Documentation for ***ProtModel***

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

Current version is 1.0

© 2022. David Ferreiro

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

08/06/2022

**

**Contents**

[Disclaimer 3](#_Toc105763142)

[Credits 3](#_Toc105763143)

[1. Purpose 3](#_Toc105763144)

[1.1. Models 4](#_Toc105763145)

[2. Versions and Graphical User Interface 4](#_Toc105763146)

[3. Executables and compilation 5](#_Toc105763147)

[4. ProtModel execution 6](#_Toc105763148)

[5. ProtModel usage 6](#_Toc105763149)

[5.1. The Settings input file 7](#_Toc105763150)

[5.1.1. Available prior distributions 7](#_Toc105763151)

[5.1.2. Simulation phase 8](#_Toc105763152)

[5.1.3. Estimation phase 12](#_Toc105763153)

[5.2. Screen information with an example 12](#_Toc105763154)

[5.2.1. ProtModel compilation: 12](#_Toc105763155)

[5.2.2. ProtModel execution: 16](#_Toc105763156)

[5.3. Output files 20](#_Toc105763157)

[5.3.1. ABCOutputs 20](#_Toc105763158)

[5.3.2. SimulationsOutputs 21](#_Toc105763159)

[5.4. Re-analysing data 21](#_Toc105763160)

[5.5. Examples 22](#_Toc105763161)

[5.6. Message errors and recommendations 22](#_Toc105763162)

[6. Models and Methods 23](#_Toc105763163)

[6.1. Evolutionary Models and Computer Simulations 23](#_Toc105763164)

[6.2. Summary Statistics 23](#_Toc105763165)

[6.3. ABC Methods 24](#_Toc105763166)

[6.4. *ProtModel* Performance 24](#_Toc105763167)

[6.5. Assumptions and Limitations 24](#_Toc105763168)

[7. Acknowledgments 24](#_Toc105763169)

[8. References 24](#_Toc105763170)

# 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

*ProtModel* is an evolutionary framework to estimate the best-fitting substitution model with approximate Bayesian computation (ABC) from protein sequence alignments. *ProtModel* is designed to be run either on Linux OS or Mac OSX and it is freely available from <https://github.com/DavidFerreiro/ProtModel>.

The user can specify a prior distribution of substitution rate as well as other evolutionary parameters that will be treated as nuisance parameters, like amino acid frequencies, proportion of invariable sites (+I) or heterogeneity change across sites according to a gamma distribution (+G). Also, there is a graphical user interface (GUI) *ProtModel\_GUI* available, which allows for a user-friendly procedure to run *ProtModel* and a version to run it on clusters (*ProtModel\_Cluster*) to save computer time and resources.

The computer simulations are performed via the coalescent program *ProteinEvolverProtABC*, a version of the simulator *ProteinEvolver* (<https://github.com/MiguelArenas/proteinevolver>) (Arenas et al., 2013) adapted to ABC. *ProtModel* implements a total of 7 summary statistics (SS). In general, they either explore the stability of the individual sequences of the alignment or the physicochemical properties of the amino acids’ replacements. Conveniently, *ProtModel* can run the simulations on parallel, according to user specifications, to save computer time. This is highly recommended because the simulations under substitution models that consider the stability of the protein and its calculation take a long time.

Three ABC methods are implemented in *ProtModel*, (*i*) the “*rejection*” algorithm which use 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 probabilities of models based on the observed statistics (Beaumont, 2010; Csilléry et al., 2012). For all algorithms, the user must specify the number of simulations to use, the number of simulations to retain (tolerance of the rejection algorithm), the number of iterations and which SS to use. *ProtModel* provides also several diagnostic plots to assess qualitatively how well the model fitted the data, namely: boxplots of the SS calculated for each investigated model; bar plot of the median distance between the simulations retained SS values and the observed for each model; histograms of the SS of the retained simulations for each model; scatterplots of the SS and the prior distribution for each model; and a plot of the two principal components 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 with a black cross.

## Models

*ProtModel* is able to work with both empirical or site independent and with site-dependent substitution models that consider the stability of the protein, called structural constrain substitution (SCS) models (Arenas et al., 2015). In case of empirical substitution models, the user can select among 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., 2006), Mtmam (Yang et al., 1998), Mtrev24 (Adachi & Hasegawa, s. f.), RtRev (Dimmic et al., 2002), VT (Müller & Vingron, 2000), WAG (Whelan & Goldman, 2001) and UserEAAM models (Arenas, 2015) (the latter is a user-defined model that must be provided as an additional input file with name UserEAAM, the format of this file is explained with details in the documentation of the simulator *ProteinEvolver* (Arenas et al., 2013)). Within SCS models, there are two available, mean-field Neutral (which does not consider population size) and mean-field Fitness (which needs user specified population size) (Arenas et al., 2015).

# Different versions and Graphical User Interface

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

## *ProtModel*

***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 choose between the mandatory parameters and to fill the optional if intended (see section 4.1). Settings must be specified carefully, incorrect specification may be detected by the framework automatically (e.g., it does not allow negative substitution rates), yet user-specified settings without biological meaning may produce errors in *ProtModel* analysis, even though when the analysis runs completely (see section 4.2). The second windows will launch the simulations, SS calculation and the consecutive ABC analyses. While the program is running, it will inform about the progress and it will be show up 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 Settings input file, considering not only about the parameters value specification but also about the format (see section 4.2). Removing or to adding some characters may yield into a *ProtModel* fail and such changes may be difficult to detect. We recommend to use one of the Settings.txt files of the “*Examples*” folder and change the values carefully. Once the framework was executed it will check Settings inputs 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).

## *ProtModel\_Cluster*

This version of *ProtModel* can be run on clusters if they are based on Linux OS environments. It enables the user to work with many more processors and in parallel without sharing memory taking advance of MPI (with many more processors) interface. Like command line version, the user must fill manually the Settings input file considering the biological meaning of the input values and the format of the file too (see section 4.2). *ProtModel\_Cluster* execution will check the Settings inputs and compute the SS of the MSA, but won’t launch the simulations and their corresponding SS calculation. While the execution, some batch script will be created and to run simulations and their corresponding SS calculation we have to submit a master batch script called “*launch\_Simu.sh*” to Slurm (a task and cluster management system). Once the analysis finished, we will find the same output files and folders as in a local computer execution (see section 4.4). However, the user has to perform the ABC estimation in their computer because the cluster don’t have an updated R version whit the ABC library (see section 4.3.2). Everything needed to carry out the ABC estimation will be placed in the “*ABCOutputs*” folder, they user only must change the directory rout of the “*ABCAnalysis.r*” file.

# Executables and compilation

*ProtModel* 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) Note that some python libraries may be already installed by default. 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 numpy*

*install.packages("abc")*

Our program works using the *ProteinEvolverProtABC* simulator (Arenas et al., 2013) and the *DeltaGREM* software (Arenas et al., 2017), which are used to perform the protein sequences simulations and to calculate the protein free energy, respectively. They are placed in the “*source*” folder with a csv file called “*Grantham*”, and all of them should not be modified by the user. The folder “*GUI*” contains the GUI executable and two images needed to run it. *ProtModel* pipeline consists on 6 executable files needed to execute the framework but the user should only execute the main file called “*ProtModelGeneral.py*”. All these folders and files must not be modified to a properly execution of *ProtModel*.

*ProtModel* must be compiled for the OS through a Makefile file which is placed on the main directory. It will compile *ProteinEvolverProtABC* and *DeltaGREM* programs and will create the executables and place them in the folder “*bin*”.

Table 1. Libraries needed to run *ProtModel*

|  |  |  |
| --- | --- | --- |
| *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 |

# *ProtModel* usage

*ProtModel* have four mandatory input files:

* **Settings file**. This input file, called 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 produce errors in *ProtModel* analysis.
* **Multiple alignment of protein sequences**. The protein sequence alignment that will be analyzed must be provided by the user in sequential phylip format (.phy). Phylip requires that each sequence identifier is exactly 10 characters long, padded with spaces as necessary. In addition, work with template structures demands that the MSA has to have the same length as template chain. So, sequences have to be aligned with the one template chain sequence and remove the positions that in the template sequence has a gap in every sequence. For all this process, we strongly recommend to use “*Align.py*” script placed in the folder “*Scripts*” (see section 4.1.). Another option could be use other programs as “*MUSCLE*” (Edgar, 2004) for sequences alignment or “*Phylogeny.fr*” (Dereeper et al., 2008) to convert into phylip format (<http://phylogeny.lirmm.fr/phylo_cgi/data_converter.cgi>), but the user must check all the process and the number of characters of the sequences identifier carefully to avoid errors.
* **Template structure**. This is a protein structure (.pdb file) which has the highest homology with the multiple alignment of protein sequences. We advised 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 wild-type (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.in.** This file contains structural information needed for stability calculation (needed for two of the SS) and for MSA simulations under SCS models. It cannot be modified.

