Documentation for ***ProteinModelerABC***

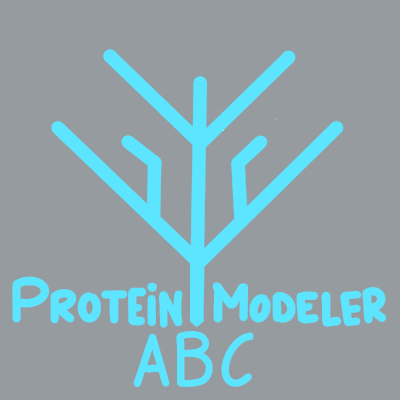
*Selection of the best-fitting substitution model of protein evolution accounting for structural constrains for protein sequences alignments by approximate Bayesian computation*

Current version 1.0

© 2023. David Ferreiro

[david.ferreiro.garcia@uvigo.es](mailto:david.ferreiro.garcia@uvigo.es) – [ferreirogarciadavid@gmail.com](mailto:ferreirogarciadavid@gmail.com)

08/02/2023

**

**Table of contents**

[Disclaimer 4](#_Toc125374534)

[Credits 4](#_Toc125374535)

[1. Purpose 4](#_Toc125374536)

[2. Versions and graphical user interface 5](#_Toc125374537)

[2.1. *ProteinModelerABC* 5](#_Toc125374538)

[2.2. *ProteinModelerABC\_Cluster* 5](#_Toc125374539)

[3. Compilation 6](#_Toc125374540)

[4. *ProteinModelerABC* usage 7](#_Toc125374541)

[4.1. Before starting 7](#_Toc125374542)

[4.2. The Settings input file 8](#_Toc125374543)

[4.2.1. Available prior distributions 9](#_Toc125374544)

[4.2.2. General input data and information 9](#_Toc125374545)

[4.2.3. Settings for the simulation phase 10](#_Toc125374546)

[4.2.4. Estimation phase 14](#_Toc125374547)

[5. *ProteinModelerABC* example 14](#_Toc125374548)

[5.1. Execution 14](#_Toc125374549)

[5.1.1. Execution using GUI 15](#_Toc125374550)

[5.1.2. Execution using computer terminal 18](#_Toc125374551)

[5.1.3. Execution on Cluster 22](#_Toc125374552)

[5.2. Output files 22](#_Toc125374553)

[5.2.1. ABCOutputs folder 23](#_Toc125374554)

[5.2.2. SimulationsOutputs folder 24](#_Toc125374555)

[5.3. Re-analysing data 24](#_Toc125374556)

[5.4. Examples 24](#_Toc125374557)

[5.5. Message errors and recommendations 29](#_Toc125374558)

[6. Models, Methods and Summary Statistics 30](#_Toc125374559)

[6.1. Evolutionary Models and Phylogenetic tree 30](#_Toc125374560)

[6.2. ABC Methods 30](#_Toc125374561)

[6.3. Summary Statistics 31](#_Toc125374562)

[7. *ProteinModelerABC* Performance 31](#_Toc125374563)

[7.1. Assumptions and Limitations 35](#_Toc125374564)

[8. Acknowledgments 35](#_Toc125374565)

[9. References 36](#_Toc125374566)

**Table of figures**

[**Figure 1. Parameters values selection window of the GUI.** 17](#_Toc125375676)

[**Figure 2. Execution window of the GUI.** 18](#_Toc125375677)

[**Figure 3. *ProteinModelerABC* finish execution message.** 18](#_Toc125375678)

[**Figure 4. Evaluation of substitution model selection in ProteinModelerABC under different number of simulations and tolerance levels.** 33](#_Toc125375679)

[**Figure 5. Evaluation of the estimation of the best-fitting substitution model under different number of simulations and tolerance thresholds for different true substitution models.** 34](#_Toc125375680)

[**Figure 6. Computer time of a ProteinModelerABC analysis using different numbers of local cores.** 35](#_Toc125375681)

# Disclaimer

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version (at your option) of the License. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place – Suite 330, Boston, MA 02111-1307, USA.

# Credits

This program was developed at,

* *Department of Biochemistry, Genetics and Immunology, University of Vigo, 36310 Vigo, Spain.*
* *Biomedical Research Centre (CINBIO), University of Vigo, 36310 Vigo, Spain.*

# Purpose

*ProteinModelerABC* is an evolutionary framework to estimate the best-fitting substitution model with approximate Bayesian computation (ABC) from protein sequence alignments. *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 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 it on clusters (*ProteinModelerABC\_Cluster*) to save computer time and resources.

The ABC approach requires computer simulations, which are 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 a total of 7 summary statistics (SS), which explore the stability of the individual sequences of the alignment and the physicochemical properties of the amino acids’ replacements. Conveniently, *ProteinModelerABC* can run the simulations on parallel to save computer time. This is highly recommended because simulations under substitution models that consider the stability of the protein and its calculation require a long time.

Three ABC methods are implemented in *ProteinModelerABC*: (*i*) the *rejection* algorithm which uses the proportion of accepted simulations to estimate the posterior probabilities of the model(s) (Csilléry et al., 2012); (*ii*) the *mnlogistic*, in which the posterior model probabilities are estimated using a multinomial logistic regression; and (*iii*) using *neuralnet*, which uses neural networks to predict the posterior probabilities of the models based on the observed SS (Beaumont, 2010; Csilléry et al., 2012). For all, the user must specify the number of simulations to use, the proportion of simulations to retain (tolerance), the number of iterations and which SS to use. *ProteinModelerABC* provides also several diagnostic plots to qualitatively assess the fitting of the data with the model(s), namely: boxplots of the SS calculated for the evaluated model(s); histogram of the distance between the values of the SS from the retained simulations and the observed for the model(s); a histogram for each evaluated substitution model representing the distance between the SS from the accepted simulations (bars) and the SS from the observed data (blue line); and a plot of the two principal components analysis (PCA) from the SS calculated from simulations. In all the plots, the values for the target protein alignment data are superimposed in blue except for PCA plot, where the target protein alignment data value is represented by a black cross.

# Versions and graphical user interface

The folder available from GitHub includes two versions of the *ProteinModelerABC* framework, one to run on the local computer (*ProteinModelerABC* folder) and another to run in a cluster (*ProteinModelerABC\_Cluster*).

## *ProteinModelerABC*

**GUI**

The user can run the framework in their computer using the command lines or using a Python graphical user interface (GUI), which allow the user to work with the program without using the command line. The GUI will display two consecutive windows. The first one allows the user to set the mandatory parameters and to fill the optional if intended (see section 4.1). Settings must be carefully specified. Even though, some incorrect specifications may be automatically detected by the framework and produce errors (e.g., it does not allow negative substitution rates), user-specified settings without biological meaning may still produce results (see section 4.2). The second window will launch the simulations, SS calculation and the consecutive ABC analyses. While the program is running, it will also inform about the progress and show a message when it ends (see section 4.3.2).

**Command line**

GUI and command line version works in the same way. However, in this version the user has to change manually the value of the parameters in the input file (Settings.txt). Importantly, the user has to maintain the format of the file (see section 4.2). Remove or add any characters may cause *ProteinModelerABC* to fail and such changes may be difficult to detect. We recommend using one of the Settings.txt files of the *Examples* folder and carefully change the values. Once the framework was executed it will check the input and launch simulations, SS calculation and the consecutive ABC analyses. Some information about the progress will be printed on the command line (see section 4.3.2).

## *ProteinModelerABC\_Cluster*

This version of *ProteinModelerABC* can run on clusters if they are based on Linux OS environments. It enables the user to work with many more processors in parallel without sharing memory, hence taking advantage of MPI interface. Similarly to the command line version, the user must manually fill the Settings file considering the biological meaning of the input values and maintaining the format of the file (see section 4.2). *ProteinModelerABC\_Cluster* execution will check the inputs and calculate the SS of the multiple sequence alignment (MSA), but won’t launch the simulations and their corresponding SS calculation. Instead, some batch scripts will be created, and to run the simulations and their corresponding SS calculation the user has to submit a master batch script called *launch\_Simu.sh* to Slurm (a task and cluster management system). Once the analysis finished, the same folders and output files as in a local computer execution are created (see section 4.4).

# Compilation

To install the framework the user has to use the computer terminal, reach the main directory of *ProteinModelerABC* and execute the “*make all*” command. It will compile 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) and the GUI through a Makefile file. The user must not modify any file nor folder because they contain the necessary files for a successfully compilation and execution. The executable file to launch the GUI will be in the *executable* folder in the *GUI* folder, while the resulting executable files for the accessory programs will be placed in a *bin* folder.

make all

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 (Table 1). Python libraries can be installed by the command: pip install *library\_name* in 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 needed to run *ProteinModelerABC.*

|  |  |  |
| --- | --- | --- |
| *Name* | *Language* | *Version* |
| abc | R | Command line and GUI |
| 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 |
| tkmacosx | Python | GUI |
| time | Python | GUI |

# *ProteinModelerABC* usage

*ProteinModelerABC* have four mandatory input files:

* **Settings file:** This file, named *Settings.txt*, must contain all the desired specifications for simulations and ABC estimation. It has to be carefully specified, incorrect parameters values will be detected (e.g., it does not allow negative substitution rates) but user-specified settings without biological meaning may still produce results.
* **Multiple alignment of protein sequences:** The protein MSA that will be analyzed must be provided by the user in a sequential *phylip* format (.phy). It requires that each sequence identifier is exactly 10 characters long, padded with spaces when necessary. In addition, working with template structures demands that the MSA has the same length as the template chain. So, sequences are aligned with the template chain sequence and, the positions that present a gap in the template sequence are removed both from the sequences and the template sequence. To ease this, we provide a python script (*Align.py*, placed in the *Scripts* folder, see section 4.1), which we recommend using. Yet, other programs can also be used, such as *MUSCLE* (Edgar, 2004) to perform the alignment or *Phylogeny.fr* (Dereeper et al., 2008) to change to *phylip* format (<http://phylogeny.lirmm.fr/phylo_cgi/data_converter.cgi>). In this case, the user must check all the process and the number of characters of the sequences identifier to avoid errors.
* **Template structure:** This is a protein structure (.pdb file) which has the highest homology with the protein MSA. We advise to use *SWISS-MODEL* (Waterhouse et al., 2018) to find the best template (<https://swissmodel.expasy.org>). *SWISS-MODEL* works with a maximum of 10 sequences, so we include the *Find-WT.py* python script in the *Scripts* folder, which return a wild-type sequence (WT; a sequence composed by the most common amino acid per site) to find the template in *SWISS-MODEL* that better represents the alignment (see section 4.1.).
* **Structures:** This file (*structures.in*) contains structural information needed for the stability calculation (specifically, to calculate the DGREM\_Mean and the DGREM\_sd) and for the MSA simulations under SCS models. It cannot be modified.

