Documentation for *ProteinModelerABC*

Current version 1.0

Selection among site-dependent structurally constrained substitution models of protein evolution by approximate Bayesian computation

David Ferreiro, Catarina Branco, Ugo Bastolla and Miguel Arenas

david.ferreiro.garcia@uvigo.es – ferreirogarciadavid@gmail.com
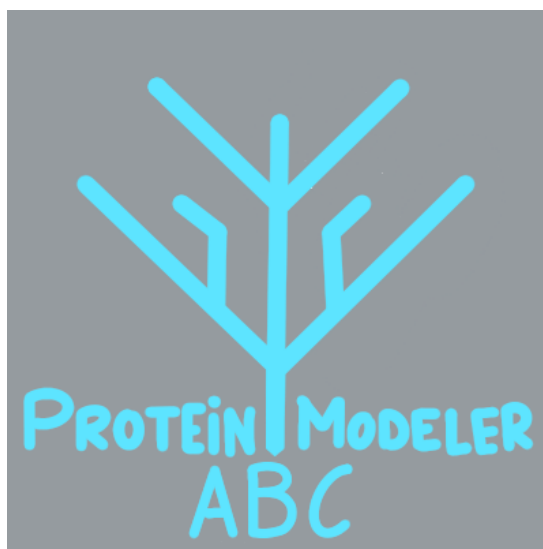
07/03/2023

Table of contents

Disclaimer

Credits

This program was developed at,
CINBIO, University of Vigo, 36310 Vigo, Spain.
Department of Biochemistry, Genetics and Immunology, University of Vigo, 36310 Vigo, Spain.

1. Purpose

ProteinModelerABC is an evolutionary framework to estimate the best-fitting substitution model of protein evolution, among a set of complex substitution models that cannot be accommodated in likelihood functions, with approximate Bayesian computation (ABC). *ProteinModelerABC* is designed to run either on Linux OS or Mac OSX and it is freely available from https://github.com/DavidFerreiro/ProteinModelerABC. It can be run on the command line or through a graphical user interface (GUI) *ProteinModelerABC_GUI*, which allows a user-friendly procedure to run *ProteinModelerABC*. We also include a version to run the framework on clusters (*ProteinModelerABC_Cluster*) to reduce the computer time. The computer simulations are internally performed with the *ProteinEvolverProtABC* program, a version of the simulator *ProteinEvolver* (https://github.com/MiguelArenas/proteinevolver) (Arenas et al. 2013) adapted to ABC. *ProteinModelerABC* implements 7 summary statistics (SS), which inform about the molecular diversity, stability and physicochemical properties of the amino acid replacements of the studied protein sequences. Conveniently, *ProteinModelerABC* can run the simulations on parallel to reduce the computer time. This is highly recommended because simulations under substitution models that consider the protein structure require a long computer time. Three ABC methods are implemented in *ProteinModelerABC*: (*i*) the rejection algorithm (*rejection*) (Csilléry et al. 2012), (*ii*) the multinomial logistic regression (*mnlogistic*) (Csilléry et al. 2012), and (*iii*) the neural networks (*neuralnet*) (Beaumont 2010; Csilléry et al. 2012). For all of them, the user has to specify the desired number of simulations, proportion of retained simulations (tolerance), number of cross-validation events for each model (iterations) and SS. *ProteinModelerABC* provides also several diagnostic plots to qualitatively assess the fitting of the data with the models, namely: boxplots of the SS from simulations under the evaluated models, histograms for all together and each evaluated substitution model with the distance between the SS from the retained simulations and the SS from the observed data, and a plot with the two principal components calculated from SS of the simulations and observed data.

2. Versions and graphical user interface

The package includes two versions of the *ProteinModelerABC* framework, one to run it on a local computer (*ProteinModelerABC* folder) and another one to run it on a computer cluster (*ProteinModelerABC_Cluster*).

2.1. *ProteinModelerABC*

The graphical user interface
The user can run the framework on their computer using the command line or a graphical user interface (GUI) written in Python. The GUI consists of two consecutive windows. The first one to specify the input parameters (some of them are mandatory and others are optional, see section 4.2). Despite the framework is designed to raise an error when detects an incorrect specification (e.g., a negative substitution rate), the user should consider that settings without biological meaning may still produce results (see section 5.5). The second window runs the analysis, including the simulations, calculation of SS and ABC estimation. When running the program informs about its progress (see section 5.1.1). Note that the GUI is optimized to run on Mac OSX so it appearance in Linux OS may differ from the one we shown in section 5.1.1

The command line
The GUI and command line versions implement the same methods. However, in the command line version the user has to change manually the value of the parameters in the main input file (Settings.txt). Importantly, the format of this input file has to be maintained (see section 4.2), otherwise the framework can fail when reading the file. We recommend the use a Settings.txt file included in the *Examples* folder as a template. Once the framework was executed, it checks the input information before performing the analyses. Information about the running progress is printed on the command line (see section 5.1.2).

2.2. *ProteinModelerABC_Cluster*

ProteinModelerABC can run on clusters based on Linux OS environments. It enables users to work in parallel with multiple processors without sharing memory, hence taking advantage of MPI (Message Passing Interface, a standard designed to run programs in multiple processors). The user has to fill the input files considering parameter values with biological meaning and maintaining the file format (see section 4.2). The *ProteinModelerABC_Cluster* execution check the input values and calculate the SS of the query multiple sequence alignment, but it does not launch the simulations and their corresponding SS calculation. Instead, the program internally produces some batch scripts to run the simulations and their corresponding SS calculation. In particular, the user has to submit a master batch script called *launch_Simu.sh* to Slurm and similar operative systems of computer clusters (see section 5.1.3). Once the analysis is finished, the framework produces the same output folders and files to those produced when running in a local computer (see section 5.2).

3. Compilation

To install the framework, the user has to use the computer terminal, and in the main directory of *ProteinModelerABC* execute the "*make install*" command. This compiles all the accessory programs: *ProteinEvolverProtABC* simulator (Arenas et al. 2013) and *DeltaGREM* software (Arenas et al. 2017) which are used to simulate the protein sequences and to calculate the protein free energy, respectively, through a *Makefile.* The framework was tested using the 9.3.0 GNU Fortran (GCC) compiler.

```
make install
```

To install the GUI through a *Makefile* the user must execute the "*make install-GUI*" command. The distributed files (*.py* and *Makefiles*) and folders (*GUI* and *source*) cannot be modified because they contain the necessary information for a successfully compilation and execution. The executable file to launch the GUI should be placed in the *Executable* folder of the *GUI* folder, while the resulting executable files for the accessory programs should be placed in a *bin* folder.

```
make install-GUI
```

Note that *ProteinModelerABC* requires that Python3 (https://www.python.org/downloads/) and R (https://www.r-project.org), as well some libraries, are installed by the user. The libraries that should be installed to run *ProteinModelerABC* are shown in Table 1. Python libraries can be installed by the command: "pip install *library_name*" on the terminal or, in the case of R, by the command "install.packages("*library_name*")" in the R environment. See further details in https://docs.python.org/3/installing/index.html, http://cran.r-project.org/doc/manuals/R-admin.html#Installing-packages. For example,

```
pip install pandas
```

```
install.packages("abc")
```

Table 1. Libraries required to run *ProteinModelerABC.*

Name	Language	Version
abc	R	All
os	Python	All
sys	Python	All
Biopython	Python	All
random	Python	All
numpy	Python	All
warnings	Python	All
pandas	Python	All
csv	Python	All
multiprocessing	Python	Command line and GUI
re	Python	All
platform	Python	All
mpi4py	Python	Cluster
threading	Python	GUI
tkinter	Python	GUI
PyInstaller	Python	GUI
time	Python	GUI

4. *ProteinModelerABC* usage

ProteinModelerABC has four mandatory input files:
- **Settings file:** This file, named *Settings.txt*, must contain all the desired specifications for simulations, SS calculation and ABC estimation. It has to be carefully specified (see section 4.2).
- **Multiple alignment of protein sequences (protein MSA):** The protein MSA must be provided by the user in sequential *phylip* format (.phy). In this format the sequence identifier is exactly 10 characters long, padded with spaces when necessary, followed by the amino acid sequence in the same line.
- **Template protein structure:** This is a protein structure (.pdb file) with high homology with the protein MSA.
- **Contacts in other structures:** The file *structures.in* contains a huge list of contact among atoms from multiple protein structures present in the PDB and that is needed for the calculation of the protein folding stability and the site-dependent structurally constrained substitution (SCS) models.

4.1. Before starting

Before starting a *ProteinModelerABC* analysis the user should obtain a template protein structure which properly represents the MSA and align the MSA with the corresponding template sequence, removing the positions or sites of the MSA that do not show homology with the template sequence. To do so, we provide some Python scripts (located in the *Accessory_Scripts* folder), details below.

