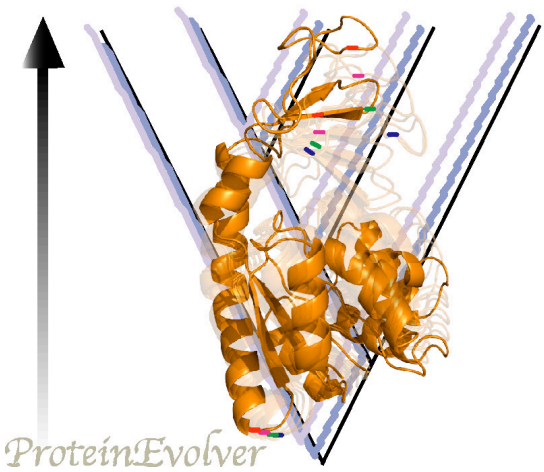
Documentation for **ProteinEvolver2**

Modeling protein evolution forward in time under empirical and structurally constrained substitution models with birth-death population processes based on the fitness of temporal variants

Current version is 2.1.1



Contents

Disclaimer ___ ***3***

Credits and Funding ___ ***3***

1. Purpose ___ ***3***

2. Executable files and compilation ___ ***4***

3. ProteinEvolver2 Usage ___ ***5***

3.1. Parameters file ___ **5**
3.1.1. Arguments for the parameters (main) file ___ 5

3.2. Additional input files ___ **8**
3.2.1. Birth-death simulations ___ 8
3.2.2. Coalescent simulations ___ 9
3.2.3. User-specified tree/s ___ 12
3.2.4. Structurally constrained substitution models ___ 12
3.3.5. Recombination hotspots ___ 14
3.3.6. Root sequence specified by the user ___ 15
3.3.7. Empirical user-specified amino acid matrix ___ 15
3.3.8. Site by site variable substitution rate ___ 15

3.4. Output Files ___ **21**
3.4.1. Output files for the SCS models ___ 21

3.5. Solving message errors ___ **22**

4. ProteinEvolver2 model ___ ***23***

4.1. Forward in time modeling of protein evolution accounting for the protein fitness and under a birth-death population process ___ **23**

4.2. Coalescent simulations ___ **24**
4.2.1. Recombination ___ 25
4.2.2 Migration ___ 30
4.2.3 Convergence of demes ___ 32
4.2.4 Demography ___ 33
4.2.5 Tip Dates ___ 34

4.3. Markov substitution models ___ **34**
4.3.1. DNA models ___ 35
4.3.2. Empirical amino acid models ___ 35

4.4. Structurally constrained substitution models ___ **35**
4.4.1 Crossing the SCS models with classic Markov substitution models ___ 37
4.4.2 Evolution of sequences along evolutionary histories under the SCS models ___ 38

5. Program benchmarking ___ ***39***

6. History ___ ***39***

7. Acknowledgments ___ ***40***

8. References ___ ***40***

Disclaimer

Credits and Funding

This program was developed at the University of Vigo, Spain.
This work was supported by the Spanish Ministry of Economy and Competitiveness and Ministry of Science and Innovation through the Grants RYC-2015-18241 and PID2019-107931GA-I00/AEI/10.13039/501100011033.

1. Purpose

ProteinEvolver2 generates samples of protein sequences evolved forward in time under empirical and structurally constrained substitution (SCS) models and upon evolutionary histories previously simulated with the birth-death population process based on the fitness of every variant at any time.
Additionally the program includes the simulation of protein evolution under the cited models and upon coalescent (modified with recombination, migration, demographics and longitudinal sampling, among other options), and user-specified phylogenetic histories.

What is exactly implemented in ProteinEvolver?
ProteinEvolver2 considers the folding stability of every protein variant to calculate its fitness, then that fitness is considered in a birth-death population process [1-3] to simulate the evolutionary history of that variant forward in time until the following node or generation. After that, the protein variant is evolved forward in time (until the following node or generation) under a substitution model of evolution.
ProteinEvolver2 implements protein stability substitution models that consider diverse thermodynamic parameters [4, 5]. In addition, these structural models can be crossed with parametric DNA substitution models such as *JC* [6], *K80* [7], *F81* [8], *HKY* [9], *SYM* [10], *GTR* [11] and even *GTnR* [extended from, 11] for the simulation of DNA data or, with empirical amino acid substitution models such as *Blosum62* [12], *CpRev* [13], *Dayhoff* [14], *DayhoffDCMUT* [15], *HIVb* [16], *HIVw* [16], *JTT* [17], *JonesDCMUT* [15], *LG* [18], *Mtart* [19], *Mtmam* [20], *Mtrev24* [21], *RtRev* [22], *VT* [23], *WAG* [24] or any user-specified matrix, for the simulation of proteins. The empirical substitution models are also included without using structural constraints. Next, heterogeneous substitution rates among sites by a gamma distribution (+G) and proportion of invariable sites (+I) are also implemented. Furthermore, the user can modify the substitution rate for each site, for example allowing fix particular sites (for example, catalytic sites).

Therefore, under the birth-death models implemented in *ProteinEvolver2*, both the phylogeny and the molecular evolution are simulated simultaneously and forward in time. The user can either specify a birth-death process (several birth-death processes are implemented, details later) [1, 2], a particular phylogenetic tree or, simulate a coalescent history [25, 26]. Concerning the birth-death process, the program implements several models where the user can just specify the birth and death rates or the birth and death rates are directly obtained from the fitness (based on the folding stability) of a protein variant. Concerning the coalescent, the program implements the coalescent with recombination [25] (which can be homogeneous or heterogeneous along the sequence [following, 27]), variable population size (by a growth rate and demographic periods), a variety of migration models (such as island [28], stepping-stone [29] and continent-island [30]) with temporal variation of migration rates and convergence of demes or subpopulations, the simulation of haploid or diploid data, and longitudinal sampling [see, 31].
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 assigned to the root of the evolutionary history. Then, a variety of input evolutionary parameters should be specified (see next sections).
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, 32, 33]) can be also 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) [34] can be printed.
We recommend check the attached folder "example_input_files", because it contains a variety of examples of evolutionary scenarios that can be simulated using *ProteinEvolver2*.

2. Executable files 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 type (it may take a few minutes): *make all*
It should print something like:
Building ProteinEvolver2 version 2.1.1
gcc -c -O3 -Wall ProteinEvolver2.1.1.c
gcc -lm -O3 -Wall -o ProteinEvolver2.1.1 ProteinEvolver2.1.1.o
Finished compiling.
A second makefile "Makefile_MPI" is provided to compile a MPI version (which could be convenient for diverse simulation experiments [e.g., 35, 36]). 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 host for running *ProteinEvolver2* in parallel. The minimum number of processors is two. An example of execution for 3 processors is the next: *mpirun -np 3 ProteinEvolver2.1.1*

3. ProteinEvolver2 Usage

The input of the program consists of a series of arguments and parameter values (Table 1) that can be specified in a text file called "parameters" that should be located at the same directory of the executable. These arguments include the parameter values used in the simulations and several printing options that control the amount of information that is sent to the console or the type of output files.
In addition, other input files are required for diverse specifications (see later):

- Birth-death population process
- Coalescent process
- User-specified tree/s
- Protein stability substitution models: including a PDB file and amino acid contacts matrix
- A user- specified empirical amino acid model
- Substitution rate per site
- A user-specified sequence for the root of the phylogeny
- Recombination hotspots

3.1. 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 this file anything within brackets will be ignored. Examples of the "*parameters*" file are included in the distribution. Then, only type:

./ProteinEvolver2.1.1

3.1.1. Arguments for the parameters (main) file

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
The user has to specify either a birth-death simulation, a coalescent simulation or provide an input tree. Only one of them can be specified.