## Before starting

Before starting a *ProtModel* 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 to use the “*Find-WT.py*” python script in the “*Scripts*” folder, which return the wild-type (WT; a sequence composed by the most common amino acid per site) sequence.

*python FindWT.py --input MSA.fasta*

Next, the WT sequences 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 to avoid high values when it’s possible. Work with template structures requires the MSA to have the same length as the template chain. So, sequences have to be aligned with the template chain sequence and positions that in the template sequence have a gap have to be remove in every sequence. The “*Align.py*” script placed in the folder “*Scripts*” will make the alignment, remove the gaps positions and change the file into *phylip* format. Another option could be using other 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 output file name, which must have the .phy extension (--output).

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

*ProtModel* 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 *ProtModel* is to **compare the two SCS models with an empirical substitution model previously selected for the alignment** using some programs as *ProtTest* (Darriba et al., 2011). To perform the simulations the user must design a prior distribution for the substitution rate per site. This could be an unfamiliar measure, so we also include a script which for a desired theta (*θ*) value will return the sequences identity of the MSA and substitution rate per site, asking if the organism is haploid or diploid and the population size. Sequence identity will give us an idea about *θ* value, if it is very high, we must not use a high *θ* (>500).

*Theta.py --input NS1.phy*

The recommend structure of a *ProtModel* analysis consists on a folder located at a user-defined path, including a Settings.txt file, the MSA, the template .*pdb*, the structures.in file and all *ProtModel* material (including executable files and folders). Examples well prepared for a standard *ProtModel* run are provided in the folder “*Examples*”. After completing a run of *ProtModel,* two folders are created in the working directory, the “*ABCOutputs*” and the “*SimulationsOuputs*” folders (see details in section 4.4). Since the estimation phase is quite fast in comparison with the simulation phase, the user is advised to explore different settings for the ABC method (see section 4.5 in “*Re-analyzing data*”) if desired, without having to run again the simulation phases.

## The Settings input file

The Settings input file must contain all the information required to perform the analysis. It is highly recommended to be carefully handled by the user since mistakes may alter results (error messages will be displayed on the screen and may suggest stop the execution by typing CTRL+C) or use the GUI.

The file consists of two main blocks, the simulation phase and 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 (it is used by the program to identify the parameter).

- Some parameters are mandatory and must be specified, these parameters contain an “*\**” (see below).

- Some parameters require a prior distribution (see section 4.1.1), this is a distribution of the parameters values used to perform the simulations which should include the presumed value (for example, if the user believes that the substitution rate per site can be 0.1 then the prior distribution of the substitution rate per site should include 0.1).

### Available prior distributions

Several distributions are included in *ProtModel* to simulate protein data under different evolutionary scenarios (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 any parameter. Details for each particular parameter are described in the following subsections.

Table 2. Available distributions in ProtModel.

|  |  |  |  |
| --- | --- | --- | --- |
| *Distribution* | *Description* | *Truncated* | *Examples* |
| fix | Fixed value (integer or non-integer) | 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 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 distribution (shape1, shape 2) | t # # | beta 1.0e-7 5.0e-7; beta 1.0e-7 5.0e-7 t 2.5e-7 1.0e-6 |
| dirichlet | Dirichlet distribution (alpha “vector”) | n.a. | dirichlet 1 1 1 1; dirichlet 1 1 1 1 1 1 |

### Simulation phase

***General settings for the simulation***

* **Name of the file with the target protein sequences alignment.** This specification is mandatory. Only specify the filename (i.e. no pathway) since the target alignment file needs to be placed in the same pathway as the Settings.txt file. For example,

*### Target alignment file ### # phylip format*

*\*NameOfPhylipFile=* *ProtSeq1.phy*

* **Number of simulations.** This parameter is mandatory. We recommend at least 1 000 computer simulations but with 10 000 simulations results should be accurate. However, the number of simulations required to obtain accurate estimates depends on many factors, especially the target multiple sequence alignment. More complex sequences (large molecular diversity) could require more simulations because the parametric landscape could be more irregular and should be more sampled. For example,

*### Total number of simulations ###*

*\*NumberOfSimulations=10000*

* **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 summary statistics if there are indels in the multiple sequence alignment. For example,

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

*\*Indels=Ignored*

* **Number of processors to run the simulations in parallel on a machine.** This parameter is mandatory (by default simulations run on all user computer processors). This parallelization works on machines using Linux OS with shared memory. Ideally, one should specify at the most the number of available processors of the machine. The number of processors must be specified in the settings file. For example,

*# Number of available processors to run the simulations in parallel. All by default*

*\*NumberOfProcessors=12*

* **Save simulated data.** This parameter is mandatory. It is recommended to not save the simulated because it requires a lot of space in the hard disk of user’s computer. 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 this data is required for calculating the Summary Statistics in the following step. For example,

*#* *Save simulated data. “No”, “Yes” (but it requires space in the disk)*

*\*SaveSimulations=No*

* **Show running information.** This parameter is mandatory. If the user chooses “*No*” the amount of information printed on the screen during *ProtModel* execution will be reduced. This option is recommended to save computing time (printing information on the screen requires more computer time). Regardless the option selected, the final results will not change, it only affects to the amount of information shown on the screen during the execution. For example,

*# Show running information (simulations and summary statistics) on the screen. “No”, “Yes” (but it slows down the running time)*

*\*ShowInformationScreen=No*

***Demographic settings***

* **Haploid/diploid simulated data.** This parameter is mandatory. Haploid is defined with a value of 1 and diploid with a value of 2. For example,

*# Haploid or Diploid data (haploid=1, diploid=2)*

*\*Haploid/Diploid=2*

* **Effective population size (*N*).** This parameter is mandatory. The parameter value must be an integer. For example,

*# Population size (i.e., 1000).*

*\*PopulationSize=1000*

***Longitudinal sampling***

* **Sampling at different times.** This parameter is optional. The user can specify the time at which the tip nodes of the tree (i.e. samples) were 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 7 and 8. Note that this option does not work if a deme converges, backwards in time, before the last sampling time.For example,

*# Logitudinal 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. For example,

*# Generation time. fix, uniform; i.e., uniform 500 1000*

*GenerationTime=fix 1200*

* **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 a good idea 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. Example: 0 1e-5 (= 0 0.00001). For demographic periods, after “*1*” the user has to specify the number of periods (from the present to the past) and N during those periods. The first number here specifies the number of periods. For each period should be three consecutive numbers indicating the size N at the beginning and at the end of the period, and the duration of the period in generations. Example: 1 3 1000 1250 1000 1300 1550 2000 1560 1000 3000.

*# Exponential growth rate or Demographic periods*

*Growthrate=0 1e-5*

* **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 n 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. Example: 2 2 3 3 (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 variable with time according to temporal periods. The first number specifies the number of temporal periods, then: For only 1 period, the second number is the migration rate (constant). Example: 1 0.001 (only 1 period with migration rate = 0.001).

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

*# Migration rate (constant or variable with time according to temporal periods). 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 consecutive numbers. The first number and the second number are the numbers of the demes to converge. The third number is the time to that convergence. With this option the user can build the demes evolutionary tree but it is only available when the migration model is activated (despite the migration rate could be zero). Examples: 1 1 2 2000 (for 2 initial demes (Migration Model = 2), convergence of deme 1with deme 2 at time 2000 to create a new deme 3). 3 1 2 400 3 4 1900 5 6 2000 (for 4 initial demes (Migration Model = 4) convergence of deme 1 with deme 2 at time 400 to create a new deme 5, convergence of deme 3 with deme 4 to create a new deme 6 at time 1900, convergence of deme 5 with deme 6 at time 2000 to create a final new deme 7).