To find a protein structure that properly represents the MSA we recommend the use of *SWISS-MODEL* (Waterhouse et al. 2018) (https://swissmodel.expasy.org). *SWISS-MODEL* provides a template structure by homology modeling for a maximum of 10 sequences, so if the MSA includes a higher number we recommend to use the Python script *Find-ConsensusSeq.py.* This script works with an input MSA in *fasta* format (a more common format than *phylip*) and returns the consensus sequence of the MSA.

```
python3 Find-ConsensusSeq.py --input MSA.fasta
```

Next, the consensus sequence can be used to find a template in *SWISS-MODEL* that properly represents the MSA. Usually, the user must download the first template of the results screen, but it is recommended to check the method used to determine the structure of a protein, including X-ray crystallography, trying to avoid structures obtained with the X-ray under a resolution higher to 2 amstrongs (Å) and structures from NMR spectroscopy or electron microscopy. SCS models work with complete template sequences (no gaps) and require the MSA to have the same length as the template chain. Thus, to obtained a proper MSA to run *ProteinModelerABC*, the user must do some extra work. The user must align the input MSA and the template sequence with programs like *MUSCLE* (Edgar 2004). If the template sequence presents any gap in the MSA, the user must remove that position or site. Finally, the template sequence must be removed and the MSA has to be converted into *phylip* format using programs like *Phylogeny.fr* (Dereeper et al. 2008). Alternatively, we provided a Python script called *Align.py* which performs all these steps. It requires the input MSA in *fasta* format (--input), a template structure (--temp), the chain of the template (--chain) and the desired name of the output file, which must have the *.phy* extension (--output).

```
python3 Align.py --input MSA.fasta --temp structure.pdb --chain A to Z --output MSA.phy
```

The main aim of *ProteinModelerABC* is to include SCS models in an evolutionary framework to identifies a best-fitting substitution model of protein evolution. It is not the best option to estimate the best-fitting substitution model among a set of empirical substitution models, for that there are other tools [e.g., *ProtTest3* (Darriba et al. 2011)]. *ProteinModelerABC* is designed to compare SCS models and compare SCS models with an empirical substitution model that was previously selected for the MSA using other programs such as *ProtTest3*. To perform the simulations the user must provide an input phylogenetic tree or specify parameters for coalescent simulations such as the population size and a prior distribution for the substitution rate per amino acid site. We include a script called *Theta.py* which for a desired the molecular evolutionary rate (theta, *θ*) value returns the sequence identity of the MSA and the substitution rate per amino acid site, asking the ploidy of the organism and the population size. The prior distribution for the substitution rate per amino acid site should be chosen considering the query alignment sequence identity to ensure that it includes the real data. A query alignment with a low sequence identity would require a wider prior distribution.

```
python3 Theta.py --input NS1.phy
```

Examples for a standard run of *ProteinModelerABC* are provided in the folder *Examples*. After completing a run of *ProteinModelerABC,* two folders are created in the working directory, the *ABCOutputs* and the *SimulationsOuputs* (see details in section 5.2). Since the estimation phase with ABC is quite fast in comparison with the simulation phase, the user can explore different settings for the ABC method (see section 5.3 in *Re-analyzing data*) without having to run again the simulation phase. Indeed, we advise the user to do it in order to explore the influence of ABC parameters on the estimations.

4.2. The Settings input file

The Settings input file contains the main information required to perform a complete *ProteinModelerABC* analysis. It is highly recommended to carefully specify this file, since mistakes may affect results (in some cases error messages can be displayed on the screen and may suggest stop the execution by typing CTRL+C). The file consists of three main blocks: general input data and information, settings for the simulation phase and settings for the estimation phase.

Important notes when working with the Settings file
- Parameter values must be introduced in the line after the parameter description, otherwise such parameter will be considered as "*not specified*" (see examples below).
- Do not modify the parameter description.
- Some parameters are mandatory and must be specified, these parameters starts with"*****".
- Some parameters require a prior distribution (see section 4.2.1), specifically a distribution of the parameters values used to perform the simulations that should include the presumed value of the query data (for example, if the user considers that the substitution rate per site may be around 0.1, then the prior distribution of the substitution rate per site should include 0.1 within its range).

4.2.1. Implemented prior distributions

Several prior distributions are included in *ProteinModelerABC* to simulate protein data (Table 2) and affect the following parameters:

The amino acid substitution rate per site is a mandatory parameter if the coalescent method is selected and includes the *fix*, *uniform*, *norm*(t), *exp*(t), *gamma*(t), *beta*(t) distributions.

The generation time is an optional parameter, it includes the *fix* and *uniform* distributions. The amino acid frequencies is a mandatory parameter if any empirical model is considered and includes *fix* or *Dirichlet* distributions.

The heterogeneity of the substitution rate among sites (+G) and the proportion of invariable sites (+I) are optional parameters and both include the *fix*, *uniform*, *norm*(t), *exp*(t), *gamma*(t), *beta*(t) distributions.

Most of these prior distributions can be truncated at lowest and highest values but not all the distributions can be applied to all the parameters (see the following subsections).

Table 2. Available prior distributions in *ProteinModelerABC*.

Distribution	Description	Truncated	Examples
Fix (**fix**)	Fixed value (integer or float)	n.a.	fix 4; fix 0.7
Uniform (**uniform**)	Random between two values (integer or non-integer): lowest highest	n.a.	uniform 1.0e-8 1.0e-5; uniform 2e-9 5e-6; uniform 0 3
Normal (**norm**)	Normal distribution (mean, sd)	t # #	norm 1.0e-8 1.0e-5; norm 1.0e-8 1.0e-5 t 1.0e-9 1.0e-6
Exponential (**exp**)	Exponential distribution (rate)	t # #	exp 1.0e-7; exp 1.0e-7 t 1.0e-8 1.0e-6
Gamma (**gamma**)	Gamma distribution (shape, rate "1/scale")	t # #	gamma 1.0e-7 5.0e-7; gamma 1.0e-7 5.0e-7 t 2.0e-7 1.0e-6
Beta (**beta**)	Beta distribution (shape1, shape2)	t # #	beta 1.0e-7 5.0e-7; beta 1.0e-7 5.0e-7 t 2.5e-7 1.0e-6
Dirichlet (**Dirichlet**)	Dirichlet distribution (alpha "vector")	n.a.	dirichlet 1 1 1 1; dirichlet 1 1 1 1 1 1

4.2.2. General input data and information

- **Name of the file with the query MSA.** This specification is mandatory. It has to be presented in sequential *phylip* format (.phy). The user has to specify only the filename (i.e., without the path) since the file should be placed in the directory of the Settings.txt file (see section 4.1).

```
# File with the target alignment of protein sequences. Phylip format, see documentation for details
-- MANDATORY --
*NameOfPhylipFile=ProtSeq1.phy
```

- **Consideration of indels.** This parameter is mandatory. The user specifies if indels (gaps) should be "*Ignored*" (by default and recommended) or considered as a new state "*NewState*". This decision can affect the SS values if there are indels in the MSA.

```
# Consideration of indels. "Ignored" (indels are ignored), "NewState" (indels are considered as a new state)
-- MANDATORY --
*Indels=Ignored
```

- **Template.** This specification is mandatory. File of protein structure (representative of the sequences of the MSA) in PDB format used for the SCS models and to calculate proteins free energy in certain summary statistics (see section 4.1).

```
# File with the protein structure (PDB file). Protein structure used for structurally constrained substitution models and
certain summary statistics -- MANDATORY --
*Template=3IXO.pdb
```

- **Chain.** This parameter is mandatory. Chain of the protein structure used to SCS models simulations and to calculate proteins free energy.

```
# Chain of the protein structure. See documentation for details -- MANDATORY --
*Chain=A
```

- **Show running information.** This parameter is mandatory. If the user specifies "*No*" the amount of information printed on the screen during the execution is small. That option is recommended because printing information on the screen requires more computational resources. Regardless of the option selected, the results do not change.

```
# Show running information (simulations and summary statistics) on the screen. It increases the computer time. See
documentation for details -- MANDATORY –
*ShowInformationScreen=No
```

4.2.3.   Settings for the simulation phase

4.2.3.1.   General simulation settings

- **Number of simulations.** This parameter is mandatory. We recommend at least 10,000 simulations. Still, the number of simulations required to obtain accurate estimates depends on many factors, especially the query MSA. More complex sequences (large molecular diversity) may require more simulations due to the irregularity of the parametric landscape that could require more sampling.

```
# Total number of simulations -- MANDATORY --
*NumberOfSimulations=10000
```

- **Number of processors to run the simulations in parallel.** This parameter is mandatory (by default, when using the GUI version the simulations run on all the available computer processors). This parallelization works on machines using Linux OS with shared memory. Ideally, one should specify the number of available processors of the machine. For the *ProteinModelerABC_Cluster* version the user should specify the number of cores in the same way. In case of running without parallelization set 1 processor.

```
# Number of available processors to run the simulations in parallel (1 in case of running without parallelization). See
documentation for details -- MANDATORY --
*NumberOfProcessors=12
```

- **Save simulated data.** This parameter is mandatory. We recommend do not save the simulated data because it requires a lot of space in the computer hard disk. If the user would like to save the data it will be placed in a compressed folder called *Simulations.tar.gz*. Note that even if choosing not to save simulated data, simulated data

is created and saved temporally since it is required for calculating the SS in the following phase.

```
# Save simulated data. We recommend do not save the simulated data because it requires a lot of space --
MANDATORY --
*SaveSimulations=No
```

4.2.3.2. Evolutionary history

The user should select a coalescent simulation (with user-specified parameters) or provide a rooted phylogenetic tree upon which protein evolution is simulated.