lNAME : Birth-death simulation
Indicate the name of the input file with the parameters for the birth-death model. Example: *lBirthDeath.in*

sNAME : Coalescent simulation
Indicate the name of the input file with the parameters for the coalescent model. Example: *sCoalescent.in*

pNAME : Input tree/s file
This option activates a user-specified tree, which can be applied for the molecular evolution. Indicate the name of the input file with the input tree/s.
Example: *ptreefile.in*

Substitution model
f4 # # # #, f20 #: Nucleotide frequencies or amino acid frequencies
The nucleotide frequencies A C G T are specified in this order.
The amino acid frequencies are specified in the order: A R N D C Q E G H I L K M F P S T W Y V.
For nucleotide models the first number must be 4 while for amino acid models it must be 20. By default all frequencies are equal. Four frequencies example: *-f4 0.4 0.2 0.1 0.3. Twenty* frequencies example: *f20 0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.06*

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*" [6] or "*F81*" [8] 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 : SCS for coding DNA simulation
This argument should be followed by the name of the file with settings for the SCS models (see later). 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 later). 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*

@NAME : empirical amino acid model
This option allows for the simulation of proteins. The empirical amino acid models implemented in *ProteinEvolver2* are the following: *Blosum62* [12], *CpRev* [13], *Dayhoff* [14], *DayhoffDCMUT* [15], *HIVb* [16], *HIVw* [16], *JTT* [17], *JonesDCMUT* [15], *LG* [18], *Mtart* [19], *Mtmam* [20], *Mtrev24* [21], *RtRev* [22], *VT* [23], *WAG* [24]; in this case just specify the name of the empirical amino acid model.

But in addition, it is possible to specify a user-defined empirical amino acid model by an input file (see later); in this case just specify the name of the input file with the empirical amino acid model. Example: @JTT. Example: @UserEAAM.

zNAME : SCS model for protein simulation
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*

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 site by site
After this argument the user should specify the name of an input file, which contains a vector with values for each site (see details later). Example: _HetRatesVector

xNAME : input file with sequence of the root node
The user can specify its own root (MRCA/GMRCA) sequence in a text file (see later). 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 other arguments (i.e., "*s in birth-death and coalescent*" or in the tree file "*-p in input tree file*"). By default if not specified, the *root sequence* is simulated from the nucleotide or amino acid frequencies (see above, *-f* argument). Example: *xseqRoot.in*

Output settings
bNAME : print sequences
When this argument is invoked, 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). Example: *c2 1 0*

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

jNAME : print genealogies
When this option is specified, genealogies for each recombinant fragment 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 coalescent times for each genealogy will be printed to the specified text file, in the *Results* folder. This option will slow down the simulations. Example: *ktimes*

**NAME : print ARGs*
When this argument is specified the ancestral recombination graph (ARG) will be printed to the specified text file in the *Results* folder. Then, this file can be directly introduced into the *NetTest* web server (http://darwin.uvigo.es/software/nettest/) [38] in order to visualize the ARG. Example: **NetworkFile*

dNAME : print breakpoints
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. When a seed is fixed the process can be always reproduced if same settings are specified. Example: *#386658297*

y# : noisy level
This option controls the level of information that will be printed to the screen.
0 : does not print run information, just the simulation progress.
1 : run settings and run information summarizing the simulations.
2 : calculation status, initial demes and event information, run settings for each replicate
3 : print ancestral status for each sequence at each event + *MRCA* status, molecular evolution.
4 : print information about all the evolutionary events.
Note that higher levels of noisy will slow down the simulations. The default level is 1 Example: *y1*

3.2. Additional input files

The following input files can be optional and if required, must be introduced in the same directory of the executable of *ProteinEvolver*. Indeed, note that the settings contained in these input files cannot be specified in the command line. The name of these files must be introduced in the main settings (parameters file or command line).

3.2.1. Birth-death simulations

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

l# (# #): Birth and death rates

The birth and death rates can be indicated by three ways:

Direct specification of the birth rate and death rate, respectively: Example *l0.2 0.1*

Birth and death rates based on the fitness of the corresponding protein variant where a fitness more similar to the real fitness (PDB) leads to a higher birth rate and lower death rate, in this case specify -1. Example *l-1*

Birth and death rates based on the fitness of the corresponding protein variant where a higher fitness leads to a higher birth rate and lower death rate, in this case specify -2. Example *l-2*

+# (#) (#): Criterion for ending the birth and death process

Two criteria for ending the birth and death process are implemented:

A given sample size is reached, use 1. Here there are three options:

Reaching a sample size that can be smaller if there are extinction events, use 1. In the example sample size is 10 nodes. *Example +1 1 10*

Reaching a forced sample size that includes extinct nodes, use 2. In the example sample size is 10 nodes. *Example +1 2 10*

Reaching a forced sample size that includes only tip nodes (without extinct nodes), use 3. In the example sample size is 10 nodes. *Example +1 3 10*

A given time is reached, use 2. In the example, 7.1 is the time in *2N* units, where N is the effective population size. *Example +2 7.1*

>#: Prune extinct nodes and lineages of the birth-death tree

If 1 extinct nodes are not pruned, if 2 extinct nodes are pruned. Example (no pruned): *>1*

o# : outgroup branch length

If this option is specified the program simulates an outgroup sequence that evolves directly and independently from the root, along a branch of the specified length. By default the outgroup is not simulated. Example: *o0.1*

u# : Substitution rate

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

e# #: Effective population size; Haploid / Diploid

Optional. The effective size (*N*) of the population from.

The second number means that the data set can be simulated as haploid (1) or diploid (2). Example: *e100 1*

s#: Number of sites

The total sequence length (in nucleotides or amino acids). Example: *s255*

3.2.2. Coalescent simulations

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

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: *e100 1*

=# : 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 [39]. 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 later). 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: *r2e-6* (= *r0.000002*)

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 recombination hotspots. Parameters for recombination hotspots must be introduced in the file here indicated (see later). Example: *hUserHetRec*

u# : Substitution rate

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

o# : outgroup branch length

If this option is specified the program simulates an outgroup sequence that evolves directly and independently from the root, along a branch of the specified length. By default the outgroup is not simulated. Example: *o0.1*

3.2.3. User-specified tree/s

Alternatively to the birth-death and coalescent simulations, the user can specify a tree in order to evolve sequences along its branches. This possibility is convenient when the user already has a tree (e.g., inferred from real data). This procedure is similar to other software such as *SeqGen* [40] or *Evolver* [41] (although note that models implemented in these programs do not consider protein structures).
The tree must be incorporated into an input file, which should be specified from the main settings by the argument "*-p*" (e.g., *-ptreefile*).
Then, 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 sites should be covered by a phylogenetic tree, so it always should start by 1 and finish by the total number of sites. Only one tree is allowed because the whole protein must exist in all nodes. The file does not allow empty lines. Trees should be rooted.
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. In that example, a total of 255 sites with 5 taxas (taxonA, taxonB, taxonC, taxonD, taxonE). When this option is applied all birth-death and coalescent input settings are ignored.

3.2.4. Structurally constrained substitution models

The main input file to specify 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 proteins). Examples of the main input file for these models can be found in the "examples" folder, with name "*Pop_evol.in*". Theory is described later but follows [5].
Importantly, this simulation requires that the MRCA/GMRCA 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 MRCA/GMRCA sequence by an input file (see argument "*-x*" in the parameters file)** because a MRCA/GMRCA sequence computed using the nucleotide or amino acid frequencies could not satisfy such a condition.