*# Events of convergence of demes. See documentation for details*

*ConvergenceDemes= 1 1 2 2000*

***Protein evolution and substitution model***

* **Amino acid substitution rate per site.** This parameter is mandatory. Distributions allowed: fix, uniform, norm(t), exp(t), gamma(t), beta(t). For example,

*# Amino acid substitution rate. i.e., fix 7.0e-6.*

*\*SubstitutionRate=uniform 0 1.67e-4*

* **Empirical substitution model of amino acid evolution.** This parameter is mandatory. The user has to specify all the desire empirical substitution model of protein evolution separated by a space among the following (Arenas, 2015): Blosum62, CpRev, Dayhoff, DayhoffDCMUT, HIVb, HIVw, JTT, JonesDCMUT, LG, Mtart, Mtmam, Mtrev24, RtRev, VT, WAG, UserEAAM (the latter is a user-defined model that must be provided as an additional input file with name UserEAAM, the format of this file is explained with details in the documentation of the simulator *ProteinEvolver* (Arenas et al., 2013)). We **highly recommend to use only the best-fitting empirical substitution model for the alignment** which can be obtained using some programs as *ProtTest* (Darriba et al., 2011).For example,

*# Model of amino acid substitution (i.e., Blosum62, CpRev, Dayhoff, DayhoffDCMUT, HIVb, HIVw, JTT, JonesDCMUT, LG, Mtart, Mtmam, Mtrev24, RtRev, VT, WAG, UserEAAM)*

*\*SubstitutionModel=JTT*

* **SCS model of amino acid evolution.** This parameter is mandatory. The user has to specify all the desire SCS models of protein evolution separated by a space among the following: Fitness, Neutral. For example,

*# Model of structural amino acid substitution (Fitness, Neutral)*

*\*StructuralSubstitutionModel=Fitness Neutral*

* **Amino acid frequencies.** This parameter is mandatory. Frequencies for each amino acid site along sequences. Distributions allowed: fix or dirichlet. If it is not specified the program will assume equally distributed frequencies (all fix with value 0.05). For example,

*# Amino acid frequencies. fix or dirichlet. By default, equally distributed frequencies. i.e., dirichlet 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1*

*\*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*

* **Rate of heterogeneity across sites (+G).** This parameter is optional. Distributions allowed: fix, uniform, norm(t), exp(t), gamma(t), beta(t). For example,

*# Rate of heteregeneity across sites, +G. fix, uniform, gamma, beta, normal, exponential; i.e., fix 0.6*

*RateHetSites=uniform 0.76 0.90*

* **Proportion of invariable sites (+I).** This parameter is optional. Distributions allowed: fix, uniform, norm(t), exp(t), gamma(t), beta(t). For example,

*# Proportion of invariable sites, +I. fix, uniform, gamma, beta, normal, exponential; i.e., exponential 0.002 t 0 1.0*

*PropInvSites=uniform 0.3 0.5*

* **Template.** This parameter is mandatory. PDB protein structure used to structural substitution models and to calculate proteins free energy. For example,

*# PDB protein structure used to structural substitution models. See documentation for details*

*\*Template=3IXO.pdb*

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

*# Chain of the PDB protein structure used to structural substitution models. See documentation for details*

*\*Chain=A*

* **GMRCA.** This parameter is optional. By default, the GMRCA/MRCA sequence is simulated according to the amino acid frequencies. However, the user can optionally specify its own root sequence by a text file which must be located in the main directory. The file just contains in a single sequence. In case the user does not select any GMRCA/MRCA template sequence will be used, which is highly recommended.

*# GMRCA input file. See documentation for details*

*GMRCA=GMRCA.txt*

***Graphical settings***

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

*#* *Multiple pages. PDF documents with multiple plost per page (No, Yes)*

*\*MultiPage=Yes*

### Estimation phase

***ABC estimation***

* **ABC iterations.** This parameter is mandatory. Number of simulations to consider in the ABC analysis. Unless testing the methodology, the user is advised to use all the simulations performed. Note that the number of simulations to use has to be obviously equal or less to the number of simulations performed (specified in *NumberOfSimulations*).For example,

*#ABC iterations. Number of simulations to consider (Iterations <= NumberOfSimulations)*

*\*ABCIterations=100*

* **ABC tolerance.** This parameter is mandatory. Proportion of simulations closest to real data to retain in the ABC procedure. A tolerance of 0.01 can be enough but the best tolerance varies among multiple sequence alignments. It is recommended to explore different values of this parameter. For example,

*#ABC tolerance. % of simulations closest to real data to retain in the ABC procedure*

*\*ABCTolerance=0.01*

* **ABC method.** This parameter is mandatory. The ABC algorithm has to be specified. The user can choose between rejection, mnlogistic or neuralent.For example,

*#ABC method (rejection, mnlogistic, neuralnet).*

*\*ABCMethod=rejection*

* **Summary statistics.** This parameter is mandatory. The user needs to choose which summary statistics to use for the ABC estimation by specifying their numeric identifiers (see section 5 “*Models and Methods*” for further details).It is recommended to specify the 7 summary statistics implemented in the program but in some cases where the folding stability standard deviation is far from the real data value we recommend to not use it (summary statistic 2). For example,

*#Summary statistics to use. See documentation for details*

*\*SummaryStatistics= 1 2 3 4 5 6 7*

## Screen information with an example

The data shown below corresponds with the normal output information of a ProtModel run. To the first phase of the program, the user has to use computer terminal. The example corresponds with a rapid project, in which we tried to distinguish between the empirical HIVw and the SCS Neutral models. Before try this example, you have to have installed Python 3 and R.

### ProtModel compilation:

* First, we need to be in the main directory of ProtModel. The bold words bellow correspond with the mandatory input files, while the rest are the normal ProtModel directory available to download.

*Test\_ProtModel %* ***ls***

**3on9.pdb** LeerSettings.py Variables.py

ChangeVariablesPE.py Makefile source

Errores.py ProtModelGeneral.py **structures.in**

Functions.py **Settings.txt**

GUI **TNF.phy**

* Then, we only have to do a “*make all*” command to compile all the programs (*ProteinEvolutionProtABC* and *DeltaGREM*).

*Test\_ProtModel %* ***make all***

*Creating bin folder ...*

*Done!*

*Copying Grantham.csv file ...*

*Done!*

*Compiling DeltaGREM ...*

*/Library/Developer/CommandLineTools/usr/bin/make -C source/DeltaGREM\_src clean*

*rm -fr main\_DeltaGREM.o REM.o random3.o Get\_pars\_DeltaGREM.o alignments.o mutations.o Codes.o Input.o gen\_code.o allocate.o output.o read\_pdb.o read.o Sec\_str\_all.o NeedlemanWunsch.o Profit\_aux.o DeltaGREM*

*/Library/Developer/CommandLineTools/usr/bin/make -C source/DeltaGREM\_src all*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o main\_DeltaGREM.o main\_DeltaGREM.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o REM.o REM.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o random3.o random3.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o Get\_pars\_DeltaGREM.o Get\_pars\_DeltaGREM.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o alignments.o alignments.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o mutations.o mutations.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o Codes.o Codes.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o Input.o Input.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o gen\_code.o gen\_code.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o allocate.o allocate.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o output.o output.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o read\_pdb.o read\_pdb.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o read.o read.c*

*read.c:251:1: warning: control may reach end of non-void function*

*[-Wreturn-type]*

*}*

*^*

*1 warning generated.*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o Sec\_str\_all.o Sec\_str\_all.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o NeedlemanWunsch.o NeedlemanWunsch.c*

*gcc -O2 -Wall -std=c99 -pedantic -g -pg -D\_FILE\_OFFSET\_BITS=64 -D\_LARGE\_FILE\_SOURCE -c -o Profit\_aux.o Profit\_aux.c*

*gcc main\_DeltaGREM.o REM.o random3.o Get\_pars\_DeltaGREM.o alignments.o mutations.o Codes.o Input.o gen\_code.o allocate.o output.o read\_pdb.o read.o Sec\_str\_all.o NeedlemanWunsch.o Profit\_aux.o -o DeltaGREM -lm -L/usr/lib64/libg2c.so.0.0.0*

*ld: warning: directory not found for option '-L/usr/lib64/libg2c.so.0.0.0'*

*Done!*

*Compiling ProteinEvolverProtABC ...*

*/Library/Developer/CommandLineTools/usr/bin/make -C source/ProteinEvolverProtABC clean*

*Removing object and executable files to save space*

*Finished cleanup.*

*/Library/Developer/CommandLineTools/usr/bin/make -C source/ProteinEvolverProtABC all*

