since $b+d=1$ at every node (see above), the time to the next event (birth or death) is the same for every node. Thus, from current time t_c, one can calculate the time to the next event (birth to obtain 2 descendants or death to obtain the extinction of the node).
$$t_n = e^{(s(b+d))}$$
3. Evaluate if the simulation ends before reaching the next event $(t_c + t_n)$.
4. If yes, the simulation of the evolutionary history finishes.
5. If no, an active node is randomly selected and it is evaluated to identify the type of the next event (a birth or a death). The probability of a birth event is $P_b = b/(b + d)$ while the probability of a death event is $P_d = d/(b + d)$. Take a random number (x) between 0 and 1 and assign a birth event if $x < P_b$, or a death event if it does not.
6. If a birth event is selected. Then, add two descendant active nodes to the node with branch lengths t_n. Therefore, two active nodes are obtained from the original (ancestral) node. Importantly, next, molecular evolution is simulated from the original (ancestral) node to every descendant node according to the specified SCS model of protein evolution (details later). Thus, a molecular sequence for every descendant node is obtained. Finally, the protein folding stability, and subsequent fitness, is calculated for every new molecular sequence.
7. If a death event is selected. Then, the node is considered as death.
8. Return to step 1 while the user-specified criterion for ending the process is not reached and there is at least one active node in the tree. Otherwise, the simulation finishes.

Notice that this process simultaneously simulates both evolutionary history and molecular evolution, thus both evolutionary history and molecular evolution consider selection from the protein folding stability. In addition, selection varies among molecular sequences of the corresponding nodes of the evolutionary history.

4.3. Markov substitution models of molecular evolution

Using these substitution models, sequences can be evolved along branches of the evolutionary history. The result is a random sample of aligned coding DNA or amino acid sequences. *ProteinEvolver* implements multiple nucleotide and amino acid Markov models through the specification of different parameters (including base frequencies, relative substitution rates, transition/transversion ratio, a proportion of invariable sites and rate variation among sites) and empirical matrices. Conveniently, the sequence of the root node can specified at random, according to the nucleotide or amino acid frequencies, or the user can specify that sequence (highly recommended for SCS models).

In addition, heterogeneous substitution rates among sites by a gamma distribution +G and proportion of invariable sites +I are implemented. Furthermore, the user can alter the substitution rate for each site, for example allowing the fixation at particular sites like those related to the activity of the protein (i.e., catalytic sites).

4.3.1. DNA substitution models

ProteinEvolver implements the general time reversible model for nucleotide substitution (GTR) [44], a non reversible version (GTnR) [44], and models nested therein, like JC [35], K80 [45], F81 [36] or HKY [46].

4.3.2. Empirical amino acid substitution models

ProteinEvolver implements a variety of empirical amino acid models: *Blosum62* [10], *CpRev* [11], *Dayhoff* [12], *DayhoffDCMUT* [13], *FLU* [6], *HIVb* [14], *HIVw* [14], *JTT* [15], *JonesDCMUT* [13], *LG* [16], *Mtart* [17], *Mtmam* [18], *Mtrev24* [19], *RtRev* [20], *VT* [21], *WAG* [22]. In addition, it is possible to input a user-defined empirical amino acid model (relative substitution rates among amino acids and amino acid frequencies at the equilibrium) by an input file. In this regard, this version of the program allows the user-specification of an empirical model applied to all the sites and, the user-specification of a site-specific empirical model (thus every site can have a substitution model that can differ from the model at other sites).

4.4. Structurally constrained substitution (SCS) models

These models were already implemented in the previous version of the program and consist of site-dependent substitution models that consider the stability of the protein structure.

These models compute the structural energy of mutated proteins (including misfolded and unfolded configurations) given the entropy and the thermodynamic temperature. Then, mutations can be accepted (substitutions) or rejected according to the Moran process [47, 48] that provides the probability of accepting the new protein state $P=(1-(f_i/f_j)^a)/(1-(f_i/f_j)^{2N})$, where i and j are the protein states before and after the evolutionary event (where a is 2 or 1 for haploid or diploid population, N is the estimated effective population size), further details are shown below. Two substitution models have been implemented, the neutral model (which does not consider population size) and the fitness model (which needs a user-specified population size). Indeed, these models can be crossed with classic site-independent substitution models to obtain the flux of the substitution process (details are shown later).

The SCS models take into account the stability of the protein structure. These models compute the structural energy of mutated proteins given the entropy, contact matrices, the protein structure and the temperature. Then, mutations are evaluated and can be accepted (substitutions) or rejected.

The implemented site-dependent SCS models estimate the stability of the mutated sequence folded into the target structure at the simulation temperature by means of a contact free energy function. The contact matrix C_{ij} takes the value 1 if residues i and j are close in space and 0 otherwise (by a threshold distance of 4.5Å), and it is sufficient to reconstruct the three-dimensional structure of the protein up to good accuracy [49]. It assumes that the free energy of a protein with sequence A folded into the contact matrix C is given by the sum of its pairwise contact interactions,

$$E(A,C) = \sum_{ij} C_{ij} U(A_i, A_j)$$

where $U(a,b)$ is the contact interaction matrix that expresses the free energy gained

when amino acids *a* and *b* are brought in contact. It adopts the contact interaction matrix determined in Bastolla et al. [50]. For proteins that fold with two-states thermodynamics, i.e. for which only the native structure and the unfolded structure are thermodynamically important, stability against unfolding is defined as the free energy difference between the folded and the unfolded states. The free energy of the unfolded state is estimated as *sL*, where *L* is protein length and *s* is an entropic parameter, is estimated as $\Delta G \sim E(A,Cnat)+sL$, where *Cnat* is the native structure and $s = 0.074$ was determined fitting the above equation to a set of 20 experimentally measured unfolding free energy, yielding a correlation coefficient r = 0.92 (U. Bastolla, unpublished data). The accuracy of this method for predicting the stability effect of mutations is comparable to state-of-the-art atomistic methods such as FoldX [51], and its computational simplicity allows to use it for simulating protein evolution for long evolutionary trajectories and complex phylogenetic histories.

Stability against unfolding is however not sufficient to characterize protein stability. Consequently, the model has also to check the stability against compact, incorrectly folded conformations of low energy that can act as kinetic traps in the folding process and, in many cases, give raise to pathological aggregation. The term positive design indicates sequence features that favor protein stability by decreasing the free energy of the native structure. On the other hand, stability against misfolding is realized by increasing the energy of key contacts that are frequently found in alternative structures, which is termed negative design [52-54]. Therefore, it is also influenced by mutations at positions that are distant in the native structure.

Stability against misfolded structures is difficult to estimate, and several models of protein evolution do not consider it, despite its importance is being more and more recognized. Here it considers the set of alternative compact matrices of *L* residues that can be obtained from non-redundant structures in the Protein Data Bank (PDB). This procedure, called threading, guarantees that the contact matrices fulfill physical constraints on chain connectivity, atomic repulsion, and hydrogen bonding (secondary structure), which are not enforced in the contact energy function.