- **Coalescent history or rooted phylogenetic tree.** This parameter is mandatory. "*Coal*" performs the coalescent simulation demanding some additional parameters information while "*Phylo*" asks for an input file with the phylogenetic tree in *newick* format.

```
# Simulate the evolutionary histories with the coalescent "Coal" or a phylogenetic tree is specified "Phylo". See
documentation for details -- MANDATORY --
*CoalescentOrPhylogeny=Coal
```

Coalescent

- **Haploid/diploid simulated data.** This parameter is mandatory if coalescent is specified. It describes the ploidy of the organism. Haploid is defined with a value of 1 and diploid with a value of 2.

```
# Haploid or Diploid data. Haploid=1, Diploid=2 -- MANDATORY IF COALESCENT --
*Haploid/Diploid=2
```

- **Amino acid substitution rate per site.** This parameter is mandatory if coalescent is specified. It describes the rate of substitution per protein site. The user can choose between the allowed distributions [*fix*, *uniform*, *norm*(t), *exp*(t), *gamma*(t), *beta*(t) (Table 2)].

```
# Amino acid substitution rate. It can be fixed "fix" or include a prior distribution: uniform, gamma, beta, normal,
exponential. See documentation for details -- MANDATORY IF COALESCENT --
*SubstitutionRate=uniform 0 1.67e-4
```

- **Effective population size (*N*).** This parameter is mandatory if coalescent is specified. It defines the effective size of the population from which the sample was collected. It must be an integer.

```
# Population size -- MANDATORY IF COALESCENT --
*PopulationSize=1000
```

- **Sampling at different times.** This parameter is optional. The user can specify the time at which the tip nodes of the tree are sampled. In the example below, 4 sampling times are specified: sampled in 1995 – sequences 1 to 10; sampled in 2003 – sequences 11 to 16; sampled in 1997 – sequences 17 to 26; sampled in 2001 – sequences 27 to 29. This option does not work if a deme converges, backwards in time, before the last sampling time.

```
# Longitudinal sampling. Requires GenerationTime. See documentation for details
DatedTips=4 1995 1 10 2003 11 16 1997 17 26 2001 27 29
```

- **Generation time.** This parameter is optional. The user can specify the time for each generation. The parameter value can be fixed (*fix*) or sampled from a uniform distribution.

```
# Generation time. See documentation for details
GenerationTime=fix 1200
```

- **Population growth rate.** This parameter is optional. The first number specifies the model, exponential growth rate (0) or demographic periods (1). It is not recommended specifying a negative growth rate for the last period, as the coalescent time could become infinite. For an exponential growth per individual per generation, after the "*0*" the growth rate must be specified. In the example the user chose an exponential growth model with a rate of 1e-5.

```
# Exponential growth rate or demographic periods. See documentation for details
GrowthRate=0 1e-5
```

For demographic periods, after the "*1*" the user has to specify the number of periods (from the present to the past). For each period should be three consecutive numbers indicating the size at the beginning and at the end of the period, and the duration of the period in generations. In the example the user specified 3 demographic periods. In the first one, the population size increases from 1000 to 1250 during 1000 generations, in the second the population size increases from 1300 to 1550 between generations 1000 and 2000, and finally during the last period from 2000 to 3000 the population size decreases from 1560 to 1000.

```
# Exponential growth rate or demographic periods. See documentation for details
GrowthRate=1 3 1000 1250 1000 1300 1550 2000 1560 1000 3000
```

- **Migration model.** This parameter is optional. 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 numbers specify the number of individuals (or sequences) per deme (note that the specified sample size must be equal to the sum of these). For the continent-island model, deme #1 will be the continent while the other demes will be islands. In this example the user specified a stepping-stone model (2), two demes (2) with three samples each (3 3).

```
# Migration model and population structure. See documentation for details
MigrationModel=2 2 3 3
```

- **Migration rate.** This parameter is optional. This parameter introduces the migration rate, which can be constant or vary through time according to temporal periods. The first number specifies the number of temporal periods: for only 1 period, the second number is the migration rate (constant). In this example the user specified only 1 period with migration rate of 0.001.

```
# Migration rate. See documentation for details
MigrationRate=1 0.001
```

For more than one period, the second number is the time for the beginning of a new migration rate and the following numbers are the migration rate for each period. In the following example, the user sets a migration rate that varied over two periods: the first period occurs from generation 0 to 100 with a migration rate of 0.001, and the second period goes from generation 100 to the end of the simulation with a migration rate of 0.005.

```
# Migration rate. See documentation for details
MigrationRate=2 100 0.001 0.005
```

In the next example the migration rate varied among three periods: the first period occurs from generation 0 generation 100 with a migration rate of 0.002, the second period occurs from generation 100 to the generation 800 with a migration rate of 0.001, and the last period goes from the generation 800 to the end of the simulation with a migration rate of 0.003.

```
# Migration rate. See documentation for details
MigrationRate=3 100 800 0.002 0.001 0.003
```

- **Convergence demes.** This parameter is optional. The first number specifies the total number of convergent events. For each convergence event should be three following numbers: identification of demes to converge and finally the time of that convergence. With this option the user can build the evolutionary tree but it is only available when the migration model is activated (despite the migration rate could be zero). In the following example there is one convergence event between demes 1 and 2 at time 2000 to create a new deme.

```
# Events of convergence of demes. See documentation for details
ConvergenceDemes=1 1 2 2000
```

In the next example there are 3 convergence events, between demes 1 and 2 at generation 400 to create a new deme (deme 5), convergence of deme 3 with deme 4 to create a new deme (deme 6) at generation 1900, and convergence of deme 5 with deme 6 at generation 2000 to create a new deme (deme 7).

```
# Events of convergence of demes. See documentation for details
ConvergenceDemes=3 1 2 400 3 4 1900 5 6 2000
```

Rooted phylogenetic tree

- **User-specified phylogenetic tree/s.** This parameter is mandatory. It includes the phylogenetic tree input file name (i.e., no path) upon which protein evolution is simulated. The tree should be in *newick* format (see section 6.1).

```
# File with user-specified phylogenetic tree. Newick format. See documentation for details -- MANDATORY IF
USER-SPECIFIED PHYLOGENETIC TREE --
*Tree=phylotree.txt
```

4.2.3.3. Substitution model

Substitution model of amino acid evolution. This parameter is mandatory. The user has to specify a minimum of two substitution models. At least one of them has to be a SCS model (Fitness or Neutral) and then the desired (if any) empirical substitution model of protein evolution from the following: Blosum62, CpRev, Dayhoff, DayhoffDCMUT, HIVb, HIVw, JTT, JonesDCMUT, LG, Mtart, Mtmam, Mtrev24, RtRev, VT and WAG; the use can also specify an own exchangeability matrix and amino acid frequencies through the argument UserEAAM. Among the empirical models, we **recommend to use only the best-fitting empirical substitution model for the alignment** which can be previously obtained using other programs, such as *ProtTest*. **Models should be separated by a space and if any empirical substitution model is specified it should be written <u>before the SCS models</u>.**

```
# Substitution models of protein evolution that are evaluated. Select at least one structurally constrained substitution
(SCS) model (Fitness or Neutral) and one desired empirical substitution model that ideally should be the best-fitting
substitution model selected with ProtTest or another framework (i.e., Blosum62, CpRev, Dayhoff, DayhoffDCMUT,
HIVb, HIVw, JTT, JonesDCMUT, LG, Mtart, Mtmam, Mtrev24, RtRev, VT, WAG). The empirical models should be
specified before the SCS models if any empirical model is selected. See documentation for details -- MANDATORY
--
*SubstitutionModel=HIVw Fitness Neutral
```

Parameters for the empirical substitution models

- **Amino acid frequencies.** This parameter is mandatory for empirical models. Amino acid frequencies. Allowed distributions: *fix* or *dirichlet*. By default, the program assumes equal frequencies (all fix with value 0.05).

```
# Amino acid frequencies. See documentation for details -- MANDATORY IF EMPIRICAL MODEL --
*AminoacidFrequencies=fix 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 0.05
```

- **Heterogeneity of the substitution rate among sites (+G).** Variation of the substitution rate among sites according to a Gamma distribution, the user specifies the alpha-shape parameter. This parameter is optional for empirical substitution models. Allowed distributions: *fix*, *uniform*, *norm*(t), *exp*(t), *gamma*(t), *beta*(t).

```
# Rate of heterogeneity across sites, +G. It can be fixed "fix" or include a prior distribution: uniform, gamma, beta,
normal, exponential
RateHetSites=uniform 0.76 0.90
```

- **Proportion of invariable sites (+I).** This parameter is optional for empirical substitution models. It can take values between 0 (full conservation) and 1 (all the sites are variable). Allowed distributions: *fix*, *uniform*, *norm*(t), *exp*(t), *gamma*(t), *beta*(t).

```
# Proportion of invariable sites, +I. It can be fixed "fix" or include a prior distribution: uniform, gamma, beta, normal,
exponential
PropInvSites=uniform 0.3 0.5
```

Parameters for the SCS models

- **Thermodynamic temperature.** This parameter is mandatory for SCS models. It describes the thermodynamic temperature used to predict the protein folding stability. A range between 1.25 and 2.0 is recommended.

```
# Thermodynamic temperature -- MANDATORY IF SCS MODEL --
*TEMP=1.8
```

- **Configurational entropy per residue (unfolded) and offset (misfolded).** These parameters are mandatory for SCS models. The first parameter corresponds with the configurational entropy per residue for the unfolded protein, the second with the configurational entropy per residue for the misfolded protein and the last one with the configurational entropy offset for the misfolded protein. A range between 0.025 and 0.075 is recommended for the two first parameters and 0 for the last one.

```
# Configurational entropies per residue (unfolded) and offset (misfolded). See documentation for details
-- MANDATORY IF SCS MODEL --
*S0=0.05
*SC1=0.05
*SC0=0.0
```

- **Third cumulant in REM calculation.** This parameter is mandatory for SCS models. It defines the third cumulant for REM calculation to predict the protein folding stability.

```
# Third cumulant in REM calculation -- MANDATORY IF SCS MODEL --
*REM3=0
```

- **Population size considered for the SCS models.** This parameter is mandatory for the site-dependent Fitness SCS model. The population size to accept/reject mutation events in the Moran model.

```
# Population size considered for the SCS model -- MANDATORY IF FITNESS SCS MODEL --
*NPOP=10
```