*Building ProteinEvolverProtABC version 1.2.0*

*gcc -c -O3 -Wall ProteinEvolverProtABC1.2.0.c*

*ProteinEvolverProtABC1.2.0.c:2035:16: warning: taking the absolute value of*

*unsigned type 'unsigned short' has no effect [-Wabsolute-value]*

*seed = seed + fabs(tmb.millitm);*

*^*

*ProteinEvolverProtABC1.2.0.c:2035:16: note: remove the call to 'fabs' since*

*unsigned values cannot be negative*

*seed = seed + fabs(tmb.millitm);*

*^~~~*

*ProteinEvolverProtABC1.2.0.c:16910:33: warning: using floating point absolute*

*value function 'fabs' when argument is of integer type [-Wabsolute-value]*

*arrayIndBreakpointsOrd\_C[i] = fabs(arrayIndBreakpointsOrd[i]/3);*

*^*

*ProteinEvolverProtABC1.2.0.c:16910:33: note: use function 'abs' instead*

*arrayIndBreakpointsOrd\_C[i] = fabs(arrayIndBreakpointsOrd[i]/3);*

*^~~~*

*abs*

*ProteinEvolverProtABC1.2.0.c:25094:33: warning: using floating point absolute*

*value function 'fabs' when argument is of integer type [-Wabsolute-value]*

*arrayIndBreakpointsOrd\_C[i] = fabs(arrayIndBreakpointsOrd[i]/3);*

*^*

*ProteinEvolverProtABC1.2.0.c:25094:33: note: use function 'abs' instead*

*arrayIndBreakpointsOrd\_C[i] = fabs(arrayIndBreakpointsOrd[i]/3);*

*^~~~*

*abs*

*ProteinEvolverProtABC1.2.0.c:25285:33: warning: using floating point absolute*

*value function 'fabs' when argument is of integer type [-Wabsolute-value]*

*arrayIndBreakpointsOrd\_C[i] = fabs(arrayIndBreakpointsOrd[i]/3);*

*^*

*ProteinEvolverProtABC1.2.0.c:25285:33: note: use function 'abs' instead*

*arrayIndBreakpointsOrd\_C[i] = fabs(arrayIndBreakpointsOrd[i]/3);*

*^~~~*

*abs*

*ProteinEvolverProtABC1.2.0.c:27134:22: warning: format string is not a string*

*literal (potentially insecure) [-Wformat-security]*

*sprintf(FILE\_CODE, FILE\_CODE\_DEF);*

*^~~~~~~~~~~~~*

*/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/secure/\_stdio.h:47:56: note:*

*expanded from macro 'sprintf'*

*\_\_builtin\_\_\_sprintf\_chk (str, 0, \_\_darwin\_obsz(str), \_\_VA\_ARGS\_\_)*

*^~~~~~~~~~~*

*ProteinEvolverProtABC1.2.0.c:27134:22: note: treat the string as an argument to*

*avoid this*

*sprintf(FILE\_CODE, FILE\_CODE\_DEF);*

*^*

*"%s",*

*/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/secure/\_stdio.h:47:56: note:*

*expanded from macro 'sprintf'*

*\_\_builtin\_\_\_sprintf\_chk (str, 0, \_\_darwin\_obsz(str), \_\_VA\_ARGS\_\_)*

*^*

*ProteinEvolverProtABC1.2.0.c:27709:22: warning: format string is not a string*

*literal (potentially insecure) [-Wformat-security]*

*sprintf(FILE\_CODE, FILE\_CODE\_DEF);*

*^~~~~~~~~~~~~*

*/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/secure/\_stdio.h:47:56: note:*

*expanded from macro 'sprintf'*

*\_\_builtin\_\_\_sprintf\_chk (str, 0, \_\_darwin\_obsz(str), \_\_VA\_ARGS\_\_)*

*^~~~~~~~~~~*

*ProteinEvolverProtABC1.2.0.c:27709:22: note: treat the string as an argument to*

*avoid this*

*sprintf(FILE\_CODE, FILE\_CODE\_DEF);*

*^*

*"%s",*

*/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/usr/include/secure/\_stdio.h:47:56: note:*

*expanded from macro 'sprintf'*

*\_\_builtin\_\_\_sprintf\_chk (str, 0, \_\_darwin\_obsz(str), \_\_VA\_ARGS\_\_)*

*^*

*ProteinEvolverProtABC1.2.0.c:33636:38: warning: for loop has empty body*

*[-Wempty-body]*

*{ int i; for (i=0; i<n; x[i]\*=a,i++) ; return(0); }*

*^*

*ProteinEvolverProtABC1.2.0.c:33636:38: note: put the semicolon on a separate*

*line to silence this warning*

*ProteinEvolverProtABC1.2.0.c:33638:40: warning: for loop has empty body*

*[-Wempty-body]*

*{ int i; for (i=0; i<n; y[i]=x[i],i++) ; return(0); }*

*^*

*ProteinEvolverProtABC1.2.0.c:33638:40: note: put the semicolon on a separate*

*line to silence this warning*

*ProteinEvolverProtABC1.2.0.c:52540:13: warning: unused function 'PrintTitle'*

*[-Wunused-function]*

*static void PrintTitle(FILE \*filep)*

*^*

*ProteinEvolverProtABC1.2.0.c:52709:13: warning: unused function 'PrintDate'*

*[-Wunused-function]*

*static void PrintDate (FILE \*filep)*

*^*

*10 warnings generated.*

*gcc -lm -O3 -Wall -o ProteinEvolverProtABC1.2.0 ProteinEvolverProtABC1.2.0.o*

*Finished compiling.*

*Done!*

*Note that R will require the following libraries: abc*

*These libraries can be installed from R by typing:*

*install.packages(abc)*

*And python will require the following libraries: os, sys, Biopython, rnamdom, numpy, warnings, pandas, csv, re, plataform and multiprocessing*

*These libraries can be installed from command line by typing:*

*pip install library\_name*

*See the documentation for additional details about ProtModel and for cluster version information*

*Compilation completed!*

*Test\_ProtModel %* ***make clean***

*Removing executables ...*

*/Library/Developer/CommandLineTools/usr/bin/make -C source/DeltaGREM\_src clean*

*rm -fr main\_DeltaGREM.o REM.o random3.o Get\_pars\_DeltaGREM.o alignments.o mutations.o Codes.o Input.o gen\_code.o allocate.o output.o read\_pdb.o read.o Sec\_str\_all.o NeedlemanWunsch.o Profit\_aux.o DeltaGREM*

*/Library/Developer/CommandLineTools/usr/bin/make -C source/ProteinEvolverProtABC clean*

*Removing object and executable files to save space*

*Finished cleanup.*

*Done!*

* *DeltaGREM* and *ProteinEvolverProtABC* executables and the “*Grantham.csv*” file will be placed in a new folder called “*bin*” and it cannot be removed or edited. On the other hand, the ProtModel\_GUI should stay in its folder “*GUI*” but it can be moved as long as the images are kept in the same location.

*Test\_ProtModel %* ***ls***

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

***bin***  *LeerSettings.py structures.in*

*ChangeVariablesPE.py Makefile TNF.phy*

*Errores.py ProtModelGeneral.py Variables.py*

*Functions.py Settings.txt*

### *ProtModel* execution:

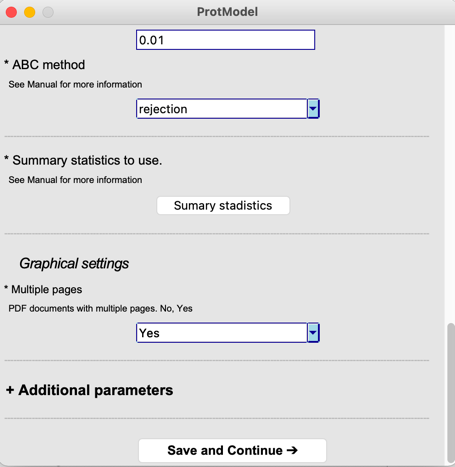
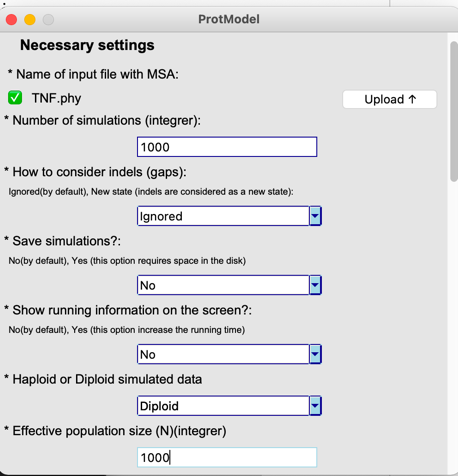
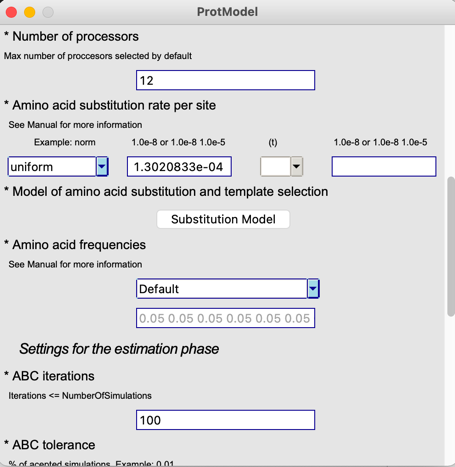
A full execution may take a lot of time. In particular, simulations under SCS models and SS calculation related with the protein stability may take a considerable amount of time especially when performing a large number of simulations or when the protein dataset includes a large number and long sequences. To reduce the execution time, simulations and SS calculation phases can be run in parallel on a Linux environment if the user specifies this in the settings file and works in a machine with many processors or if the user uses a cluster selecting many processors. Note that if simulation settings are unexpected the analysis may fail (see section 4.7).