The free energy of this misfolded ensemble can be estimated in analogy with the Random Energy Model (REM, [55]) as,

$$G_{misfold} \approx \langle E(A,C) \rangle - \frac{\sigma^2}{2k_B T} - k_B T s_c L$$

where $\langle E(A,C) \rangle$ is the mean and σ^2 is the variance of the energy of alternative structures [56]. This formula holds for temperatures above the freezing temperature at which the entropy of the misfolding ensemble vanishes. At lower temperatures the free energy maintains the same frozen value [55]. A precise computation showed that the third moment of the energy cannot be neglected (Minning et al. 2013). Therefore, it implements the equation,

$$G_{misfold} \approx \langle E \rangle - \frac{\left\langle \left(E - \langle E \rangle\right)^2 \right\rangle}{2k_B T} + \frac{\left\langle \left(E - \langle E \rangle\right)^3 \right\rangle}{6(k_B T)^2} - k_B T s_c L$$

$$= \sum_{i<j} \left\langle C_{ij} \right\rangle U_{ij} - \sum_{i<jk<l} D_{ijkl} \frac{U_{ij} U_{kl}}{2T} + \sum_{i<j,k<l,m<n} F_{ijklmn} \frac{U_{ij} U_{kl} U_{mn}}{6T^2} - k_B T s_c L$$

where the tensors $\left\langle C_{ij} \right\rangle, D_{ijkl}$ and F_{ijklmn} are the averages over the set of contact matrices of *L* residues respectively of single contacts, contact correlations and triples of contacts and, $U_{ij} = U(A_i, A_j)$ only depends on the protein sequence [54]. The computing time is

considerably reduced by approximating the above formula with one that only depends on pairs of residues,

$$G_{misfold} \approx A^{(1)}\langle U\rangle - \frac{A^{(2)}\langle U\rangle^2 + \sum_{ij} B_{ij}^{(1)} U_{ij}^2}{2k_B T} + \frac{A^{(3)}\langle U\rangle^3 + \langle U\rangle \sum_{ij} B_{ij}^{(2)} U_{ij}^2 + \sum_{ij} B_{ij}^{(3)} U_{ij}^3}{6(k_B T)^2} - k_B T s_c L$$

The quantities $A^{(1)}\, A^{(2)}\, A^{(3)}\, B_{ij}^{(1)}\, B_{ij}^{(2)}\, B_{ij}^{(3)}$only depend on the set of alternative contact matrices and on protein length *L*, and they are pre-computed before the simulation starts. In this way, we can evaluate how the misfolded free energy changes upon mutation only performing order *L* operations for computing $U(A_i = b, A_j) - U(A_i = a, A_j)$when the residue at the mutated site *i* changes from state *a* to *b*. The stability of the native state is finally evaluated as the difference in free energy between the native, the unfolded and the misfolded states, $\Delta G = E(A, C_{nat}) - G_{misfold} - k_B T s_u L$.

Note that, even if the two configurational entropies per redisue s_u(unfolded ensemble) $s_C$act additively, the free energy is not simply a function of their sum, since it is only $s_C$that determines the freezing temperature of the misfolded ensemble.

For modeling protein evolution, we still have to define how protein stability influences fitness. The simplest possibility is a neutral fitness landscape where the fitness is a binary variable and all proteins with stability above a given threshold, i.e. $\Delta G < \Delta G_{thr}$are considered viable and equally fit, whereas all proteins below threshold are considered lethal and therefore, discarded. The threshold is $\Delta G_{thr} = \Delta G(A_0, C_{nat})$ where *A0* is the protein sequence in the PDB, which means that the neutral SCS model is not sensible to variations of the temperature or configurational entropies.

The neutral fitness landscape can be generalized to a landscape in which fitness is an increasing function of stability, and in particular it is proportional to the fraction of protein that is in the native state [56],

$$f(A) = \frac{1}{1 + e^{\Delta G(A, C_{nat})/kT}}$$

Note that the fitness landscape can be reduced to the neutral landscape in the limit of very small temperature, since in this limit the fitness tends to 1 if $\Delta G < 0$and to zero if $\Delta G > 0$.

Then, it assumes that the mutation rate is very small so that the population is monomorphic, and model selection through the Moran process [47], which yields the fixation probability of a mutation event,

$$\Pi(ij) = \frac{1 - \left(\frac{f_i}{f_j}\right)^a}{1 - \left(\frac{f_i}{f_j}\right)^{2N}}$$

where $f_i$is the fitness of the wild-type, $f_j$is the fitness of the mutant, *N* is the effective population size and *a* = 2 or 1 for a haploid or diploid population, respectively. Given the probability of fixation, the succession of mutant fixations can be depicted as a Markov process, in which the genotype of the population moves from one sequence to another one according to the mutation and fixation probabilities.

4.4.1 Crossing the SCS models with classic Markov substitution models

The SCS models implemented in *ProteinEvolver* can be crossed with any parametric DNA substitution model (such as JC, HKY or GTR) or any empirical amino acid substitution model (such as JTT, WAG or LG). The matrices of rate of change and the nucleotide or amino acid frequencies are considered to compute the rates of change per site and state. Two types of material can be simulated using SCS models, coding DNA and amino acid sequences.

4.4.1.1. Simulation of coding DNA data with the SCS models

Basically, the simulation of coding DNA data consists in the following algorithm (which can be specified with the argument "*-m*" in the parameters file).

I) Mutation rates among DNA states and per site are computed according to the rates of change provided by the matrix of the classic Markov model (JC, HKY or GTR, parameters specified with the command *"-v*" in the parameters file) and the nucleotide frequencies "*-f4*".

II) According to the mutation rates, a mutation at DNA level is introduced.

III) The DNA sequence is translated to an amino acid sequence and the mutation is classified as synonymous or non-synonymous.

IV) If the mutation is synonymous, it is accepted since it does not alter the protein sequence and structure.

V) If the mutation produces a nonsynonymous change, it is evaluated by the computation of the structural protein energy for the mutated amino acid protein (details above), the energy is used to compute a fitness value (details above) and the mutation can be accepted or rejected according to the Moran process [47] (details above). The mutation can be accepted or rejected and the DNA and protein sequences are updated according to that result.

In SCS models, the number of mutations or substitutions (accepted mutations) is computed according to the branch length and the user can specify if such branch length should be considered as mutations or substitutions.

4.4.1.2. Simulation of protein data with the SCS models

The simulation of protein data consists of the following algorithm (which can be specified with the argument "*-z*" in the parameters file).

I) Mutation rates among amino acid states and per site are computed according to the rates of change provided by an empirical amino acid matrix (WAG, JTT or HIVb, parameters specified with the command *"-@*" in the parameters file) and the nucleotide frequencies "*-f*".

II) According to the mutation rates, an amino acid mutation is applied.

III) The mutation is evaluated by the computation of the structural protein energy for the mutated amino acid protein (SCS model), the energy is used to compute its fitness and the mutation can be accepted or rejected [47]. The protein sequence is updated according to the obtained result.

The number of mutations or substitutions (accepted mutations) is computed according to the branch length and the user can also specify if the branch length is again considered as number of mutations or number of substitutions.

4.4.2 Evolution of sequences along trees under the SCS models

While under classic Markov substitution models the evolution of the sequence is preformed site by site along the branches of the tree, because the site-independent aspect of these models, the simulation under SCS models requires the evolution of the whole molecule along the tree because the site-dependent aspect.
This leads to a problem when dealing with recombination because half a protein cannot be evolved independently. Consequently, the user-specified tree cannot be reticulated and recombination is only allowed in *ProteinEvolver*, for these structural substitution models, by a special algorithm described later. Here, the evolution starts from a molecule assigned to the GMRCA node of the ancestral recombination graph (ARG) [32]. By this procedure all the material involved in the recombination events is considered.