4.2.4. Estimation phase

- **ABC iterations.** This parameter is mandatory. It defines the number of cross-validation events for each model. We do not recommend a high value since this step may take a long time. Note that the number of simulations must be lower than the number of iterations.

```
# ABC iterations used for the cross validation. See documentation for details -- MANDATORY –
*ABCIterations=100
```

- **ABC tolerance.** This parameter is mandatory. It describes the proportion of simulations that generated SS close to the real SS and thus that are retained to perform the ABC estimation. A tolerance of 0.005 is recommended but the best tolerance value can vary among MSA. Still, choosing the tolerance is not trivial: it has to be large enough to provide a good characterization of the posterior distribution, but on the other hand, it has to be small enough to retain the simulations that are closer to the target sequences alignment. Thus, we recommend the users to explore different values.

```
# ABC tolerance. Proportion of simulations closest to real data to retain in the ABC procedure.  See documentation for
details -- MANDATORY --
*ABCTolerance=0.005
```

- **ABC method.** This parameter is mandatory. The chosen ABC algorithm to perform the analysis. By default we recommend the rejection algorithm, which uses the proportion of accepted simulations to estimate the posterior probabilities of the model(s) (*rejection*) (Csilléry et al. 2012). The multinomial logistic regression (*mnlogistic*) (Csilléry et al. 2012) and the neural networks (*neuralnet*) (Beaumont 2010; Csilléry et al. 2012) are only recommended if the tolerance is high (i.e., 0.1).

```
# ABC method (rejection, mnlogistic or neuralnet).  See documentation for details -- MANDATORY --
*ABCMethod=rejection
```

- **Summary statistics.** This parameter is mandatory. The user has to choose the SS to use for the ABC estimation by specifying their numeric ID (see section 6 *Models, Methods and Summary Statistics* for further details). For initial exploratory analyses, we recommend the use of all the implemented SS. Later, by analysing the output plots the user can inspect which SS better explains the query data. For example, if the standard deviation of folding stability from the simulated data is far from that of the real data we recommend exclude it.

```
# Summary statistics that are used for the ABC estimation. See documentation for details -- MANDATORY --
*SummaryStatistics= 1 2 3 4 5 6 7
```

- **Multiple pages.** This parameter is mandatory. If the user wants to obtain the output PDF documents with multiple output plots with results (see Section 6) in one page ("*Yes*") or by a single plot per page ("*No*").

```
# PDF documents with multiple output plots per page -- MANDATORY --
*MultiPage=Yes
```

4.3. The Phylogenetic tree input file

This input file consists of one text line (empty lines are not allowed) with the range of sites and a tree in *newick* format. The range specifies the first and last amino acid position. Note that only one tree is allowed and it should be rooted (see fast-example 2).

```
1 373
((A0A068J9:0.005294,NCAP_EB1:0.000001):0.030656,(((A0A1Q1NM:0.180375,(NCAP_MAB:0.0080
08,K4N3C7_9:0.000001):0.154238):0.460608,G8EFI1_L:0.215419):0.146965,NCAP_EBO:0.05684
4):0.027132,C6G8D3_E:0.056416);
```

5. *ProteinModelerABC* example

5.1. Execution

The data shown below corresponds with the normal output information of a *ProteinModelerABC* run. The example corresponds to a fast analysis, in which the aim is to distinguish between the empirical HIVw and the Fitness and Neutral SCS models. Before trying this example, Python 3 and R must be installed and the framework was compiled (see section 3). To execute *ProteinModelerABC* all main the Python script (*.py*) must be placed jointly with the mandatory input files (Settings.txt, the multiple alignment of protein sequences, the template structure and the *structures.in* file). *ProteinModelerABC* was developed using Python3.9 version so it is recommended to use Python3.9 version but every version of Python3 is expected to work.

5.1.1. Execution with the GUI

1. To run *ProteinModelerABC* GUI the user can go to the *GUI* directory and type "python3 *ProteinModelerABC_GUI.py*" in the command line, or go to the *Executable* directory and click the executable *ProteinModelerABC_GUI*. Note that GUI *.py* or executable do not have to be in the GUI folder, they can be moved with the corresponding images in the case of the *.py* file.

```
python3 ProteinModelerABC_GUI.py
```

2. Next, the user can fill in all the necessary settings. Once everything is filled, the user must click on the *Save and Continue* button to start the simulation and estimation phases. *ProteinModelerABC* can show warnings if a parameter was not specified or there is any error. See section 4.2 for more information.

Figure 1. Specification of input parameters in the GUI.

3. A new window will appear and, to run the simulation and the estimation phases, the user has to click on the *Start* button. This window will display information about the execution. During the ABC estimation phase the cross-validation step may fail due to a low tolerance value (especially with the ABC multinomial logistic regression and neural networks estimation methods), a low number of simulations and/or a high iterations value. Indeed, the software will try to successfully perform the ABC estimation up to 10 times. In case it does not success, we recommend increasing the tolerance up to a maximum of 0.05. If the error persists, we recommend repeating the analysis considering more simulations. If the user does not select enough simulations the estimation will fail continuously, even after increasing the tolerance or decreasing the number of iterations.
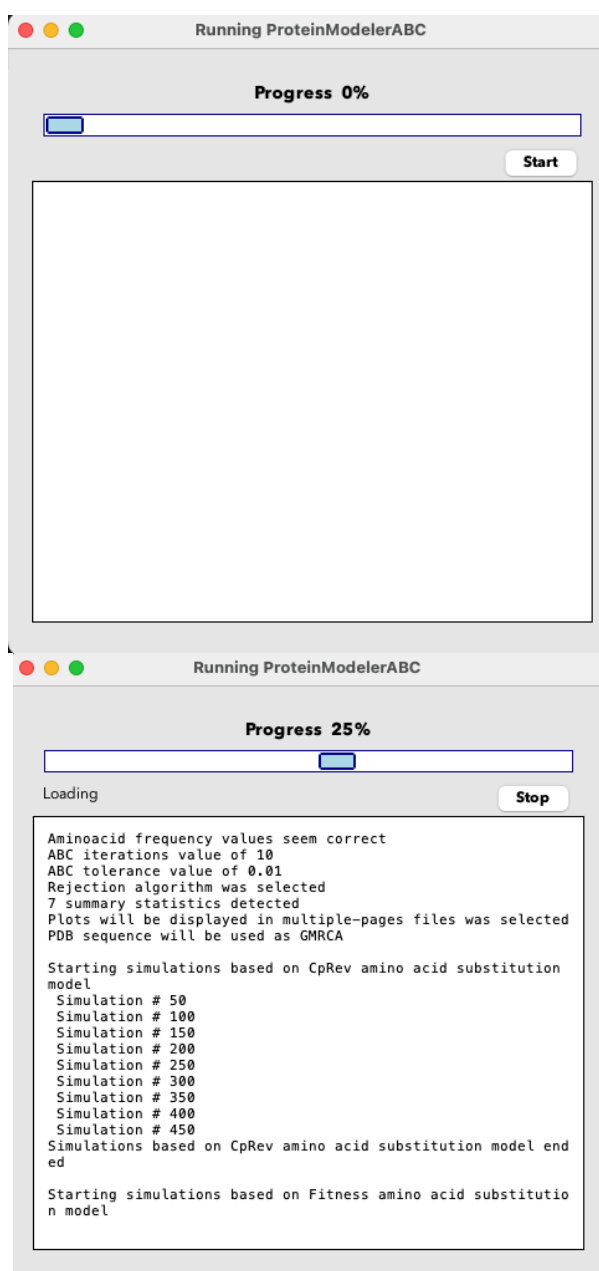
Figure 2. Execution window of the GUI.

4. Once *ProteinModelerABC* finished the analysis, a message will show up.
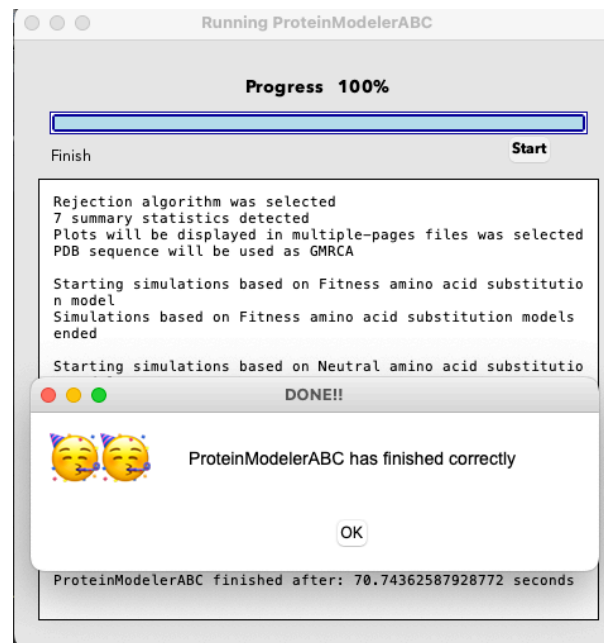
Figure 3. End of *ProteinModelerABC* execution.