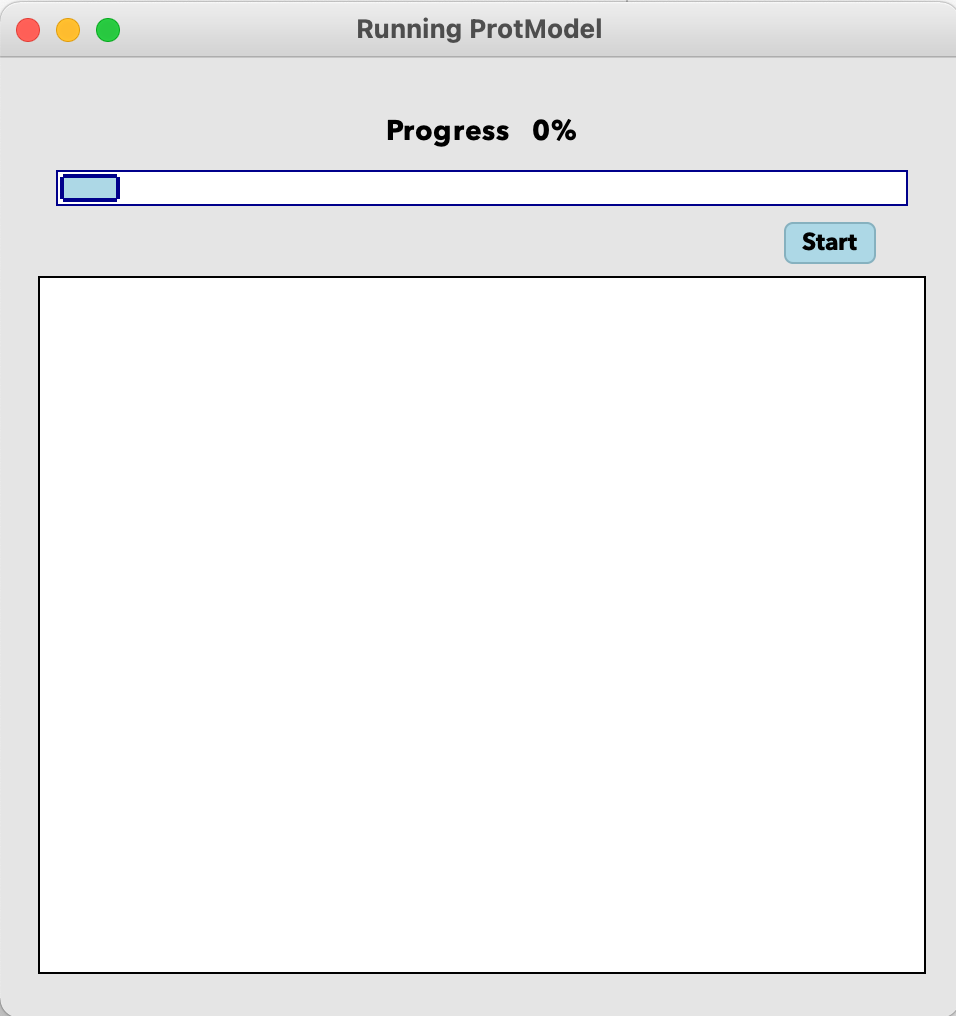
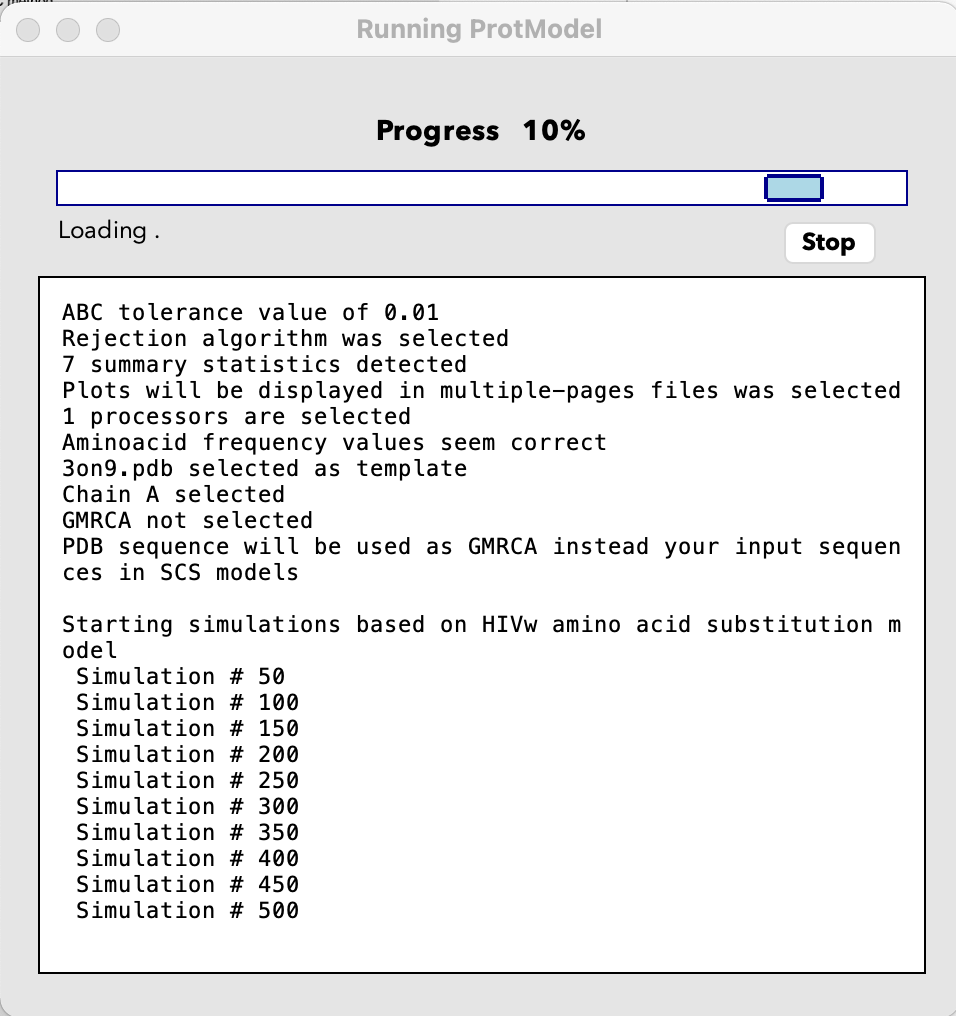
To execute *ProtModel* all the python scripts (.py) must be placed within the mandatory input files (Settings.txt, the multiple alignment of protein sequence, template structure and structures.in). *ProtModel* was developed using python3.9 version and tested under most of python3 versions so is recommended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

* **Execution with GUI:**

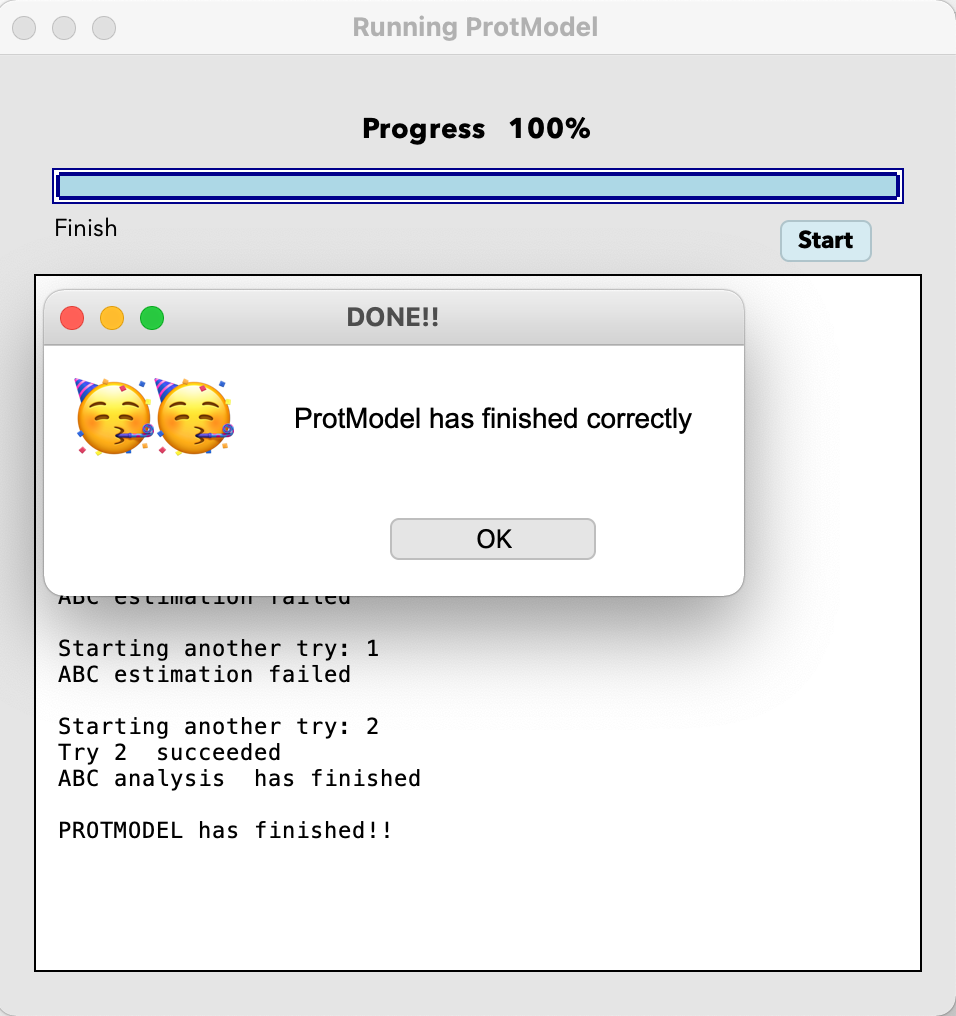
1. To open ProtModel GUI we have two options. The user has to go to the “*GUI”* directory and type “*python.3.9 ProtModel\_GUI.py*” or can go to the “*Executable”* directory and click the executable “*ProtModel\_GUI*”. Remember that GUI “*.py*” or executable don’t have to be in the GUI folder, thry can be moved but always with the corresponding images in the case of the “*.py*” file.