## Before starting

Before starting a *ProteinModelerABC* analysis the user must do some extra work. First, although the framework uses *phylip* format, the user needs the protein alignment in *fasta* format. Then, we recommend to use *SWISS-MODEL* (Waterhouse et al., 2018) to find the best template (<https://swissmodel.expasy.org>). If the MSA includes more than 10 sequences, *SWISS-MODEL* will fail, in which case we recommend using the *Find-WT.py* python script in the *Scripts* folder, which returns the WT sequence.

python3 FindWT.py --input MSA.fasta

Next, the WT sequence will be used to find the template in *SWISS-MODEL* that better represents the alignment. Usually, the user must download the first template of the results screen, but it is recommended to check the X-ray when possible, to avoid high values. Working with template structures requires the MSA to have the same length as the template chain. Thus, sequences have to be aligned with the template chain sequence and positions that have a gap in the template sequence have to be removed in every sequence. The *Align.py* script placed in the folder *Scripts* will make the alignment, remove the gap positions and change the file into *phylip* format. Another option could be using programs as *MUSCLE* (Edgar, 2004) for sequences alignment or *Phylogeny.fr* (Dereeper et al., 2008) to convert into *phylip* format. *Align.py* works with an input *fasta* MSA (--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

*ProteinModelerABC* aim is to include SCS models in an evolutionary framework to estimate the best-fitting substitution model. Thus, it is not the best option to estimate the best-fitting substitution model between empirical models. The recommended use of *ProteinModelerABC* is to **compare the two SCS models with an empirical substitution model previously selected for the alignment** using other programs, such as *ProtTest3* (Darriba et al., 2011). To perform the simulations the user must design a prior distribution for the substitution rate per site. This can be an unfamiliar measure, so we also include a script that for a desired theta (*θ*) value returns the sequences identity of the MSA and the substitution rate per site, asking the ploidy of the organism and the population size. Substitution rate prior should be chosen considering the alignment sequence identity to ensure that it encompass the real data. Sequence identity will give an idea about the *θ* value, if it is very high (>70%), a low *θ* value should be used (<300) while if it is low (<50%), we recommend a high *θ* value (>500).

python3 Theta.py --input NS1.phy

The recommended structure of a *ProteinModelerABC* analysis consists on a folder located at a user-defined path, including the Settings.txt file, the MSA, the template .*pdb*, the *structures.in* file and all *ProteinModelerABC* material (including executable files and folders). Examples well prepared for a standard *ProteinModelerABC* run 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 the ABC parameters on the estimations.

## The Settings input file

The Settings input file must contain all the information required to perform the analysis. It is highly recommended to carefully handle the file, since mistakes may alter 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, the general input data, the settings for the simulation phase and the settings for the estimation phase.

**Important notes when handling 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 contain an “*\**” (see below).

- Some parameters require a prior distribution (see section 4.2.1), specifically a distribution of the parameters values used to perform the simulations which should include the presumed value (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).

### Available prior distributions

Several distributions are included in *ProteinModelerABC* to simulate protein data (Table 2). In addition, most of them can be truncated at lowest and highest values. However, note that some parameters must be specified by an integer number (e.g., generation time) while others by a float number (e.g., amino acid frequencies). Therefore, not all distributions can be applied to all the parameters. Details for each parameter are described in the following subsections.

Table 2. Available 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 |

### General input data and information

* **Name of the file with the query MSA.** This specification is mandatory. Only specify the filename (i.e., no path) since the target alignment file needs to be placed in the same directory as the Settings.txt file (see section 4).

# Target alignment file. 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 *NewState*. This decision can affect the SS if there are indels in the multiple sequence alignment.

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

\*Indels=Ignored

* **Template.** This specification is mandatory. Query MSA representative PDB protein structure used to structural substitution models and to calculate proteins free energy (see section 4).

# PDB protein structure. PDB protein structure used to structural substitution models simulations and to calculate proteins free energy -- MANDATORY --

\*Template=3IXO.pdb

* **Chain.** This parameter is mandatory. PDB protein chain used to structural substitution models simulations and to calculate proteins free energy.

# Chain of the PDB protein structure used. See documentation for details -- MANDATORY --

\*Chain=A

### Settings for the simulation phase

* + - 1. **General simulation settings**
* **Number of simulations.** This parameter is mandatory. We recommend at least 1,000 computer simulations but 10,000 simulations should provide accurate results. Still, the number of simulations required to obtain accurate estimates depends on many factors, especially the target MSA. More complex sequences (large molecular diversity) may require more simulations due to the irregularity of the parametric landscape that requires more sampling.

# Total number of simulations -- MANDATORY --

\*NumberOfSimulations=10000

* **Number of processors to run the simulations in parallel on a machine.** This parameter is mandatory (by default, when using the GUI version the simulations run on all the 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.

# Number of available processors to run the simulations in parallel. See documentation for details -- MANDATORY --

\*NumberOfProcessors=12

* **Save simulated data.** This parameter is mandatory. It is recommended not to save the simulated data because it requires a lot of space in the hard disk of the user’s machine. If the user wants 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 step.

# Save simulated data. It is recommended do not save the simulated data because it requires a lot of space -- MANDATORY --

\*SaveSimulations=No

* **Show running information.** This parameter is mandatory. If the user chooses “*No*” the amount of information printed on the screen during the execution will be reduced. This option is recommended because printing information on the screen requires more computational resources. Regardless of the option selected, the results will not change, it only affects the amount of information displayed on the screen during the execution.

# Show running information (simulations and summary statistics) on the screen It will increase the computer time. See documentation for details -- MANDATORY --

\*ShowInformationScreen=No

* + - 1. **Evolutionary history**

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

* **Coalescent history or rooted phylogenetic tree.** This parameter is mandatory. *Coal* will require some additional parameters information while *Phylo* asks for the phylogenetic tree input file name.

# Perform the coalescent history of the alignment. “Coal”: coalescent simulation, “Phylo”: user-specified phylogenetic tree. See documentation for details -- MANDATORY –

\*CoalescentOrPhylogeny=Coal

**Coalescent**

* **Haploid/diploid simulated data.** This parameter is mandatory if coalescent is specified. 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. Allowed distributions: *fix*, *uniform*, *norm*(t), *exp*(t), *gamma*(t), *beta*(t) (Table 2).

# Amino acid substitution rate: fix, uniform, gamma, beta, normal or 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 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 years**. In the example, 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. Note that 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 (in years) 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). These parameters are looking back in time, so it is not recommended to specify a negative growth rate for the last period, as the coalescent time could become infinite in the past. For an exponential growth per individual per generation, after “*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 “*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 chose a growth model based on demographic periods with 3 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 island-continent model, deme #1 will be the continent while the other demes will be islands. In this example the user chose a stepping-stone model, two demes with three samples each.

# 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 chose only 1 period with migration rate = 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 set 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 this 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: the first and the second number are the numbers of demes to converge, while the third number is 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 this next example there are 3 convergence events, between deme 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**

* **Amino acid substitution rate per site.** This parameter is mandatory. Allowed distributions: *fix*, *uniform*, *norm*(t), *exp*(t), *gamma*(t), *beta*(t) (Table 2).

# User-specified phylogenetic tree/s. See documentation for details -- MANDATORY IF PHYLOGENETIC --

\*Tree=phylotree.txt

* + - 1. **Substitution model**
* **Substitution model of amino acid evolution.** This parameter is mandatory. The user has to specify at least one structurally constrained substitution (SCS) model (Fitness or Neutral) and the desired empirical substitution model of protein evolution from the following: Blosum62, CpRev, Dayhoff, DayhoffDCMUT, HIVb, HIVw, JTT, JonesDCMUT, LG, Mtart, Mtmam, Mtrev24, RtRev, VT, WAG, UserEAAM. We **highly recommend to use only the best-fitting empirical substitution model for the alignment** which can be previously obtained using other programs, such as *ProtTest* (Darriba et al., 2011). **Models should be separated by a space and if any empirical substitution model is specified it should be written before SCS models.**

# Model of amino acid substitution. Select at least one structurally constrained substitution (SCS) model (Fitness or Neutral) and the desired empirical substitution model (i.e., Blosum62, CpRev, Dayhoff, DayhoffDCMUT, HIVb, HIVw, JTT, JonesDCMUT, LG, Mtart, Mtmam, Mtrev24, RtRev, VT, WAG, UserEAAM). The empirical models should be specified before SCS if any is selected. See documentation for details -- MANDATORY --

\*SubstitutionModel=HIVw Fitness Neutral

* **Amino acid frequencies.** This parameter is mandatory. Frequencies for each amino acid site along the sequences for the empirical substitution models. Allowed distributions: fix or dirichlet. If it is not specified the program will assume equally distributed 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).** 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: fix, uniform, gamma, beta, normal, exponential

RateHetSites=uniform 0.76 0.90

* **Proportion of invariable sites (+I).** This parameter is optional for empirical substitution models. Allowed distributions: *fix*, *uniform*, *norm*(t), *exp*(t), *gamma*(t), *beta*(t).