5.1.2. Execution on the command line of a local computer

1. First, the user has to modify the *Settings.txt* file.

2. Once the input files are ready, in the *ProteinModelerABC* main directory the user should type "*python3 ProteinModelerABC.py*".

```
Test_ProteinModelerABC % ls
3on9.pdb                  GUI                       source
bin                       structures.in             Makefile
TNF.phy                   ProteinModelerABC.py      Settings.txt

Test_ProteinModelerABC % python3 ProteinModelerABC.py
```

3. Then, *ProteinModelerABC* reads the *Settings.txt* file and prints the main inputs.

```
_____System detection_____
Working in a Darwin machine


_____Checking Settings.txt_____
Alignment file exists
Indels are ignored
3on9.pdb selected as template
Chain A selected
Correct number of simulations
12 processors are selected
Simulated data is not saved
Running information will not be displayed on the screen
Coalescent method selected
Diploid data are selected
PopulationSize selected
Parameter values sampled from a uniform distribution of 0 - 1.3020833e-04
Parameter values sampled from a uniform distribution of 0 - 1.3020833e-04 seem correct
HIVw substitution model selected
Fitness substitution model selected
Neutral substitution model selected
```

```
Aminoacid frequency values seem correct
Thermodynamic temperature set at 1.78
Unfolded protein configurational entropy per residue set at 0.05
Misfolded protein configurational entropy per residue set at 0.05
Misfolded protein entropy offset set at 0.0
The population size to simulate molecular evolution under Fitness site-dependent SCS
model set at 10
ABC iterations value of 100
ABC tolerance value of 0.01
Rejection algorithm was selected
7 summary statistics detected
Plots will be displayed in multiple-pages files was selected
PDB sequence will be used as GMRCA
```

4. If any parameter is incorrect, *ProteinModelerABC* will stop the execution and print a message error (see section 5.5), for example:

```
Number of processors introduced is higher than your computer number of processors
Please check your computer number of processors
ERROR!! Please check NumberOfProcessors value
```

5. Next, the simulations under the substitution models defined by the user are launched:

```
_____Simulations_____
Starting simulations based on HIVw amino acid substitution model
 Simulation # 50
 Simulation # 100
 Simulation # 150
 Simulation # 200
 Simulation # 250
 Simulation # 300
 Simulation # 350
 Simulation # 400
 Simulation # 450
 Simulation # 500
 Simulation # 550
 Simulation # 600
 Simulation # 650
 Simulation # 700
 Simulation # 750
 Simulation # 800
 Simulation # 850
 Simulation # 900
 Simulation # 950
 Simulation # 1000
Simulations based on HIVw amino acid substitution models ended


Starting simulations based on Fitness amino acid substitution model
 Simulation # 1050
 Simulation # 1100
 Simulation # 1150
 Simulation # 1200
 Simulation # 1250
 Simulation # 1300
 Simulation # 1350
 Simulation # 1400
 Simulation # 1450
 Simulation # 1500
 Simulation # 1550
 Simulation # 1600
 Simulation # 1650
 Simulation # 1700
 Simulation # 1750
 Simulation # 1800
```

```
 Simulation # 1850
 Simulation # 1900
 Simulation # 1950
 Simulation # 2000
Simulations based on Fitness amino acid substitution models ended


Starting simulations based on Neutral amino acid substitution model
 Simulation # 2050
 Simulation # 2100
 Simulation # 2150
 Simulation # 2200
 Simulation # 2250
 Simulation # 2300
 Simulation # 2350
 Simulation # 2400
 Simulation # 2450
 Simulation # 2500
 Simulation # 2550
 Simulation # 2600
 Simulation # 2650
 Simulation # 2700
 Simulation # 2750
 Simulation # 2800
 Simulation # 2850
 Simulation # 2900
 Simulation # 2950
 Simulation # 3000
Simulations based on Neutral amino acid substitution models ended
```

6. Then, *ProteinModelerABC* calculates the SS of the query and simulated data:

```
_____Summary statistics calculation_____
Calculating TNF summary statistics
Calculating simulation # 50 summary statistics
Calculating simulation # 100 summary statistics
Calculating simulation # 150 summary statistics
Calculating simulation # 200 summary statistics
Calculating simulation # 250 summary statistics
Calculating simulation # 300 summary statistics
Calculating simulation # 350 summary statistics
Calculating simulation # 400 summary statistics
Calculating simulation # 450 summary statistics
Calculating simulation # 500 summary statistics
Calculating simulation # 550 summary statistics
Calculating simulation # 600 summary statistics
Calculating simulation # 650 summary statistics
Calculating simulation # 700 summary statistics
Calculating simulation # 750 summary statistics
Calculating simulation # 800 summary statistics
Calculating simulation # 850 summary statistics
Calculating simulation # 900 summary statistics
Calculating simulation # 950 summary statistics
Calculating simulation # 1000 summary statistics
Calculating simulation # 1050 summary statistics
Calculating simulation # 1100 summary statistics
Calculating simulation # 1150 summary statistics
Calculating simulation # 1200 summary statistics
Calculating simulation # 1250 summary statistics
Calculating simulation # 1300 summary statistics
Calculating simulation # 1350 summary statistics
Calculating simulation # 1400 summary statistics
Calculating simulation # 1450 summary statistics
Calculating simulation # 1500 summary statistics
Calculating simulation # 1550 summary statistics
Calculating simulation # 1600 summary statistics
```

```
Calculating simulation # 1650 summary statistics
Calculating simulation # 1700 summary statistics
Calculating simulation # 1750 summary statistics
Calculating simulation # 1800 summary statistics
Calculating simulation # 1850 summary statistics
Calculating simulation # 1900 summary statistics
Calculating simulation # 1950 summary statistics
Calculating simulation # 2000 summary statistics
Calculating simulation # 2050 summary statistics
Calculating simulation # 2100 summary statistics
Calculating simulation # 2150 summary statistics
Calculating simulation # 2200 summary statistics
Calculating simulation # 2250 summary statistics
Calculating simulation # 2300 summary statistics
Calculating simulation # 2350 summary statistics
Calculating simulation # 2400 summary statistics
Calculating simulation # 2450 summary statistics
Calculating simulation # 2500 summary statistics
Calculating simulation # 2550 summary statistics
Calculating simulation # 2600 summary statistics
Calculating simulation # 2650 summary statistics
Calculating simulation # 2700 summary statistics
Calculating simulation # 2750 summary statistics
Calculating simulation # 2800 summary statistics
Calculating simulation # 2850 summary statistics
Calculating simulation # 2900 summary statistics
Calculating simulation # 2950 summary statistics
Calculating simulation # 3000 summary statistics
```

7. Next, the ABC estimation (calculation of posterior probabilities) runs. Again, if the cross-validation step fails the framework will try to successfully perform the ABC estimation up to 10 times. In case it does not work, we recommend increasing the tolerance to a maximum of 0.05. If the error persists, we recommend repeating the analysis with more simulations (see section 5.1.1).

```
_____Executing ABC analysis_____
ABC estimation failed

Starting another try: 1
ABC estimation failed

Starting another try: 2
Try 2 succeeded
ABC analysis has finished
```

8. Finally, a message is printed to confirm that *ProteinModelerABC* finished:

```
_____Finish!_____
ProteinModelerABC has finished!!
ProteinModelerABC finished after: 2150.90 seconds
```

The execution information will be printed in the *ProteinModelerABC.out* file.

5.1.3.   Execution on the command line of a computer cluster

1. First, the user has to modify the *Settings.txt* file. Note that, although *ProteinModelerABC* is designed to work with any number of processors, it is highly recommended to use a number of simulations divisible by the number of processors to save computer time.

2. Once the input files are ready, from the main directory of *ProteinModelerABC* the user should type "*python ProteinModelerABC_Cluster.py*". For example,

```
python3 ProteinModelerABC_Cluster.py
```

3. Now, instead of launching the simulations and the SS calculations as in the other versions (i.e., GUI and terminal), some executables bash files are created. The user should focus on *launch_Simu.sh* file. It will be created following our cluster format so maybe it has to be adapted to the features of the users' computer cluster (possibly changing the first lines that either start with "#" or load modules needed for the execution, such as Python and mpi4py). We advise against changing the other lines that launch the simulations and the SS calculations. Then, the user has to submit the *launch_Simu.sh* file to Slurm (a task and cluster management system). The running information will be the same as in the other versions and will be written in a *slurm.out* file.

```
sbatch launch_Simu.sh
```

4. The ABC estimation will start automatically. Again, if the cross-validation step fails the framework will try to successfully perform the ABC estimation up to 10 times. In case it does not work, we recommend increasing the tolerance to a maximum of 0.05. If the error persists, we recommend repeating the analysis with more simulations (see section 5.1.1).