*python3.9 ProtMoldel\_GUI.py*

1. ****Now, the user must click or fill in all the mandatory entries. Once everything is filled, the user must click on the “*Save and* *Continue*” button to start the simulation and estimation phases. *ProtModel* 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, which would may produce incorrect estimations.
2. A new window will appear and to launch the simulation and the estimation phase the user only have to click in the “*Start*” button. Inside this window there are a box which will be display all the *ProtModel* execution information.

******

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

****

* **Execution with terminal:**

1. First, the user has to modify the Settings.txt file. Again, we recommend to not change anything more than the parameters values in the file.
2. Once we have our input files ready, we have to go to the ProtModel main directory and type “*python.3.9 ProtModelGeneral.py*”.

*Test\_ProtModel %* ***ls***

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

***bin***  *LeerSettings.py structures.in*

*ChangeVariablesPE.py Makefile TNF.phy*

*Errores.py ProtModelGeneral.py Variables.py*

*Functions.py Settings.txt*

*Test\_ProtModel %* **python3.9 ProtModelGeneral.py**

1. First, Settings.txt file is read and the main inputs are printed:

Alignment file exists

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, *ProtModel* will stop the execution and print a message error (see section 4.6) as:

Number of processors introduce is higher than your computer number of processors

Please check your computer number of processors

ERROR!! Please check NumberOfProcessors value

1. Next, if the user has selected empirical substitution model, simulations will be 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

1. Later, if the user has selected structural substitution model, simulations will be launched:

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. In the next step ProtModel will calculate the summary statistics of the observed and of the 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, ABC can fail but if we repeat the estimation, it can success. So, ABC estimation will be performed up to 10 times.

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 *ProtModel* has finished:

PROTMODEL has finished!!

* **Execution on Cluster:**

1. First, the user has to modify the Settings.txt file. Again, we recommend to not change anything more than the parameters values in the file. Note that, although *ProtModel* is designed to work with no matter the number of processors, is highly recommended to use a number of simulations divisible by the number of processors.
2. Once we have our input files ready, we have to go to the ProtModel main directory and type “*python3 ProtModel\_Clusterl.py*”.

python3 ProtModel\_Clusterl.py

1. Now, instead of launching the simulations and the summary statistics calculations as the other versions, 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 user’s cluster changing any of the first lines (those which start with “*#*”) or the next line, which load CESGA modules needed for the execution (e.g. python, mpi4py). However, change the other lines, which launch the simulations and the summary statistics calculations, is not recommended. *ProtModel* running information will be the same as in the others versions and will be written in a *“slurm.out”* file.

sbatch launch\_simu.sh

1. In this version, the **ABC estimation will not be performed.** Our cluster didn´t have the R version needed to work with the “*abc library*” so the user must to download the folder and perform the estimation running the “*ABCAnalysis.r*” file. **The user must change the path to corresponding of the directory once downloaded.** See section 4.4. Note that sometimes ABC may fail so, in this case, its highly recommended to re-execute R-script several times.

## Output files

Several output files are generated by *ProtModel* 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.

### ABCOutputs

In this folder is saved the information of the ABC estimations and all the necessary files to run again the ABC estimation. A normal *ProtModel* run will produce:

* The file “*Histogram\_Priors.pdf*” show illustrative histogram of the prior distributions used for the simulations of substitution rates. In addition, a *θ* histogram is also provided.
* In case the user selects all summary statistics, the files “*Results\_SS\_Energy.pdf*” and “*Results\_SS\_AAReplacements.pdf*” will be created. “*Results\_SS\_Energy.pdf*” shows two boxplot graphs. The first one representing the stability mean of simulated proteins of every substitution model studied (ΔG kcal/mol) and the second one their corresponding standard deviation. “*Results\_SS\_AAReplacements.pdf*” shows five boxplot graphs. The first one with the number of segregating sites and the other four with the corresponding mean, standard deviation, skeaness and the kurtosis of the Grantham distances of the simulated proteins. If the user doesn’t select all de summary statistics, only one file called “*Results\_SS.pdf*” is created with the boxplots corresponding with the selected summary statistics. For more information about the summary statistics see section 5.2.
* The file “*Histograms\_SStats.pdf*” shows a histogram of the values of summary statistics from the retained simulations for each summary statistic considered for every substitution model evaluated. Vertical blue lines corresponding to the value of the summary statistic of the target sequences alignment are added. 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 bar plot of every substitution model tested representing the median of the distance between accepted simulations summary statistics (bars) and observed ones (blue line). The P-value is also computed to test the fit to every substitution model.
* The file “*PCA.pdf*” shows plot of the two first principal components of a principal component analysis (PCA) of the SS values of the considered substitution models. The black cross corresponds to a sample of the target sequences alignment and area inside the colours lines represent the retained simulations. Ideally the black cross will be inside only one model area. 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 of the confusion matrix, a specific table layout that allows visualization of the performance of the distinction between models. Note that the name will change depending on the number of ABC iterations selected (i.e. “*Results\_ConfusionMatrix\_10sampSSSimulations.pdf” or “Results\_ConfusionMatrix\_1000sampSSSimulations.pdf”).*
* The file “*Scaterplots\_SStatsVSParams.pdf*” shows a scatter plot of the values of the SS and corresponding values of *θ* prior distribution. Again, a blue line corresponding to the value of the summary statistic of the target sequences alignment is added. Ideally the blue line and points should be as close as possible, evenly distributed across substitution rate per site prior distribution.
* The file “*Results\_text.txt*” shows the results of the all the analysis performed. First, the confusion matrix followed by an ABC method matrix, second, 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.
* A copy of the summary statistics computed from each simulated data are printed in “*SSSimulations.csv*” file.
* A copy of the summary statistics computed from the target multiple alignment of protein sequences are printed in the “*SSRealData.csv*” file.
* A copy of the file “*PSimulations.txt*” shows the values sampled from the substitution rate prior distribution and its corresponding *θ* value.
* The script “*ABCAnalysis.r*” contains all the instructions to the ABC estimation phase and to create all the plots.

### SimulationsOutputs

* The other copy of the summary statistics computed from each simulated data are printed in “*SSSimulations.csv*” file.
* The other copy of the summary statistics computed from the target multiple alignment of protein sequences are printed in the “*SSRealData.csv*” file.
* The other copy of the file “*PSimulations.txt*” shows the values sampled from the substitution rate prior distribution and its corresponding θ value.
* 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.

*ProtModel* executables files created during the framework execution will be places in this directory to allow user to modify code to adapt it to their corresponding clusters.

## Re-analysing data

*ProtModel* allows to choose a few different settings for the ABC estimation without having to repeat the high time-consuming simulation and SS calculation phases. These settings can be changed for a better tuning of the estimations. Among these, the most important to consider are the following:

- ABC iterations. This setting defines the number of simulations to consider in the ABC analysis. By considering different values of this setting, the user may get some insights on how well the entire space of the simulations is being characterized.

- ABC tolerance. This setting defines the number of simulations closest to the target sequences alignment that are retained. Choosing the tolerance is not trivial: it has to be large enough so that to provide a good characterization of the posterior distribution, but on the other hand, it has to be small enough so that the simulations retained are not very far from the target sequences alignment.