Possible arguments 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 component in REM calculation (to compute the structural protein energy) must be specified (see later) [42]. 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 (see section 3.6.1). "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*

3.3.5. Recombination hotspots

The user can optionally simulate recombination hotspots if the coalescent is specified following [5], based on the algorithm implemented in SNPsim [27] when using coalescent simulations. This file must be specified in the coalescent settings (*hNAME*) and should be placed in the directory of the executable. An example file "*UserHetRec*" is included in the folder of examples.
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 smallest 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*

3.3.6. Root sequence specified by the user

By default, the root (GMRCA or MRCA) sequence 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.
The option is highly recommended for protein stability substitution 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) for the amino acid sequence of the PDB file. **Consequently, under SCS models, it is highly recommended introduce directly (by using this seqRoot.in file) the amino acid sequence of the PDB file or a coding sequence that codifies for such a PDB sequence**.

3.3.7. Empirical user-specified amino acid matrix

The user can optionally specify a particular empirical amino acid matrix from a text file, which has to be located in the directory of the executable. An example file "*userEAAM*" is included in the folder with examples.
This file consists in two sets of parameter values: First, the substitution rates for each amino acid (values must start after the letter for the corresponding amino acid). Second, 20 amino acid frequencies can be introduced (values must start after a "z"). The order of amino acids must be the following: A R N D C Q E G H I L K M F P S T W Y V.

3.3.8. Site by site variable substitution rate

The user can optionally modify the substitution rate of every site. This file must be specified in the main settings by "*_NAME*" and should be located in the directory of the executable. An example file is included with the package (file named "*HetRatesVector.in*") and another example is shown below.
The file consists on two sets of numbers specified after an "*r*". First, the sequence length (255 in the example). Second, values for each site separated by a single space. Note that the number of values 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 value for each site consists in a factor that multiplies the original substitution rate for that site and should be between 1 and 0. Thus, for example a value of 1 means that the original substitution rate remains invariant while a value of 0 means that the substitution rate is 0 and therefore that site will never mutate. This option can be useful when the user already know the functional importance of the amino acids in the protein, so for example catalytic sites could have a value of 0 in order to keep the protein activity.

Table 1. Key arguments for *ProteinEvolver2*. The user can specify diverse parameters to define different simulation scenarios.

Parameter	Arg	Example value	Input file	Application
Number of replicates	n	200	Parameters	All
Birth-death model	l	-	Parameters	Evolutionary history
Coalescent model	s	-	Parameters	Evolutionary history
User-specified tree/s	p	Treefile	Parameters	Evolutionary history
Nucleotide frequencies	f	0.4 0.3 0.1 0.2	Parameters	DNA evolution
Transition / transversion ratio	v	2.1	Parameters	DNA evolution
Relative symmetrical substitution rates	v	1.0 2.3 2.1 3.0 4.2 1.0	Parameters	DNA evolution
Relative asymmetrical substitution rates	v	0.1 0.2 0.3 0.4 0.9 0.6 0.9 0.8 0.9 0.1 0.2 0.3	Parameters	DNA evolution
File for the coding DNA – SCS[5] models	m	Pop_evol.in	Parameters	DNA evolution
Amino acid frequencies	f	0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.06	Parameters	Protein evolution
Empirical amino acid model	@	JTT	Parameters	Protein evolution
File for the Protein – SCS[5] models	z	Pop_evol.in	Parameters	Protein evolution
Rate variation among sites[4]	a	0.4	Parameters	DNA/protein evolution
Variable substitution rates site by site	_	HetRatesVector.in	Parameters	DNA/protein evolution
Proportion of invariable sites	i	0.2	Parameters	DNA/protein evolution
User-defined sequence for the root node	x	seqRoot.in	Parameters	DNA/protein evolution

Print sequences using diverse options	b, c	sequences, 1 1 0	Parameters	Print DNA/protein sequences
Print root sequence	$	-	Parameters	Print DNA/protein sequences
Print simulated trees	j	trees	Parameters	Print trees
Print simulated ARG	*	NetworkFile	Parameters	Print network
Print times for genealogies	k	times	Parameters	Print times of ancestral nodes
Print breakpoints	d	breakpoints	Parameters	Print recombination breakpoints
Apply a seed	#	3444556	Parameters	Seed
Level of output information printed in the screen	y	2	Parameters	Printed output files
Birth and death rates (user-specified, higher protein stability, realistic protein stability)	l	0.2 0.1 -1 -2	BirthDeath.in	Evolutionary history
Ending of birth-death history (sample size, forward time)	+	1 1 10 2 7.5	BirthDeath.in	Evolutionary history
Prune extinct nodes of birth-death tree	>	1	BirthDeath.in	Evolutionary history
Outgroup branch length	o	0.1	BirthDeath.in	Evolutionary history
Substitution rate	u	5.1×10^{-4}	BirthDeath.in	Evolutionary history
Effective population size – Haploid/diploid organism	e	1000 1	BirthDeath.in	Evolutionary history
Number of sites (nucleotides or amino acids)	s	255	BirthDeath.in	Evolutionary history
Sample size – Number of sites (nucleotides or amino acids)	s	8 255	Coalescent.in	Evolutionary history
Effective population size – Haploid/diploid organism	e	1000 1	Coalescent.in	Evolutionary history
Tip dates	=	2 1995 1 3 2003 4 8	Coalescent.in	Evolutionary history[1]
Generation Time	/	400	Coalescent.in	Evolutionary history
Exponential growth	g	0 $2.1\text{x}10^{-5}$	Coalescent.in	Evolutionary history

rate				
Demographic periods[2]	g	1 1000 5000 200	Coalescent.in	Evolutionary history
Migration model	q	1	Coalescent.in	Evolutionary history
Number of demes	q	"4" 2 2 3 1	Coalescent.in	Evolutionary history
Population structure	q	4 "2 2 3 1"	Coalescent.in	Evolutionary history
Migration rate (constant or variable with time)	t	1.2×10^{-4} or 100 "0.001 0.005"	Coalescent.in	Evolutionary history
Convergence of demes[3]	%	1 2 5000	Coalescent.in	Evolutionary history
Homogeneous recombination rate	r	5×10^{-6}	Coalescent.in	Evolutionary history
Fixed number of recombination events	w	3	Coalescent.in	Evolutionary history
Recombination hotspots	h	UserHetRec .in	Coalescent.in	Evolutionary history
Substitution rate	u	5.1×10^{-4}	Coalescent.in	Evolutionary history
Outgroup branch length	o	0.1	Coalescent.in	Evolutionary history
User-specified tree/s	-	1 sequence length, tree in newick format	treefile.in	Evolutionary history
Protein structure (file)	-	1TRE.pdb	Pop_evol.in	Protein evolution (models that consider the protein structure)
Chain of the protein structure	-	A	Pop_evol.in	Protein evolution (models that consider the protein structure)
Contacts matrix (file)	-	structures.in	Pop_evol.in	Protein evolution (models that consider the protein structure)
Thermodynamic temperature	-	1.8	Pop_evol.in	Protein evolution (models that consider the protein structure)
Configurational entropy S0	-	0.05	Pop_evol.in	Protein evolution (models that consider the protein structure)
Configurational entropy SC1	-	0.05	Pop_evol.in	Protein evolution (models that consider the protein structure)
Configurational entropy SC0	-	0.0	Pop_evol.in	Protein evolution (models that consider the protein structure)