4.5. Coalescent models to simulate the evolutionary history

The input phylogeny can be user-specified or simulated with the birth-death process (described before) or with the coalescent. Regarding the coalescent, which was already implemented in the previous version of the program, the program implements an extension of the coalescent with recombination, demographics and migration based on the neutral Wright-Fisher model [23, 26, 38] following [31, 57]. Given the specified recombination and migration rates (and other parameters like the effective population size (N) and population growth rate) random genealogies are produced. Recombination hotspots are also implemented following *SNPsim* [25]. Several migration models (island [28], stepping-stone [27] and continent-island [28]) are allowed and migration rate can change with time. The evolution of the demes (or species tree) can be fixed. Complex demographic histories can be implemented by defining demographic periods in which population sizes augment, reduce, or remain constant. Samples can be collected at same or different times [see, 29] and simulated data can be haploid or diploid. Further details are presented below.
The coalescent proceeds backwards in time starting from a sample of s gametes. Time is scaled in units of $2N$ generations, where N is the effective population size. For a given site, without recombination or migration, and under constant population size, the time to the most recent common ancestor (TMRCA) is,

$$E(TMRCA) = 2\left(1-\frac{1}{s}\right)$$

$$Var(TMRCA) = \sum_{i=2}^{s}\frac{4}{i^2(i-1)^2}$$

The times to a coalescence (CA), recombination (RE) or migration (MI) event are exponentially distributed, with intensity equal to their respective rates (see below). The next event will be the one that would occur before according to these expectations.

$$Time\ to\ CA \propto Exp\left[rateCA\right]\cdot 2N$$
$$Time\ to\ RE \propto Exp\left[rateRE\right]\cdot 2N$$
$$Time\ to\ MI \propto Exp\left[rateMI\right]\cdot 2N$$

The rate of coalescence depends only on the number of lineages (k),

$$rateCA = \frac{k(k-1)}{2}$$

4.5.1. Recombination

The recombination rate depends on the population size (N) and on the total recombination rate at all valid recombination sites (G). A *valid* recombination site must have at both sides ancestral material did not find its MRCA yet.

$$G = \sum_{j=1}^{k} \sum_{i=1}^{L} r_{Gi}$$

$$\rho = rateRE = 2NG$$

Given that a recombination event occurs, a gamete is chosen according to the total rate at potential recombining sites in that gamete. Breakpoint sites are chosen according to the recombination probabilities per site (r_{Gi}). The expected number of recombination events in a panmictic population with constant size is,

$$E(number\ of\ recombination\ events) = \rho \sum_{i=1}^{s-1} \frac{1}{i}$$

Importantly, under recombination, different regions of the alignment might have evolved with different genealogies (Figure 1), which together conform the ancestral recombination graph. The number of genealogies is the number of breakpoints + 1.

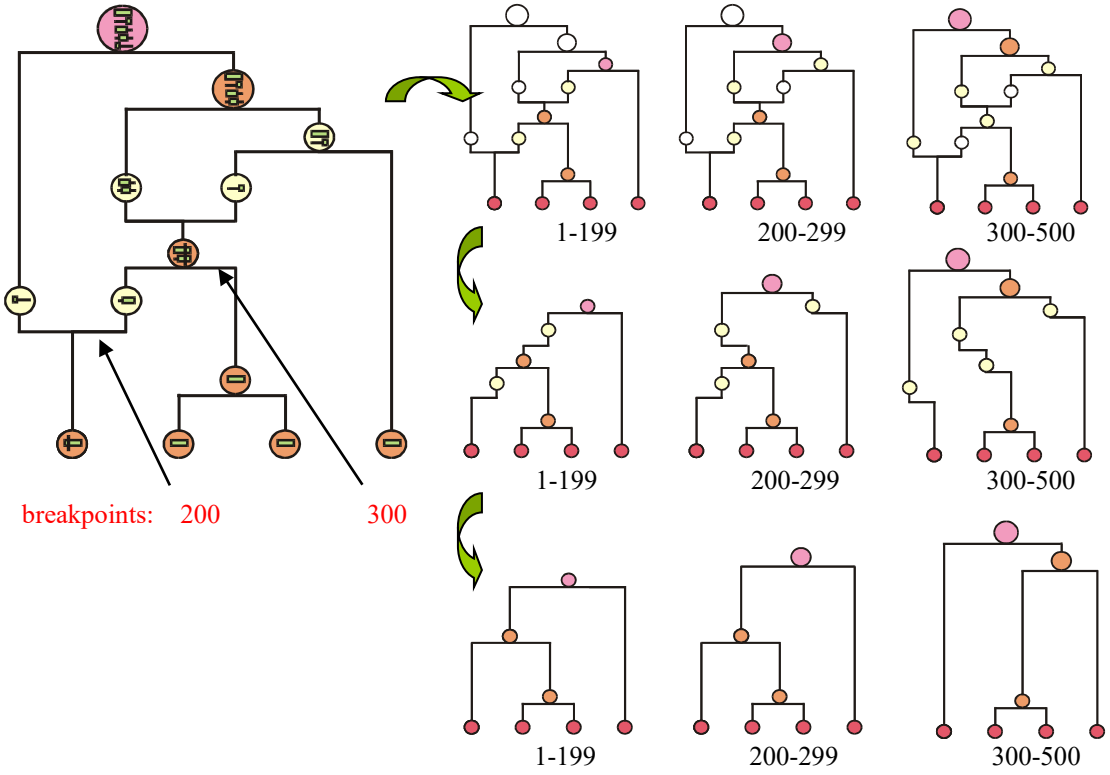


Figure 1. Representation of the ancestral recombination graph and the binary trees embedded. It also shows how nodes with only one ancestral and descendant node are removed.

The simulated ancestral recombination graph (ARG) can be directly exported to an output file in order to visualize and characterize it [42, 58]. It can be printed in branches format [42] (argument "*" in the parameters file).

4.5.1.1 Recombination hotspots

This implementation is based on the framework *SNPsim* [25] and was already implemented in the previous version of *ProteinEvolver* [1]. Given an expected number of recombination hotspots, a background homogenous recombination rate and a hotspot recombination rate, the algorithm starts by choosing the position and number of recombination hotspots for a particular sample. Adding recombination events around the hotspot center results in the specification of a probability distribution for the recombination rate along the region of interest. Other recombination events coming from recombinational hotspots centered outside the region of interest are also considered. This fast simulation results in different recombination rates for different sites along the region (hotspots and coldspots). Next, model for hotspot recombination is described following the algorithm [25].

Hotspot recombination model

The hotspots recombination model aims to represent the idea that some sites in a chromosome are more likely to recombine than others ("recombination hotspots"). This general model is composed of two basic recombination rates and a set of hotspots sites. The background recombination rate (R_B) is the per generation recombination rate at any site in the chromosome of length L, while the hotspot recombination rate (R_H) is the additional per generation recombination rate at the recombination hotspot. X is the number of hotspots.

$$Background\ recombination\ rate\ (R_B) = 4Nr_BL$$
$$Hotspot\ recombination\ rate\ (R_H) = 4Nr_HX$$
$$Global\ recombination\ rate\ (R_G) = R_B + R_H$$

In real life we do not expect the recombination hotspot to be always restricted to the same single site, to have always the same intensity, or to occur independently from other hotspots. The model described above can be generalized to include these relevant biological features. When the hotspot is not restricted to a single site, and under constant population size, the expected number of recombination events is

$$E(number\ of\ recombination\ events) = R_G \sum_{i=1}^{s-1} \frac{1}{i}$$

Hotspot location: interference

We will assume that the distance between hotspots is Gamma distributed, $\Gamma(m, \lambda)$, $m > 0$. If $m=1$, we have a Poisson process with intensity λ. Allowing $m \neq 1$ introduces interference. If $m>1$ hotspots are pushed away from each other; if $0<m<1$ they tend to be clustered (although the algorithm will only accept integer values for m). The average number of hotspots in the gene is λ/m (Figure 2).