# Proportion of invariable sites, +I # fix, uniform, gamma, beta, normal, exponential

PropInvSites=uniform 0.3 0.5

### Estimation phase

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

# ABC iterations. Number of simulations to consider (Iterations < NumberOfSimulations). See documentation for details -- MANDATORY --

\*ABCIterations=100

* **ABC tolerance.** This parameter is mandatory. The proportion of simulations that generated SS close to the real SS to be retained to perform the ABC procedure. A tolerance of 0.01 may be enough but the value of tolerance that gives the best results may 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 user 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=0.01

* **ABC method.** This parameter is mandatory. The chosen ABC algorithm to perform the analysis. The user can choose between *rejection*, *mnlogistic* or *neuralnet*.

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

\*ABCMethod=rejection

* **Summary statistics.** This parameter is mandatory. The user needs to choose the SS to use for the ABC estimation by specifying their numeric ID (see Table 4 and section 5 *Models and Methods* for further details).For initial exploratory analyses, we recommend to use all the available SS. Later, by analysing the output plots the user can inspect which suit better to the data. For example, in cases where the folding stability standard deviation (SS 2) is far from the real data value we recommend to disregard it.

# Summary statistics to use. See documentation for details -- MANDATORY --

\*SummaryStatistics= 1 2 3 4 5 6 7

* **Multiple pages.** This parameter is mandatory. PDF documents with multiple plots per page (No, Yes).

# Multiple pages. PDF documents with multiple pages -- MANDATORY --

\*MultiPage=Yes

# *ProteinModelerABC* example

## Execution

The data shown below corresponds with the normal output information of a *ProteinModelerABC* run. For the first phase of the program, the user has to use the computer terminal. The example corresponds to a rapid project, in which the goal is to distinguish between the empirical HIVw and the SCS Fitness and Neutral models. Before trying this example, Python 3 and R must be installed and the framework compiled (see section 3).

A complete execution may take a lot of time, particularly when considering SCS models, large protein datasets, numerous simulations or calculating SS related with the protein stability. To reduce the execution time, the user can run the simulations and the SS calculation in parallel on a Linux environment. For that, the user needs to specify it in the settings file and, either work in a machine with many processors or use a computer cluster. Note that if any simulation setting value is unexpected the analysis may fail (see section 5.5).

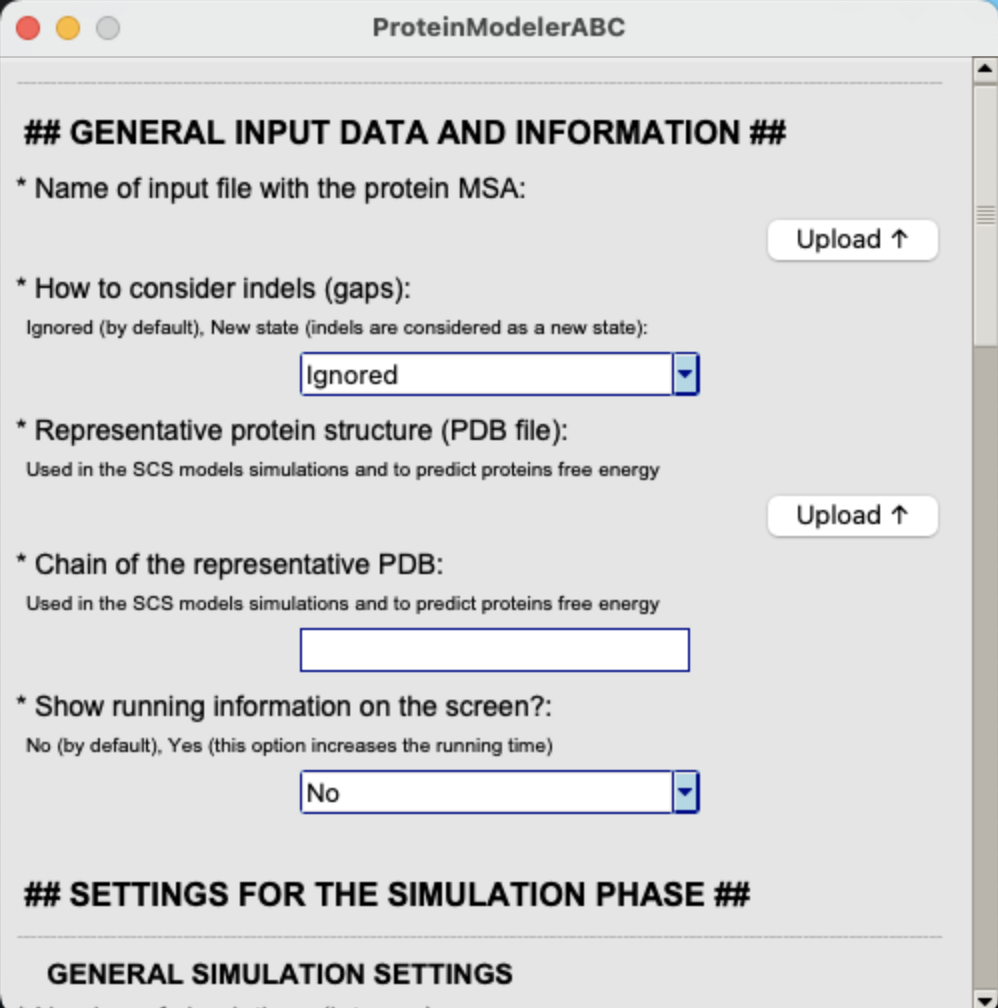
To execute *ProteinModelerABC* all the python scripts (.*py*) must be placed jointly with the mandatory input files (Settings.txt, the multiple alignment of protein sequence, template structure and *structures.in*). *ProteinModelerABC* was developed using python3.9 version and tested under most of python3 versions so it is recommended to use python3.9 version but every version python3 is expected to work. Then, depending on the version the user must:

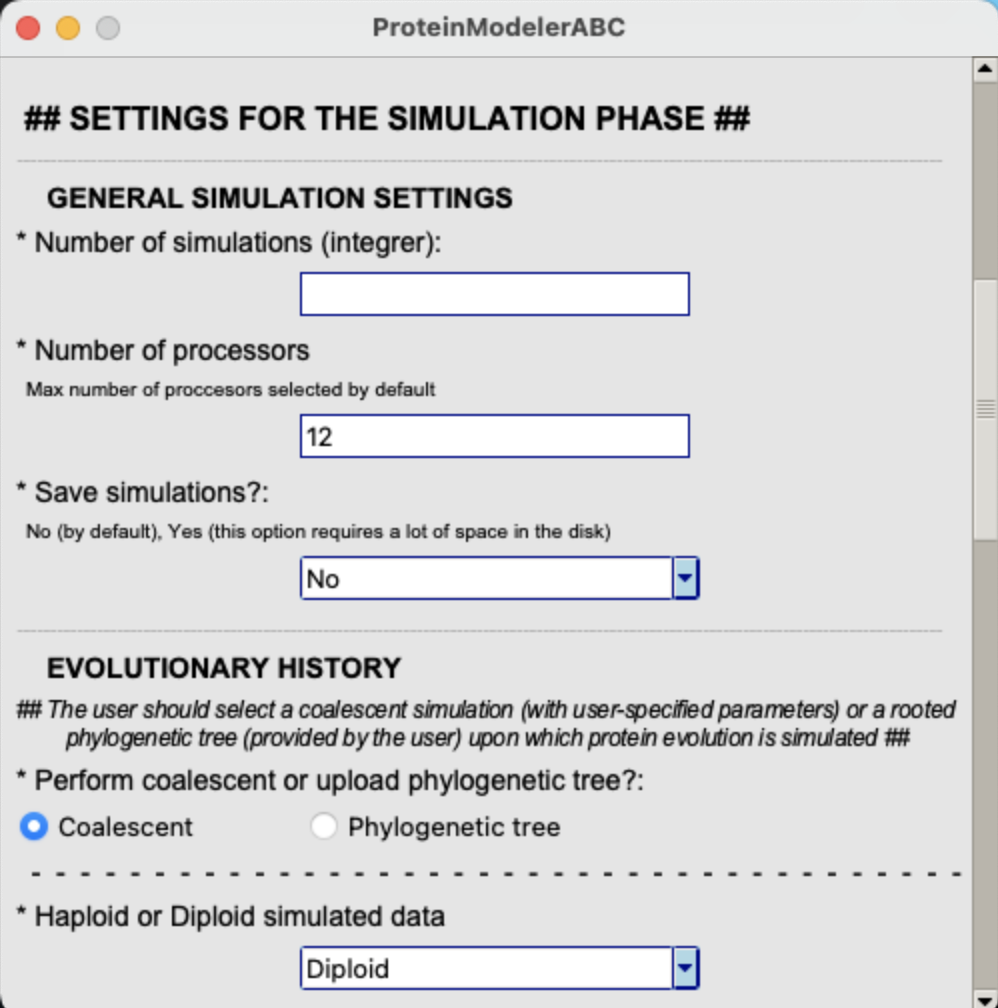
### Execution using GUI

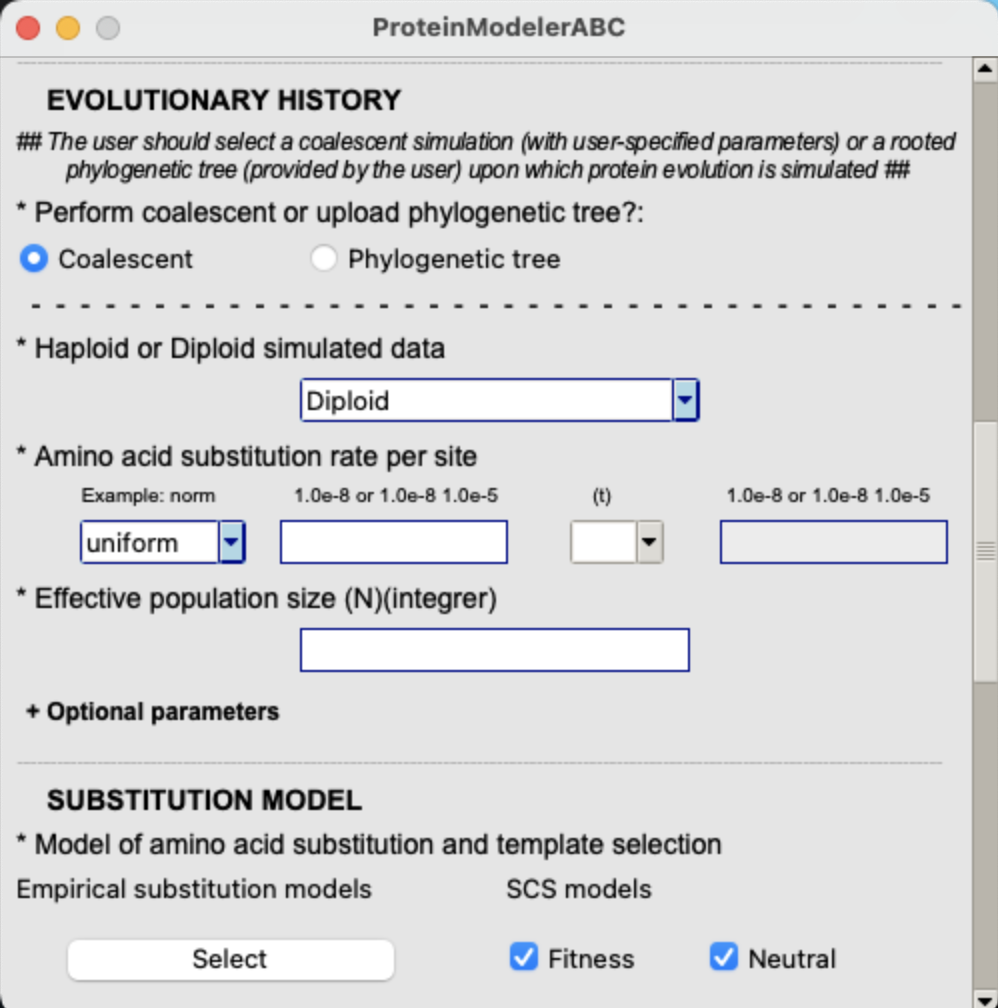
1. To open *ProteinModelerABC* GUI the user can go to the *GUI* directory and type “python3 *ProteinModelerABC\_GUI.py*” in the command line, or go to the *Executable* directory and click the executable *ProteinModelerABC\_GUI*. Remember that GUI *.py* or executable do not have to be in the GUI folder, they can be moved but always with the corresponding images in the case of the *.py* file.

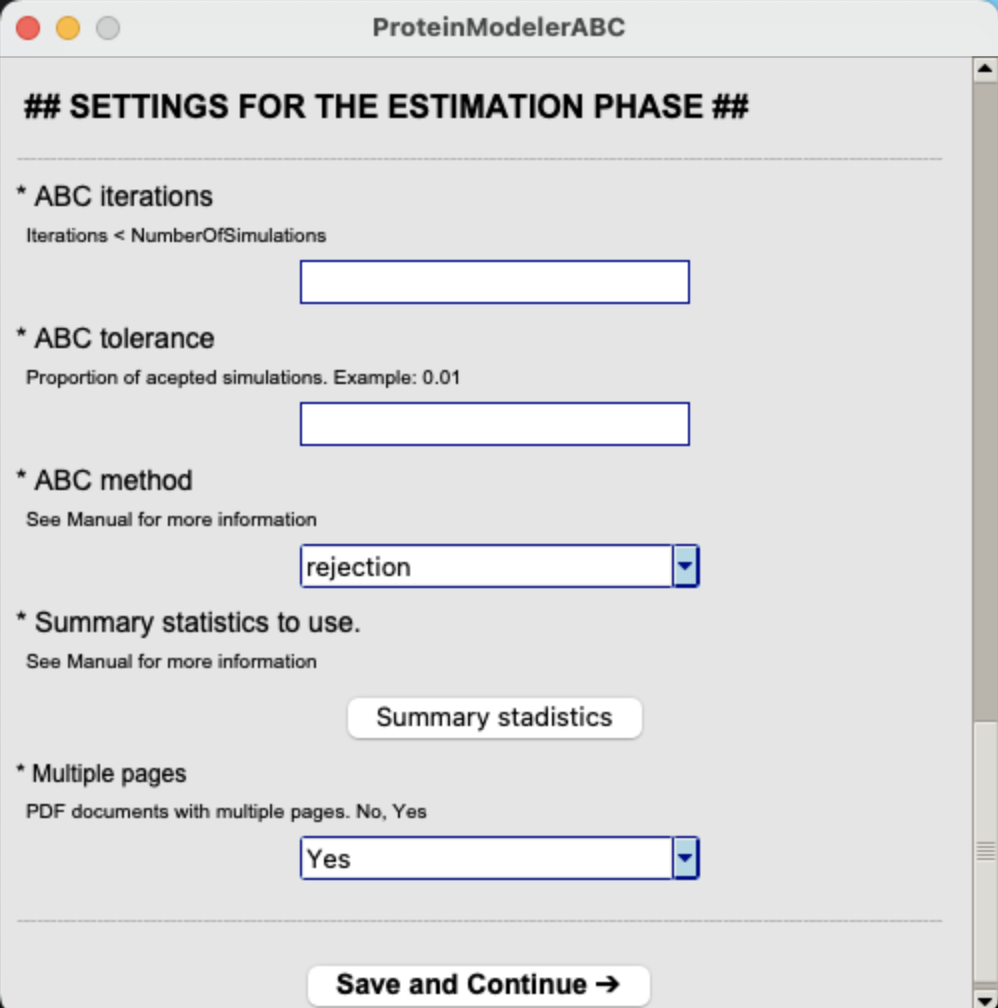
python3 ProteinModelerABC\_GUI.py

1. Now, the user fills 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* will show up warnings if the user forgets to fill an entry or does it wrong, but user-specified settings without biological meaning won’t be detected and may produce incorrect estimations. See section 4.2 for more information.

****

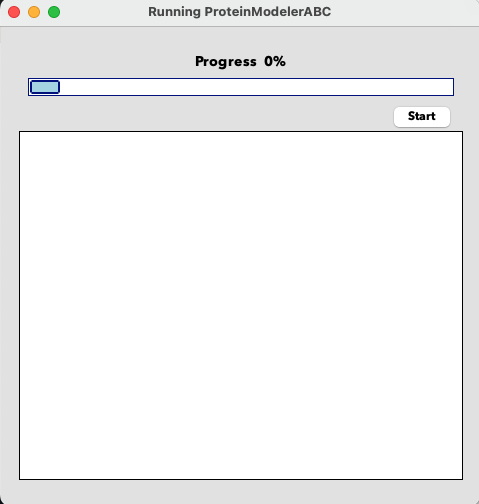
****

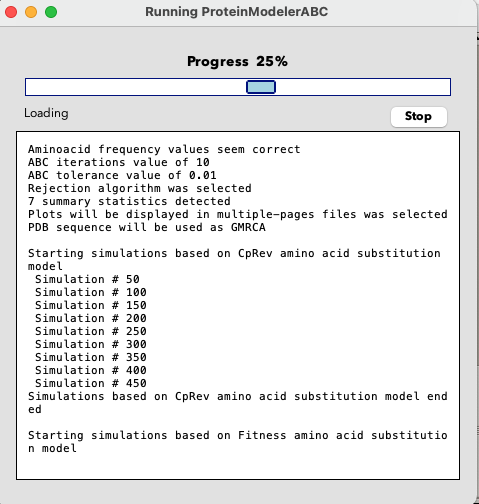




**Figure 1. Parameters values selection window of the GUI.**

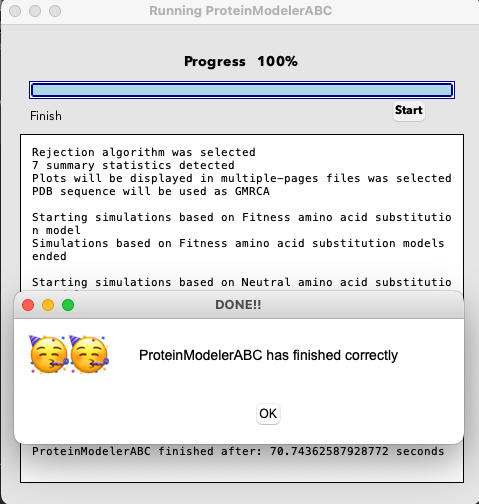
1. A new window will appear and to launch the simulation and the estimation phase the user has to click on the *Star*t button. This window will display all the *ProteinModelerABC* execution information.

****



**Figure 2. Execution window of the GUI.**

1. Once *ProteinModelerABC* is done, a new message will show up.

****

**Figure 3. *ProteinModelerABC* finish execution message.**

### Execution using computer terminal

1. First, the user has to modify the *Settings.txt* file. Again, we recommend not to change anything more than the parameters values in the file.
2. Once the input files are ready, inside the *ProteinModelerABC* main directory the user types “*python3 ProteinModelerABC.py*”.

Test\_ProteinModelerABC % **ls**

3on9.pdb **GUI** **source**

**bin**  LeerSettings.py structures.in

ChangeVariablesPE.py Makefile TNF.phy

Errores.py ProteinModelerABC.py Variables.py

Functions.py Settings.txt

Test\_ProteinModelerABC % **python3 ProteinModelerABC.py**

1. Then, *ProteinModelerABC* reads the *Settings.txt* file the printes main inputs.

Alignment file exists

Correct number of simulations

Indels are ignored

Simulated data is saved

Running information will not be displayed on the screen

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

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

12 processors are selected

Aminoacid frequency values seem correct

3on9.pdb selected as template

Chain A selected

GMRCA not selected

PDB sequence will be used as GMRCA instead your input sequences in SCS models

1. 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

1. Next, the simulations under the substitution model defined by the user are launched:

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

1. Then, *ProteinModelerABC* will calculate the SS of the observed 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

1. Next, the ABC estimation will be launched. Sometimes, the cross-validation step may fail due to a low tolerance value, a low number of simulations and/or a high iterations value. However, repeating the analysis generally generates the results. Indeed, the software will try to successfully perform the ABC estimation up to 10 times. In case it doesn’t success, we recommend repeating the analysis considering more simulations. If the user doesn’t select enough simulations the estimation will fail continuously, even after increasing the tolerance or decreasing the number of iterations.