REM3 component	-	0	Pop_evol.in	Protein evolution (models that consider the protein structure)
Neutral/fitness structure model	-	1	Pop_evol.in	Protein evolution (models that consider the protein structure)
Population size for protein evolution (NPOP)	-	10	Pop_evol.in	Protein evolution (models that consider the protein structure)
Type of branch lengths (mutations 1, substitutions 2)	-	2	Pop_evol.in	Protein evolution (models that consider the protein structure)
Level of printed output files (0-2)	-	2	Pop_evol.in	Protein evolution (models that consider the protein structure)
Protein structure file	-	-	File.pdb	Protein evolution (models that consider the protein structure)
Contacts matrix file	-	-	structures.in	Protein evolution (models that consider the protein structure)
Sequence for the root node	-	-	seqRoot.in	DNA/protein evolution
Heterogeneous substitution rate among sites	r	-	HetRatesVector.in	DNA/protein evolution
User-specified exchangeability matrix	-	-	UserEAAM	Protein evolution (user-specified)
Hotspot recombination rate	k	1e-4	UserHetRec.in	Evolutionary history
Expected # Poisson hotspots	h	1.1	UserHetRec.in	Evolutionary history
Fixed #hotspots	q	4	UserHetRec.in	Evolutionary history
Hotspot imprecision	v	0	UserHetRec.in	Evolutionary history
Hotspot width	m	1	UserHetRec.in	Evolutionary history
Hotspot heterogeneity	t	0.5	UserHetRec.in	Evolutionary history
Hotspot interference	z	1	UserHetRec.in	Evolutionary history

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

3.4. Output Files

All output files produced by the program are written to the folder "*Results*":

- *Sequences file*: contains the aligned sequences in *Phylip* sequential, *Fasta* or *Nexus* formats.
- *Trees file*: contains the simulated tree/s in newick format.
- *Times file*: contains information about time and branch length for each tree.
- *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.
- *CatMRCA/GMRCA*: contains the corresponding sequences of the root node, catMRCA (concatenated MRCA fragments) and GMRCA (sequence of the GMRCA node) in the ARG.
- *Breakpoints file (in case of coalescent)*: contains the breakpoints ordered by event time. Disabled when input tree/s are user-specified.
- *Network file (in case of coalescent)*: contains the simulated ARG by a branches format (each line contains two connected nodes) [38]. Disabled when input tree/s are user-specified.

3.4.1. Output files for 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, it could require a lot of space in the hard disk and might slow down the simulations. See the argument *OUTPUT_LEVEL* in the section 3.3.1 to control the amount of printed output files.
The name for each output file consist in 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 coding DNA evolution,
ProteinStability_Replicate1_Branch0_1TREA_DeltaG_2.dat
ProteinStability_Replicate1_Branch0_1TREA_T1.80_S00.05_N10_GC0.40_ave.dat
ProteinStability_Replicate1_Branch0_1TREA_T1.80_S00.05_N10_GC0.40_dna.dat
ProteinStability_Replicate1_Branch0_1TREA_T1.80_S00.05_N10_GC0.40_stab.dat
ProteinStability_Replicate1_Branch0_1TREA_T1.80_S00.05_N10_GC0.40_final.dat

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.

3.5. Solving message errors

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

- In case of using the birth-death population process one should consider a reasonable ending point of the forward in time simulation, such as finishing the birth-death process without a too large sample size or a too long forward period of time to avoid not enough RAM memory.

- The simulation under structurally constrained substitution models also requires a lot of RAM memory, especially if the protein structure (PDB file) is large. In case of finding this problem simulate only one replicate at every execution and/or reduce the sample size (number of simulated sequences).

- There is a common message of error due to not enough RAM memory in scenarios of the coalescent with very high recombination rates. 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 (scenarios of the coalescent) 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 (scenarios of the coalescent) 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.

If you experience any unexpected error or there is any doubt, please do not hesitate to contact us: marenas@uvigo.es. Thanks for your contribution!

4. ProteinEvolver2 model

ProteinEvolver2 implements a variety of models to simulate protein and DNA sequences, although it is mainly focused on protein sequences. First, the evolutionary history can be simulated using the population birth-death process, the coalescent process or can just be given by the user. After that, sequence evolution can be simulated upon that evolutionary history to obtain the sequences of the sample, a multiple sequence alignment.

4.1. Forward in time modeling of protein evolution accounting for the protein fitness and under a birth-death population process

The main goal of *ProteinEvolver2* is the evolution of proteins forward in time considering their own fitness, as occur in nature, though a birth-death process. Given a protein variant (sequence), the program calculates the protein folding stability (ΔG) following methods developed in a previous work [5]. Next, the folding stability can be used to calculate the fitness f, considering a thermodynamic temperature T, by a Boltzmann distribution,

$$f=1/(1+\exp(\Delta G/kT))$$, following [43, 44].

Next, after applying an evolutionary event (i.e., an amino acid substitution), a *Moran* process [45, 46] 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 (*a* is 1 or 2 for haploid or diploid organism, respectively, and *N* is the effective population size).
Altogether, by this approach evolutionary events are evaluated according to the fitness landscape.
Next, the fitness function is used to build the forward in time (from generation *t* to generation *t+1*) evolutionary history with the *birth-death* population process [1-3]. We assume that the birth and death rates are directly proportional and inversely proportional to the fitness, respectively. In this concern, *ProteinEvolver2* implements three approaches:

1. The birth and death rates are user-specified, so they are not derived from the fitness of the protein variant.
2. The birth and death rates are directly obtained from the fitness of the protein variant where a variant with higher folding stability has a higher birth rate and lower death rate. Thus, the fitness is only based on the folding stability as previously mentioned and we favor that variants with higher folding stability are more successful.
3. The birth and death rates are directly obtained from the fitness of the protein variant but where a variant with folding stability more similar to the observed stability (input protein structure) has a higher birth rate and lower death rate. Thus, the fitness is based on the more realistic folding stability as previously mentioned and we favor that variants with more realistic folding stability are more successful. This comes from considering that a higher stability not necessarily means a higher fitness.

We implemented a standard birth-death population process following [47].

4.2. Coalescent simulations

ProteinEvolver2 implements the coalescent simulations of the previous version of the program [5]. The program implements an extension of the coalescent with recombination, demographics and migration based on the neutral Wright-Fisher model [25, 28, 39] following [33, 48]. Given the specified recombination and migration rates (and other parameters like the effective population size (N) and growth rate) random genealogies are produced. Recombination hotspots are also implemented following *SNPsim* [27]. Several migration models (island [30], stepping-stone [29] and continent-island [30]) 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, 31] and simulated data can be haploid or diploid.

The coalescent proceeds backwards starting from the sample of *s* gametes. Time is scaled in units of 2*N* 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*):

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

4.2.1. Recombination

The rate of recombination depends on the population size (*N*) and on the total recombination rate at all valid recombination sites (G). A *valid* recombination site has to have at both sides ancestral material that has not found 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, in the presence of recombination, different regions of the alignment might evolve under different genealogies (Figure 1), which together conform to the ancestral recombination graph. The number of genealogies will be the number of breakpoints + 1.