5.2. Output files

Aside from the running information file (*ProteinModelerABC.out* or *slurm.out*), several output files are generated by *ProteinModelerABC* during the process. These files are saved in the output folders *ABCOutputs* and *SimulationsOutputs* that will be placed in the working directory.

```
Test_ProteinModelerABC %ls
3on9.pdb                GUI                     SimulationsOutputs
ABCOutputs              source                  bin
Settings.txt            Makefile                structures.in
ProteinModelerABC.py    ProteinModelerABC.out   TNF.phy
```

5.2.1. *ABCOutputs* folder

This folder contains the information regarding the ABC estimations and all the necessary files to repeat the ABC estimation. A typical *ProteinModelerABC* run will produce:
- The file *Results_text.txt* shows the results of all the analysis performed. It contains the confusion matrix followed by an ABC method matrix that shows the corresponding mean model posterior probabilities. Next, the best-fitting substitution model estimation. When a regression ABC method (*mnlogistic* or *neuralnet*) is selected, it included the proportion of accepted simulations made by *rejection* algorithm followed by the estimation of posterior probability of every substitution model. When the *rejection* ABC method, it directly shows the estimation of posterior probability of every substitution model. Finally, it includes the p-value of the goodness of fit analyses followed by the distance between the accepted and observed SS.
- The script *ABCAnalysis.r*, which contains all the instructions to perform the ABC estimation phase and to create all the plots.
- The file *Histogram_Priors.pdf*, which shows a histogram of the prior distribution used for the substitution rate. Also, a theta histogram is provided.

o In case the user chooses all the available SS, two files are generated with the boxplots of the SS values distribution under the evaluated substitution models. Specifically, the file *Results_SS_Energy.pdf* includes two boxplots: one representing the mean stability of the simulated proteins (ΔG, kcal/mol), and the other one showing the corresponding standard deviation. The file *Results_SS_AAReplacements.pdf* shows five boxplots: one with the number of segregating sites, and the remaining with the mean, standard deviation, skewness and kurtosis of the Grantham distances of the simulated proteins. In case the user does not select all the SS, only one file is created (named *Results_SS.pdf*) with boxplots for the selected SS. In all the boxplots we superimposed the value of the query MSA SS value shown with a blue horizontal line. For more information about the SS see section 6.3.

o The file *Histograms_SStats.pdf* shows a histogram of the SS values from the retained simulations under the evaluated substitution models. Vertical blue lines correspond to the value of SS from the query MSA. Ideally the histograms should have a gaussian-like shape with the blue line in the centre of the distribution. These plots are used to assess how the model assumed for the simulations fits with the query data.

o The file *Histogram_GoodnessOfFit.pdf* shows a histogram for each evaluated substitution model representing the distance between the SS from the accepted simulations and the SS from the observed data (blue line). Also, the p-value is computed to test the fit of every substitution model to the observed data and showed in the file *Results_text*.

o The file *PCA.pdf* shows a plot of the two first principal components of the SS values of the evaluated substitution models. The black cross corresponds to the SS of the query MSA and the area inside the coloured lines represent the SS of the retained simulations. The black cross should fall inside of at least one model. This plot is also used to assess how the model assumed for the simulations fits with the query data.

o The file *Results_ConfusionMatrix_100sampSSSimulations.pdf* shows a plot representing the results from the confusion matrix, a specific table that summarizes the cross-validation step between the substitution models considered. The cross-validation considers a simulation as pseudo-observed data and try to assign it to the model with the highest posterior probability. This process is repeated as many times as the number of iterations specified (ABC iterations). The diagonal of the confusion matrix shows the number of simulations that were correctly classified. Note that the name of this output file can change depending on the number of ABC iterations selected (e.g., *Results_ConfusionMatrix_10sampSSSimulations.pdf* or *Results_ConfusionMatrix_1000sampSSSimulations.pdf)*.

o The file *SSSimulations.csv* file contains the SS computed from each simulated dataset.

o The file *SSRealData.csv* presents the SS computed from the query MSA.

o The file *PSimulations.txt* shows the parameter values sampled from the prior distribution of the substitution rate and its corresponding theta value used for the simulation.

5.2.2.   *SimulationsOutputs* folder

- o   A duplicate of the *SSSimulations.csv* file, which contains the SS values computed from each simulated dataset.
- o   A duplicate of the *SSRealData.csv* file, which presents the SS values computed from the query MSA.
- o   A duplicate of the *PSimulations.txt* file, which shows the values sampled from the prior distribution of the substitution rate and its corresponding theta value used for the simulation.

Additionally, if the user works with the *ProteinModelerABC_Cluster* version, the bash executable files created during the framework execution will be saved in this folder to allow the users to modify code to adapt it to user cluster if required.

5.3.   Re-analysing data

ProteinModelerABC allows different ABC estimations by changing the ABC variables (method, tolerance and number of iterations) in the script *ABCAnalysis.r,* placed in the *ABCOutputs* folder, without having to repeat the simulations and the SS calculation phases. For a complete description and example of usage on these settings see 4.2.4 and the provided examples in section 5.4, respectively. To re-analyse the data the user must edit the desired ABC parameters in the first lines of the *ABCAnalysis.r* script.

```
######################################################
################### ABC VARIABLES ################
######################################################
ABC_Method <- "rejection"
ABC_Tolerance <- 0.01
ABC_N_Iterations <- 100
######################################################
######################################################
######################################################
```

If the user needs to change the path to the *ABCOutputs* folder only one line has to be modified.

```
#Path
address<-paste("/Users_route_to_the_new_directory", sep="")
setwd(address)
###########################
```

5.4.   Examples

The package includes the following examples in the folder *Examples*, including all input and output files. All examples can be run in any *ProteinModelerABC* version, but they were run on the command line of cluster computer to save time. For all the examples, we previously selected the best-fitting empirical substitution model with *ProtTest3* (Darriba et al. 2011). Two additional examples were included to test the framework usage (folder *Fast-examples*). In the first one, we simulated the coalescent history while for the second we used an input phylogenetic tree.

1. **Example1-TNF_Pox:** Analysis of a real target alignment [10 sequences, 160 amino acids, 0.95 sequence identity] of the tumour necrosis factor receptor (TNF) of pox virus using 10,000 simulations comparing the HIVw and the two SCS models (Table 3). For the θ parameter we applied a uniform prior distribution ranging from 0 to 100 which produced alignments with a sequence identity that

encompass the query sequence identity (1.00-0.65) and performed the estimation with the rejection ABC method under a tolerance of 0.005. The posterior probability show that the Neutral model is the best fitting model for the MSA with a probability of 0.52, while the Fitness and HIVw models obtained a probability of 0.33 and 0.15 respectively (Table 3).

2. **Example2-Protease_HIV:** Analysis of a real target alignment [95 sequences, 99 amino acids, 0.91 sequence identity] of the protease of HIV-1 virus using 10,000 simulations comparing the JTT and the two SCS models (Table 3). We used a uniform prior distribution for θ ranging from 0 to 150 obtaining alignments with a sequence identity that encompass the query sequence identity (1.00-0.46) and performed the estimation with the rejection ABC method under a tolerance of 0.005. We obtained that the Fitness model fits with the MSA with a probability of 0.43, while the HIVw and Neutral models obtained a probability of 0.25 and 0.32 respectively (Table 3).

3. **Example3-GAG_HIV:** Analysis of a real target alignment [128 sequences, 288 amino acids, 0.69 sequence identity] of the protease of HIV-1 virus using 10,000 simulations comparing the RtRev and the two SCS models (Table 3). We used a uniform prior distribution for θ ranging from 0 to 500 obtaining alignments with a sequence identity that encompass the query sequence identity (1.00-0.41) and applied the rejection ABC method for the estimation under a tolerance of 0.005. The Fitness model is the best fitting model for the MSA with a probability of 0.48, while the Neutral and RtRev models obtained a probability of 0.27 and 0.25 respectively (Table 3).

4. **Example4-NS1_Flu:** Analysis of a real target alignment [25 sequences, 202 amino acids, 0.83 sequence identity] of the NS1 of influenza virus using 10,000 simulations comparing the JTT and the two SCS models (Table 3). We used a uniform prior distribution for θ ranging from 0 to 200 which produced alignments with a sequence identity that encompass the query sequence identity (1.00-0.54) and performed the estimation with the rejection ABC method under a tolerance of 0.005. The Neutral model is the one provided the best fit for the MSA with a probability of 0.74, while the Fitness and JTT models obtained a probability of 0.01 and 0.25 respectively (Table 3).

5. **Example5-C30_COVID:** Analysis of a real target alignment [30 sequences, 299 amino acids, 0.53 sequence identity] of the C30 endopeptidase of SARS-CoV virus using 10,000 simulations comparing the LG and the two SCS models (Table 3). We used a uniform prior distribution for θ of 0-500 which produced alignments with a sequence identity that encompass the query sequence identity (1.00-0.42) and applied the rejection ABC method for the estimation under a tolerance of 0.005. The Fitness model is the best-fitting model for the MSA with a probability of 0.86, while the LG and Neutral models obtained a probability of 0.03 and 0.11 respectively (Table 3).

6. **Example6- Methyltr-2 _COVID:** Analysis of a real target alignment [28 sequences, 298 amino acids, 0.62 sequence identity] of the SARS-CoV 2'-O-methyltransferase protein using 10,000 simulations comparing the LG and the two SCS models (Fitness and Neutral) (Table 3). We used a uniform prior distribution

for the θ parameter of 0-500 which produced alignments with a sequence identity that encompass the query sequence identity (1.00-0.42) and applied the rejection ABC method for the estimation under a tolerance of 0.005. We obtained that the Fitness model is the one which best fit the MSA with a probability of 0.65, while the LG and Neutral models obtained a probability of 0.14 and 0.21 respectively (Table 3).

7. **Example7-TIR:** Analysis of a real target alignment [23 sequences, 171 amino acids, 0.3 sequence identity] of the Toll-Interleukin receptor using 10,000 simulations comparing the WAG and the two SCS models (Table 3). We used a uniform prior distribution for the θ parameter of 0-700 obtaining alignments with a sequence identity that encompass the query sequence identity (1.00-0.25) and applied the rejection ABC method for the estimation under a tolerance of 0.005. We excluded the DGREM_sd (ID 2) from the ABC estimation. The Fitness model is the best-fitting model for the MSA with a probability of 0.98 while the Neutral and WAG models obtained a probability of 0.01 and 0.01 respectively (Table 3).

8. **Example8-TOM7:** Analysis of a real target alignment [54 sequences, 50 amino acids, 0.51 sequence identity] of the mitochondria membrane translocase using 10,000 simulations comparing the WAG and the two SCS models (Table 3). We used a uniform prior distribution for the θ parameters of 0-500 which produced alignments with a sequence identity that encompass the query sequence identity (1.00-0.14)) and applied the rejection ABC method for the estimation under a tolerance of 0.005. The Neutral model fits with a best-fit the MSA with a probability of 0.41, while the Fitness and WAG models obtained a probability of 0.33 and 0.26 respectively (Table 3).