Hotspot imprecision

One will consider that the hotspot location actually represents the center of the hotspot, and where recombination is more likely, but there are also some sites around where recombination occurs with some frequency that decreases with increasing distance from the hotspot center. This can be represented by a Normal or a Uniform distribution for the location of the recombination. When the Normal distribution is used, this has a mean equal to the location of the hotspot center and variance called hotspot imprecision (σ^2) (Figure 2). When the variance is small the hotspot tends to be narrow. If the variance is 0, the hotspot is 1 bp wide. When the Uniform distribution is used, recombination events occur with the same probability along a given width for the

hotspot. In addition there is the possibility of recombination events coming from hotspots located outside the region of interest of length L. To implement this idea one can extend the region of interest by a number of sites K at each end. In the Normal distribution an arbitrary, but seemingly reasonable value for K that assures that wide hotspots outside L are taking into account is,

$$K = 10\sqrt{\sigma^2}$$

If the uniform distribution is used, K equals half the width of the hotspot.

Hotspot heterogeneity

In addition, not all hotspots have to be equally "hot". It can model this heterogeneity of recombination rates using a gamma distribution with a mean of 1. The shape of this distribution (α) will determine the strength of this hotspot heterogeneity. The smaller α, the bigger the heterogeneity of recombination rates at the hotspots.

Implementation

A nice feature of the hotspot model is that it allows for the construction of a distribution of recombination rate along the chromosome ($\Re$)(Figure 2) for every sample.

The algorithm starts by building $\Re$. The first step is to set up the number of hotspot centers (X) along the extended region L+2K. The average number of hotspots in the gene is λ/m, and when m>1 it uses a thinning algorithm to locate these hotspots (Figure 2). If m=1 it distributes the hotspots according to a Poisson distribution with intensity λ.

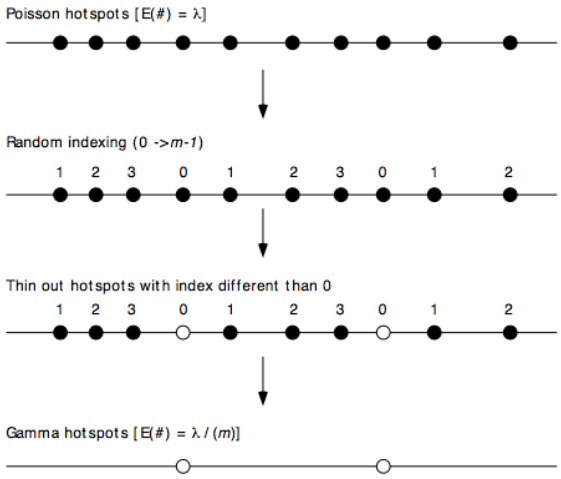


Figure 2. Thinning algorithm for the location of hotspots with interference (m=4). The index for the first hotspot site is chosen uniformly between 0 and m-1. This plot was taken from SNPsim manual [25].

Alternatively, the user can specify a fixed number of hotspots, which will be located uniformly (see argument "q" in recombination hotspots file).

The distribution $\Re$ can be constructed according to a Normal (x_i, σ^2), or a Uniform, (hotspot width) distribution. If there is hotspot recombination, a random gamma variable will scale the recombination events at each hotspot. Figures 3 and 4 represent a realization of this process.

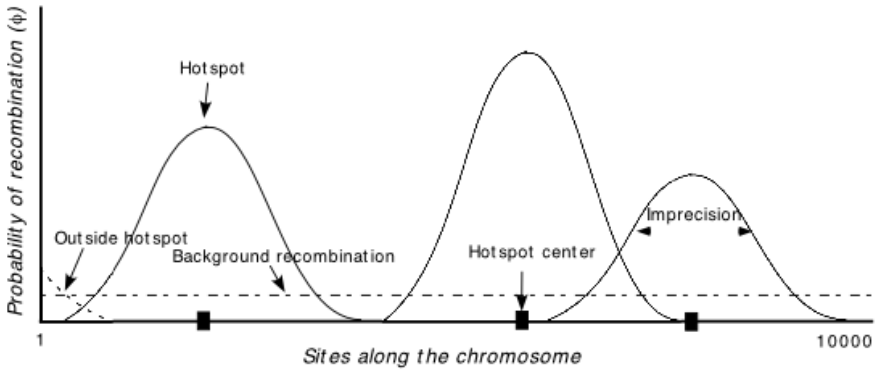


Figure 3. Schematic representation of $\Re$ in the case of three Normal hotspots for the region of interest (L=10000). In this case the hotspot imprecision is quite big. Note that some recombination probability is contributed by a hotspot outside the region of interest. The background recombination is the same for all sites. In this case there is hotspot heterogeneity. This plot was taken from SNPsim manual [25].

We can use now $\Re$ now to set the global recombination rate per each site *i*,

$$r_{Gi} = r_{Bi} + r_{Hi}\varphi_i$$

where φi is the probability of block recombination at site *i*.

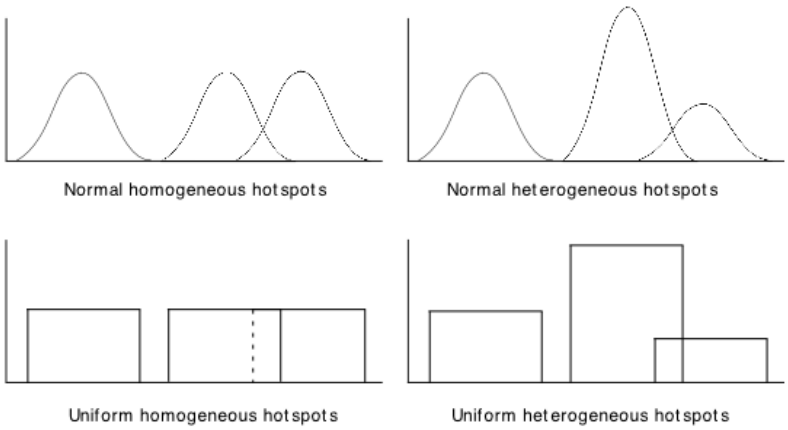


Figure 4. Schematic representation of different $\Re$. This plot was taken from SNPsim manual [25].

Finally, the total recombination rate at all valid recombination sites is considered to compute the times of the recombination events (as described before). Breakpoint sites are chosen according to the recombination probabilities per site (r_{Gi}).

4.5.1.2 Algorithm to evolve molecules with the SCS models along the ARG

In the presence of recombination (coalescent), classic Markov substitution models of evolution independently evolve each site along the evolutionary history of the corresponding recombinant fragment [see for example 31, 57, 59, 60]. Nevertheless, the SCS models require the evolution of the whole molecule because of the dependence

among sites. This implied the development of a new algorithm to evolve whole molecules under recombination. Thus, after building the ARG, the molecular evolution starts by the assignation of a sequence to the GMRCA node. *ProteinEvolver* incorporates an algorithm for the evolution of the whole molecule along the ARG (Figure 5), which is an adaptation of the algorithm developed by Arenas and Posada to simulate intracodon recombination [31]. The process starts from the sequence assigned to the GMRCA node. This sequence can suffer substitutions along the branches and new whole sequences are assigned to the descendant nodes (see steps 1-2 in Figure 5). Later the evolution process can reach a recombinant node (shown in grey, see step 3) and a sequence is assigned to such a recombinant node. However, at that point the evolution stops and comes back to continue by other path. This is forced to occur since there is no information about the sequence at the parental recombinant node. Later, the process reaches the parental recombinant node (step 5), at this point note that there are whole sequences in both recombinant nodes, and there is a combination of the material according to the recombinant breakpoint. Consequently, a new whole sequence is generated (step 6) and this new sequence continues the evolutionary process. Therefore, this algorithm considers the molecular evolution with exchanges of molecular material introduced by recombination and described in the ARG.