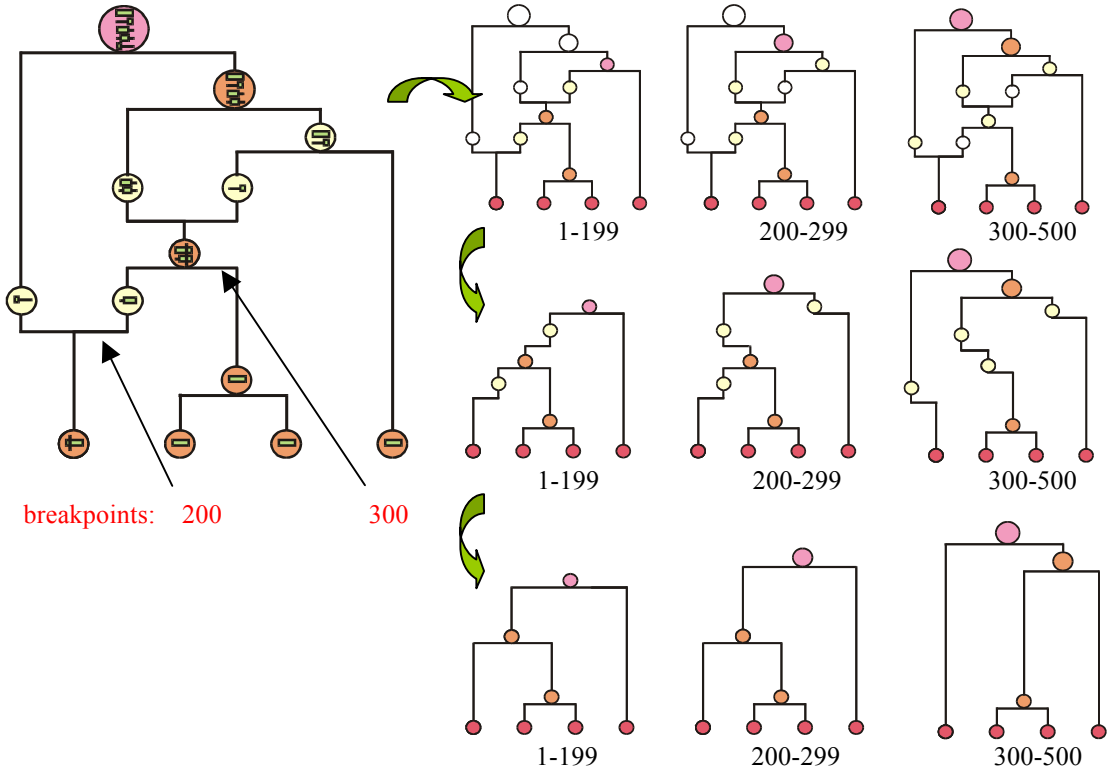


Figure 1. Representation of the ancestral recombination graph and the binary tree embedded.

4.2.1.1 Recombination hotspots

This implementation is based on *SNPsim* [27], the population genetic model of recombination hotspots developed by Wiuf and Posada, to result in heterogeneous recombination rates along the chromosomes.

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 [27].

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.

$$Background\ recombination\ rate\ (R_B) = 4Nr_BL$$

$$Hotspot\ recombination\ rate\ (R_H) = 4Nr_H X$$

$$Global\ recombination\ rate\ (R_G) = R_B + R_H$$

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 (see Figure 2).

Hotspot imprecision

We will consider that the hotspot location actually represents the center of the hotspot, and where recombination is more likely, but that 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) (see 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 we 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". We 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 we use a thinning algorithm to locate these hotspots (Figure 2). If m=1 we distributed the hotspots according to a Poisson distribution with intensity λ.

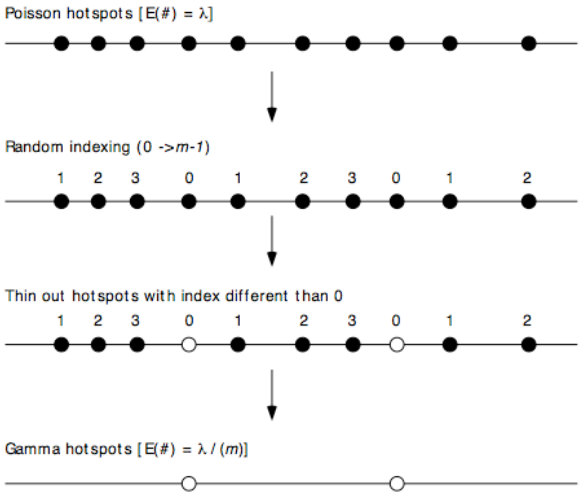


Figure 2. Thinning algorithm for the location of hotspots with interference (m=4). The index for the first hotspot site is chosen uniformly between 0 and m-1. This plot was taken from SNPsim manual [27].

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

The distribution $\Re$ is 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.

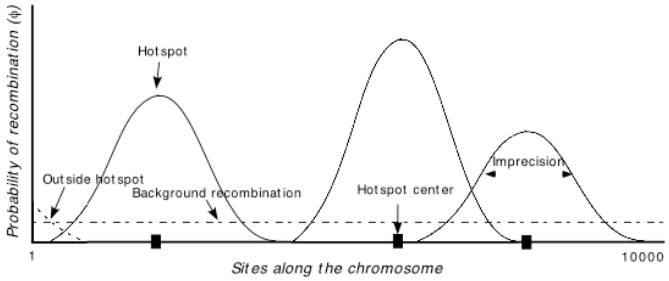


Figure 3. Schematic representation of $\Re$ in the case of three Normal hotspots for the region of interest (L=10000). In this case the hotspot imprecision is quite big. Note that some recombination probability is contributed by a hotspot outside the region of interest. The background recombination is the same for all sites. In this case there is hotspot heterogeneity. This plot was taken from SNPsim manual [27].

We can use now ℜ 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.

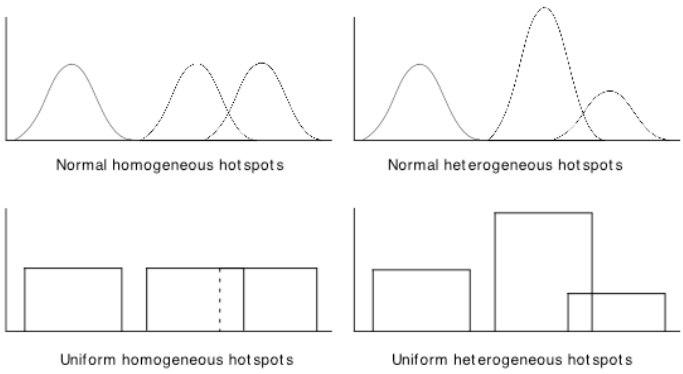


Figure 4. Schematic representation of different ℜ. This plot was taken from SNPsim manual [27].

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

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

In the presence of recombination, classic Markov substitution models of evolution independently evolve each site along the evolutionary history of the corresponding recombinant fragment [see for example 33, 48, 49, 50]. Nevertheless, the SCS models require the evolution of the whole molecule because the dependence among sites. This implied the development of a new algorithm to evolve whole molecules under recombination.

After building the ARG, the molecular evolution starts by the assignation of a sequence to the GMRCA node. *ProteinEvolver2* incorporates an algorithm for the evolution of the whole molecule along the ARG (see Figure 5), which is an adaptation of the algorithm developed to by Arenas and Posada to simulate intracodon recombination [33]. 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 the 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. By this algorithm the molecular

evolution considers the exchanges of material introduced by the recombination and that are described in the ARG.

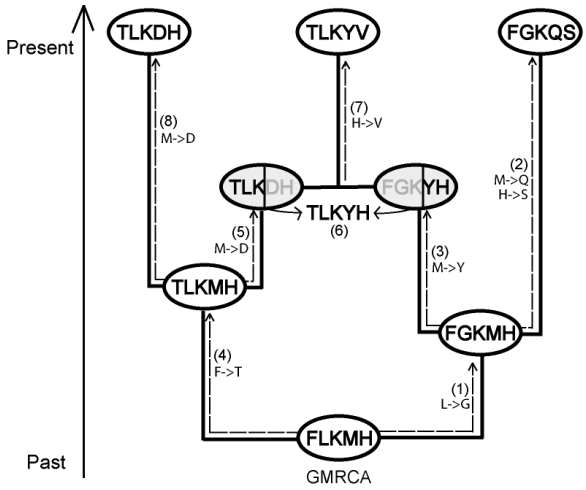


Figure 5. Example of the evolution of a whole molecule along an ARG.