Executing ABC analysis

ABC estimation failed

Starting another try: 1

ABC estimation failed

Starting another try: 2

Try 2 succeeded

ABC analysis has finished

1. Finally, a message will be printed to confirm that *ProteinModelerABC* finished:

ProteinModelABC has finished!!

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

### Execution on Cluster

1. First, the user has to modify the *Settings.txt* file. Again, we recommend to not change anything aside from the parameter’s values in the 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.
2. Once the input files are ready, from the main directory of *ProteinModelerABC* the user should type “*python ProteinModelerABC\_Cluster.py*”.

python3 ProteinModelerABC\_Cluster.py

1. Now, instead of launching the simulations and the SS calculations as in the other versions (i.e., GUI and terminal), some executables files are created. The user only has to focus on *launch\_Simu.sh* file. It will be created following the CESGA cluster format so maybe it has to be adapted to the features of the user’s computer cluster (possibly changing the first lines that either start with “*#*” or load CESGA 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. 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

1. 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 doesn’t success, we recommend repeating the analysis considering more simulations. If the user doesn’t select enough simulations the estimation will fail continuously, even after increasing the tolerance or decreasing the number of iterations.

## Output files

Apart from the running information file (*ProteinModelerABC.out* or *slurm.out*), several output files are generated by *ProteinModelerABC* during the different stages of the estimation. These files are saved in the output folders *ABCOutputs* and *SimulationsOutputs* that will be placed in the working directory. The script *ABCAnalysis.r* within the folder *ABCOutputs* contains all the instructions to perform the ABC estimation phase and to create all the plots.

|  |  |  |
| --- | --- | --- |
| Test\_ProteinModelerABC %**ls** |  |  |
| 3on9.pdb | GUI | **SimulationsOutputs** |
| **ABCOutputs** | LeerSettings.py | source |
| bin | Makefile | structures.in |
| ChangeVariablesPE.py | **ProteinModelerABC.out** | TNF.phy |
| Errores.py | ProteinModelerABC.py | Variables.py |
| Functions.py | Settings.txt |  |

### *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 *Histogram\_Priors.pdf*, which shows a histogram of the prior distributions used for the simulations of substitution rates. Also, a *θ* histogram is provided.
* In case the user chooses to calculate all the available SS, two files displaying the boxplots of the distribution of the SS calculated under the evaluated substitution model(s) are generated. Specifically, the file *Results\_SS\_Energy.pdf* displays two boxplots: one representing the mean stability of the simulated proteins (ΔG kcal/mol), and the other 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, skeaness and the 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 SS of the target sequences alignment (blue horizontal line). For more information about the SS see section 5.2.
* The file *Histograms\_SStats.pdf* shows a histogram of the SS values from the retained simulations under the evaluated substitution model(s). Vertical blue lines correspond to the value of the SS of the target sequences alignment. 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 if the model assumed for the simulations fits well the data.
* The file *Histogram\_GoodnessOfFit.pdf* shows a histogram for each evaluated substitution model representing the distance between the SS from the accepted simulations (bars) 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*.
* The file *PCA.pdf* shows a plot of the two first principal components of a PCA of the SS values of the evaluated substitution model(s). The black cross corresponds to the target sequences alignment and the area inside the coloured lines represent the retained simulations. The black cross should fall inside of at least one model. This plot is also used to assess if the model assumed for the simulations fits well the data.
* The file *Results\_ConfusionMatrix\_100sampSSSimulations.pdf* shows a plot representing the results from the confusion matrix, a specific table that allows the visualization of the performance of the chosen ABC algorithm to distinguish between models. Note that the name will change depending on the number of ABC iterations selected (e.g., *Results\_ConfusionMatrix\_10sampSSSimulations.pdf or Results\_ConfusionMatrix\_1000sampSSSimulations.pdf).*
* The file *Results\_text.txt* shows the results of all the analysis performed. Firstly, the confusion matrix followed by an ABC method matrix, then the best-fitting substitution model estimation followed by a bayes factors matrix, and finally the p-value of the goodness of fit analyses followed by simulations distances.
* The file *SSSimulations.csv* file contains the SS values computed from each simulated data.
* The file *SSRealData.csv* presents the SS values computed from the target MSA.
* The file *PSimulations.txt* shows the values sampled from the prior distribution of the substitution rate and its corresponding *θ* value.

### *SimulationsOutputs* folder

* A copy of the *SSSimulations.csv* file.
* A copy of the *SSRealData.csv* file.
* A copy of the *PSimulations.txt* file.
* The simulated alignments are compressed into the file *Simulations.tar.gz* if the user specified in the settings file the option of save the simulations.

*ProteinModelerABC* executables files created during the framework execution will be placed in this folder to allow user to modify code to adapt it to their corresponding clusters.

## Re-analysing data

*ProteinModelerABC* allows to explore different settings for the ABC estimation editing 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 steps. 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 simply edit the desired ABC parameters values in the first lines of the *ABCAnalysis.r* script. This is common to all the versions of the software (i.e., GUI, terminal or computer cluster).

#####################################################

################### ABC VARIABLES ###################

#####################################################

ABC\_Method <- "rejection"

ABC\_Tolerance <- 0.01

ABC\_N\_Iterations <- 100

#####################################################

#####################################################

#####################################################

If the user needs to change the path only one line has to be edited.

#Path

address<-paste("/**User\_route\_to\_the\_new\_directory**", sep="")

setwd(address)

############################

## 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 we run them on cluster. For all the examples, we previously selected the best-fitting empirical substitution model with *ProtTest3* (Darriba et al., 2011).

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 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 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 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 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 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 od 0-500 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 300-700 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 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 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 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. Substitution model selection performed with ProteinModelerABC for the studied real protein families. For every studied protein family, the table shows the accession code, the number of sequences and the sequence length of the dataset, the sequence identity, the prior for the population substitution rate (θ) and the corresponding approximate range of sequence identity (SeqID), a representative protein structure (PDB code) of the dataset, the best-fitting empirical substitution model selected with ProtTest3, the probability of selecting every substitution model (empirical, Fitness site-dependent SCS or Neutral site-dependent SCS model) and the execution time of the analysis.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Protein family*** | ***Sequences database entry*** | ***Number of sequences; sequences length*** | ***Sequence identity*** | ***Prior for the population substitution rate (θ); (SeqID)*** | ***Template protein structure*** | ***Best-fitting empirical substitution model*** | ***Probability of substitution model selection*** | | | ***Ejecution time (hours)*** |
| Tumor necrosis factor monkeypox | self-created1 | 10, 160 | 0.95 | Uniform | 3on9 | HIVw | Fitness | HIVw | **Neutral** | 0.22 |
| (0-100) | 0.33 | 0.15 | **0.52** |
| (1.00-0.65) |
| HIV protease (PR) | PS50175 | 95, 99 | 0.91 | Uniform | 1tcx | HIVb | **Fitness** | HIVb | Neutral | 1.53 |
| (0-150) | **0.43** | 0.25 | 0.32 |
| (1.00-0.46) |
| HIV gag polyprotein | PF00540 | 128, 288 | 0.69 | Uniform | 1l6n | RtRev | **Fitness** | Neutral | RtRev | 15.45 |
| (0-500) | **0.48** | 0.27 | 0.25 |
| (1.00-0.41) |
| Influenza NS1 | PF00600 | 25, 202 | 0.83 | Uniform | 4oph | JTT | Fitness | JTT | **Neutral** | 0.43 |
| (0-200) | 0.01 | 0.25 | **0.74** |
| (1.00-0.54) |
| Coronavirus  endopeptidase C30 | PF05409 | 30, 299 | 0.53 | Uniform | 1lvo | LG | **Fitness** | LG | Neutral | 0.48 |
| (0-500) | **0.86** | 0.03 | 0.11 |
| (1.00-0.42) |
| Coronavirus 2'-O-methyltransferase | PF06460 | 28, 298 | 0.62 | Uniform | 7c2i | LG | **Fitness** | LG | Neutral | 0.45 |
| (0-500) | **0.65** | 0.14 | 0.21 |
| (1.00-0.42) |
| Toll-Interleukin receptor domain | PF01582 | 23, 171 | 0.3 | Uniform | 5ku7 | WAG | **Fitness** | Neutral | WAG | 0.43 |
| (0-700) | **0.98** | 0.01 | 0.01 |
| (1.00-0.25) |
| Mitochondria membrane translocase | PF08038 | 54, 50 | 0.51 | Uniform | 6ucv | WAG | Fitness | **Neutral** | WAG | 0.59 |
| (0-500) | 0.33 | **0.41** | 0.26 |
| (1.00-0.14) |
| Squalene epoxidase | PF08491 | 12, 450 | 0.66 | Uniform | 6c6n | WAG | **Fitness** | Neutral | WAG | 0.21 |
| (0-500) | **0.97** | 0.03 | 0 |
| (1.00-0.50) |
| Ebola nucleoprotein | PF05505 | 8, 373 | 0.67 | Uniform | 6c54 | LG | Fitness | LG | **Neutral** | 0.17 |
| (0-500) | 0 | 0.01 | **0.99** |
| (1.00-0.47) |

1Sequences accession number in the GenBank: AAB94354, AAB94356, AAB94388, ADZ29547, YP\_010085450, AXN75227, AIE41152, AAB94364, URF91555 and AAB94363

## Message errors and recommendations

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

The simulation phase can be computational expensive with long running times, in particular, when performing many simulations or simulating (or analyzing) long and/or large number of sequences. Therefore, reducing the number of simulations and the size of the input alignment of protein sequences can dramatically reduce the computer time. However, sometimes this is not possible, in which case we recommend parallelizing the simulations (if the machine has more than one processor) to reduce the computer times. Some of the simulations may fail due to intrinsic problems of substitution models, particularly for the Fitness model. Conveniently, after the last simulation *ProteinModelerABC* will check if some of them failed and will automatically rerun each failed simulation.