- Summary statistics to use. This setting defines which summary statistics to consider when performing ABC estimations. As the number of simulations and the tolerance interval, the choice on the summary statistics is not trivial in ABC. For the estimation of the best-fitting substitution model, we showed that the use of all the summary statistics provided in *ProtModel* works fairly well but, sometimes, the folding stability standard deviation of the real data is too high and far from the simulations value. In that case, we recommend to not consider it either repeating the analysis without considering it or editing the “*ABCAnalysis.r*” script removing the stability standard deviation values of real target and simulations. See section 5.2

To re-analyse the data, no matter the version used, the user must go to the ABCOutputs folder. Then, the user must open the “*ABCAnalysis.r*” file and change the desired ABC parameters values in the first lines.

*#####################################################*

*################### ABC VARIABLES ###################*

*#####################################################*

*ABC\_Method <- "rejection"*

*ABC\_Tolerance <- 0.01*

*ABC\_N\_Iterations <- 100*

*#####################################################*

*#####################################################*

*#####################################################*

If the user needs to change the path, because of the Cluster version or due to other reasons, 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. Every example can be run in every *ProtModel* version but we divided the 8 examples in two folder, 1 rapid examples to run on the command line or using the GUI and other folder with 7 examples to run on cluster:

1. **Example1-TNF\_Monkeypox:** Analysis of a simulated real target alignment [10 sequences, 160 amino acids] of the tumour necrosis factor receptor (TNF) of monkeypox virus using only 1 000 simulations (a few simulations with the aim of only performing a rapid exploration of the framework but producing accurate results) comparing the HIVw (best-fitting empirical substitution model according *ProtTest* (Darriba et al., 2011)) and the two SCS models (Fitness and Neutral) (Table 3). It was run using the terminal version.

If we observed the ABC results, we obtained that the best-fitting substitution modes is Fitness model with a probability of 0.4333, while the Neutral and HIVw models takes a probability of 0.4 and 0.1667 respectively. Looking closer the summary statistics plots, we realize that some values of the simulations’ summary statistics related with the amino acid replacements are far enough for the real data values (Results\_SS\_AAReplacements.pdf, Histograms\_SStats.pdf and Scaterplots\_SStatsVSParams.pdf). So, although we used a realistic substitution rate, we decided to repeat the analysis reducing it (**Example1-TNF-Monkeypox-2**) (Table 3). This time, target and simulations summary statistics values are closer and we obtained that Neutral model is the best-fitting substitution model with a probability of 0.6667, while Fitness and HIVw model had a probability of 0.2333 and 0.1 respectively. This in only an example of the importance of choosing suitable evolutionary parameters values for the simulations.

1. **Example2-Protease\_HIV:** Analysis of a simulated real target alignment [50 sequences, 99 amino acids] of the protease of HIV-1 virus using 10 000 simulations comparing the JTT (best-fitting empirical substitution model according *ProtTest* (Darriba et al., 2011)) and the two SCS models (Fitness and Neutral) (Table 3). It was run on the cluster.
2. **Example3-GAG\_HIV:** Analysis of a simulated real target alignment [27 sequences, 288 amino acids] of the protease of HIV-1 virus using 10 000 simulations comparing the JTT (best-fitting empirical substitution model according *ProtTest* (Darriba et al., 2011)) and the two SCS models (Fitness and Neutral) (Table 3). It was run on the cluster.
3. **Example4-NS1\_Flu:** Analysis of a simulated real target alignment [25 sequences, 202 amino acids] of the NS1 of influenza virus using 10 000 simulations comparing the JTT (best-fitting empirical substitution model according *ProtTest* (Darriba et al., 2011)) and the two SCS models (Fitness and Neutral) (Table 3). It was run on the cluster.
4. **Example5-C30\_COVID:** Analysis of a simulated real target alignment [30 sequences, 299 amino acids] of the C30 endopeptidase of SARS-CoV virus using 10 000 simulations comparing the LG (best-fitting empirical substitution model according *ProtTest* (Darriba et al., 2011)) and the two SCS models (Fitness and Neutral) (Table 3). It was run on the cluster.

If we observed the ABC results, we obtained that the best-fitting substitution modes is LG model with a probability of 1. Looking closer the summary statistics plots, we realize that the summary statistic DGREM\_sd is so far from the real data that it does not appear in the plots (Results\_SS\_Energy.pdf, Histograms\_SStats.pdf and Scaterplots\_SStatsVSParams.pdf). So, we decided to repeat the analysis without considering DGREM\_sd summary statistic (**Example5-C30\_COVID-2**) (Table 3). This time, we obtained that Fitness model is the best-fitting substitution model with a probability of 0.88, while Neutral and LG model had a probability of 0.1067 and 0.0133 respectively. This in only an example where the sd of the folding stability of the simulations is not representing the real target.

1. **Example6-** **Methyltr-2 \_COVID:** Analysis of a simulated real target alignment [28 sequences, 298 amino acids] of the SARS-CoV 2'-O-methyltransferase proein using 10 000 simulations comparing the LG (best-fitting empirical substitution model according *ProtTest* (Darriba et al., 2011)) and the two SCS models (Fitness and Neutral) (Table 3). It was run on the cluster.
2. **Example7-EFC\_Ca:** Analysis of a simulated real target alignment [18 sequences, 263 amino acids] of the Calcium-binding EGF domain using 10 000 simulations comparing the Blosum62 (best-fitting empirical substitution model according *ProtTest* (Darriba et al., 2011)) and the two SCS models (Fitness and Neutral) (Table 3). It was run on the cluster.
3. **Example8-TIR:** Analysis of a simulated real target alignment [23 sequences, 171 amino acids] of the Toll-Interleukin receptor using 10 000 simulations comparing the WAG (best-fitting empirical substitution model according *ProtTest* (Darriba et al., 2011)) and the two SCS models (Fitness and Neutral) (Table 3). It was run on the cluster using all the summary statistics except the stability sd.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Protein family description* | *Number and length of sequences* | *Template* | *Substitution Rate prior* | *Theta prior* | *Summary Statistics* | *Nº Simu x model* | *Emp Subs Model* | *ABC Method* | *ABC estimation* |
| Monkeypox tumour necrosis receptor | 10 sequences  160 amino acid | 3on9 | Uniform  0 7.8125e-04 | Uniform  0 500 | 1 2 3 4 5 6 7 | 1 000 | HIVw | rejection | **Fitness** HIVw Neutral  **0.4333** 0.1667 0.4 |
| Monkeypox tumour necrosis receptor | 10 sequences  160 amino acid | 3on9 | Uniform  0 1.8750e-04 | Uniform  0 120 | 1 2 3 4 5 6 7 | 1 000 | HIVw | rejection | Fitness HIVw **Neutral**  0.2333 0.1 **0.6667** |
| HIV protease | 50 sequences,  99 amino acid | 1tcx | Uniform 0 2.5253e-04 | Uniform  0 100 | 1 2 3 4 5 6 7 | 10 000 | JTT | rejection | **Fitness** JTT Neutral  **0.8533** 0.0667 0.08 |
| HIV Gag polyprotein | 27 sequences 288 amino acid | 1l6n | Uniform 0 1.041667e-04 | Uniform 0 120 | 1 2 3 4 5 6 7 | 10 000 | JTT | rejection | Fitness HIVb **Neutral**  0.4333 0.1067 **0.4600** |
| Influenza NS1 | 25 sequences  202 amino acid | 4OPH | Uniform 0 2.4752e-04 | Uniform  0 200 | 1 2 3 4 5 6 7 | 10 000 | JTT | rejection | Fitness JTT **Neutral**  0.0067 0.22 **0.7733** |
| Coronavirus endopeptidase C30 | 30 sequences  299 amino acid | 1LVO | Uniform 0 4.180602e-04 | Uniform 0 500 | 1 2 3 4 5 6 7 | 10 000 | LG | rejection | Fitness **LG** Neutral  0 **1** 0 |
| Coronavirus endopeptidase C30 | 30 sequences  299 amino acid | 1LVO | Uniform 0 4.180602e-04 | Uniform 0 500 | 1 3 4 5 6 7 | 10 000 | LG | rejection | **Fitness** LG Neutral  **0.88** 0.0133 0.1067 |
| Coronavirus 2'-O-methyltransferase | 28 sequences  298 amino acid | 7c2i | Uniform 0 4.194631e-04 | Uniform 0 500 | 1 2 3 4 5 6 7 | 10 000 | LG | rejection | **Fitness** LG Neutral  **0.6733** 0.1267 0.2 |
| Calcium-binding EGF domain | 18 sequences  263 amino acid | 6pog | Uniform 0 2.118644e-04 | Uniform 0 100 | 1 2 3 4 5 6 7 | 10 000 | Blosum62 | rejection | **Blosum62** Fitness Neutral  **0.9933** 0 0.0067 |
| Toll-Interleukin receptor domain | 23 sequences  171 amino acid | 5ku7 | Uniform 0 7.309942e-04 | Uniform 0 500 | 1 3 4 5 6 7 | 10 000 | WAG | rejection | **Fitness** Neutral WAG  **0.9933** 0.0067 0 |

## Message errors and recommendations

Errors generated from incorrect settings are usually shown on the screen and the program will suggest abort the execution by typing CTRL+C in the command line. However, if the user uses the GUI, incorrect values will be automatically detected. If the user doesn’t want to work with GUI, I recommend copying a provided input file “Settings.txt” from the “examples” folder and edit it with the desired settings.