4.2.2 Migration

The simplest and most widely used model of population structure is the "finite island model" [28, 30, 51-53] however the stepping-stone [29] and an approximation to the continent-island [30] are also implemented. The rate of migration 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).

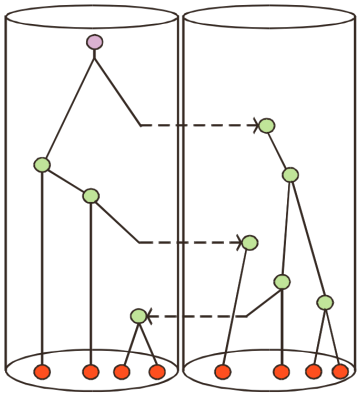


Figure 6. Coalescent with migration for two demes.

4.2.2.1 Island model

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

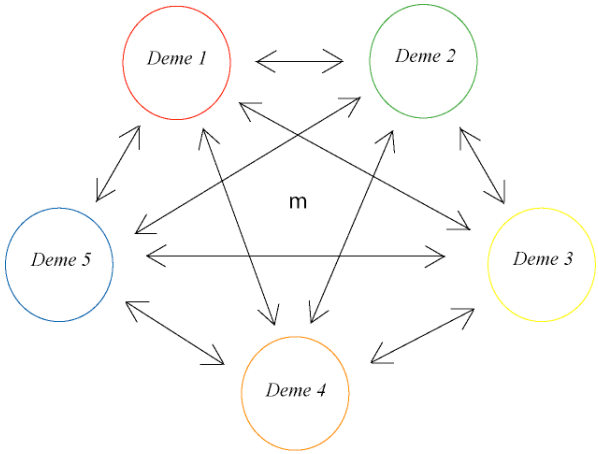


Figure 7. Island model. All demes have the same population size and the same migrant rate per deme.

4.2.2.2 Stepping-stone model

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

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

4.2.2.3 Continent-island model

The continent-island model implemented in *ProteinEvolver2* allows symmetrical migration between a deme (continent) and the other demes (islands), see Figure 9. Note that there is not possible to directly migrate individuals among islands.

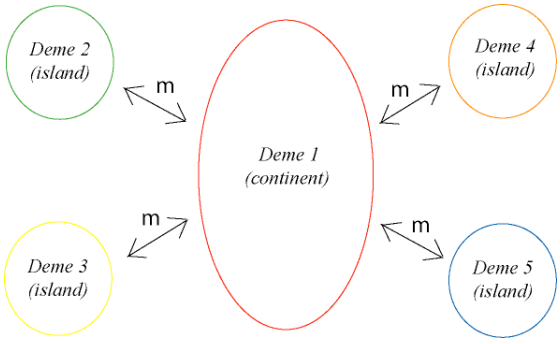


Figure 9. Continent-island model implemented in ProteinEvolver.

4.2.2.4 Temporal variation of the migration rate

ProteinEvolver2 allows for temporal variation where a migration rate can be applied at different time periods (Figure 10).

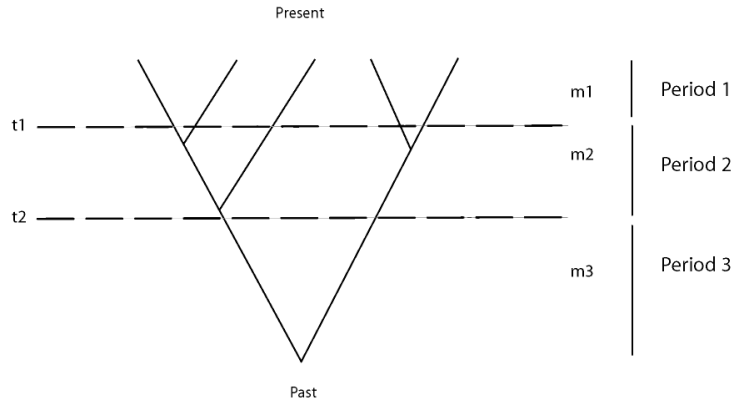


Figure 10. Temporal variation of the migration rate implemented in ProteinEvolver. The figure shows 3 temporal periods defined by two times (t1 and t2) and where each period implements a given migration rate (m1, m2 or m3).

4.2.3 Convergence of demes

The evolution of the demes can be user-specified. Two demes can be converged at a given time in order to build a new big deme with the lineages of the previous demes. The convergence of demes always occurs under a migration model. An example is shown in the Figure 11. It is not possible use convergence of demes in presence of tip dates when the time of the convergence of demes is younger that the time of the tip dates.

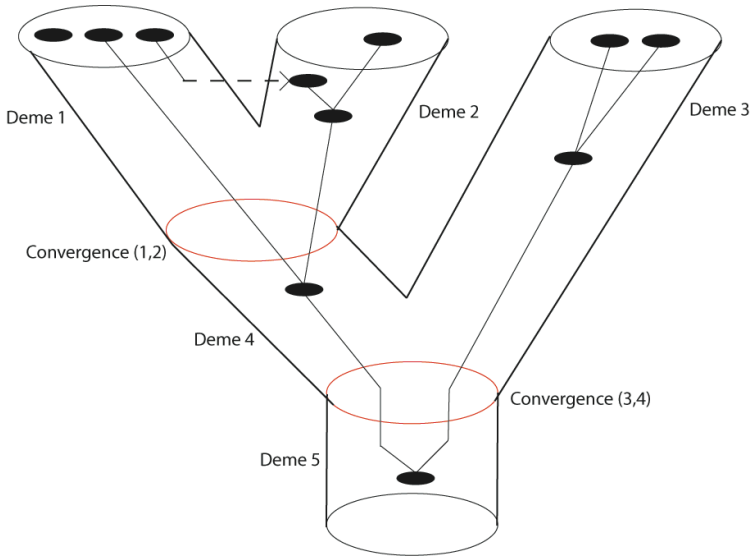


Figure 11. Demes evolution. Two demes can evolve convergences to generate an ancestral big deme, these convergences are introduced by the user.

4.2.4 Demography

ProteinEvolver2 allows for the specification of exponential population growth rate. The only modification concerns to 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., coalescence does not happen), and *ProteinEvolver2* will stop and issue an error message. Alternatively, the user can define any number of demographic periods (Figure 12). β_i is the growth rate inferred for a demographic period i that goes from size N_{Bi} in the past to size N_{Ei} in I_i generations:

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

The time to coalescence will be:

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

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

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

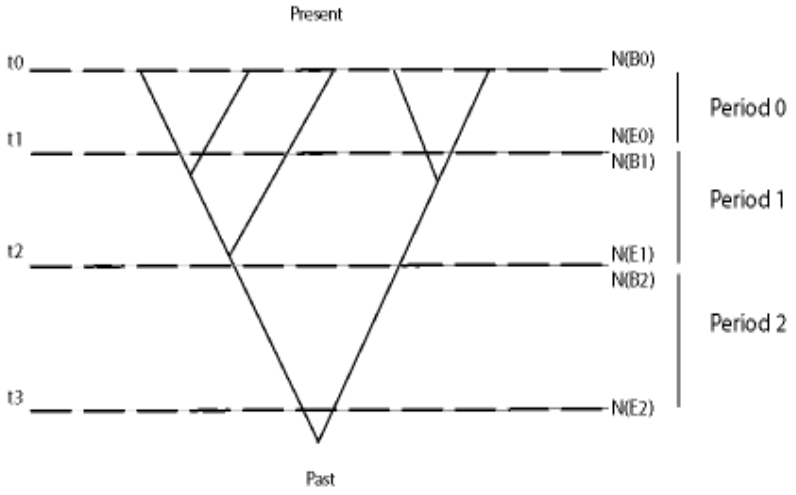


Figure 12. Demographic periods. The growth rate after the last period will the same as the one implied by the last period.