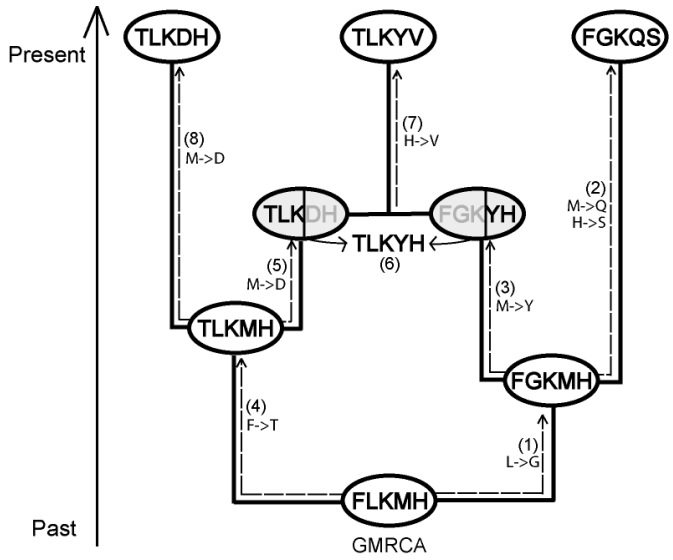


Figure 5. Example of the evolution of a whole molecule along an ancestral recombination graph (ARG).

4.5.2 Migration and convergence of demes

The simplest and most widely used model of population structure is the "finite island model" [26, 28, 61-63], however the stepping-stone [27] and an approximation to the continent-island [28] are also implemented. The migration rate depends on the population size (N), the migration rate per deme (m) and the number of available demes (q).

$$rateMI = 2Nmq$$

Importantly, in the coalescent with migration, lineages can only coalesce with other lineages in the same deme. When a migration event occurs, a given lineage changes from one deme to another (Figure 6).

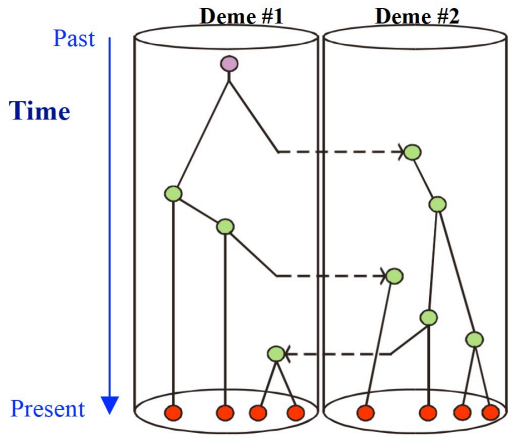


Figure 6. Coalescent with migration for two demes.

4.5.2.1 Island model

The Island Model describes an array of *q* demes, each of constant population size (Figure 7). Migration events can occur between any two demes.

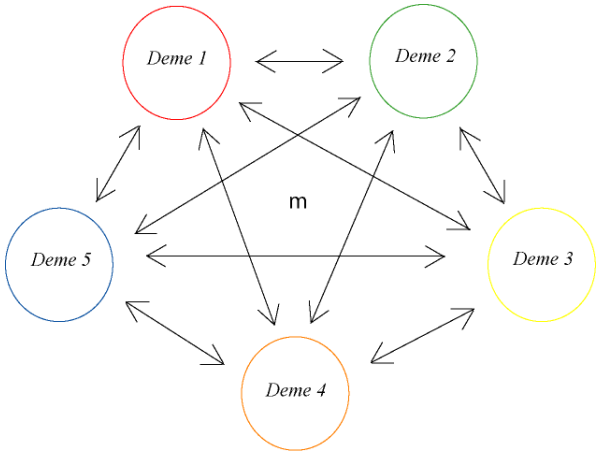


Figure 7. Island model. All the demes have the same population size and the same migration rate with other demes.

4.5.2.2 Stepping-stone model

Under the stepping-stone migration model, migration events occur between neighboring demes (Figure 8).

Figure 8. 1D stepping-stone migration model implemented in ProteinEvolver.

4.5.2.3 Continent-island model

The continent-island model implemented in *ProteinEvolver* allows symmetrical migration between a deme (continent) and other demes (islands) (Figure 9). Note that it does not allow direct migration events between islands.

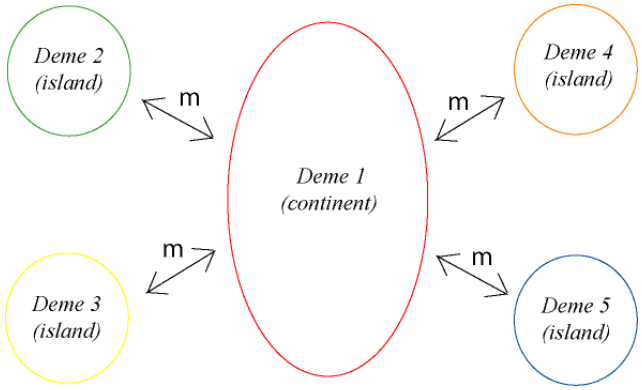


Figure 9. Continent-island model implemented in ProteinEvolver.

4.5.2.4 Temporal variation of the migration rate

ProteinEvolver allows for temporal variation where different migration rates can be applied at different periods of time (Figure 10).

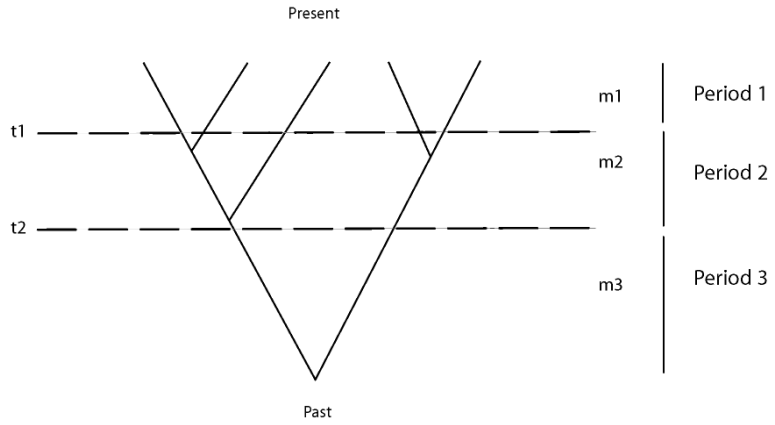


Figure 10. Temporal variation of the migration rate implemented in ProteinEvolver. The figure shows 3 temporal periods separated by two times (t1 and t2) and where each period implements a particular migration rate (m1, m2 or m3).

Convergence of demes

The evolutionary history of demes can be user-specified. Going backwards in time, two demes can be converged at a given time into a new big deme that contains the lineages of the previous demes. The convergence of demes is implemented with the migration models. An example is shown in the Figure 11. It is not possible use the convergence of demes in presence of tip dates (longitudinal sampling) when the time of the convergence of demes is younger than the time of the tip dates.

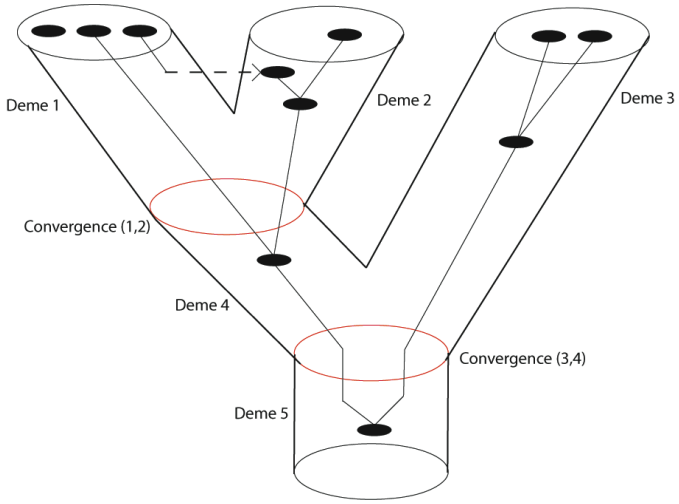


Figure 11. Example of evolutionary history of demes. Going backwards in time, two demes can converge into an ancestral big deme. The convergences are user-specified.