The input multiple sequence alignment must be presented in *standard phylip sequential* format. Importantly, the amino acid sequences of the input multiple sequence alignment should only include any of the 20 amino acids (one-letter code) or indels (as “-”). Other letters or symbols (i.e., X, $, ., etc) will produce an error displayed on the screen. Next, *ProtModel* check Settings.txt inputs. If any of them is incorrect (i.e., number when string is expected, parameters out of limits) *ProtModel* will stop the execution printing the parameter which is incorrect.

In general, note that the simulation phase can be computational costly with long running times. This is particularly found when performing a large number of simulations, or when the simulated data includes a large number and long sequences. Therefore, obviously, reducing the number of simulations and the size of the input alignment of protein sequences can dramatically reduce computer time. Sometimes reducing the input dataset is not be possible and then one can try to reduce the number of simulations. Indeed, running the simulations in parallel (if the machine has more than one processors) can reduce computer times. During sequences simulations some of them may failed due to intrinsic problems of simulation models, especially when Fitness SCS model is used. However, after last simulations was finished *ProtModel* will check if some of them failed and rerun each failed simulation.

During the ABC procedure the estimation may fail due to an incorrect value of the ABC tolerance or iterations or when mnlogistic or neuralnet methods are selected. If the ABC estimate cannot be carried out completely, no matter the reason, *ProtModel* will launch a big size error.

In this situation this the ABC estimation can be executed again (section 5.4) varying the ABC tolerance, iterations or method in the “*.r*” file. Concerning the ABC method, the rejection approach can be more robust (in terms of being able to work) than the others to analyze real data. In addition to varying this values, one can increase the number of simulations. This could be enough to run successfully the ABC estimation but implies repeat the whole *ProtModel* execution with the new number of simulations.

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 and Methods

*ProtModel* is a framework to the best-fitting substitution model of protein sequence alignments using an ABC procedure. The simulator includes a variety of evolutionary models that may help to better mimic real scenarios. Then, informative summary statistics are calculated from the simulated data. Finally, the computation of the posterior distributions is performed using ABC methods.

## Evolutionary Models and Computer Simulations

The first step of *ProtModel* is based on the simulator *ProteinEvolverProtABC*, which is an adapted version of the program *ProteinEvolver* (<https://github.com/MiguelArenas/proteinevolver>) (Arenas et al., 2013) to perform ABC. *ProteinEvolverProtABC* implements a variety of evolutionary models and generates protein sequence alignments (see below) collected at same or different times (temporal longitudinal sampling or tip dates [see, Navascués et al., 2010]), from a population evolved using the coalescent approach (Hudson, 2002; Kingman, 1982). Protein sequences are evolved under a variety of empirical substitution models of protein evolution 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., 2006), Mtmam (Yang et al., 1998), Mtrev24 (Adachi & Hasegawa, s. f.), RtRev (Dimmic et al., 2002), VT (Müller & Vingron, 2000), WAG (Whelan & Goldman, 2001) and UserEAAM models (Arenas, 2015). Of course, protein sequences can be evolved under SCS models of protein evolution too, 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. Indeed, the program also implements heterogeneity across sites according to a gamma distribution (+G) and proportion of invariable sites (+I) (Yang, 1996).

## Summary Statistics

*ProtModel* includes a total of 7 summary statistics (Table 3). The first two summary statistics are related with the proteins folding stability. Folding stability was measured with another framework called *DeltaGREM* (Arenas et al., 2017) using Gibbs free energy difference between folded and unfolded states (ΔG kcal/mol). In addition, *ProtModel* uses the number of animo 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 3. Summary statistics implemented in *ProtModel* and their identifiers (ID).

|  |  |  |
| --- | --- | --- |
| *Name* | *Description* | *ID* |
| DGREM\_Mean | Mean of alignment folding stability | 1 |
| DGREM\_sd | Standard deviation of alignment folding stability | 2 |
| SegSites | Number of segregation sites | 3 |
| Grantham\_mean\_Position | Mean of the Grantham distance between aa replacements per protein site | 4 |
| Grantham\_sd\_Position | Standard deviation of the Grantham distance between aa replacements per protein site | 5 |
| Grantham\_sk\_Position | Skeaness of the Grantham distance between aa replacements per protein site | 6 |
| Grantham\_ku\_Position | Kurtosis of the Grantham distance between aa replacements per protein site | 7 |

## ABC Methods

Different ABC estimation methods are implemented in *ProtModel*. In the first one, “rejection”, the posterior probability of a given model is approximated by the proportion of accepted simulations given this model. This approximation holds when the different models are a priori equally likely, and the same number of simulations is performed for each model. On the other hand, “mnlogistic”, estimate the posterior probabilities using a multinomial logistic regression using neural networks and, finally, “neuralnet”, use neural networks too to predict the probabilities of models based on the observed statistics. This method can be useful if many summary statistics are used.

## *ProtModel* 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 including SCS models, which successfully yielded accurate inferences of protein evolution overcoming 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).

We performed an evaluation of *ProtModel* considering three substitution models Dayhoff (empirical model) (Dayhoff et al., 1978), Fitness and Neutral (Arenas et al., 2015) (both SCS models), and with a dataset of 27 sequences and the 1TDE structure (Waksman et al., 1994) of the thioredoxin reductase protein family. We used different number of protein simulations (10 000, 50 000 and 100 000) parameterized under a *θ* uniform (0, 500) prior distribution which encompass values that are commonly observed in real data (Lopes et al., 2014) with *ProteinEvolverProtABC*. We applied the three ABC methods (rejection, mnlogistic and neuralnet) and, as in previous studies (Arenas et al., 2020), we made a cross-validation of the ABC methods based on 100 pseudo-observed simulations with a tolerance of 0.005, 0.01 and 0.05 (Csilléry et al., 2012). We simulated 100 multiple sequence alignments under each substitution model considered (Dayhoff, Fitness and Neutral) (test datasets) and for every combination of parameters (3 x 3 x 3 = 27 combinations), we tested if *ProtModel* can correctly estimate the best-fitting substitution model in the test datasets using ABC.

We found that using the rejection ABC method, regardless the number of simulations we obtained similar results while the estimations accuracy decreases while tolerance value increase. However, mnlogistic and neuralnet ABC methods failed when we consider 10 000, 50 000 and 100 000 simulations with any ABC tolerance value, using the rejection method instead of mnlogistic or neuralnet in some of the 100 estimations. These results show that the best-fitting substitution model estimation are generally accurate using the rejection method (Figure 1), being able to distinguish between the empirical and the SCS models but also between SCS models.



Figure 1. Evaluation of the estimation of the best-fitting substitution model. The estimation was performed under the rejection method and shows the probability of selecting the true substitution model within the 100 simulations per model for every ABC tolerance (0.005, 0.01 and 0.05) and number of simulations (10 000, 50 000 and 100 000). Error bars indicate 95% confidence intervals from the mean.

We also evaluate 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 HIV-1 protease data (see section 4.6), performing independent runs with 1, 2, 4, 8 and 12 processors in our computer and with 50, 100, 200, 250 and 500 processors in the cluster. Note that for cluster analysis is highly recommended to use a number of simulations divisible by the number of processors. We obtained that, in bother cases, the number of processors and the computer time don’t follow a linear function. In the computer analysis it was expected, due to each processor doesn’t work independently, they shared memory and even the computer time can be affected by the simultaneous computer use. On the other hand, we expected that computer time and the number of processors during a cluster analysis follows a nearly linear function, but we obtained that they follow the same function as the computer analysis. In this case, processor don’t share memory and they are not affected by any other cluster usage, but we think that the fact that to start another process (i.e. start SS calculation) the previous process has to be finished for all the processors. So, if one processor is working slower, all the process will be affected.

## Assumptions and Limitations

ABC assumes that simulated data can mimic the 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 apply 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. (2006). 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., & 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., Gorostiza, A., Baquero, J. M., Campoy, E., Branco, C., Rangel-Villalobos, H., & González-Martín, A. (2020). The Early Peopling of the Philippines based on mtDNA. *Scientific Reports*, *10*(1), 4901. https://doi.org/10.1038/s41598-020-61793-7

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

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

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

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

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