4.2.5 Tip Dates

Tip dates can be specified by the user to simulate sequences with different times, such as samples taken 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).

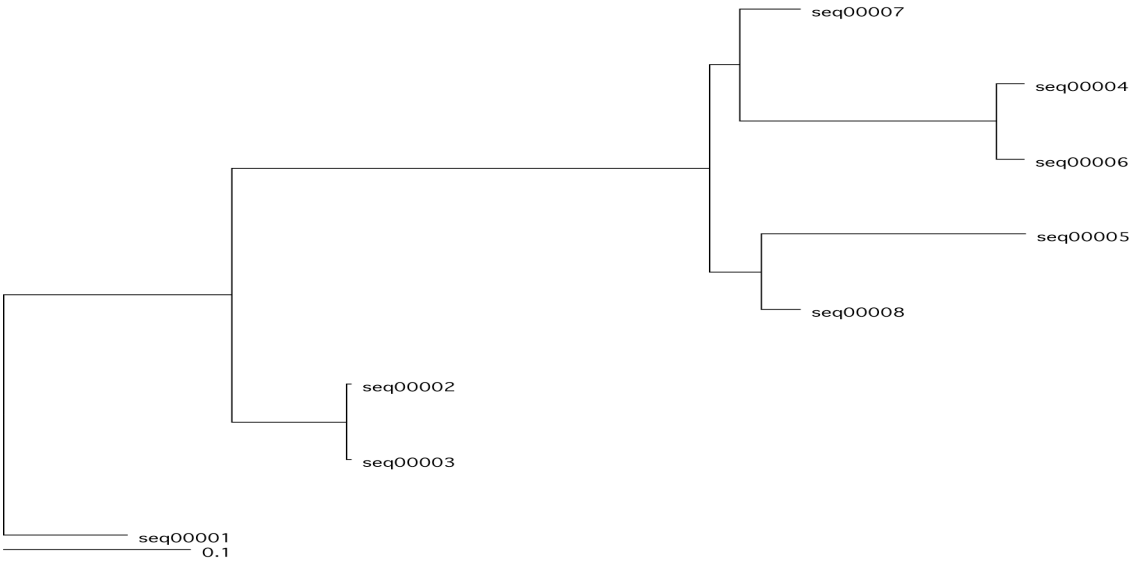


Figure 13. Tip dates simulation examples. Three examples of four samples taken at different times with 1, 2, 2 and 3 sequences per sample (from present to past).

4.3. Markov substitution models

Nucleotide or amino acid data can be simulated along its branches. *ProteinEvolver2* 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 its own sequence (highly recommended for structural protein stability substitution models).

Variation in the substitution rate among sites can be user-established by the shape of a gamma distribution (+G). Indeed, *ProteinEvolver2* also implements variation of substitution rates by a user-specified site-by-site vector. Finally, a proportion of invariable sites (+I) can be also simulated. Furthermore, the user can alter the substitution rate for each site, this allows to fix particular sites. For example, it would make sense to fix sites related with the activity of the protein such as catalytic positions.

4.3.1. DNA models

ProteinEvolver2 implements the general time reversible model for nucleotide substitution (GTR) [11], a non reversible version (GTnR) [11], and models nested therein, like JC [6], K80 [7], F81 [8] or HKY [9].

4.3.2. Empirical amino acid models

ProteinEvolver2 implements a variety of empirical amino acid models: *Blosum62* [12], *CpRev* [13], *Dayhoff* [14], *DayhoffDCMUT* [15], *HIVb* [16], *HIVw* [16], *JTT* [17], *JonesDCMUT* [15], *LG* [18], *Mtart* [19], *Mtmam* [20], *Mtrev24* [21], *RtRev* [22], *VT* [23], *WAG* [24]. In addition, it is possible to specify a user-defined empirical amino acid model by an input file (see section 3.4.4).

4.4. Structurally constrained substitution models

ProteinEvolver2 implements site-dependent structurally constrained substitution (SCS) models that consider the stability of the protein structure and that are derived from [5]. These models compute the structural energy of mutated proteins (including misfolded and unfolded configurations) given the entropy and the temperature. Then, mutations can be accepted (substitutions) or rejected according to the Moran's model. 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 (see below) to compute substitution rates per site and state.
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.

Our site-dependent SCS models estimates 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 [54]. We assume 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. We adopt the contact interaction matrix determined in Bastolla et al. [55]. 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 Fold-X [56], 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 [57-59]. 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 we consider the set of alternative compact matrices of L residues that can be obtained from non-redundant structures in the Protein Data Bank. 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, [42]] as,

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

where $\langle E(A,C) \rangle$ is the mean and σ^2 is the variance of the energy of alternative structures [43]. 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 [42]. A precise computation showed that the third moment of the energy can not be neglected (Minning et al. 2013). We therefore consider the equation,

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

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

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 [59]. The computing time is considerably reduced by approximating the above formula with one that only depends on pairs of residues,

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

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 A_0 is the protein sequence in the Protein Data Bank, 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 [43],

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

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 $\Delta G > 0$.

We then assume that the mutation rate is very small so that the population is monomorphic, and model selection through the Moran's birth-death process [45], which yields the fixation probability,

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

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 *ProteinEvolver2* 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 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 protein stability substitution models, coding DNA and amino acid sequences.

4.4.1.1. Simulation of coding DNA data with the SCS models

The simulation of coding DNA data consists basically in the following algorithm (which can be specified with 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 leads to a nonsynonymous change, it is evaluated by the computation of the structural protein energy for the mutated amino acid protein (section 4.2.2), the energy is used to compute a fitness value (section 4.2.3) and the mutation can be accepted or rejected according to the Moran's birth-death process [45]. The mutation can be accepted or rejected and the DNA and protein sequences are updated according to such a result.

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 (see section 3.3.1).

4.4.1.2. Simulation of protein data with the SCS models

The simulation of protein data consists in the following algorithm (which can be specified with 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 "*-f20*".

II) According to the mutation rates, a mutation at amino acid level is introduced. Note that the mutation will be always classified as non-synonymous.

III) The mutation is evaluated by the computation of the structural protein energy for the mutated amino acid protein (section 4.2.2), the energy is used to compute a fitness value (section 4.2.3) and the mutation can be accepted or rejected [45]. The protein sequence is updated according to such a result.

The number of mutations or substitutions (accepted mutations) is computed according to the branch length and here the user can specify if such branch length should be again considered as mutations or substitutions (see section 3.3.1).

4.4.2 Evolution of sequences along evolutionary histories 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 in 4.3.1.3. Here, the evolution starts from a molecule assigned to the GMRCA node of the ARG [34]. By this procedure all the material involved in the recombination events is considered.

5. Program benchmarking

The models on which *ProteinEvolver2* have been separately validated.
The evolutionary histories simulated with the implemented birth-death population processes successfully showed tree balance by the Colless index [60].
The method to estimate protein folding stability against unfolding and misfolding has been developed in [59, 61, 62] and yields good correlations with experimentally measured free energies [63]. Its computational simplicity allows its application over long evolutionary trajectories.
The substitution processes implemented in the SCS models were verified using HYPHY [64] and agreed with the branch lengths. The implemented site-independent DNA substitution models were previously validated [33, 48]. The empirical amino acid substitution models were accurately estimated using *ProtTest* [65]. 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 [48, 66]. Recombination hotspots agree with theoretical expectations and breakpoints were likely to occur in recombination hotspots regions [27].