4.5.3 Demography

ProteinEvolver allows the specification of exponential population growth rate. The only modification concerns the expected time to a coalescence, where t is the current time,

$$time\ to\ CA\ \sim\ \frac{\log\left[e^{\beta t}+\beta Exp\left[\frac{k(k-1)}{2}\right]2N\right]}{\beta}-t$$

If the growth rate is negative, the coalescence time may be infinite (i.e., a coalescence event does not occur), and *ProteinEvolver* stops with an error message. Alternatively, the user can define demographic periods (Figure 12). β_i is the growth rate inferred for a demographic period i with population size N_{Bi} in the past to population size N_{Ei} in I_i generations,

$$\beta_i=\frac{-\log\left(\frac{N_{Bi}}{N_{Ei}}\right)}{I_i}$$

The time to a coalescence event is,

$$\text{Time to CA}\sim\frac{\log\left[Exp\left(\frac{k(k-1)}{2}\right)\beta_i 2N_{Ei}e^{-\beta(t-t_{i-1})}+1\right]}{\beta_i}$$

where t is the current time and t_i is the cumulative time from the present,

$$t_i=\sum_{j=0}^{j=i}I_j$$

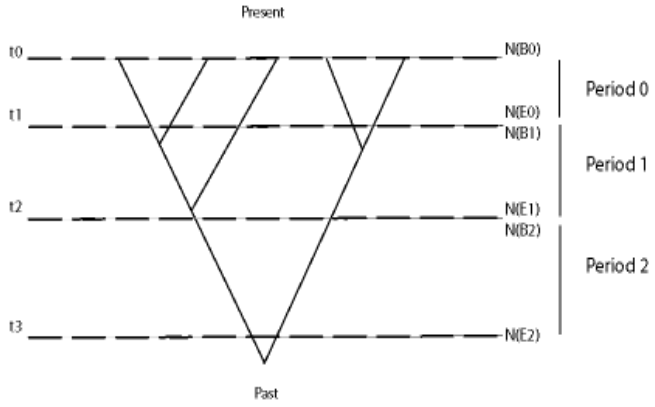


Figure 12. Illustrative example of demographic periods. The population growth rate after the last period of time is assumed to be the same as that one applied in the last period of time.

4.5.4 Tip Dates

Tip dates (longitudinal sampling) can be specified by the user to simulate sequences that belong to different times (samples collected at different times). For this option, the user must introduce a generation time. Figure 13 shows an example of four samples taken at different times, the oldest sample contains only one sequence (seq00001), and the younger sample contains three sequences (seq00004, seq00006 and seq00005).

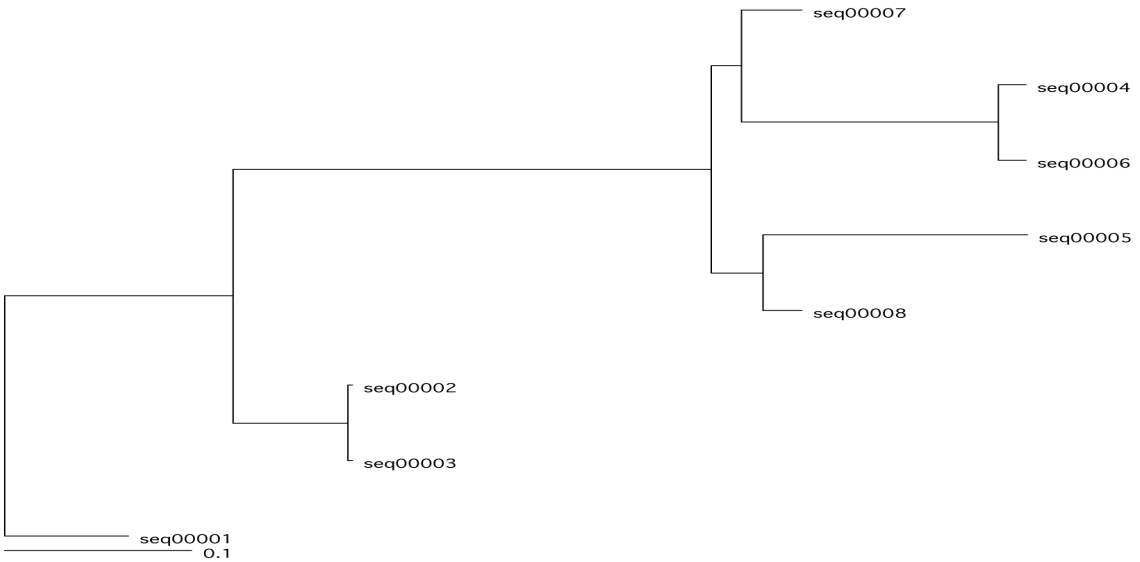


Figure 13. Example of simulated tip dates. Four samples taken at different times with 1, 2, 2 and 3 sequences per sample from the present to the past.

5. Program benchmarking

The models on which *ProteinEvolver* is based have been separately validated.
The balance of evolutionary histories simulated with the population birth-death process was evaluated with the Colless' index I_c [4, 5], which is implemented in the current version of *ProteinEvolver*. I_c is calculated with the sum of the difference in the number

of tips at each side of every node in the tree, standardized by the maximum that such a sum can achieve,

$$I_c = \frac{\sum_{nodes}(N_L - N_R)}{(N-1)(N-2)/2}$$

where N is the total number of nodes in the tree and N_L and N_R are the nodes on the left and on the right, respectively, at every evaluated node. If the tree is properly balanced (only possible when N is a power of 2, e.g. 2, 4, 8, 16, etc.), then $Ic = 0$. If the tree is completely unbalanced, then $Ic = 1$. All phylogenetic trees must have values of Ic between 0 and 1. Phylogenetic trees simulated with *ProteinEvolver* under the birth-death process were, in general (considering the stochasticity of the simulations), balanced.

The method to estimate the protein folding stability has been developed and evaluated in previous works [54, 64, 65] and yields good correlations with experimentally measured free energies. It has been widely used with some variations in many simulations of protein evolution [e.g., 64], since its computational simplicity allows its application over long evolutionary trajectories.

The substitution processes implemented in the SCS models was verified with HYPHY [66] and agreed with the branch lengths. The implemented site-independent DNA substitution models were previously validated [31, 57]. The empirical amino acid substitution models were accurately estimated using *ProtTest* [67].

Coalescent simulations were contrasted with the theoretical expectations for the mean and variances for different values, like the number of recombination and migration events, or the times to the most recent common ancestor [57, 68]. Recombination hotspots agree with theoretical expectations and breakpoints were likely to occur in recombination hotspots regions [25].

6. History

Version 1.0.0 (June 2012)

- Implementation of Pop_evol (code developed by Ugo Bastolla) in ProteinEvolver, a mutational model to evolve DNA sequences accounting for structural protein stability (DNA -> AA -> DNA) and the HKY substitution model.

Version 1.0.1 (July-August 2012)

- pdb from input file.

- Seed for structural protein stability substitution model comes now from the main input file (parameters).

Version 1.0.2 (August 2012)

- Structural protein stability accounting for rates of change among all nucleotide states (GTR and GTnR).