During the ABC procedure the estimation may fail due to a low value of the tolerance, a high number of iterations or when *mnlogistic* or *neuralnet* methods are selected (see section 5.4). If the ABC estimate 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 5.4) 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](mailto:ferreirogarciadavid@gmail.com). Thanks for your contribution!

# Models, Methods and Summary Statistics

*ProteinModelerABC* is a framework to the estimate the best-fitting substitution model of protein MSA using an ABC procedure. The simulator includes a variety of evolutionary models that may help to mimic real processes. Informative SS are calculated from the simulated data and compared with the same SS calculated from the real protein sequence alignment. Finally, the computation of the posterior distributions is performed using ABC methods.

## Evolutionary Models and Phylogenetic tree

The first step of *ProteinModelerABC* is based on the simulator *ProteinEvolverProtABC*, which in turn is an version of the program *ProteinEvolver* (<https://github.com/MiguelArenas/proteinevolver>) (Arenas et al., 2013) adapted to perform ABC. The molecular evolution in *ProteinEvolverProtABC* is simulated forward in time along the phylogeny. The user can either specify a particular phylogenetic tree or simulate a coalescent history (Hudson, 2002; Kingman, 1982). Concerning the latter, it includes a variety of evolutionary models and generates protein sequence alignments collected at same or different times (temporal longitudinal sampling or tip dates [see, Navascués et al., 2010]) from a population evolved. The empirical substitution models for the protein sequences implemented in the software are: Blosum62 (Henikoff & Henikoff, 1992), CpRev (Adachi et al., 2000), Dayhoff (Dayhoff et al., 1978), DayhoffDCMUT (Kosiol & Goldman, 2005), HIVb (Nickle et al., 2007), HIVw (Nickle et al., 2007), JTT (Jones et al., 1992), JonesDCMUT (Kosiol & Goldman, 2005), LG (Le & Gascuel, 2008), Mtart (Abascal et al., 2007), Mtmam (Yang et al., 1998), Mtrev24 (Adachi & Hasegawa, 1996), RtRev (Dimmic et al., 2002), VT (Müller & Vingron, 2000), WAG (Whelan & Goldman, 2001) and UserEAAM models (Arenas, 2015). Additionally, protein sequences can also evolve under SCS models, suing mean-field Neutral (which does not consider population size) and mean-field Fitness (which needs user specified population size) (Arenas et al., 2015). Both haploid and diploid data can be simulated. Furthermore, the program also implements heterogeneity across sites according to a gamma distribution (+G) and proportion of invariable sites (+I) (Yang, 1996).

## 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*), they both have an additional step in which they consider the closeness of the simulated SS to the observed SS. However, while the *mnlogistic* assumes a linear regression, the *neuralnet* method considers a non-linear regression and allow users to reduce the dimension of the set of summary statistics, thus it can be particularly useful if many SS are used.

## Summary Statistics

*ProteinModelerABC* includes a total of 7 SS (Table 4). Two of them are related with the proteins folding stability, and is measured with another framework named *DeltaGREM* (Arenas et al., 2017) using the Gibbs free energy difference between folded and unfolded states (ΔG, kcal/mol). Moreover, the *ProteinModelerABC* also uses the number of amino acids segregating sites, and the mean, standard deviation, skeaness and kurtosis of Grantham distance between amino acids replacements per protein site. It assesses the difference between replaced amino acids (Grantham, 1974).

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

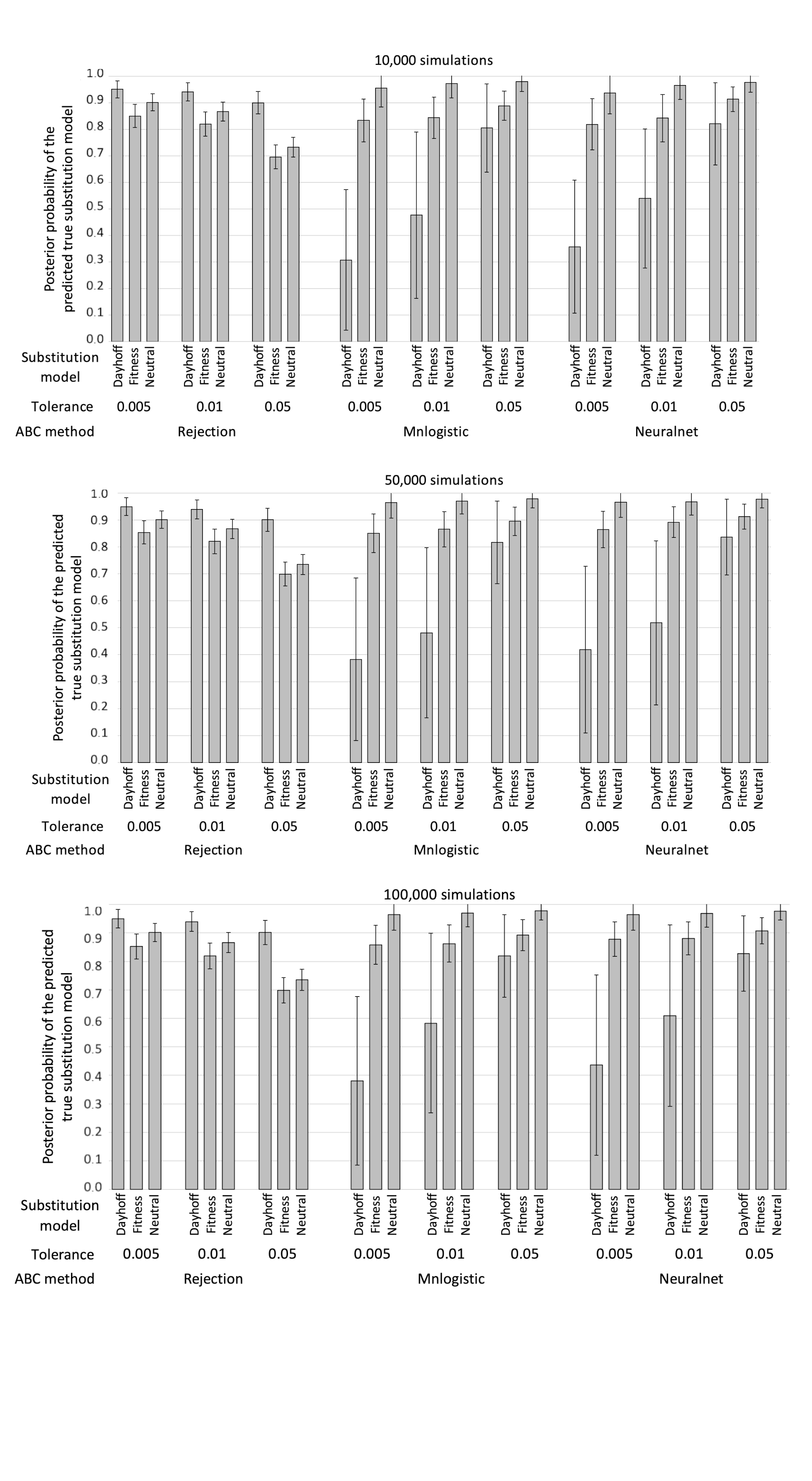
|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| 1 | *DGREM\_mean* | Mean of folding stability (free energy) of the proteins of the dataset. |
| 2 | *DGREM\_sd* | Standard deviation of folding stability (free energy) of the proteins of the dataset. |
| 3 | *SegSites* | Number of segregating sites. |
| 4 | *Grantham\_mean\_Position* | Mean of the Grantham distance between amino acid replacements for every protein position. |
| 5 | *Grantham\_sd\_Position* | Standard deviation of the Grantham distance between amino acid replacements for every protein position. |
| 6 | *Grantham\_sk\_Position* | Skewness of the Grantham distance between amino acid replacements for every protein position. |
| 7 | *Grantham\_ku\_Position* | Kurtosis of the Grantham distance between amino acid replacements for every protein position. |