9. **Example9-SE:** Analysis of a real target alignment [12 sequences, 450 amino acids, 0.66 sequence identity] of the squalene epoxidase using 10,000 simulations comparing the WAG and the two SCS models (Table 3). We used a uniform prior distribution for the θ parameters ranging from 0 to 500 which produced alignments with a sequence identity that encompass the query sequence identity (1.00-0.50)) and applied the rejection ABC method for the estimation under a tolerance of 0.005. The Fitness model is the best-fitting model for the MSA with a probability of 0.97 while the Neutral and WAG models obtained a probability of 0.03 and 0, respectively (Table 3).

10. **Example10-NP_Ebola:** Analysis of a real target alignment [28 sequences, 298 amino acids, 0.62 sequence identity] of the Ebola nucleoprotein using 10,000 simulations comparing the LG and the two SCS models (Table 3). We used a uniform prior distribution for the θ parameter of 0-500 obtaining produced alignments with a sequence identity that encompass the query sequence identity (1.00-0.57) and applied the rejection ABC method for the estimation under a tolerance of 0.005. The *Neutral* model is the best-fitting model for the MSA with a probability of 0.99, while the Fitness and LG models obtained a probability of 0 and 0.01 respectively (Table 3).

Table 3. Real protein families and their substitution model selection with *ProteinModelerABC*. For every studied protein family, the table shows the Pfam accession code (GenBank for the fist dataset, see table foot), the number of sequences and the sequence length, the sequence identity (average of pairwise sequence identities), the prior for the population substitution rate (including the derived approximate range of average pairwise sequence identity), a representative protein structure (PDB code), the best-fitting empirical substitution model selected with ProtTest3 and the probability of selecting every substitution model (empirical, site-dependent Fitness SCS and site-dependent Neutral SCS models) with ProteinModelerABC.

Protein family	Sequences database entry	Number of sequences and sequences length	Sequence identity	Prior for the population substitution rate and sequence identity	Template protein structure	Best-fitting empirical substitution model	Probability of substitution model selection		
Tumor necrosis factor monkeypox	*	10, 160	0.95	Uniform (0-100) (1.00-0.65)	3on9	HIVw	Fitness 0.33	HIVw 0.15	**Neutral** **0.52**
HIV protease	PS50175	95, 99	0.91	Uniform (0-150) (1.00-0.46)	1tcx	HIVb	**Fitness** **0.43**	HIVb 0.25	Neutral 0.32
HIV gag polyprotein	PF00540	128, 288	0.69	Uniform (0-500) (1.00-0.41)	1l6n	RtRev	**Fitness** **0.48**	Neutral 0.27	RtRev 0.25
Influenza NS1	PF00600	25, 202	0.83	Uniform (0-200) (1.00-0.54)	4oph	JTT	Fitness 0.01	JTT 0.25	**Neutral** **0.74**

Name	Pfam				Prior (Uniform)	PDB	Model	Best	2nd	3rd
Coronavirus endopeptidase C30	PF05409	30, 299	0.53		Uniform (0-500) (1.00-0.42)	1lvo	LG	**Fitness** 0.86	LG 0.03	Neutral 0.11
Coronavirus 2'-O-methyltransferase	PF06460	28, 298	0.62		Uniform (0-500) (1.00-0.42)	7c2i	LG	**Fitness** 0.65	LG 0.14	Neutral 0.21
Toll-Interleukin receptor domain	PF01582	23, 171	0.3		Uniform (0-700) (1.00-0.25)	5ku7	WAG	**Fitness** 0.98	Neutral 0.01	WAG 0.01
Mitochondria membrane translocase	PF08038	54, 50	0.51		Uniform (0-500) (1.00-0.14)	6ucv	WAG	Fitness 0.33	**Neutral** **0.41**	WAG 0.26
Squalene epoxidase	PF08491	12, 450	0.66		Uniform (0-500) (1.00-0.50)	6c6n	WAG	**Fitness** 0.97	Neutral 0.03	WAG 0
Ebola nucleoprotein	PF05505	8, 373	0.67		Uniform (0-500) (1.00-0.47)	6c54	LG	Fitness 0	LG 0.01	**Neutral** **0.99**

[*] GenBank accession numbers: AAB94354, AAB94356, AAB94388, ADZ29547, YP_010085450, AXN75227, AIE41152, AAB94364, URF91555 and AAB94363

5.5. Message errors and recommendations

ProteinModelerABC included two accessory programs, *ProteinEvolverProtABC* simulator (Arenas et al. 2013) and *DeltaGREM* software (Arenas et al. 2017), which are compiled through a *Makefile. DeltaGREM* includes "*libg2c*" library, which was commonly used in Fortran 77 programs on some Unix-like systems, but it has become outdated and is no longer commonly used. We recommend using the 9.3.0 GNU Fortran (GCC) compiler with which we obtained a successfully compilation.

Errors generated from incorrect settings at running the program on the command line, are usually shown on the screen and the program will suggest aborting the execution by typing CTRL+C. To reduce the probability of errors, we recommend using an input file *Settings.txt* from the *Examples* folder and carefully edit it as desired or to use the GUI version.

The input MSA must be presented in standard *phylip* sequential format. Importantly, the amino acid sequences of the input MSA should only include any of the 20 amino acids (one-letter code) or indels (as "-"). Other letters or symbols (e.g., X or $) will produce an error displayed on the screen.

```
Alignment file exists
Error in alignment file: $ character was found in line 2, position 375
Error in alignment file: $ character is not allowed
```

Next, *ProteinModelerABC* checks the inputs from *Settings.txt*. If any of them is incorrect (e.g., integer when string is expected or parameters out of limits), the *ProteinModelerABC* will stop the execution and print the incorrect parameter.

```
Coalescent method selected
Haploid/Diploid information value are incorrect
ERROR!! Please check Haploid/Diploid value
```

Through the simulation phase, some of the simulations may fail due to intrinsic problems of substitution models, particularly for the Fitness SCS model. Conveniently, after the last simulation *ProteinModelerABC* will check if any simulation failed and will automatically rerun each failed simulation if necessary. During the ABC procedure the estimation may fail due to a low value of the tolerance, a high number of iterations or under the multinomial logistic regression (*mnlogistic*) or the neural networks (*neuralnet*) methods (see section 5.3). If the ABC estimation cannot be carried out completely, no matter the reason, *ProteinModelerABC* will launch an error message. In this situation, the ABC estimation can be executed again (section 5.3) varying the ABC tolerance, iterations or method in the R script. Alternatively, one can increase the number of simulations. This may be enough to run successfully the ABC estimation, but implies repeat the whole *ProteinModelerABC* execution with the new number of simulations. Concerning the ABC method, the rejection approach can be more robust (it rarely fails) than the others to analyze real data.

If you find any unexpected error, or there is any doubt, do not hesitate to contact ferreirogarciadavid@gmail.com. Thanks for your contribution!

6. Models and Methods

6.1. Phylogenetic History and Evolutionary Models

The first step of *ProteinModelerABC* is based on the simulator *ProteinEvolverProtABC*, developed from the program *ProteinEvolver* (https://github.com/MiguelArenas/proteinevolver) (Arenas et al. 2013) and adapted to perform ABC. The molecular evolution in *ProteinEvolverProtABC* is simulated forward in time along a phylogeny. Concerning that phylogeny, the user can either specify a particular phylogenetic tree or simulate a coalescent history (Kingman 1982; Hudson 2002). The implemented coalescent simulation, it includes a variety of population evolutionary models (i.e., variation of the population size over time, migration and temporal longitudinal sampling or tip dates [see, Navascués et al., 2010]). Concerning the user-specified phylogenetic tree, this option is recommended when the user already has a phylogenetic tree.

The empirical substitution models for the protein sequences implemented in the software are: Blosum62 (Henikoff and Henikoff 1992), CpRev (Adachi et al. 2000), Dayhoff (Dayhoff et al. 1978), DayhoffDCMUT (Kosiol and Goldman 2005), HIVb (Nickle et al. 2007), HIVw (Nickle et al. 2007), JTT (Jones et al. 1992), JonesDCMUT (Kosiol & Goldman, 2005), LG (Le and Gascuel 2008), Mtart (Abascal et al. 2007), Mtmam (Yang et al. 1998), Mtrev24 (Adachi and Hasegawa 1996), RtRev (Dimmic et al. 2002), VT (Müller and Vingron 2000), WAG (Whelan and Goldman 2001) models. The user can also specify another exchangeability matrix and amino acid frequencies at the equilibrium through an input file. In addition, the framework implements heterogeneity of the substitution rate among sites according to a gamma distribution (+G) and the proportion of invariable sites (+I) for the empirical substitution models (Yang 1996). On the other hand, protein sequences can also evolve under SCS models. We included the site-dependent Neutral (which does not consider the population size) and Fitness (which requires a user-specified population size for the Moran process that evaluates the probability of rejecting/accepting mutation events) SCS models (Arenas et al. 2013). In practice, the Fitness model produced amino acid distributions more similar to the real observations than those obtained with the Neutral model for some protein families, while the Neutral model was in general more robust than the Fitness model to analyse diverse data. See Arenas et al. (2013) for additional information.

6.2. ABC Methods

Different ABC estimation methods are implemented in *ProteinModelerABC*: the rejection method (*rejection*), the multinomial logistic regression method (*mnlogistic*), and the neural networks-based method (*neuralnet*). Regarding the *rejection* method the posterior probability of a given model is approximated by the proportion of accepted simulations given this model. Concerning the two implemented regression methods (*mnlogistic* and *neuralnet*), the model indicator is treated as the response variable of a polychotomous regression, where the summary statistics are the independent variables. However, while the *mnlogistic* assumes a linear regression, the *neuralnet* method considers a non-linear regression and allow to reduce the dimension of the set of summary statistics, thus it can be particularly useful if many SS are used. These two regression methods could fail during the ABC estimation if the tolerance is below 0.05 (not enough simulations were retained).