Version 1.0.3 (September 2012)

- Solved bug in the "Compute_load" function. Other warnings from Pop_Evol were solved.

- Solved bug in GTR and GTnR models for PopEvol section (in "Mutate_nuc" and "Compute_load" functions).

- Structural protein stability substitution model crossed with any empirical substitution model (Blosum62, CpRev, Dayhoff, DayhoffDCMUT, HIVb, HIVw, JTT, JonesDCMUT, LG, Mtart, Mtmam, Mtrev24, RtRev, VT, WAG, UserEAAM).

- Validation. User-specified tree was inferred by Hyphy from the simulated alignmets (for both DNA and amino acid data) under the structural protein stability substitution model.

Version 1.0.4 (September 2012)

- Coalescent with recombination is allowed for the simulation under structural protein stability substitution models. When a recombination occurs, the evolution stops at the first parental recombinant node and goes in other direction. Later, when the other parental recombinant node is reached, there is a combination of the material, according to the recombination breakpoint, to generate the material for the descendant node, which continues the evolution.

Recombination can be homogeneous (recombination rate is constant among all sites) or heterogeneous (recombination hotspots, functions developed by David Posada). However, recombination cannot be applied using user-specified input trees and structural protein stability substitution models because part of a protein cannot be evolved under such a substitution model.

Version 1.0.5 (September 2012)

- Included options to print the energy output files per branch produced by the structural protein stability substitution models.

- Print information about the attempted mutations in the structural protein stability substitution models.

Version 1.1.0 (September-December 2012)

- Proportion of invariable sites +I and variable substitution rates per site (heterogeneity) according to a gamma distribution +G, for the structural protein stability substitution models.

- Heterogeneity by a user-defined vector. Each site can evolve under a particular rate user-defined. This is compatible with the specification of +G.

- Print number and mean of the introduced substitution events and, those nonsynonymous for DNA structural protein evolution models.

- Maximum number of characters for the name of taxa is 10.

- More robust generation of the random seed.

- Work with bigger input trees (MAX_LINE).

Version 1.2.0 (September-December 2012)

- Implementation of the neutral evolution model to evolve sequences accounting for the protein structure. Here population size is not required, follow this with the variable "NEUTRAL".

- A bug in the simulation of proteins accounting for structural constraints was fixed. Bug related with the computation of DeltaG.

Version 2.0 (2021-2022)

- Implementation of the standard birth-death model.

Version 2.0.1 (2022)

- Option to fix a given number of sample and/or tip nodes in birth-death simulations.

Version 2.0.2

- Option to prune extinct nodes and lineages from the tree simulated with the birth-death processes.

Version 2.0.3

- Implementation of the tree balance metric, Colless index, to evaluate the simulated birth-death evolutionary histories.

Version 2.1.0

- Reorganization of all the input information. The input information is now much better organized with diverse input files for the desired models of simulation of the evolutionary history.

Version 2.1.1 (2023)

- Birth and death rates can be obtained and applied from the fitness of the protein of every ancestral node.

- Birth and death rates from fitness toward proteins with stability close to the stability of the real protein (SCS models are applied) (option l-1 in BirthDeath.in).

- Birth and death rates from fitness toward more stable proteins (SCS models are applied) (option l-2 in BirthDeath.in).

- Allow variation of site-specific substitution rate according to a user input file for the SCS models.

Version 2.1.2-2.2.0 (2024)

- New input site-specific amino acid matrices and frequencies (i.e., Prot_evol).

- Update of treatment of amino acid frequencies for simplicity.

- Added the FLU empirical substitution model of protein evolution. Tested with ProtTest3, the model is correctly identified.

- Update of input format of site-specific amino acid matrices and frequencies (ReadEAAMfromFile).

- Update of usage information (PrintUsage).

- Cleaning code. Some warnings were detected and removed.

7. Acknowledgments

Several functions were taken from code provided by R. Nielsen and Z. Yang, other functions were provided from *SNPsim* [25] (D. Posada and C. Wiuf) and *Mosaic* (D. Posada). We want the Supercomputing Center of Galicia (CESGA) for the resources to perform computer simulations.

8. References

1. Arenas M, Dos Santos HG, Posada D, Bastolla U: **Protein evolution along phylogenetic histories under structurally constrained substitution models**. *Bioinformatics* 2013, **29**(23):3020-3028.

2. Stadler T: **Sampling-through-time in birth–death trees**. *J Theor Biol* 2010, **267**(3):396-404.
3. Stadler T: **Simulating trees with a fixed number of extant species**. *Syst Biol* 2011, **60**(5):676-684.
4. Colless DH: **Review of Phylogenetics: The Theory and Practice of Phylogenetic Systematics**. *Syst Zool* 1982, **31**(1):100-104.
5. Lemant J, Le Sueur C, Manojlović V, Noble R: **Robust, Universal Tree Balance Indices**. *Syst Biol* 2022, **71**(5):1210-1224.
6. Dang CC, Le QS, Gascuel O, Le VS: **FLU, an amino acid substitution model for influenza proteins**. *BMC Evol Biol* 2010, **10**:99.
7. Darriba D, Taboada GL, Doallo R, Posada D: **ProtTest 3: fast selection of best-fit models of protein evolution**. *Bioinformatics* 2011, **27**(8):1164-1165.
8. Mendez R, Fritsche M, Porto M, Bastolla U: **Mutation bias favors protein folding stability in the evolution of small populations**. *PLoS Comput Biol* 2010, **6**(5):e1000767.
9. Bastolla U, Porto M, Roman HE, Vendruscolo M: **Structural approaches to sequence evolution**. Berlin, Heidelberg: Springer; 2007.
10. Henikoff S, Henikoff JG: **Amino acid substitution matrices from protein blocks**. *Proc Natl Acad Sci U S A* 1992, **89**(22):10915-10919.
11. Adachi J, Waddell PJ, Martin W, Hasegawa M: **Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA**. *J Mol Evol* 2000, **50**(4):348-358.
12. Dayhoff MO, Schwartz RM, Orcutt BC: **A model of evolutionary change in proteins**. In: *Atlas of protein sequence and structure.* Edited by Dayhoff MO, vol. 5, Suppl. 3. Washington D. C.; 1978: 345-352.
13. Kosiol C, Goldman N: **Different versions of the Dayhoff rate matrix**. *Mol Biol Evol* 2005, **22**(2):193-199.
14. Nickle DC, Heath L, Jensen MA, Gilbert PB, Mullins JI, Kosakovsky Pond SL: **HIV-specific probabilistic models of protein evolution**. *PLoS One* 2007, **2**(6):e503.
15. Jones DT, Taylor WR, Thornton JM: **The rapid generation of mutation data matrices from protein sequences**. *Comput Appl Biosci* 1992, **8**(3):275-282.
16. Le SQ, Gascuel O: **An improved general amino acid replacement matrix**. *Mol Biol Evol* 2008, **25**(7):1307-1320.
17. Abascal F, Posada D, Zardoya R: **MtArt: A New Model of Amino Acid Replacement for Arthropoda**. *Mol Biol Evol* 2007, **24**(1):1-5.
18. Yang Z, Nielsen R, Masami H: **Models of amino acid substitution and applications to mitochondrial protein evolution**. *Mol Biol Evol* 1998, **15**(12):1600-1611.
19. Adachi J, Hasegawa M: **MOLPHY version 2.3: programs for molecular phylogenetics based in maximum likelihood**. *Comput Sci Monogr* 1996, **28**:1-150.
20. Dimmic MW, Rest JS, Mindell DP, Goldstein RA: **rtREV: an amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny**. *J Mol Evol* 2002, **55**(1):65-73.
21. Muller T, Vingron M: **Modeling amino acid replacement**. *J Comput Biol* 2000, **7**(6):761-776.
22. Whelan S, Goldman N: **A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach**. *Mol Biol Evol* 2001, **18**(5):691-699.