# *ProteinModelerABC* Performance

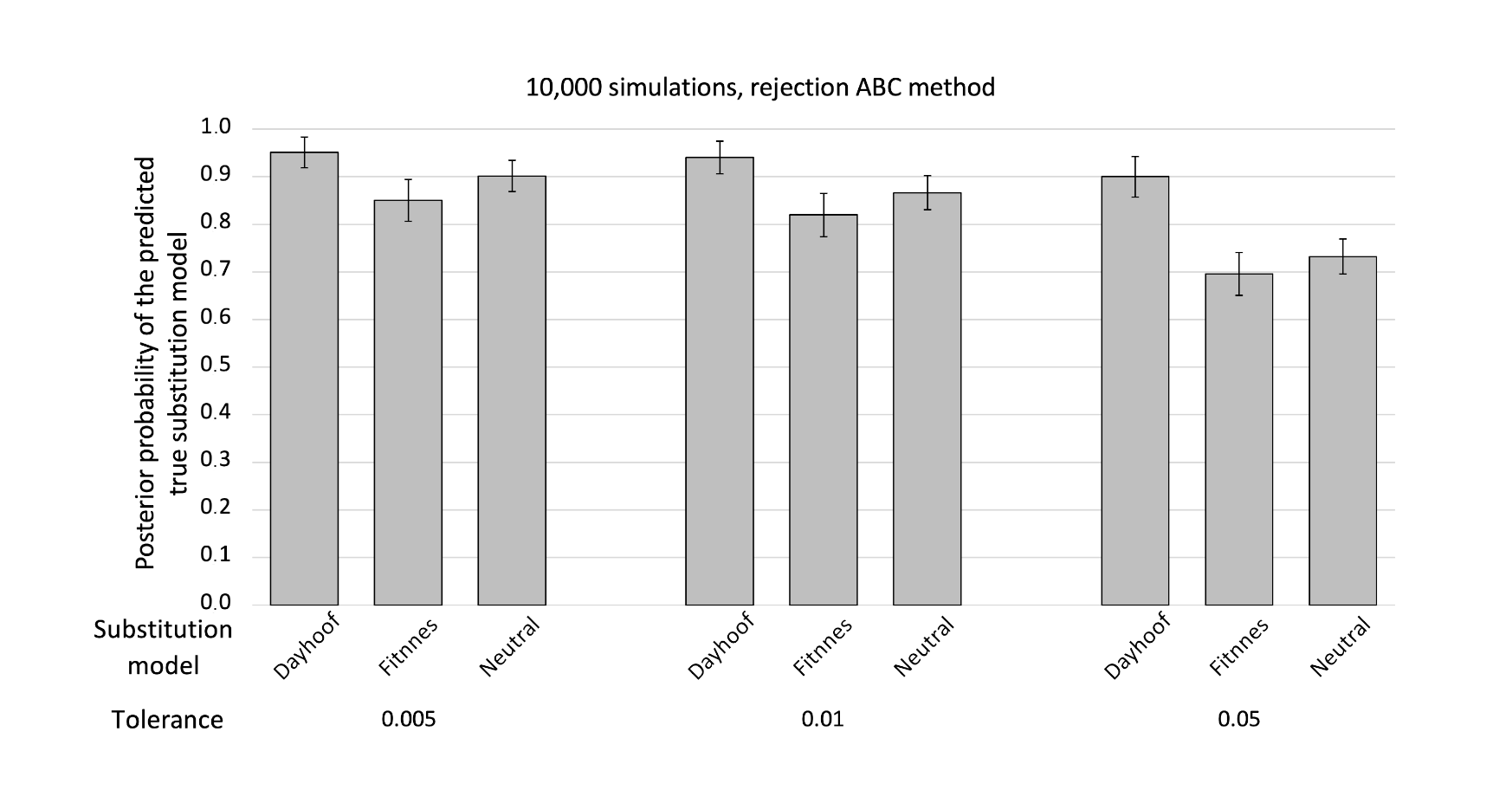
Substitution models of amino acids replacement are employed routinely to study evolutionary processes of protein evolution (Arenas 2015, Thorne 2000). However, there is a lack of evolutionary frameworks to select the best-fitting substitution model considering the SCS models, which successfully yielded accurate inferences of protein evolution, and sometimes even overcame empirical substitution models (e.g. Arenas et al., 2013, 2015, 2017; Arenas & Bastolla, 2019; Challis & Schmidler, 2012; García-Portugués et al., 2018; Golden et al., 2017; Herman et al., 2014; Norn et al., 2021; Perron et al., 2019).

The use of ABC for selecting among evolutionary scenarios in population genetics is well-established (e.g., Branco et al., 2022; Leuenberger & Wegmann, 2010; Sousa et al., 2012). However, ABC was not yet extended to substitution model selection even though it can be a suitable solution to evaluate complex substitution models that cannot be analyzed with a likelihood function. In this section we evaluated the application of ABC to estimate the best-fitting substitution model with *ProteinModelerABC* by testing its performance under different scenarios: (*i*) number of simulations (10,000, 50,000 and 100,000), (*ii*) tolerance (0.005, 0.01, and 0.05) and, (*iii*) three different ABC methods: rejection (*rejection*), weighted multiple linear regression (*mnlogistic*) and neural networks (*neuralnet*). We performed the evaluations using data simulated under the Dayhoff empirical substitution model (which is a widely used empirical substitution model), the Fitness site-dependent SCS model and the Neutral site-dependent SCS model (Arenas et al. 2013). The simulations were inspired in the thioredoxin protein family (27 sequences and 316 amino acids, showing a sequence identity of 0.44; PFAM code PF00070) and a representative protein structure (PDB code 1TDE) (Waksman et al. 1994) selected by SWISS-MODEL (Arnold et al. 2006) from the alignment consensus sequence. Next, we simulated the protein sequence alignments upon coalescent evolutionary histories (Kingman 1982), based on a population size of 1,000 individuals and a population substitution rate sampled from a uniform prior distribution between 0 and 500 that include values commonly observed nature (e.g. Arenas, 2022; Carvajal-Rodriguez, 2006; Lopes et al., 2014). For every scenario (3 substitution models × 3 different number of simulations per model × 3 ABC tolerance levels × 3 ABC estimation methods = 81 scenarios), we evaluated the power of ProteinModelerABC to distinguish between the three substitution models with a cross-validation based on 100 simulations (Csilléry et al. 2012). We found that the framework distinguishes between the studied substitution models with few error regardless of the applied number of simulations, the tolerance level and the ABC estimation method used.

Once we demonstrated that the method can distinguish between SCS and empirical models, we evaluated the accuracy of the framework in identifying the best-fitting substitution model. To do so, we used 100 multiple alignments of protein sequences simulated under each studied substitution model (Dayhoff, Fitness SCS and Neutral SCS models), hereafter named as pseudo-observed data. These analyses were also performed considering ABC based on 10,000, 50,000 and 100,000 simulations, tolerance levels of 0.005, 0.01, and 0.05 and the cited three ABC estimation methods. Again, we found that the accuracy of the substitution model selection is not affected by the number of simulations (thus 10,000 simulations are sufficient to distinguish between the studied models) while the optimal tolerance varies among the studied ABC methods (Figure 4). The rejection method showed a high robustness (all the pseudo-observed data produced acceptable results) and accuracy for predicting the true model, although the accuracy slightly decreased when increasing the tolerance (Figure 5), a pattern not observed for the substitution model selection with the weighted multiple linear regression and neural networks methods (Figure 4). Indeed, analyses based on the latter methods could not converge if the tolerance is small (not enough retained simulations to allow the prediction), thus those methods were more sensible to the tolerance (at least, a tolerance of 0.05 was required to obtain a sufficient number of retained simulations for the estimation) (Figure 4). Next, we explored the influence of every type of substitution model (i.e., empirical or site-dependent SCS model) as true model on the accuracy of the substitution model selection. In general, we found an acceptable distinction between the models using the rejection method (Figure 5). Concerning the weighted multiple linear regression and neural networks methods, again they required a high tolerance to obtain acceptable predictions especially when the true model was the empirical substitution model (Figure 4). Altogether, these findings recommend using the rejection method because it was less sensible to the tolerance level. We did not observe effects from the number of simulations on the accuracy of the predictions (Figure S1; compare the three plots).



**Figure 4. Evaluation of substitution model selection under different number of simulations and tolerance thresholds for different true substitution models.** The figure shows the probability for predicting every true substitution model (Dayhoff, Fitness site-dependent SCS and Neutral site-dependent SCS models) using 100 pseudo-observed datasets per model at different levels of tolerance (0.005, 0.01 and 0.05) with the different ABC estimation methods (rejection “rejection”, weighted multiple linear regression “mnlogistic” and neural networks “neuralnet”) and different number of simulations (10000, 50000 and 100000, which correspond with the upper, middle and lower plots, respectively). Error bars indicate 95% confidence intervals from the mean of the predictions from the pseudo-observed datasets.



**Figure 5. Evaluation of substitution model selection with *ProteinModelerABC* as a function of the number of simulations and tolerance**. The figure shows the probability for predicting a true substitution model (including the substitution models *Dayhoff*, *Fitness* site-dependent SCS and *Neutral* site-dependent SCS models as true models) with the different ABC estimation methods (rejection “*rejection*”, weighted multiple linear regression *“mnlogistic”* and neural networks *“neuralnet”*) and using 100 pseudo-observed datasets per model at different levels of tolerance (0.005, 0.01 and 0.05) and number of simulations (10000, 50000 and 100000). Error bars indicate 95% confidence intervals from the mean of the predictions from the pseudo-observed datasets.

We also evaluated the computer time with different number of processors. We used 1,000 simulations for terminal executions, and 10,000 for cluster executions. For both, we used the monkeypox TNF data (see section 5.4 example 1) to perform independent runs with 1, 2, 4, 8 and 12 processors in our computer (Figure 3). Note that, the number of processors and the computer time do not follow a linear function (Figure 3). This was expected because the processors do not work independently. Instead, they shared memory and even the computer time can be affected by the simultaneous computer use.



**Figure 6. Computer time of a ProteinModelerABC analysis using different numbers of local cores.** The analysis was carried out using a local machine (2.6 GHz Intel Core i7) simulating 1,000 simulations with 1, 2, 4, 8 and 12 cores. The protein sequences alignment used included 10 sequences of 160 amino acid each (see example 1 for more information). When running the simulations in parallel the RAM memory is shared among cores, and/or a core starts when another core finishes, thus reducing the required time without following a linear function.

## Assumptions and Limitations

ABC assumes that simulated data can mimic real data. Simulations are based on models and are never fully representative of the real process. This is actually a limitation present in any estimation approach based on a model of evolution. The idea is to perform simulations as realistic as possible. The data-generating process assumed by *ProteinEvolverABC* consists of drawing a sample genealogy from the standard coalescent (Kingman, 1982), and then, given the genealogy, generating the sequences evolved under a substitution model of protein evolution (Arenas et al., 2013). This is a well-established methodology to simulate protein sequences upon evolutionary histories (Yang, 2006).

# Acknowledgments

This work was supported by the Xunta de Galicia through the Grants [ED481A-2020/192].

# References

Abascal, F., Posada, D., & Zardoya, R. (2007). MtArt: A New Model of Amino Acid Replacement for Arthropoda. *Molecular Biology and Evolution*, *24*(1), 1-5. https://doi.org/10.1093/molbev/msl136

Adachi, J., & Hasegawa, M. (1996). *Programs for Molecular Phylogenetics Based on Maximum Likelihood*. *28*, 1-150.