6.3. Summary Statistics

ProteinModelerABC includes a total of 7 SS (Table 4). Two of them are related with the protein folding stability. The protein folding stability is quantified with the framework, *DeltaGREM* (Arenas et al. 2015) by the Gibbs free energy difference between folded and unfolded states (ΔG, kcal/mol). In the parameters section we recommended to use by default some structural parameters (e.g., thermodynamic temperature and configurational entropies) that showed a more realistic modelling of proteins (Arenas et al. 2015). In addition, *ProteinModelerABC* also considers as SS the number of amino acid segregating sites, and the mean, standard deviation, skewness and kurtosis of the Grantham distance between amino acids at every protein site (Grantham 1974).

Table 4. Summary statistics implemented in *ProteinModelerABC*. The table includes a numeric identifier (ID) for each SS and a brief description.

ID	Name	Description
1	*DGREM_mean*	Mean of folding stability (free energy) of the proteins of the dataset.
2	*DGREM_sd*	Standard deviation of folding stability among the proteins of the dataset.
3	*SegSites*	Number of segregating sites (amino acids).
4	*Grantham_mean_Position*	Mean of the Grantham distance of amino acid replacements at every protein site.
5	*Grantham_sd_Position*	Standard deviation of the Grantham distance of amino acid replacements at every protein site.
6	*Grantham_sk_Position*	Skewness of the Grantham distance of amino acid replacements for every protein site.
7	*Grantham_ku_Position*	Kurtosis of the Grantham distance of amino acid replacements for every protein site.

7. Acknowledgments and funding information

8. References

Abascal F, Posada D, Zardoya R. 2007. MtArt: A New Model of Amino Acid Replacement for Arthropoda. *Mol. Biol. Evol.* 24:1–5.

Adachi J, Hasegawa M. 1996. Programs for Molecular Phylogenetics Based on Maximum Likelihood. *Comput. Sci. Monogr.* 28:1–150.

Adachi J, Waddell PJ, Martin W, Hasegawa M. 2000. Plastid Genome Phylogeny and a Model of Amino Acid Substitution for Proteins Encoded by Chloroplast DNA. *J. Mol. Evol.* 50:348–358.

Arenas M. 2015. Trends in substitution models of molecular evolution. *Front. Genet.* 6:319.

Arenas M. 2022. ProteinEvolverABC: coestimation of recombination and substitution rates in protein sequences by approximate Bayesian computation. *Bioinformatics* 38:58–64.

Arenas M, Bastolla U. 2019. ProtASR2: Ancestral reconstruction of protein sequences accounting for folding stability. *Methods Ecol. Evol.* 11:248–257.

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

Arenas M, Sánchez-Cobos A, Bastolla U. 2015. Maximum-Likelihood Phylogenetic Inference with Selection on Protein Folding Stability. *Mol. Biol. Evol.* 32:2195–2207.

Arenas M, Weber CC, Liberles DA, Bastolla U. 2017. ProtASR: An Evolutionary Framework for Ancestral Protein Reconstruction with Selection on Folding Stability. *Syst. Biol.* 66:1054–1064.

Arnold K, Bordoli L, Kopp J, Schwede T. 2006. The SWISS-MODEL workspace: a web-based environment for protein structure homology modelling. *Bioinformatics* 22:195–201.

Beaumont MA. 2010. Approximate Bayesian Computation in Evolution and Ecology. *Annu. Rev. Ecol. Evol. Syst.* 41:379–406.

Branco C, Kanellou M, González-Martín A, Arenas M. 2022. Consequences of the Last Glacial Period on the Genetic Diversity of Southeast Asians. *Genes* 13:384.

Carvajal-Rodriguez A. 2006. Recombination Estimation Under Complex Evolutionary Models with the Coalescent Composite-Likelihood Method. *Mol. Biol. Evol.* 23:817–827.

Challis CJ, Schmidler SC. 2012. A Stochastic Evolutionary Model for Protein Structure Alignment and Phylogeny. *Mol. Biol. Evol.* 29:3575–3587.

Csilléry K, François O, Blum MGB. 2012. abc: an R package for approximate Bayesian computation (ABC): *R package: abc. Methods Ecol. Evol.* 3:475–479.

Darriba D, Taboada GL, Doallo R, Posada D. 2011. ProtTest 3: fast selection of best-fit models of protein evolution. *Bioinformatics* 27:1164–1165.

Dayhoff MO, Schwartz RM, Orcutt BC. 1978. A model of evolutionary change in proteins. In: Atlas of Protein Sequence and Structure. Vol. 5. Dayhoff, M.O. Edition. Washington DC: National Biomedical Research Foundation. p. 345–352.

Dereeper A, Guignon V, Blanc G, Audic S, Buffet S, Chevenet F, Dufayard J-F, Guindon S, Lefort V, Lescot M, et al. 2008. Phylogeny.fr: robust phylogenetic analysis for the non-specialist. *Nucleic Acids Res.* 36:W465–W469.

Dimmic MW, Rest JS, Mindell DP, Goldstein RA. 2002. rtREV: An Amino Acid Substitution Matrix for Inference of Retrovirus and Reverse Transcriptase Phylogeny. *J. Mol. Evol.* 55:65–73.

Edgar RC. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* 32:1792–1797.

García-Portugués E, Golden M, Sørensen M, Mardia KV, Hamelryck T, Hein J. 2018. Toroidal diffusions and protein structure evolution. Available from: http://arxiv.org/abs/1804.00285

Golden M, García-Portugués E, Sørensen M, Mardia KV, Hamelryck T, Hein J. 2017. A Generative Angular Model of Protein Structure Evolution. *Mol. Biol. Evol.* 34:2085–2100.

Grantham R. 1974. Amino Acid Difference Formula to Help Explain Protein Evolution. *Science* 185:862–864.

Henikoff S, Henikoff JG. 1992. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.* 89:10915–10919.

Herman JL, Challis CJ, Novák Á, Hein J, Schmidler SC. 2014. Simultaneous Bayesian Estimation of Alignment and Phylogeny under a Joint Model of Protein Sequence and Structure. *Mol. Biol. Evol.* 31:2251–2266.

Hudson RR. 2002. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics* 18:337–338.

Jones DT, Taylor WR, Thornton JM. 1992. The rapid generation of mutation data matrices from protein sequences. *Bioinformatics* 8:275–282.

Kingman JFC. 1982. The coalescent. *Stoch. Process. Their Appl.* 13:235–248.

Kosiol C, Goldman N. 2005. Different Versions of the Dayhoff Rate Matrix. *Mol. Biol. Evol.* 22:193–199.

Le SQ, Gascuel O. 2008. An Improved General Amino Acid Replacement Matrix. *Mol. Biol. Evol.* 25:1307–1320.

Leuenberger C, Wegmann D. 2010. Bayesian Computation and Model Selection Without Likelihoods. *Genetics* 184:243–252.

Lopes JS, Arenas M, Posada D, Beaumont MA. 2014. Coestimation of recombination, substitution and molecular adaptation rates by approximate Bayesian computation. *Heredity* 112:255–264.

Müller T, Vingron M. 2000. Modeling Amino Acid Replacement. *J. Comput. Biol.* 7:761–776.

Navascués M, Depaulis F, Emerson BC. 2010. Combining contemporary and ancient DNA in population genetic and phylogeographical studies. *Mol. Ecol. Resour.* 10:760–772.

Nickle DC, Heath L, Jensen MA, Gilbert PB, Mullins JI, Kosakovsky Pond SL. 2007. HIV-Specific Probabilistic Models of Protein Evolution. *PLoS ONE* 2:e503.

Norn C, André I, Theobald DL. 2021. A thermodynamic model of protein structure evolution explains empirical amino acid substitution matrices. *Protein Sci.* 30:2057–2068.

Perron U, Kozlov AM, Stamatakis A, Goldman N, Moal IH. 2019. Modeling Structural Constraints on Protein Evolution via Side-Chain Conformational States. *Mol. Biol. Evol.* 36:2086–2103.

Sousa VC, Beaumont MA, Fernandes P, Coelho MM, Chikhi L. 2012. Population divergence with or without admixture: selecting models using an ABC approach. *Heredity* 108:521–530.

Waksman G, Krishna TS, Williams CH, Kuriyan J. 1994. Crystal structure of Escherichia coli thioredoxin reductase refined at 2 A resolution. Implications for a large conformational change during catalysis. *J. Mol. Biol.* 236:800–816.

Waterhouse A, Bertoni M, Bienert S, Studer G, Tauriello G, Gumienny R, Heer FT, de Beer TAP, Rempfer C, Bordoli L, et al. 2018. SWISS-MODEL: homology modelling of protein structures and complexes. *Nucleic Acids Res.* 46:W296–W303.

Whelan S, Goldman N. 2001. A General Empirical Model of Protein Evolution Derived from Multiple Protein Families Using a Maximum-Likelihood Approach. *Mol. Biol. Evol.* 18:691–699.

Yang Z. 1996. Among-site rate variation and its impact on phylogenetic analyses. *Trends Ecol. Evol.* 11:367–372.

Yang Z, Nielsen R, Hasegawa M. 1998. Models of amino acid substitution and applications to mitochondrial protein evolution. *Mol. Biol. Evol.* 15:1600–1611.