23. Hudson RR: **Properties of a neutral allele model with intragenic recombination**. *Theor Popul Biol* 1983, **23**(2):183-201.
24. Kingman JFC: **The coalescent**. *Stochastic Processes and their Applications* 1982, **13**:235-248.
25. Wiuf C, Posada D: **A coalescent model of recombination hotspots**. *Genetics* 2003, **164**(1):407-417.
26. Hudson RR: **Island models and the coalescent process**. *Mol Ecol* 1998, **7**(4):413-418.
27. Kimura M, Weiss GH: **The Stepping Stone Model of Population Structure and the Decrease of Genetic Correlation with Distance**. *Genetics* 1964, **49**(4):561-576.
28. Wright S: **Evolution in Mendelian populations**. *Genetics* 1931, **16**:97-159.
29. Navascues M, Depaulis F, Emerson BC: **Combining contemporary and ancient DNA in population genetic and phylogeographical studies**. *Mol Ecol Resour* 2010, **10**(5):760-772.
30. Arenas M, Posada D: **The effect of recombination on the reconstruction of ancestral sequences**. *Genetics* 2010, **184**(4):1133-1139.
31. Arenas M, Posada D: **Coalescent simulation of intracodon recombination**. *Genetics* 2010, **184**(2):429-437.
32. Griffiths RC, Marjoram P: **An ancestral recombination graph**. In: *Progress in population genetics and human evolution.* Edited by Donelly P, Tavaré S, vol. 87. Berlin: Springer-Verlag; 1997: 257-270.
33. Beaumont MA: **Approximate Bayesian Computation in Evolution and Ecology**. *Annu Rev Ecol Evol Syst* 2010, **41**:379-405.
34. Beaumont MA, Zhang W, Balding DJ: **Approximate Bayesian computation in population genetics**. *Genetics* 2002, **162**(4):2025-2035.
35. Jukes TH, Cantor CR: **Evolution of protein molecules**. In: *Mammalian Protein Metabolism.* Edited by Munro HM. New York, NY: Academic Press; 1969: 21-132.
36. Felsenstein J: **Evolutionary trees from DNA sequences: A maximum likelihood approach**. *J Mol Evol* 1981, **17**:368-376.
37. Yang Z: **Among-site rate variation and its impact on phylogenetic analysis**. *Trends Ecol Evol* 1996, **11**(9):367-372.
38. Hudson RR: **Generating samples under a Wright-Fisher neutral model of genetic variation**. *Bioinformatics* 2002, **18**(2):337-338.
39. Rambaut A, Grassly NC: **Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees**. *Comput Appl Biosciences* 1997, **13**(3):235-238.
40. Yang Z: **PAML: a program package for phylogenetic analysis by maximum likelihood**. *Comput Appl Biosciences* 1997, **13**(5):555-556.
41. Yang Z: **PAML 4: phylogenetic analysis by maximum likelihood**. *Mol Biol Evol* 2007, **24**(8):1586-1591.
42. Arenas M, Patricio M, Posada D, Valiente G: **Characterization of phylogenetic networks with NetTest**. *BMC Bioinformatics* 2010, **11**(1):268.
43. Goldstein RA: **Population Size Dependence of Fitness Effect Distribution and Substitution Rate Probed by Biophysical Model of Protein Thermostability**. *Genome Biol Evol* 2013, **5**(9):1584-1593.
44. Tavaré S: **Some probabilistic and statistical problems in the analysis of DNA sequences**. In: *Some mathematical questions in biology - DNA sequence*

analysis. Edited by Miura RM, vol. 17. Providence, RI: Amer Math Soc; 1986: 57-86.

45. Kimura M: **A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences**. *J Mol Evol* 1980, **16**(2):111-120.
46. Hasegawa M, Kishino K, Yano T: **Dating the human-ape splitting by a molecular clock of mitochondrial DNA**. *J Mol Evol* 1985, **22**:160-174.
47. Ewens WJ: **Mathematical Population Genetics**, vol. 9. Berlin: Springer-Verlag; 1979.
48. Sella G, Hirsh AE: **The application of statistical physics to evolutionary biology**. *Proc Natl Acad Sci USA* 2005, **102**(27):9541-9546.
49. Vendruscolo M, Kussell E, Domany E: **Recovery of protein structure from contact maps**. *Fold Des* 1997, **2**(5):295-306.
50. Bastolla U, Roman HE, Vendruscolo M: **Neutral evolution of model proteins: diffusion in sequence space and overdispersion**. *J Theor Biol* 1999, **200**(1):49-64.
51. Guerois R, Nielsen JE, Serrano L: **Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations**. *J Mol Biol* 2002, **320**(2):369-387.
52. Berezovsky IN, Zeldovich KB, Shakhnovich EI: **Positive and negative design in stability and thermal adaptation of natural proteins**. *PLoS Comput Biol* 2007, **3**(3):e52.
53. Noivirt-Brik O, Horovitz A, Unger R: **Trade-off between positive and negative design of protein stability: from lattice models to real proteins**. *PLoS Comput Biol* 2009, **5**(12):e1000592.
54. Minning J, Porto M, Bastolla U: **Detecting selection for negative design in proteins through an improved model of the misfolded state**. *Proteins* 2013.
55. Derrida B: **Random Energy Model: An exactly solvable model of disordered systems**. *Phys Rev B* 1981, **24**:2613-2626.
56. Goldstein RA: **The evolution and evolutionary consequences of marginal thermostability in proteins**. *Proteins* 2011, **79**(5):1396-1407.
57. Arenas M, Posada D: **Recodon: coalescent simulation of coding DNA sequences with recombination, migration and demography**. *BMC Bioinformatics* 2007, **8**:458.
58. Arenas M, Valiente G, Posada D: **Characterization of reticulate networks based on the coalescent with recombination**. *Mol Biol Evol* 2008, **25**(12):2517-2520.
59. Arenas M: **Simulation of Molecular Data under Diverse Evolutionary Scenarios**. *PLoS Comput Biol* 2012, **8**(5):e1002495.
60. Fletcher W, Yang Z: **INDELible: a flexible simulator of biological sequence evolution**. *Mol Biol Evol* 2009, **26**(8):1879-1888.
61. Latter BD: **The island model of population differentiation: a general solution**. *Genetics* 1973, **73**(1):147-157.
62. Maruyama T: **A simple proof that certain quantities are independent of the geographical structure of population**. *Theor Popul Biol* 1974, **5**(2):148-154.
63. Matsen FA, Wakeley J: **Convergence to the island-model coalescent process in populations with restricted migration**. *Genetics* 2006, **172**(1):701-708.
64. Bastolla U, Farwer J, Knapp EW, Vendruscolo M: **How to guarantee optimal stability for most representative structures in the Protein Data Bank**. *Proteins* 2001, **44**(2):79-96.

65. Bastolla U, Demetrius L: **Stability constraints and protein evolution: the role of chain length, composition and disulfide bonds**. *Protein Eng Des Sel* 2005, **18**(9):405-415.
66. Kosakovsky Pond SL, Frost SD, Muse SV: **HYPHY: Hypothesis testing using phylogenies**. *Bioinformatics* 2005, **21**:676-679.
67. Abascal F, Zardoya R, Posada D: **ProtTest: selection of best-fit models of protein evolution**. *Bioinformatics* 2005, **21**(9):2104-2105.
68. Hudson RR: **Gene genealogies and the coalescent process**. *Oxf Surv Evol Biol* 1990, **7**:1-44.