6. History

From ProteinEvolver1,

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
- Evaluation of the BD model (tree balance metric, Colless index).
- Option of time discrete (by generation) for 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)

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

7. Acknowledgments

ProteinEvolver2 was supported by the Spanish Ministry of Economy and Competitiveness and Ministry of Science and Innovation through the Grants RYC-2015-18241 and PID2019-107931GA-I00/AEI/10.13039/501100011033.

8. References

1. Stadler T: **Sampling-through-time in birth–death trees**. *J Theor Biol* 2010, **267**(3):396-404.
2. Stadler T: **Simulating trees with a fixed number of extant species**. *Syst Biol* 2011, **60**(5):676-684.
3. Gernhard T: **The conditioned reconstructed process**. *J Theor Biol* 2008, **253**(4):769-778.
4. Bastolla U, Porto M, Roman HE, Vendruscolo M: **Structural approaches to sequence evolution**. Berlin, Heidelberg: Springer; 2007.
5. 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.
6. Jukes TH, Cantor CR: **Evolution of protein molecules**. In: *Mammalian Protein Metabolism*. Edited by Munro HM. New York, NY: Academic Press; 1969: 21-132.
7. 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.
8. Felsenstein J: **Evolutionary trees from DNA sequences: A maximum likelihood approach**. *J Mol Evol* 1981, **17**:368-376.
9. 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.
10. Zharkikh A: **Estimation of evolutionary distances between nucleotide sequences**. *J Mol Evol* 1994, **39**(3):315-329.
11. 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.
12. Henikoff S, Henikoff JG: **Amino acid substitution matrices from protein blocks**. *Proc Natl Acad Sci U S A* 1992, **89**(22):10915-10919.

13. 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.
14. 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.
15. Kosiol C, Goldman N: **Different versions of the Dayhoff rate matrix**. *Mol Biol Evol* 2005, **22**(2):193-199.
16. 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.
17. Jones DT, Taylor WR, Thornton JM: **The rapid generation of mutation data matrices from protein sequences**. *Comput Appl Biosci* 1992, **8**(3):275-282.
18. Le SQ, Gascuel O: **An improved general amino acid replacement matrix**. *Mol Biol Evol* 2008, **25**(7):1307-1320.
19. Abascal F, Posada D, Zardoya R: **MtArt: A New Model of Amino Acid Replacement for Arthropoda**. *Mol Biol Evol* 2007, **24**(1):1-5.
20. 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.
21. Adachi J, Hasegawa M: **MOLPHY version 2.3: programs for molecular phylogenetics based in maximum likelihood**. *Comput Sci Monogr* 1996, **28**:1-150.
22. 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.
23. Muller T, Vingron M: **Modeling amino acid replacement**. *J Comput Biol* 2000, **7**(6):761-776.
24. 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.
25. Hudson RR: **Properties of a neutral allele model with intragenic recombination**. *Theor Popul Biol* 1983, **23**(2):183-201.
26. Kingman JFC: **The coalescent**. *Stochastic Processes and their Applications* 1982, **13**:235-248.
27. Wiuf C, Posada D: **A coalescent model of recombination hotspots**. *Genetics* 2003, **164**(1):407-417.
28. Hudson RR: **Island models and the coalescent process**. *Mol Ecol* 1998, **7**(4):413-418.
29. 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.
30. Wright S: **Evolution in Mendelian populations**. *Genetics* 1931, **16**:97-159.
31. 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.
32. Arenas M, Posada D: **The effect of recombination on the reconstruction of ancestral sequences**. *Genetics* 2010, **184**(4):1133-1139.
33. Arenas M, Posada D: **Coalescent simulation of intracodon recombination**. *Genetics* 2010, **184**(2):429-437.

34. 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.
35. Beaumont MA: **Approximate Bayesian Computation in Evolution and Ecology**. *Annu Rev Ecol Evol Syst* 2010, **41**:379-405.
36. Beaumont MA, Zhang W, Balding DJ: **Approximate Bayesian computation in population genetics**. *Genetics* 2002, **162**(4):2025-2035.
37. Yang Z: **Among-site rate variation and its impact on phylogenetic analysis**. *Trends Ecol Evol* 1996, **11**(9):367-372.
38. Arenas M, Patricio M, Posada D, Valiente G: **Characterization of phylogenetic networks with NetTest**. *BMC Bioinformatics* 2010, **11**(1):268.
39. Hudson RR: **Generating samples under a Wright-Fisher neutral model of genetic variation**. *Bioinformatics* 2002, **18**(2):337-338.
40. 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.
41. Yang Z: **PAML: a program package for phylogenetic analysis by maximum likelihood**. *Comput Appl Biosciences* 1997, **13**(5):555-556.
42. Derrida B: **Random Energy Model: An exactly solvable model of disordered systems**. *Phys Rev B* 1981, **24**:2613-2626.
43. Goldstein RA: **The evolution and evolutionary consequences of marginal thermostability in proteins**. *Proteins* 2011, **79**(5):1396-1407.
44. 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.
45. Ewens WJ: **Mathematical Population Genetics**, vol. 9. Berlin: Springer-Verlag; 1979.
46. Sella G, Hirsh AE: **The application of statistical physics to evolutionary biology**. *Proc Natl Acad Sci USA* 2005, **102**(27):9541-9546.
47. Harmon LJ: **Introduction to birth-death models**. In: *Phylogenetic Comparative Methods.* 2019: https://lukejharmon.github.io/pcm/chapter10_birthdeath/.
48. Arenas M, Posada D: **Recodon: coalescent simulation of coding DNA sequences with recombination, migration and demography**. *BMC Bioinformatics* 2007, **8**:458.
49. Arenas M: **Simulation of Molecular Data under Diverse Evolutionary Scenarios**. *PLoS Comput Biol* 2012, **8**(5):e1002495.
50. Fletcher W, Yang Z: **INDELible: a flexible simulator of biological sequence evolution**. *Mol Biol Evol* 2009, **26**(8):1879-1888.
51. Latter BD: **The island model of population differentiation: a general solution**. *Genetics* 1973, **73**(1):147-157.
52. Maruyama T: **A simple proof that certain quantities are independent of the geographical structure of population**. *Theor Popul Biol* 1974, **5**(2):148-154.
53. Matsen FA, Wakeley J: **Convergence to the island-model coalescent process in populations with restricted migration**. *Genetics* 2006, **172**(1):701-708.
54. Vendruscolo M, Kussell E, Domany E: **Recovery of protein structure from contact maps**. *Fold Des* 1997, **2**(5):295-306.
55. 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.

56. 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.
57. 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.
58. 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.
59. Minning J, Porto M, Bastolla U: **Detecting selection for negative design in proteins through an improved model of the misfolded state**. *Proteins* 2013.
60. Colless DH: **Review of Phylogenetics: The Theory and Practice of Phylogenetic Systematics**. *Syst Zool* 1982, **31**(1):100-104.
61. 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.
62. 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.
63. Bastolla U, Moya A, Viguera E, van Ham RC: **Genomic determinants of protein folding thermodynamics in prokaryotic organisms**. *J Mol Biol* 2004, **343**(5):1451-1466.
64. Kosakovsky Pond SL, Frost SD, Muse SV: **HYPHY: Hypothesis testing using phylogenies**. *Bioinformatics* 2005, **21**:676-679.
65. Abascal F, Zardoya R, Posada D: **ProtTest: selection of best-fit models of protein evolution**. *Bioinformatics* 2005, **21**(9):2104-2105.
66. Hudson RR: **Gene genealogies and the coalescent process**. *Oxf Surv Evol Biol* 1990, **7**:1-44.