Adachi, J., Waddell, P. J., Martin, W., & Hasegawa, M. (2000). Plastid Genome Phylogeny and a Model of Amino Acid Substitution for Proteins Encoded by Chloroplast DNA. *Journal of Molecular Evolution*, *50*(4), 348-358. https://doi.org/10.1007/s002399910038

Arenas, M. (2015). Trends in substitution models of molecular evolution. *Frontiers in Genetics*, *6*. https://doi.org/10.3389/fgene.2015.00319

Arenas, M. (2022). ProteinEvolverABC: Coestimation of recombination and substitution rates in protein sequences by approximate Bayesian computation. *Bioinformatics*, *38*(1), 58-64. https://doi.org/10.1093/bioinformatics/btab617

Arenas, M., & Bastolla, U. (2019). ProtASR2: Ancestral reconstruction of protein sequences accounting for folding stability. *Methods in Ecology and Evolution*, *11*, 248-257. https://doi.org/10.1111/2041-210X.13341

Arenas, M., Dos Santos, H. G., Posada, D., & Bastolla, U. (2013). Protein evolution along phylogenetic histories under structurally constrained substitution models. *Bioinformatics*, *29*(23), 3020-3028. https://doi.org/10.1093/bioinformatics/btt530

Arenas, M., Sánchez-Cobos, A., & Bastolla, U. (2015). Maximum-Likelihood Phylogenetic Inference with Selection on Protein Folding Stability. *Molecular Biology and Evolution*, *32*(8), 2195-2207. https://doi.org/10.1093/molbev/msv085

Arenas, M., Weber, C. C., Liberles, D. A., & Bastolla, U. (2017). ProtASR: An Evolutionary Framework for Ancestral Protein Reconstruction with Selection on Folding Stability. *Systematic Biology*, *66*(6), 1054-1064. https://doi.org/10.1093/sysbio/syw121

Arnold, K., Bordoli, L., Kopp, J., & Schwede, T. (2006). The SWISS-MODEL workspace: A web-based environment for protein structure homology modelling. *Bioinformatics*, *22*(2), 195-201. https://doi.org/10.1093/bioinformatics/bti770

Beaumont, M. A. (2010). Approximate Bayesian Computation in Evolution and Ecology. *Annual Review of Ecology, Evolution, and Systematics*, *41*(1), 379-406. https://doi.org/10.1146/annurev-ecolsys-102209-144621

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*(2), 384. https://doi.org/10.3390/genes13020384

Carvajal-Rodriguez, A. (2006). Recombination Estimation Under Complex Evolutionary Models with the Coalescent Composite-Likelihood Method. *Molecular Biology and Evolution*, *23*(4), 817-827. https://doi.org/10.1093/molbev/msj102

Challis, C. J., & Schmidler, S. C. (2012). A Stochastic Evolutionary Model for Protein Structure Alignment and Phylogeny. *Molecular Biology and Evolution*, *29*(11), 3575-3587. https://doi.org/10.1093/molbev/mss167

Csilléry, K., François, O., & Blum, M. G. B. (2012). abc: An R package for approximate Bayesian computation (ABC): *R package: abc*. *Methods in Ecology and Evolution*, *3*(3), 475-479. https://doi.org/10.1111/j.2041-210X.2011.00179.x

Darriba, D., Taboada, G. L., Doallo, R., & Posada, D. (2011). ProtTest 3: Fast selection of best-fit models of protein evolution. *Bioinformatics*, *27*(8), 1164-1165. https://doi.org/10.1093/bioinformatics/btr088

Dayhoff, M. O., Schwartz, R. M., & Orcutt, B. C. (1978). A model of evolutionary change in proteins. En *Atlas of Protein Sequence and Structure* (Dayhoff, M.O. Edition, Vol. 5, pp. 345-352). National Biomedical Research Foundation.

Dereeper, A., Guignon, V., Blanc, G., Audic, S., Buffet, S., Chevenet, F., Dufayard, J.-F., Guindon, S., Lefort, V., Lescot, M., Claverie, J.-M., & Gascuel, O. (2008). Phylogeny.fr: Robust phylogenetic analysis for the non-specialist. *Nucleic Acids Research*, *36*(Web Server), W465-W469. https://doi.org/10.1093/nar/gkn180

Dimmic, M. W., Rest, J. S., Mindell, D. P., & Goldstein, R. A. (2002). rtREV: An Amino Acid Substitution Matrix for Inference of Retrovirus and Reverse Transcriptase Phylogeny. *Journal of Molecular Evolution*, *55*(1), 65-73. https://doi.org/10.1007/s00239-001-2304-y

Edgar, R. C. (2004). MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, *32*(5), 1792-1797. https://doi.org/10.1093/nar/gkh340

García-Portugués, E., Golden, M., Sørensen, M., Mardia, K. V., Hamelryck, T., & Hein, J. (2018). *Toroidal diffusions and protein structure evolution*. https://doi.org/10.1201/9781315228570

Golden, M., García-Portugués, E., Sørensen, M., Mardia, K. V., Hamelryck, T., & Hein, J. (2017). A Generative Angular Model of Protein Structure Evolution. *Molecular Biology and Evolution*, *34*(8), 2085-2100. https://doi.org/10.1093/molbev/msx137

Grantham, R. (1974). Amino Acid Difference Formula to Help Explain Protein Evolution. *Science*, *185*(4154), 862-864. https://doi.org/10.1126/science.185.4154.862

Henikoff, S., & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, *89*(22), 10915-10919. https://doi.org/10.1073/pnas.89.22.10915

Herman, J. L., Challis, C. J., Novák, Á., Hein, J., & Schmidler, S. C. (2014). Simultaneous Bayesian Estimation of Alignment and Phylogeny under a Joint Model of Protein Sequence and Structure. *Molecular Biology and Evolution*, *31*(9), 2251-2266. https://doi.org/10.1093/molbev/msu184

Hudson, R. R. (2002). Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, *18*(2), 337-338. https://doi.org/10.1093/bioinformatics/18.2.337

Jones, D. T., Taylor, W. R., & Thornton, J. M. (1992). The rapid generation of mutation data matrices from protein sequences. *Bioinformatics*, *8*(3), 275-282. https://doi.org/10.1093/bioinformatics/8.3.275

Kingman, J. F. C. (1982). The coalescent. *Stochastic Processes and Their Applications*, *13*(3), 235-248. https://doi.org/10.1016/0304-4149(82)90011-4

Kosiol, C., & Goldman, N. (2005). Different Versions of the Dayhoff Rate Matrix. *Molecular Biology and Evolution*, *22*(2), 193-199. https://doi.org/10.1093/molbev/msi005

Le, S. Q., & Gascuel, O. (2008). An Improved General Amino Acid Replacement Matrix. *Molecular Biology and Evolution*, *25*(7), 1307-1320. https://doi.org/10.1093/molbev/msn067

Leuenberger, C., & Wegmann, D. (2010). Bayesian Computation and Model Selection Without Likelihoods. *Genetics*, *184*(1), 243-252. https://doi.org/10.1534/genetics.109.109058

Lopes, J. S., Arenas, M., Posada, D., & Beaumont, M. A. (2014). Coestimation of recombination, substitution and molecular adaptation rates by approximate Bayesian computation. *Heredity*, *112*(3), 255-264. https://doi.org/10.1038/hdy.2013.101

Müller, T., & Vingron, M. (2000). Modeling Amino Acid Replacement. *Journal of Computational Biology*, *7*(6), 761-776. https://doi.org/10.1089/10665270050514918

Navascués, M., Depaulis, F., & Emerson, B. C. (2010). Combining contemporary and ancient DNA in population genetic and phylogeographical studies: MOLECULAR POLYMORPHISM ANALYSIS OF ANCIENT DNA. *Molecular Ecology Resources*, *10*(5), 760-772. https://doi.org/10.1111/j.1755-0998.2010.02895.x

Nickle, D. C., Heath, L., Jensen, M. A., Gilbert, P. B., Mullins, J. I., & Kosakovsky Pond, S. L. (2007). HIV-Specific Probabilistic Models of Protein Evolution. *PLoS ONE*, *2*(6), e503. https://doi.org/10.1371/journal.pone.0000503

Norn, C., André, I., & Theobald, D. L. (2021). A thermodynamic model of protein structure evolution explains empirical amino acid substitution matrices. *Protein Science*, *30*(10), 2057-2068. https://doi.org/10.1002/pro.4155

Perron, U., Kozlov, A. M., Stamatakis, A., Goldman, N., & Moal, I. H. (2019). Modeling Structural Constraints on Protein Evolution via Side-Chain Conformational States. *Molecular Biology and Evolution*, *36*(9), 2086-2103. https://doi.org/10.1093/molbev/msz122

Sousa, V. C., Beaumont, M. A., Fernandes, P., Coelho, M. M., & Chikhi, L. (2012). Population divergence with or without admixture: Selecting models using an ABC approach. *Heredity*, *108*(5), 521-530. https://doi.org/10.1038/hdy.2011.116

Waksman, G., Krishna, T. S., Williams, C. H., & Kuriyan, J. (1994). Crystal structure of Escherichia coli thioredoxin reductase refined at 2 A resolution. Implications for a large conformational change during catalysis. *Journal of Molecular Biology*, *236*(3), 800-816.

Waterhouse, A., Bertoni, M., Bienert, S., Studer, G., Tauriello, G., Gumienny, R., Heer, F. T., de Beer, T. A. P., Rempfer, C., Bordoli, L., Lepore, R., & Schwede, T. (2018). SWISS-MODEL: Homology modelling of protein structures and complexes. *Nucleic Acids Research*, *46*(W1), W296-W303. https://doi.org/10.1093/nar/gky427

Whelan, S., & Goldman, N. (2001). A General Empirical Model of Protein Evolution Derived from Multiple Protein Families Using a Maximum-Likelihood Approach. *Molecular Biology and Evolution*, *18*(5), 691-699. https://doi.org/10.1093/oxfordjournals.molbev.a003851

Yang, Z. (1996). Among-site rate variation and its impact on phylogenetic analyses. *Trends in Ecology & Evolution*, *11*(9), 367-372. https://doi.org/10.1016/0169-5347(96)10041-0

Yang, Z. (2006). *Computational Molecular Evolution*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780198567028.001.0001

Yang, Z., Nielsen, R., & Hasegawa, M. (1998). Models of amino acid substitution and applications to mitochondrial protein evolution. *Molecular Biology and Evolution*, *15*(12), 1600-1611. https://doi.org/10.1093/oxfordjournals.molbev.a025888